University of Nottingham
UK | CHINA | MALAYSIA

U-Net based Deep Convolutional Neural Network Models for Liver Segmentation from CT Scan Images

PhD Thesis (Sep 2020)

Mahmoud Abdel Azim Helmy Mahmoud Khattab

Supervisor: Dr. Iman Yi Liao

Table of Contents

Table of figures 5

List of tables 11

List of Terms 14

Acknowledgment 15

Abstract 16

1 Chapter 1 Introduction 18

1.1 Motivation 18

1.2 Research problems 20

1.3 Project background 23

1.4 Research questions 23

1.5 Research objectives 24

1.6 Proposed solutions 25

1.7 Research scope 26

1.8 The structure of the thesis 26

2 Chapter 2 Literature review 28

2.1 Image segmentation 28

2.1.1 Threshold 29

2.1.2 Edge Based Segmentation Method 30

2.1.3 Region Based Segmentation Method 31

2.1.4 Clustering Based Segmentation Method 31

2.1.5 Watershed Based Methods 32

2.1.6 Partial Differential Equation (PDE) Based Segmentation Method 32

2.1.7 Artificial Neural Network Based Segmentation Method 33

2.2 Medical Image segmentation 34

2.2.1 Conventional approaches 36

2.2.2 Deformable Methods 38

2.2.3 Machine Learning models/methods 39

2.3 Deep Learning approaches in Computer Vision and for image segmentation 47

2.3.1 Deep Learning with CNN for Medical Imaging 47
2.3.2 CNN architecture and its variations 51
2.3.3 Loss function variations 58
2.3.4 Regularization techniques 72
2.3.5 Optimization techniques 77
2.3.6 Transfer learning in medical imaging 79
2.4 Liver segmentation 80
2.4.1 General techniques for liver segmentation 80
2.4.2 Machine Learning based methods for liver segmentation 84
2.4.3 Deep learning-based methods for liver segmentation 86
2.5 Summary 90
3 Chapter 3 Research Methodology 94
3.1 Datasets 97
3.2 Hardware/Software framework 99
3.2.1 Hardware 99
3.2.2 Software framework 99
3.3 Data augmentation 100
3.4 Baseline Models 101
3.4.1 U-net 103
3.4.2 Bridge-Net 105
3.5 Evaluation metrics 108
3.6 Main proposed architecture designs 109
3.6.1 Deeper U-Net based models 110
3.6.2 Wider U-Net based models 112
3.6.3 Skip connections 113
3.6.4 Loss functions design 116
3.7 Training/validation process 123
3.8 Testing process 124
4 Chapter 4 Experimental Results and Discussions 126
4.1 Model structure variations 127
4.1.1 Deeper U-Net based models 128
4.1.2 Wider U-Net based models 154

4.1.3 Bridging and Skip-Connections 162
4.2 Loss functions to solve the flipping issues 232
4.2.1 Regression Loss functions 232
4.2.2 Deep- supervision with different loss functions 237
4.2.3 Results Summary 251
4.2.4 Qualitative analysis for flipping issue's proposed solutions 252
4.2.5 Discussion for flipping issue 255
4.2.6 Flipping issue recommendations 256
4.3 Segmentation results validation by Institut Kanser Negara (IKN) 258
4.3.1 Validation guidelines and criteria 259
4.3.2 IKN Validation results 260
5 Chapter 5: Conclusions and Future work 262
6 Appendices 268
6.1 The implemented U-Net model 268
6.2 The implemented 2 bridged U-Net with original connection 270
6.3 Models summary 274
6.3.1 2 Bridge U-Net with Original connections 274
6.4 Analysis of all deeper models based on the number of filters applied at the deepest level 277
6.5 Qualitative analysis for flipping issue's proposed solutions 283
7 References 289

Table of figures

Figure 1-1 Estimated number of new cases of cancers in 2018, worldwide, all ages, both sexes, The chart adapted from "World Health Organization,2020.https://gco.iarc.fr/" [2] 19
Figure 1-2 Estimated number of deaths in 2018 of all cancers worldwide, all ages, both sexes The chart adapted from "World Health Organization,2020.https://gco.iarc.fr/" [2] 19
Figure 2-1 Medical image segmentation techniques 36
Figure 2-2 U-Net model structure for medical image segmentation, ... 52
Figure 2-3 W-Net: A Deep Model for Fully Unsupervised Image Segmentation , The figure adopted from "Xide Xia, Brian Kulis" [139] 56
Figure 2-4 prostate segmentation using 2bridged U-Net 58
Figure 2-5 Semantic taxonomy for loss functions 59
Figure 2-6 Cosh(x) function is the average of e^{X} and e^{--X} 65
Figure 2-7 tanh(x) function is continuous and finite. It ranges from [-1; 1] 65
Figure 2-8 Hausdorff Distance between point sets X and Y [157] 68
Figure 2-9 Training versus testing error with early stopping 75
Figure 2-10 Training and Validation loss 76
Figure 2-11 Perofmance comparasonfor SGD, Adagrad, Adam optemizers 78
Figure 3-1 Methodology modules and processes in each module and the relationships between different models and processes. The highlighted processes represent the research contributions 97
Figure 3-2 Examples of 3Dircadb1 dataset, DICOM image and associated mask shows a lot of variations in the liver shape even for the same patient 98
Figure 3-3 Examples for IKN dataset, the patient position from left to right is face up, face down, right side 45 dgree, right side 90 degree 99
Figure 3-4 Software framework 100
Figure 3-5 Augmentation techniques examples, A- Rotation 180^{o}, B-Horizontal Flip, C- Vertical Flip, D-Rotation 45^{o}, E- Rotation 60^{o}, Rotation 90^{o}, G- Rotation 270^{o} 101
Figure 3-6 U-Net model structure for medical image segmentation. Number of filters (red) from 64, 128, 256, 512, 1024. Feature maps size (black) outside starting with 512*512 and ends to 32*32. 105
Figure 3-7 W net with bridge and skip connections model structure 105
Figure 3-8 Bridged 2U-Net model 108
Figure 3-9 Top-Down approach, example for model starts with 8 filters applied to the first layer 32 filters applied to the deepest layer then increasing the depth of the model by adding extra layer with 200% of the filters each step, ending with 2048 filters applied on the deepest layer of model of 9 levels. 111
Figure 3-10 Bottom-Up approach, example for model starts with 256 filters applied to the first layer 1024 filters applied to the deepest layer then increasing the depth of the model by adding extra layer with 50% of the filters each step to the top of the model, ending with 8 filters applied on the highest layer of model of 8 levels. 112
Figure 3-11 Wide models based on U-Nets, A- two stacked U-Nets, B- Three Stacked U-Nets, C- Four stacked U-Nets 113

Figure 3-12 Different types of bridged connections and skip connections with 2 U-Nets models 114
Figure 3-13 Representation for examples of models developed based on three U-Net models using the different connections style (Original, Modified, and Compound) 115
Figure 3-14 Representation for examples of 3 U-Net models with long connections between U-Nets (1 and 3) 116
Figure 3-15 The flipping issue appeared due to augmentation process, The ground truth (Green line), and the predicted liver (Red line) 117
Figure 3-16 Dice similarity coefficient representation 118
Figure 3-17 Distance between center of mass of two blobs (Centroid distance) 119
Figure 3-18 Deep supervision approach using de-convolution processes 121
Figure 3-19 Deep supervision approach using Multi-resolution masks 122
Figure 3-20 Testing result example, A) original, B) Ground Truth, C) Predicted Mask, D) mapped masks to the original image 124
Figure 4-1 Accuracy of training, validation, testing, and testing using augmented data for all models started with 8 filters at the first level 129
Figure 4-2 Accuracy of training, validation, testing, and testing using augmented data for all models started with 16 filters at the first level 130
Figure 4-3 Accuracy of training, validation, testing, and testing using augmented data for all models started with 32 filters at the first level 131
Figure 4-4 Accuracy of training, validation, testing, and testing using augmented data for all models started with 64 filters at the first level 132
Figure 4-5 Accuracy of training, validation, testing, and testing using augmented data for all models started with 128 filters at the first level 133
Figure 4-6 Accuracy of training, validation, testing, and testing using augmented data for all models started with 256 filters at the first level 134
Figure 4-7 Accuracy of training, validation, testing, and testing using augmented data for all models with 2048 filters at the deepest level 136
Figure 4-8 Accuracy of training, validation, testing, and testing using augmented data for all models with 1024 filters at the deepest level 137
Figure 4-9 Accuracy of training, validation, testing, and testing using augmented data for all models with 512 filters at the deepest level 139
Figure 4-10 Accuracy of training, validation, testing, and testing using augmented data for all models with 256 filters at the deepest level 140
Figure 4-11 Accuracy of training, validation, testing, and testing using augmented data for all models with 128 filters at the deepest level 141
Figure 4-12 Accuracy of training, validation, testing, and testing using augmented data for all models with 64 filters at the deepest level 142
Figure 4-13 Training accuracy for models included in based on the first layer filters approach 145
Figure 4-14 Validation accuracy for models included in based on the first layer filters approach 146
Figure 4-15 Testing (Test_Avg) accuracy for models based on the first layer filters 147
Figure 4-16 Testing using augmented data (Test_Avg_Aug) accuracy for models based on the first layer filters 148

Figure 4-17 Comparing U-Net model with stacked 2, 3, and 4 U-Nets models using image size 256*256 155
Figure 4-18 Comparing U-Net model against stacked 2, 3, 4 U-Nets using image size 128*128 156
Figure 4-19 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right) 157
Figure 4-20 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right) 158
Figure 4-21 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right) 158
Figure 4-22 Different types of bridged connections and skip connections with 2 U-Nets models 163
Figure 4-23 Original, Modified, Compound models with filters structure 32-512 164
Figure 4-24 Original, Modified, Compound models with filters structure 64-1024 165
Figure 4-25 Original Model with filters' numbers 32-512 and 64-1024 166
Figure 4-26 Modified Model with filters' numbers 32-512 and 64-1024 166
Figure 4-27 Compound model filters numbers 32-512 and 64-1024 167
Figure 4-28 Representation for examples of models developed based on three U-Net models using the different connections style (Original, Modified, and Compound) 170
Figure 4-29 Representation for examples of 3 U-Net models with long connections between U-Nets (1 and 3) 170
Figure 4-30 Training accuracy for all models based on 3U-Net models 173
Figure 4-31 Validation accuracy for all models based on 3U-Net models 173
Figure 4-32 Testing (Test_Avg) accuracy for all models based on 3U-Net models 174
Figure 4-33 Testing using augmented data (Test_Avg_Aug) accuracy for all models based on 3U-Net models 174
Figure 4-34 Training accuracy for all models with Long connection 176
Figure 4-35 Validation accuracy for all models with Long connection 177
Figure 4-36 testing (Test_Avg) accuracy for all models with Long connection 177
Figure 4-37 Testing using augmented data (Test_Avg_Aug) accuracy for all models with Long connection 178
Figure 4-38 Training accuracies for 3U-Net models versus 3U-Net models with long connection 180
Figure 4-39 Validation accuracies for 3U-Net models versus 3U-Net models with long connection 180
Figure 4-40 Testing (Test_Avg) accuracies for 3U-Net models versus 3U-Net models with long connection 181
Figure 4-41 Testing using augmented data (Test_Avg_Aug) accuracies for 3U-Net models versus 3U-Net models with long connection 181
Figure 4-42 5 Models with long connection recorded better accuracy against the same models without long connections (Original-Compound, Compound-Original, Modified-Compound, Compound-Modified, Modified-Modified) 184
Figure 4-43 4 Models show better accuracy against the same models with long connections (Compound-Compound, Original-Original, Original-Modified, Modified-Original) 185
Figure 4-44 2U-Net models versus 3U-Net models 187

Figure 4-45 Training and Validation accuracies for 2U-Net models versus 3U-Net models with long connections 188
Figure 4-46 Testing and Testing using augmented data (Test_Avg, Test_Avg_Aug) accuracies for 2U-Net models versus 3U-Net models with long connections 189
Figure 4-47 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Original connections for 2U-Net and 3U-Net and 3U-Net with long connection 191
Figure 4-48 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Modified connections for 2U-Net and 3U-Net and 3U-Net with long connection 192
Figure 4-49 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Compound connections for 2U-Net and 3U-Net and 3U-Net with long connection 192
Figure 4-50 Training, Validation, Test_Avg, Test_Avg_Aug accuracies for 3U-Net models that used only one connection type between each two U-Nets 194
Figure 4-51 Training, Validation, Test_Avg, Test_Avg_Aug accuracies for 3U-Net models with long connection that used only one connection type between each two U-Nets 195
Figure 4-52 Training accuracy for all models of three U-Nets with Original connections between the first two U-Nets and variant connections between the second and third U-Nets 197
Figure 4-53 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Original connections between the first two U-Nets and variant connections between the second and third U-Nets 197
Figure 4-54 Training accuracies for all models of three U-Nets with Modified connections between the first two U-Nets and variant connections between the second and third U-Nets 199
Figure 4-55 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Modified connections between the first two U-Nets and variant connections between the second and third U-Nets 200
Figure 4-56 Training accuracies for all models of three U-Nets with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets 202
Figure 4-57 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets 202
Figure 4-58 Training accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets 206
Figure 4-59 Validation accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets 206
Figure 4-60 Testing (Test_Avg) accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Net 207
Figure 4-61 Testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Net 207

Figure 4-62 Training accuracy for all models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets 210
Figure 4-63 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets 210
Figure 4-64 Training accuracy for all models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets 212
Figure 4-65 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets 213
Figure 4-66 Training accuracy for all models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 215
Figure 4-67 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 215
Figure 4-68 Training accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 219
Figure 4-69 Validation accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 220
Figure 4-70 Testing (Test_Avg) accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 220
Figure 4-71 Testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets 221
Figure 4-72 Training accuracy for all models 223
Figure 4-73 Validation accuracy for all models 224
Figure 4-74 Testing accuracy for all models 225
Figure 4-75 Testing with augmented data accuracy for all models 226
Figure 4-76 The results for U-Net, Bridge U-Net and 2U-Nets with Compound connection. Compound model have the best performance for (P20) and enhanced the flipping issue for P5 234
Figure 4-77 Using (DSC : Centroid) weights with 5:1recorded accuracy better than 3:1 for the first half of samples while for the second half the accuracy decreased even worse than using DSC loss function ... 237
Figure 4-78 Deep-Supervision using De-Convolutional layers with DSC loss function achieved accuracy better than the Compound model, and better than the centroid function with 3:1 ratio for the first half of samples for (P5) but the Centroid function is better for the second half of the samples. 240
Figure 4-79 sample output for Compound model with Deep-Supervision approach using De-Convolutional layers and loss function is sum of weighted loss of the four outputs using DSC function. Predicted mask at left, mapped ground truth (green) and predicted mask (Red) at right 241

Figure 4-80 Deep-Supervision using Multi-Resolution masks with output weights (10:2:2:10) using the loss function DSC recorded the best output over Deep-Supervision using De-Convolutional layers, and better than compound model with Centroid function with weight (3:1) in the first half while the accuracy become lower during the second half of the samples. 243
Figure 4-81 Example for Deep-Supervision using Multi-Resolution masks. Predicted mask (Left), mapping the predicted mask (red) and Ground truth (Green) on the original image..... 244
Figure 4-82 Deep-Supervision with De-Convolutional layers using weighted outputs (10:2:2:10) using weighted loss function for (DSC: Centroid) equal (3:1). The model recorded the best accuracy for the first half samples over all the previous loss functions while for the second half it recorded the worst accuracy over all the previous proposed loss functions 246
Figure 4-83 All models using Deep-Supervision with de-convolution and Multi-Resolution masks with weighted outputs (10:2:2:10) and loss functions are DSC and weighted loss for (DSC:Centroid) functions with weights (3:1), (2:1) and (5:1) 250
Figure 4-84 Example of models' output comparison 254
Figure 4-85 Calculating the weights and total weights for each model over all the images..... 254
Figure 4-86 calculating the final ranking for the models based on the total weights 254
Figure 4-87 Examples of IKN dataset, images with treatment needle appeared as a light beam in addition to black line (left top and left bottom), patient lies down on the left side (right) 258
Figure 4-88 Example of tested data for IKN validation, original image (right), generated liver mask (middle), mask mapped on the original image in red (left). 260
Figure 6-1 Training accuracy for all models based on the deepest layer filters 278
Figure 6-2 Validation accuracy for all models based on the deepest layer filters..... 279
Figure 6-3 Testing accuracy for all models based on the deepest layer filters..... 280
Figure 6-4 Testing using augmented data (Test_Avg_Aug) accuracy for all models based on the deepest layer filters 281

List of tables

Table 2-1 Image segmentation methods advantages and limitations..... 47
Table 2-2 Semantic taxonomy for the loss functions been used in CNN [141] 59
Table 2-3 Loss functions and description..... 71
Table 3-1 Model parameters and training process settings 124
Table 4-1 Models start with layer of 8 filters and different depth (8-32, 8-64, 8-128, 8-256, 8-512, 8-1024, 8-2048) using image size 256*256 , Highest accuracy (Green) lowest accuracy (Red) 128
Table 4-2 Models start with layer of 16 filters and different depth (16-64, 16-128, 16-256, 16-512, 16-1024, 16-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red) 130
Table 4-3 Models start with layer of 32 filters and different depth (32-128, 32-256, 32-512, 32-1024, 32-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red) 131
Table 4-4 Models start with layer of 64 filters and different depth (64-256, 64-512, 64-1024, 64-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red) 132
Table 4-5 Models start with layer of 128 filters and different depth (128-512, 128-1024, 128-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 133
Table 4-6 Models start with layer of 256 filters and different depth (256-512, 256-1024, 256-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 134
Table 4-7 All models with 2048 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 135
Table 4-8 All models with 1024 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 137
Table 4-9 All models with 512 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 138
Table 4-10 All models with 256 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 140
Table 4-11 All models with 128 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 141
Table 4-12 All models with 64 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 142
Table 4-13 All models with same start and different number of levels compared with all possible starts. Within each group,Highest accuracy (Green) lowest accuracy (Red)..... 144
Table 4-14 Maximum accuracy achieved by models based on the number of the first layer filters for Training, Validation, Testing, and testing using augmented data factors (Top-Down approach). Highest accuracy (Green) within each group..... 151
Table 4-15 Maximum accuracy achieved by models based n the number of the deepest layer's filters for Training, Validation, Testing, and testing using augmented data factors(Bottom-up approach). Highest accuracy (Green) within each group..... 151
Table 4-16 U-net, 2, 3 and 4 stacked U-Net with image size 256*256. Highest accuracy (Green) lowest accuracy (Red)..... 155

Table 4-17 U-net, 2, 3 and 4 stacked U-Net with image size128*128. Highest accuracy (Green) lowest accuracy (Red)..... 156
Table 4-18 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256. Highest accuracy (Green) lowest accuracy (Red) within each group 157
Table 4-19 Training , validation, and testing accuracy for original, Modified, and Compound models with filters structure 32-512 and 64-1024. Highest accuracy (Green) within each group 163
Table 4-20 Training, Validation, and testing for 3U-Net based model with and without long bridge connections, Highest accuracy (Green) lowest accuracy (Red) within each group..... 171
Table 4-21 Models based on 3U-Net, Highest accuracy (Green) lowest accuracy (Red)..... 172
Table 4-22 Models based on 3U-Net with long connections, Highest accuracy (Green) lowest accuracy (Red)..... 176
Table 4-23 3U-Net based models with versus without long connections, Highest accuracy (Green) lowest accuracy (Red) within each group..... 179
Table 4-24 2U-Net models versus 3U-Net models, Highest accuracy (Green) lowest accuracy (Red) within each group 186
Table 4-25 2U-Net models versus 3U-Net models with long connections, Highest accuracy (Green) lowest accuracy (Red) within each group..... 188
Table 4-26 compare U-Net with the respective 3U-Net models and with long connection models, Highest accuracy (Green) lowest accuracy (Red) within each group..... 191
Table 4-27 Accuracy for 2U-Net models, 3U-Net models, and 3U-Net models with long connections with same connection between all 3 U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group..... 194
Table 4-28 All models of three U-Nets with Original connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 196
Table 4-29 All models of three U-Nets with Modified connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 199
Table 4-30 All models of three U-Nets with Compound connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 201
Table 4-31 All models of three U-Nets with same connections between the first two U-Nets and variant connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group 205
Table 4-32 All models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 209
Table 4-33 All models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 212

Table 4-34 All models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)..... 214
Table 4-35 All models of three U-Nets with the same connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group..... 219
Table 4-36 All models of 2U-Nets, 3U-Nets, and 3U-Nets with long connection. Max accuracy (Green), Min accuracy (Red) 222
Table 4-37 number of models versus the accuracy ranges for Validation, Test_Avg, Test_Avg_Aug for all models (2U-Net, 3U-Nets, 3U-Nets with long connection) 228
Table 4-38 Training, Validation, Testing (Test_Avg) and testing using augmented data (Test_Avg_Aug) accuracies for all 2U-Net models with Dice Similarity coefficient (DSC) loss function..... 233
Table 4-39 Different weights for DSC and Centroid distance function showed the best ratio is (DSC:Centroid) is (3:1), then 2:1 then 5:1..... 236
Table 4-40 Deep-Supervision approach using De-Convolutional layers with DSC loss function showed that the best performance recorded for the weight combination (10:2:2:10 for the four outputs of the model. 239
Table 4-41 Deep-Supervision using Multi-Resolution masks showed the best accuracy recorded for the output weights (10:2:2:10) 242
Table 4-42 Deep-Supervision with De-Convolutional layers using weighted outputs (10:2:2:10) using weighted loss function for (DSC:Centroid) equal (3:1) , Highest accuracy (Green) lowest accuracy (Red) within each group 245
Table 4-43 Deep-Supervision using Multi-Resolution masks with weighted outputs (10:2:2:10) and loss function is weighted loss for (DSC:Centroid) functions with weights (3:1), Highest accuracy (Green) ... 247
Table 4-44 The accuracy for all loss functions used to solve the flipping issue with different approachs and multiple ouput weights and DSC:Centroid weights, highest accuracy (Green)..... 252
Table 4-45 statistical results based on IKN validation 261
Table 6-1 All models with same start and different number of levels compared with all possible starts 277

List of Terms

Terms	Meaning
CT	Computed tomography
DSC	Dice Similarity Coefficient
Centroid	Distance between Centers of mass
NN	Neural Network
CNN	Convolutional Neural Network
ASM	Active Shape Model
AAM	Active Appearance Model
FCN	Fully Convolutional Network
CONV	Convolutional layer
ELU	Exponential Linear Unit
ReLU	Rectified Linear Units
BN	Batch Normalization
DALY	The Disability-Adjusted Life Year (DALY) is a metric that captures the total burden of disease – both from years of life lost due to premature death and from years lived with the disease. One DALY equals one lost year of healthy life.

Acknowledgment

First and foremost I am extremely grateful to my supervisor, Dr. Iman Yi Leao for her invaluable advice, continuous support, and patience during my PhD study. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank my supervisors Dr. Ooi Ean Hin and Dr. Chong Siang Yew for all their help and advice with my PhD. I would like to express gratitude to Dr. TOMAS MAUL for his treasured support which was really influential in shaping my experiment methods and critiquing my results.

My gratitude extends to Ministry of Science, Technology and innovation (MOSTI) for the funding opportunities to undertake my studies; I also thank Dr. Dr. Shalini A/P Rajandran Nair and Dr. Faizal Ali from National Cancer Institute (Institut Kanser Negara (IKN), Malaysia) for all their support.

Finally, I would like to express my gratitude to my family, colleagues and my best friends Ms. Gloria Gao Xiuqing. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Abstract

Liver segmentation is a critical task for diagnosis, treatment and follow-up processes of liver cancer. Computed Tomography (CT) scans are the common medical image modality for the segmentation task. Liver segmentation is considered a very hard task for many reasons. Medical images are limited for researchers. Liver shape is changing based on the patient position during the CT scan process, and varies from patient to another based on the health conditions. Liver and other organs, for example heart, stomach, and pancreas, share similar gray scale range in CT images. Liver treatment using surgery operations is very critical because liver contains significant amount of blood and the position of liver is very close to critical organs like heart, lungs, stomach, and crucial blood veins. Therefore the accuracy of segmentation is critical to define liver and tumors shape and position especially when the treatment surgery conducted using radio frequency heating or cryoablation needles.

In the literature, convolutional neural networks (CNN) have achieved very high accuracy on liver segmentation and the U-Net model is considered the state-of-the-art for the medical image segmentation task. Many researchers have developed CNN models based on U-Net and stacked U-Nets with/without bridged connections. However, CNN models need significant number of labeled samples for training and validation which is not commonly available in the case of liver CT images. The process of generating manual annotated masks for the training samples are time consuming and need involvement of expert clinical doctors. Data augmentation has thus been widely used in boosting the sample size for model training.

Using rotation with steps of 15o and horizontal and vertical flipping as augmentation techniques, the lack of dataset and training samples issue is solved. The choice of rotation and flipping because in the real life situations, most of the CT scans recorded while the

while patient lies on face down or with 45o, 60o,90o on right side according to the location of the tumor. Nonetheless, such process has brought up a new issue for liver segmentation. For example, due to the augmentation operations of rotation and flipping, the trained model detected part of the heart as a liver when it is on the wrong side of the body.

The first part of this research conducted an extensive experimental study of U-Net based model in terms of deeper and wider, and variant bridging and skip-connections in order to give recommendation for using U-Net based models. Top-down and bottom-up approaches were used to construct variations of deeper models, whilst two, three, and four stacked U-Nets were applied to construct the wider U-Net models. The variation of the skip connections between two and three U-Nets are the key factors in the study. The proposed model used 2 bridged U-Nets with three extra skip connections between the U-Nets to overcome the flipping issue. A new loss function based on minimizing the distance between the center of mass between the predicted blobs has also enhanced the liver segmentation accuracy. Finally, the deep-supervision concept was integrated with the new loss functions where the total loss was calculated as the sum of weighted loss functions over each weighted deeply supervision. It has achieved a segmentation accuracy of up to 90%.

The proposed model of 2 bridged U-Nets with compound skip-connections and specific number of levels, layers, filters, and image size has increased the accuracy of liver segmentation to ~90% whereas the original U-Net and bridged nets have recorded a segmentation accuracy of ~85%. Although applying extra deeply supervised layers and weighted compound of dice coefficient and centroid loss functions solved the flipping issue with ~93%, there is still a room for improving the accuracy by applying some image enhancement as pre-processing stage.

Chapter 1 Introduction

1.1 Motivation

Cancers recorded in 2017 as the second reason of death all over the world whereby 9.6 million people are estimated to have died from the various forms of cancer. Liver cancer was ranked the 4th among all cancers with a record of 819,435 cases with the rate ~13 per 100,000. According to the disease burden rates, Disability-Adjusted Life Years (DALY) rates in 2017 where one DAILY equal one year loss of healthy life, Liver cancer ranked the second of all cancers that affects the healthy life. [1]

In 2018, the number of liver cancer cases was ranked the 6th ranked between different cancers by 4.7% and 8,141,080 of total 18,078,957 Figure 1-1, while it became the 4th cause of death among all cancers with 626,679 of 9,555,027 , 6.6% Figure 1-2 [2]. According to 'Global Cancer Observatory, International Agency for Research on Cancer, World Health Organization" [2] the number of deaths because of cancers would be increased to reach 13 million in 2030 and 17 million in 2040.

In the United States Since 1980, the number of liver cancer cases has tripled. Between 2007 and 2016, the number of people diagnosed with the disease increased by approximately 2% annually. Men are about 3 times more likely than women to be diagnosed with the disease. When compared with the United States, liver cancer is much more common in Africa and Southeast Asia. In some countries, it is the most common cancer type. [3], [4]

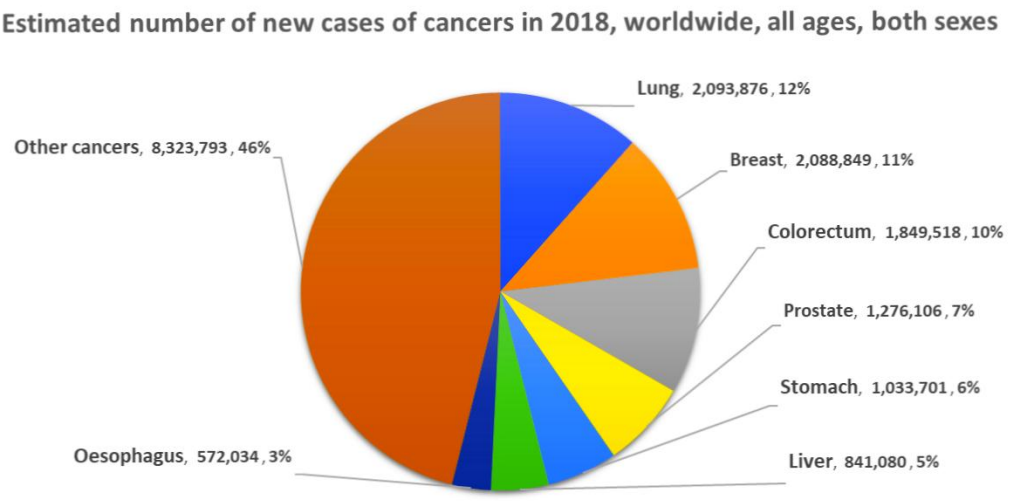


Figure 1-1 Estimated number of new cases of cancers in 2018, worldwide, all ages, both sexes, The chart adapted from "World Health Organization,2020.https://gco.iarc.fr/" [2]

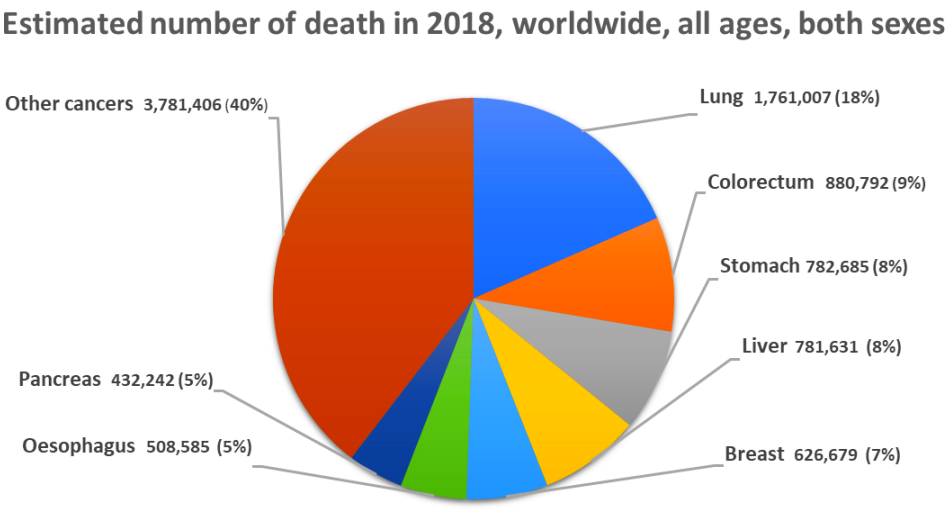


Figure 1-2 Estimated number of deaths in 2018 of all cancers worldwide, all ages, both sexes The chart adapted from "World Health Organization,2020.https://gco.iarc.fr/" [2]

1.2 Research problems

Medical image analysis is used to help clinical doctors to diagnose, planning, surgery, treatment, and follow up. Medical image segmentation aims to extract the organ of interest and display it separately. Liver segmentation aims to automatically identify the liver shape, location, and boundaries using one of the medical image modality, e.g. CT scans, MRI, Ultrasound, etc. The output of the segmentation is a mask for liver in white and the remaining image in black. Liver segmentation is an essential step in the processes of diagnoses, treatment, and follow-up for liver diseases including cancers. Segmenting the liver and tumors gives clinical doctors a clear view of the location, shape, and size of the liver and tumors, these details are needed to decide the method and dose of treatment.

Liver segmentation introduced as a challenging task in Medical Image Computing and Computer-Assisted Intervention conference (MICCAI 2007). A significant number of surveys concluded the approaches of liver segmentation into **1) Gray level based** (region growing, active contour, Graph cut, threshold based, clustering based). **2) Statistical model**s **Active Shape Model (ASM), Active Appearance Model (AAM)**. **3) Texture based** (Machine learning, Pattern recognition). **4) Other methods** (deformable model based methods, Probabilistic atlas based methods , level set based methods) [5]–[11].

Several methods based on Deep learning approaches, including Convolutional Neural Network (CNN) have been reported in the literature, obtaining the best recently published automatic liver segmentation methods, as good performances as **Dice Similarity Coefficient (DSC)** of 0.96 [12], [13]. U-Net model had been built upon the elegant architecture of FCN. U-Net benefits from a superior design of skip connections between different stages of the network. The model consists of two paths with the same number of levels. The contracting path that apply convolutional filters on the input image to catch the image features and down-sampling to reduce the image size after each layer, while the expansion path concatenate the up-sampled feature maps from the previous layer with the

feature maps from the convolutional layer at the same level on the contracting path then apply de-convolution to increase the size of the feature maps with each level to end up with the same size as the original image. For U-Net model proposed in 2015 and achieved 92% accuracy for 2D segmentation of neuronal structure in electronic Microscopy (EM) images stack challenge by International Symposium on Biomedical Imaging (ISBI), U-Net considered as a state-of-the-art for medical image segmentation [14]. Many research papers in the literature proposed models based on U-Net structure with /without modifications .e.g. Stacked 2U-Nets with conditional random field (CRF) approach as a post-processing step [15], integrating Graph cut with 2U-Net stacked, 2D bridged U-Net for prostate segmentation that connect two U-Nets using two bridge connections. The first bridge concatenates the output feature maps for each level from the expansion path of the first U-Net with the inputs of the same level on the contracting path of the second U-Net, while the second connection concatenates the output feature maps from each level of the contracting path of the first U-Net to the inputs of the same level on the expansion path of the second U-Net. [16] .etc.

Although U-Net considered as state-of-the-art for medical image segmentation, the increasing in the number of models designed based on U-Net may cause confusion in terms of what is the best depth for U-Net? Is it better to stack more than one U-Net together? What is the best number of filters applied to each convolutional layer? Are there any recommendations for connecting two or more U-Nets to get better accuracy? In order to answer these questions and provide some recommendations for using U-Net in medical image segmentation, the research contains part for the results of the empirical study conducted for that reason.

Liver segmentation is a very hard task for many reasons. The segmentation algorithm will be affected by different image modalities sources, e.g. MRI, CT, Ultrasound,

X-Ray, and endoscopy. Even with the same modality, using different machines, machine brands, and the scanning setting may result in different image characteristics.

Most medical image devices generate only gray-scale images where internal organs share or have similar intensity (e.g. liver, kidney, stomach, and heart). Some organs are very small in size in some layers of the scans. For example, some CT slices show the lowest parts of the liver as just a very small point.

The shape and size of liver could change from patient to another patient. Even for the same patient, liver shape, size, and position vary according to the patient position during the scanning process where the patient may lie as face up, face-down, or on one side.

The number of datasets available for public research is quite small because of the personal data confidentiality, and generating manually annotated masks for liver is a time consuming task that needs clinical experts. The manually annotated masks serve as a ground truth for model training, validation and accuracy testing.

A new issue appeared during exploring different models based on U-Net we will refer to it as "Flipping Issue". In order to increase the number of training samples, flipping vertical and horizontal and rotation with 15^{o} angel step implemented as augmentation techniques that will cover the variations in real life patient position during the scanning process, e.g. facing down, facing up, lying on the right or left side with some angels (30^{o}, 45^{o}, 60^{o}, 90^{o}) based on the tumor location during and after the treatment process. The tested models including U-Net and bridged U-Net had segmented two parts on both sides of the image, while the ground truth has only one part. The second segmented part is actually part of the heart not liver. The problem shows that some features related to the position of the liver hadn't been captured by the model due to using the augmented data during the model training. This research contains a section of experiments to solve this issue.

The segmentation accuracy is still an open challenge in medical image segmentation especially for the liver. In addition to, there is no single approach can segment multiple organs, or work with different modalities. Even with only liver and same modality for CT scans, models' accuracy will vary with different datasets.

1.3 Project background

The research is part of a project sponsored by **eSienceFund** from **M**inistry of **S**cience, **T**echnology and **I**nnovation (**MOSTI**) in collaboration with Monash University and National Cancer Institute, Malaysia **(IKN)**. The project objective is to segment the liver and tumors from CT images and study the effect of treatment using Radio Frequency and Cryoablation and the needed dose without affecting the liver healthy cells and the blood veins. The prototype that simulates the treatment process and the shortest and best position for radio frequency needles insertion to avoid the critical organs surrounding the liver and calculate the dose and effect on the healthy liver cells represents the novel contribution of the project. The accuracy of the liver segmentation is crucial for the treatment process in terms of position, location, shape, and size.

1.4 Research questions

The research questions aim to enhance the accuracy of liver segmentation in terms of modifying U-Net model as the state-of-the-art of medical image segmentation and handling the shortage of medical images that needed to train U-Net models. By answering the following questions there will be some recommendations for extending U-Net model and explaining for some limitations.

1- Can augmentation solve the issue of limited number of available samples of medical images and masks?
2- When trying to use U-Net model, does it give better performance if more layers added (Meaning deeper U-Net)?
3- Will making U-Net wider by stacking more than one U-Net together enhance the accuracy?
4- Which image size is preferred when using U-Net, deeper U-Net, and stacked U-Net? 512*512, 256*256, or 128*128?
5- Will changing the filters' number for the Bridged U-Net enhance the accuracy?
6- Is there a better connection or skip connections than the original 2 bridged U-Nets?
7- Does extending 2 bridged U-Net to be 3 U-Net make better performance?
8- Which type of connections would be better for 3 bridged U-Net?
9- What are the recommendations of using U-Net for liver segmentation?

1.5 Research objectives

The research has two main objectives. The first objective is to investigate different models structures that inherit the main U-Net structure with variant of modifications e.g. depth, width, number of filters , and skip connections .etc. to introduce recommendations for using models based on U-Net model.

The second objective is to overcome the flipping issue that appeared as a result of applying the rotation and flipping as an example of augmentation techniques. All the tested model including U-Net, 2Bridged U-Net, modified U-Net and the compound 2 U-Nets showed the flipping issue. The flipping issue shows that, the model segment two parts as liver on the right and left while the ground truth only have one part to the left. The problem shows that the wider image context is not captured in the model. The flipping issue illustrated that

the model could not capture some geospatial features of the liver because the training augmented data showed the liver in different locations of the image due to the rotation.

1.6 Proposed solutions

In order to achieve the first objective, an extensive empirical study has been conducted to answer each of the research questions. Top-down and bottom-up approaches are proposed to generate the models to study the performance of the deeper U-Net. Top-Down approach starts with a U-Net model consists of 3 levels with a number of filters at each level, e.g. (8, 16, 32). Then adding a level at the bottom most layer will generate the next deeper mode, e.g. (8, 16, 32, 64). The process of adding a new level at the bottom will continue until the deepest applicable level reached, according to the computational resource limitation, the whole process will be repeated with different U-Net that has a number of filters applied to the first layer different than 8.e.g. (16, 32, 64, 128, 256). Bottom-Up approach is similar to the Top-Down approach, but the bottom layer is the fixed layer and the next deeper model will be generated by adding the layer over the topmost layer, e.g. starting with 4 levels U-Net with a number of filters (256, 512, 1024, 2048) then add a new level on top to generate the deeper model with 5 levels and number of filters (128, 256, 512, 1024, 2048), the process will keep repeating with different layer at the deepest layer, .e.g. (32, 64, 128, 256, 512, 1024). Rotation and flipping techniques applied to overcome the lack of datasets need to train the CNN models. Stacking two, three, and four U-Nets are proposed to test the accuracy of wider U-Nets. Proposed models based on 2 and 3 bridged U-Nets with modified and compound skip and bridge connections are used as a new model for liver segmentation, which has achieved better segmentation accuracy than the state-of-the-art.

Regarding the second objective, a new model based on the bridged U-Net is proposed with deep-supervision approach integrated in the model. In addition to that, a new proposed weighted loss function was applied to minimize the total loss of the model.

The new model based on 2 bridged U-Net with modified and extended skip connections and extra bridged connections recorded better accuracy over the original U-Net and bridged net. Applying the new approach of deep-supervision to the model reduced the flipping issue while integrating the new proposed weighted loss functions within the model with deep-supervision solved ~93% of the flipping issue.

1.7 Research scope

The research only focused on CT scans as the main medical images and not explored any other modality, e.g. X-Ray or MRI because CT scans are commonly used for Liver tumors diagnosis and treatment process. The research focus on Liver segmentation and no other organs had been investigated. Other organs, for example, kidney, and blood vessels are not included because of the shortage of datasets, and significant difference between liver and other organs in features and texture and shape. Liver tumors couldn't be investigated because of time limitation.

1.8 The structure of the thesis

The rest of the thesis consists of 4 more chapters. **Chapter 2 (Literature Review)** contains the related work regarding image segmentation in general and specifically medial image segmentation focusing on the techniques, approaches, algorithms for liver segmentation. The research methodology will be detailed explained in **chapter 3** with full details of the dataset and materials, software and hardware, the stat-of-the-art-models used in the research, the new models and techniques and the related experiments, the

contribution of loss functions and how it work to achieve the research objectives. **Chapter 4** includes all the results that recorded from each experiment compared with the rest of results, discussion of the results indicators and the effect of model modifications and loss functions on the models' performance in terms of accuracy. Finally, the conclusion of the research and the relation to the objectives will be stated in **chapter 5**, in addition to the future research work and vision.

Chapter 2
Literature review

This chapter consists of 4 main sections. **First section** is an overview of image segmentation and a brief introduction for the most common used techniques. **Section 2** contains a review for medical image segmentation and the general used methods for segmentation. **Section 3** illustrates in details the common deep learning approaches that used in medical image segmentation. **Section 4** will focus on liver segmentation and the related models and approaches.

2.1 Image segmentation

Segmentation is defined as the process of partitioning an image into a set of non-overlapping regions whose union is the entire image. These regions should ideally correspond to objects and their meaningful parts, and background. The level to which the subdivision is carried depends on the problem being solved. The segmentation should stop when the objects of interest in an application have been isolated. Image segmentation algorithms generally are based on one of two basic proper ties of intensity values: discontinuity and similarity. In the **discontinuity** category, the approach is to partition an image based on changes in intensity, such as edge detection. The principal approaches in the **Similarity** category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category[17][18].

The popular techniques used for image segmentation are: thresholding method, edge detection based techniques, region based techniques, clustering based techniques, watershed based techniques, partial differential equation based and artificial neural network based techniques etc. These all techniques are different from each other with respect to the method used by these for segmentation.[19]

2.1.1 Threshold

A thresholding procedure attempts to determine an intensity value, called the threshold, which separates the desired classes. The segmentation is then achieved by grouping all pixels with intensity greater than the threshold into one class, and all other pixels into another class[20]. The output of the thresholding operation is a binary image whose gray level of 0 (black) will indicate a pixel belonging to the object of interest and a gray level of 1 (white) will indicate the background [21]. There are different types of thresholding based on the threshold value and how it has been calculated such as single, multiple, global, local, and Otsu thresholding.

While **single thresholding** depends on a single intensity value and transforms input image to a binary image by grouping the pixels with intensities, higher than the threshold into one class, and the other pixels into another class. [22], [23], **Multi-thresholding** use more than one thresholding point [24]. The output image, resulting from multi-thresholding, is no longer binary, but consists of a limited number of grey levels based on the number of thresholding values. **Global thresholding** is The simplest and fastest method is called global thresholding, where one threshold value is used for the entire image[22], The threshold can be fixed through all the image[25], while **Local OR Adaptive Thresholding** is used when the brightness of the image varies from part to another [26], [27].In adaptive thresholding, a criterion function is devised that yields some measure of separation between regions. A criterion function is calculated for each intensity and that which maximizes this

function is chosen as the threshold [21] and the image can be divided into small pieces, and apply thresholding to each piece individually [22]. While the advantage of thresholding method is no need of previous information, simplest method, its disadvantage is highly dependent on peaks, spatial details are not considered.

2.1.2 Edge Based Segmentation Method

The edge based segmentation methods are based on the rapid change of intensity value in an image because a single intensity value does not provide good information about edges. Edge detection techniques locate the edges where either the first derivative of intensity is greater than a particular threshold or the second derivative has zero crossings. In edge based segmentation methods, first of all the edges are detected and then are connected together to form the object boundaries to segment the required regions. The basic two edge based segmentation methods are: Gray histograms and Gradient based methods. The edges can be detected by applying one filter (operator) to the whole image in terms of matrix multiplications. sobel operator, canny operator and Robert's operator .etc. can be used. Result of these methods is basically a binary image. These are the structural techniques based on discontinuity detection. Usually edges occur at the point of intersection of two regions with varying intensities[28]. The advantage of these techniques is that they work very well only on images with good contrast between different regions. Their disadvantages include; they detect all the edges; hence, it is very difficult to find the relation between the edges and the region of interest. In addition, the algorithms are sensitive to noise[22] [21].

2.1.3 Region Based Segmentation Method

The region based segmentation methods are the methods that segments the image into various regions having similar characteristics. There are two basic techniques based on this method [29]. **Region growing methods**: The region growing based segmentation methods are the methods that segments the image into various regions based on the growing of seeds (initial pixels). These seeds can be selected manually (based on prior knowledge) or automatically (based on particular application). Then the growing of seeds is controlled by connectivity between pixels and with the help of the prior knowledge of problem, this can be stopped. **Region splitting and merging methods** the region splitting and merging based segmentation methods uses two basic techniques i.e. splitting and merging for segmenting an image into various regions. Splitting stands for iteratively dividing an image into regions having similar characteristics and merging contributes to combining the adjacent similar regions. Region based methods are immune to the noise and useful when it is easy to define similarity criteria easy but it consumes significant time and memory.

2.1.4 Clustering Based Segmentation Method

The clustering based techniques are the techniques, which segment the image into clusters having pixels with similar characteristics. There are two basic categories of clustering methods: Hierarchical method and Partition based method. The hierarchical methods are based on the concept of trees. In this the root of the tree represents the whole database and the internal nodes represent the clusters. On the other side the partition based methods use optimization methods iteratively to minimize an objective function. In between these two methods there are various algorithms to find clusters. There are basic two types of clustering. [30] **Hard Clustering**: Hard clustering is a simple clustering

technique that divides the image into set of clusters such that one pixel can only belong to only one cluster. An example of a hard clustering based technique is one k-means clustering based technique known as HCM. In this technique, first of all the centers are computed then each pixel is assigned to nearest center. **Soft clustering** soft clustering techniques are most useful for image segmentation in which division is not strict. The example of such type of technique is fuzzy c-means clustering where one pixel can belong to more than one clusters and this degree of belonging is described by membership values. Fuzzy c-means clustering technique is more flexible than other techniques but determining the membership function is not easy.

2.1.5 Watershed Based Methods

The watershed based methods uses the concept of topological interpretation. In this the intensity represents the basins having hole in its minima from where the water spills. When water reaches the border of basin the adjacent basins are merged together. To maintain separation between basins dams are required and are the borders of region of segmentation. These dams are constructed using dilation. The watershed methods consider the gradient of image as topographic surface. The pixels having more gradient are represented as boundaries which are continuous.[19]. The results are more stable, detected boundaries are continuous but gradients calculation is complex and computationally expensive.

2.1.6 Partial Differential Equation (PDE) Based Segmentation Method

The partial differential equation based methods are the fast methods of segmentation. These are appropriate for time critical applications. There are basic two PDE methods: non-linear isotropic diffusion filter (used to enhance the edges) and convex non-

quadratic variation restoration (used to remove noise). The results of the PDE method is blurred edges and boundaries that can be shifted by using close operators. The fourth order PDE method is used to reduce the noise from image and the second order PDE method is used to better detect the edges and boundaries [30] although the computation is more complex.

2.1.7 Artificial Neural Network Based Segmentation Method

The artificial neural network based segmentation methods simulate the learning strategies of human brain for the purpose of decision making. Now days this method is mostly used for the segmentation of medical images. It is used to separate the required image from background. A neural network is made of large number of connected nodes and each connection has a particular weight. In this the problem is converted to issues which are solved using neural network. This method has basic two steps: extracting features and segmentation by neural network [19]. Although the need to write complex program is eliminated, it needs significant training time and number of samples for training.

Segmentation of nontrivial images is a very hard problem made even harder by non-uniform lighting, shadows, overlapping among objects, poor contrast between objects and background, and so on—that has been approached from many different angles, with limited success. Many image segmentation techniques and algorithms have been proposed and implemented during the past 40 years and yet, except for relatively "easy" scenes, the problem of segmentation remains unsolved [18]

2.2 Medical Image segmentation

Medical imaging has originated around 120 years ago [31], [32], and now with different modalities and processes, it plays an important role in public health through improving the process of diagnosis, treatment, and follow-up of a disease that has been diagnosed and/or treated [33]. However studying medical images depends mainly on the visual interpretation by the radiologists, and this may consume significant time and it is usually subjective, depending on the experience of the radiologist[22] . Using Computer-aided systems for medical image analysis help doctors in many applications. Such as stomatology [34], dental surgery and teeth segmentation and labeling[35]–[37], mandible segmentation [38]–[40], reconstruction [41]–[43], and visualization[44], bones fractures detection[45], muscles segmentation[46], and brain visualization and tumor detection, liver, lung, and kidney visualization.

The medical imaging devices and output quality have been tremendously developed through the last 120 years [47]. X-ray , Computed Tomography (CT) , Magnetic Resonance Imaging (MRI), ultrasound, endoscopy are examples for the most currently used modalities for medical images [47]–[51][31].

In Medical image analysis, segmentation is one of the initial steps to identify the object (anatomical organs) which has been examined. The output of the segmentation step cruelly affects the whole process of medical image analysis. Some examples of medical image segmentation include, border detection in angiograms of coronary, surgical planning, simulation of surgeries, brain, liver, lung, kidney segmentation and tumor detection, blood cells automated classification, mass detection in mammograms, heart segmentation and analysis of cardiac images[48].

Due to the needs of exact definition of Region of Interest (ROI), complex visual characteristics of diseases and difficulty of basic knowledge provision complicate the

segmentation step in medical image analysis. Furthermore the segmentation methods are subject to the dimensionality and the modality of imaging and number of available datasets. Thus segmentation has remained a challenge and an active research field. So Medical image analysis is highly required to be automatic for better understanding the type and location of disease and detection of the disease progression as well, [52].

The segmentation method may fail at the same anatomical structure if the images of the structure are obtained by using a different modality or even using the same modality but in different imaging machines[53]. However, there is no universal method which works for all kinds of anatomical structures, the segmentation of medical images is a challenging task. Segmentation of medical image faces many problems because of which the quality of segmentation process gets affected [54]. The problem of uncertainty arises when there is noise in the image which makes the segmentation and classification of image difficult. The reason is that intensity values of pixels are amended because of noise in the image. This alteration in the intensity values of pixels disturbs uniformity in the intensity range of image [55]. Noise can be in the image because of motion in the picture, blurring effect and lack of diverse features etc. The problem of partial volume averaging causes the issue of inconsistency in the intensity values of image pixels. So in order to handle this uncertainty in the medical diagnosis systems image segmentation is playing a vital role.

The most common used techniques for medical image segmentation can be divided into three categories. **1) Conventional approaches** that include basic techniques e.g. Thresholding based methods, Region based techniques, and Edge based techniques. **2) Deformable models** include parametric/ non-parametric and level-set models. **3) Machine learning models** include Supervised approaches (Active Shape Model-ASM, Active appearance Model-AAM, Statistical models, Markov Random Fields-MRF, Conditional

Random Fields-CRF, Atlas-Based, Neural Network, Convolutional Neural Network CNN etc…) and unsupervised techniques (Graph cut, K-Means, Fuzzy C-Mean)

Figure 2-1 will be reviewed in the following subsections.

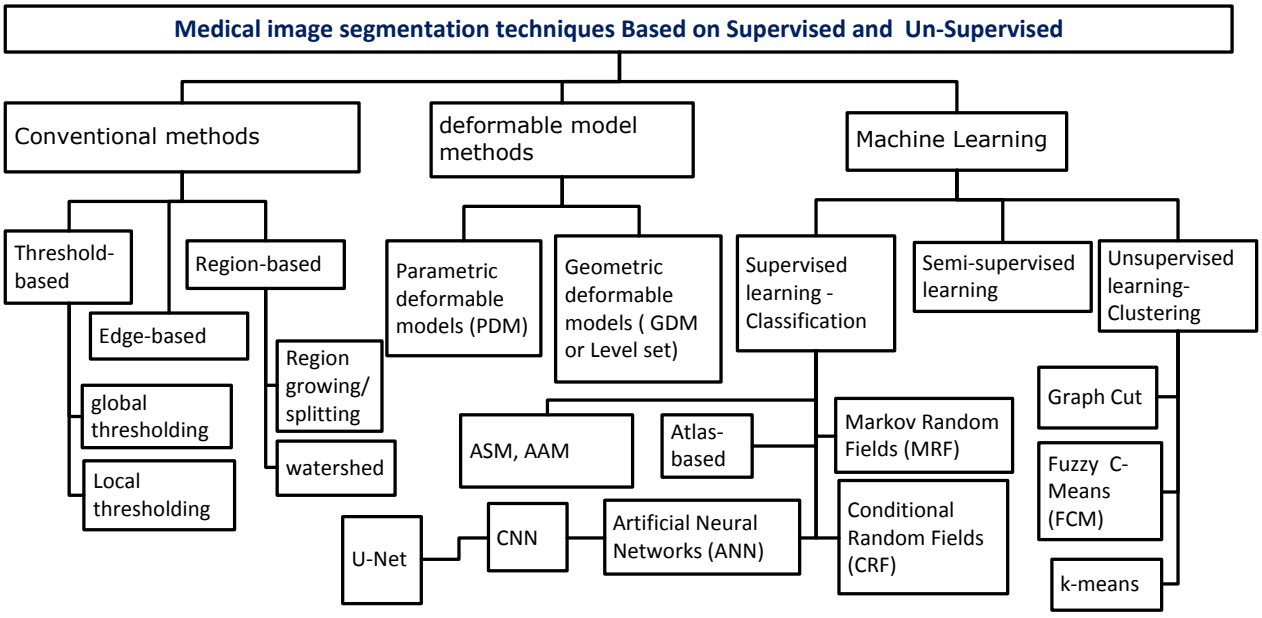


Figure 2-1 Medical image segmentation techniques

2.2.1 Conventional approaches

All the methods in this category used for Nature image segmentation as well as used in medical image segmentation. The three approaches thresholding, Region based, Edge based methods are explained before in section, 2.1.2, 2.1.3.

- **Thresholding**

Thresholding is one of the most common methods used for image segmentation. The image pixel values are classified by comparing it with a threshold value [56]. In [57] an overview of image segmentation techniques based on thresholding process. The five techniques of thresholding discussed in the paper include Mean method, P-tile method,

Histogram Dependent Technique (HDT), Edge Maximization Technique (EMT) and visual technique. A hybrid algorithm based on a self-adaptive thresholding method is proposed to optimize the threshold of the Otsu's method. The attractive features of the algorithm are that its segmentation results are stable, it is robust to noises and it holds for both bi-level and multi-level thresholding cases using multiscale 3D OTSU[58], [59]

- **Region Growing**

In this method an initial point is defined manually and then all points which are connected to that initial point having the same intensity values as that point are selected [60]. The main application of this method in the medical field is to depict the tumor regions. Region growing method cannot be utilized on its own. Additional operations are required to be performed before application of this method. The main disadvantage of this method is that it requires manual depiction of the initial point because of which there is a need to initialize an initial point for every region that is to be extracted. An automatic approach for masses segmentation from ultrasound images proposed in [61]. The method can be said optimal for the segmentation of ultrasound images because they preserve the spatial information and are insensitive to speckle noise. A hybrid approach composed of region growing and region merging introduced with an effective results [62].

- **Edge Based Approaches**

These approaches are the most common way of detecting discontinuity and boundaries of objects with in an image. In this method the two connected pixels have same intensity distribution form the edge and it is not essential that they will form a closed path [63]. The distinction between the pixels is this case is carried out by estimating the intensity gradient. These methods are mainly used as base or central technique for other segmentation approaches.

2.2.2 Deformable Methods

Deformable method works on the basis of object boundaries. The features considered in view of image boundaries are the shape, smoothness and internal forces together with the external forces on the object under consideration [64]. All these factors influence the effectiveness of obtainable results. Closed curves and shapes in the image are utilized to outline the object boundaries. The process of outlining the boundary of an object is a closed curvature or plane that is initially positioned close to the preferred edge and later permitted to experience an iterative reduction progression. In order to keep the segmentation process smooth internal forces are derived within the image. The external forces are derived in order to originate a plane towards the preferred element in the image. The main advantage of these methods is the piece wise continuity.

- **Parametric Deformable Models (Explicit)**

In the statistics of deformable models parametric models are the one that can be described using finite number of parameters. These methods are also called active contours and make use of parameters generated curves for the representation of shape model. Parametric models are further divided into two categories which are: Edge based methods and Region based methods. [65].

- **Non-Parametric Models (Implicit)**

These methods are also called geometric active contour methods. These methods are the level set approaches and are based on the concepts of convolution theory. In the process of defining curve for the segmentation a level set function is utilized together with

the additional time aspect. The evaluation of curve in this case is independent of parametric values [64]. In [66] a 3D medical image segmentation can be analyzed using minimal path deformable model is presented in. The approach is based on extracting the organ contours.

2.2.3 Machine Learning models/methods

- **Unsupervised methods:**

- **Graph cut**

Graph-based segmentation approaches play an important role in medical image segmentation. A graph interprets pixels or regions in the original image into nodes in the graph. Then, the segmentation problem can be transformed into a labeling problem which requires assigning correct label to each node according to its properties. Markov random field (MRF) is successfully used in computer vision and machine learning to model contexture information of pixels. This contexture information provides a mechanism for obtaining image properties.[67]

Although GCs/ GS methods and their variants have been receiving big success in medical image segmentation, there are still some limitations for the clinic applications. A common problem of graph-based approaches is the computation complexity. Since graph-based approaches use graph as a representation of an image, with the incensement of resolutions, dimensions, and modalities of medical images, the corresponding nodes and edges are dramatically increased. Another problem for GCs is the "small cut" or shrinking behavior and leakages, which tends to have small segmentations due to minimizing the sum of edge weights in the cut [68].

In recent years, the emergence of new algorithms, such as multimodality image technologies, hybrid method, deep learning, etc., provide wider research spaces for them

and made the use of GCs/ GS more flexible and more powerful. Many deep neural network models have been adopted successfully in various fields. Recent works also extended deep learning technique to solve the complex medical image segmentation problem, for example, brain tumor segmentation problem [69]. Among them, the combination of the deep learning and the traditional method is a new powerful technique which achieved high performance in some aspects [70]–[73]. In recent published work, Fang et al. [70] presented a novel framework, called CNN-GS, integrating convolutional neural networks (CNN) with GS method to segment nine-layer boundaries on retinal optical coherence tomography images. Fig. 11 illustrates the outline of the CNN-GS algorithm. CNN-GS method was composed of two main steps. One is CNN layer boundary classification and another is GS layer segmentation based on the CNN probability maps. CNN was used to extract features of retinal layer boundaries and train a classifier. Then, GS method used the probability maps created by the CNN to detect the layer boundary position. Sui et al. [71] proposed a similar method for the choroid segmentation of OCT retinal images using multiscale CNNs combined with GS. Lu et al. [72] developed a deep learning algorithm with GC refinement to segment the liver in CT scans. 3-D convolutional neural network was applied to obtain an initial segmentation and learn probability map. GC was then used to refine the initial segmentation. [73]

- **Clustering**

If we compare the functions of clustering and classifiers we can say that both are carrying out the same function with the difference in their way of working. The classifiers make use of training data to classify the image and thus are called supervised methods. Clustering approach contains unsupervised methods as it does not make use of training data. This inability of learning in clustering approach is compensated by iteratively dividing the image through the segmentation process and then illustrating the possessions of every division. In other words we can say that clustering techniques instruct themselves by means

of existing statistics [74]. Clustering process is mainly suitable for applications where the intensities distributions of pixels in the image are fit detached. The main application of this method can be observed in the segmentation of MRI.

In **K-Means** clustering, the clustering is carried out by iteratively calculating the mean of intensities values of each separated class or cluster of the image. And the segmentation is carried out by categorizing each pixel with the closest obtained mean of the image [75] , while in **Fuzzy C-Means** the Segmentation through this process is carried out on the basis of fuzzy set premise. This process is also called generalization of k-means process. The difference between the two processes is that the points are categorized in separate classes in k-means process whereas fuzzy c-means permits the points to be connected with more than one class[76] .

➢ Supervised methods:

• Atlas Guided Approaches

Medical images segmentation based on Atlas guided approaches is a way of analyzing image through labeling a preferred structure or set of framework commencing images made through modalities of medical imaging. The main purpose of this approach is to lend a hand to radiologists in the discovery and identification of diseases. The working flow of approach is optimized by identifying significant anatomy in the medical images [77]. These approaches are also called adaptable templates. The segmentation in this case is carried out by preparing an atlas using compiled information of anatomy. After the generation of atlas it is used as a reference structure for the segmentation of fresh images. These approaches consider registration problem to handle the segmentation process. Atlas wrapping is used for the segmentation process that works by mapping the generated atlas on the objected image [78]. The main application of these approaches is in the images where there is no well-defined relation between image pixels and regions. The other main

applications include its use in clinical practice and computer aided diagnosis to analyze shape and morphological differences between image regions.

- **Active Shape Model**

Active Shape Model (ASM) is a subtype of statistical model that manipulates a shape model to describe the location of structures in a target image[79]. Given a rough starting approximation, an instance of a model can be fit to an image. By choosing a set of shape parameters, for the model we define the shape of the object in an object-centered coordinate frame. We can create an instance of the model in the image frame by defining the position, orientation and scale[80], [81].

ASM had been used to extract the mandible and mandible canal based and extended from 2D to 3D image and introduce enhancements for more accurate segmentation results[82]. While a fully automatic methodology to segment the mandible based on a statistical shape model (SSM) obtained a 3D reconstruction of the mandible in order to locate some structures [83], it was not focused on the analysis of bone tissues. Some of these features, such as density, are relevant in implantology since the quality of the bone is related to the success of the treatment. Another method proposed to solve three problems. The 3D mandible location, the cross section extraction of the mandible bone, and the bone tissue segmentation.by locating the three-dimensional bounding box that contains the mandible using the Particle Swarm Optimization (PSO) technique and a discrete snake model (active contour model) is then defined as a spline over the semi-ellipse obtained from the best PSO particle in addition to use canny for final edge detection[84]. The method that employed a multi-atlas segmentation to obtain an initial segmentation for the considered organs at risk and Active Shape Model (ASM) segmentation to refine the initial segmentation of some of the organs got the second position on MICCAI 2015[85].

- **Active Appearance Model (AAM)**

While Active Shape Model does not take advantage of all the available information - the texture across the target object, this can be modeled using an Active Appearance Model (AAM). this algorithm allows to find the parameters of such a model which generates a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation.[39], [79], [86]–[92].

While an approach focused to improve AAM and decrease annotation time needed for each CT slice to create and train AAM and develop a semi-automatic landmarking technique[39], a technique proposed in MICCAI 2009 based on initialization the AAM with a parts-and-geometry model, search with a global AAM followed by search with local AAMs, then post-processing using linear regression to enhance the segmented mandible[93]. On the other hand a development of AAM to be used with 3D medical images is proposed for mandible and mandible canal segmentation[82].

In MICCAI 2015, the first rank for a paper that succeeded to segment mandible scoring the best results proposed a new method based on Active Appearance Models (AAM) built from manually segmented examples of High quality anatomical correspondences for the models are generated using a Minimum Description Length (MDL) GroupWise Image Registration (GIR) method. A multi start optimization scheme is used to robustly match the model to new images to obtain the anatomical correspondences on the surfaces using a variant of the Minimum Description Length approach to GroupWise image registration (MDL-GIR). [94].

- **Atlas as Individual Image**

The atlas is generated by compiling information on the anatomy that requires segmenting. This atlas is then used as a reference frame for segmenting new images. Atlas-guided approaches are a powerful tool for medical image segmentation when a standard

atlas or template is available. Conceptually, atlas-guided approaches are similar to classifiers except they are implemented in the spatial domain of the image rather than in a feature space.

The standard Atlas-guided approach treats segmentation as a registration problem. It first finds a one-to-one transformation that maps a pre-segmented atlas image to the target image that requires segmenting. This process is often referred to as atlas warping. The warping can be performed using linear transformations but because of anatomical variability [22]. This process results in a correspondence field, which maps each pixel in the atlas space to one in the patient coordinate system[95].

When a single atlas would be constructed and used for registration and segmentation it called single atlas approach[96] [25]. While the underlying fusion of Multi Atlas-Based method is that, multiple independent classifiers might produce better classification .Multi atlas-based segmentation registers many independently built atlases to a target image and then combines their segmentation labels. There exist different ways for segmenting a particular target image, e.g., to select all the atlases or only their subset as well as to choose one or another strategy of combining the selected atlases to produce the goal region map[25][97].

Atlas based approaches proposed in MICCAI 2009 (**M**edical **I**mage **C**omputing and **C**omputer-**A**ssisted **I**ntervention) by three different groups to solve segmentation challenge. A proposed approach combined a multiple atlas fusion strategy and a hierarchical atlas registration approach using the advantages of GPU technology to accelerate the deformable atlas registration and to make multi-atlas segmentation computationally feasible[98].

Atlas-based segmentation approach had been used in combination with label fusion in order to initialize a segmentation pipeline that is based on using statistical appearance models and geodesic active contours. An anatomically correct approximation of the

segmentation result is provided by atlas-based segmentation acts as a starting point for an iterative refinement of this approximation[99].

Moreover, a hybrid method that combined atlas registration using intensity based non-rigid registration based on b-spline and a level set function for segmenting the mandible in the test image. Nevertheless, the most similar atlas selection strategy need improved since registration to each atlas image is a time-consuming work. A probabilistic atlas can be taken into consideration to solve the problem[100].

Atlas image also applied in a combination with a hybrid deformable image registration, the result of which is then refined using a deformable surface model approach. Segmentation fusion using multiple atlases is also employed to further improve the segmentation accuracy in MICCAI 2010 [101].

While Multi-atlas registration-based segmentation is refined by a graph-cut optimization step [102], a patient specific atlas was estimated from a spiral CT atlases using a sparse label propagation strategy, then, the patient-specific atlas is integrated into a convex segmentation framework based on maximum a posteriori probability (MAP) for accurate segmentation [97].

In MICCAI 2015 challenges, the second and third ranked approaches were based on multiple atlases. In the third place a multi-atlas approach for the segmentation of multiple structures in the head and neck CT images, a patient image was first aligned with an average head and neck CT atlas on the global level. The aligned image was further processed by per-forming local non-rigid registrations with multiple atlases. The subsequent labels deformed from the atlases were then combined with weights determined by the local correlation coefficients between the patient image and the registered atlas images[103]. While in the second place an algorithm consists of two building blocks. First, a multi-atlas segmentation employed to obtain an initial segmentation for the considered organs at risk.

Secondly, Active Shape Model (ASM) had been used to refine the initial segmentation of some of the organs[85].

- **Artificial Neural Network**

The most related ANN models to medical image segmentation including Convolutional neural network CNN, U-Net, Stacked U-Net, and Bridged U-Net would be reviewed with details in the following section 2.3.2

Method	**Description**	**Advantages**	**Limitations**
Thresholding	based on the histogram peaks of the image to find particular threshold values	These methods are fastest, simplest and easiest to implement no need of previous information, simplest method	These methods are responsive to artifacts and piecewise continuity is not assured by them highly dependent on peaks, spatial details are not considered based
Region Growing	based on partitioning image into homogeneous regions	These methods assure the piecewise continuity and are less sensitive to noise useful when it is easy to define similarity criteria	Position of the start point and blurring affects are the main limitations of these methods expensive method in terms of time and memory
Clustering	based on division into homogeneous clusters	These methods are easy to implement and can also be used as starting point for other approaches. fuzzy uses partial membership therefore more useful for real problems	They require a spatial constraint to perform well. determining membership function is not easy
Deformable Methods		They effectively handle the topological changes and assure the piecewise continuity. These methods are noise insensitive and provide sub-pixel accuracy.	Requires the tuning of parameters and thus can affect speed of the system.

Atlas guided Approaches		These approaches are fact and assure an optimum solution for two class segmentation.	Precise segmentation of difficult composition is itself difficult.
Edge based methods	based on discontinuity detection	They are easy to implement and offer effective computational factor. good for images having better contrast between objects	Not appropriate to figure out all kinds of problems. not suitable for wrong detected or too many edges

Table 2-1 Image segmentation methods advantages and limitations

2.3 Deep Learning approaches in Computer Vision and for image segmentation

2.3.1 Deep Learning with CNN for Medical Imaging

Convolutional networks convolutional (LeCun, 1989), also known as neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers[104], [105].

- **Deep CNN architectures for classification**

LeNet [106] and AlexNet [106] introduced over a decade later, were in essence very similar models. Both networks were relatively shallow, consisting of two and five convolutional layers, respectively, and employed kernels with large receptive fields in layers

close to the input and smaller kernels closer to the output. AlexNet did incorporate rectified linear units instead of the hyperbolic tangent as activation function.

After 2012 the exploration of novel architectures took off. By stacking smaller kernels, instead of using a single layer of kernels with a large receptive field, a similar function can be represented with less parameter. Simonyan and Zisserman (2014) [107] were the first to explore much deeper networks, and employed small, fixed size kernels in each layer. A 19-layer model often referred to as VGG19 or OxfordNet won the ImageNet challenge of 2014 [108].

On top of the deeper networks, more complex building blocks have been introduced that improve the efficiency of the training procedure and again reduce the number of parameters. Szegedy et al. (2014) [109] introduced a 22-layer network named GoogLeNet, also referred to as Inception, which made use of so-called inception blocks [110], a module that replaces the mapping with a set of convolutions of different sizes. Similar to the stacking of small kernels, this allows a similar function to be represented with less parameters. The ResNet architecture [111] won the ImageNet challenge in 2015 and consisted of so-called ResNet-blocks. Rather than learning a function, the residual block only learns the residual and is thereby pre-conditioned towards learning mappings in each layer that are close to the identity function. This way, even deeper models can be trained effectively. Consequently, AlexNet or other simple models such as VGG are popular for medical data, though recent landmark studies all use a version of GoogleNet called Inception v3[112]–[114]. Whether this is due to a superior architecture or simply because the model is a default choice in popular software packages is again difficult to assess.

Three papers used an architecture leveraging the unique attributes of medical data: two use 3D convolutions[115], [116] instead of 2D to classify patients as having Alzheimer;

Kawahara et al. (2016b)[117] applied a CNNlike architecture to a brain connectivity graph derived from MRI diffusion-tensor imaging (DTI). In order to do this, they developed several new layers which formed the basis of their network, so-called edge-to-edge, edge-to-node, and node-to-graph layers. They used their network to predict brain development and showed that they outperformed existing methods in assessing cognitive and motor scores.

- **Deep CNN architectures for Segmentation**

Segmentation is a common task in both natural and medical image analysis and to tackle this, CNNs can simply be used to classify each pixel in the image individually, by presenting it with patches extracted around the particular pixel. A drawback of this sliding-window approach is that input patches from neighboring pixels have huge overlap and the same convolutions are computed many times. Fortunately, the convolution and dot product are both linear operators and thus inner products can be written as convolutions and vice versa. By rewriting the fully connected layers as convolutions, the CNN can take input images larger than it was trained on and produce a likelihood map, rather than an output for a single pixel. The resulting 'fully convolutional network' (fCNN) can then be applied to an entire input image or volume in an efficient fashion. However, because of pooling layers, this may result in output with a far lower resolution than the input. 'Shift-and-stitch'[118] is one of several methods proposed to prevent this decrease in resolution. The fCNN is applied to shifted versions of the input image. By stitching the result together, one obtains a full resolution version of the final output, minus the pixels lost due to the valid convolutions.

Segmentation is the most common subject of papers applying deep learning to medical imaging [108], and as such has also seen the widest variety in methodology, including the development of unique CNN-based segmentation architectures and the wider application of RNNs.

Although these specific segmentation architectures offered compelling advantages, many other methods have also obtained excellent segmentation results with patch-trained neural networks. One of the earliest papers covering medical image segmentation with deep learning algorithms used such a strategy and was published by Ciresan et al. (2012) [119]. They applied pixel-wise segmentation of membranes in electron microscopy imagery in a sliding window fashion.

fCNNs have also been extended to 3D and have been applied to multiple targets at once: Korez et al. (2016)[120], used 3D fCNNs to generate vertebral body likelihood maps which drove deformable models for vertebral body segmentation in MR images, Zhou et al. (2016) segmented nineteen targets in the human torso, and Moeskops et al. (2016b)[121] trained a single fCNN to segment brain MRI, the pectoral muscle in breast MRI, and the coronary arteries in cardiac CT angiography (CTA).

One challenge with voxel classification approaches is that they sometimes lead to spurious responses. To combat this, groups have tried to combine fCNNs with graphical models like MRFs [122], [123] and Conditional Random Fields (CRFs)[124] to refine the segmentation output. In most of the cases, graphical models are applied on top of the likelihood map produced by CNNs or fCNNs and act as label regularizes.

2.3.2 CNN architecture and its variations

This section explain U-Net model with full details and some of the most common used architecture based on U-Net.

- **U-Net**

One of the most well-known structures for medical image segmentation is U-Net which is a special type of Convolutional neural networks (CNN), initially proposed by Ronneberger et al. using the concept of deconvolution. This model is built upon the elegant architecture of FCN. Besides the increased depth of network to 19 layers, U-Net benefits from a superior design of skip connections between different stages of the network [14]. It employs some modifications to overcome the trade-off between localization and the use of context. This trade-off rises since the large-sized patches require more pooling layers and consequently will reduce the localization accuracy. On the other hand, small-sized patches can only observe small context of input. The proposed structure consists of two paths of analysis and synthesis. The analysis path follows the structure of CNN Figure 2-2. The synthesis path, commonly known as expansion phase, consists of an upsampling layer followed by a deconvolution layer. The most important property of U-Net is the shortcut connections between the layers of equal resolution in analysis path to expansion path. These connections provide essential high-resolution features to the deconvolution layers.

This novel structure has attracted a lot of attention in medical image segmentation and based on which many variations have been developed. For instance, Gordienko et al. [125] explored lung segmentation in X-ray scans with a U-Net structure-based network. The obtained results have demonstrated that U-Net is capable of fast and precise image segmentation. In the same study, the proposed model was tested on single CPU and compared with multiple CPUs and GPUs to evaluate the effect of hardware on model

performance. The demonstrated results showed 3 and 9.5 times speedup respectively. DCAN [126] is another model which applied multi-levelcontextual information and benefitted from the auxiliary classifier on top of the U-Net. Their design showed 0.8001 of segmentation accuracy on gland segmentation which is almost 2% higher than the original U-Net in a shorter time of 1.5 s per testing image. The improved accuracy is due to the capability DCAN structure to combat the errors of touching object segmentation. [127]

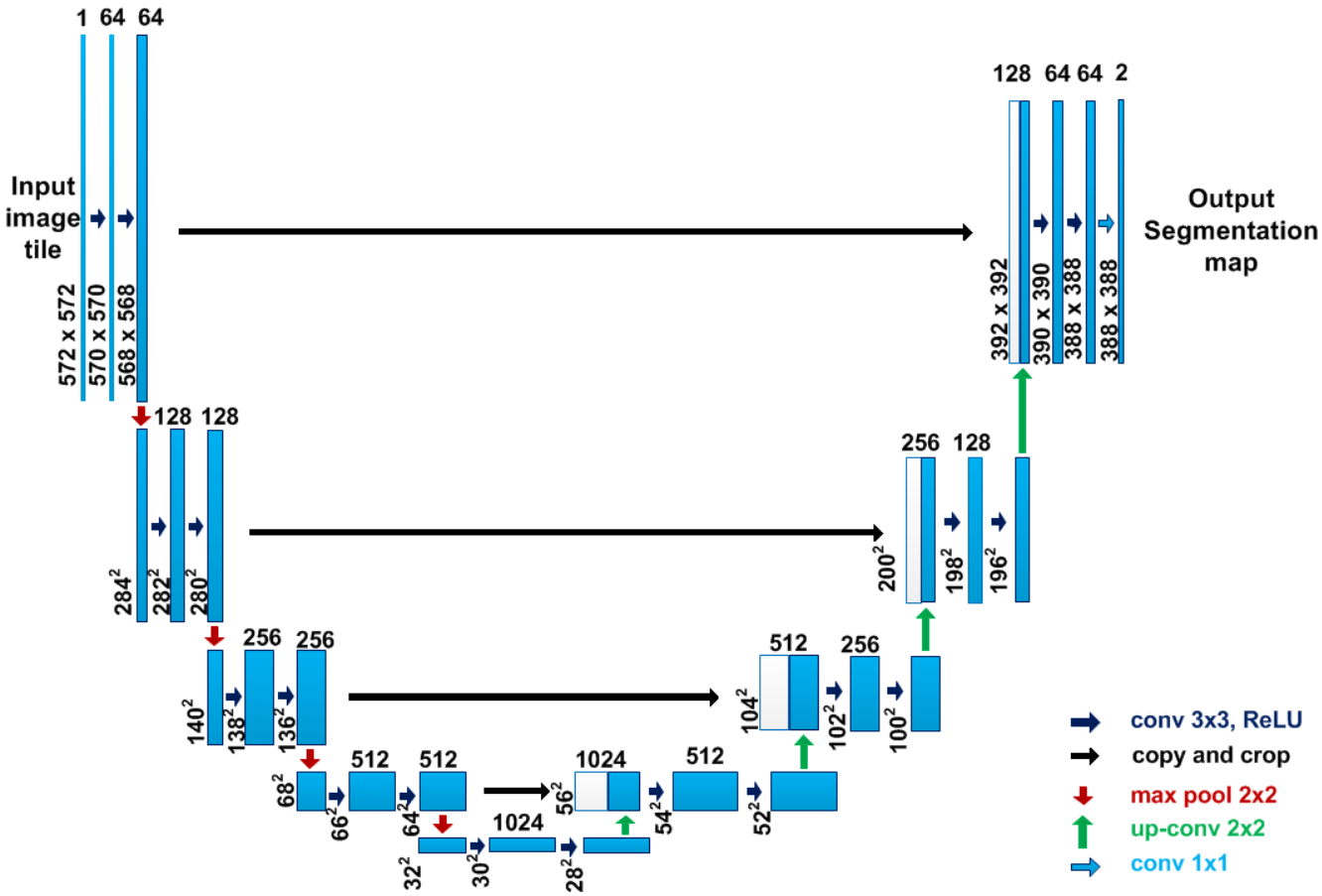


Figure 2-2 U-Net model structure for medical image segmentation,

The figure adopted from "Olaf Ronneberger, Philipp Fischer, and Thomas Brox" [14]

Ronneberger et al. (2015) [14] took the idea of the fCNN one step further and proposed the U-net architecture, comprising a 'regular' fCNN followed by an upsampling

part where ’up’-convolutions are used to increase the image size, coined contractive and expansive paths. Although this is not the first paper to introduce learned upsampling paths in convolutional neural networks[118], the method combined it with so called skip-connections to directly connect opposing contracting and expanding convolutional layers. A similar approach was used by Cicek et al. (2016)[128] for 3D data. Milletari et al. (2016) [129]proposed an extension to the U-Net layout that incorporates ResNet-like residual blocks and a Dice loss layer, rather than the conventional cross-entropy, that directly minimizes this commonly used segmentation error measure.

The most well-known, in medical image analysis, of these novel CNN architectures is U-net, published by Ronneberger et al. (2015) [14]. The two main architectural novelties in U-net are the combination of an equal amount of upsampling and downsampling layers. Although learned upsampling layers have been proposed before, U-net combines them with so-called skip connections between opposing convolution and deconvolution layers which concatenate features from the contracting and expanding paths. From a training perspective this means that entire images/scans can be processed by U-net in one forward pass, resulting in a segmentation map directly. This allows U-net to take into account the full context of the image, which can be an advantage in contrast to patch-based CNNs. Furthermore, in an extended paper by Cicek et al. (2016)[128], it is shown that a full 3D segmentation can be achieved by feeding U-net with a few 2D annotated slices from the same volume. Other papers have also built derivatives of the U-net architecture; Milletari et al. (2016b)[129], for example, proposed a 3D-variant of U-net architecture, called V-net, performing 3D image segmentation using 3D convolutional layers with an objective function directly based on the Dice coefficient. Drozdzal et al. (2016)[130] investigated the use of short ResNet-like skip connections in addition to the long skip-connections in a regular U-net.

Segmentation of lesions combines the challenges of object detection and organ and substructure segmentation in the application of deep learning algorithms. Global and local context are typically needed to perform accurate segmentation, such that multi-stream networks with different scales or non-uniformly sampled patches are used as in for example [131]. In lesion segmentation, U-net and similar architectures to leverage both this global and local context are applied. The architecture used in[132], similar to the U-net, consists of the same downsampling and upsampling paths, but does not use skip connections. Another U-net-like architecture was used in[133] to segment white matter lesions in brain MRI. However, they used 3D convolutions and a single skip connection between the first convolutional and last deconvolutional layers.

RNNs have recently become more popular for segmentation tasks. For example, Xie et al. (2016)[134] used a spatial clockwork RNN to segment the perimysium in H&E-histopathology images. This network considers prior information from both the row and column predecessors of the current patch. To incorporate bidirectional information from both left/top and right/bottom neighbors, the RNN is applied four times in different orientations and the end-result is concatenated and fed to a fully-connected layer. This produces the final output for a single patch. Stollenga et al. (2015) [135] where the first to use a 3D LSTM-RNN with convolutional layers in six directions. Andermatt et al. (2016) [136]used a 3D RNN with gated recurrent units to segment gray and white matter in a brain MRI data set. Chen et al. (2016d)[126] combined bi-directional LSTM-RNNs with 2D U-net-like-architectures to segment structures in anisotropic 3D electron microscopy images. Last, Poudel et al. (2016) [137] combined a 2D U-net architecture with a gated recurrent unit to perform 3D segmentation.

- **V-Net**

Probably one of the most famous derivations of U-Nets is the V-Net proposed by Milletari et al. [138]. They applied the convolutions in the contracting path of the network, both for extracting the features and reducing the resolution by selecting appropriate kernel size and stride (kernel size is 2 × 2 × 2, and stride is 2). The convolutions serve as pooling with the advantage of having smaller memory footprint since unlike pooling layers, switches that map the output of pooling layer back to the input do not need to be stored for backpropagation. This is similar to application

deconvolution instead of up-pooling. The expansion phase will extract features and expand the concatenated low-resolution feature map and ultimately produce two channels volumetric segmentation at the last convolutional layer. Then, the output turns to probabilistic segmentation maps and passes to voxel-wise softmax for background and foreground segmentation. V-Net has been used in with a larger receptive field (covers 50–100% of the input image) and multi-scale (four different resolutions) and delivered up to 12% higher Dice coefficient compared to original V-Net.

- **W-Net Staked net with soft N-cut**

This model introduced a deep learning-based approach for fully unsupervised image segmentation. The proposed algorithm is based on concatenating together two fully convolutional networks into an encoder-decoder framework, where each of the FCNs are variants of the U- Net architecture. Training is performed by iteratively minimizing the reconstruction error of the decoder along with a soft normalized cut of the encoder layer. As the resulting segmentations are typically coarse and over-segmented, CRF smoothing and hierarchical merging applied to produce the final outputted segments. On the Berkeley Segmentation Data Set, the model outperformed a number of existing classical and recent techniques, achieving performance near human level by some metrics. This method will be

useful in cases where it is difficult to obtain labeled pixelwise supervision, for in- stance in domains such as biomedical image analysis where new data sets may require significant re-labeling for semantic segmentation methods to work well. [139]

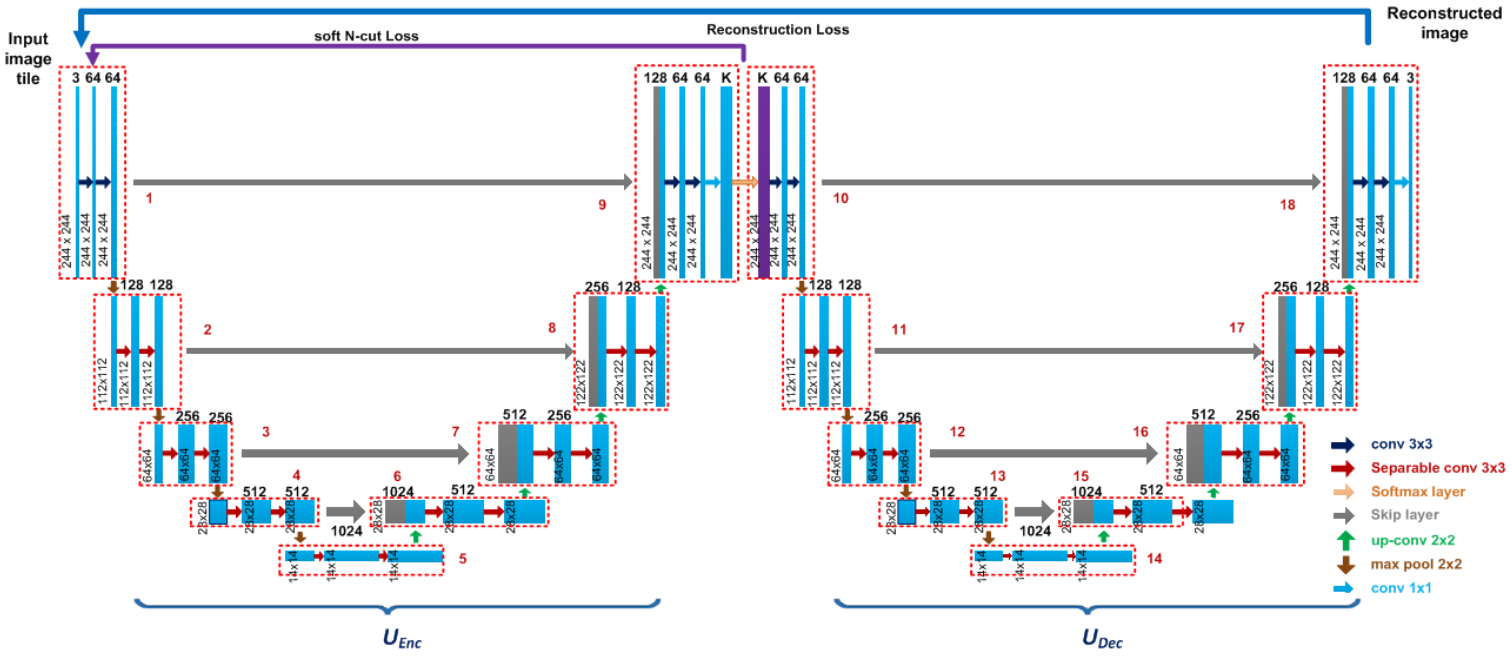


Figure 2-3 W-Net: A Deep Model for Fully Unsupervised Image Segmentation , The figure adopted from "Xide Xia, Brian Kulis" [139]

➢ 2Bridged U-Net

In this paper, we focus on three problems in deep learning based medical image segmentation. Firstly, U-net, as a popular model for medical image segmentation, is difficult to train when convolutional layers increase even though a deeper network usually has a better generalization ability because of more learnable parameters. Secondly, the exponential ReLU (ELU), as an alternative of ReLU, is not much different from ReLU when the network of interest gets deep. Thirdly, the Dice loss, as one of the pervasive loss functions for medical image segmentation, is not effective when the prediction is close to ground truth and will cause oscillation during training. To address the aforementioned three

problems, we propose and validate a deeper network that can fit medical image datasets that are usually small in the sample size. Meanwhile, we propose a new loss function to accelerate the learning process and a combination of different activation functions to improve the network performance. The experimental results suggest that our network is comparable or superior to state-of-the-art methods.

The network is based on U-net, which is a classical encoder-decoder net in medical image application. Based on U-net, a stacked U-net is proposed. The stacked U-net improves network performance by using the first U-net to find a coarse feature and use the second U-net to obtain a fine result. The stacked U-net is, however, not useful for medical image segmentation. It is hard to reach convergence and usually dive into a sub-optimal solution because the increasing complexity of network. To overcome the issue, the model proposed a network bridging method. Different from the previous stacked U-net which acquires large number training data, bridging two U-nets can reduce the training cost and makes the network fit for medical application where the training data are usually not sufficient. This is because bridging two U-nets can fully use different features in multi levels, which will accelerate the convergence of neural network. Our network structure is shown on Fig. 2. The gray block represents a ELU cluster (2conv-BN-ELU blocks), and the yellow block represents a ReLU cluster (2 conv-BN-ReLU blocks). The dotted lines represent network bridging. The red lines represents skip connections. This model use MICAAI PROMISE12 dataset to evaluate our network and the result shows that our network performs better than original U-net, stacked U-net and other state-of-the-art methods. [16]

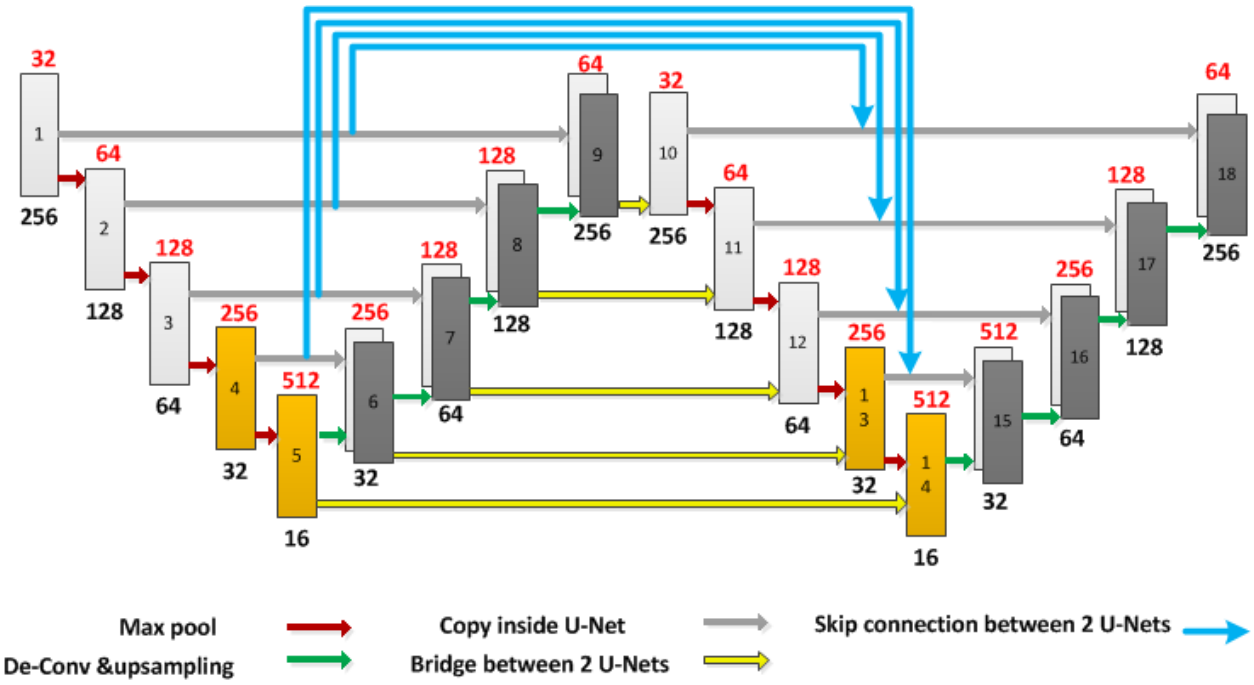


Figure 2-4 prostate segmentation using 2bridged U-Net

2.3.3 Loss function variations

Deep Learning algorithms use stochastic gradient descent approach to optimize and learn the objective. To learn an objective accurately and faster, we need to ensure that our mathematical representation of objectives, also known as loss functions are able to cover even the edge cases. The introduction of loss functions have roots in traditional machine learning, where these loss functions were derived on basis of distribution of labels [140]. For example, Binary Cross Entropy is derived from Bernoulli distribution and Categorical Cross-Entropy from Multinoulli distribution. In this paper, we have focused on Semantic Segmentation instead of Instance Segmentation; therefore the number of classes at pixel level is restricted to 2. Here, we will go over 15 widely used loss functions and understand their use-case scenarios. [141]

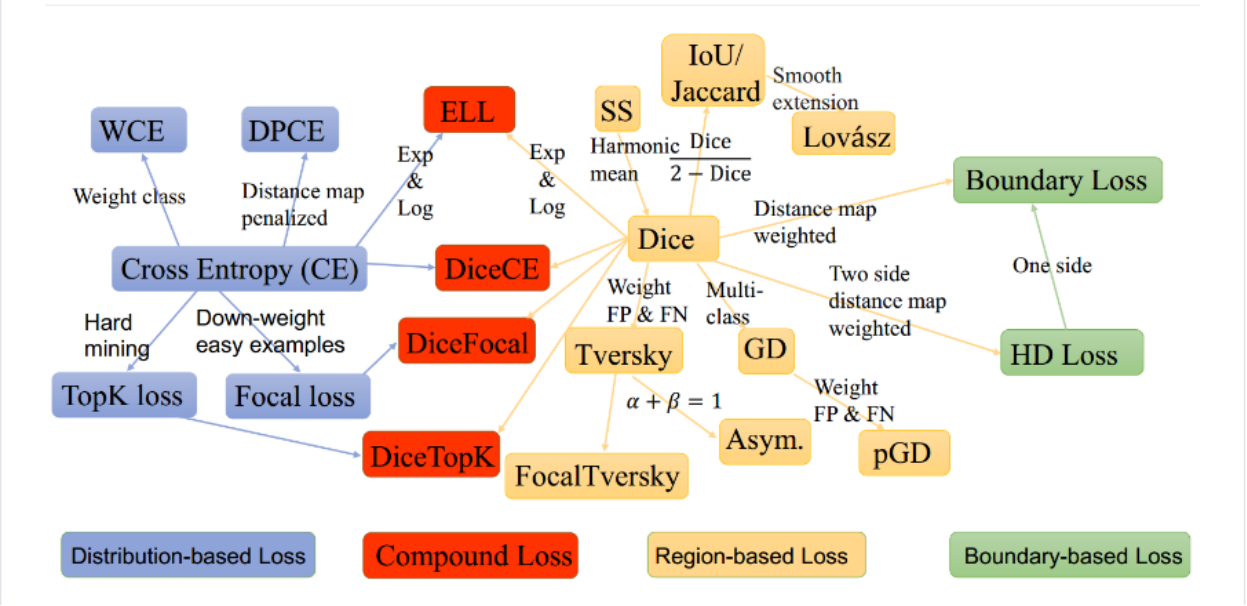


Figure 2-5 Semantic taxonomy for loss functions

This figure is adapted from "Jun Ma" [140]

Type Loss Function	Type Loss Function
Distribution-based Loss	Binary Cross-Entropy Weighted Cross-Entropy Balanced Cross-Entropy Focal Loss Distance map derived loss penalty term
Region-based Loss	Dice Loss Sensitivity-Specificity Loss Tversky Loss Focal Tversky Loss Log-Cosh Dice Loss(ours)
Boundary-based Loss	Hausdorff Distance loss Shape aware loss
Compounded Loss	Combo Loss Exponential Logarithmic Loss

Table 2-2 Semantic taxonomy for the loss functions been used in CNN [141]

➢ Distribution-based Loss

- **Binary Cross-Entropy**

Cross-entropy [142] is defined as a measure of the difference between two probability distributions for a given random variable or set of events. It is widely used for classification objective, and as segmentation is pixel level classification it works well. Binary Cross-Entropy is defined as:

$$L_{BCE}\,(y, y\hat{}) = -(ylog(y\hat{}) + (1-y)\,log(1-y\hat{}))$$

Here, $\boldsymbol{y}\hat{}$ is the predicted value by the prediction model.

- **Weighted Binary Cross-Entropy**

Weighted Binary cross entropy (WCE) [143] is a variant of binary cross entropy variant. In this the positive examples get weighted by some coefficient. It is widely used in case of skewed data [144] Weighted Cross Entropy can be defined as:

$$L_{W-BCE}(y, y\hat{}) = -(\beta * ylog(y\hat{}) + (1-y)log(1-y\hat{}))$$

Note: β value can be used to tune false negatives and false positives. E.g.; if you want to reduce the number of false negatives then set $\beta > 1$, similarly to decrease the number of false positives, set $\beta < 1$.

- **Balanced Cross-Entropy**

Balanced cross entropy (BCE) [145] is similar to Weighted Cross Entropy. The only difference is that in this apart from just positive examples, we also weight the negative examples. Balanced Cross-Entropy can be defined as follows:

$$L_{BCE}\,(y, y\hat{}) = -(\,\beta * ylog(y\hat{}) + (1 - \beta\,) * (1 - y)\,log(1 - y\hat{}))$$

Here, $\boldsymbol{\beta}$ is defined as $1 - \frac{y}{H*W}$

- **Focal Loss**

Focal loss (FL) [146] can also be seen as variation of Binary Cross-Entropy. It down-weights the contribution of easy examples and enables the model to focus more on learning hard examples. It works well for highly imbalanced class scenarios, as shown in fig 1. Let's look at how this focal loss is designed. We will first look at binary cross entropy loss and learn how Focal loss is derived from cross-entropy.

This loss is an improvement to the standard cross-entropy criterion. This is done by changing its shape such that the loss assigned to well-classified examples is down-weighted. Ultimately, this ensures that there is no class imbalance. In this loss function, the cross-entropy loss is scaled with the scaling factors decaying at zero as the confidence in the correct classes increases. The scaling factor automatically down weights the contribution of easy examples at training time and focuses on the hard ones.

$$FL(p_t) = -(1 - p_t)^{\gamma}\,log(p_t)$$

$$CE = \begin{cases} -log(p)\,, & if\ y = 1 \\ -log(1 - p), & otherwise \end{cases}$$

To make convenient notation, Focal Loss defines the estimated probability of class as:

$$p_t = \begin{cases} p, & if\ y = 1 \\ 1 - p, & otherwise \end{cases}$$

Therefore, Now Cross-Entropy can be written as,

$$CE(p, y) = CE(p_t) = -log(p_t)$$

- **Distance map derived loss penalty term**

Distance Maps can be defined as distance (euclidean, absolute, etc.) between the ground truth and the predicted map. There are two ways to incorporate distance maps, either create neural network architecture where there's a reconstruction head along with segmentation, or induce it into loss function. Following same theory, Caliva et al. [147] have used distance maps derived from ground truth masks and created a custom penalty based loss function. Using this approach, its easy to guide the networks focus towards hard-to-segment boundary regions. The loss function is defined as:

$$L(y, p) = \frac{1}{N}\sum_{i=1}^{N}(1 + \emptyset)(\odot)L_{CE}(y, p)$$

Here, $\emptyset$ are generated distance maps

Note here, constant 1 is added to avoid vanishing gradient problem in U-Net and V-Net architectures.

➢ Region-based Loss

Region-based loss functions aim to minimize the mismatch or maximize the overlap regions between ground truth and predicted segmentation.

- **Dice Loss**

The Dice coefficient is widely used metric in computer vision community to calculate the similarity between two images. Later in 2016, it has also been adapted as loss function known as Dice Loss [148] .This loss is obtained by calculating smooth dice coefficient function. This loss is the most commonly used loss is segmentation problems.

$$DL(y,p) = 1 - \frac{2yp\hat{} + 1}{y + p\hat{} + 1}$$

Here, 1 is added in numerator and denominator to ensure that the function is not undefined in edge case scenarios such as

$$y = p\hat{} = 0$$

- **Tversky Loss**

Tversky index (TI) [149] can also be seen as an generalization of Dices coefficient. It adds a weight to FP (false positives) and FN (false negatives) with the help of coefficient.

$$TI(p,p\hat{}) = \frac{pp\hat{}}{pp\hat{} + \beta(1-p)p\hat{} + (1-\beta)p(1-p\hat{})}$$

Here, when $\beta = 1/2$, It can be solved into regular Dice coefficient. Similar to Dice Loss, Tversky loss can also be defined as:

$$TL(p,p\hat{}) = 1 - \frac{1 + pp\hat{}}{1 + pp\hat{} + \beta(1-p)p\hat{} + (1-\beta)p(1-p\hat{})}$$

It sets different weights to false negative (FN) and false positive (FP), which is different from dice loss using the equal weights for FN and FP.

- **Focal Tversky Loss**

Similar to Focal Loss, which focus on hard example by down-weighting easy/common ones. Focal Tversky loss [150] also attempts to learn hard-examples such as with small ROIs(region of interest) with the help of coefficient as shown below:

$$FTL = \sum_{c}(1 - TI_c)^{\gamma}$$

Here TI indicates tversky index, γ and can range from [1,3].

- **Sensitivity Specificity Loss**

The total loss is the weighted sum of the mean squared difference of sensitivity and specificity. Similar to Dice Coefficient, Sensitivity and Specificity are widely used metrics to evaluate the segmentation predictions. In this loss function, we can tackle class imbalance problem using w parameter. The loss [151] is defined as:

$$SSL = w * sensitivity + (1 - w) * specificity$$

Where

$$sensitivity = \frac{TP}{TP + FN}$$

And

$$specificity = \frac{TN}{TN + FP}$$

- **Log-Cosh Dice Loss**

Dice Coefficient is a widely used metric to evaluate the segmentation output. It has also been modified to be used as loss function as it fulfills the mathematical representation of segmentation objective. But due to its non-convex nature, it might fail in achieving the optimal results. Lovsz-Softmax loss [152] aimed to tackle the problem of non-convex loss function by adding the smoothing using Lovsz extension. Log-Cosh approach has been widely used in regression based problem for smoothing the curve.

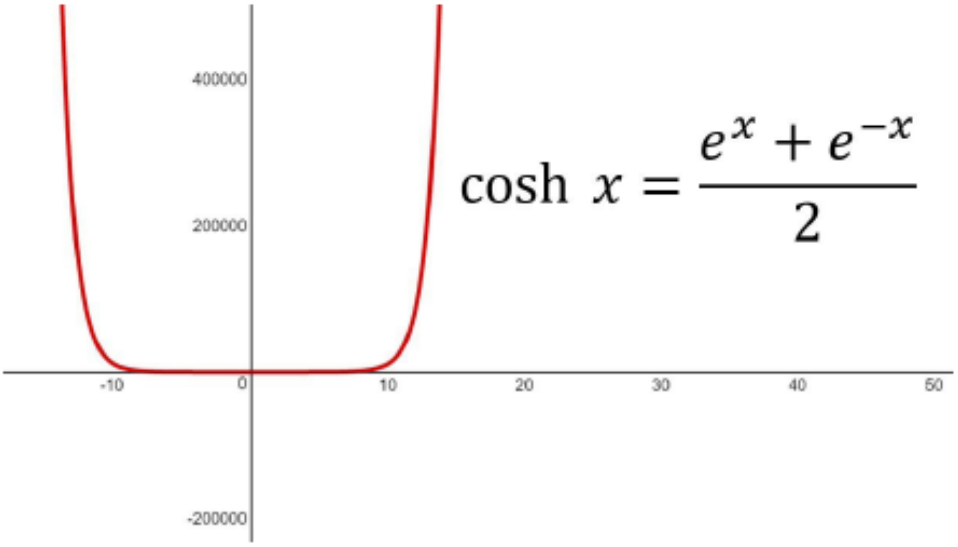


Figure 2-6 Cosh(x) function is the average of e^{X} and e^{--X}

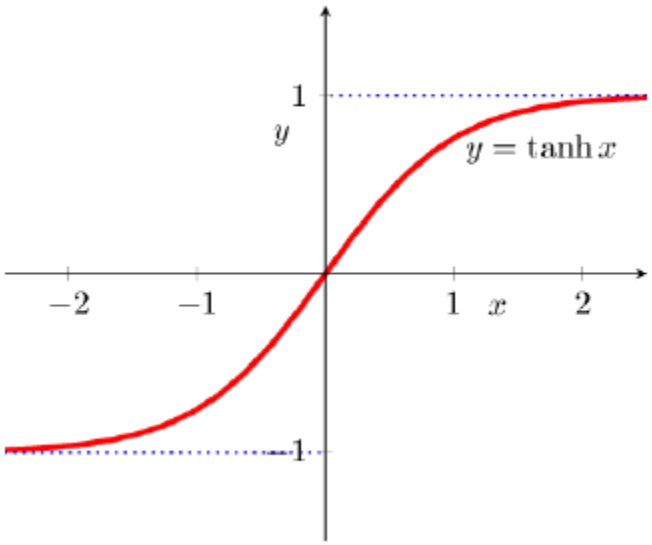


Figure 2-7 tanh(x) function is continuous and finite. It ranges from [-1; 1]

Hyperbolic functions have been used by deep learning community in terms of non-linearities such as $tanhx$ layer. They are tractable as well as easily differentiable. $\cosh(x)$ is defined as

$$coshx = \frac{e^x + e^{-x}}{2}$$

And

$$cosh'x = \frac{e^x - e^{-x}}{2} = sinhx$$

but, at present $coshx$ range can go up to infinity. So, to capture it in range, log space is used, making the $log - cosh$ function to be:

$$L(x) = log(coshx)$$

and using chain rule

$$L'(x) = \frac{sinhx}{coshx} = tanhx$$

which is continuous and finite in nature, as $tanhx$ ranges from[-1 ; 1]. On basis of above proof which showcased that Log of $cosh$ function will remain continuous and finite after first order differentiation. We are proposing $log - \cosh$ Dice Loss function for its tractable nature while encapsulating the features of dice coefficient. It can be defined as:

$$L_{lc-dce} = log(cosh(DiceLoss))$$

- **Boundary-based Loss**

One variant of the boundary loss is applied to tasks with highly unbalanced segmentations. This loss's form is that of a distance metric on space contours and not regions. In this manner, it tackles the problem posed by regional losses for highly imbalanced segmentation tasks.

$$Dist(\partial G, \partial S) = \int_{\partial G} \left|\left|y\partial s(p) - p\right|\right|^2 dp$$

Boundary-based loss, a recent new type of loss function, aims to minimize the distance between ground truth and predicted segmentation. Usually, to make the training more robust, boundary-based loss functions are used with region-based loss.

- **Shape-aware Loss**

Shape-aware loss [153], [154] as the name suggests takes shape into account. Generally, all loss functions work at pixel level, however, Shape-aware loss calculates the average point to curve Euclidean distance among points around curve of predicted segmentation to the ground truth and use it as coefficient to cross-entropy loss function. It is defined as follows:

$$E_i = D(C\text{^} , C_{GT})$$

$$L_{shape-aware} = -\sum_i CE(y, y\text{^}) - \sum iE_i CE(y, y\text{^})$$

Using E_i the network learns to produce a prediction masks similar to the training shapes.

- **Hausdorff Distance Loss**

Hausdorff Distance (HD) is a metric used by segmentation approaches to track the performance of a model. It is defined as:

$$d(X, Y) = max_{x\in} X\ min_{y\in} Y\ ||x - y||2$$

The objective of any segmentation model is to maximize the Hausdorff Distance [155], [156], but due to its non-convex nature, it is not widely used as loss function. Karimi et al. [157] has proposed 3 variants of Hausdorff Distance based loss functions which incorporates the metric use case and ensures that the loss function is tractable. These 3 variants are designed on basis of how we can use Hausdorff Distance as part of loss

function: (i) taking max of all HD errors, (ii) minimum of all errors obtained by placing a circular structure of radius r, and (iii) max of a convolutional kernel placed on top of missing segmented pixels.

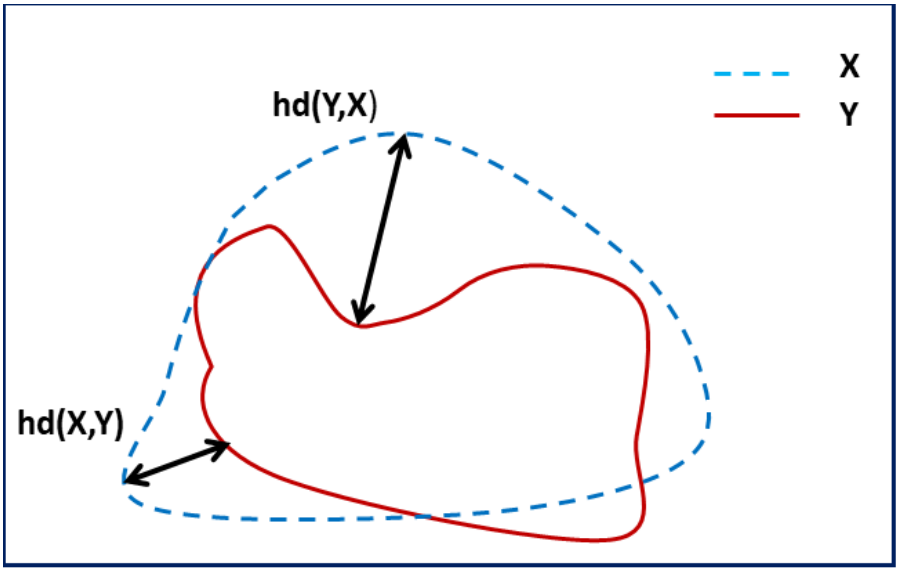


Figure 2-8 Hausdorff Distance between point sets X and Y [157]

➢ Compounded Loss

• Combo Loss

Combo loss [158]is defined as a weighted sum of Dice loss and modified cross entropy. It attempts to leverage the flexibility of Dice loss of class imbalance and at same time use cross-entropy for curve smoothing. It's defined as:

$$L_{m-bce} = -\frac{1}{N}\sum_{i} \beta(y - log(y\hat{}\,)) + (1-\beta)(1-y)log(1-y\hat{}\,)$$

$$CL(y, y\hat{}\,) = \propto L_{m-bce} - (1-\propto)DL(y, y\hat{}\,)$$

Here $\boldsymbol{DL}$ is Dice Loss

- **Exponential Logarithmic Loss**

Exponential Logarithmic loss [159] function focuses on less accurately predicted structures using combined formulation of Dice Loss and Cross Entropy loss. Wong et al. proposes to make exponential and logarithmic transforms to both Dice loss a cross entropy loss so as to incorporate benefits of finer decision boundaries and accurate data distribution. It is defined as:

$$L_{Exp} = w_{Dice} L_{Dice} + w_{cross} L_{cross}$$

Where

$$L_{Dice} = E(-ln(DC)^{\gamma Dice})$$

$$L_{cross} = E(w_j(-ln(p_j))^{\gamma cross}))$$

- **Correlation Maximized Structural Similarity Loss**

A lot of semantic based segmentation loss functions focus on classification error at pixel level while disregarding the pixel level structural information. Some other loss functions have attempted to add information using structural priors such as CRF, GANs, etc. In this loss functions, zhao et al. [160] have introduced a Structural Similarity Loss (SSL) to achieve a high positive linear correlation between the ground truth map and the predicted map. It's divided into 3 steps: Structure Comparison, Cross-Entropy weight coefficient determination, and mini-batch loss definition. As part of Structure comparison, the e coefficient has calculated which can measure the degree of linear correlation between ground truth and prediction:

$$e = \left| \frac{y - \mu_y + C_4}{\sigma_y + C_4} - \frac{p - \mu_p + C_4}{\sigma_p + C_4} \right|$$

Here, C_4 is stability factor set to 0.01 as an empirical observed value. $\mu_y \; and \; \sigma_y$ are the local mean and standard deviation of the ground truth y respectively. y locates at the center of the local region and p is the predicted probability. After calculating the degree of correlation, zhao et al. have used it as coefficient for cross entropy loss function, defined as:

$$f_{n,c} = 1 * e_{n,c} \; > \; \beta e_{max}$$

Using this coefficient function, we can define SSL loss as:

$$Loss_{ssl}(y_{n,c}\,, p_{n,c}) = e_{n,c} f_{n,c} L_{CE}(Y_{n,c}, P_{n,c})$$

and finally for mini-batch loss calculation, The SSL can be defined as:

$$L_{ssl} = \frac{1}{M} \sum_{n=1}^{N} \sum_{c=1}^{C} L_{ssl}(y_{n,c}\,, p_{n,c})$$

Where

$$M = \sum_{n=1}^{N} \sum_{c=1}^{C} f_{n,c}$$

Using above formula, loss function will automatically abandon those pixel level predictions, which doesn't show correlation in terms of structure.

Loss Function	Use Cases
Binary Cross-Entropy	Works best in equal data distribution among classes scenarios Bernoulli distribution based loss function
Weighted Cross-Entropy	Widely used with skewed dataset Weighs positive examples by $\mathbf{B}$ coefficient
Balanced Cross-Entropy	Similar to weighted-cross entropy, used widely with skewed dataset weighs both positive as well as negative examples by $\mathbf{B}$ and $1 - \mathbf{B}$ respectively
Focal Loss	works best with highly-imbalanced dataset down-weight the contribution of easy examples, enabling model to learn hard examples
Distance map derived loss penalty term	Variant of Cross-Entropy Used for hard-to-segment boundaries
Dice Loss	Inspired from Dice Coefficient, a metric to evaluate segmentation results. As Dice Coefficient is non-convex in nature, it has been modified to make it more tractable.
Sensitivity-Specificity Loss	Inspired from Sensitivity and Specificity metrics Used for cases where there is more focus on True Positives.
Tversky Loss	Variant of Dice Coefficient Add weight to False positives and False negatives.
Focal Tversky Loss	Variant of Tversky loss with focus on hard examples
Log-Cosh Dice Loss(ours)	Variant of Dice Loss and inspired regression log-cosh approach for smoothing Variations can be used for skewed dataset
Hausdorff Distance loss	Inspired by Hausdorff Distance metric used for evaluation of segmentation Loss tackle the non-convex nature of Distance metric by adding some variations
Shape aware loss	Variation of cross-entropy loss by adding a shape based coefficient used in cases of hard-to-segment boundaries.
Combo Loss	Combination of Dice Loss and Binary Cross-Entropy used for lightly class imbalanced by leveraging benefits of BCE and Dice Loss
Exponential Logarithmic Loss	Combined function of Dice Loss and Binary Cross-Entropy Focuses on less accurately predicted cases
Correlation Maximized Structural Similarity Loss	Focuses on Segmentation Structure. Used in cases of structural importance such as medical images.

Table 2-3 Loss functions and description

2.3.4 Regularization techniques

Regularization can be defined as any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. This regularization is often done by putting some extra constraints on a machine learning model, such as adding restrictions on the parameter values or by adding extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values. If chosen correctly these can lead to a reduced testing error.

- **L2 and L1 regularization**

L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

Cost function = Loss (say, binary cross entropy) + Regularization term

Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

However, this regularization term differs in L1 and L2.

In L2, we have:

$$Cost\ function = Loss + \frac{\Lambda}{2m} * \sum ||w||^2$$

Here**, lambda** is the regularization parameter. It is the hyperparameter whose value is optimized for better results. L2 regularization is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero).

In L1, we have:

$$Cost\ function = Loss + \frac{\Lambda}{2m} * \sum ||w||$$

In this, we penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here. Hence, it is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it.

- **Dropout**

This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.

To understand dropout, let's say our neural network structure is akin to the one shown below: So what does dropout do? At each iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections. So each iteration has a different set of nodes and this result in a different set of outputs. It can also be thought of as an ensemble technique in machine learning.

This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function. Dropout can be applied to both the hidden layers as well as the input layers. Dropout is a computationally inexpensive but powerful regularization method. Dropout provides an inexpensive approximation to training and evaluating a bagged ensemble of exponentially many neural networks.Another significant advantage of dropout is that it does not significantly limit the type of model or training procedure that can be used. It works well with nearly any model that uses a distributed representation and can be trained with stochastic gradient descent. Due to these reasons, dropout is usually preferred when we have a large neural network structure in order to introduce more randomness.

- **Data augmentation**

Dataset augmentation is a process of generating data artificially from the existing training data by doing minor changes like rotation, flips, adding blur to some pixels in the original image, or translations. Augmenting with more data will make it harder for the neural network to drive the training error to zero. By generating more data, the network will have a better chance of performing better on the test data. Depending on the task at hand, we might use all the augmentation techniques and generate more training data.

To apply data augmentation, we can make use of the existing methods present in the frameworks like Keras, PyTorch. In Keras, we can use ImageDataGenerator to augment or create more data by doing transformations, and similarly, we can use the transforms class present in torchvision from PyTorch to augment data.

Dataset Augmentation is a very popular approach for Computer vision tasks such as Image classification or object recognition as Images are high dimensional and include an enormous variety of factors of variation, many of which can be easily simulated. Operations like translating the training images a few pixels in each direction, rotating the image or scaling the image can often greatly improve generalization, even if the model has already been designed to be partially translation invariant by using the convolution and pooling techniques.

- **Early stopping**

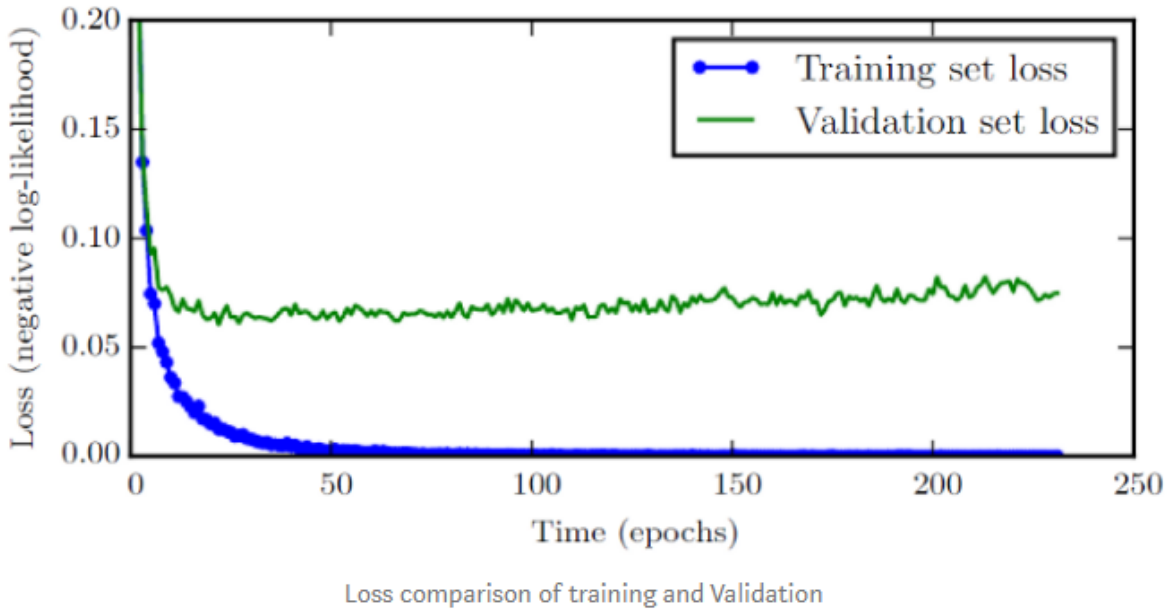
The idea behind early stopping is that when we're fitting a neural network on the training data and model is evaluated on the unseen data after each iteration. If the performance of the model on the validation data is not improving i.e…validation error is increasing or remaining the same for certain iterations, then there is no point in training the model further. This process of stopping model training before it reaches the lowest training

error is known as **early stopping**. Let's consider that we have set the patience of 5 epochs (i.e. the number of epochs to wait before early stop). For 5 epochs, we'll monitor the validation error, and if it isn't improving (either remains constant or increases) while the training error decreases, then we don't want to train any further. Figure 2-9

By using the early stopping technique, we're making sure that the model doesn't remember the patterns and noise present in the training data. Instead, we're pushing it towards generalizing the training data. Early stopping can be applied manually during the training process, or you can do even better by integrating these rules in your experiment through the hooks/callbacks provided in most common frameworks like Pytorch, Keras and TensorFlow. Figure 2-10

When training a large model on a sufficiently large dataset, if the training is done for a long amount of time rather than increasing the generalization capability of the model, it increases the overfitting. As in the training process, the training error keeps on reducing but after a certain point, the validation error starts to increase hence signifying that our model has started to overfit.

Figure 2-9 Training versus testing error with early stopping



Loss comparison of training and Validation

Figure 2-10 Training and Validation loss

One way to think of early stopping is as a very efficient hyperparameter selection algorithm. The idea of early stopping of training is that as soon as the validation error starts to increase we freeze the parameters and stop the training process. Or we can also store the copy of model parameters every time the error on the validation set improves and return these parameters when the training terminates rather than the latest parameters.

Early stopping has an advantage over weight decay that early stopping automatically determines the correct amount of regularization while weight decay requires many training experiments with different values of its hyperparameter.

- **Noise Robustness**

Noise is often introduced to the inputs as a dataset augmentation strategy. The addition of noise with infinitesimal variance at the input of the model is equivalent to

imposing a penalty on the norm of the weights. Noise injection is much more powerful than simply shrinking the parameters, especially when the noise is added to the hidden units.

Another way that noise has been used in the service of regularizing models is by adding it to the weights. This technique has been used primarily in the context of recurrent neural networks. This can be interpreted as a stochastic implementation of Bayesian inference over the weights.

- **Bagging**

Bagging or bootstrap aggregating is a technique for reducing generalization error by combining several models. The idea is to train several different models separately, then have all of the models vote on the output for test examples. This is an example of a general strategy in machine learning called model averaging. Techniques employing this strategy are known as ensemble methods. This is an efficient method as different models don't make the same types of errors.

2.3.5 Optimization techniques

Here are three of the most common used optimization functions (Stochastic Gradient Decent, Adagrad, Adam.

- **Gradient Descent**

Gradient descent calculates gradient for the whole dataset and updates values in direction opposite to the gradients until we find a local minima. Stochastic Gradient Descent performs a parameter update for each training example unlike normal Gradient Descent which performs only one update. Thus it is much faster. Gradient Decent algorithms can further be improved by tuning important parameters like momentum, learning rate etc. [161]

- **Adagrad**

Adagrad is more preferable for a sparse data set as it makes big updates for infrequent parameters and small updates for frequent parameters. It uses a different learning Rate for every parameter θ at a time step based on the past gradients which were computed for that parameter. Thus we do not need to manually tune the learning rate. [161]

- **Adam**

Adam stands for Adaptive Moment Estimation. It also calculates different learning rate. Adam works well in practice, is faster, and outperforms other techniques. [161].

Stochastic Gradient Decent was much faster than the other algorithms but the results produced were far from optimum. Both, Adagrad and Adam produced better results that SGD, but they were computationally extensive. Adam was slightly faster than Adagrad. Thus, while using a particular optimization function, one has to make a trade off between more computation power and more optimum results.

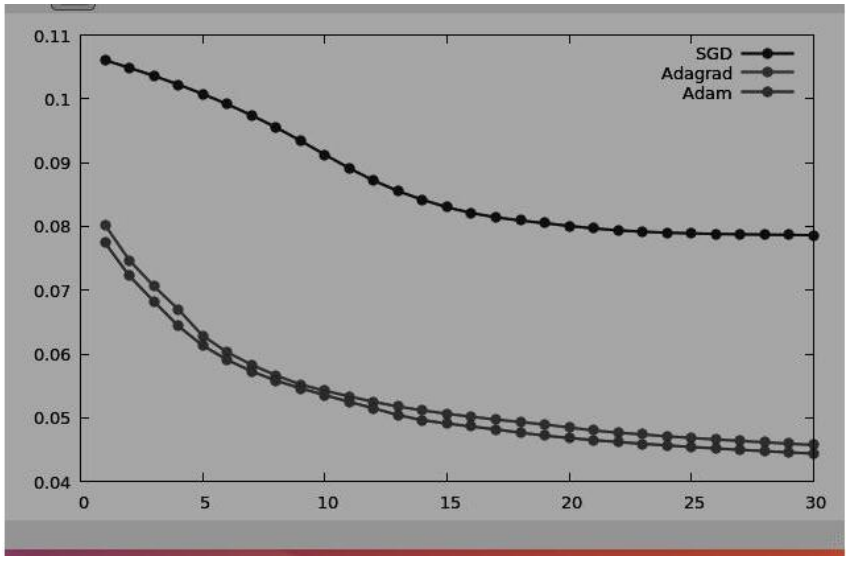


Figure 2-11 Perofmance comparasonfor SGD, Adagrad, Adam optemizers

2.3.6 Transfer learning in medical imaging

Transfer learning is essentially the use of pre-trained networks (typically on natural images) to try to work around the (perceived) requirement of large data sets for deep network training. Two transfer learning strategies were identified: (1) using a pre-trained network as a feature extractor and (2) fine-tuning a pre-trained network on medical data. The former strategy has the extra benefit of not requiring one to train a deep network at all, allowing the extracted features to be easily plugged in to existing image analysis pipelines. Both strategies are popular and have been widely applied. However, few papers perform a thorough investigation in which strategy gives the best result. The two papers that do, Antony et al. (2016)[162] and Kim et al. (2016)[163], offer conflicting results.

In the case of Antony et al. (2016), fine tuning clearly outperformed feature extraction, achieving 57.6% accuracy in multi-class grade assessment of knee osteoarthritis versus 53.4%. Kim et al. (2016), however, showed that using CNN as a feature extractor outperformed fine-tuning in cytopathology image classification accuracy (70.5% versus 69.1%). Two recent papers, published in high-ranking journals, which fine-tuned a pre-trained version of Google's Inception v3 architecture on medical data and achieved (near) human expert performance [113], [114]. Such results have not yet been achieved by simply using pre-trained networks as feature extractors.

In the more recent papers using CNNs the network architectures had been trained from scratch instead of using pre-trained networks. Menegola et al. (2016) [164]performed some experiments comparing training from scratch to fine-tuning of pre-trained networks and showed that fine-tuning worked better given a small data set of around a 1000 images of skin lesions. However, these experiments are too small scale to be able to draw any general conclusions from.

2.4 Liver segmentation

Liver detection and segmentation in medical images has been reported using CT, MRI and Ultrasound. In a liver medical diagnosis system, fully automatic methods are desirable, however, is a challenging task suffering several stages of development. Methods in the literature have focused on tumor segmentation in abdominal Dynamic Contrast-Enhanced MRI[165]. However, the most widely used method for liver diagnosis is CT, and hence, it is also the imaging modality with most prevalence the literature[166].

As an overview, the study of CT-based liver segmentation has been validated until the last decade using mainly statistical and atlas-based segmentation models and were in the past decade overcome by the higher performance presented by machine learning methods. Moreover, in the past decade, three liver and liver lesion segmentation workshop challenges were organized, boosting the interest of the scientific community towards this task.

Many research papers proposed methods for liver segmentation. Significant number of surveys concluded the approaches of liver segmentation into **1) Gray level based** (region growing, active contour, Graph cut, threshold based, clustering based) , **2) Statistical model**s (ASM, AAM), **3) Texture based** (Machine learning, Patern reconition) , **4) Other methods** (deformable model based methods, Probabilistic atlas based methods , level set based methods) [5]–[11].

The following subsections will review the different techniques for liver segmentation divided into General techniques, Machine learning based techniques, and Deep learning based techniques for liver segmentation.

2.4.1 General techniques for liver segmentation

In 2007 a CT liver segmentation challenge was organized for MICCAI 2007 conference. Until the past decade, the greatest contributions made in the field of whole liver segmentation were reported upon the MICCAI Sliver 2007 challenge, making available a

dataset of 40 contrast-enhanced CT images [11]. In this setting, the top ten methods proposed in the automatic segmentation category, included deformable models [167]–[171], Statistical shape models (SSMs) [172]–[174], level-set methods [175], [176], atlas-based methods [177]. The semi-automatic methods proposed by the top six papers in this category included graph-cuts segmentation [178], flood-filling segmentation [179], a level-set based deformable segmentation [180], radial basis function guided level-sets segmentation [181] and atlas-based segmentation[182] , further refined by manual interaction.

As could be expected, in the vast majority, the semi-automatic segmentation methods proposed at the time surpassed the performance of the automatic methods. The challenge evaluated the results by comparison to expert-generated references and using in a combined scoring method which evaluated a set deviation metrics to ground truths. The performance results of the MICCAI Sliver showed that, the top three best performing automatic segmentation methods in the challenge were all based on Statistical shape models. SSMs, primarily presented by Cootes et al[79], [89], [93], consisted in statically model prior shape data as a parametric set of equations that focus rather on the boundary of the region to be segmented instead of its internal voxels [81]. These types of algorithms became very popular in medical image segmentation, despite presenting lack of flexibility. To overcome this, Kainmuller et al. presented a combined SSM-constrained segmentation followed by a deformable mesh guided by a heuristic tissue classifier. The method included a thresholding initialization step and allies the free-form intensity dependent flexibility conferred by the deformable step, being the best automatic method presented in the challenge [173]. Semi-automatic methods based on user interactive refined graph-cut (GC) segmentation won the competition [178]. Graph partitioning methods consider the image to be segmented as a graph, composed of nodes representing image voxels, and edges connecting the neighboring nodes, weighted by a given dissimilarity rule. This graph

representation can be partitioned into connected components according to criteria describing the properties of the expected segments resulting in the segmentation of the image. The dissimilarity measure can incorporate gradient, intensity, texture or any other images features. GC methods use a minimum cost function between all possible cuts of a graph representation of the images, requiring background and object manual initialization.

In subsequent years, other methods have continuously been proposed in the literature and tested on the same MICCAI grand challenge dataset, allowing the direct performance comparison. Later, automatic segmentation methods were submitted to the challenge outperforming the abovementioned studies. SSM methods were continuously proposed with different modifications, such as combinations with free-form constraining [173],[183], level-set [184], graph-cut [185] methods, among others. Graph-based methods such as GC was proposed by many papaers in semi-automatic segmentation methods due to its ability to interactively edit the segmentation[186].

The level-set algorithm is another mathematical formulation of an iteratively evolving surface or contour. According to this technique the contour is represented using a signed function, i.e. the level-set function, where the zeros valued locations corresponds to the actual contour. The level-set function requires a contour initialization which then is computed, incorporating the contour propagation speed that is defined at each voxel. The advantage of this approach is that it can handle topological changes of the contour, but these methods can be time consuming and it is difficult to handle over-segmentation.
Image-based methods, besides level-set algorithms were proposed in semi-automatic methods. Ruskó et al. proposed an improved histogram-guided Region Growing (RG) method, obtaining competitive results in multiphase CT images [187].

More sophisticated SSMs such as Active Appearance Models (AAMs) have also been proposed in the literature. Transversal to SSMs and image and contour-based algorithms, initialization always stood as the biggest issue in the performance of these methods. Chen et al. proposed a novel combined AAM with a free-form segmentation based on Livewire and GC algorithms, for abdominal multi-organ pose estimation, initialization and segmentation [185]. The method was validated and compared with the dataset and results of the MICCAI Sliver challenge presenting similar performance and demonstrating less computational cost. Tomoshige et al. proposed a relaxed conditional SSM with conditional features error model, with a subsequent free deformation step demonstrating very positive results for automatic liver segmentation, on a private dataset of 144 non-contrast CT images, expressing performance as a Jaccard index valued 0.86 [188]. Another pertinent graph-based algorithm that became popular in medical image analysis is the Random Walker (RW) algorithm for segmentation, proposed by Grady (2006) [189]. Moghbel et al. tested the performance RW, after rib-cage removal using B-spline contouring, of on an image dataset composed of healthy and unhealthy livers, achieving a promising average performance expressed as a Dice similarity index (DSC) valued 0.94, and 0.91 for the MICCAI Sliver challenge.

In turn, another popular type of segmentation algorithm were atlas-based methods. Probabilistic atlases (PA) use shape priors and spatial relationship information. Atlas-based methods were also presented in the MICCAI Sliver challenge; however, their initial formulation did not outperform the previous model- and image- based methods, not being able to handle properly the very high variation that characterized the liver shape among different patients. Nonetheless, Okada et al. proposed a novel method combining the properties of both probabilistic atlases for initialization followed by an optimized SSMs fitting to the image intensities [190]. Lastly, Xu et al. proposed a registration based on dense 3D-scale invariant feature transform (SIFT) features to find correspondences between source

images and target atlas. The labels of the source image are later used to segment the target image, and the method was validated on the MICCAI Sliver challenge, with performances of 96% dice overlap [12]. In turn, Platero and Tobar developed a method combining the spatial normalization with the segmentation method based on standard CRF models to guide 19 atlases for liver segmentation, presenting high DSC valued 95%–97% for the MICCAI 2007 Grand Challenge [191].

In the following year, MICCAI 2008 conference launched the MICCAI 2008 Workshop on 3D Liver Tumor Segmentation Challenge, making available CT data from 30 liver tumors. At this time, research in medical image started to incorporate machine learning methods. Hence the proposed methods are composed by a mixed variety of algorithms. Similarly, to the previous challenge, semi-automatic methods outperformed automatic ones. In this competition, the top ranked method presented, consisted in a semi-automatic method combining graph-cuts with a watershed low-level segmentation[192]. Moreover, region-growing algorithms [193], thresholding only [194] and combined with filtering techniques[195] , intensity-based analysis [196], [197], deformable models[169], [198]. [5]

2.4.2 Machine Learning based methods for liver segmentation

Apart from the mentioned methods, novel segmentation techniques incorporating machine learning algorithms have gained attention by researchers. Machine learning, as in many other fields of Image analysis, has grown preponderantly, presenting superior results in a varied number of computational tasks. More recently, a sub-field of machine learning algorithms, denominated as deep learning, is the current state of the art set of algorithms that are being validated in the literature for signal and image processing tasks. Medical image analysis research field has also followed the deep learning trend and is currently applying these algorithms to a wide number of segmentation tasks. An overview of the key

first machine learning incorporations with standard methods, pure machine learning algorithms and more advanced deep learning methods applied to liver organ segmentation follow. Among these, data-driven methods for pixel labelling were widely used for target object segmentation. Goryawala et al. propose a 3D whole liver volume segmentation method, as a semi-automatic k-means clustering initialization followed by 3D RG algorithm, ensuring a parallel computational processing framework [199].

In the context of the MICCAI 2008 workshop challenge, learning-based algorithms were proposed based on Adaboost models[200] , Bayesian probabilistic methods and Support Vector Machines (SVM) voxel classification.

SVM belongs to the supervised learning methods that combine linear algorithms with linear or non-linear kernel functions SVM by finding the best generalizing hyperplane with maximal margin separating the two classes. The second best method proposed in the MICCAI 2008 competition was a semi-automatic method using supervised SVM voxel classification strategy in 2D slices, further propagated adjacent slices for tumor segmentation[201] . Other Bayesian learning algorithms [193] were also proposed.

In subsequent years also, many papers used the challenge dataset for method validation. Zhang et al. proposed a watershed liver initialization followed by SVM-based regional classification [202]. Sophisticated graph-based analyses of images have also been attempted in the literature. Wu et al. propose a supervoxels analysis, generated by the 2D simple linear iterative clustering, of images segmented with graph-cut algorithm [203]. Selver et al. developed a fully automated liver segmentations method that employs pre-processing for exclusion of neighboring structures, k-means clustering, and multilayer perceptron (MLP) for feature based boundary recognition, and post-processing for removing

miss-segmented objects and smoothing liver contours [204]. The method is validated in CT images with and without contrast media.

Another feature-based segmentation using SVM classification of textural features with combined morphological operations [205]. Hu et al. proposed a three-level AdaBoost-guided ASM to segment liver in CT images, including an Adaboost voxel labelling initialization, a profile classifier, refined by an ASM mesh model[13] . Zheng et al. propose a feature-learning-based random walk method for liver segmentation using CT images. A learning step consisting on Haar, HOG LBP, and GLCM features are learned by an Adaboost guided Support Vector Machines (SVM) model, which generates automatic seed points on the original test image, to carry the automatic segmentation step via Random Walks algorithm[206].

Markov Random Fields theory has also been applied to image segmentation. This algorithm considers that hidden node representing a label (e.g. object of interest, background, etc.) is assigned to each observation node. The method computes the hidden node configuration with the highest probability given the observation nodes and the built-in model.

2.4.3 Deep learning-based methods for liver segmentation

Several methods well established in the literature have been reported in the literature obtaining the best recently published automatic liver segmentation methods, as good performances as DSC of 0.96 [12], [13]. The evolution of deep learning methods has evolved from the first applications of these methods to signal processing. Hence, deep architectures range from Deep Belief Neural Networks (DBNs), Sparse Autoencoders (SAEs) and finally Convolutional Neural Networks (CNNs). First approaches were proposed using SAEs. The first method using deep neural networks for liver segmentation was proposed by Shin et al. in a multi-organ segmentation pipeline of Dynamic Contrast Enhanced Magnetic

Resonance Imaging (DCE-MRI) [206]. The models base their knowledge in previous applications of deep architectures to object recognition in natural images and propose an SAE neural network to separately learn 256 visual and temporal features in an unsupervised and automatic manner and detect multiple organs in a time series. The research seek to propose the first deep architecture procedure for medical image segmentation tasks, tackling the substantial shape changes that characterize the abdominal organs such as kidneys, liver, heart among others, hence using a data sets from two studies of liver metastases and one study of kidney metastases. The SAE was characterized by sparsity constraint application to logistic sigmoid activations, interleaved with max-pooling operations after each layer. A first step of tissue type labelling was performed, followed by tissue to separation of each organ recurring a parameter optimization using to context-specific feature manipulation, to identify each organ in a supervised manner, obtaining a classification accuracy of approximately 64.8% for the liver.

Lu et al. proposed a segmentation method combining a 3D CNN to obtain liver probability maps which would be further incorporated in the image information term of the energy functional of a GC algorithm. The papers validated the method on forty CT volumes, 20 taken from the MICCAI Sliver07 challenge and 20 from the 3Dircadb datasets. The papers use stacked convolutional and average pooling layers, trained by gradient-based backpropagation. The papers take advantage of the capability of GC method of handling loose boundaries between tissues with similar intensity distributions. The proposed method is the first 3D CNN application to liver segmentation, and obtained a score of 77,8 by the Sliver challenge evaluation method [207].

Hu et al. proposed an automatic 3D CNN, trained to output liver probability maps complemented with a globally optimized surface segmentation. The model used 42 CT

images from MICCAI Sliver challenge dataset obtaining an overall score of 80.3, surpassing any other methods previously proposed.

Christ et al. proposed a cascaded CNN in 2D with a 3D dense conditional random field (CRF) approach as a post-processing step, to achieve higher segmentation accuracy while preserving low computational cost and memory consumption [15].

Dou et al. presented a novel 3D CNN equipped with fully convolutional architecture and a 3D deep supervision mechanism to comprehensively address the challenges of volumetric medical image segmentation [15]. The deep supervision method, to which the authors name DSN, uses an objective function that guided the training of the upper and lower layer, propagating more efficiently the representation of the features, as well as speeding up the training process, by patch-based methods and volume-to-volume learning and inference. The CNN model consists in 6 convolutional layers, 2 max-pooling layers, and one softmax output layer layers finalized with 2 deconvolutional layers, trained via stochastic gradient descent (SGD), in a fully convolutional architecture with 3D kernels, outputting a spatially arranged classification across the whole input pixels. The resulting predictions are rougher and obtained in low-dimensional layers, which is thus solved by the deconvolutional final layers. Furthermore, the papers empathies the gradient instability that has been reported in previous works, where propagated 'fading', exploding of vanishing of gradient magnitudes occurs. A DSN mechanism is proposed in the network architecture by softmax extraction of outputs predictions from intermediate layers convolutional results, connecting them directly to the final output layer, and whose weighting is incorporated in the backpropagation objective function. Finally, the papers propose the incorporation of a fully connected CRF, which has advantage in capturing complicated shaped object such as those with holes or thin structures, was used for liver segmentation refinement. The results were evaluated on the MICCAI Sliver challenge dataset presenting superior Volume Overlap Error and computational cost.

From this point, the ISBI 2017 conference, released a challenge focusing on liver lesion segmentation, this time making available a total of 200 abdominal contrast-enhanced CT images. The dataset was obtained from several different clinical sites with different scanners and protocols, thus having largely varying spatial resolution and fields-of-view. The in-plane resolution ranges from 0.60 mm to 0.98 mm, and the slice spacing from 0.45 mm to 5.0 mm. The axial slices of all scans have an identical size of 512 × 512, but the number of slices in each scan differs greatly and varies between 42 and 1026. The following works presented, were developed under this scope of this challenge and will be reviewed in detail. Given the advances of these methods, it was only natural that they consisted in implementations using Convolutional Neural Network.

Chlebus et al. proposed a two-step segmentation method, initiated by a liver volume segmentation followed by a liver lesion candidate detection and classification [15]. In the first step, a coarse segmentation was obtained from an ensemble of three CNNs trained with each of the three orthogonal interpolated volumes, followed by a 3D CNN liver refinement. The axial, sagittal and coronal volumes trained three 2D U-net models which whose softmax outputs were used to train a final 3D U-net that originated a full mask volume of original resolution. Based on the liver mask ROI extraction the second step consists in a two-step tumor segmentation using a 2D CNN for tumor segmentation, tumor candidate detection via 3D connected component analysis of the mask volumes, followed by a Random Forest tumor candidate filtering. Similar to the first step, a 2D U-net was used for the tumor segmentation and were trained with class balanced boundary patches of the axial dataset, however, the paper hypothesize that the patching step used for training penalized the specificity of these segmentations, requiring thus, the incorporation of a 46 features extraction and classification step of these candidate tumor mask. The method scored a Dice coefficient of 0.65, and second place in the challenge, including a tumor candidate classification accuracy with the random forest approach of 90%.

Bi et al. propose a cascaded deep residual networks (ResNet) approach to segment the liver lesions [208]. As pre-processing this paper apply data augmentation strategies including random scaling, crops and flips. ResNet uses shortcut connections to avoid training degradation though deeper layers, whereas the optimal results are calculated by an averaging output of the different networks. The key feature of the architecture proposed is the cascading of parameter learning, made from learning the training data and from the previous iteration result outputs. The testing segmentations are further incorporated with multi-scale rescaling strategies whose outputs averaging produce the final predictions. Given the architecture depth of 20 hidden layers training the entire volumes would become computationally non-viable, so the authors randomly pick a balanced dataset of axial slices, of a total of 8802 images to train the neural network via stochastic gradient descent. The network is pretrained firstly on the ImageNet dataset for parameter fine-tuning, which is then further fine-tuned with the liver dataset.

2.5 Summary

Convolutional neural networks implemented fully convolutional layers to extract features from the input images by applying serious of convolution and pooling processes. The extracted features will enable the CNN to accomplish the task of classification or segmentation of the image with a significant accuracy. U-Net is sub type of CNN and considered as the-state-of-the-art for medical image segmentation. The reason why U-net provides accurate segmentation image is that it combines the details in the contraction path with the global information in the expansion path. At the bottom of contraction path, the output image with global information of the shrinking unit is very small, and the global information is fused with the previous layer of the contraction path after the convolution. But this fusion operation makes detail information and global information are both from the

contraction path, and the details have not yet been pooled as global information. Based on that behavior, the convolution kernel of the contraction path serves as the tasks of extracting details and global information at the same time and it may cause the redundancy of parameters, reduce the pertinence of convolution kernel, and reduce the efficiency of convolution kernel extraction feature. Another drawback for the U-Net structure is, the vast network structure slows down the training process. The training process getting results from the data set takes for a long time, so for some special medical images whose characteristics are not obvious enough (such as lymph nodes), people are more likely to adopt the traditional segmentation method, morphological operation. If the training speed can be speeded up and the network precision can be improved, the method of deep learning will be more widely used in image segmentation. [209]

Segmentation or contouring processes are usually affected by the edge of the object. Despite the skip connection in the conventional U-Net more effectively handling edge information, there are still some drawbacks of the U-Net. First, the U-Net architecture duplicates low resolution information (low frequency components) of features. After pooling (down sampling), low resolution information of features pass on to the convolution layer in the next stage. However, this low resolution information of features is transferred by the skip connection of the U-Net as well. Duplication of low resolution information may then cause smoothing of the object boundary information in the network, which is more serious in the case of fuzzy object boundaries. Another drawback of the U-Net architecture is that it may not sufficiently estimate high level features for high resolution edge information of the input object. The U-Net can use the skip connection to transfer high resolution information. However, unlike low resolution features after pooling, high resolution edge information does not pass through any convolution layers during transfer by the skip connection. Thus, higher level feature maps learned by the network do not contain enough of the input object's high resolution edge information. Thus, in the conventional U-Net, high level features are extracted disproportionately from low resolution information. [210]

The literature shows significant number of varieties of the models based on U-Net structure with many modifications had been implemented for medical image segmentation while a few of them focused on liver and liver tumors.

Since U-Net model initially introduced on 2015, significant number of researches inherited the U-Net structure with different kind of modifications in order to achieve high segmentation accuracy using specific datasets. stacked U-Net, V-Net, Bridged 2U-Net are examples of the modifications on the U-Net main structure. Zhang et al., 2018 added a separated path to extract the global features and detail features separately with changing the number of convolutional channels of the original contraction path, the original expansion path and the new path is reduced. These two modifications make the training more rapid and improve the efficiency of the convolution kernel extraction feature. [209]. While the adjacent network with less number of parameters speeded up the training process it has a limited accuracy [211], Integrating U-Net with other traditional registration and segmentation techniques such as Conditional Random Field segmented the liver tumor with limited number of samples[212] [213]. Cascading U-Net [214] and cascaded U-Net with Residual mapping [215] is another form of U-Net modification by replacing the convolutional layers with a complete residual block or layers that successfully segment the liver and liver tumor on a two cascaded phases. [216]–[219]. UNet++ is a successful model for integrating the skip connections to form a basic style of deep supervision approach [220].

Based on the analysis of the mentioned approaches in the literature, the modifications on U-Net model generated based on the specifications of the research problem and no specific detailed study had been conducted to compare between different type of modifications on U-Net in terms of the number of levels on each path of the U-Net, the number of filters that can be applied on each level, the effect of stacking multiple U-Net , the effect of changing

the skip connections between the two paths of the U-Net and the bridge connections between stacked U-Nets.

Although Dice similarity coefficient (DSC) proved to be one of the best loss function and metrics for segmentation accuracy, we decided to integrate the proposed distance between centers of mass (Centroid) with two types of deep-supervision techniques to add more constrained on the false positive output of the liver segmentation solve the flipping issue because DSC couldn't solve it. The combination of the weighted loss functions (DSC and Centroid) with De-convolutional deep-supervision and multi-resolution masks deep-supervision is a novel approach to enhance the accuracy of liver segmentation. As far as we know from the literature, DSC, weighted DSC, and Deep-supervision implemented separately in different researches and achieved good accuracy for specific segmentation task.

Chapter 3
Research Methodology

The research methodology designed to answer the main research questions stated in the introduction and overcome the issues, problems, and difficulties that had been raised during the research. The research questions concluded in investigating the effect of increasing the depth, width of the U-Net on the performance, investigate different connections between different model components. Overcome the issues of shortage in medical images datasets available for research and the flipping issue that appeared because of implementing specific augmentation techniques.

The methodology consists of four main modules, "**Model design**", data **pre-processing**, **training the model**, and finally the **testing process**. Figure 3-1 shows the modules and the processes in each module and the relationships between different modules and processes. The main contributions represented in the highlighted (Blue color) processes.

Model Design

Model design is the first module of the methodology where the CNN model designed with full details to accomplish the liver segmentation task. Designing the model contains 6 sub-processes based on the generated model for each task. **1- 'Model Structure'**, **2- 'Model Depth'**, **3-'Model Width', 4- 'Skip Connections', 5- 'Loss function design', and 6- 'Model parameters'.**

All the six processes within this module define part of the contribution of the research.

In **Model structure** process, the base model structure is chosen. U-Net and 2 Bridged U-Net models are chosen to be the base models based on the investigated question, in addition to integrate the concept of deep supervision and multiple outputs with the bridged U-Nets. Image sizes, number of layers in each level, number of filters are hyper parameters that define the U-Net based models. The second process is '**Model Depth'** where number of levels in the contracting and expansion paths of the U-Net and the type of layers in each model's level e.g. (Conv, maxpooling, dropout…etc.) will vary to generate different models for accuracy investigation, while '**Model width'** process shows how many U-Net will be stacked to construct the wider model to test the accuracy. **Skip connections** process where different skip connections between different levels and layers in each U-Net and between different U-Nets are designed and investigated to generate various models in order to recommend the best setting for U-Net based models. **Design the loss function** is a critical process that significantly affect the overall model performance. Integrating DSC with newly implemented centroid loss and applied weight factors on deep supervision style model represents one of the major contributions of the research to solve the flipping issue. **Model parameters' stetting** process plays a crucial role that affects the overall model performance. The model parameters include but not limited to filter size, number of filters applied at each layer and the percentage of increasing or decreasing between different levels, and the stride size, maxpooling size, dropout percentage, learning rate and it's decay, the activation function, and the optimizer selection.

Pre-Processing

This module focus on preparing the data samples to be suitable for training and testing CNN based mode. Because CNN models need significant number of samples for training and validation while the available dataset is very limited, data augmentation

techniques are used to significantly increase the number of data samples. Deciding the splitting percentage of the data for training, validation, and testing in addition to choosing the samples in each group can affect the performance, for example the training samples should contain most of the variations in the data. A normalization technique is implemented to keep the DICOM pixel data values within a suitable integer range for CNN model.

Model Training

Training the model contains 3 sub-processes, 1- uploading the training and validation data, 2- Setting the model and process parameters, 3- training process. **The main contribution in this module is defining the model parameters (2)**. Deciding the values of the model hyper parameters had significantly affected the model performance. Examples for the model's training parameters are learning rate, number of epochs, image size, loading batch size…etc.

Model Testing

The testing module contains only two processes, loading the testing sample, and testing and recording the results. Using augmented data for model training is common for classification task [221][222] and rarely used for image segmentation [223] because the classification needs to be applied on real world data e.g. handwriting numbers can't be found flipped or aggressively rotated otherwise it can give wrong prediction for 6 and 9. While all the models included in this research trained, validated on both original and augmented data, **testing the CNN model using both original and augmented data for medical images could be an addition contribution**. The same augmentation techniques applied to the testing samples. The tested models recorded very close accuracy on both original and augmented data.

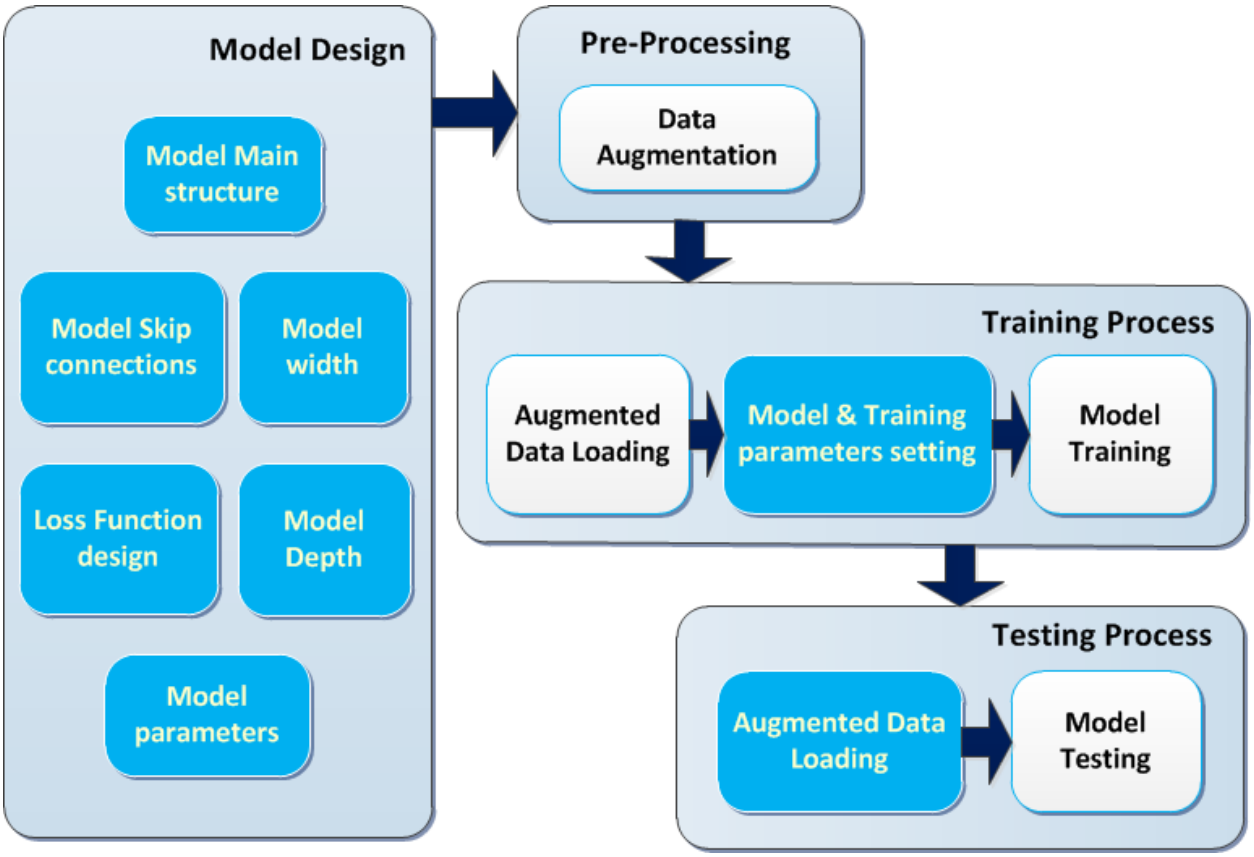


Figure 3-1 Methodology modules and processes in each module and the relationships between different models and processes. The highlighted processes represent the research contributions

3.1 Datasets

3Dircadb1(**3D** **I**mage **R**econstruction for **C**omparison of **A**lgorithm **D**atabase) is created by Hôpitaux Universitaires France as a public dataset for researchers in medical image segmentation. The dataset composed of 3D CT-scans for 20 patients with hepatic tumors in 75% of cases. For each patient there are number of CT scans in addition to manually annotated mask for several structures of interests e.g. liver, left kidney, right kidney, and hepatic tumors performed by clinical experts. All Ct scans and masks are in DICOM format with pixel size (512*512). The total number of CT scans are 2823.[224]. The CT scans of 20 patients divided into 14 patients (P6 to P19) for training, 4 patients (P1 to P4) for validation , and 2 patients (P5 and P20) testing. Figure 3-2. Excluding 13% of the

total samples used for testing, the remaining samples divided into training and validation with percentage 75% and 25%.

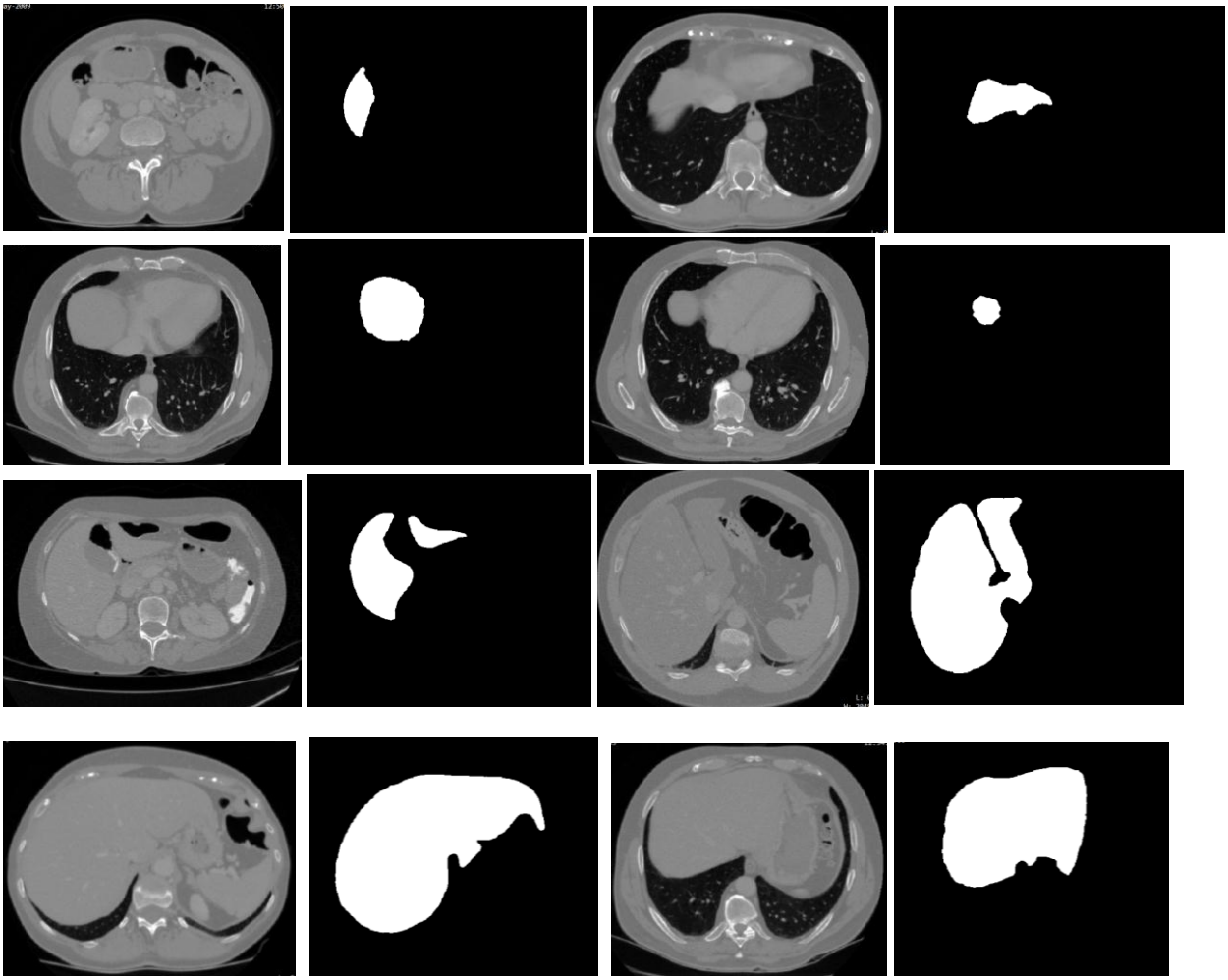


Figure 3-2 Examples of 3Dircadb1 dataset, DICOM image and associated mask shows a lot of variations in the liver shape even for the same patient

IKN (Institut Kanser Negara - Malaysia)

IKN is the National cancer institute, Negara, Malaysia. IKN is governmental hospital and research center for cancer treatment. The collaboration with IKN supports the research with enough IKN data for testing the model after signing of a confidential data agreement.

The data contains CT scans for 15 patients with liver or kidney tumors. The CT scans for each patient obtained before, during and after the treatment process. The dataset

consists of 26,948 images in DICOM format. Because the dataset doesn't have any annotated mask for any organ, it can't be used for model training. All images contained in this dataset would be used for testing the trained model. During the treatment and scanning process, the patient position changed from lying on left or right side to face up or face down depending on the treatment procedure and the tumor position. Based on the position changing, the liver position and shape is different from patient to another and between each group of images. Figure 3-3

Figure 3-3 Examples for IKN dataset, the patient position from left to right is face up, face down, right side 45 dgree, right side 90 degree

3.2 Hardware/Software framework

3.2.1 Hardware

Intel® Core™ i7-6700 CPU @ 3.40GHz × 8, with 16GB RAM and GPU GeForce GTX 1080/PCIe/SSE2 with 8GB RAM. Storage space of 500GB is needed to store the data samples used for training and validation and testing results.

3.2.2 Software framework

The software framework composed of the following packages, CUDA v9.0, anaconda 1.6.0, conda 4.3.25, python 2.7.3, TensorFlow 1.2.1, Theano 0.8.2, Keras 2.1.1. In addition to the needed libraries for the implementation, e.g. openCV ,numpy, pydicom, skimage, matplotlip, PIL, pandas, and keras (model, layers, optimizers..etc). Ubuntu 14.04.05 is the used operating system. Figure 3-4

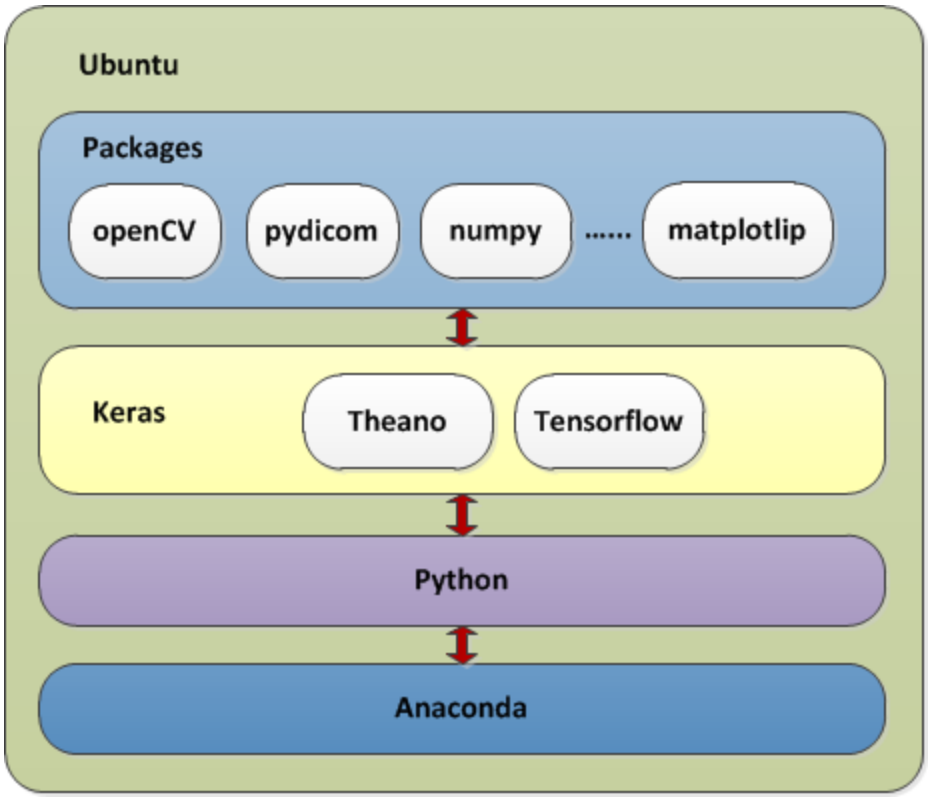


Figure 3-4 Software framework

3.3 Data augmentation

There is significant number of augmentation techniques used to increase the number of data samples[221]. Only two augmentation techniques implemented in this research, horizontal and vertical flipping and rotation with 15 degrees step because CT scans in real situations recorded while patients lies on left or right side or rotated with some angels to make sure of easy and shortest pass to access the tumors locations. As a result of applying the augmentation techniques, the total number of CT scans increased from 2,823 to 112,920. Training, validation, and testing samples became 74,680 (14 patients), 23,680(4 patients), and 14,560 (2 patients) respectively. Figure 3-5. The testing samples represent 13% of the total samples of 3Dircadb1, while the remaining samples divided into training and validation with percentage 75% and 25%.

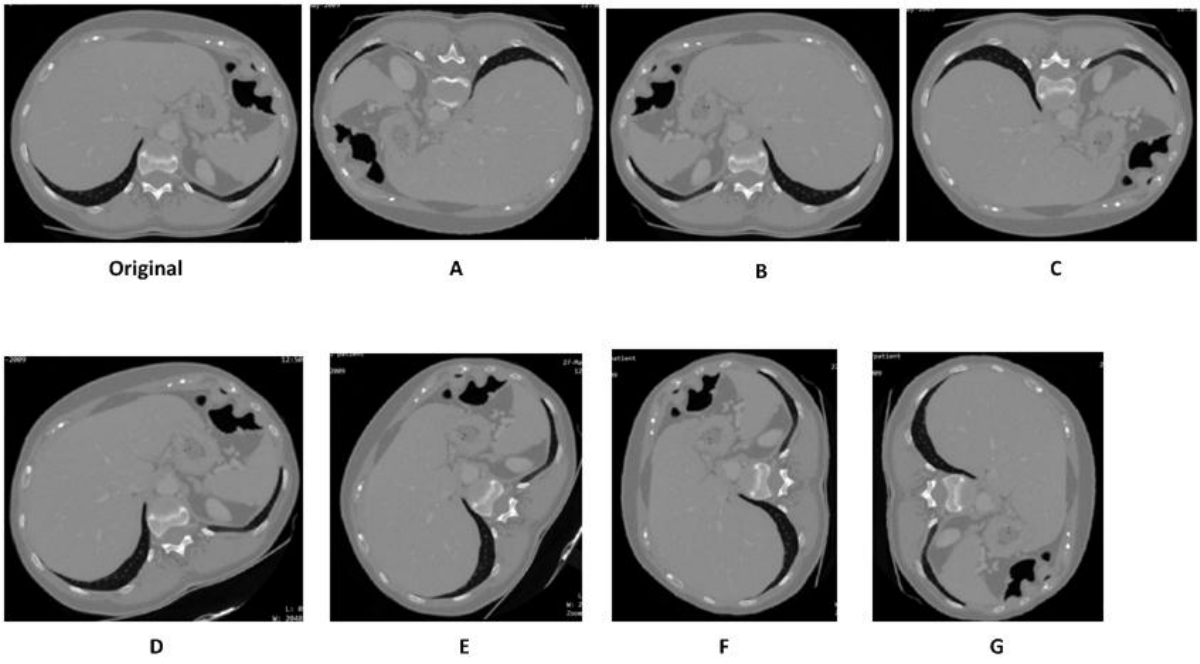


Figure 3-5 Augmentation techniques examples, A- Rotation 180°, B-Horizontal Flip, C- Vertical Flip, D- Rotation 45 °, E- Rotation 60 °, Rotation 90 °, G- Rotation 270 °

3.4 Baseline Models

In this piece of work, U-Net as the state-of-the-art for medical image segmentation is selected to be the main model for most of the experiments. The u-net is convolutional network architecture for fast and precise segmentation of images. Up to now it has outperformed the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neural structures in electron microscopic stacks [225].It has won the Grand challenge for Computer-Automated detection of caries in bitewing radiography at ISBI 2015 [226], and it had won the cell tracking challenge ISBI 2015 on the two most challenging transmitted light microscopy categories (Phase contrast and DIC microscopy) by a large margin [227]. U-Net benefits from a superior design of skip connections between different stages of the network. The most important property of U-Net is the shortcut connections between the layers of equal resolution in analysis path to

expansion path. These connections provide essential high-resolution features to the deconvolution layers. U-Net became one of state-of-the-art for medical image segmentation since it had been introduced by Ronneberger et al on 2015. Significant number of papers used U-Net model with some modifications to achieve specific tasks in medical image segmentation.

The second base model is 2Bridged U-Net, where it overcome three problems in deep learning based medical image segmentation. Firstly, U-net, as a popular model for medical image segmentation, is difficult to train when convolutional layers increase even though a deeper network usually has a better generalization ability because of more learnable parameters. Secondly, the exponential ReLU (ELU), as an alternative of ReLU, is not much different from ReLU when the network of interest gets deep. Thirdly, the Dice loss, as one of the pervasive loss functions for medical image segmentation, is not effective when the prediction is close to ground truth and will cause oscillation during training. [16]

During the testing phase of U-Net and 2Bridged U-Net, the flipping issue appeared to show that, some context features are not correctly captured through the convolutional processes. The results show the flipping issue that segment two polygons as liver on both image sides while it supposed to be only on object according to the ground truth. To solve this issue, we need to decrease the distance between the centers of two polygons to remove one of them. We proposed a solution that combines two parts. The first part is the proposed centroid loss function that minimizes the distance between the center of mass between the two polygons. The second part is integrating deep-supervision approach to the model. Deep-supervision will add extra loss functions (weighted loss function of DSC and Centroid) at each level of the proposed model. The extra loss functions will add more constrains on the deeper levels of the model to minimize the loss. The total loss of the model is the weighted sum for the loss over all the model levels.

All the proposed models in the research are designed based on two main models that are considered as the state-of-the-art for medical image segmentation. The first model is U-Net [14] using CNN in medical image segmentation. U-Net structure will be used to generate multiple models with variations in the number of layers and levels to investigate the effect of adding more layers to the U-net (meaning Deeper U-Net based models). The U-Net model with the original structure of 5 levels with number of filters 64-1024 will be used to explore the effect of stacking more than one U-Net together to for wider models.

The second model is based on two bridged U-Net and consists of two U-Nets connected with two bridged connections for prostate segmentation.[16]. The model will be used to generate different models based on two and three stacked U-Nets with different bridge and skip connections between the U-Nets; the generated models will be investigated to explore the effect of different connections on the model's performance and design a model with better accuracy. The structure of U-Net and Bridge net models is explained in full details in the following sections.

3.4.1 U-net

U-Net is considered as a state-of-the-art for medical image segmentation, initially proposed by Ronneberger et al[14] using the concept of de-convolution. This model is built upon the elegant architecture of fully convolutional network (FCN). U-Net benefits from a superior design of skip connections between different stages of the network. The model consists of two paths with the same number of levels. The **contracting path** consists of 4 levels; each level consists of 2 convolution layers that apply 64 filters with size 3*3 without any stride or padding followed by pooling layer using the maxpooling process with size (2*2) to decrease the feature maps size by 50%. The last process at each level is applying dropout with size (0.5) to avoid the over-fitting and increase the regularization. The number

of filters applied on each level will be doubled from level to the next level .e.g. starting with 64 filters at the first level then became 128, 256, and 512 for the next levels. At Level 5, 1024 filters with size 3*3 applied for two layers of convolution without any pooling or dropout at this level. The **expansion path** consists of 4 levels, an up-sampling process using stride with size (2*2) would be applied on the output feature maps from the previous level to increase the size of the feature maps, and then a concatenation operation will concatenate the up-sampled feature maps with the output feature maps from the same level at the contracting path before applying de-convolutional process using 3*3 filters with the same number as the number of filters applied at the same level on the contracting path then dropout layer with size (0.4) is applied for regularization, e.g. from level 4 to the topmost level 1, the number of filters applied are 512, 256, 128, 64. After all the up-sampling and de-convolution, the output size will be the same as the input image. Last layer for the whole model is apply convolution with size (1*1) using one filter and sigmoid activation function to classify the out pixels to one of 2 classes to define if the pixel is part of the mask or not. The following are some customized setting for the hyper-parameters for this study and different from the original U-Net paper. Rectified Linear Unit (ReLu) is the used activation function for all layers except the last layer. The model used learning rate ($1e^{-5}$) and (adam) optimizer with DSC loss function. Detailed model structure included in the appendices 6.1. Figure 3-6

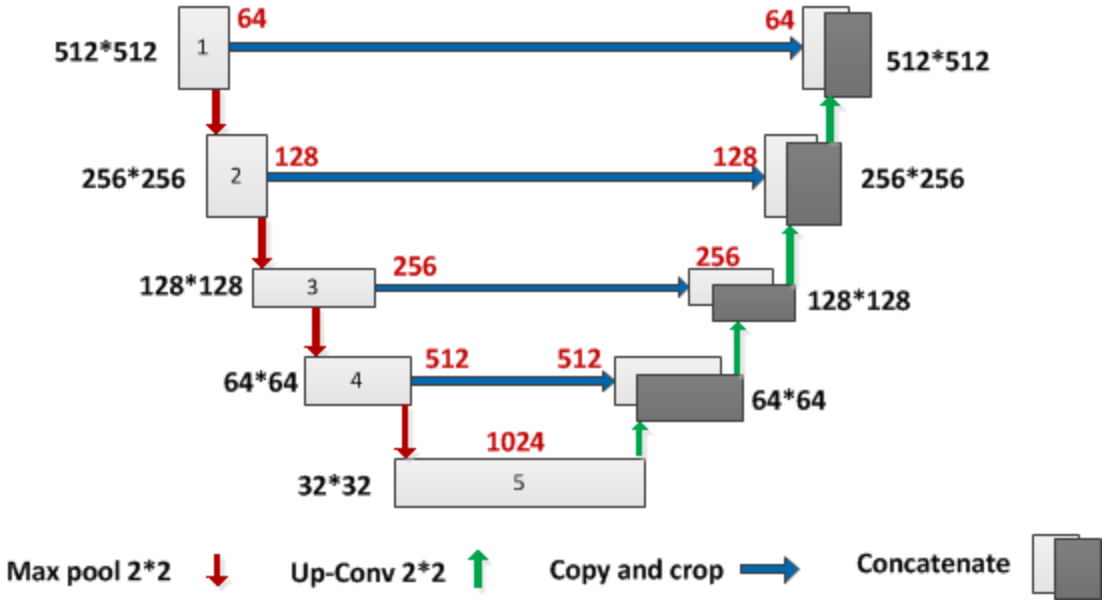


Figure 3-6 U-Net model structure for medical image segmentation. Number of filters (red) from 64, 128, 256, 512, 1024. Feature maps size (black) outside starting with 512*512 and ends to 32*32.

3.4.2 Bridge-Net

2Bridged U-Net model consists of two U-Net connected by two types of bridged connections in addition to the skip connections between the contracting and expansion path of each U-Net. B1 and B2 represent the two paths of the first U-Net while B3 and B4 represent the two paths of the second U-Net. The gray connections represent the skip connections between each two layers at the same level in each U-Net. The first bridge connection adds the output feature maps from B2 to the inputs of branch 3 (yellow), while the second bridge connection concatenates the output from B1 to the input of B4 (red). Figure 3-7 Figure 3-8 [16]

Figure 3-7 W net with bridge and skip connections model structure

The detailed structure for 2 bridged U-Nets model showed in Figure 3-8. B1 and B2 connected to create the first U-Net and the same with B3 and B4 for second U-Net. Branches 1 and 3 consisted of 4 levels of (convolution → maxpooling) layers using number of filters that increased by 200% from level 1 onward (32-46-128-256) (red), while branches 2 and 4 compound of 4 levels of layers (upsampling → De-Convolution) with decreasing number of filters applied from level 4 to level 1 to reach the original image size at the output. Level 5 in each U-Net contains a convolutional layer with 512 filters for flattened images. The original image size is 256*256 and will be decreased by 50% after each max pool layer in the contracting path to reach the minimum of 16*16 pixels. After each de-convolution and up-sampling layer on the expansion path, the feature maps size is increased by 200% to reach the original input size at the final output. The bridge and skip connections copy and concatenate high resolution feature maps between layers at the same level that may be lost during the convolution process. The first skip connection between the contracting and expansion path within each of the two U-Nets (Gray), the second connection is a bridge between the output of the expansion path of the first U-Net (B2) to the inputs of contracting path of the second U-Net(B3) (Yellow). The output from the contracting path of the first U-Net (B1) is concatenated to the inputs of the expansion path of the second U-Net (B4) (Blue). Each level at B2 concatenate the output feature maps from the same level at the contracting path in the same U-Net (Gray) in addition to the output feature maps from the previous level at the same expansion path (Green) then apply the Up-Sampling and De-Convolution. Each level at (B3) concatenate the output feature maps from the previous level at the same contracting path (Red) with the output feature maps from the same level at the expansion path of the first U-Net B2 through the bridge connection (Yellow), each level at the expansion path of the second U-Net (B4) concatenate the output feature maps from the same level at the contracting path at the same U-Net B3 (Gray) with the feature maps from the previous level at the same expansion path at second U-Net B4 (Red) in addition to the output feature maps from the same level at the

contracting path at the first U-Net B1 (Blue) before applying the Up-Sampling and De-Convolutional process Figure 3-8. To accelerate the convergence of neural network, the model proposed the gray block represents an ELU (Exponential Linear Unit) cluster (2conv-BN-ELU blocks), and the yellow block represents a ReLU (Rectified Linear Unit) cluster (2 conv-BN-ReLU blocks) where BN stands for Batch Normalization process. The yellow lines represent network bridging. The blue lines represents skip connections. [16]. Figure 3-8 Exponential liner unit (ELU) replaces the negative part in ReLU with exponential function, which is helpful to make the average of output close to zero. Neural networks usually suffer low coverage rate because of vanishing gradient, especially for deep network. ELU provides a buffer in negative axis so that it will not saturate immediately. However, ELU still suffers the saturation problem when network gets deeper. A combination of ReLU and ELU improved the segmentation performance where only the deeper layers at level 4 and 5 used ReLu and the top level layers used ELU. After all the up-sampling and de-convolution, the output size will be the same as the input image. Last layer for the whole model is apply convolution with size (1*1) using one filter and sigmoid activation function to classify the out pixels to one of 2 classes to define if the pixel is part of the mask or not. The following are some customized setting for the hyper-parameters for this study and different from the original U-Net paper. The model used learning rate (1e-5) and (adam) optimizer with DSC loss function. Detailed model structure included in the appendices section 6.2

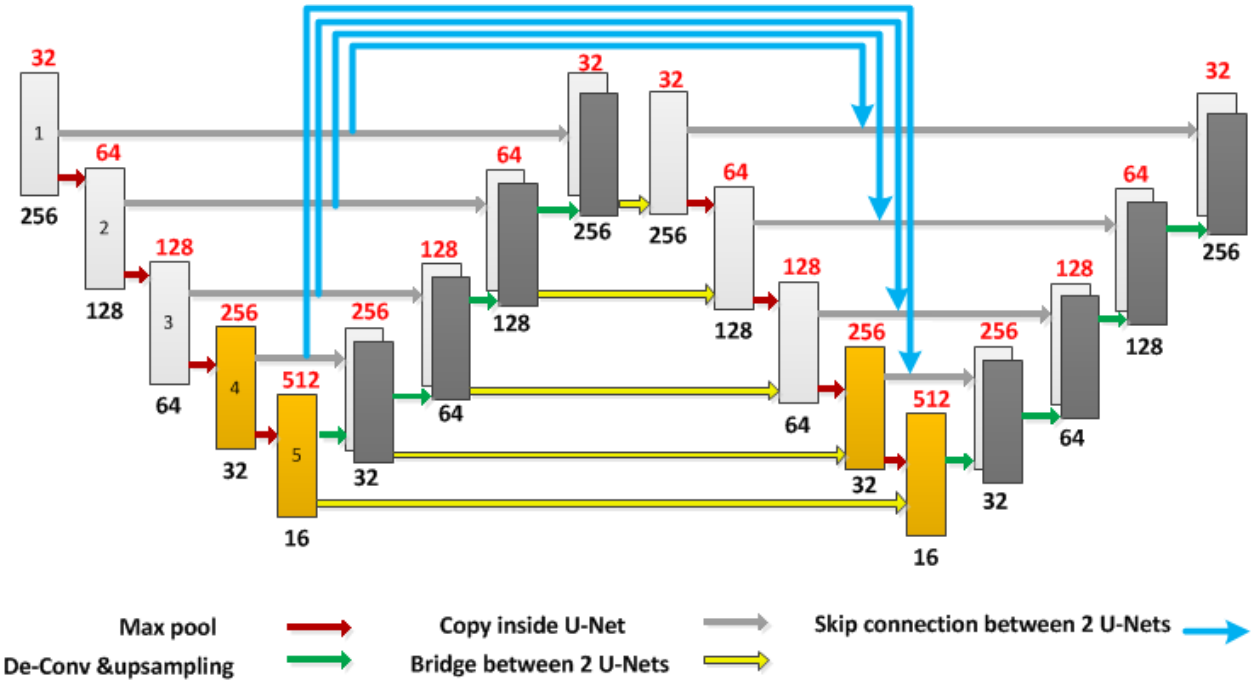


Figure 3-8 Bridged 2U-Net model

3.5 Evaluation metrics

Two categories of evaluation methods would be applied for most of the models in the study.

Quantitative evaluation

The accuracy for Training, evaluation and testing would be measured based on Dice Similarity Coefficient (DSC) that measure bitwise similarity between the ground truth and the prediction in terms of the intersection between the prediction and the ground truth divided by the absolute value of the sum of them

$$DSC = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

Where $\cap$ the intersection between two sets of points, X and Y is are the ground truth and prediction set of pixels. The total accuracy is the average of the accuracy over all the testing samples.

Qualitative evaluation

Qualitative or Subjective methods can be also called observation method. The most convenient method of assessment is subjective assessment, where segmentation results are judged by human evaluators. A qualitative comparison between the ground truth mask and the predicted liver mask image will be used to judge the accuracy of the models while using different loss functions to solve the flipping issue in section 4.2.

3.6 Main proposed architecture designs

The model designs are categorized into four main categories. The **first** category called Deeper models aim to reply the question stated that, how U-Net model performance will be affected with changing the depth of the model?, while the **second** category called Wider models will answer the question about the effect of increasing the width of the U-Net on the model performance, the wider models generated by stacking more than one U-Net together. The **third** category investigate the effect of modifying, or adding skip connections between bridged U-Nets on the performance to find a better performance model for the liver segmentation task. The research covered the different style of skip and bridged connections between 2 U-Nets and 3 U-Nets. The **last** category of models based on one of the best performance models from category 3 with a modified the structure and connections of 2 bridged U-Nets. The generated models with different loss functions and deep supervision structure aim to solve the issue of flipping that appear as a result of applying rotation and flipping augmentation techniques on the training and testing dataset. The following sections show detailed explanation for all the models included in the four categories.

3.6.1 Deeper U-Net based models

The main objective of this group of models is to investigate the effect of increasing the depth of a single U-Net model on the total model performance. The depth of the model is represented by the number of level in each path of the U-Net. For example, model (32, 64, 128, 64, and 32) consists of 3 levels, 2 levels in each path and the deepest level. The number of applied filters are32 on the first level and 128 at the deepest level. To increase the depth of the model we add one level on the beginning of the model or level at the deepest level based on one of the following approaches, Top-Down or Bottom-Up.

- **Top-Down approach**

Starting with one U-Net model with the only three levels with applied filters (8, 16, 32) for levels one, two, and three respectively. then change the structure by adding extra level at the bottom end to be 4 levels with number of filters (8,16,32, 64) and so on until we reach the model with maximum number of levels 9 and number of applied filters as (8, 16, 32, 64, 128, 256, 512, 1024, 2048). Then repeat the experiments with model that have 16 filters at the first layer and continue repeating with models with different number of filters at the starting layer (16, 32, 64, 128, and 256). In total we used 28 models, divided into 6 groups based on the number of filters applied on the first level (8, 16, 32, and 64,128,256). Each group consists of number of models based on the contained number of levels (2, 3, 4, 5, 6, 7, 8, and 9).Figure 3-9

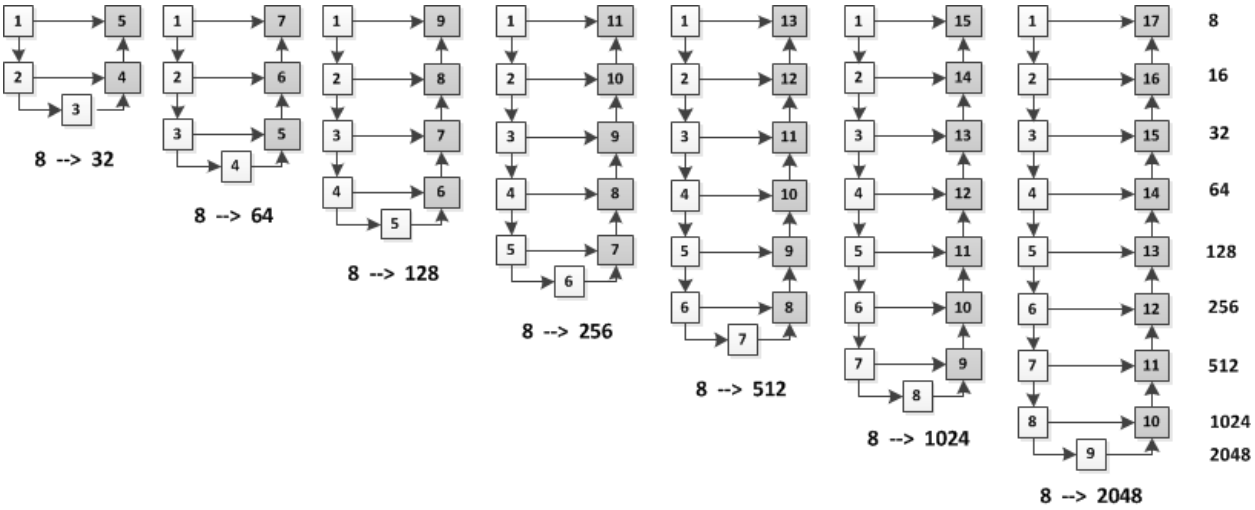


Figure 3-9 Top-Down approach, example for model starts with 8 filters applied to the first layer 32 filters applied to the deepest layer then increasing the depth of the model by adding extra layer with 200% of the filters each step, ending with 2048 filters applied on the deepest layer of model of 9 levels.

➢ Bottom-Up approach

The second approach is to start at the deepest level with the original number of filters (1024) and with three levels model (128, 256, and 1024). We get the other models by adding more level to the beginning of the model. For example, the second deeper model would have the structure of 4 levels with number of filters (64,128,256,1024), and continue adding extra level for each experiment until the deepest structure with 8 levels with number of filters for each level as (8,16,32,64,128,256,1024) respectively.

The same approach would be repeated for 3 groups of models based on the number of filters applied to the deepest level. The number of filters at the deepest level would be 512, 1024, and 2048 number of filters. The total number of models for this experiment is 27 models. In order to test the effect of the used image size on the segmentation accuracy, all the conducted experiments would be repeated 3 times for different image sizes (128*128, 256*256, and 512*512). Some of the image size wouldn't be applicable to be used with certain model structure because the decreasing of the size to half after each level could lead to zero image size output with deeper model, for example using image size 128*128 with model structure (8, 16, 32, 64, 128, 256, 512, 1024, and 2048).Figure 3-10

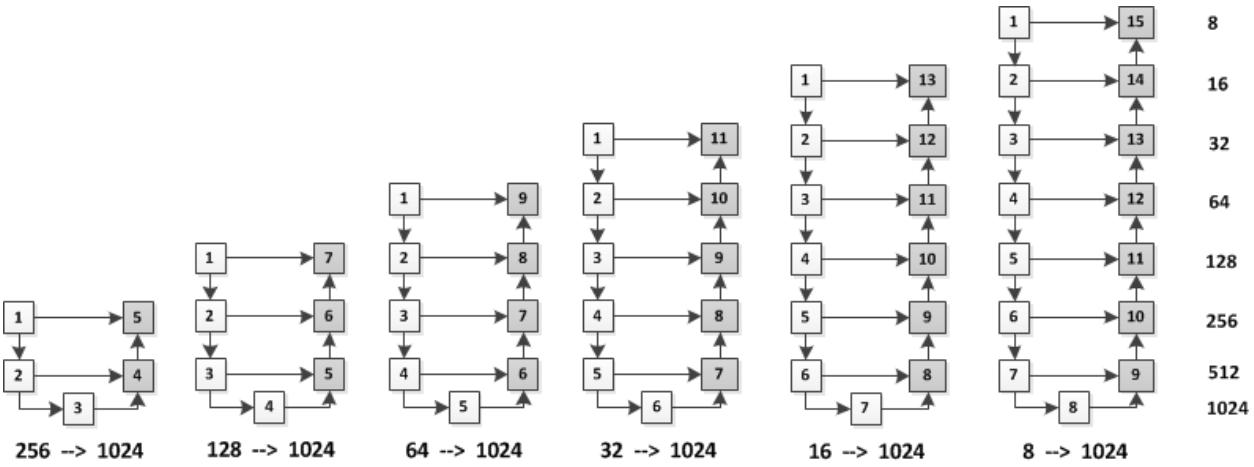


Figure 3-10 Bottom-Up approach, example for model starts with 256 filters applied to the first layer 1024 filters applied to the deepest layer then increasing the depth of the model by adding extra layer with 50% of the filters each step to the top of the model, ending with 8 filters applied on the highest layer of model of 8 levels.

3.6.2 Wider U-Net based models

The main objective of this category is to investigate the effect of increasing the width of the model on the total performance. Increasing the width of the model means adding one complete U-Net model with the same structure. The connection between each two models will be only a direct adding of the output feature maps of the final level of first U-Net to the input of the first level of the second U-Net. This approach will allow exploring the effect of stacking more than U-Nets on the accuracy and the limitations in terms of how many stacked U-Net will give better accuracy? Will the accuracy keep increasing with adding 3 or 4 or 5 U-Nets? Pear in mind stacking U-Nets without any extra connections or integrating other algorithm can be considered a special type of deeper U-Net.

Starting with U-Net model with the original structure, 5 levels, with number of filters applied for each level is (64, 128, 256, 512, and 1024) then stacking a second U-Net with the same structure, The second U-Net would get the input from the output of the first U-

Net. Three main models are implemented, first model consists of two stacked U-Nets, the second contains 3 stacked U-Nets, and the last model consists of 4 stacked U-Nets. Because of computational resources limitations only 4 stacked U-Net could be implemented and only images with sizes (128*128 and 256*256) could be used. Figure 3-11

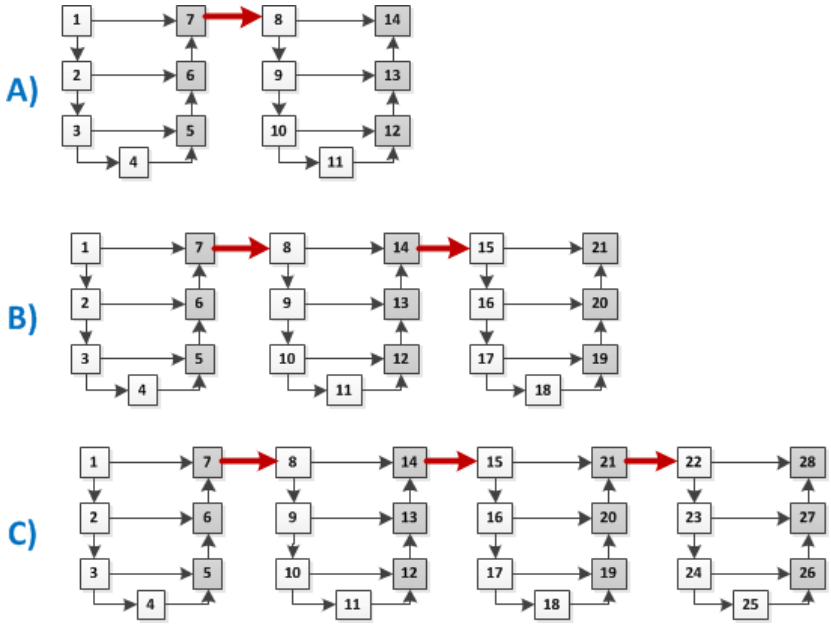


Figure 3-11 Wide models based on U-Nets, A- two stacked U-Nets, B- Three Stacked U-Nets, C- Four stacked U-Nets

3.6.3 Skip connections

This category aims to investigate different connections and skip connections between two or more U-Net model and the effect of different connections on the model performance in terms of accuracy and loss enhancement. The models are based on "2 bridged U-Net" model section 3.4.2. The developed models are divided into three groups based on the number of connected U-Nets. The first group used 2 bridged U-Net, the second group used 3 U-Nets and the third group used three U-Net will long connections.

➢ Two bridged U-Net

The original model used 2 bridged U-Net. The model has direct skip connection from B2 to B3 (Yellow) and a skip connection from B1 to B4 (Red) in addition to the normal skip connections in each U-Net from B1 to B2 and from B3 to B4 (Blue). In the modified model a new skip connection from B2 output to the input of B4 (Green) replaced the original connection from B1 to B4 (Red), while the compound model used both skip connections from B1 and B2 to the input of B4. The skip connections concatenate the output feature maps as the output of one of the U-Net branches to the inputs of another U-Net branch. The skip and bridged connections objective is to added features that might lost during the convolution and down-sampling in the contraction path to the expansion path of the same or other U-Nets. Figure 3-12

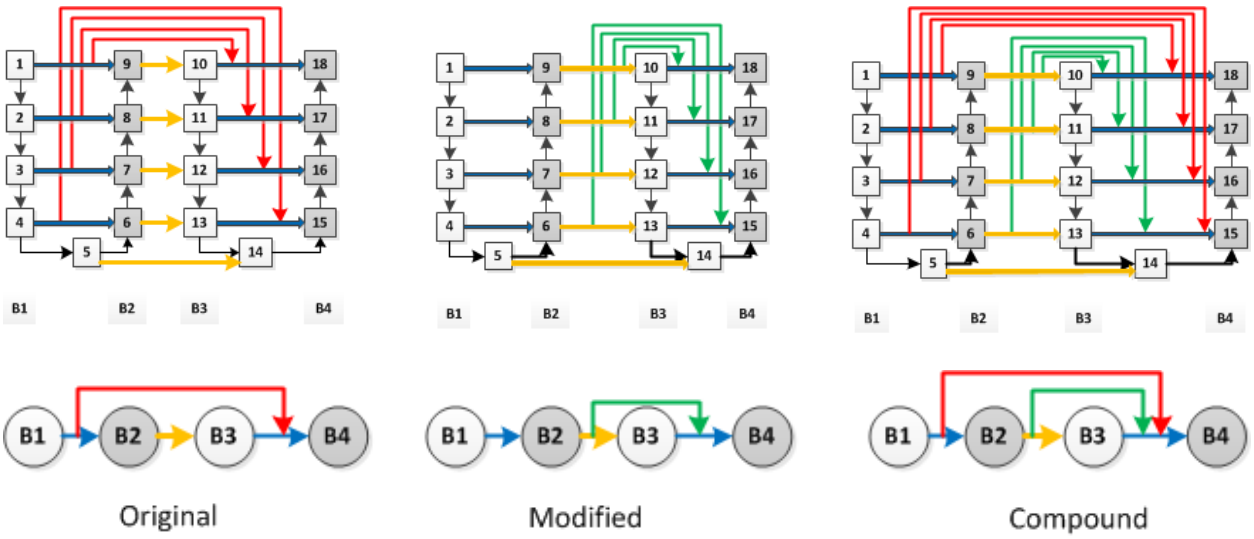


Figure 3-12 Different types of bridged connections and skip connections with 2 U-Nets models

➢ Three Bridged U-Net

The models will use three U-Nets with different combinations of skip connections. The connections would add features from fist U-Net to the second and from the second to the third U-Net. The connections between each two U-Net will be one of the previous

category models (Original, Modified, and Compound). the name of any of three U-Net model will have two parts, first part will explain the connection between first and second U-Nets and the second part will state the type of connection between second and third U-Net, for example (Original – Compound) means that, the first and second U-Nets connected in the Original style (Red) while the second and third U-Nets implemented the Compound style (Red + Green).Figure 3-13

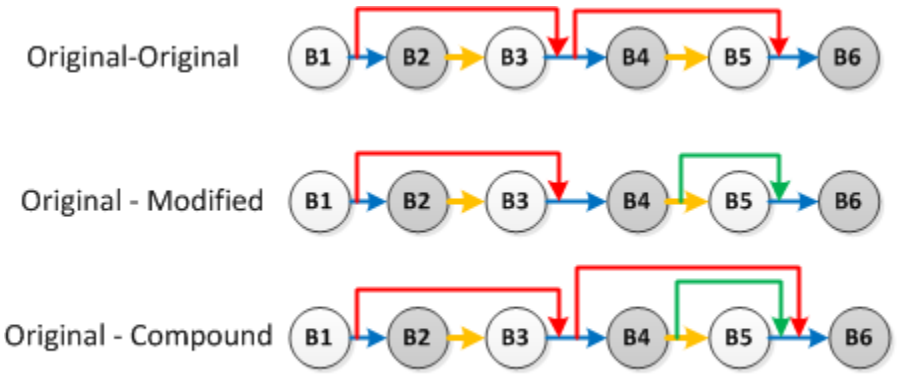


Figure 3-13 Representation for examples of models developed based on three U-Net models using the different connections style (Original, Modified, and Compound)

- **Three Bridged U-Net with long connection**

This group of models based on 3 U-Net models. These models have exactly similar connections as in the previous group. In addition to the skip connections between first and second U-Nets and second to third U-Net, a long connection had been added to each model to concatenate the output feature maps B1 in the first U-Net to the inputs of B6 of the third U-Net (Blue). The names and list of variant connections are displayed in Figure 3-14

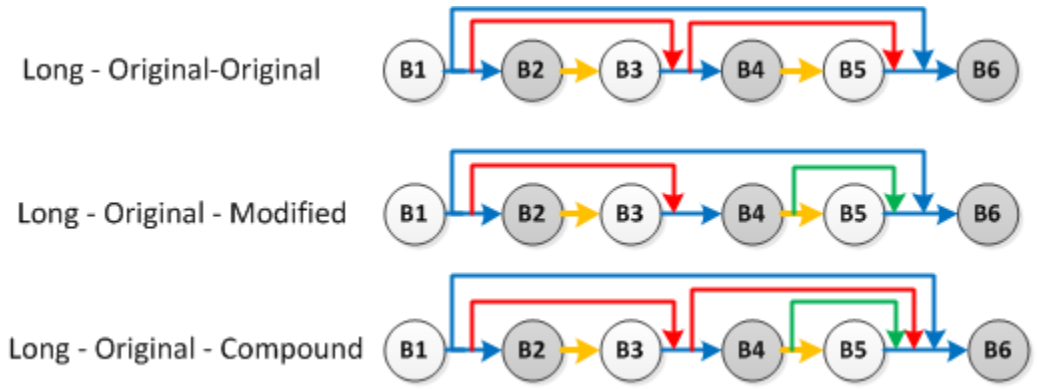


Figure 3-14 Representation for examples of 3 U-Net models with long connections between U-Nets (1 and 3)

3.6.4 Loss functions design

The objective of this group of experiments is to investigate, develop and modify the loss function in the modified model to solve the problem of flipping issue that appeared during the liver segmentation process. While using augmented data in the training process using both U-Net and 2 bridged U-Nets models, the testing results showed that, the modes detected part of the heart as a separate part of the liver. This issue resulted from using the rotation and flipping approaches during the augmentation. The models did not learn features that can relate the liver to a specific position of the CT image. The model segmented liver and part of the heart as liver in both sides of the CT image. Figure 3-15

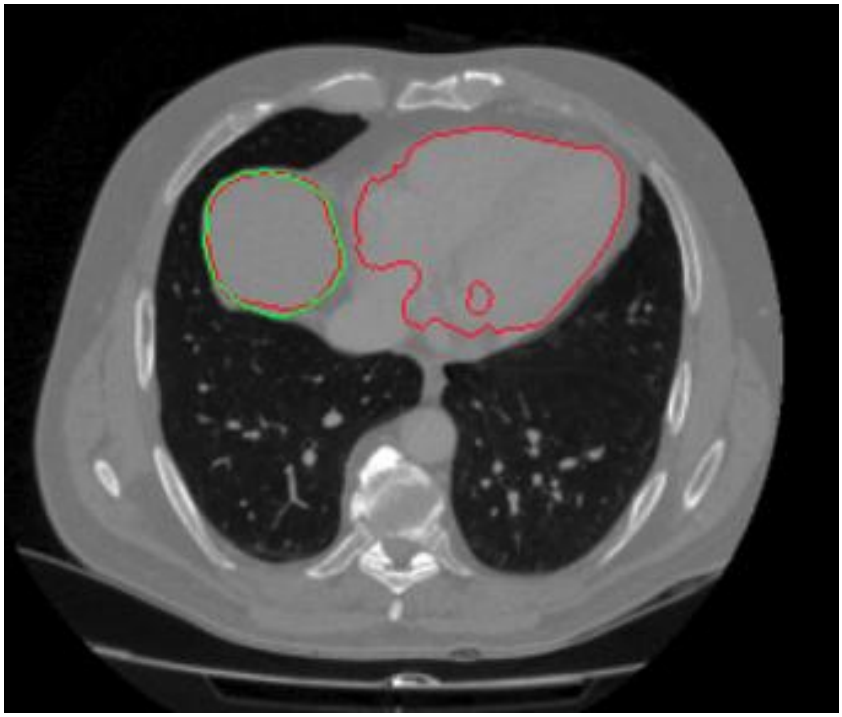


Figure 3-15 The flipping issue appeared due to augmentation process, The ground truth (Green line), and the predicted liver (Red line)

- **Regression loss**

Using augmented data with original U-Net or 2 Bridged U-Net showed that, the model can detect the liver in two sides of the CT at the same time; in fact the model detected the heart as a second liver (or part of the liver because of similar texture and flipping and rotation augmentation techniques.

- **Dice similarity coefficient (DSC)**

The models implemented the Dice Similarity Coefficient (DSC) as the main loss function. The Dice similarity coefficient is a statistical tool which measures the similarity between two sets of data. This index has become arguably the most broadly used tool in the validation of image segmentation algorithms.

The equation for this concept is:

$$DSC = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

- Where X and Y are two sets. The two sets represent ground truth (Masks) and the predicted liver respectively. a set with vertical bars either side refers to the cardinality of the set, i.e. the number of elements in that set, e.g. $|X|$ means the number of elements in set X .

- $\cap$ is used to represent the intersection of two sets, and means the elements that are common to both sets. Figure 3-16. The model use backpropagation to update the weights to minimize the loss value.

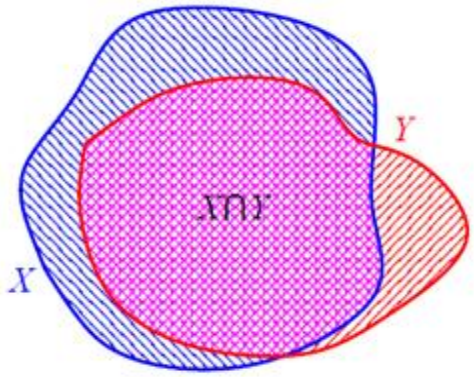


Figure 3-16 Dice similarity coefficient representation

- **Distance between centroid points of blobs**

The **center of mass** is a position defined relative to an object or system of objects. It is the average position of all the parts of the system, weighted according to their masses.

Figure 3-17. For simple rigid objects with uniform density, the center of mass is located at the centroid.

$$x = \frac{\sum_{i=1}^{n} xi * wi}{\sum_{i=1}^{n} w_i} \quad ; \quad y = \frac{\sum_{i=1}^{n} yi * wi}{\sum_{i=1}^{n} w_i}$$

Where x , y are the coordinates of the centroid point of a blob.

n is the number of points in the blob, w is weighted mass of the point of i

The distance between centroid points of two blobs $Dist$

$$Dist = \left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right)$$

Where $(X1, y1)$ and $(X2, y2)$ are the x and y coordinates of the centroid points of blob 1 and blob 2 respectively.

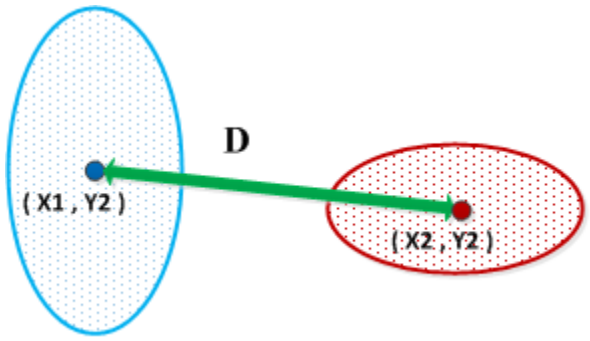


Figure 3-17 Distance between center of mass of two blobs (Centroid distance)

- **Weighted sum of DSC and Centroid functions**

The proposed loss function will to calculate the distance between the centroid points of the blobs and add it to the dice similarity coefficient. The total loss of the model will be

the total weighted sum of centroid distance between blobs and dice similarity coefficient. In other words, to minimize the loss the model will minimize the distance between centroid points and maximize the Dice similarity coefficient.

$$L = DSC * \beta 1 + Dist * \beta 2$$

Where L is the total loss, DSC is Dice Similarity Coefficient; $Dist$ is the distance between centroid points if the blobs, $\beta 1$ and $\beta 2$ are constant weights for DSC and $Dist$ respectively.

- **Deep- supervision with different loss functions**

Deep supervision concept based on the concept of generating output at each level of the model and calculating the total loss as the weighted sum for all losses at all levels. Using deep-supervision allow the model to apply loss function at deeper levels of the model and accumulate the total loss with weights for each loss.

$$\text{Loss} = \sum_{i=1}^{n} L_i * \alpha_i$$

Where n is the number of levels in the model, α is the constant weight for each loss function, L is the loss at level i.

- **Deep supervision with De-Convolution using DSC loss function**

This approach applies one or more De-Convolution layer at each level to increase the size of the output feature maps to reach the same size as the original ground truth and the original image. The loss function would be applied on the resulted feature maps after the De-Convolution and the ground truth. The figure shows the number of de-convolution layers

that would be applied at each level before applying the loss function. For example, if the ground truth and training sample size is 256*256, while the size of feature maps at level 4 is 32*32, the De-convolution layers would be used three times to increase the feature maps size to 46*46 then 128*128 then 256*256 equal to the ground truth. ***Dice similarity coefficient (DSC) will be used as the loss function and will be applied at all levels of the model***. Figure 3-18

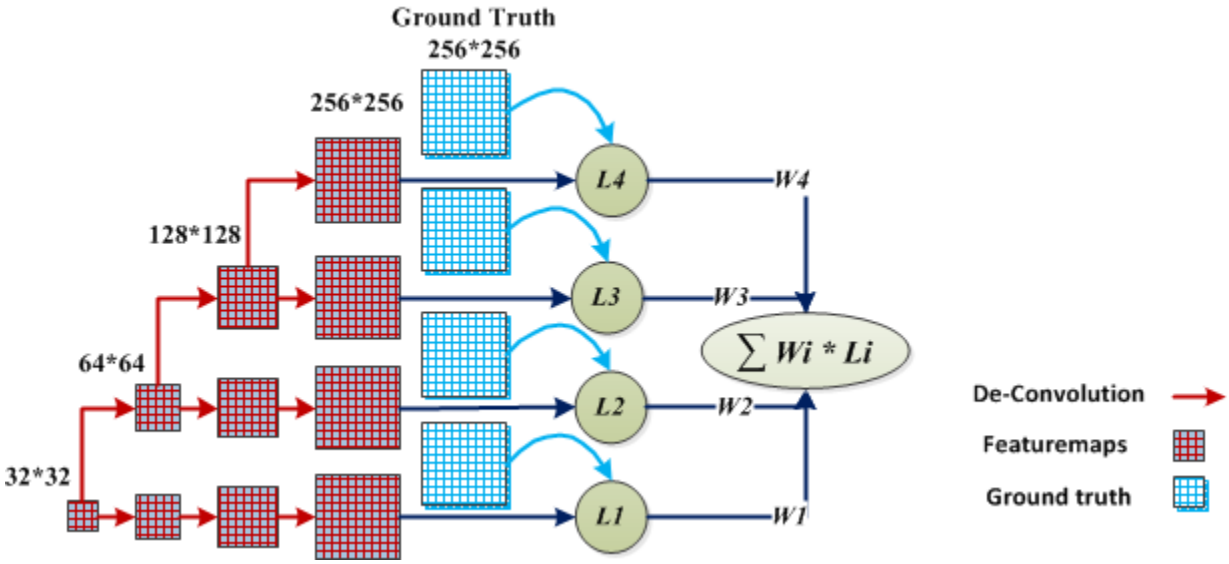


Figure 3-18 Deep supervision approach using de-convolution processes

- **Deep supervision with multi-resolution mask using DSC loss function**

This category implements the concept of deep supervision by accumulative sum of loss functions applied at all levels of the model on the expansion path. This category will use a ground truth mask with the same size as the feature maps generated at each model levels. For example, when the out feature maps size from a certain level is 64*64, the model will use a ground truth mask with size 64*64 and apply the loss function. The total loss will be the total weighted sum for the sub loss functions applied at each level of the model. Similar to the previous category of models, ***Dice similarity coefficient (DSC) will be used as the loss function***. Figure 3-19

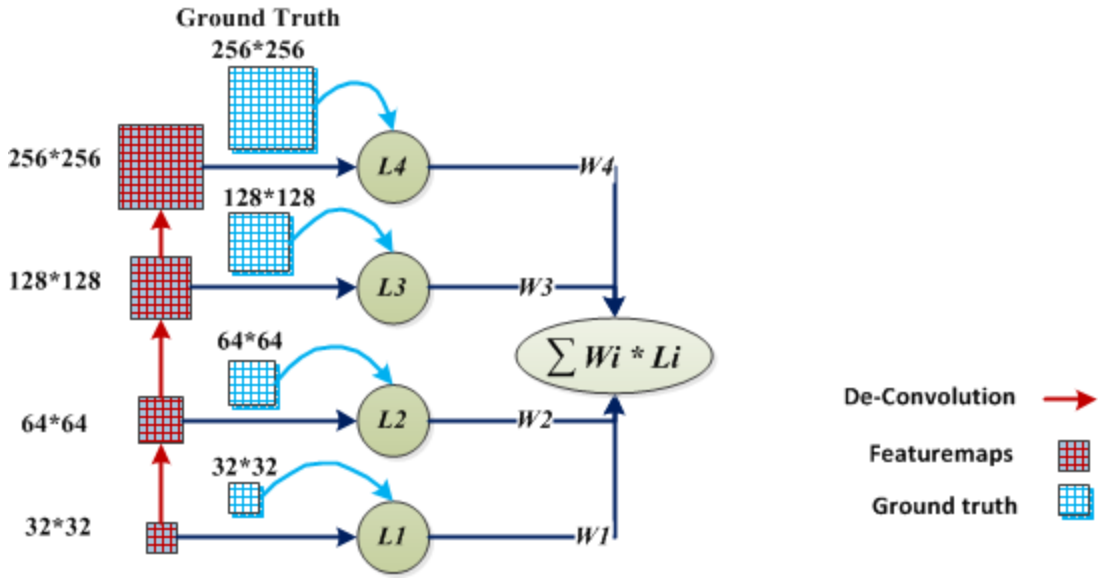


Figure 3-19 Deep supervision approach using Multi-resolution masks

- **Deep supervision with De-Convolution using DSC and Centroid loss functions**

This category of models implemented a proposed loss function that represented as weighted sum of sub-loss functions at each level while the sub loss function is represented as the weighted sum of DSC and centroid distance loss functions.

$$L = DSC * \beta 1 + Dist * \beta 2$$

$$\text{Total Loss} = \sum_{i=1}^{n} L_{i} * \alpha_i$$

$$\text{Total Loss} = \sum_{i=1}^{n} (DSC * \beta 1 + Dist * \beta 2)_{i} * \alpha_i$$

Where n is the number of levels in the model, α is the constant weight for each loss function, L is the loss at level i. $\beta 1$ is the constant weight for **D**ice **S**imilarity **C**oefficient ***(DSC)*** , $\beta 2$ is the constant weight for distance between centroid points loss function ***(Dist).***

The model in this category is similar to the **Deep supervision with De-Convolution** model except that, the DSC loss function will be replaced by the weighted sum of DSC and centroid distance loss function as explained in the previous equations. Figure 3-18

- **Deep supervision with multi-resolution mask using DSC and Centroid loss functions**

The model in this category is similar to the **Deep supervision with multi-resolution mask** model except that, the DSC loss function will be replaced by the weighted sum of DSC and centroid distance loss function as explained in the previous equations. Figure 3-19

3.7 Training/validation process

The training process consists of three sub-processes. First, ~74,000 augmented DICOM image loaded using customized batch generation functions. Same number of ground truth masked loaded for training the model. The second process is setting up all the needed parameters for training. The model trained for 10 epochs and the learning rate started with 1e-5 with calculated decay rate after each epoch. Each model trained three times used different image size (512*512, 256*256, 128*128) to find the best image size that result the best accuracy. The batch size changed from training cycle to another based on the memory and resource limitations. The model trained with kernel size 3*3 and max pooling with size 2*2. The contracting path implemented dropout layer with 0.5 rate while the expansion path used dropout rate 0.4 and the de-convolution layers applied stride size 2*2. The number of filters applied at each level increased by 200% from level n to level n+1 while decreased by 50% from level n to n-1 at the expansion path. Table 3-1

The number of samples in 3Dircadb1 dataset became 112,920 after the augmentation. Training, validation, and testing samples became 74,680 (14 patients), 23,680 (4 patients), and 14,560 (2 patients) respectively. The testing samples represent 13% of the total samples of 3Dircadb1, while the remaining samples divided into training and validation with percentage 75% and 25%.

Parameters	Fixed	Variable	values
Batch size		Y	2,4,8,16,32 based on model and memory size
Image size		Y	(128*128) , (256*256), (512*512)
Number of epochs	Y		10
Learning Rate	Y		1e-5
Filter size	Y		3 * 3
Pooling size	Y		2 * 2
Dropout rate contracting path	Y		0.5
Dropout rate expanding path	Y		0.4
Fitters per layer	Y		Previous layer's filters * 2

Table 3-1 Model parameters and training process settings

3.8 Testing process

The trained models are tested using two different datasets. The model tested using original DICOM images and augmented DICOM for two patients from **3Dircadb1** dataset. The number of tested samples is 400 and 14,000 images for original and augmented image respectively. For visualization purpose, the ground truth and predicted mask are mapped on each tested sample image. Figure 3-20

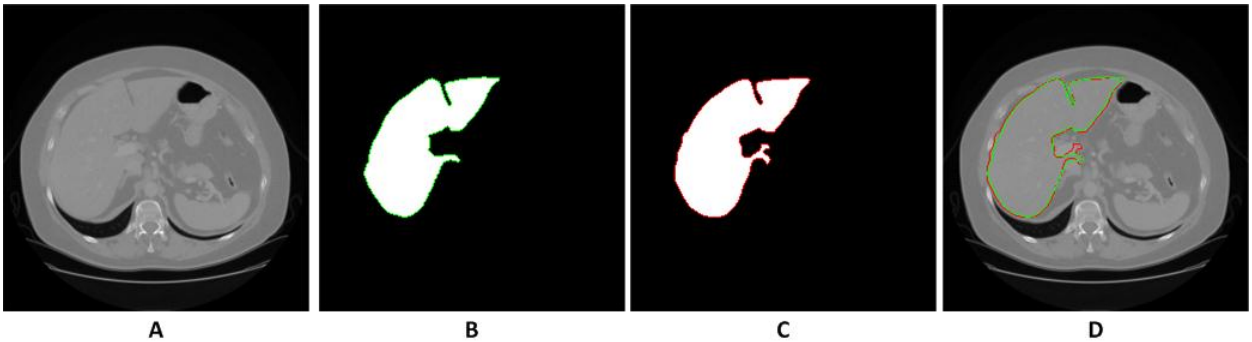


Figure 3-20 Testing result example, A) original, B) Ground Truth, C) Predicted Mask, D) mapped masks to the original image

The testing samples were not present in either training or validation process. The model tested using IKN dataset that was not included in the training or validation process because there are no annotated masks for the dataset. The testing samples contains ~1500 selected DICOM image for 5 patients to cover different cases.

Chapter 4
Experimental Results and Discussions

The experiments divided into two parts. The first part aims to investigate the variations of models based on U-Net structures and study the effect of making the U-Net model wider by adding extra U-Net, deeper by adding extra layers, and different types of connections between 2U-Nets and 3U-Nets that known as skip-connections on the model performance. One of the objectives of this part is to propose recommendation for using models based on U-Net according to the indications and implications of the achieved results. Another objective for this part is to create a model based on one or more U-Nets with bridge and skip-connections that achieved better accuracy for liver segmentation.

The second part aims to solve the flipping issue that appeared because of training the models on augmented data using rotation and flipping techniques. The flipping issue showed that, when the models trained using rotation and flipping augmentation techniques, for significant number of samples the predicted masks detected the liver at both sides of the image at the same time while it should appear at one side. The second part used W-Net model that consists of 2-U-Nets with compound bridge and skip connections. The sub- parts will test 3 different loss functions starting with DSC, Centroid loss, weighted loss of (DSC and Centroid). Two types of deep-Supervision approaches will be tested using DSC, and combined (DSC + Centroid) loss functions. One of the two deep-supervision approach will use De-convolution process to increase the feature maps size to be equal to the original

mask 256*256 , while the second approach will use Multi-Resolution masks with different size to be used at different level of the W-Net outputs.

Section 4.1 and its subsections cover the first part of the experiments which aim to achieve the first objective that aim to conclude recommendations of using U-Net for medical image segmentation for liver. Section 4.2 and its subsections describe the second part of the experiments which aim to resolve the flipping issue.

4.1 Model structure variations

This section covers the first part of the experiments that investigate different structures of U-Net and it includes three sub-categories of experiments, 1 investigating the Deeper Models (section 4.1.1), 2 Investigating the wider Models (section 4.1.2), and 3 Study for models based on 3U-Net with different connections (section 4.1.3). The testing samples include the images for Patient 5 **(P5**) and patient 20**(P20),** in addition to testing the augmented data for the two patients **(P5_Aug, and P20_Aug).** The average accuracy for tested data and segmented data **(Test_Avg, Test_Avg_Aug)** is calculated for comparison purpose against **Training** and **Validation** accuracy. The testing accuracy calculated for each patient as Dice similarity coefficient for each image then average over the number of the patient's images. The results will only focus on 4 measures **(Training, Validation, Test_Avg, and Test_Avg_Aug)** while the remaining results for P5, P20, P5_Aug, and P20_Aug will be shown for reference only.

4.1.1 Deeper U-Net based models

Generating deeper models based on single U-Net model by increasing the number of levels in both contracting and expansion paths. Each level of the model in the contraction path contains two convolution layers in addition to one maxpooling and one dropout layers while in the expansion path, each level contains de-convolution layer applied on the concatenation of the output feature maps from the previous level and the layer at the same level on the contracting path. Deeper model can be generated by of two approaches, Top-Down or Bottom-Up.

4.1.1.1 Models start with layer of 8 filters and different depth

All four measured factors (Training, validation, testing, and testing using augmented data) have the same behavior, starting with very low value at the model with the minimum number of layers 8-32 then kept fluctuating until reached the maximum accuracy with the deeper model 8-512 then decreased at next deeper model 8-1024 and increased again at the deepest model 8-2048 Figure 4-1. The model with the minimum number of layers 8-32 recorded the minimum value for each of the factors' validation, testing, and testing using augmented data while a deeper model 8-256 recorded the minimum accuracy for training. The accuracy of all models started with 8 filters at the first layer increased when getting deeper to reach the maximum at 8-512 with 7 levels. Table 4-1

Name	Test_Avg	Test_Avg_Aug	Training	Validation
8--32	**19.08%**	**12.78%**	73.29%	**19.95%**
8--64	41.33%	36.97%	76.61%	75.02%
8--128	34.09%	22.44%	74.82%	48.72%
8--256	42.16%	46.11%	**56.25%**	76.52%
8--512	**78.56%**	**73.89%**	**86.32%**	**89.89%**
8--1024	34.82%	29.09%	82.09%	57.89%
8--2048	74.50%	73.20%	82.54%	82.93%

Table 4-1 Models start with layer of 8 filters and different depth (8-32, 8-64, 8-128, 8-256, 8-512, 8-1024, 8-2048) using image size 256*256 , Highest accuracy (Green) lowest accuracy (Red)

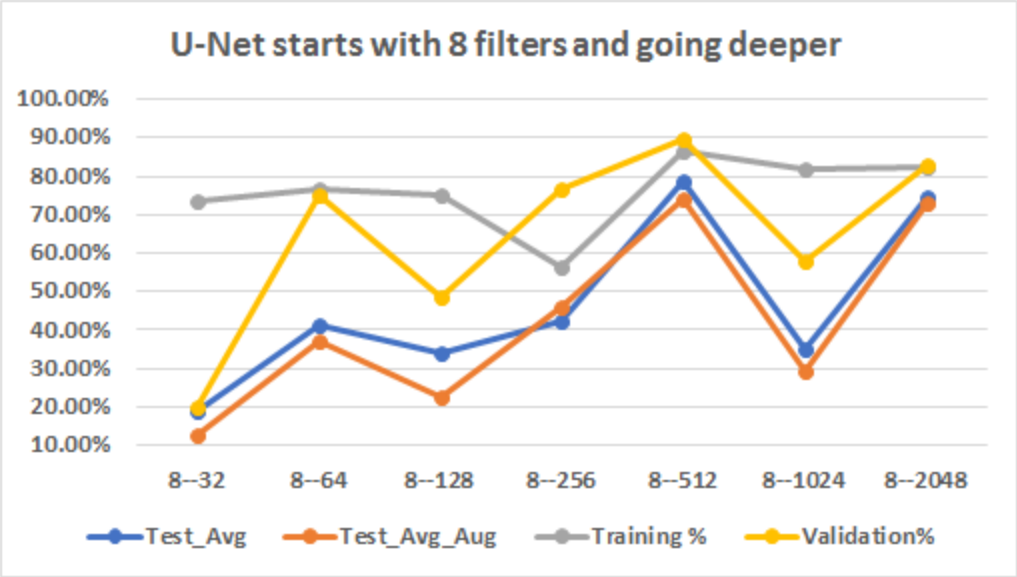


Figure 4-1 Accuracy of training, validation, testing, and testing using augmented data for all models started with 8 filters at the first level

4.1.1.2 Models start with layer of 16 filters and different depth

Moving from model 16-64 with 3 levels depth to the deepest model 16-2048, the three factors of Training, Validation, and Test_Avg_Aug have the same behavior starting with the minimum value at model 16-64 then fluctuate to the maximum accuracy at a deeper model 16-1024 then decreased again with the deepest model while Test_Avg accuracy start with medium value then increased to the maximum at model 16-1024 then decreased at the deepest model 16-2048.Figure 4-2. While the model with the lowest number of layers 16-64 recorded the minimum accuracy with Training, Validation, and Test_Avg_Aug, the next deeper model 16-128 recorded the minimum Test_Avg accuracy. The model with 7 levels and number of filters 16-1024 recorded the maximum accuracy with all factors, training, and validation, testing, and testing using augmented data Table 4-2. The accuracy of all models started with 16 filters at the first layer increased when getting deeper to reach the maximum at 16-1024 with 7 levels. Figure 4-2

Name	Test_Avg	Test_Avg_Aug	Training	Validation
16--64	44.47%	**35.87%**	**82.01%**	**71.83%**
16--128	**40.41%**	54.83%	83.97%	78.58%
16--256	52.60%	67.13%	84.59%	85.00%
16--512	40.79%	39.95%	93.00%	77.85%
16--1024	**69.81%**	**70.80%**	**95.07%**	**90.23%**
16--2048	45.36%	67.05%	93.24%	85.31%

Table 4-2 Models start with layer of 16 filters and different depth (16-64, 16-128, 16-256, 16-512, 16-1024, 16-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

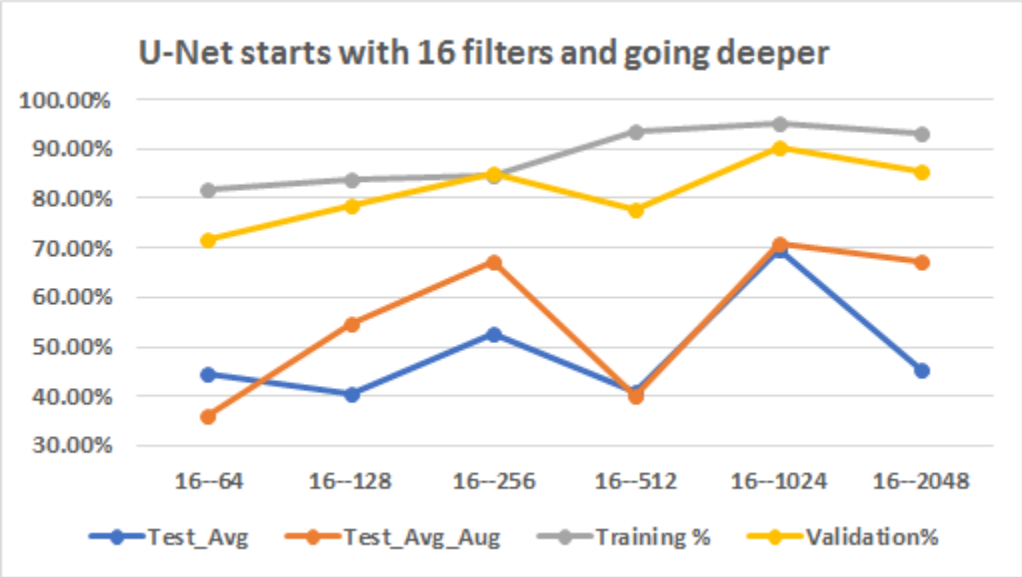


Figure 4-2 Accuracy of training, validation, testing, and testing using augmented data for all models started with 16 filters at the first level

4.1.1.3 Models start with layer of 32 filters and different depth

Training and Test_Avg almost have the same curve behavior which starts with a minimum and medium value at first model 32-128 respectively then kept increasing until reached the maximum accuracy at the deepest model 32-2048. Validation and Test_Avg_Aug also shared similar behavior by starting with medium value then decreased to the minimum at the next deeper model 32-256 then increased to the maximum at deeper models 32-512 and 23-1024 respectively then decreased at the deepest model Figure 4-3. The model with 4 levels 32-256 recorded the minimum accuracy for validation, Test_Avg

and Test_Avg_Aug while model 32-128 with 3 levels recorded minimum training accuracy. While the deepest model with 7 levels 32-2048 recorded maximum accuracy from training and Test_Avg, model 32-512 with 5 levels recorded maximum validation and 32-1024 model achieved the maximum accuracy for Test_Avg_Aug Table 4-3. Going deeper with models started with 32 filters increase the accuracy even if Validation and Test_Avg_Aug decreased after got the maximum accuracy.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
32--128	45.08%	38.81%	**84.54%**	80.05%
32--256	**41.84%**	**34.11%**	89.73%	**64.30%**
32--512	46.49%	60.07%	96.50%	**98.00%**
32--1024	66.42%	**70.45%**	96.91%	92.72%
32--2048	**68.59%**	64.62%	**97.27%**	86.96%

Table 4-3 Models start with layer of 32 filters and different depth (32-128, 32-256, 32-512, 32-1024, 32-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

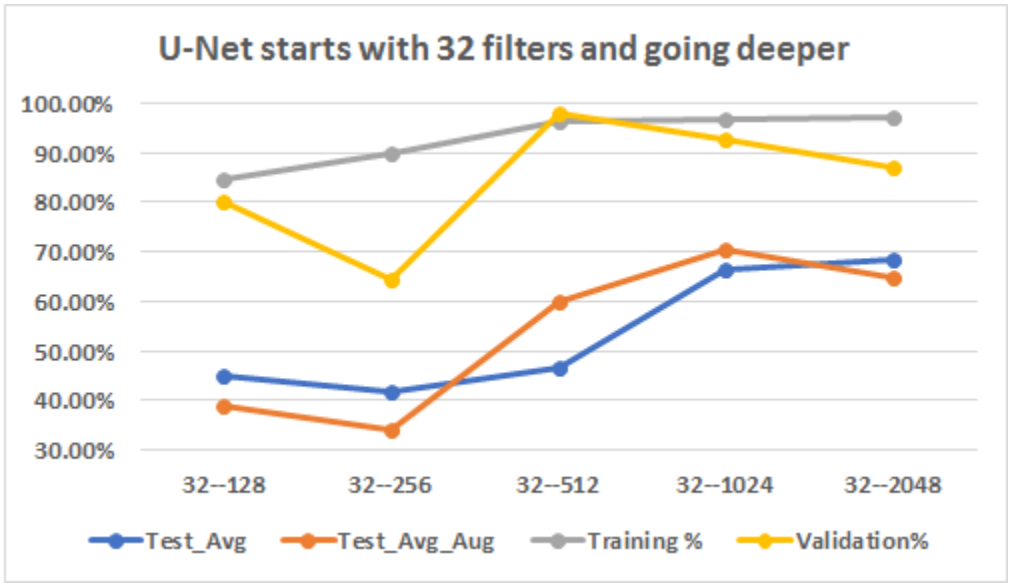


Figure 4-3 Accuracy of training, validation, testing, and testing using augmented data for all models started with 32 filters at the first level

4.1.1.4 Models start with layer of 64 filters and different depth

All the four measures recorded the same behavior starting with the minimum accuracy at the model with minimum number of 3 levels 64-256 and increased with deeper models to reach the maximum accuracy at the deepest model 64-2048. The model with the minimum number of levels 64-256 recorded the minimum for all measures while the deepest model 64-2048 with 6 levels recorded the maximum for all four factors of training, validation, Test_Avg, and Test_Avg_Aug. Table 4-4 Figure 4-4

Name	Test_Avg	Test_Avg_Aug	Training	Validation
64--256	45.57%	39.25%	89.33%	72.55%
64--512	49.96%	44.23%	96.15%	83.52%
64--1024	58.73%	75.93%	97.40%	91.50%
64--2048	60.24%	80.39%	97.81%	93.68%

Table 4-4 Models start with layer of 64 filters and different depth (64-256, 64-512, 64-1024, 64-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

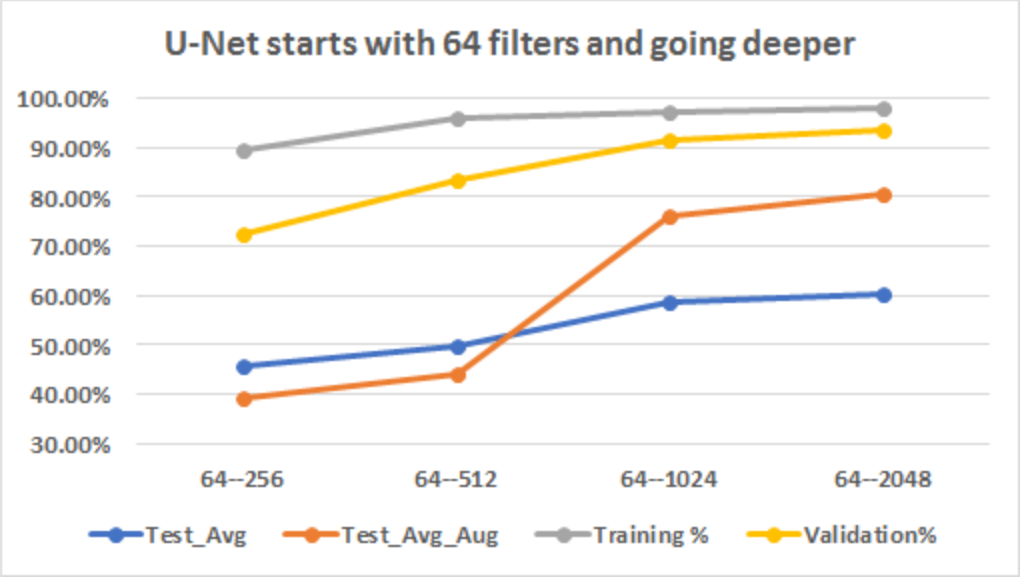


Figure 4-4 Accuracy of training, validation, testing, and testing using augmented data for all models started with 64 filters at the first level

4.1.1.5 Models start with layer of 128 filters and different depth

Training and validation have similar curve with moving toward the deepest model, starting with minimum accuracy at model 128-512 with minimum number of 3 levels then increased to the maximum at the next deeper model 128-1024 then decreased again at the deepest model. Test_Avg and Test_Avg_Aug also have similar curve as starting from the minimum accuracy at model 128-512 and kept increasing with the deeper models to reach the maximum accuracy at the deepest model 128-2048 with 5 levels. The model with the minimum number of levels 128-512 recorded the minimum accuracy for all four factors while the next deeper model 128-1024 recorded the maximum for training and validation and the deepest model 128-2048 recorded the maximum for Test_Avg and Test_Avg_Aug. Figure 4-5 Table 4-5

Name	Test_Avg	Test_Avg_Aug	Training	Validation
128--512	50.86%	52.50%	94.88%	85.72%
128--1024	60.98%	66.26%	97.81%	88.82%
128--2048	69.61%	74.63%	97.51%	87.99%

Table 4-5 Models start with layer of 128 filters and different depth (128-512, 128-1024, 128-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

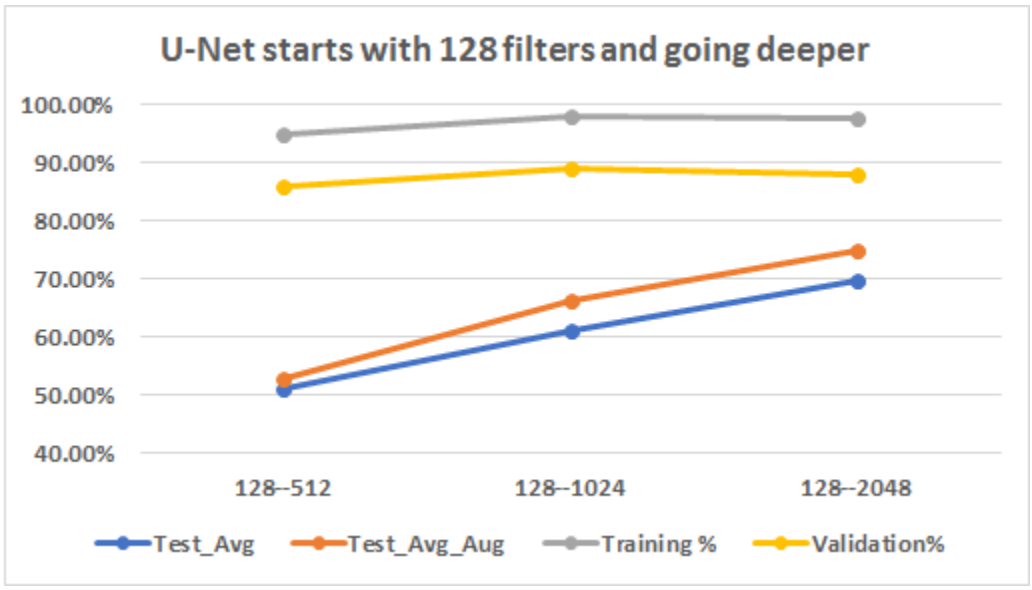


Figure 4-5 Accuracy of training, validation, testing, and testing using augmented data for all models started with 128 filters at the first level

4.1.1.6 Models start with layer of 256 filters and different depth

Models 256-2048 couldn't be testing because of memory limitations. Table 4-6 shows that, the deeper model 256-1024 recorded the maximum accuracy over the first model 256-512 for all factors, training, and validation, testing, and testing using augmented data Figure 4-6.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
256--512	28.74%	41.02%	86.78%	78.33%
256--1024	53.23%	70.20%	95.75%	83.64%
256--2048	NA	NA	NA	NA

Table 4-6 Models start with layer of 256 filters and different depth (256-512, 256-1024, 256-2048) using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

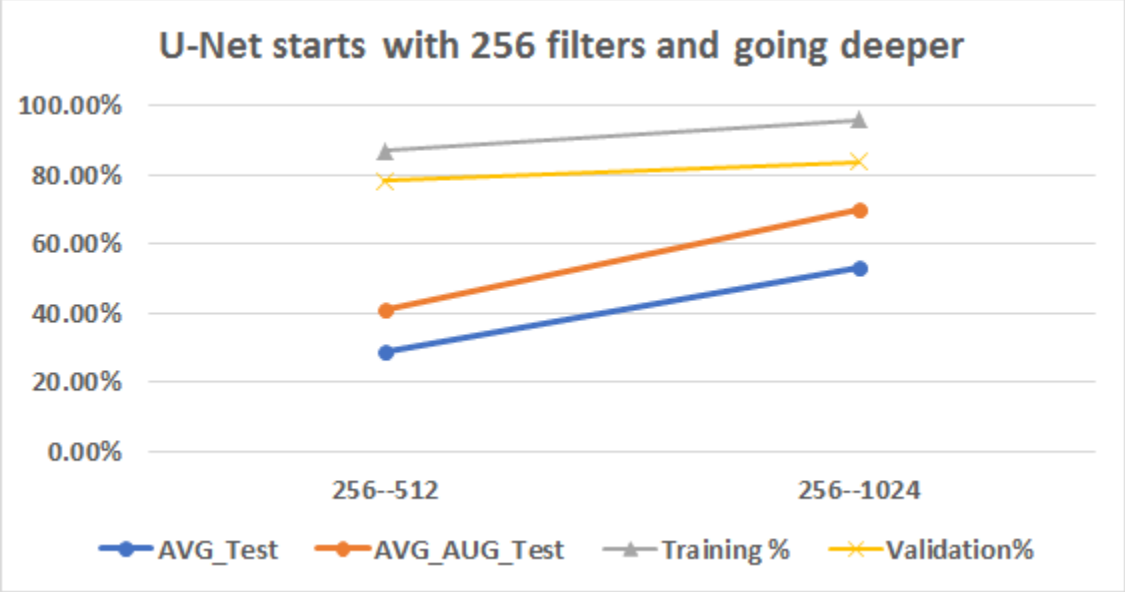


Figure 4-6 Accuracy of training, validation, testing, and testing using augmented data for all models started with 256 filters at the first level

4.1.1.7 All models with 2048 filters at the deepest layer

Model 256-2048 couldn't be tested because of memory limitations. Training and validation followed the same curve as started with medium value at model 128-2048 then increased to the maximum at model 64-2048 then decreased with deeper models to reach the minimum accuracy at the deepest model 8-2048. Test_Avg_Aug factor start with medium value at model 128-2048 then increased to the maximum at the next deeper model 64-2048 then decreased to the minimum at the next model 8-2048 then increased gradually with going through the next deeper models. The Test_Avg factor kept fluctuating until reached the minimum accuracy at model 16-2048 then increased to the maximum at the deepest model 8-2048. Figure 4-7 Shows that, model 64-2048 with 5 levels recorded maximum accuracy for training, validation, and Test_Avg_Aug while the deepest model with 8 levels 8—2048 recorded the maximum accuracy for testing. The deepest model 8-2048 recorded the minimum value for training and validation and maximum of testing while the minimum for testing and testing using augmented data recorded for models 16-2048 and 32-2048 respectively Table 4-7 Figure 4-7. Deeper models enhance the accuracy until certain levels (6) after that the accuracy will decrease again except for Test_Avg where maximum accuracy recorded at the deepest model with 9 levels.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
256--2048	NA	NA	NA	NA
128--2048	69.61%	74.63%	97.51%	87.99%
64--2048	60.24%	**80.39%**	**97.81%**	**93.68%**
32--2048	68.59%	**64.62%**	97.27%	86.96%
16--2048	**45.36%**	67.05%	93.24%	85.31%
8--2048	**74.50%**	73.20%	**82.54%**	**82.93%**

Table 4-7 All models with 2048 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

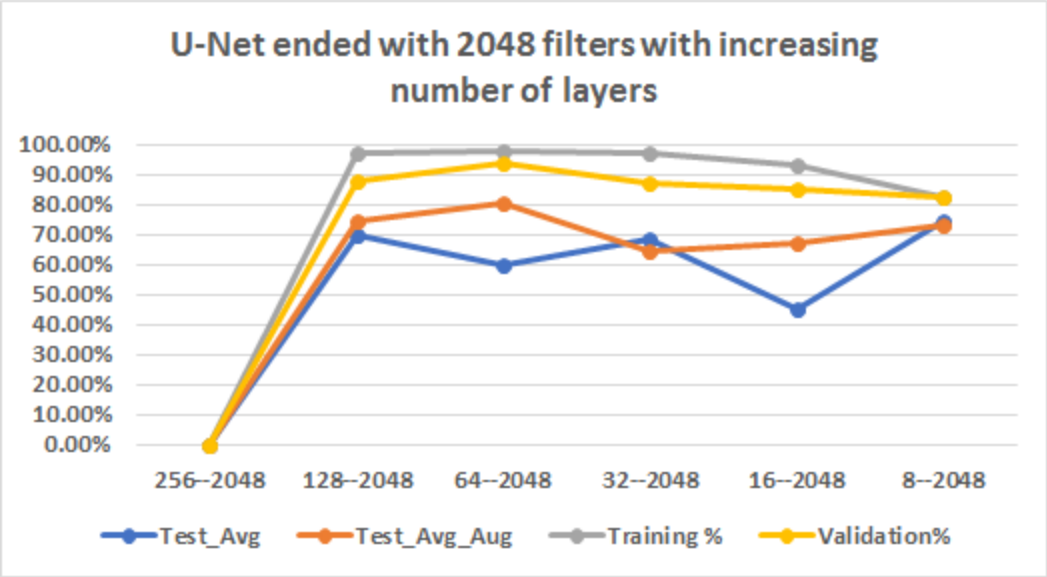


Figure 4-7 Accuracy of training, validation, testing, and testing using augmented data for all models with 2048 filters at the deepest level

4.1.1.8 All models with 1024 filters at the deepest layer

All the four measured factors shared the same behavior of starting to increase after the first model with 3 levels (256-1024) then reached the maximum at different models then continue decreasing to the minimum at the deepest model 8-1024. The maximum accuracy for training, validation, Test_Avg, and Test_Avg_Aug achieved at different models (128-1024 , 32-1024, 16-1024, 64-1024) respectively Figure 4-8 , Table 4-8. The deepest model 7 levels recorded the minimum accuracy for all measured factors, while the maximum value recorded for different models for each factor Table 4-8. Deeper models enhance the accuracy until certain levels (4, 5, 6, or 7) after that the accuracy will decrease again.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
256--1024	53.23%	70.20%	95.75%	83.64%
128--1024	60.98%	66.26%	**97.81%**	88.82%
64--1024	58.73%	**75.93%**	97.40%	91.50%
32--1024	66.42%	70.45%	96.91%	**92.72%**
16--1024	**69.81%**	70.80%	95.07%	90.23%
8--1024	**34.82%**	**29.09%**	**82.09%**	**57.89%**

Table 4-8 All models with 1024 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

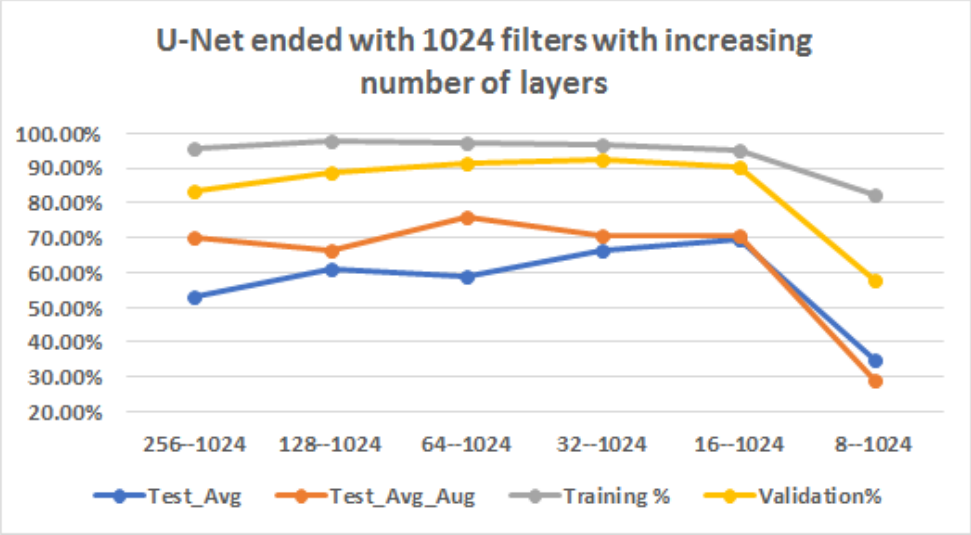


Figure 4-8 Accuracy of training, validation, testing, and testing using augmented data for all models with 1024 filters at the deepest level

4.1.1.9 All models with 512 filters at the deepest layer

With the moving from model with the minimum number of layers 256-512 (2 levels) toward the deepest model 8-512 with7 levels, validation and Test_Avg_Aug have the same behavior, starting with a medium value then decrease for the next model then increase then decrease then increase at the deepest model. While validation reached the maximum accuracy at model 32-512 (5 levels), Test_Avg_Aug recorded maximum at the deepest model 8-512(7 levels) and both of them recorded the minimum at model 16-512. Training started with medium value then kept increasing to reach the maximum at model 32-512

then kept decreasing to reach the minimum at the deepest model 8-512, while Test_Avg start with the minimum accuracy at the first model 256-512 with two levels and increased at the second level then continues decreasing with the deeper models until reached the second minimum at model 16-512 then increased directly to the maximum at the next model 8-512 with the deepest structure of 7 levels Figure 4-9Figure 4-9. If the first model results ignored because it only contains 2 levels which is very rare to happen during the study, model 32-512 with medium depth of 5 levels recorded maximum accuracy for training and validation while the deepest of 7 levels recorded the maximum accuracy for Test_Avg and Test_Avg_Aug. The second deepest model 16-512 recorded the minimum accuracy for all factors except training which recorded the minimum accuracy at the deepest model 8-512 Table 4-9. Deeper models enhance the accuracy for training and validation until certain levels (5) after that the accuracy will decrease again while for Test_Avg and Test_Avg_Aug kept fluctuating until the maximum at the deepest model.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
256--512	**28.74%**	41.02%	**86.78%**	78.33%
128--512	50.86%	52.50%	94.88%	85.72%
64--512	49.96%	44.23%	96.15%	83.52%
32--512	46.49%	60.07%	**96.50%**	**98.00%**
16--512	**40.79%**	**39.95%**	93.00%	**77.85%**
8--512	**78.56%**	**73.89%**	**86.32%**	89.89%

Table 4-9 All models with 512 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

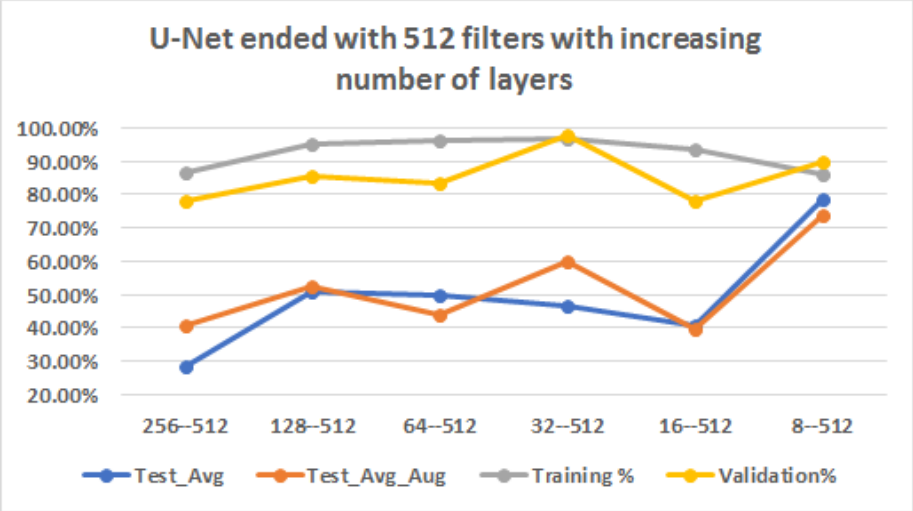


Figure 4-9 Accuracy of training, validation, testing, and testing using augmented data for all models with 512 filters at the deepest level

4.1.1.10 All models with 256 filters at the deepest layer

Validation, testing (Avg_test), and testing using augmented data (Avg_AUG_Test) have the same accuracy curve behavior, started with a medium value at first model 64-256 then decreased to the minimum accuracy at the next deeper model 32-256 then increased directly to the maximum accuracy at model 16-256 with 5 levels then decreased again. Training increased from the first model to reach the maximum at the next deeper model 32-256 with 4 levels then decreased gradually to reach the minimum at the deepest model 8-256 Figure 4-10. The model 32-256 with 4 levels recorded the minimum accuracy for all factors except training which recorded the maximum accuracy, while the next deeper model 16-256 recorded the maximum accuracy for the all factors except training Table 4-10. Deeper models enhance the accuracy until certain levels (4 or 5) after that the accuracy will decrease again.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
64--256	45.57%	39.25%	89.33%	72.55%
32--256	**41.84%**	**34.11%**	**89.73%**	**64.30%**
16--256	**52.60%**	**67.13%**	84.59%	**85.00%**
8--256	42.16%	46.11%	**56.25%**	76.52%

Table 4-10 All models with 256 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

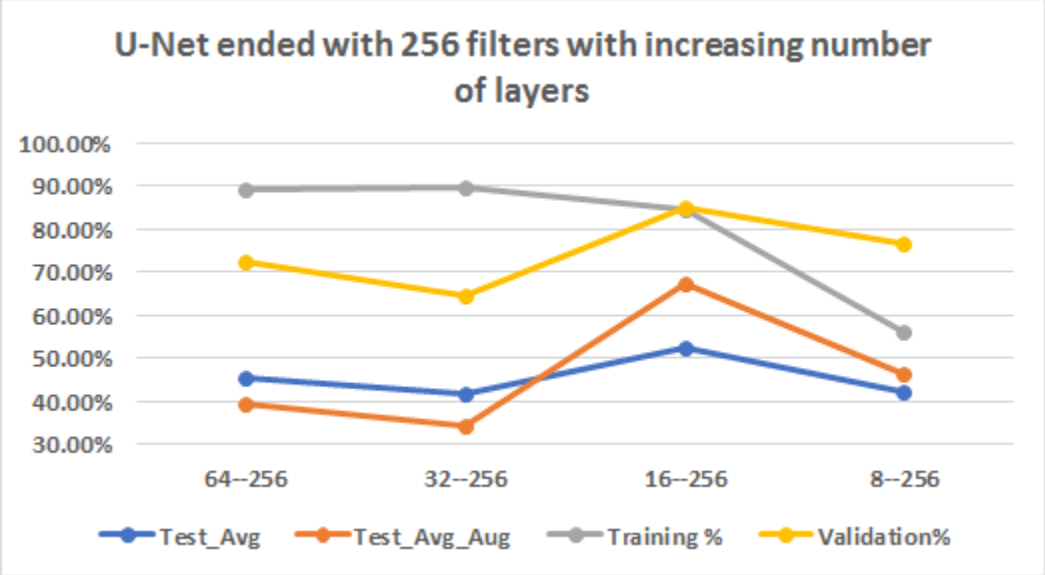


Figure 4-10 Accuracy of training, validation, testing, and testing using augmented data for all models with 256 filters at the deepest level

4.1.1.11 All models with 128 filters at the deepest layer

Training, validation, and testing (Avg_Test) have exactly the same accuracy curve behavior that started with the maximum accuracy at first model 32-128 with 3 levels then decreasing with deeper models until reached the minim accuracy at the deepest model 8-128 with 5 levels. Testing using augmented data (Avg_AUG_Test) started with minimum accuracy at model 32-128 with 3 levels then increased to the maximum at the next deeper model 16-128 then decreased again with the deepest model Figure 4-11. Model 32-128 with 3 levels recorded the maximum accuracy for all factors except (Avg_AUG_Test) where recorded the minimum, while the deepest model 8-128 recorded the minimum accuracy for

all factors also except (Avg_AUG_Test). The model 16-128 recorded the maximum accuracy for Avg_AUG_Test and intermediate values for the other factors Table 4-11. Deeper models recorded worse accuracy and minimum accuracy at the deepest model except Test_Avg_Aug have the opposite behavior.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
32--128	**45.08%**	**38.81%**	**84.54%**	**80.05%**
16--128	40.41%	**54.83%**	83.97%	78.58%
8--128	**34.09%**	22.44%	**74.82%**	**48.72%**

Table 4-11 All models with 128 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

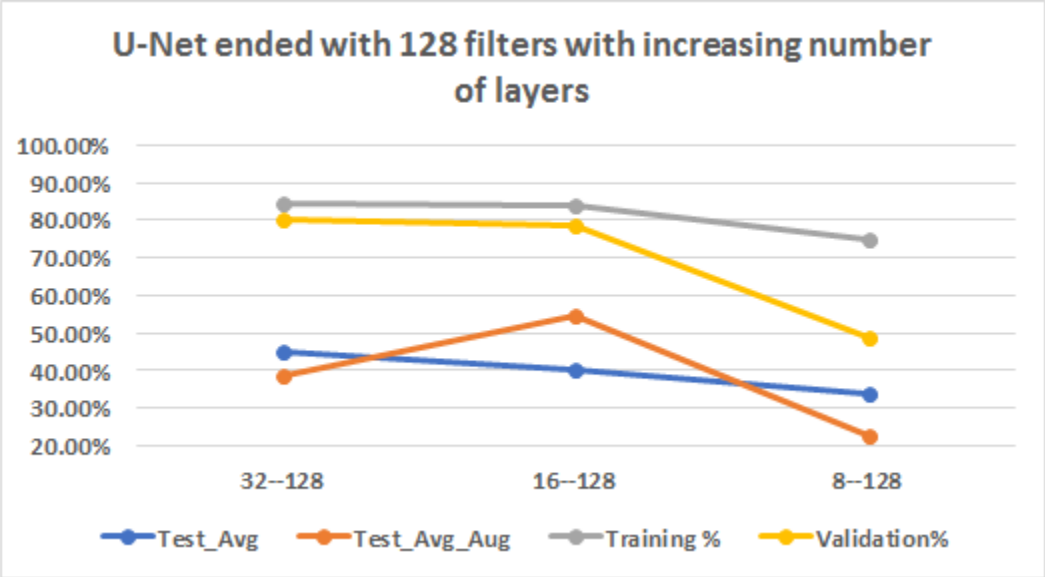


Figure 4-11 Accuracy of training, validation, testing, and testing using augmented data for all models with 128 filters at the deepest level

4.1.1.12 All models with 64 filters at the deepest layer

The accuracy curve for Training and testing (Avg_Test) changed from maximum at first model 16-64 to the maximum at the deepest model 8-64 which is the opposite direction of the curve for validation and AVG_AUG_Test that went from minimum at the first model to the maximum at the second model 8-64. Model 16-64 recoded the maximum

accuracy for training and Test_Avg while the deepest model 8-64 recorded the maximum for validation and Test_Avg_Aug Table 4-12 Figure 4-12.

Name	Test_Avg	Test_Avg_Aug	Training	Validation
16--64	44.47%	35.87%	82.01%	71.83%
8--64	41.33%	36.97%	76.61%	75.02%

Table 4-12 All models with 64 filters at the deepest layer using image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

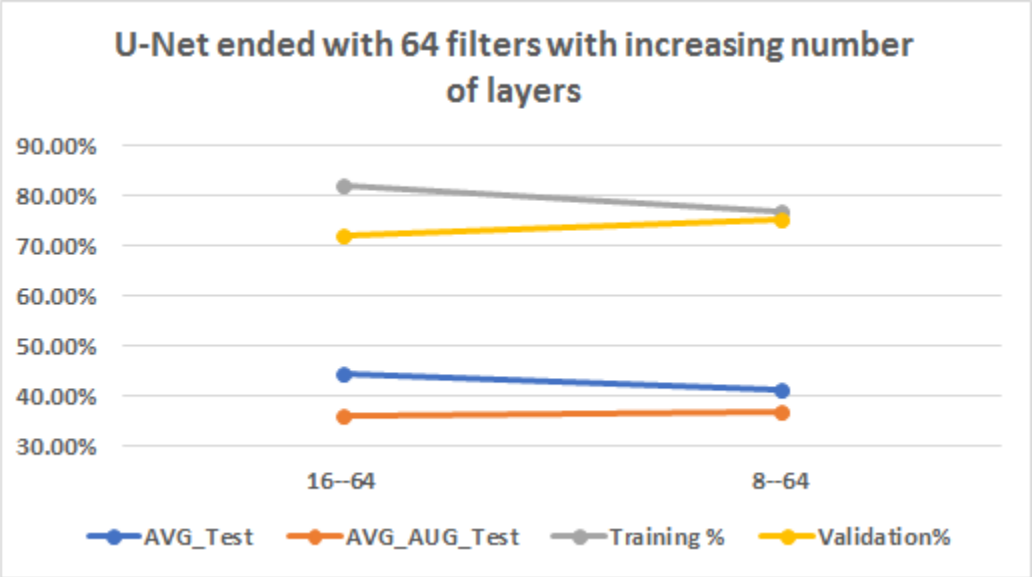


Figure 4-12 Accuracy of training, validation, testing, and testing using augmented data for all models with 64 filters at the deepest level

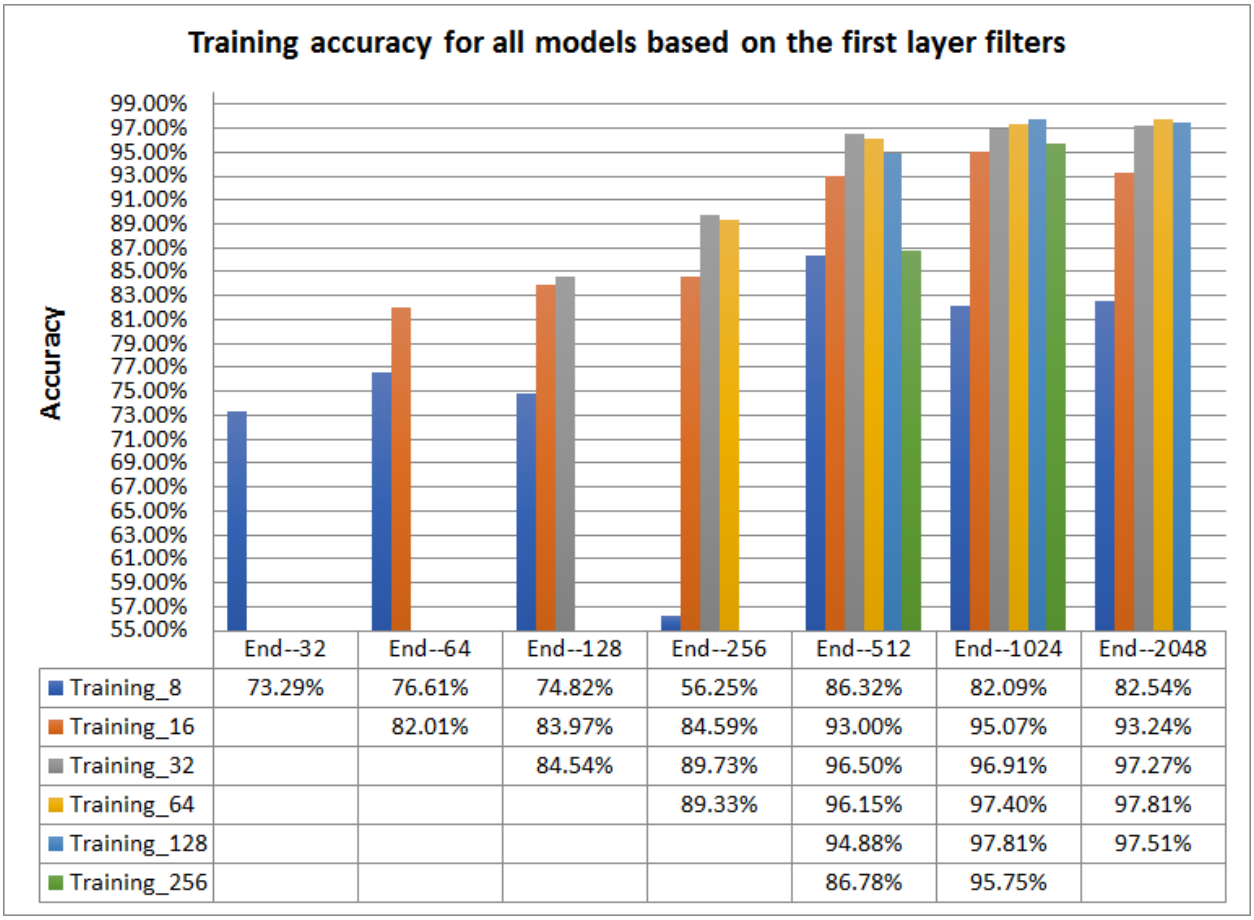
4.1.1.13 Analysis of all deeper models based on the number of filters applied at the first level

The To-Down and Bottom-up approaches are alternatives to each other's as they represent the same U-Net based models with different approach of sorting and grouping. The Top-Down approach groups the models based on the number of filters applied to the first level of 3 levels U-Net and adding one level to the bottom most level to get a deeper mode and continue the adding process until reached the deepest level applicable for ex, group of models that start with 64 filters applied at the top most level contains the models (64-256, 64-512, 64-1024, 64-2048) while models started with 8 filters contains the models (8-32, 8-64, 8-128, 8-256, 8-512, 8-1024, 8-2048) and so on. The Bottom-Up approach categorized the models based on the number of filters that can be applied at the deepest level of 3 levels U-Net and getting the deeper models by adding a level to the top most level of the U-Net and continue until reaching the highest level applicable e.g. the category of models started with 1024 filters at the deepest level after adding all applicable levels are (256-1024, 128-1024, 64-1024, 32-1024, 16-1024, 8-1024) while the category of models started with 128 filters at the deepest levels contains the models (32-128, 16-128, 8-128).

In this section, all models included in deeper U-Net experiments will be compared and full dataset in Table 4-13

Name	Test_Avg	Test_Avg_Aug	Training	Validation	Parameters	Levels	Min Image size
8--32	**19.08%**	**12.78%**	73.29%	**19.95%**	29,321	3	64
8--64	41.33%	36.97%	76.61%	75.02%	120,681	4	32
8--128	34.09%	22.44%	74.82%	48.72%	485,673	5	16
8--256	42.16%	46.11%	**56.25%**	76.52%	1,944,745	6	8
8--512	**78.56%**	**73.89%**	**86.32%**	**89.89%**	**7,779,241**	**7**	**4**
8--1024	34.82%	29.09%	82.09%	57.89%	31,113,641	8	2
8--2048	74.50%	**73.20%**	82.54%	82.93%	142,444,073	9	1
16--64	44.47%	**35.87%**	**82.01%**	**71.83%**	116,753	3	64
16--128	**40.41%**	54.83%	83.97%	78.58%	481,745	4	32
16--256	52.60%	67.13%	84.59%	85.00%	1,940,817	5	16
16--512	**40.79%**	39.95%	93.00%	77.85%	7,775,313	6	8
16--1024	**69.81%**	**70.80%**	**95.07%**	**90.23%**	**31,109,713**	**7**	**4**
16--2048	45.36%	67.05%	93.24%	85.31%	124,440,145	8	2
32--128	45.08%	38.81%	**84.54%**	80.05%	465,953	3	64
32--256	**41.84%**	**34.11%**	89.73%	**64.30%**	1,925,025	4	32
32--512	46.49%	60.07%	96.50%	**98.00%**	7,759,521	5	16
32--1024	66.42%	**70.45%**	96.91%	92.72%	**31,093,921**	**6**	**8**
32--2048	**68.59%**	64.62%	**97.27%**	86.96%	**124,424,353**	**7**	**4**
64--256	**45.57%**	**39.25%**	**89.33%**	**72.55%**	1,861,697	3	64
64--512	49.96%	44.23%	96.15%	83.52%	7,696,193	4	32
64--1024	58.73%	75.93%	**97.40%**	91.50%	31,030,593	5	16
64--2048	**60.24%**	**80.39%**	**97.81%**	**93.68%**	**124,361,025**	**6**	**8**
128--512	**50.86%**	**52.50%**	**94.88%**	**85.72%**	7,442,561	3	64
128--1024	60.98%	66.26%	**97.81%**	**88.82%**	**30,776,961**	**4**	32
128--2048	**69.61%**	**74.63%**	**97.51%**	87.99%	**124,107,393**	**5**	**16**
256--512	**28.74%**	**41.02%**	**86.78%**	**78.33%**	6,427,393	2	128
256--1024	**53.23%**	**70.20%**	**95.75%**	**83.64%**	**29,761,793**	**3**	**64**
256--2048	N/A	N/A	N/A	N/A	N/A	N/A	N/A

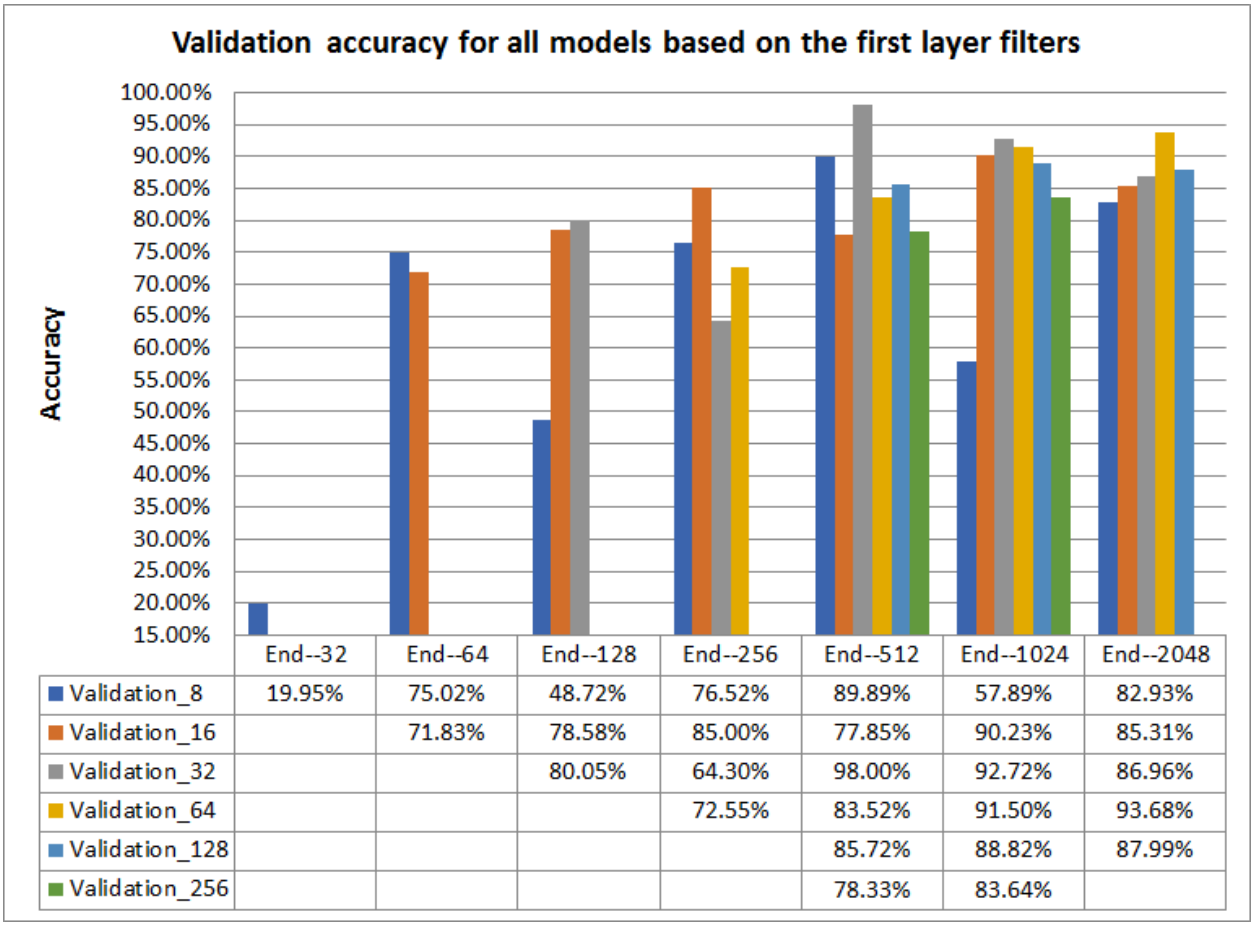
Table 4-13 All models with same start and different number of levels compared with all possible starts. Within each group,Highest accuracy (Green) lowest accuracy (Red)



	End--32	End--64	End--128	End--256	End--512	End--1024	End--2048
Training_8	73.29%	76.61%	74.82%	56.25%	86.32%	82.09%	82.54%
Training_16		82.01%	83.97%	84.59%	93.00%	95.07%	93.24%
Training_32			84.54%	89.73%	96.50%	96.91%	97.27%
Training_64				89.33%	96.15%	97.40%	97.81%
Training_128					94.88%	97.81%	97.51%
Training_256					86.78%	95.75%	

Figure 4-13 Training accuracy for models included in based on the first layer filters approach

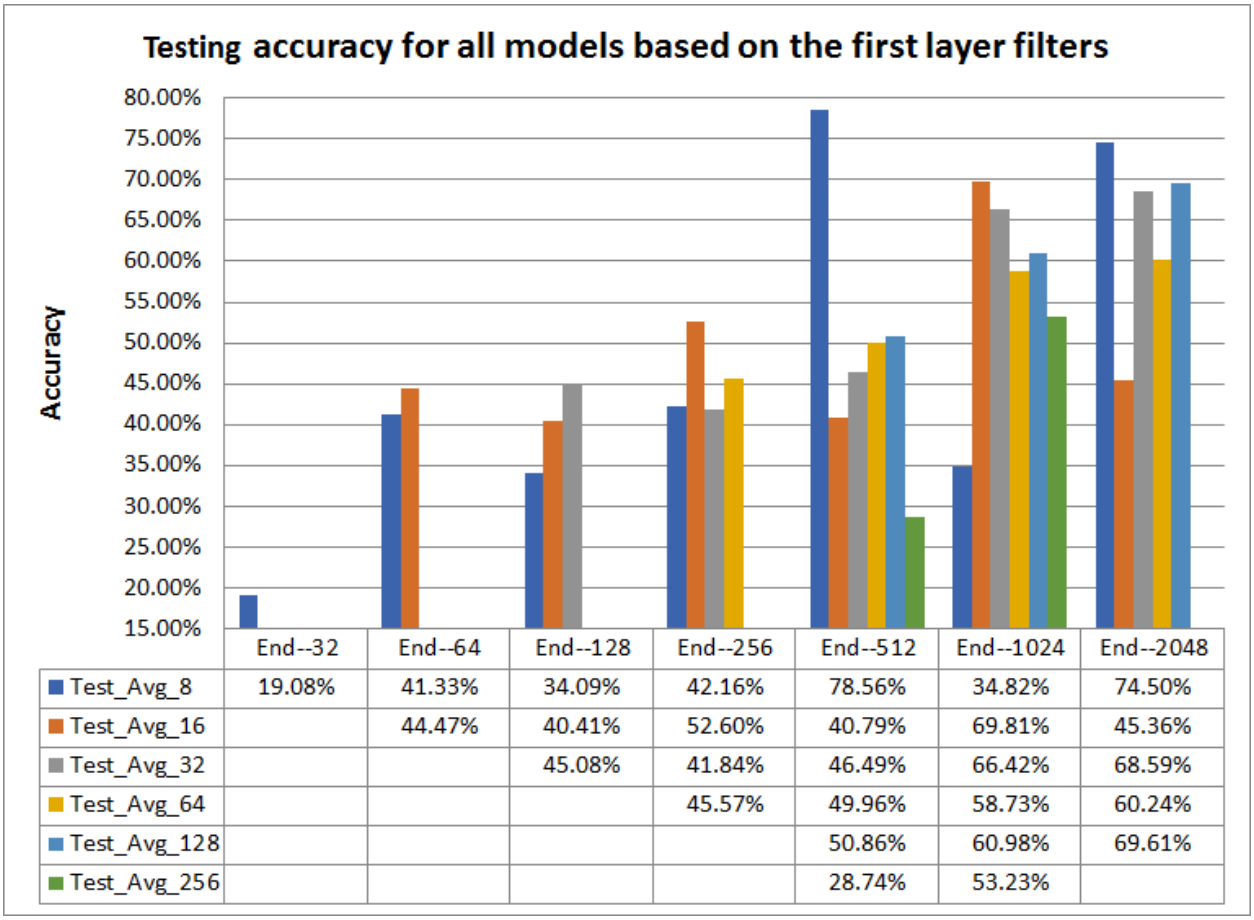
The **training** accuracy for the models start with 8 filters applied on the first layer kept fluctuating and recorded the minimum at (8—256) and the maximum with model (8—512) then continue fluctuating. The remaining 5 of 6 categories of models with different start (16, 32, 64, 128, and 256) have the same curve behavior with increasing the depth. The training accuracy increased while increasing the depth until reach the maximum when the deepest level use filters (1024, 2048, 2048, 1024, 1024). Only models started with 16 or 128 filters after reached the maximum at filters (1024) slightly decreased when using 2048 models. The highest 4 maximum values achieved for models (128-1024, 64-2048, 128-2048, 64-1024) in sequence, 2 of them start with 64 and 2 start with 128, while the 4 minimum values achieved for models started with 8 filters (8-256, 8-32, 8-128, 8-64)in sequence. The number of models achieved the maximum accuracy divided into 1 model ended with 512, 3 models ended with 1024 and 2 models ended with 2048.Figure 4-13 Table 4-13



	End--32	End--64	End--128	End--256	End--512	End--1024	End--2048
Validation_8	19.95%	75.02%	48.72%	76.52%	89.89%	57.89%	82.93%
Validation_16		71.83%	78.58%	85.00%	77.85%	90.23%	85.31%
Validation_32			80.05%	64.30%	98.00%	92.72%	86.96%
Validation_64				72.55%	83.52%	91.50%	93.68%
Validation_128					85.72%	88.82%	87.99%
Validation_256					78.33%	83.64%	

Figure 4-14 Validation accuracy for models included in based on the first layer filters approach

For the **validation** accuracy, 3 of 6 categories of the models started with (64, 128, 256) increase the accuracy with increasing the model depth to reach the maximum at the deepest model with filters (2048, 1024, 1024) with exception for the model start with 128 the decrease after the maximum. The other 3 categories that start with filters (8, 16, 32) increased with the deeper models and decreased with models with filters (128, 512, 256) then increased to reach the maximum at models ends with filters (512, 1024, 512) in sequence then decreased with the deeper models. The highest 4 maximum values achieved for models (32-512, 64-2048, 32-1024, 64-1024) in sequence while the 4 minimum values achieved for models started with 8 filters (8-32, 8-12, 8-1024, 32-256) in sequence. The number of models achieved the maximum accuracy divided into 2 model ended with 512, 3 models ended with 1024 and 1 model ended with 2048. Figure 4-14 Table 4-13

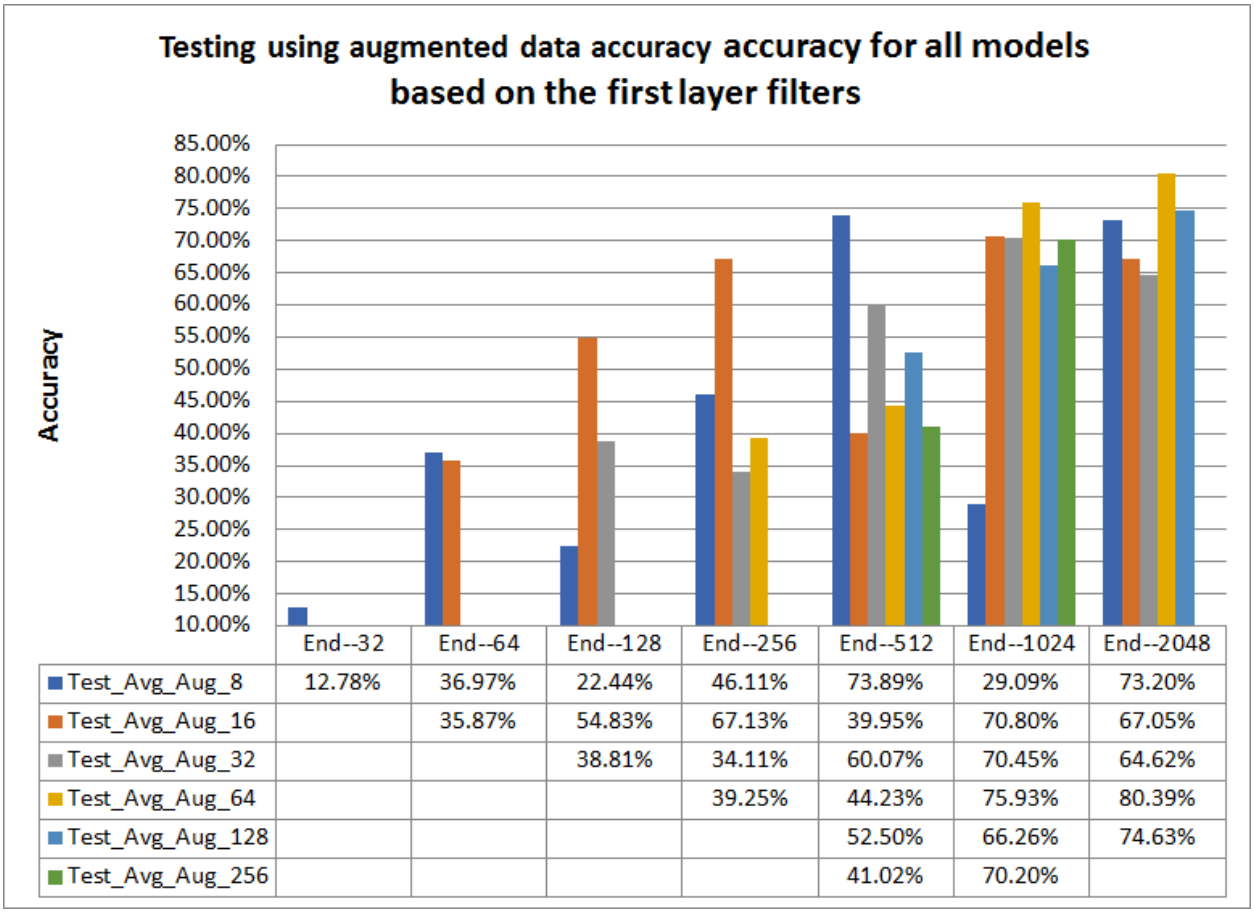


	End--32	End--64	End--128	End--256	End--512	End--1024	End--2048
Test_Avg_8	19.08%	41.33%	34.09%	42.16%	78.56%	34.82%	74.50%
Test_Avg_16		44.47%	40.41%	52.60%	40.79%	69.81%	45.36%
Test_Avg_32			45.08%	41.84%	46.49%	66.42%	68.59%
Test_Avg_64				45.57%	49.96%	58.73%	60.24%
Test_Avg_128					50.86%	60.98%	69.61%
Test_Avg_256					28.74%	53.23%	

Figure 4-15 Testing (Test_Avg) accuracy for models based on the first layer filters

For **Testing** factor (Test_Avg), 4 of 6 categories of models that started with filters (32, 64, 128, 256) increased the accuracy with increasing the depth to reach the maximum at the deepest applicable model with filters (2048, 2048, 2048, 1024) in sequence, while the models started with 32 filters has a drop when using (32-256) filters. The remaining 2 of 6 categories that start with filters (8, 16) fluctuating while going with deeper models and reached the maximum when used filters at the deepest level are (512, 1024) in sequence. The highest 4 maximum values achieved for models (8-512, 8-2048, 16-1024, 128-2048) in sequence while the 4 minimum values achieved for models started with 8 filters (8-32, 8-512, 8-128, 8-1024) in sequence. The number of models that achieved the maximum accuracy in each group can be categorized based on the number of filter applied at the

deepest level into 1 model ended with 512, 2 models ended with 1024 and 3 models ended with 2048. Table 4-13



	End--32	End--64	End--128	End--256	End--512	End--1024	End--2048
Test_Avg_Aug_8	12.78%	36.97%	22.44%	46.11%	73.89%	29.09%	73.20%
Test_Avg_Aug_16		35.87%	54.83%	67.13%	39.95%	70.80%	67.05%
Test_Avg_Aug_32			38.81%	34.11%	60.07%	70.45%	64.62%
Test_Avg_Aug_64				39.25%	44.23%	75.93%	80.39%
Test_Avg_Aug_128					52.50%	66.26%	74.63%
Test_Avg_Aug_256					41.02%	70.20%	

Figure 4-16 Testing using augmented data (Test_Avg_Aug) accuracy for models based on the first layer filters

For **Test_Avg_Aug** factor (Testing using augmented data), 3 of 6 categories of models that started with filters (64, 128, 256) increased the accuracy with increasing the depth to reach the maximum at the deepest applicable model with filters (2048, 2048, 1024) in sequence. The remaining 3 of 6 categories that start with filters (8, 16, 32) increased the accuracy with increasing the depth to reach the maximum at models with depth (512, 1024, 1024) in sequence with a drop of the accuracy during the increasing at the models (8-128, 16-512, 32-512) in sequence. The highest 5 maximum values achieved for models (64-2048, 64-1024, 128-2048, 8-512, 8-2048) in sequence while the 5 minimum values achieved for models started with 8 filters (8-32, 8-128, 8-1024, 32-512,

16-64)in sequence. The number of models achieved the maximum accuracy divided into 1 model ended with 512, 3 models ended with 1024 and 2 models ended with 2048. Figure 4-16,Table 4-13

Over all maximum values for Validation, Test_Avg, Test_Avg_Aug factors recorded for models (32-512, 8-512, 64-2048) with values (98.00%, 78.56%, 80.39%) while the maximum accuracy for Training is (97.81%) that achieved by 2 models (64-2048, 128-1024). The model (64-2048) recorded the maximum accuracy for 2 of 4 factors (Training, Test_Avg_Aug).

For **Top-Down** approach, the total models generated by this approach are 27 models divided into 6 categories based on the number of filters at the first top level (8, 16, 32, 64, 128, 256). The maximum accuracy for all 4 factors (Training, Validation, Test_Avg, Test_Avg_Aug) over all the 27 models recorded for models that ended with number of filters at the deepest level are one of three values(**512, 1024, 2048**). Over all achieved 24 maximum accuracies for the 4 factors, 45.83% recorded for models ended with 1024 filters and 33.83% recorded for models with 2048 filters at the deepest level while the remaining 20.83% recorded for models with 512 filters at the deepest level. For Testing_avg and Test_Avg_Aug, 16.66% of the achieved maximum accuracies recorded for models that ended with512 filters while both models with 1024 and 2048 recorded 41.66% of the achieved maximum values.

The starting image size for the Deeper models investigation is 256*256 and will be decreased by 50% after each level on the contracting path because of the maxpooling layer with size (2,2) that limited the number of models would be produced where the minimum image size that can be reached is 1*1. From 24 maximum values achieved for 4 factors over the 6 categories, 41.66% recorded for models with 7 levels deep and minimum image size reached 4*4 while 20.83% recorded for models with 6 levels deep and minimum image

size 8*8. The remaining maximum values recorded for models with depth (3, 5, 4) levels with minimum image size (64*64, 16*16, 32*32) in sequence with the percentage (16.66%, 12.5%, 8.33%) of the maximum values respectively.

For **Bottom-Up** approach, the total models generated by this approach are 27 models divided into 6 categories based on the number of filters at the deepest level (64, 128, 256, 512, 1024, 2048). The maximum accuracy for all 4 factors (Training, Validation, Test_Avg, Test_Avg_Aug) over all the 27 models recorded for models that started with number of filters at the first level are one of five values(8, 16, 32, 64, 128). Over all achieved 24 maximum accuracy for the 4 factors the models started with number of filters (8, 16, 32, 64, 128) with percentage (20.83%, 33.33%, 25%, 16.66%, 4.16%) respectively. For Testing_avg and Test_Avg_Aug, 50% of the achieved maximum accuracies recorded for models that started with 16 filters while models started with 8 and 64 filters recorded 33.33% and 16.66% of the achieved maximum values respectively.

From 24 maximum values achieved for 4 factors for each of the 6 categories, 25% recorded for each of the models with (4 and 5) levels deep and minimum image size reached (32*32 and 16*16) respectively while the models with (3 and 6) levels deep and minimum image size (64*64 and 8*8) recorded 16.66% of the total number of models that achieved one of the maximum values. The remaining maximum values recorded for models with depth (7 and 9) levels with minimum image size (4*4 and 1*1) in sequence with the percentage (12.5% and 4.16%) of the maximum values respectively.

Name	Test_Avg	Test_Avg_Aug	Training	Validation	Parameters	Levels	Min Image size
8--512	**78.56%**	**73.89%**	**86.32%**	**89.89%**	7,779,241	7	4
16--1024	**69.81%**	**70.80%**	**95.07%**	**90.23%**	31,109,713	7	4
32--512	46.49%	60.07%	96.50%	**98.00%**	7,759,521	5	16
32--1024	66.42%	**70.45%**	96.91%	92.72%	31,093,921	6	8
32--2048	**68.59%**	64.62%	**97.27%**	86.96%	124,424,353	7	4
64--2048	**60.24%**	**80.39%**	**97.81%**	**93.68%**	124,361,025	6	8
128--1024	60.98%	66.26%	**97.81%**	**88.82%**	30,776,961	4	32
128--2048	**69.61%**	**74.63%**	**97.51%**	87.99%	124,107,393	5	16
256--1024	**53.23%**	**70.20%**	**95.75%**	**83.64%**	29,761,793	3	64

Table 4-14 Maximum accuracy achieved by models based on the number of the first layer filters for Training, Validation, Testing, and testing using augmented data factors (Top-Down approach). Highest accuracy (Green) within each group

Name	Test_Avg	Test_Avg_Aug	Training	Validation	Parameters	Levels	Min Image size	Batch size
64--2048	60.24%	**80.39%**	**97.81%**	**93.68%**	124,361,025	6	8	8
8--2048	**74.50%**	73.20%	82.54%	82.93%	142,444,073	9	1	32
128--1024	60.98%	66.26%	**97.81%**	88.82%	30,776,961	4	32	8
64--1024	58.73%	**75.93%**	97.40%	91.50%	31,030,593	5	16	16
32--1024	66.42%	70.45%	96.91%	**92.72%**	31,093,921	6	8	32
16--1024	**69.81%**	70.80%	95.07%	90.23%	31,109,713	7	4	32
32--512	46.49%	60.07%	**96.50%**	**98.00%**	7,759,521	5	16	16
8--512	**78.56%**	**73.89%**	86.32%	89.89%	7,779,241	7	4	32
32--256	41.84%	34.11%	**89.73%**	64.30%	1,925,025	4	32	32
16--256	**52.60%**	**67.13%**	84.59%	**85.00%**	1,940,817	5	16	32
32--128	**45.08%**	38.81%	**84.54%**	**80.05%**	465,953	3	64	32
16--128	40.41%	**54.83%**	83.97%	78.58%	481,745	4	32	32
16--64	**44.47%**	35.87%	**82.01%**	71.83%	116,753	3	64	32
8--64	41.33%	**36.97%**	76.61%	**75.02%**	120,681	4	32	32

Table 4-15 Maximum accuracy achieved by models based n the number of the deepest layer's filters for Training, Validation, Testing, and testing using augmented data factors(Bottom-up approach). Highest accuracy (Green) within each group

4.1.1.14 Discussion

The results and analysis in the previous sub-sections highlighted the following key findings. 1- Within the each group of models either stating with the same number of filters at the top level or ending with same number of filters at the deepest level, the accuracy increase with increasing the number of levels in other words, **the accuracy increase while going deeper with the models. 2-** The best recorded accuracy achieved by the models ended with 512, 1024, 2048 filters at the deepest level. **3-**Tthe best performance recorded for models that apply convolutional process on image size 4*4 or 8*8 at the deepest level. **4-** Most of the top performer models consists of 5, 6, 7 levels and start with number of filters 8,16,32,64 at the first level. **5-** Models with 3 or 4 levels never achieve good accuracy even with large number of filters. **6-** Starting with small number of filters at the first level e.g. 8 or 16 will need high number of levels 7 or 9.

In line with the hypotheses, within the same group of models that have the same number of filters at the first level or the deepest level , the accuracy will increase with the model going deeper or the number of model's level increase. The reason is that, increasing the number of levels will increase the number of nodes and the number of trainable parameters hence increasing the number of filters that applied on the feature-maps while the feature maps size will decrease after each pooling layer.

The fact of increasing the accuracy is correlated with increasing the number of levels has some restrictions while applying across the models from different groups (different number of filters at the first level, or different number of filters at the deepest layer).

The study is limited to use original image size of 256*256 and 128*128 and could not use 512*512 because of the computational resource limitations especially with deeper models that use 2048 filters at the deepest model. There are some constraints due to the limitations of the image size to achieve high accuracy e.g. the minimum image size at the

deepest level should not be less than 4*4 or 8*8, levels should be within 5, 6, 7 levels and the other conditions listed as findings.

The results comply completely with results from the original paper of U-Net, Bridge Net, and W-Net [14] [16] [138] [139] where the U-Net model consists of 5 levels and start with number of filters 32 or 64 at the first level and ended with 512 or 1024 filters at the deepest level.

Some unexpected results for some models happened where the image size at the deepest level is very small e.g. 1*1 or very large 32*32 or 64*46 where the model has 3 levels starting with 8 filters, in such cases the number of filters at the deepest level is 32 or 64 which is very small to extract enough feature to train the model.

Some of the models could achieve better accuracy if created deeper by adding deeper layers e.g. the model of 3 levels with filters 256, 512, 1024 where the minimum image size at the deepest level is 64, the model assume to record higher accuracy if another levels with 2048 filters or more but that could not be tested because of the resource limitations.

4.1.1.15 Recommendations for deeper models based on U-Net structure

Based on the results of the models based on u-Net with variations of the depth (number of levels and layers) and number of filters applied at the first and last level the set of recommendations could be helpful for designing the U-Net based models. The image size used with these models is **256*256.**

- The number of filters applied at the first top level better to be one of these values (8, 16, 32, 64). If the filters less than 16 it would need more levels to get better accuracy.

- The number of levels should be within these values (5, 6, 7) levels that leads the minimum image size used at the deepest level to be within the values (32*32, 16*16, 8*8, 4*4).
- The number of filters to be applied at the deepest level should be one of these values (512, 1024, 2048), models with less than 512 filter at the deepest level will not achieve good accuracy.

4.1.2 Wider U-Net based models

In this section, U-Net model's accuracy would be compared against the stacked models of 2, 3, and 4 U-Nets based on two different image sizes 256*256 and 128*128.

According to the recommendations from the U-Net deeper models section, the implemented U-Net model has the original structure where the model consists of **5 levels** on the contraction path and **4 levels** on the expansion path. The number of filters applied started with 64 at the first level and ended with 1024 at the deepest (64, 128, 256, 512, and 1024). Another reason for using the U-Net with the original structure is to have a valid comparison with the state-of-the-art structure model structure.

4.1.2.1 Comparing U-net, 2, 3 and 4 stacked U-Net with image size 256*256

The accuracy of the models using image size 256*256 shows that, Validation accuracy started with the maximum value with one U-Net and kept decreasing with adding more U-Net to the model to reach the minimum when the model consist of 4 stacked U-Nets. The other three factors, training, testing **(Test_Avg)** and testing using augmented data (**Test_Avg_Aug**) have the same curve where starting with a medium value for one U-Net model then increased to reach the maximum accuracy with 2U-Nets then decreased to the minimum when using 4 U-Nets Figure 4-17. Stacked 2U-Nets model recorded the

maximum performance for all factors except validation where one U-Net recorded the maximum accuracy. The model with 4 stacked U-Net recorded the minimum accuracy for all factors Table 4-166

Models	Training	Validation	Test_Avg	Test_Avg_Aug
U-Net	97.40%	**91.50%**	58.73%	75.93%
Stacked 2 U	**97.44%**	90.36%	**77.96%**	**77.95%**
Stacked 3 U	96.68%	89.27%	56.30%	75.04%
Stacked 4 U	**4.71%**	**10.52%**	**29.45%**	**36.32%**

Table 4-16 U-net, 2, 3 and 4 stacked U-Net with image size 256*256. Highest accuracy (Green) lowest accuracy (Red)

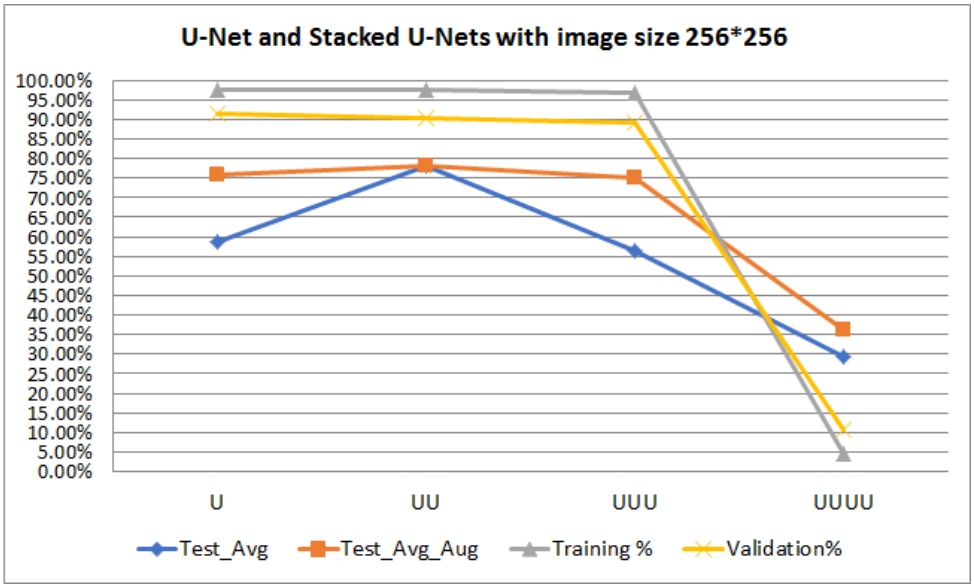


Figure 4-17 Comparing U-Net model with stacked 2, 3, and 4 U-Nets models using image size 256*256

4.1.2.2 Comparing U-net with 2, 3 and 4 stacked U-Net with image size 128*128

Validation and Test_Avg factors started with the maximum accuracy at the model using one U-Net then keep decreasing with stacking more U-Nets to reach the minimum at 4U-Nets and 3U-Nets respectively. Training has the opposite directions where started with minimum at one U-Net and increased to the maximum at 4U-Nets while Test_Avg_Aug started with medium accuracy with one U-Net then keep decreasing to the minimum at 3U-

Net model then suddenly increased to the maximum at 4U-Net model Figure 4-18. The 4U-Nets model recorded maximum accuracy for training and Test_Avg_Aug while recorded minimum of validation. The one U-Net model recorded maximum accuracy for validation and Test_Avg but recorded minimum with training while the 3U-Net model recorded minimum for both Test_Avg and Test_Avg_Aug Table 4-17.

Models	Training %	Validation%	Test_Avg	Test_Avg_Aug
U-Net	**96.41%**	**91.91%**	77.20%	82.28%
Stacked 2 U	97.09%	88.97%	**78.28%**	80.37%
Stacked 3 U	97.08%	90.11%	**68.33%**	**71.88%**
Stacked 4 U	**97.21%**	**88.77%**	74.47%	**87.00%**

Table 4-17 U-net, 2, 3 and 4 stacked U-Net with image size128*128. Highest accuracy (Green) lowest accuracy (Red)

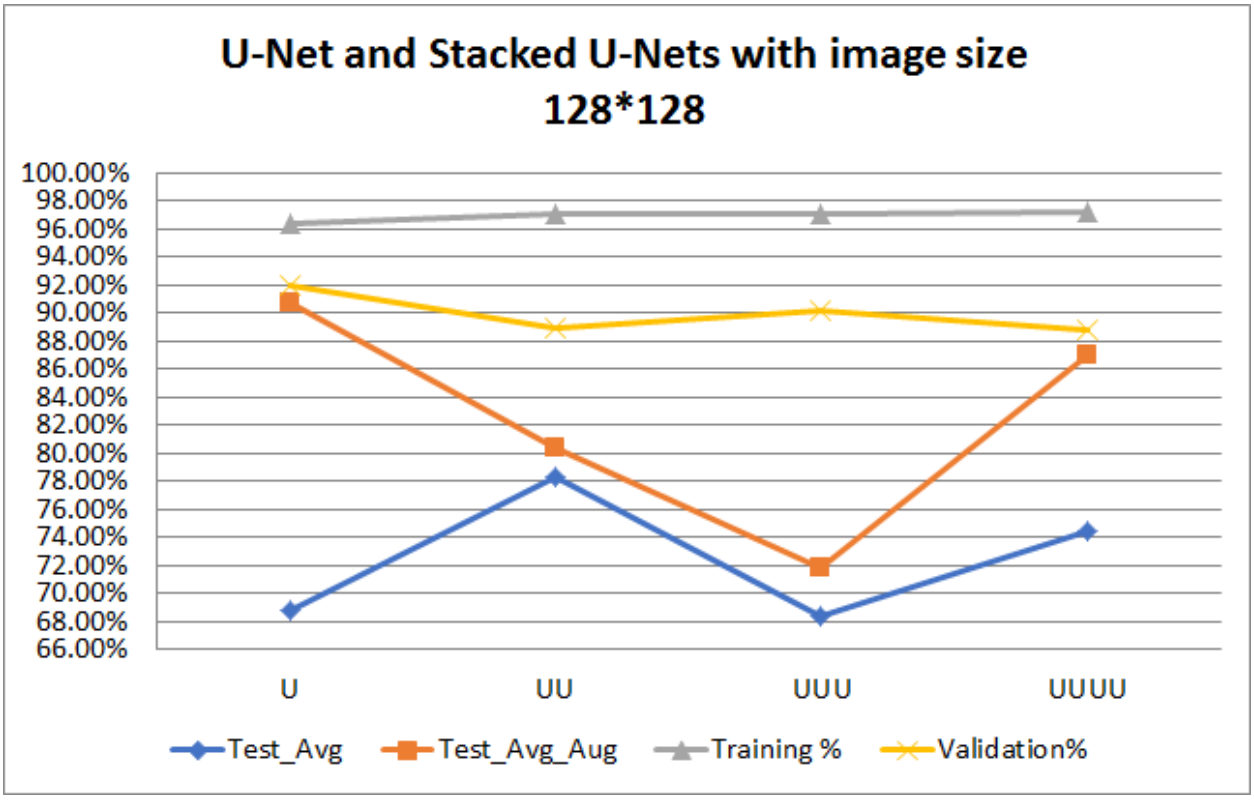


Figure 4-18 Comparing U-Net model against stacked 2, 3, 4 U-Nets using image size 128*128

4.1.2.3 Compare U-Net with two and 3 and 4 stacked U-Net with image size 256*256 versus 128*128

For the **Test_Avg** factor, using image size 128*128 recorded accuracy higher than using image size 256*256 for all models (U-Net, 2 U-Nets, 3U-Nets, and 4U-Nets) Figure 4-20. Testing using augmented data (**Avg_AUG_Test**) with image size 128*128 recorded higher accuracy than using image size 256*256 for all models except 3U-Net model recorded higher accuracy using 256*256 than using 128*128 Figure 4-21. The accuracy for **training** recorded higher values using image size 256*256 than using 128*128 for all models except for 3U-Nets is the opposite. Using image size 128*128 recorded higher accuracy than using 256*256 for Validation for all models except the 2U-Net model Figure 4-19 Table 4-18

Models	Image size	Training %	Validation%	Test_Avg	Test_Avg_Aug
U-Net	256*256	97.40%	91.50%	58.73%	75.93%
Stacked 2 U		97.44%	90.36%	77.96%	77.95%
Stacked 3 U		96.68%	89.27%	56.30%	75.04%
Stacked 4 U		4.71%	10.52%	29.45%	36.32%
U-Net	128*128	96.41%	91.91%	77.20%	82.28%
Stacked 2 U		97.09%	88.97%	78.28%	80.37%
Stacked 3 U		97.08%	90.11%	68.33%	71.88%
Stacked 4 U		97.21%	88.77%	74.47%	87.00%

Table 4-18 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256. Highest accuracy (Green) lowest accuracy (Red) within each group

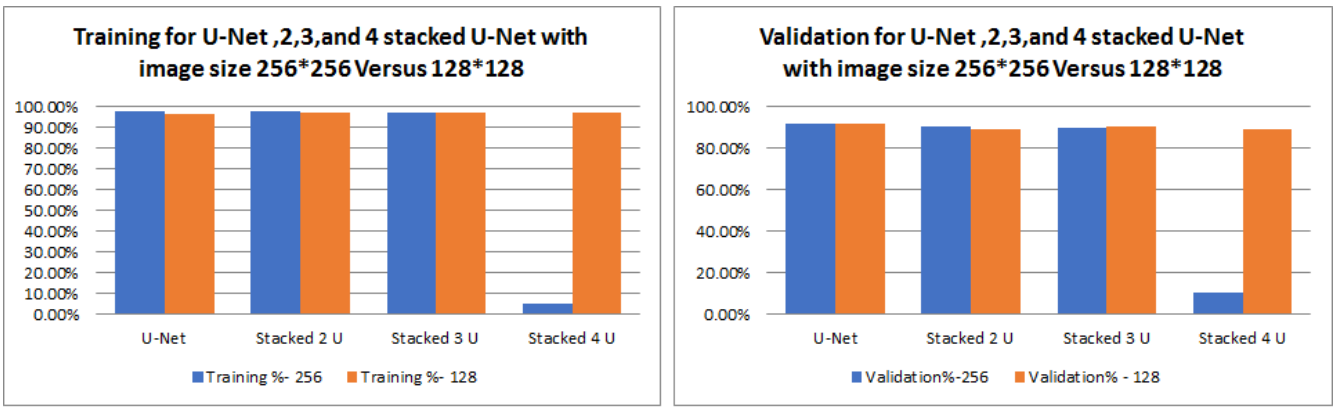


Figure 4-19 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right)

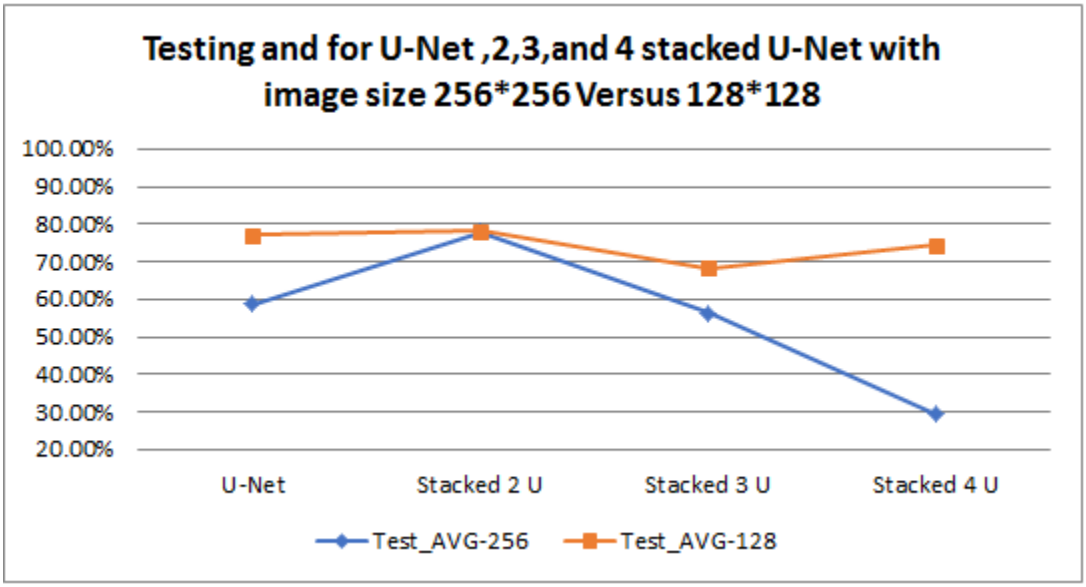


Figure 4-20 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right)

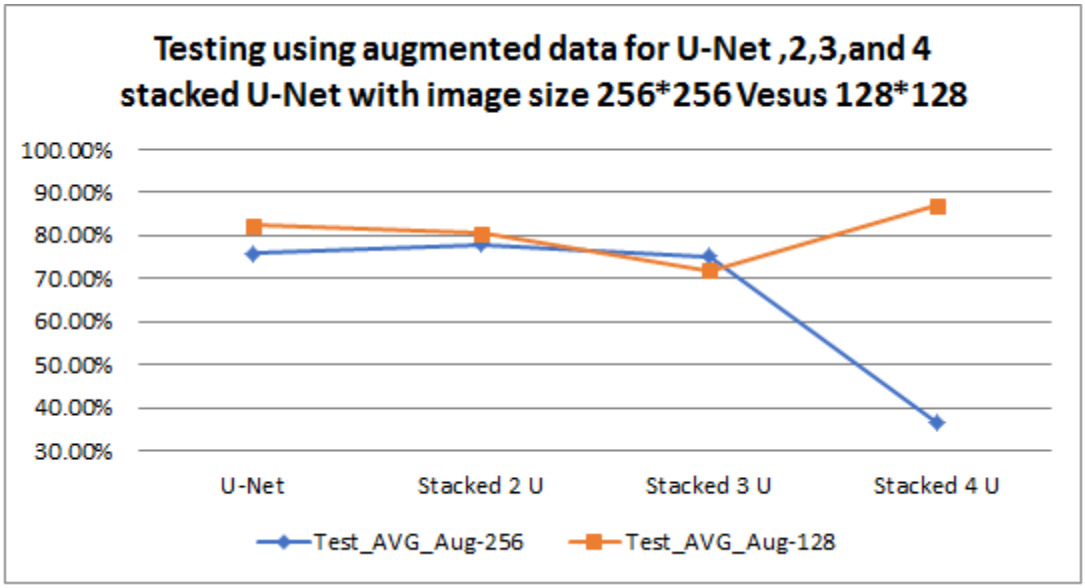


Figure 4-21 Models with one, two, three, and four U-Nets using image sizes 128*128 and 256*256, Training (left), validation (right)

4.1.2.4 Discussion

Generally, the image resolution shows difference in the segmented mask in terms of edge smoothness, the higher resolution gives higher smoothness in the segmented mask edge. Although all images used in training, validation and testing came with original size 512*512 with DICOM format, each model trained and tested using variation of image sizes (256*256 and 128*128) because of the computational resource limitations. The different image sizes will widen the range of the investigation of the effect of varying the image size on the accuracy of the model while getting wider. The study could not test a model with more than 4 u-nets because of the resource limitations. The study used U-Net with the original structure of 5 levels and fitter numbers for each level (64, 128, 256, 512, 1024) to be able to compare with the original U-Net.

The key findings of this section can be concluded into four points. **1-** With image size 256*256 the best accuracy achieved by 2 U-Nets model while the worst accuracy recorded with 4 U-Nets model. **2-** With image size 128*128 the best accuracy recorded for 2 and 4 U-Nets models and worst with 3 U-Nets. **3-** With image size 128*128 the accuracy is always higher compared to the same model using image size 256*256. **4-** The highest accuracy overall achieved by 2 U-Net models.

In line with the hypothesis of increasing the model performance by widen the model with additional U-Net, the 2 U-Net stacked model shows better performance than the original one U-Net. The results come in line with the proposed models in [139] [16] while the study did not use any bridging connection between the two U-Nets that will be under more investigation in section 4.1.3.

The results contradict with the hypothesis when adding a third and fourth U-Net because without any bridging between the u-nets. The reason is that, the loss of the features during

the convolutional layers and pooling with be increased and propagated from the U-Net to the next u-net leading to poor accuracy with 3U-Net and 4 U-Net stacked models.

When using image size 256*256 with U-Net model, the deepest level will apply convolutional process over the minimum image size 16*16 because each level has maxpooling layer with size 2*2 which decrease the image size by 50% after each level. The accuracy results showed that, **stacked 2U-Nets** recorded the best accuracy (77.96%, 77.95%) for (Avg_Test, and Avg_AUG_test) over the original U-Net (58.73%, 75.93%) and over all other structures using 3 or 4 stacked U-Nets, While using 4 stacked U-Nets recorded the minimum accuracy. the results indicates that, increasing the number of stacked U-Net that increased the number of trainable parameters can enhance the model accuracy until certain values which limited to 2 stacked U-Nets with around 62millions of parameters. increasing the number of U-Nets and parameters over 2 stacked will decrease the accuracy.

Although image size 128*128 recorded accuracy higher than using 256*256 for 75% of the models and factors, using images with higher resolution (256*256) will produce better shape of the liver segmented mask in terms of edge smoothness.

4.1.2.5 Recommendations for wider models based on U-Net structure

Based on the results of testing the U-Net based wider models, using the U-Net with original structure of 5 levels with number of filters (64, 128, 256, 512, 1024) shows that:

- In case of using image size of ***128*128, the study recommend to use 4 stacked U-Net*** even if the model will have significant number of trainable parameters and the output mask will not be smooth because of the low resolution.

- The study recommend to use ***25*256 image size with two stacked U-Net*** for better accuracy than the original one U-Net model even without any additional connections between the two U-Nets. The accuracy will be decreased with more than 2 U-Nets.
- It is better to keep the model structure based on two stacked 2U-Net with 256*256 because the results are more robust than using 128*128 image size.

4.1.3 Bridging and Skip-Connections

Based on the recommendations from the wider and deeper U-Net section and the original 2Bridged U-Net model for prostate segmentation [16], the first category of the models in this section are based on two U-Net stacked. Each U-Net in the model will have the same structure as recommended in deeper model section (5 Levels with number of filters 64-1024 or 32-512). The two stacked U-Net will be connected by the original bridge and skip connections and compared with similar models structure but with different the bridge and skip connections called (Modified and Compound connections). The second category consists of three stacked U-Net with the recommended structure in addition to various types of skip and bridge connections between the 3 U-Nets based on the Original, Modified and Compound style.

This section shows the results of the experiments that investigated the effect of using various connections between 2U-Nets and 3U-Nets based models on the model performance.

4.1.3.1 Two Bridged U-Nets

Three models introduced for training and testing within this section. All three models based on 2 Bridged U-Net structures. First model has the same structure as the **Original** 2D bridged U-Net [16] where the model have two connections to link the first and second U-Nets. Firstly, the output feature maps from the expansion path of the first U-Net (B2) concatenated to the inputs of contraction path of second U-Net (B3) while the second connections concatenated the output feature maps from the contraction path of the first U-Net (B1) to the inputs of the expansion path of the second U-Net (B4) Figure 4-22. The second model represent a **Modified** skip connection that replaces the bridge from B1 output

to B4 inputs (Red line) with a modified bridge connects between output B2 to the input of B4 (Green line). The third model called Compound model where it use both the connections from the previous models (Original and Modified). In the **Compound** model, the inputs of B4 concatenated with the output from B1 (Red) and the output from B2 (Green) in addition to the normal inputs based on the U-Net structure. All models used Dice Similarity Coefficient (DSC) as the loss function and combination of Rectified Linear Unit (ReLu) and Exponential Linear Unit (ELU) for activation Figure 4-22 Different types of bridged connections and skip connections with 2 U-Nets models. The training, validation, and Testing (Test_Avg) and testing using augmented data (Test_Avg_Aug) accuracies are measured using Dice similarity coefficient. Two types of U-Nets will be tested based on the number of filters applied on each level, the first U-Net with filters structure 32-512 (32, 64, 128, 256, 512, 256, 128, 64, 32) and the second U-Net used filters structure 64-1024 (64, 128, 256, 512, 1024, 512, 256, 128, 64) because these are the most commonly used in the literature [14], [16], [128], [138]

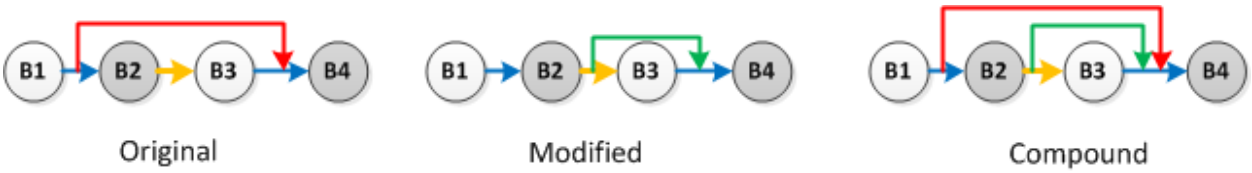


Figure 4-22 Different types of bridged connections and skip connections with 2 U-Nets models

Model	Filters	Training %	Validation%	Test_Avg	Test_Avg_Aug
Original	32-512	97.55%	70.65%	73.21%	60.31%
Modified	32-512	**97.85%**	67.02%	79.12%	60.10%
Compound	32-512	97.52%	**90.11%**	**89.88%**	**94.42%**
Original	64-1024	97.38%	87.75%	79.89%	75.34%
Modified	64-1024	98.12%	**92.50%**	81.37%	**83.68%**
Compound	64-1024	**98.12%**	91.13%	**83.03%**	78.36%

Table 4-19 Training , validation, and testing accuracy for original, Modified, and Compound models with filters structure 32-512 and 64-1024. Highest accuracy (Green) within each group

- **Different 2U-Net based models used model structure 32-512**

While the Compound model recorded the highest accuracy for validation, Test_Avg, and Test_Avg_Aug (90.11%, 89.88%, 94.42%), it recorded the minimum accuracy for training. The modified model got the highest training accuracy (97.85 %) even if the difference between maximum and minimum training accuracy is less than 0.5 % with values (97.55%, 97.85%, 97.52%) for original, Modified, and Compound model respectively. The modified model recorded accuracies higher than the Original Model for Training and Test_Avg while it recorded accuracy lower than original for validation and Test_Avg_Aug. Table 4-19 Figure 4-23

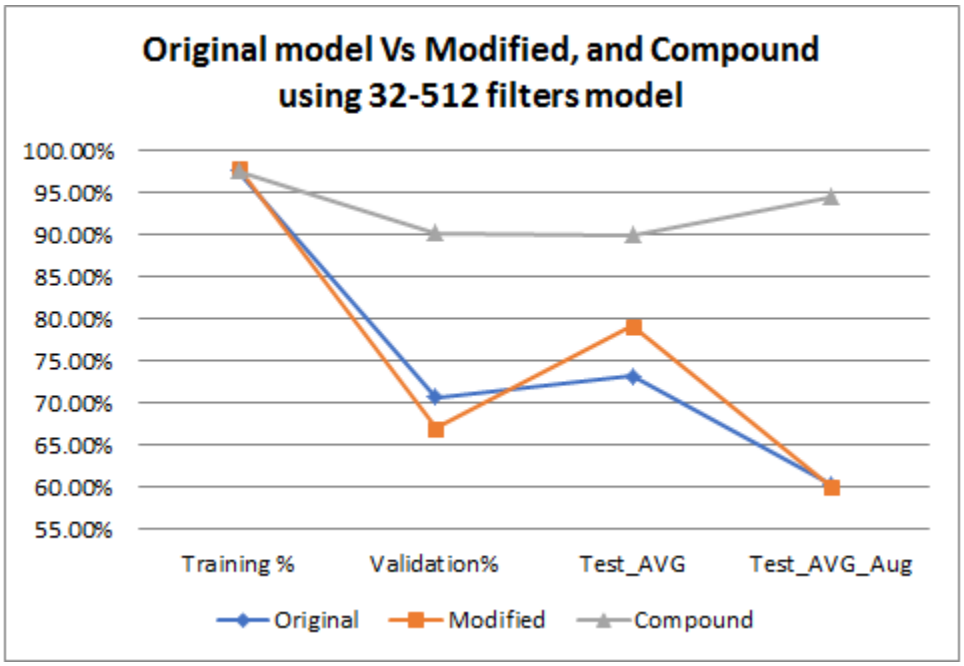


Figure 4-23 Original, Modified, Compound models with filters structure 32-512

- **Different 2U-Net based models used model structure (64-1024)**

The original model shows the lowest accuracy between the three models with all training, validation, Test_Avg, and Test_AVG_Aug. While Compound model recorded the maximum accuracy for Test_Avg (83.03%), the modified model achieved the best accuracy for validation and Test_Avg_Aug. Modified and Compound models have the same accuracy for training (98.12%). Figure 4-24 Table 4-19

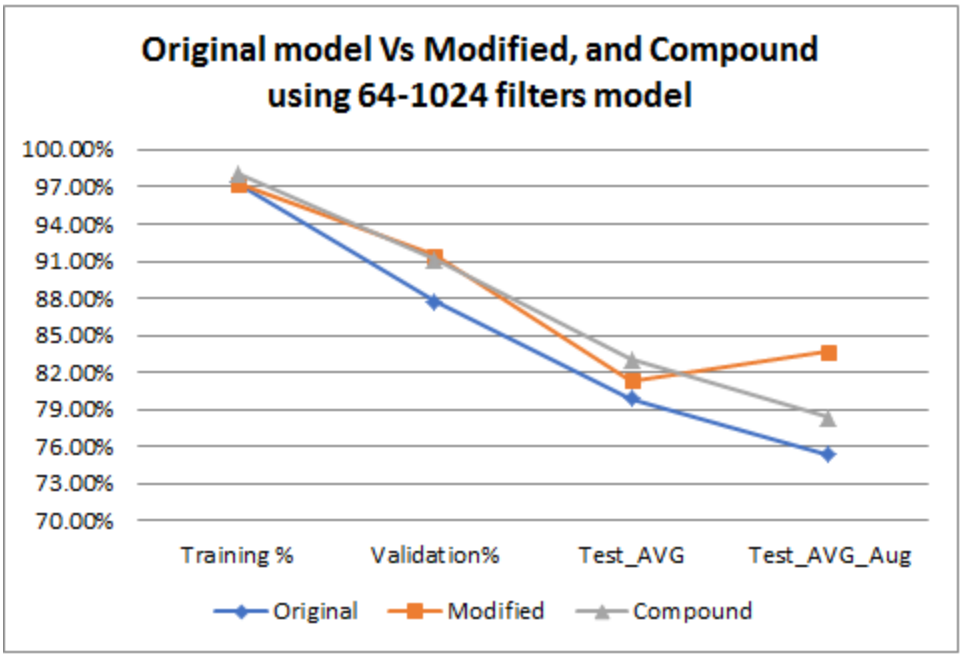


Figure 4-24 Original, Modified, Compound models with filters structure 64-1024

- **Comparing all models with filters 32-512 versus 64-1024**

Using the original and modified models with filters structure 64-1024 recorded higher accuracy than using 32-512 filters structure for all factors, training, validation, Test_Avg, and Test_Avg_Aug Figure 4-25 Figure 4-26. Using compound model with 32-512 filters structure achieved better accuracy for Test_Avg and Test_Avg_Aug while using 64-1024

filters recorded better accuracy for training and validation with less than 1% Figure 4-27 Table 4-19

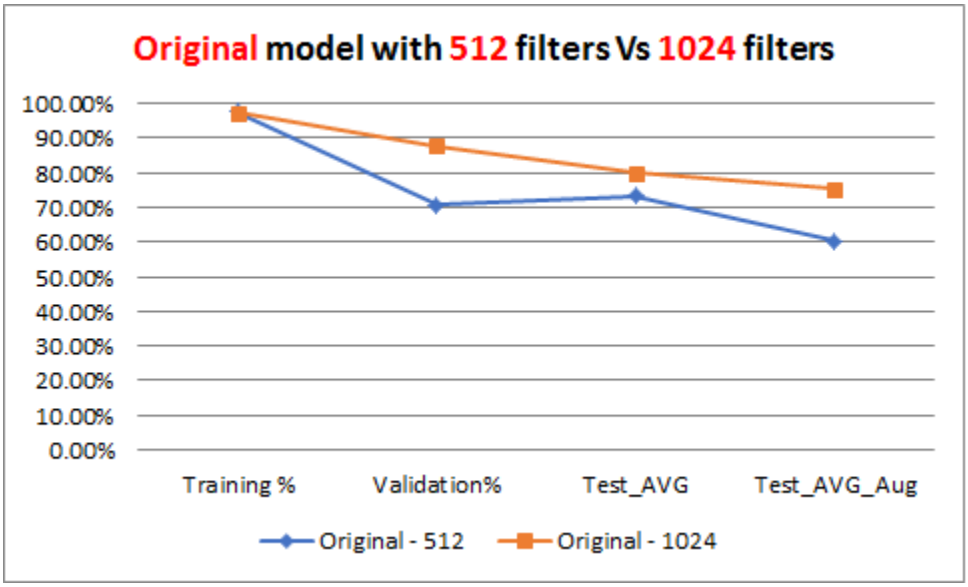


Figure 4-25 Original Model with filters' numbers 32-512 and 64-1024

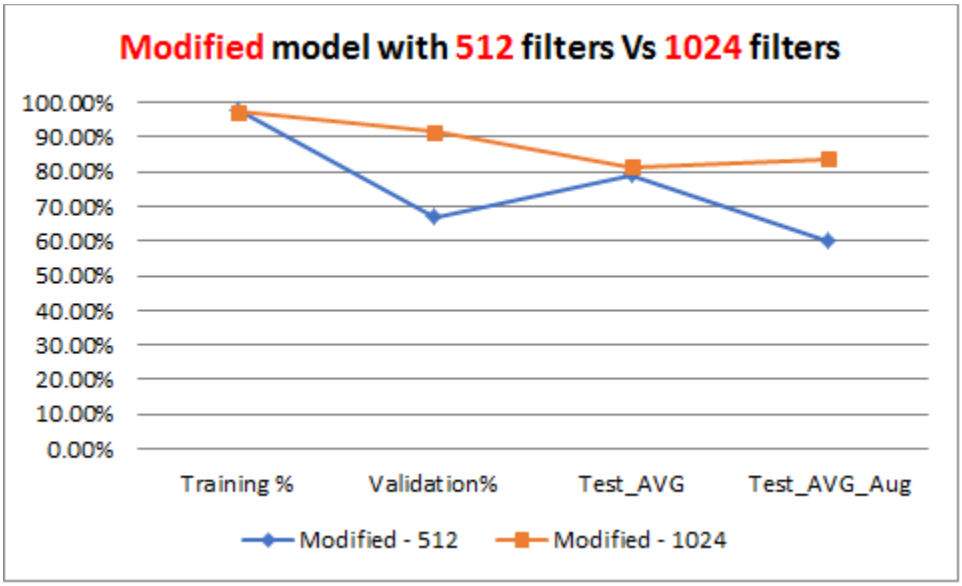


Figure 4-26 Modified Model with filters' numbers 32-512 and 64-1024

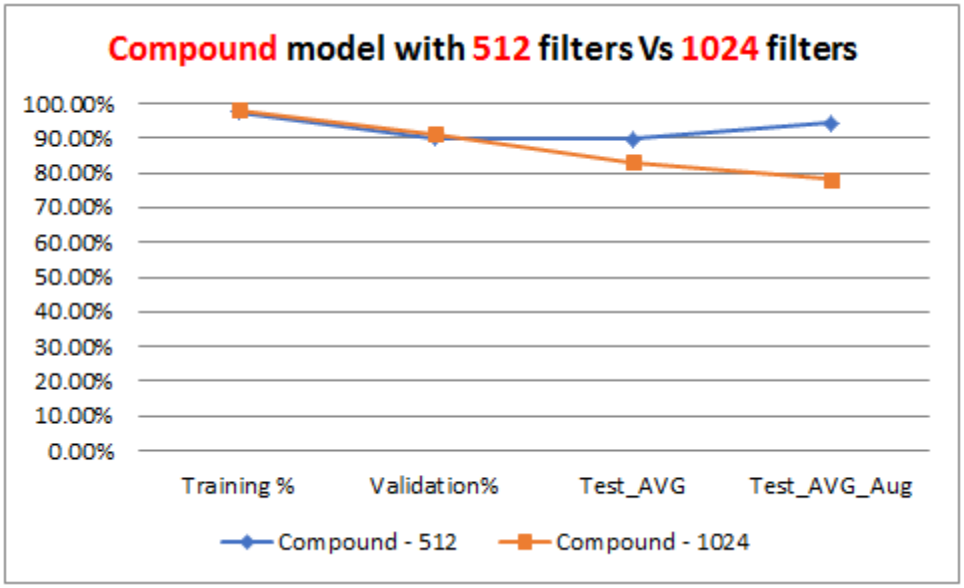


Figure 4-27 Compound model filters numbers 32-512 and 64-1024

4.1.3.2 Discussion of two bridged U-Nets

The model with 2 bridged U-Nets contains 4 paths, contracting and expansion path for each U-Net. Using the compound connections allowed the model to concatenate the output feature maps from the first three paths to the inputs of the last expansion path (4th path) then apply the de-convolutional process to the result of the concatenation process. The concatenation of all previous outputs minimized the effect of any lost features during the previous convolutional process at any level of the model.

The results in Table 4-19 can highlight the four key findings in this part of the research. **1-** The models based on the modified connections or Compound connections recorded higher accuracy than the original connections with both structures 32-512 and 64-1024. **2-** Using the filter structure 64-1024 recorded higher accuracy over the filter structure 32-512 except for compound connections. **3-** The models based on the compound connections recorded the best accuracy over original and modified connections for both filter

structures 32-512 and 64-1024. **4-** The best overall accuracy for testing with and without augmented data recorded for compound model using 32-512 filters structure.

The findings came in line with the main theory of U-Net that based on adding the global features from the contraction path of the U-Net to the feature-maps on the same level of the expansion path will overcome the loss of the feature that might happen during the convolution and pooling layers. The modified model and the compound models add extra bridging connections from the first U-Net to the final expansion path of the second U-Net that shows better performance over the original model.

Although most of the models in the literature used u-net with filter structure 32-512, the findings show that, using filter structure of 64-1024 recorded higher accuracy because the number of training parameters is higher than the parameters of the model with filters 32-512.

The findings shows significant enhancement of the compound model performance when using filters 32-512 over using filters 64-1024 although the model used less parameters that emphasis the high impact of using the compound connections over the other bridging connections.

The study recommend the 2Bridged U-Net model with compound connections with filters structure 32-512 for the 5 levels U-Net over all the other structures e.g. U-Net, original 2Bridged U-Net , Modified 2Bridged U-Nets with any other filter structure.

4.1.3.3 Three Bridged U-Nets

This section contains the results of testing the models based consists of 3 U-Net stacked with different types of connections. All the models in this section used 32-512 filter structure. The models divided into two main categories, first category is 3U-Net models with all possible connections that links first and second U-Nets then second and third U-Nets. The second category contains the same models as in first category with additional long connection between 1st and 3rd U-Nets.

In the first category, the models based on 3 U-Nets with bridged connections have different types of connections, for example the model named Original-Compound means the bridge between first U-Net and second U-Net used Original connections and the connections between the second U-Net and third U-Net used compound connections Figure 4-28. The models in the second category have similar connections as in the first category in addition to a long connection between first and third U-Nets. For example, the model called (Original-Compound-Long) means the connection between first and second U-Net used the original bridged connection and the connection between the second and third U-Nets used the compound connections, in addition to that, there is a long connection that concatenates the output of contracting path from the first U-Net (B1) to the inputs of the expansion path in the third U-Net (B6) (Blue line) Figure 4-29

Table 4-20 shows accuracy for Training, Validation, and Test_Avg and Test_Avg_Aug for 3U-Net based model with and without long bridge connections and the 2U-Net models based.

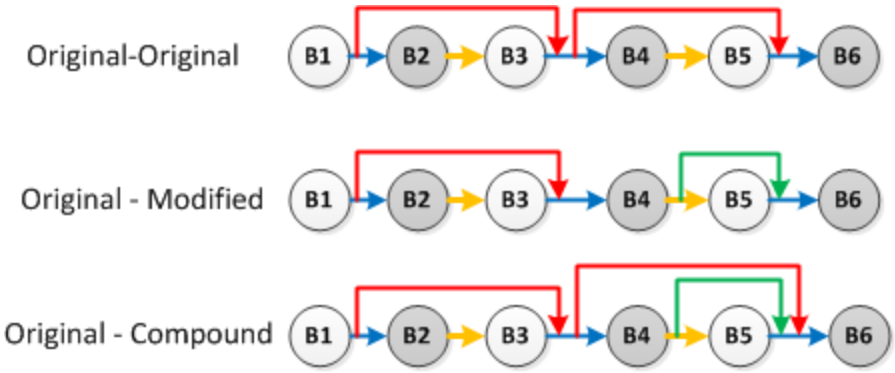


Figure 4-28 Representation for examples of models developed based on three U-Net models using the different connections style (Original, Modified, and Compound)

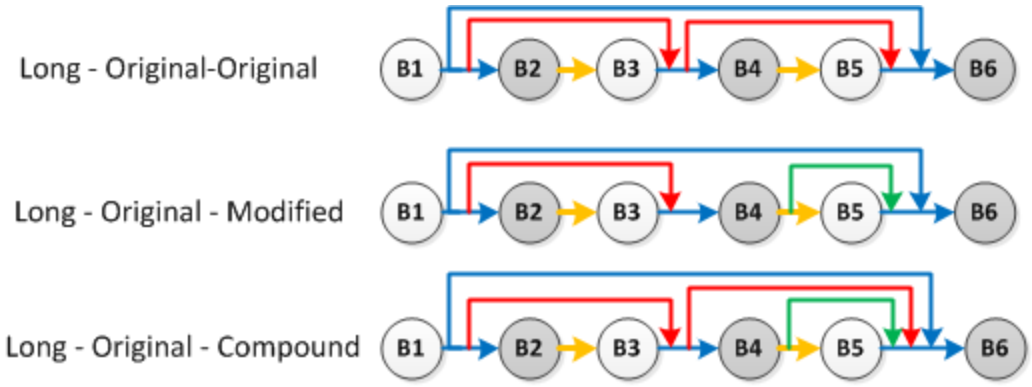


Figure 4-29 Representation for examples of 3 U-Net models with long connections between U-Nets (1 and 3)

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original	97.70%	84.62%	82.42%	67.22%
Original_Modified	97.58%	85.45%	91.52%	89.26%
Original_Compound	97.64%	86.84%	69.09%	61.48%
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Modified_Compound	97.59%	80.71%	78.08%	78.13%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	60.16%	62.77%
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	97.56%	89.68%	81.91%	78.19%
Modified_Original_Long	97.59%	62.84%	66.02%	**43.52%**
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Original_Long	97.65%	83.51%	**93.18%**	93.29%
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	97.69%	88.14%	78.30%	74.07%
Original	97.55%	70.65%	73.21%	60.31%
Modified	**97.85%**	67.02%	79.12%	60.10%
Compound	**97.52%**	**90.11%**	89.88%	**94.42%**

Table 4-20 Training, Validation, and testing for 3U-Net based model with and without long bridge connections, Highest accuracy (Green) lowest accuracy (Red) within each group

- **Models based on 3U-Net**

The Original_Modified model and the Compound_Compound model achieved the maximum accuracy for Test_Avg (91.52%) and Test_Avg_Aug (92.64%) respectively, while the maximum training and validation accuracy for Original_Original model (97.70%) and Modified_Modified model (87.27%) respectively. While the minimum accuracy for both validation (59.82%) and Test_Avg (60.416%) recorded for the Compound_modified model, the minimum training accuracy with 97.58% and Test_Avg_Aug (61.48%) achieved by Original_Modified and Original_Compound respectively. Even if the original_Modified model

achieved the minimum training accuracy (97.58%), it recorded the maximum testing accuracy with 91.52%. Table 4-21 Figure 4-30

Compound_Modified model recorded the lowest validation accuracy while all the other models recorded higher accuracy within the range from 87.27 to 84.6% Figure 4-31. The three models (Compound_Compound , Original_Modified , Compound_Original) recorded the highest 3 accuracies for Test_Avg_Aug with values 92.64%, 89.26%, 88.08% while the same 3 models recorded the highest 3 accuracies for Test_Avg but with different order(Original_Modified, Compound_Compound , Compound_Original) with values 91.52%, 89.76%, 87.99% in sequence Figure 4-32. The difference between the maximum and 3rd maximum is around 4%. For training accuracy, the difference between the maximum 97.70% and the minimum 97.58% is 0.12%. Figure 4-33

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Original_Modified	**97.58%**	85.45%	**91.52%**	89.26%
Original_Compound	97.64%	86.84%	69.09%	**61.48%**
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	**87.27%**	78.99%	70.67%
Modified_Compound	97.59%	80.71%	78.08%	78.13%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	62.77%
Compound_Compound	97.59%	85.39%	89.76%	**92.64%**

Table 4-21 Models based on 3U-Net, Highest accuracy (Green) lowest accuracy (Red)

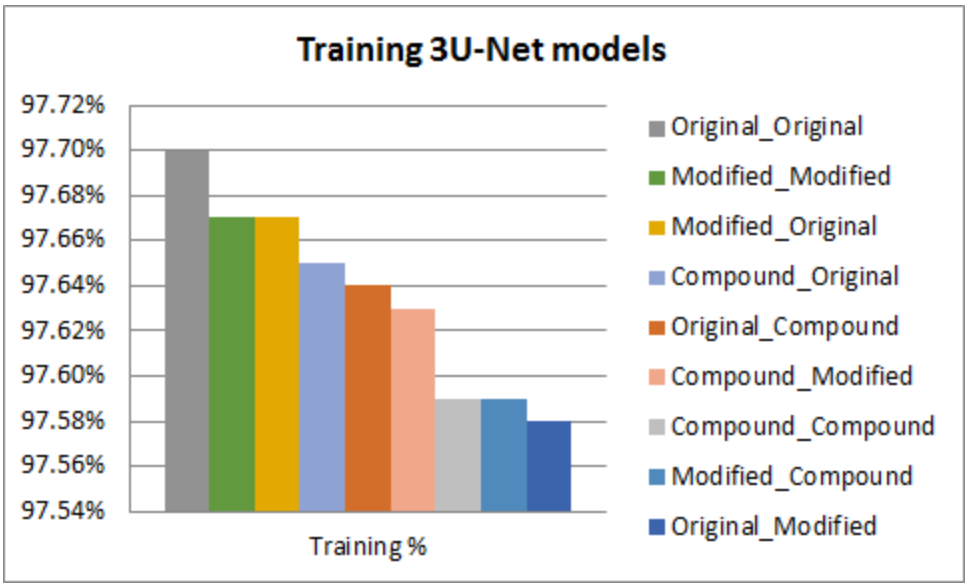


Figure 4-30 Training accuracy for all models based on 3U-Net models

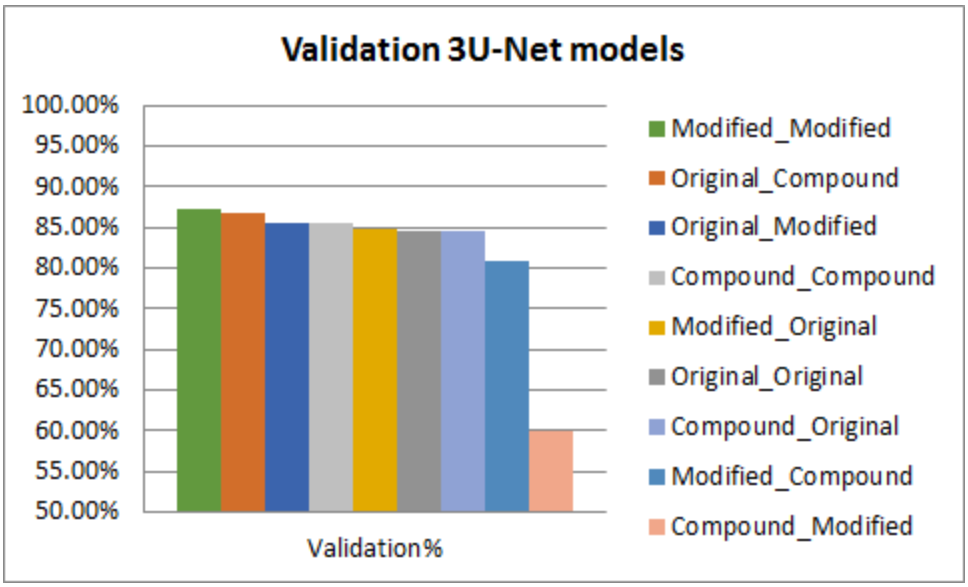


Figure 4-31 Validation accuracy for all models based on 3U-Net models

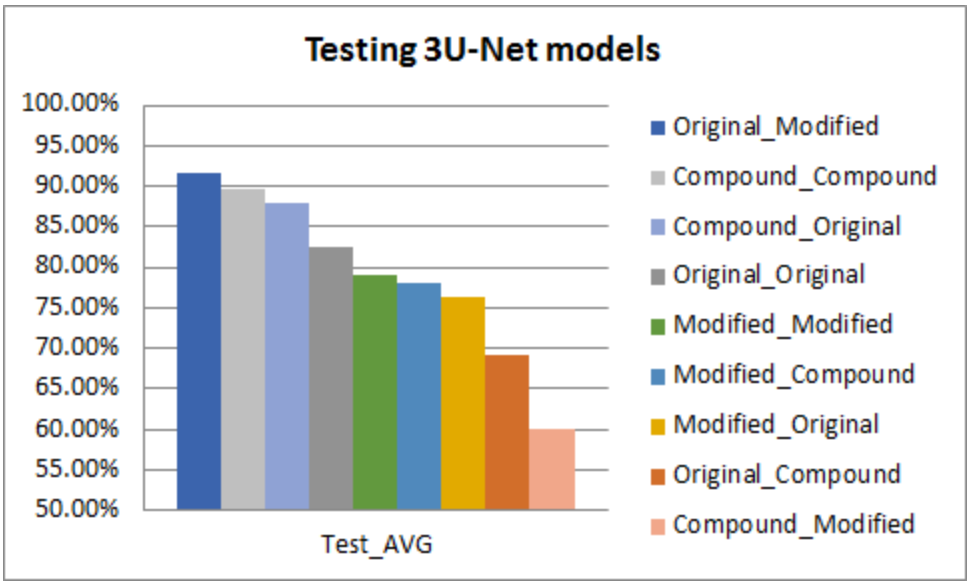


Figure 4-32 Testing (Test_Avg) accuracy for all models based on 3U-Net models

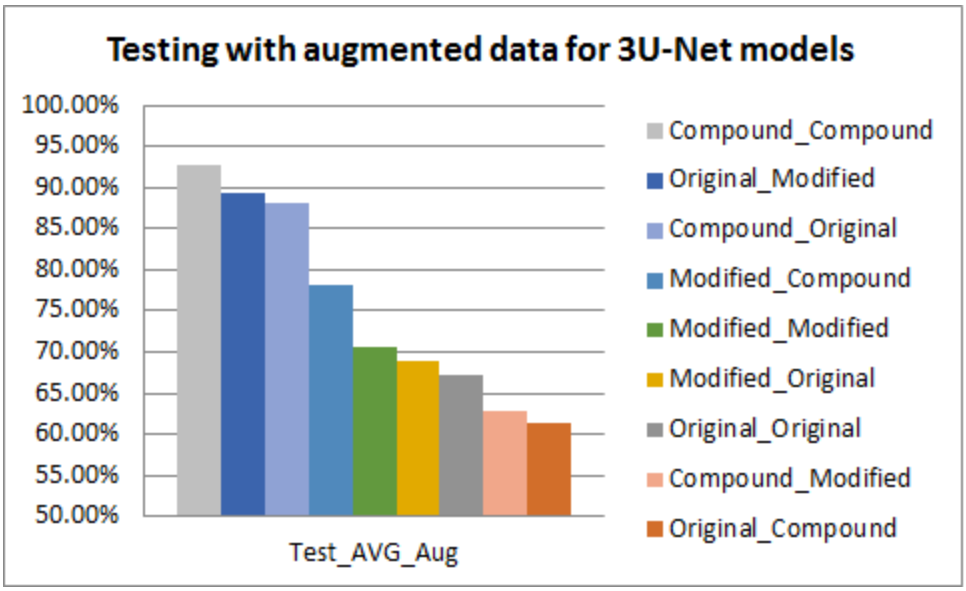


Figure 4-33 Testing using augmented data (Test_Avg_Aug) accuracy for all models based on 3U-Net models

- **Models based on 3U-Net with long connection**

The maximum Test_Avg accuracy is 93.18% and 93.29% for Test_Avg_Aug recorded for the same model Compound_Original_Long model. While maximum training accuracy (97.69%) achieved by Compound_Compound_Long model, the maximum validation accuracy (89.68%) recorded for Original_Compound_Long model Figure 4-34. The minimum accuracy for validation (62.84%) and Test_Avg_Aug recorded for the same model Modified_Original_Long. While the minimum Test_Avg accuracy (49.34%) recorded for Original_Original_Long model, the Original_Compound_Long model recorded the minimum training accuracy 97.56%. Table 4-22 Figure 4-36

For training accuracy, the difference between the maximum 97.69% and the minimum 97.56% is 0.13%.

The validation accuracies without the minimum value of 62.84% can be divided into 3 groups. Group 1 recorded almost the same accuracy (89.68%, 89.59%, 89.47%) for models Original_Compound_Long, Modified_Compound_Long, then Compound_Modified_Long. Group 2 recorded difference =~ 1% (88.14%, 87.17%) for models Compound_Compound_Long and Modified_Modified_Long. Group 3 recorded almost the same accuracy (83.95%, 83.51%) for models Original_Original_Long and Compound_Original_Long. Group 4 contains only one model with validation accuracy 80.35% for the Original_Modified_Long model Figure 4-35. The lowest two accuracies for Test_Avg and Test_Avg_Aug recorded for the same models Modified_Original_Long then Original_Original_Long models for Test_Avg_Aug while Original_Original_Long then Modified_Original_Long models for Test_Avg. Figure 4-37

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	**97.56%**	**89.68%**	81.91%	78.19%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Original_Long	97.65%	83.51%	**93.18%**	**93.29%**
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-22 Models based on 3U-Net with long connections, Highest accuracy (Green) lowest accuracy (Red)

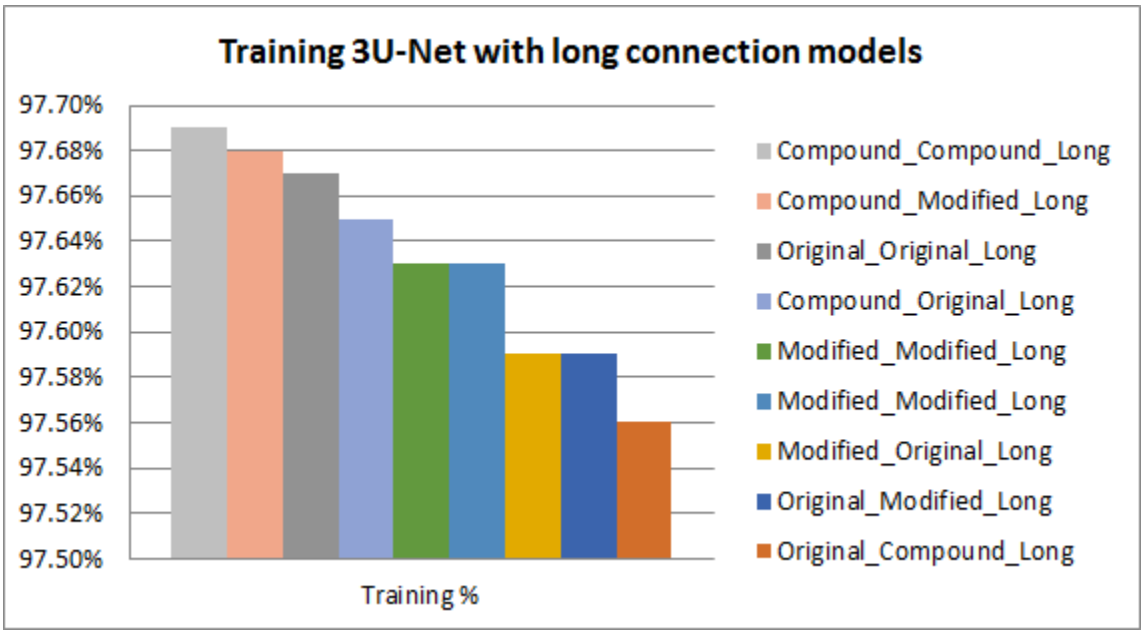


Figure 4-34 Training accuracy for all models with Long connection

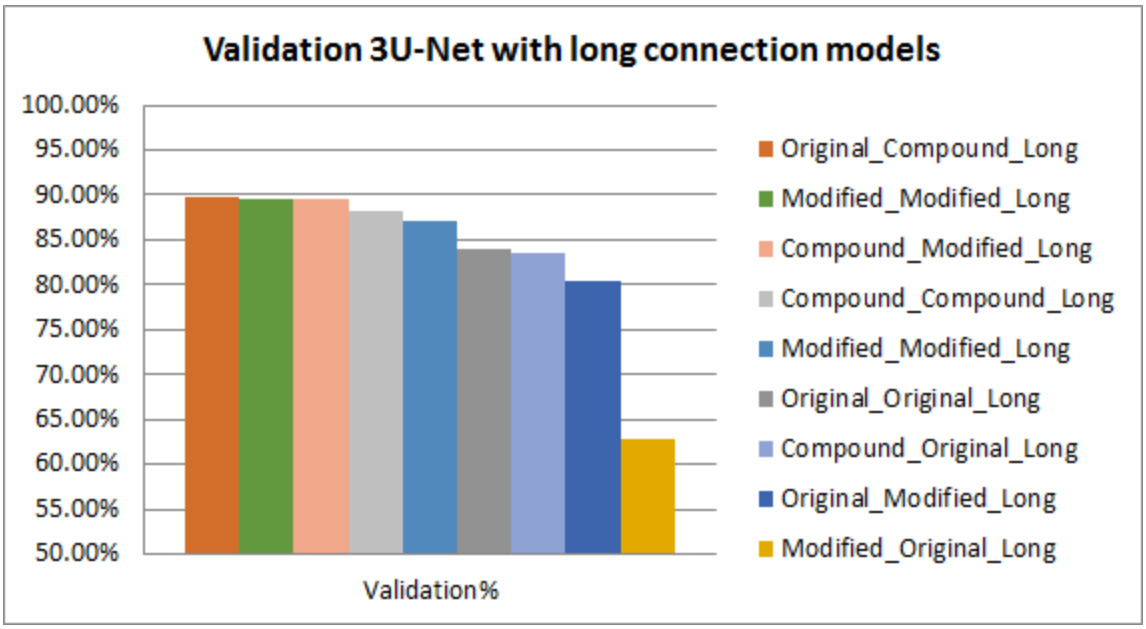


Figure 4-35 Validation accuracy for all models with Long connection

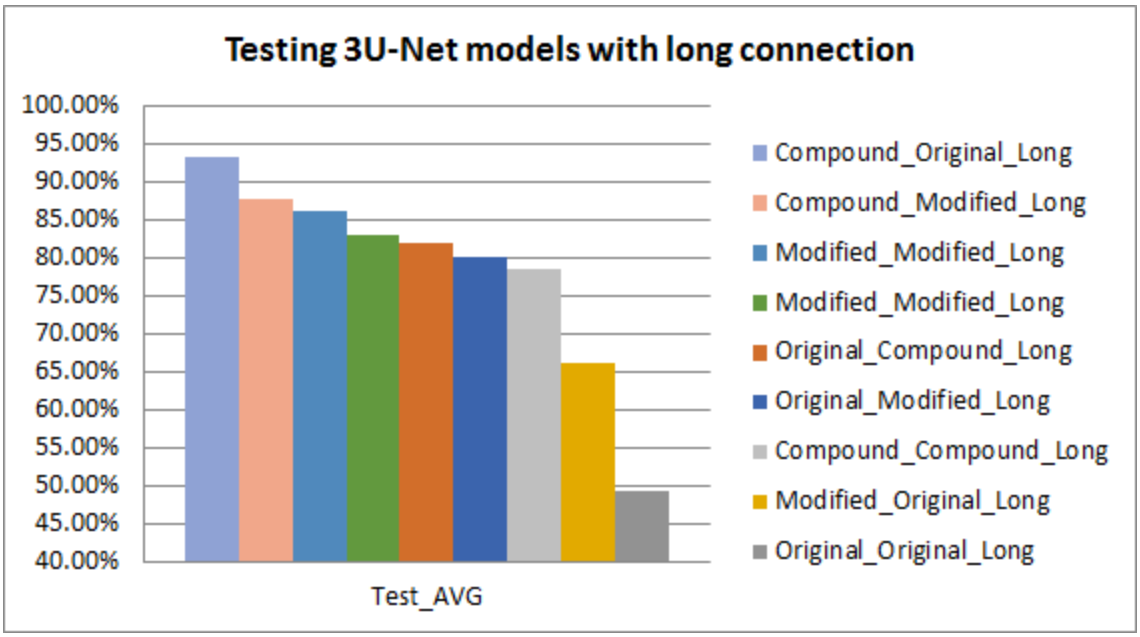


Figure 4-36 testing (Test_Avg) accuracy for all models with Long connection

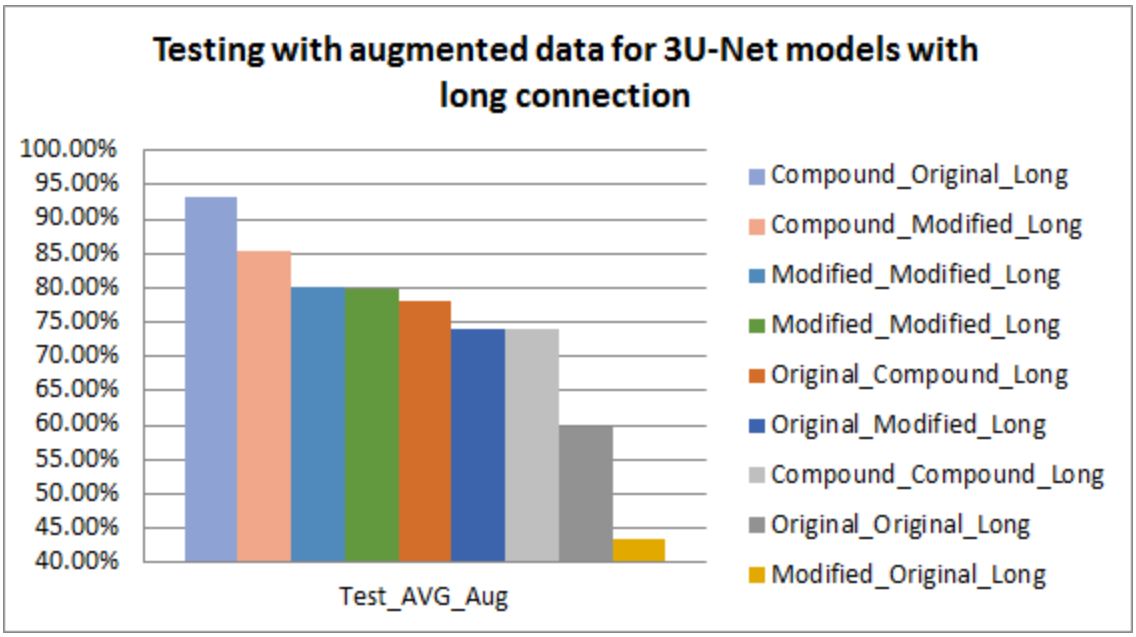


Figure 4-37 Testing using augmented data (Test_Avg_Aug) accuracy for all models with Long connection

- **Compare 3U-net models versus models with long connections**

The maximum accuracy for (validation, Test_Avg, and Test_Avg_Aug) for models with long connections (89.68%, 93.18%, 93.29%) are higher than the normal models (87.27%, 91.52%, and 92.64%) , while for training the maximum for normal models is greater than the maximum for long connection models with 0.01% (97.70% -- 97.69%) Table 4-23. The minimum values recorded for models with long connections (97.56%, 49.34%, and 43.52%) are less than the minimum for the normal models (97.58%, 60.16%, and 61.48%) for all training, Test_Avg, and Test_Avg_Aug. The minimum of validation has the opposite trend where the minimum for normal models (59.82%) is lower than the minimum for models with long connections (62.84%).

Comparing the models with/without long connections in Figure 4-40 and Figure 4-41 showed that, Test_Avg and Test_Avg_Aug have the same curve behavior. 5 of 9 models recorded accuracy for long connections models higher than the models without long

connections (Original_Compound, Modified_Compound, Modified_Modified, Compound_Original, Compound_Modified) and the opposite for the remaining 4 models (Original_Modified, Original_Original, Modified_Original, Compound_Compound) where using long connections recorded lower accuracy. Almost a similar behavior exists for validation except for two models (Compound_Compound, Compound_Original) where the accuracy reversed Figure 4-39. The training factors recorded the same behavior as testing except for 4 models having opposite behavior (Original_Modified, Original_Compound, Modified_Modified, Compound_Compound) Figure 4-38

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Original_Modified	**97.58%**	85.45%	**91.52%**	89.26%
Original_Compound	97.64%	86.84%	69.09%	**61.48%**
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	**87.27%**	78.99%	70.67%
Modified_Compound	97.59%	80.71%	78.08%	78.13%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	62.77%
Compound_Compound	97.59%	85.39%	89.76%	**92.64%**
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	**97.56%**	**89.68%**	81.91%	78.19%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Original_Long	97.65%	83.51%	**93.18%**	**93.29%**
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-23 3U-Net based models with versus without long connections, Highest accuracy (Green) lowest accuracy (Red) within each group

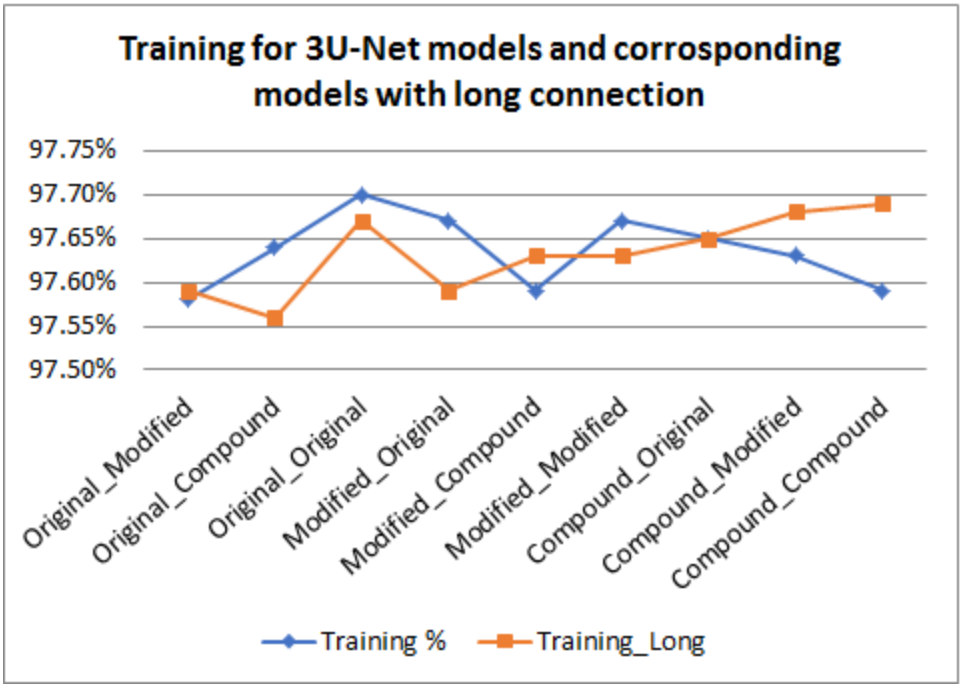


Figure 4-38 Training accuracies for 3U-Net models versus 3U-Net models with long connection

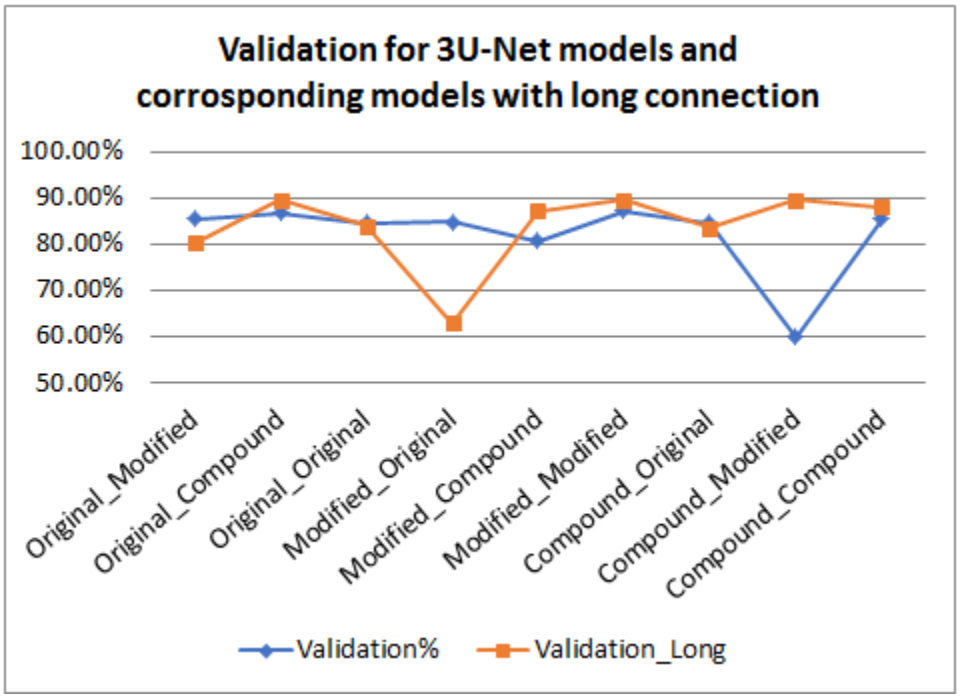


Figure 4-39 Validation accuracies for 3U-Net models versus 3U-Net models with long connection

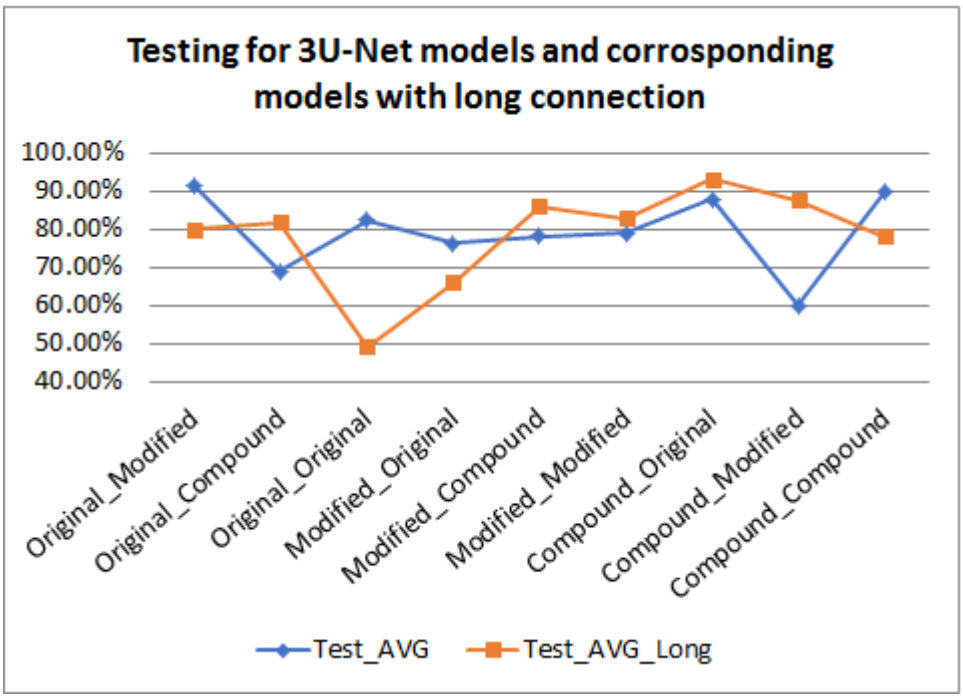


Figure 4-40 Testing (Test_Avg) accuracies for 3U-Net models versus 3U-Net models with long connection

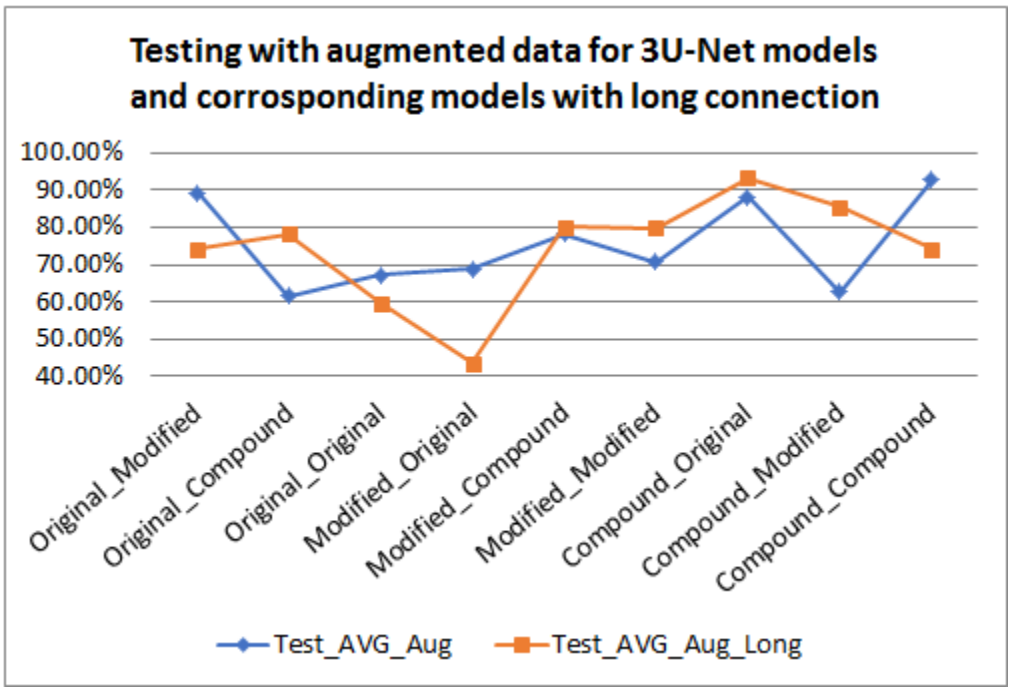


Figure 4-41 Testing using augmented data (Test_Avg_Aug) accuracies for 3U-Net models versus 3U-Net models with long connection

- **Compare each model with the respective long bridge model**

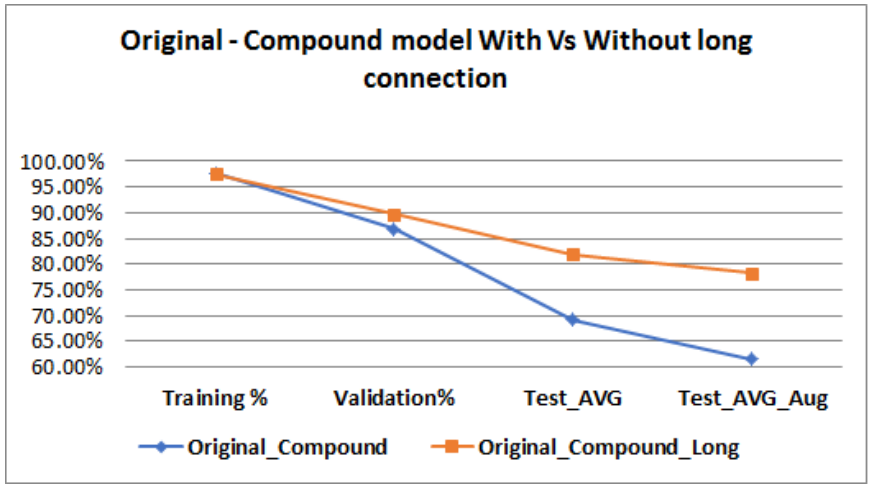
5 of 9 models showed enhancement in the accuracy for all training, validation, testing, and testing using augmented data when using long connecting rather than the normal model. These models are Original_Compound_Long, Compound_Original_Long, Modified_Compound_Long, Compound_Modified_Long , and Modified_Modified_Long. Figure 4-42.

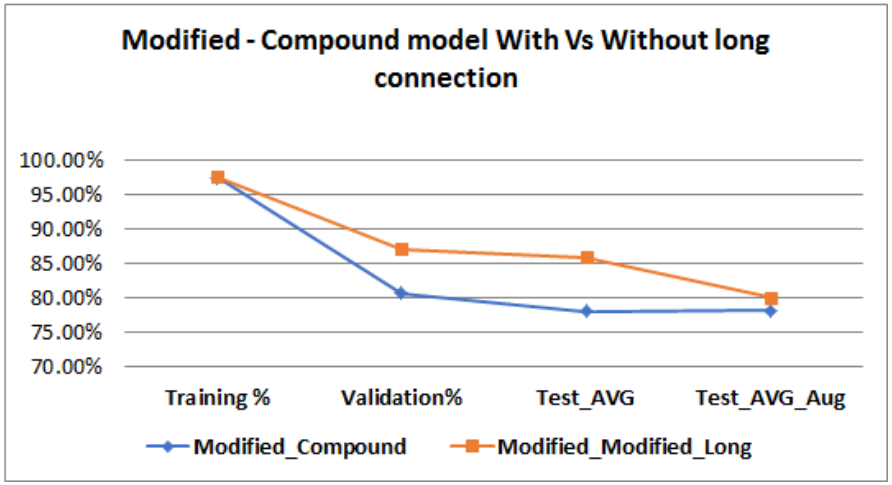
All the models that contains Compound model either at the beginning or at the end showed better accuracy when using long connections except Compound_Compound better accuracy without using long connections.

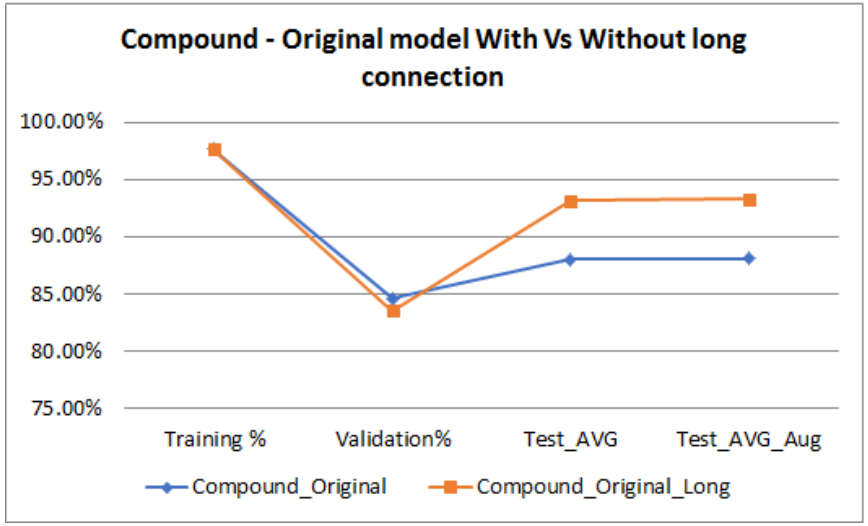
The rest 4 models showed the opposite trend, where the normal models recorded better performance than using long connections with all training, validation, testing, and testing using augmented data. These models are Original_Modified, Modified_Original, Compound_Compound, and Original_Original Figure 4-43

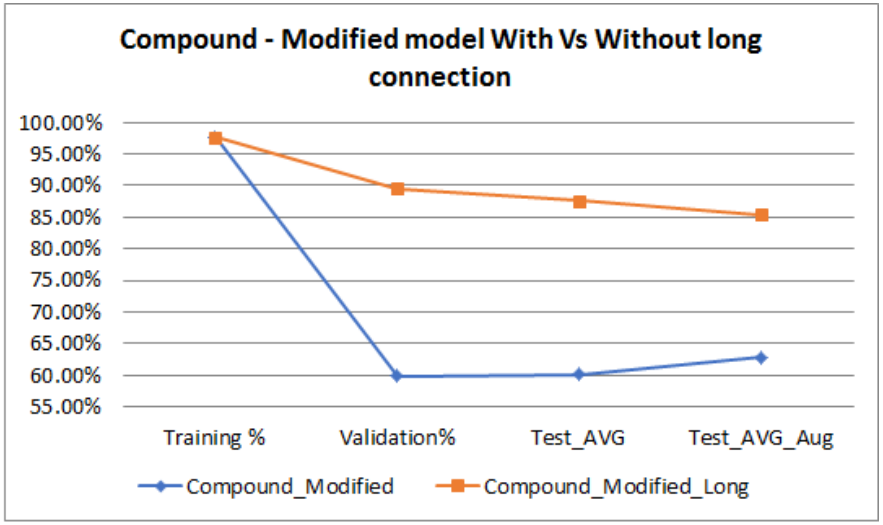
All models contain the same structure (original_original, Compound_Compound) recorded lower accuracy when using long connection except Modified_Modified which showed better accuracy with long connection.

Original_modified or Modified_Original models showed that, the accuracy decreased when using long connections.


Original - Compound model With Vs Without long connection
100.00%
95.00%
90.00%
85.00%
80.00%
75.00%
70.00%
65.00%
60.00%
Training %
Validation%
Test_AVG
Test_AVG_Aug
Original_Compound
Original_Compound_Long


Modified - Compound model With Vs Without long connection
100.00%
95.00%
90.00%
85.00%
80.00%
75.00%
70.00%
Training %
Validation%
Test_AVG
Test_AVG_Aug
Modified_Compound
Modified_Modified_Long


Compound - Original model With Vs Without long connection
100.00%
95.00%
90.00%
85.00%
80.00%
75.00%
Training %
Validation%
Test_AVG
Test_AVG_Aug
Compound_Original
Compound_Original_Long


Compound - Modified model With Vs Without long connection
100.00%
95.00%
90.00%
85.00%
80.00%
75.00%
70.00%
65.00%
60.00%
55.00%
Training %
Validation%
Test_AVG
Test_AVG_Aug
Compound_Modified
Compound_Modified_Long

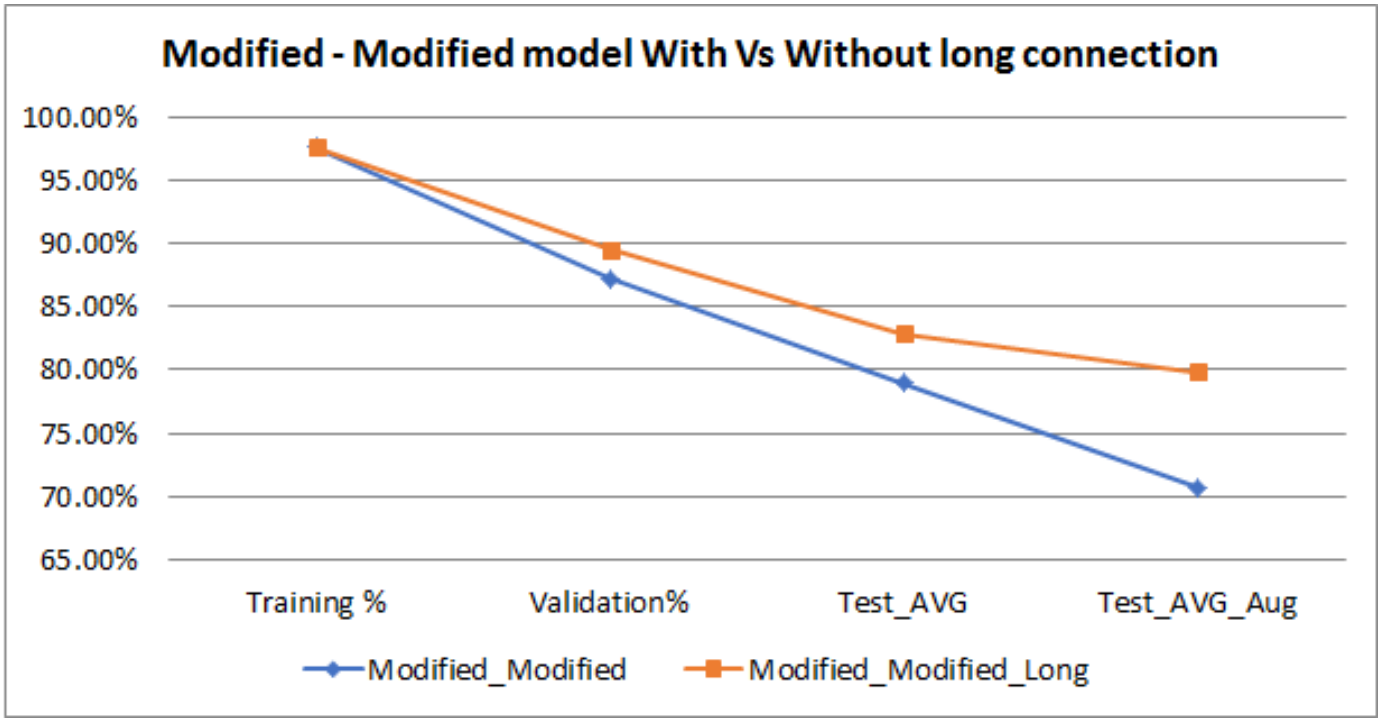
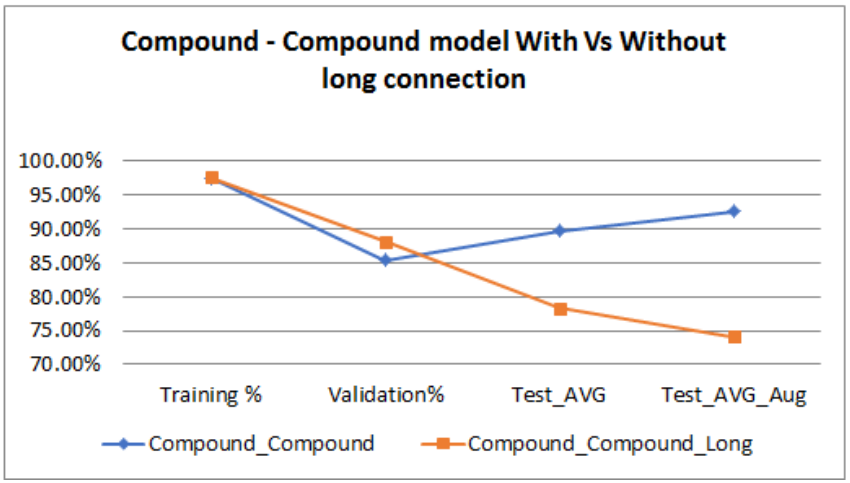
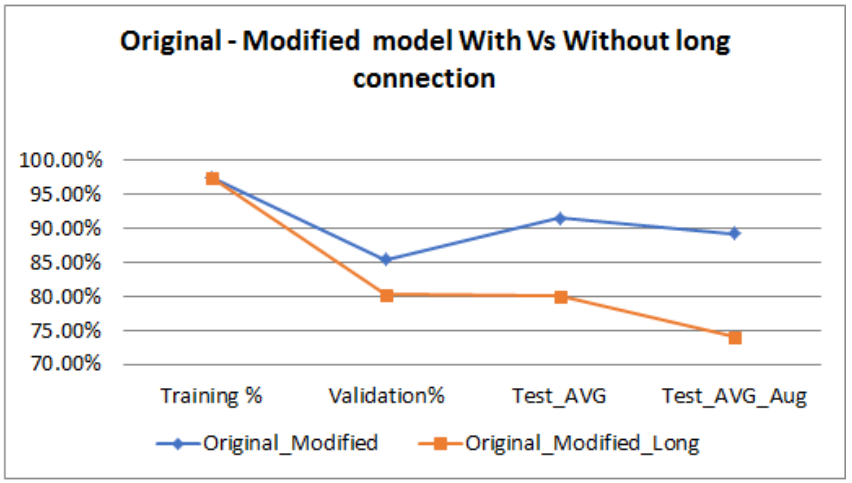
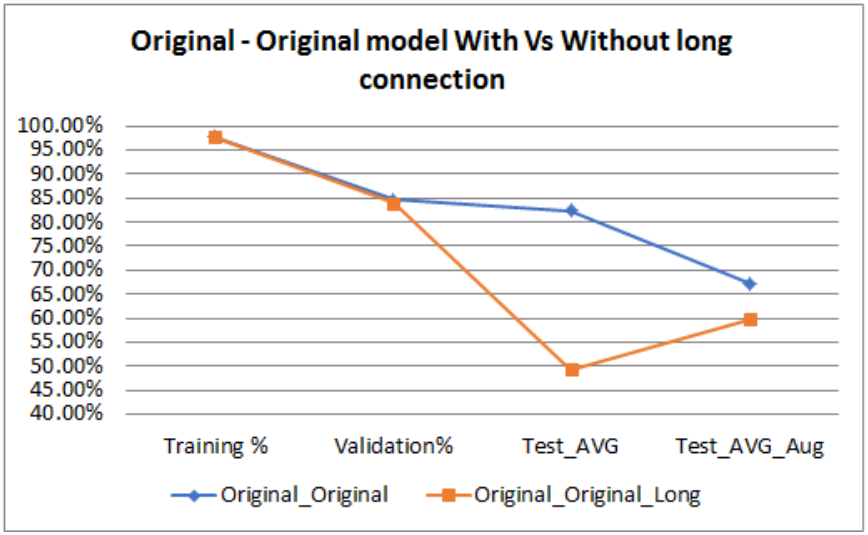


Figure 4-42 5 Models with long connection recorded better accuracy against the same models without long connections (Original-Compound, Compound-Original, Modified-Compound, Compound-Modified, Modified-Modified)

.







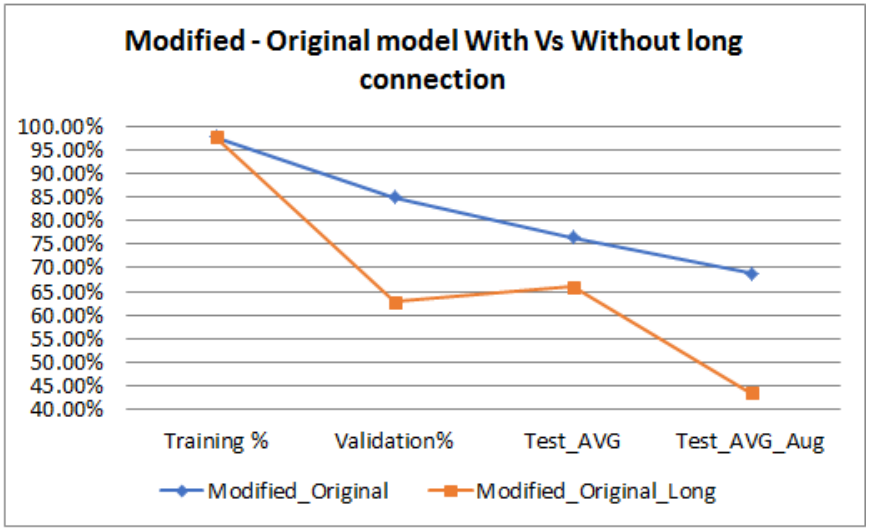


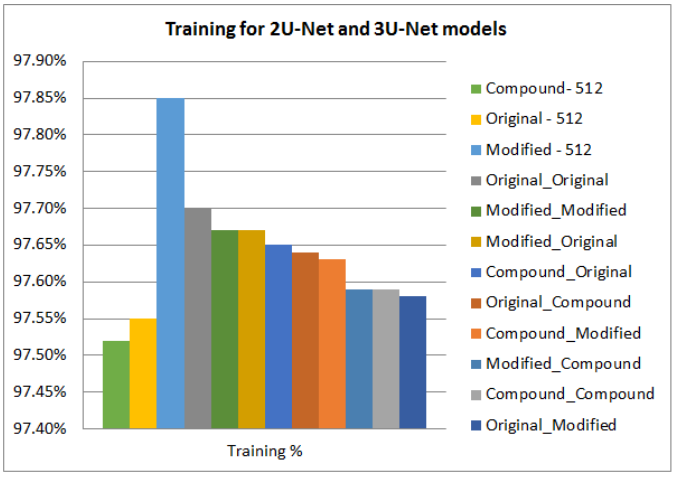
Figure 4-43 4 Models show better accuracy against the same models with long connections (Compound-Compound, Original-Original, Original-Modified, Modified-Original)

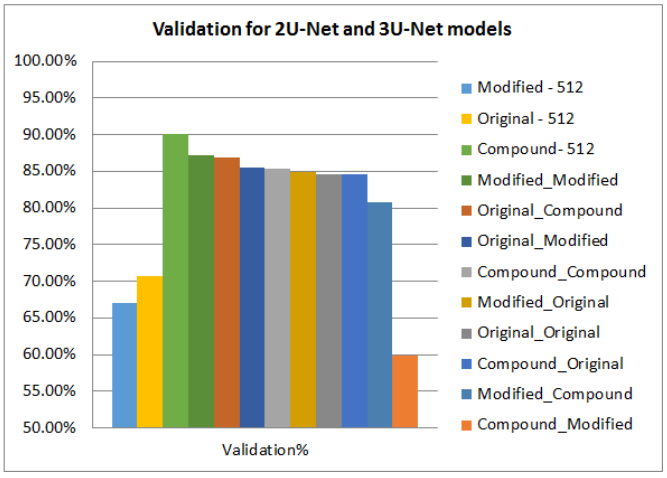
- **Compare 2U-Net models with 3U-Net models**

The maximum accuracy for 2U-Net models is greater than 3U-Net models except for Test_Avg where 2U-Net models is lower than the 3U-Net models (Compound < Original_Modified) while the minimum for 3U-Net is lower than the minimums for 2U-Net Figure 4-44 . Excluding the lowest value of validation for 3U-Net models, all the values for 3U-Net models (training, validation, and Test_Avg_Aug) are lower than the maximum and higher than the second maximum of the 2U-Net models except for the Test_Avg accuracy. Table 4-24

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Original_Modified	**97.58%**	85.45%	**91.52%**	89.26%
Original_Compound	97.64%	86.84%	69.09%	**61.48%**
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	**87.27%**	78.99%	70.67%
Modified_Compound	97.59%	80.71%	78.08%	78.13%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	62.77%
Compound_Compound	97.59%	85.39%	89.76%	**92.64%**
Original	97.55%	70.65%	**73.21%**	60.31%
Modified	**97.85%**	**67.02%**	79.12%	**60.10%**
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**

Table 4-24 2U-Net models versus 3U-Net models, Highest accuracy (Green) lowest accuracy (Red) within each group





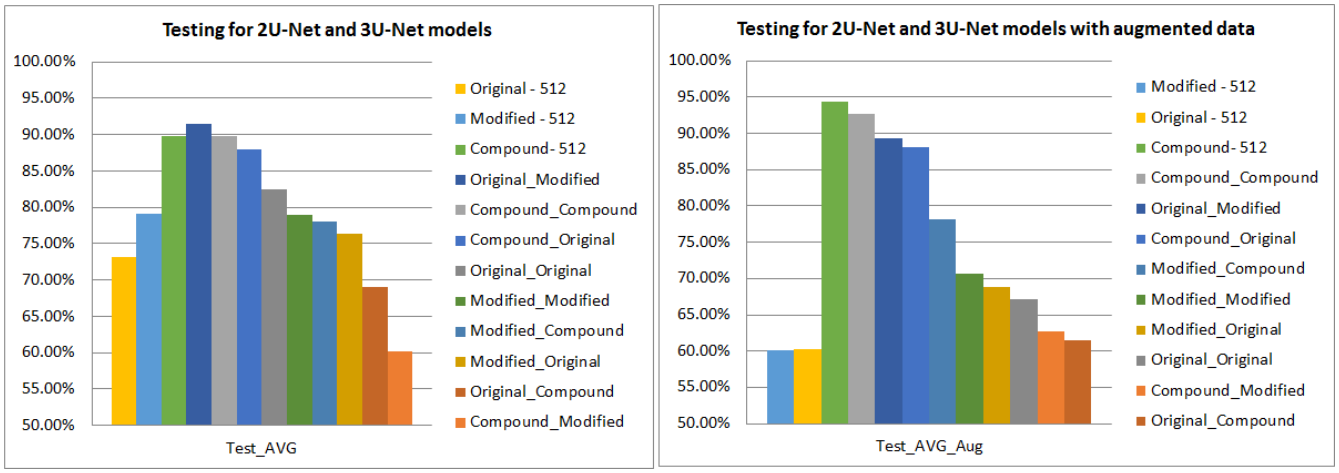


Figure 4-44 2U-Net models versus 3U-Net models

- **Compare 2U-Net models with 3U-Net with long bridge**

All 3U-Net models with long connections have the same behavior as with 3U-Net model. The maximum accuracy recorded for 2U-Net models always greater than the maximum for 3U-Net models with long connections except for Test_Avg while the minimum for all factors for models with long connections is lower than the minimums for 2U-Net models Figure 4-45. If we exclude the lowest accuracy for 3U-Net models with long connections for validation and the lowest two values in Test_Avg_Aug, then the training, validation, and Test_Avg_Aug will follow the same behavior as 3U-Net models which shows that, all the values recorded for 3U-Net models are in the range between the maximum and the second maximum accuracy for 2U-Net models for training, validation, testing, and testing with augmented data. Table 4-25 Figure 4-46

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	**97.56%**	**89.68%**	81.91%	78.19%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Original_Long	97.65%	83.51%	**93.18%**	**93.29%**
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%
Original	97.55%	70.65%	**73.21%**	60.31%
Modified	**97.85%**	**67.02%**	79.12%	**60.10%**
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**

Table 4-25 2U-Net models versus 3U-Net models with long connections, Highest accuracy (Green) lowest accuracy (Red) within each group

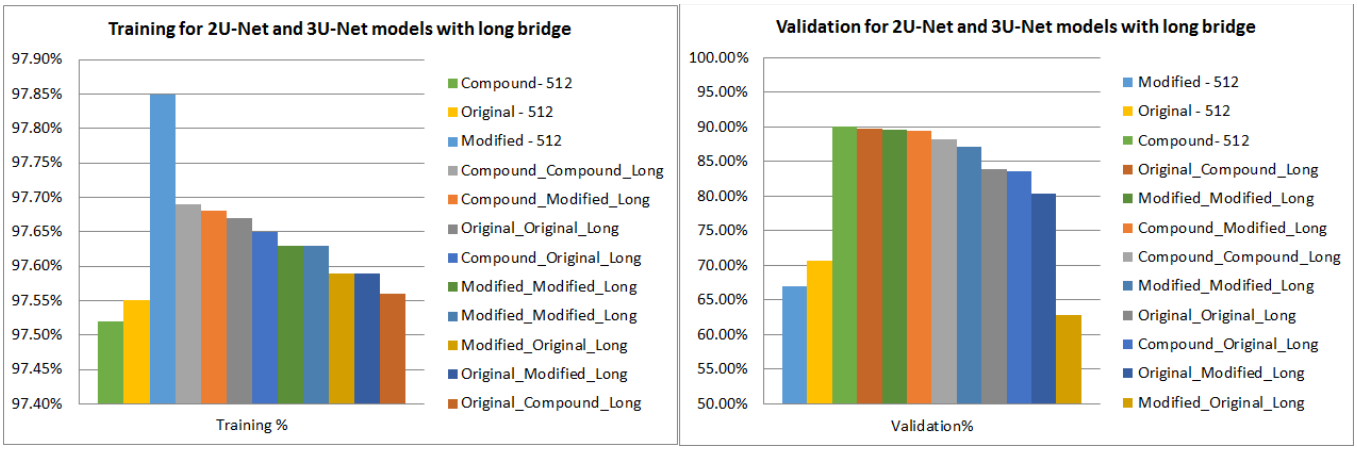


Figure 4-45 Training and Validation accuracies for 2U-Net models versus 3U-Net models with long connections

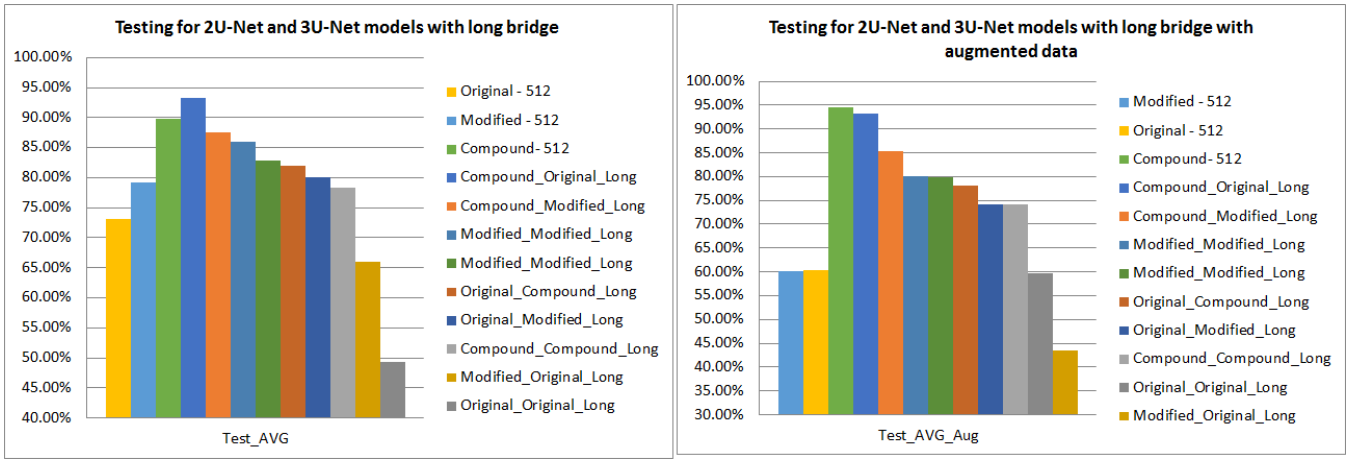


Figure 4-46 Testing and Testing using augmented data (Test_Avg, Test_Avg_Aug) accuracies for 2U-Net models versus 3U-Net models with long connections

- **Compare models with only on type of connection for the 3U-Net with the respective 2U-Net model and long connection**

In this section the effect of adding another U-Net of the same structure then adding long bridge connection would be investigated for the three model types. For example, the 2U-Net model with Compound connection will be compared with 3U-Net model Compound_Compound and Compound_Compound_Long.

❖ **Original models**

Adding another U-Net with original structure to make the model Original_Original increased the accuracy to the maximum for all measures, training, and validation, Test_Avg, and Test_Avg_Aug. While the minimum accuracy for training and validation recorded for the Original mode, the minimum Test_Avg and Test_Avg_Aug reached when adding long connection for Original_Original_Long model. Figure 4-47

❖ **Modified models**

The accuracy of Test_Avg and Test_Avg_Aug started with minimum values when using 2U-Net modified model then increased by adding a third U-Net and increased to reach the maximum by adding long connection to the modified model. The opposite behavior

happened for training, where started with the maximum with 2 U-Net model (Modified) then decreased when adding another modified connection (Modified_Modified) to reach the minimum when adding long connection (Modified_Modified_Long). The validation accuracy started with minimum accuracy when using 2U-Net then increased to the maximum after adding the third U-Net then decreased again after adding the long connections. Figure 4-48

- **Compound models**

The Compound model for 2U-Net recoded the maximum accuracy for validation, Test_Avg, and Test_Avg_Aug while the training accuracy was minimum value. Adding a third U-Net then long connection decreased the accuracy to be the minimum for validation with Compound_Compound model and the minimum for Test_Avg and Test_Avg_Aug with Compound_Compound_Long. The opposite behavior recorded for training accuracy where started with minimum value with Compound model then increased to reach the maximum accuracy with Compound_Compound_Long model. Figure 4-49

The overall results prove that the best model using original structure is (Original_Original) and for Modified structure is (Modified_Modified_Long) while for Compound structure (Compound for 2U-Net). The two models with Compound connections (Compound, Compound_Compound) recorded the 2 maximum accuracies over all the other models with similar connections with/without long connections for all factors except training. Table 4-26

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	**97.55%**	**70.65%**	73.21%	60.31%
Original_Original	**97.70%**	**84.62%**	**82.42%**	**67.22%**
Original_Original_Long	97.67%	83.95%	**49.34%**	**59.73%**
Modified	**97.85%**	**67.02%**	**79.12%**	**60.10%**
Modified_Modified	97.67%	**87.27%**	78.99%	70.67%
Modified_Modified_Long	**97.63%**	87.17%	**85.95%**	**80.10%**
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**
Compound_Compound	97.59%	**85.39%**	89.76%	92.64%
Compound_Compound_Long	**97.69%**	88.14%	**78.30%**	**74.07%**

Table 4-26 compare U-Net with the respective 3U-Net models and with long connection models, Highest accuracy (Green) lowest accuracy (Red) within each group

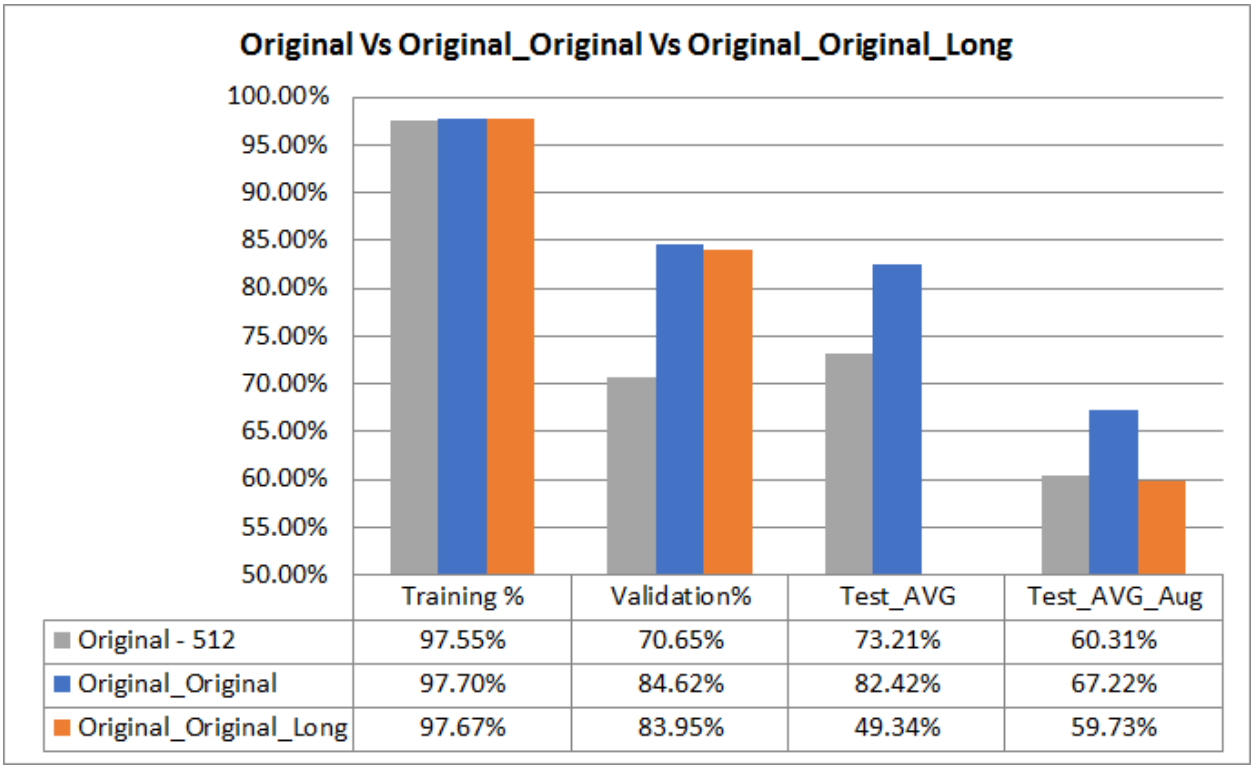
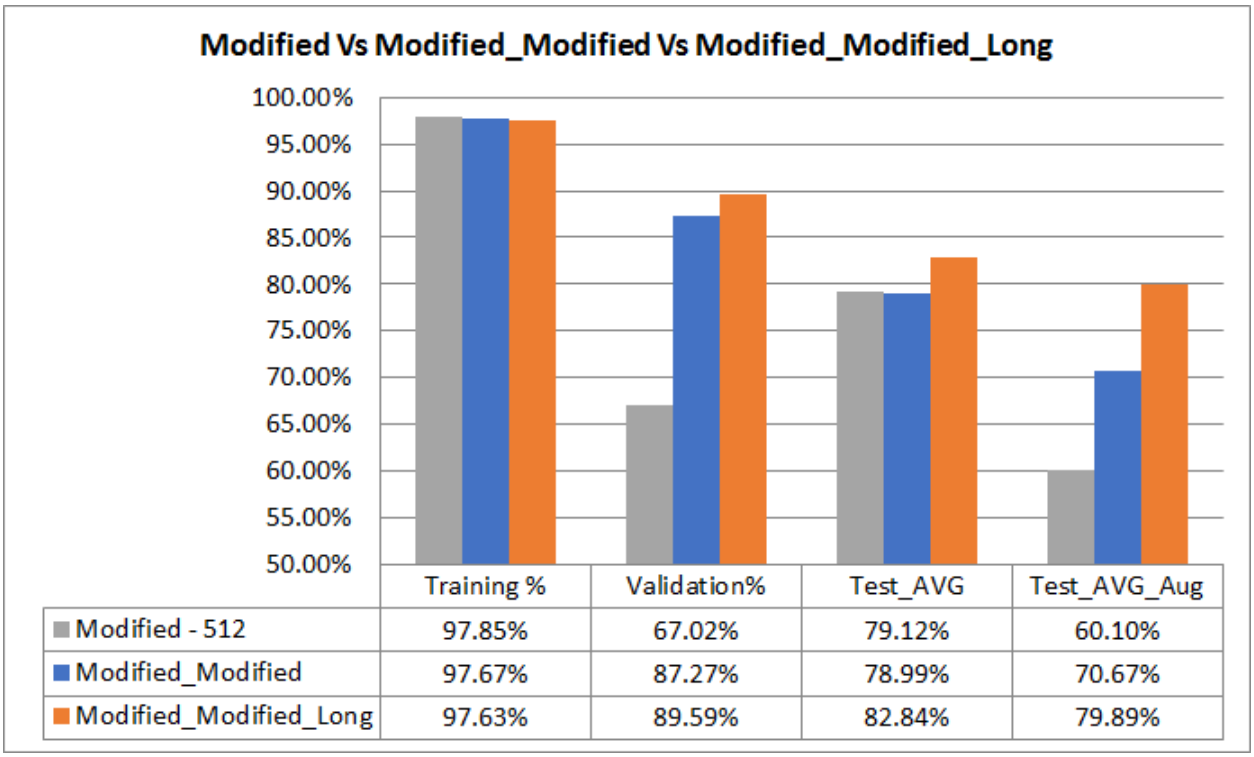
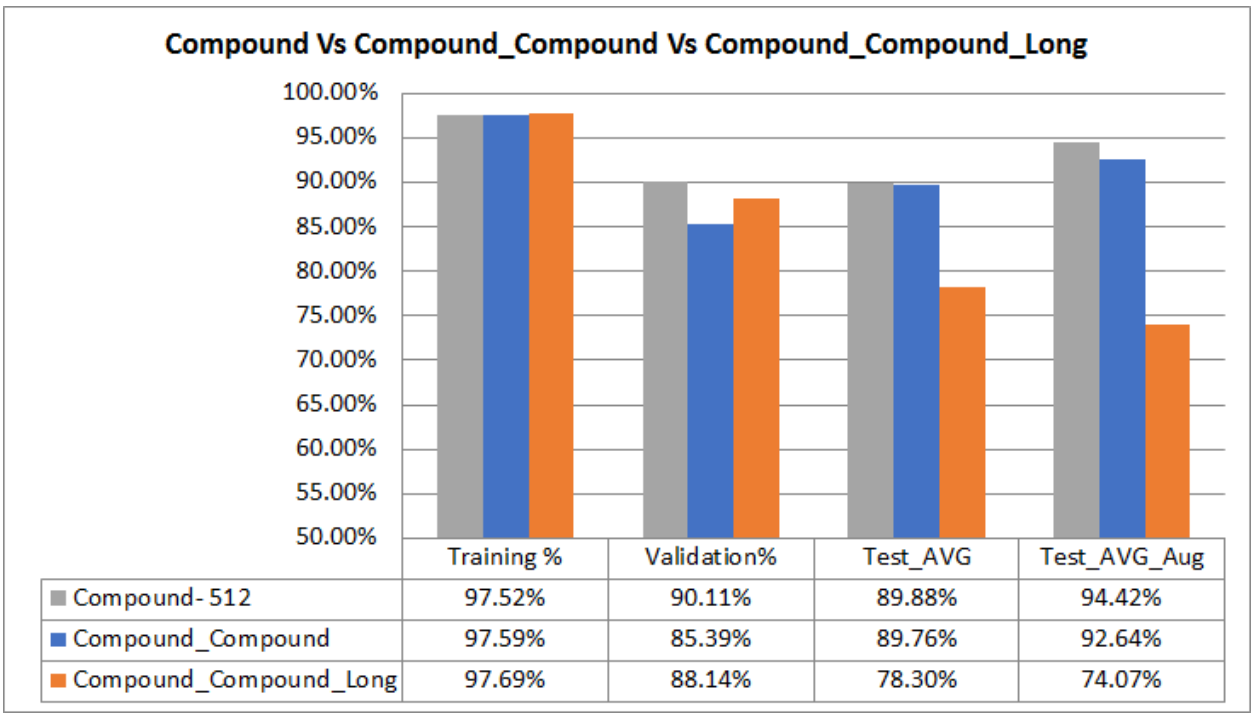


Figure 4-47 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Original connections for 2U-Net and 3U-Net and 3U-Net with long connection



	Training %	Validation%	Test_AVG	Test_AVG_Aug
Modified - 512	97.85%	67.02%	79.12%	60.10%
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Modified_Modified_Long	97.63%	89.59%	82.84%	79.89%

Figure 4-48 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Modified connections for 2U-Net and 3U-Net and 3U-Net with long connection



	Training %	Validation%	Test_AVG	Test_AVG_Aug
Compound- 512	97.52%	90.11%	89.88%	94.42%
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Compound_Compound_Long	97.69%	88.14%	78.30%	74.07%

Figure 4-49 Training, Validation, Test_Avg, and Test_Avg_Aug for all modles with only Compound connections for 2U-Net and 3U-Net and 3U-Net with long connection

- **Compare2U-Net models versus 3U-Net models that have the same connections with/without long connection**

Table 4-27 illustrates the results of comparison between 2U-Net models and 3U-Net models versus 3U-Net with long connection models where the same bridge connection structure used between each two U-Nets.

❖ **2U-Net models**

Compound model recorded the maximum accuracy for validation, Test_Avg, and Test_Avg_Aug while recorded the minimum for training. The minimum accuracy for validation and Test_Avg_Aug recorded for modified model while minimum accuracy for Test_Avg recorded for Original model.

❖ **3U-Net models**

Compound_Compound model achieved the maximum accuracy for Test_Avg and Test_Avg_Aug, while maximum validation and maximum training recorded for Modified_Modified model and Original_Original model respectively. The Original_Original model recorded the minimum accuracy for validation and Test_Avg_Aug while minimum Test_Avg and minimum training recorded for Modified_Modified and Compound_Compound respectively. Table 4-277 Figure 4-50

❖ **3U-Net models with long connections**

The minimum accuracy for Test_Avg and Test_Avg_Aug recorded for Original_Original_Long model while the maximum values reached with Modified_Modified_long model. Training and validation accuracies recorded the maximum

for Compound_Compound_Long model while was minimum accuracy recorded by Modified_Modified_Long model and Original_Original_Long Model respectively. Figure 4-51

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	97.55%	70.65%	**73.21%**	60.31%
Modified	**97.85%**	**67.02%**	79.12%	**60.10%**
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**
Original_Original	**97.70%**	**84.62%**	82.42%	**67.22%**
Modified_Modified	97.67%	**87.27%**	**78.99%**	70.67%
Compound_Compound	**97.59%**	85.39%	**89.76%**	**92.64%**
Original_Original_Long	97.67%	**83.95%**	**49.34%**	**59.73%**
Modified_Modified_Long	**97.63%**	87.17%	**85.95%**	**80.10%**
Compound_Compound_Long	**97.69%**	**88.14%**	78.30%	74.07%

Table 4-27 Accuracy for 2U-Net models, 3U-Net models, and 3U-Net models with long connections with same connection between all 3 U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group

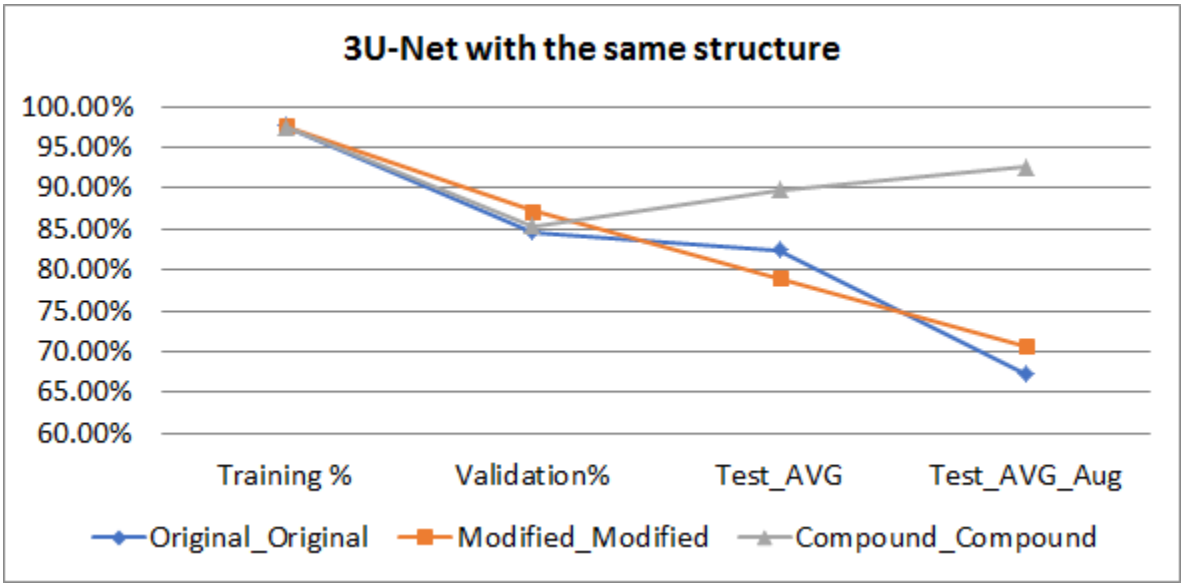


Figure 4-50 Training, Validation, Test_Avg, Test_Avg_Aug accuracies for 3U-Net models that used only one connection type between each two U-Nets

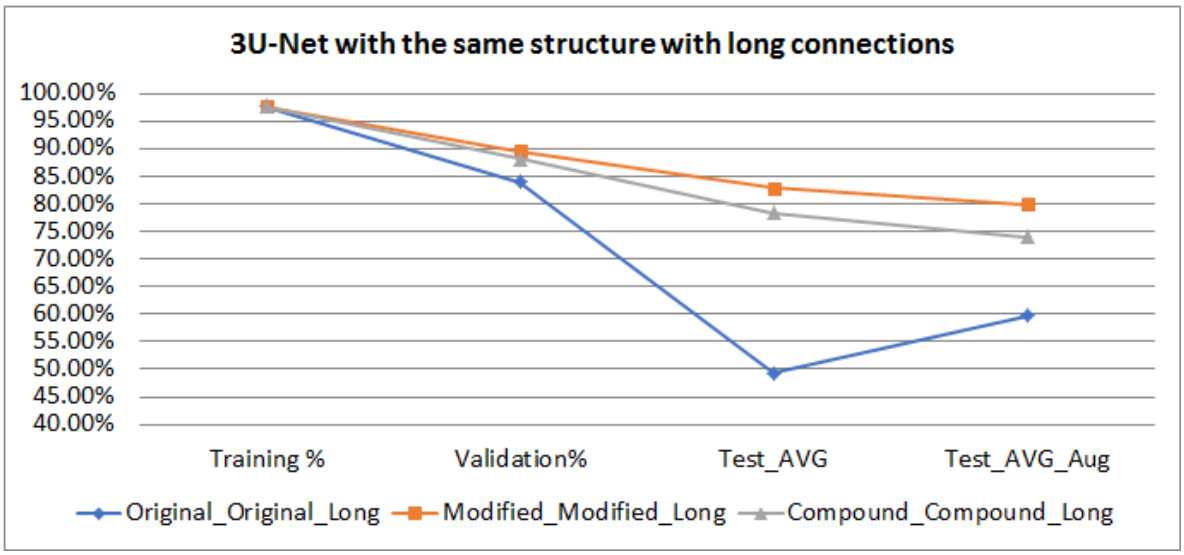


Figure 4-51 Training, Validation, Test_Avg, Test_Avg_Aug accuracies for 3U-Net models with long connection that used only one connection type between each two U-Nets

- **Compare all Models have the same start connections between the first and second U-Net**

In this section, the models based on 2U-Nets structure with filters 32-512 would be compared with all other models based on 3U-Net models where the first and second U-Nets having the same connections as the 2U-Net based model with/without long connections. For example, the 2U-Net model with Modified connections would be compared with all 3U-Net models started with Modified connections between first and second U-Net (Modified, Modified_Original, Modified_Modified, Modified_Compound, Modified_Original_Long, Modified_Modified_Long, Modified_Compound_Long).

- **Models start with Original connections**

All the models included in this section consist of 3U-Nets where the first and second U-Nets are connected based on the **original** structure while the second U-Net and third U-Net connected with one of three structures (Original, Modified, Compound). The other

models have the same structure as the previous ones in addition to the Long connections. Table 4-28

The Original-Modified model recorded the maximum accuracy for Test_Avg and Test_Avg_Aug factors while the maximum for training and validation recorded for Original_Original and Original_compound_Long models respectively. The minimum accuracy for Test_Avg and Test_Avg_Aug recorded for the Original_Original_Long model while the Original model recorded the minimum accuracy for Training and validation.

For training factor, all models used Long connections recorded lower accuracy than the models without long connections except Original_Modified_Long that having higher training accuracy than Original_Modified model. Figure 4-52 Validation, Test_Avg and Test_Avg_Aug had the same curve behavior, the models with long connections recorded lower accuracies than the same model structure without long connections except for Original_Compound_Long model which achieved accuracy higher than Original_Compound model. Figure 4-53 The two U-Net based model with Original connections achieved the minimum accuracy over all of three U-Nets with/without long connections for Training and validation, while recorded the second minimum accuracy model for Test_Avg_Aug after Original_Original_Long model and the third minimum for Test_Avg after Original_Original_Long and Original_Compound models. Table 4-28

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	**97.55%**	**70.65%**	73.21%	60.31%
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Original_Modified	97.58%	85.45%	**91.52%**	**89.26%**
Original_Compound	97.64%	86.84%	69.09%	61.48%
Original_Original_Long	97.67%	83.95%	**49.34%**	**59.73%**
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	97.56%	**89.68%**	81.91%	78.19%

Table 4-28 All models of three U-Nets with Original connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)

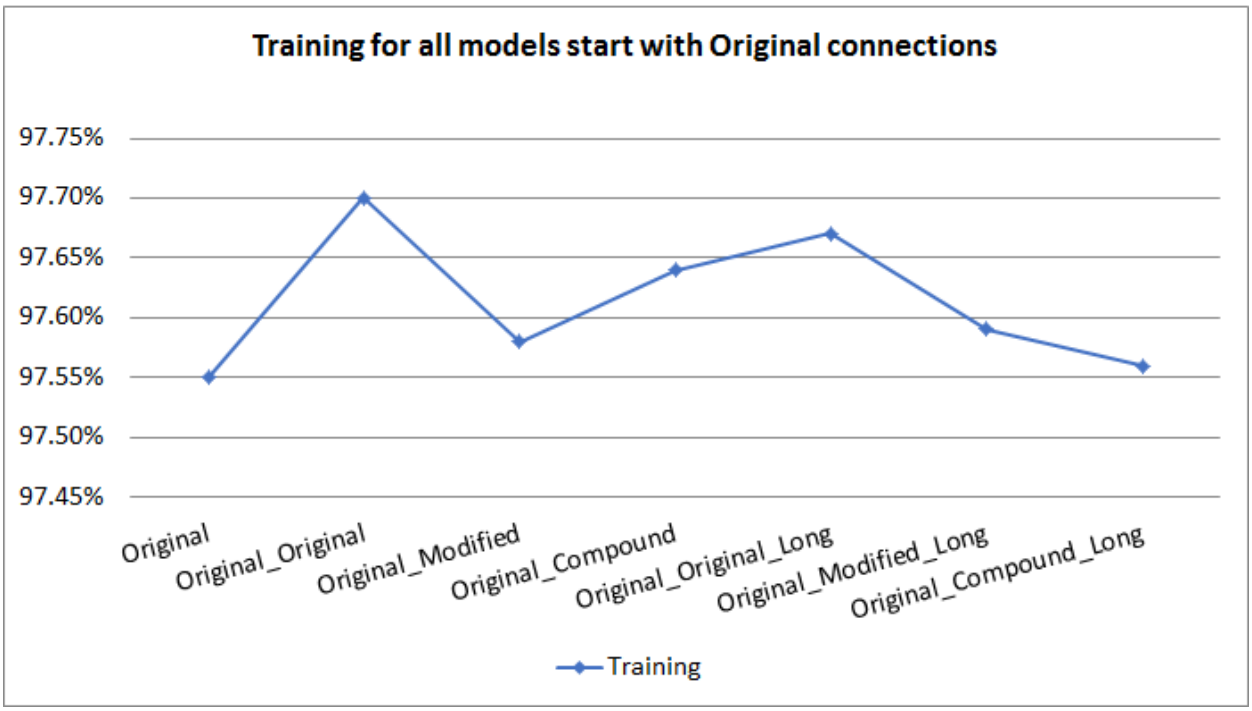


Figure 4-52 Training accuracy for all models of three U-Nets with Original connections between the first two U-Nets and variant connections between the second and third U-Nets

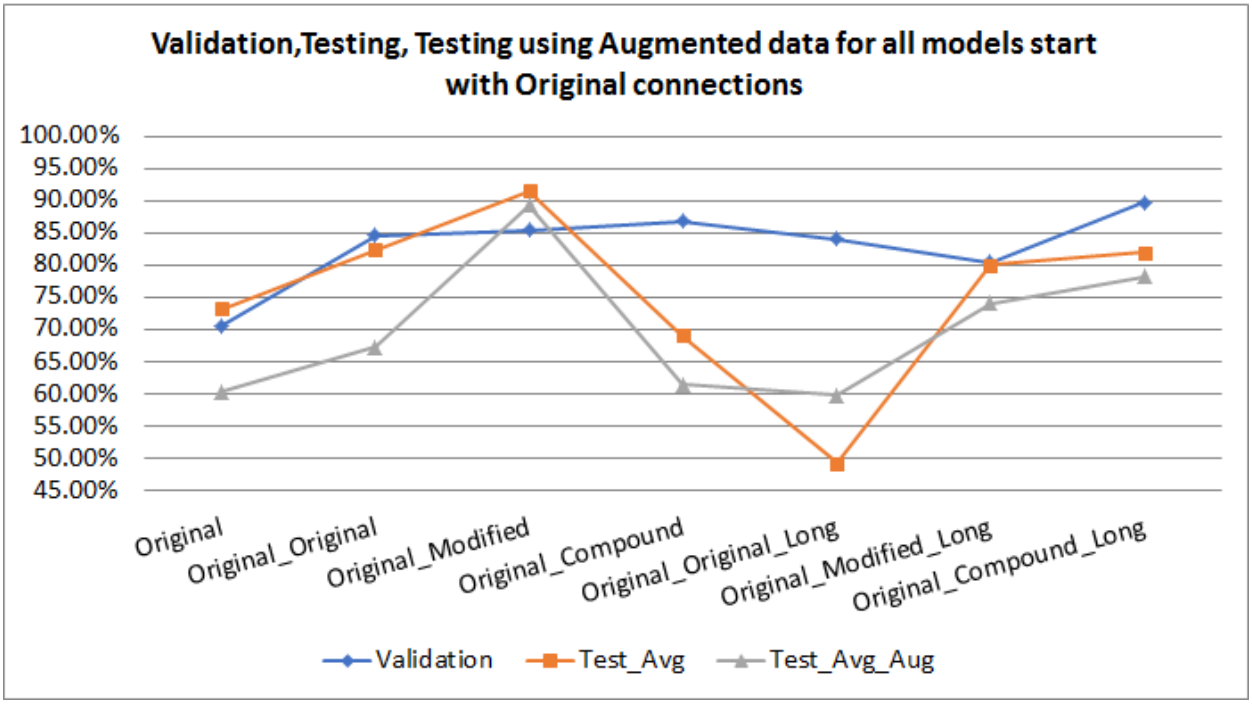


Figure 4-53 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Original connections between the first two U-Nets and variant connections between the second and third U-Nets

- **Models start with Modified connections**

All the models included in this section consist of 3U-Nets where the first and second U-Nets are connected based on the Modified structure while the second U-Net and third U-Net connected with one of three structures (Original, Modified, Compound). The other models have the same structure as the previous ones in addition to the Long connections.

The Modified model recorded the highest accuracy for Training while the maximum for Test_Avg and Test_Avg_Aug achieved by Modified_Modified_Long and the maximum for validation recorded for Modified_Compound_Long model. The Modified_Original_Long model recorded the minimum accuracy for all four factors Training, Validation, Test_Avg, and Test_Avg_Aug and shared the same minimum accuracy for training with Modified_Compound model. Table 4-29

For training factor, all models used Long connections recorded lower accuracy than the models without long connections except Modified_Compound_Long that having higher training accuracy than Modified_Compound model. Figure 4-54

The validation accuracy using models with long connections recorded values lower than the models without long connections except for model Modified_Compound_Long that recorded accuracy higher than Modified_Compound model. Figure 4-55

Test_Avg and Test_Avg_Aug have the same curve behavior, the models with long connections recorded higher accuracies than the same model structure without long connections except for Modified_Original_Long model that recorded accuracy lower than Modified_Original model. Figure 4-55

The two U-Net based model with Modified connections achieved the best accuracy over all of three U-Nets with/without long connections for Training while recorded the second minimum accuracy after Modified_Original_Long model for Validation and

Test_Avg_Aug for all 3U-Net based models with/without long connections. Modified model using 2U-Net recorded Test_Avg accuracy higher than all 3U-Net based models structure without Long connections but lower than models with long connections except Modified_Original_Long model which is the overall minimum. Table 4-29

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Modified	**97.85%**	67.02%	79.12%	60.10%
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Modified_Compound	**97.59%**	80.71%	78.08%	78.13%
Modified_Original_Long	**97.59%**	**62.84%**	**66.02%**	**43.52%**
Modified_Modified_Long	97.63%	87.17%	**85.95%**	**80.10%**
Modified_Compound_Long	97.63%	**89.59%**	82.84%	79.89%

Table 4-29 All models of three U-Nets with Modified connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)

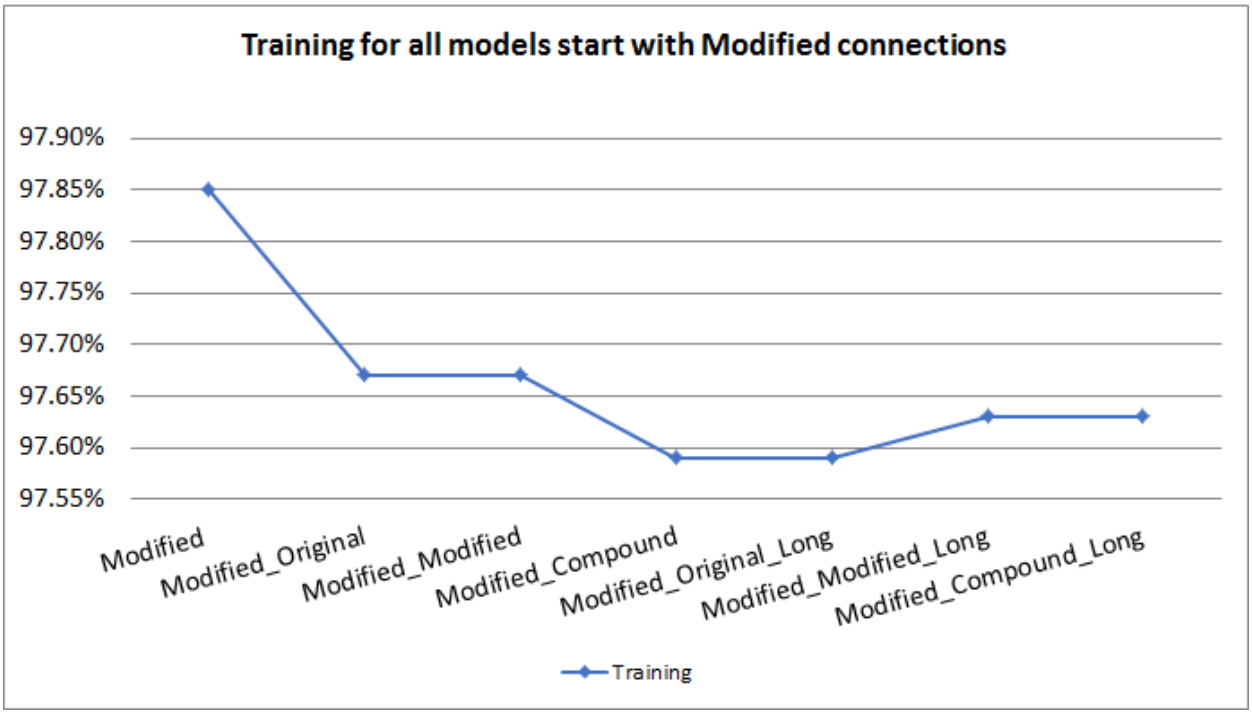


Figure 4-54 Training accuracies for all models of three U-Nets with Modified connections between the first two U-Nets and variant connections between the second and third U-Nets

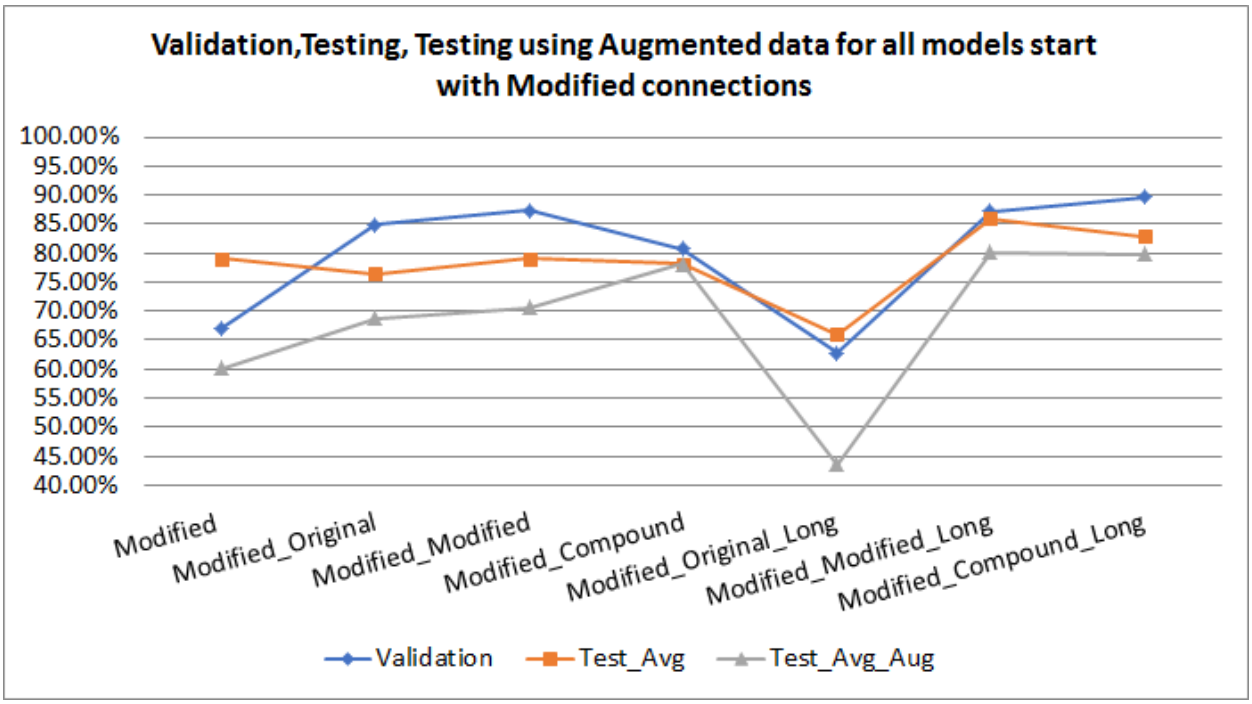


Figure 4-55 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Modified connections between the first two U-Nets and variant connections between the second and third U-Nets

- **Models start with Compound connections**

All the models included in this section consist of 3U-Nets where the first and second U-Nets are connected based on the Compound structure while the second U-Net and third U-Net connected with one of three structures (Original, Modified, Compound). The other models have the same structure as the previous ones in addition to the Long connections.

Compound model recorded the highest accuracy for Validation and Test_Avg_Aug while the maximum for Training and Test_Avg achieved by Compound_Compound_Long and Compound_Original_Long models respectively. The Compound_Modified model recorded the minimum accuracy for Validation, Test_Avg, and Test_Avg_Aug while the minimum accuracy for Training recorded by Compound model. Table 4-30

For training factor, all models used Long connections recorded higher accuracy than the models without long connections except Compound_Original and

Compound_Original_Long that having the same training accuracy. Validation factor almost have the same behavior as training where adding Long connections achieved higher accuracy than the models without long connections except for Compound_Original which recorded accuracy higher than Compound_Original_Long. Figure 4-56

Test_Avg and Test_Avg_Aug having the same curve behavior. The models with long connections recorded higher accuracies than the same model structure without long connections except for Compound_Compound_Long model which achived accuracy lower than Compound_compound model. Figure 4-57

The two U-Net based model with compound connections achieved the best accuracy over all other models based on three U-Nets with/without long connections for Validation and Test_Avg_Aug while recorded the lowest accuracy over all models based on 3U-Nets with/without long connections for Training. For Test_Avg, the 2U-Net based model recorded the second highest accuracy after Compound_Original_Long over all 3U-Nets based model with/without long connections. Table 4-30

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Compound	**97.52%**	**90.11%**	89.88%	**94.42%**
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	**62.77%**
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Compound_Original_Long	97.65%	83.51%	**93.18%**	93.29%
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-30 All models of three U-Nets with Compound connections between the first two U-Nets and varient connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red)

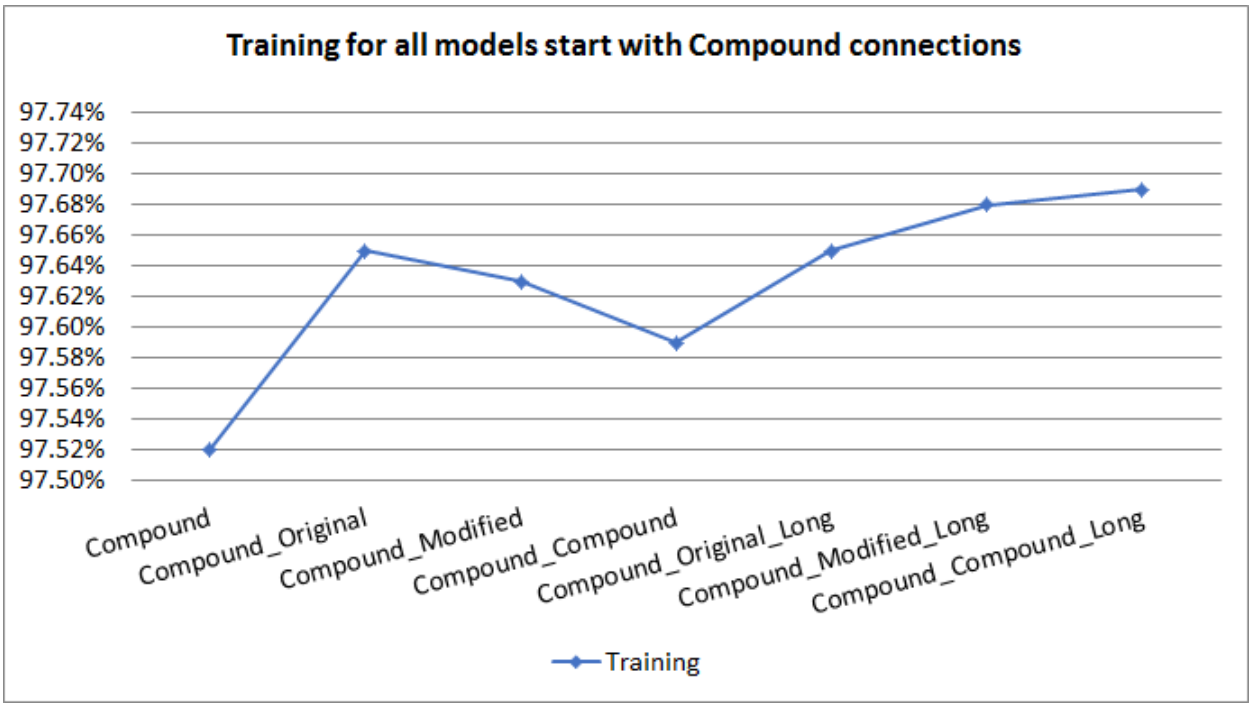


Figure 4-56 Training accuracies for all models of three U-Nets with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets

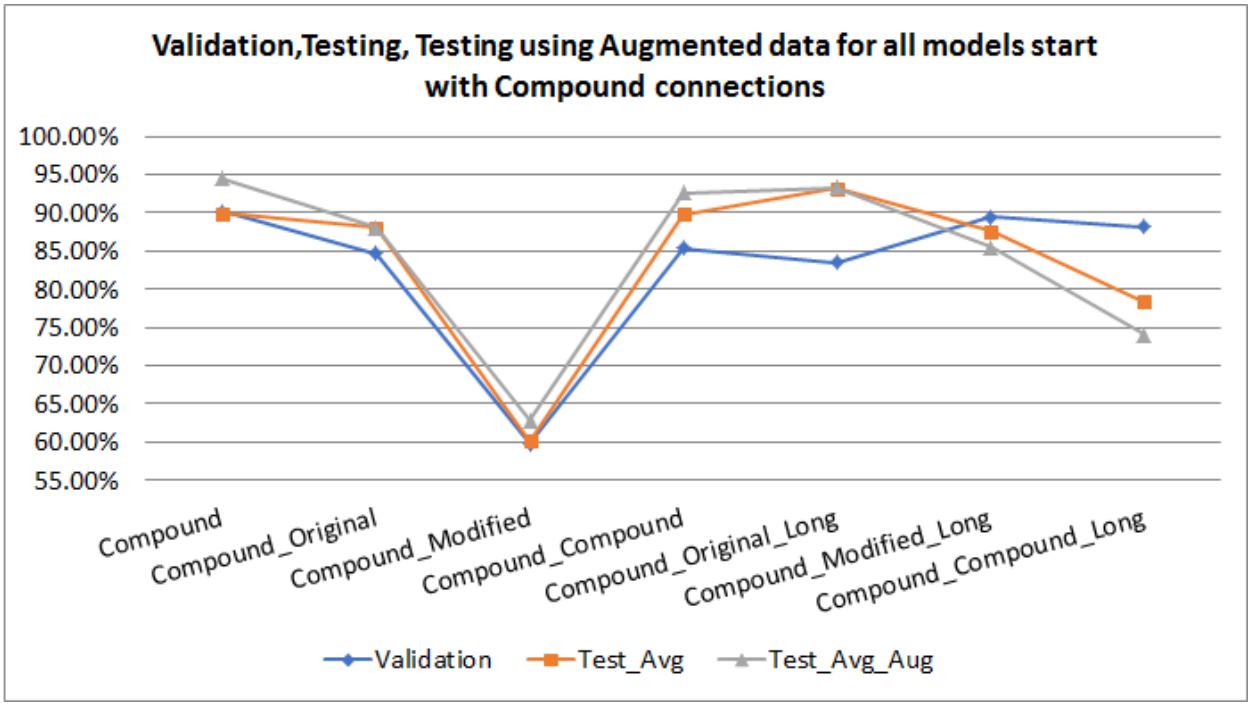


Figure 4-57 Validation, Testing (Test_Avg), and testing using augmented data (Test_Avg_Aug) accuracies for all models of three U-Nets with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets

- **All models start with the same connections (Original, Modified, Compound)**

In this section, the curve of the accuracies resulted from the models that start with the same connections between the first and second U-Net (for Ex, start with Compound connections) is compared with curve of results of the models stated with the other two connections (Original, Modified). For example, each point on any curve will be compared with the point of the other curves for the same models with the other starts. For example, the model start with Original will be compared with the respective models started with Modified and Compound connection (Original_Compound, Modified_Compound, Compound_Compound).Table 4-31

The **Training** factor accuracy shows that, For The models started with Modified connections recorded higher training accuracy than the models started with Compound for all 2U-Net and 3U-Net models but reversed when adding Long connection where the Long connection models with modified connections recorded accuracy lower than the models started with compound connections. The models started with modified connections recorded accuracy higher than the similar models started with Original connections except for three models Modified_Original, Modified_Compound, Modified_Original_Long that recorded accuracies lower than models (Original_Original, Original_compound, Oiginal_Original_Long) in sequence. The models start with Original connections recorded accuracy higher than the similar models start with Compound connections except for 3 models Original_Modified, Original_Modified_Long, Original_Compound_Long this recorded accuracy lower than Compound_Modified, Compound_Modified_Long, Compound_Compound_Long in sequence. Figure 4-58

For **Validation** factor, For The models started with Compound connections recorded higher Validation accuracy than the models started with Modified connections for models except for models Compound_Original, Compound_Modified, Compond_Compound_Long where recorded accuracy lower than the models Modified_Original, Modified_Modified, Modified_Compound_Long. The models start with Compound connections recorded accuracy lower than the similar models start with Original connections except for 2 models Compound, Compound_Modified_Long this recorded accuracy higher than Original and Original_Modified_Long in sequence. The models started with Original connections recorded accuracy higher than the similar models started with Modified connections except for three models Original_Original, Original_Modified, Oiginal_Modified_Long) that recorded accuracies lower than models (Modified_Original, Modified_Modified, Modified_Modified_Long) in sequence. Figure 4-59

For **Testing** factor (**Test_Avg**), For The models started with Compound connections recorded higher Test_Avg accuracy than both the models started with Modified and original connections except for models Compound_Modified and Compond_Compound_Long where recorded accuracy lower than both models started with Modified connections (Modified_Modified and Modified_Compound_Long) and models started with Original connections Original_Modified and Original_Compound_Long. The models started with Modified connections recorded accuracy higher than the similar models started with Original connections except for two models Modified_Original and Modified_Modified that recorded accuracies lower than models Original_Original and Original_Modified in sequence. Figure 4-60

The **Test_Avg_Aug (Testing using augmented data) shows that,** For The models started with Compound connections recorded higher Test_Avg_Aug accuracy than both the models started with Modified and original connections except for models Compound_Modified and Compond_Compound_Long where recorded accuracy lower than

both models started with Modified connections (Modified_Modified and Modified_Compound_Long) and models started with Original connections Original_Modified and Original_Compound_Long. The models started with Modified connections recorded accuracy higher than the similar models started with Original connections except for two models Modified_Modified and Modified_Original_Long that recorded accuracies lower than models Original_Modified and Original_Original_Long in sequence. Figure 4-61

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	**97.55%**	**70.65%**	73.21%	60.31%
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Original_Modified	97.58%	85.45%	**91.52%**	**89.26%**
Original_Compound	97.64%	86.84%	69.09%	61.48%
Original_Original_Long	97.67%	83.95%	**49.34%**	**59.73%**
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	97.56%	**89.68%**	81.91%	78.19%
Modified	**97.85%**	67.02%	79.12%	60.10%
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Modified_Compound	**97.59%**	80.71%	78.08%	78.13%
Modified_Original_Long	**97.59%**	**62.84%**	**66.02%**	**43.52%**
Modified_Modified_Long	97.63%	87.17%	**85.95%**	**80.10%**
Modified_Compound_Long	97.63%	**89.59%**	82.84%	79.89%
Compound	**97.52%**	**90.11%**	89.88%	**94.42%**
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	**62.77%**
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Compound_Original_Long	97.65%	83.51%	**93.18%**	93.29%
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-31 All models of three U-Nets with same connections between the first two U-Nets and variant connections between the second and third U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group

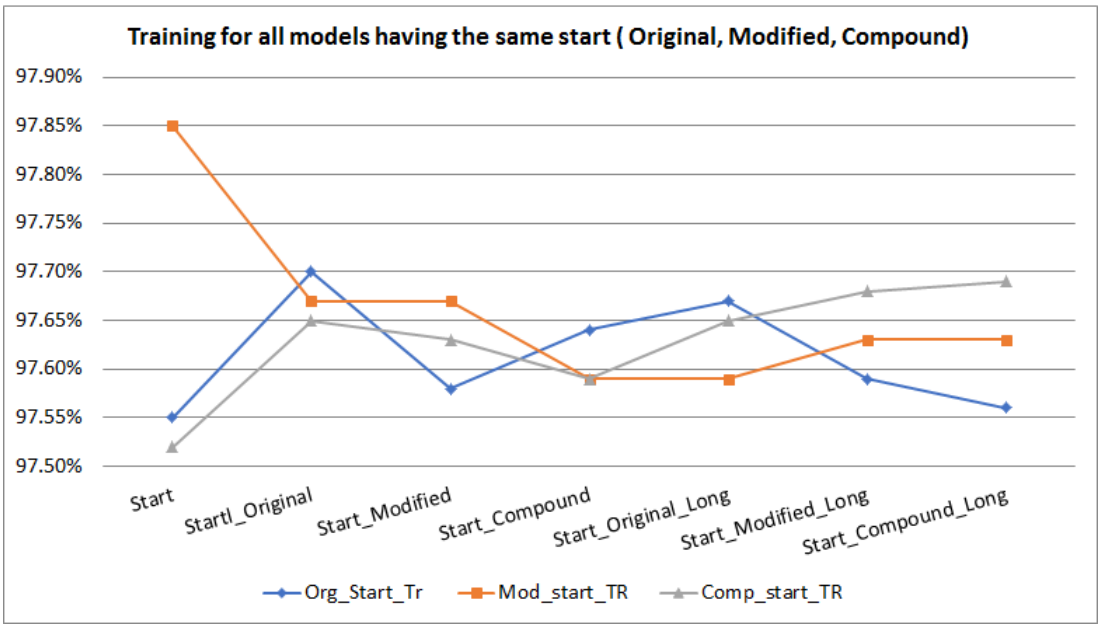


Figure 4-58 Training accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets

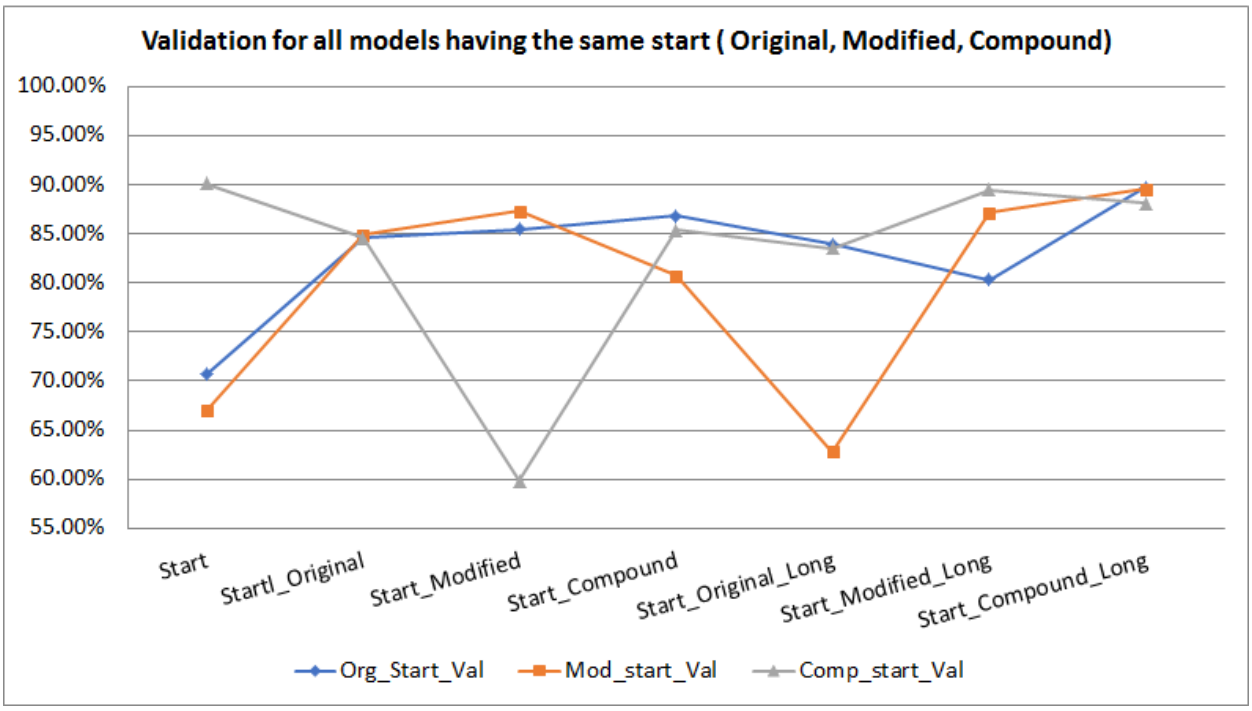


Figure 4-59 Validation accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Nets

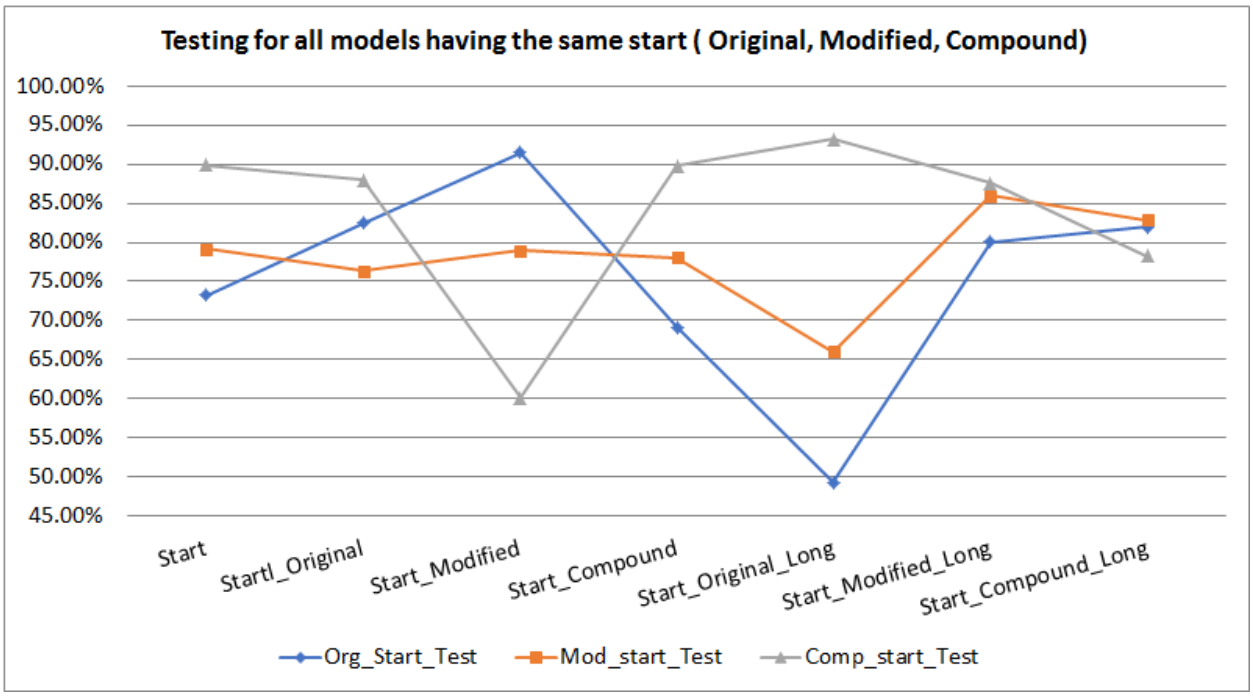


Figure 4-60 Testing (Test_Avg) accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Net

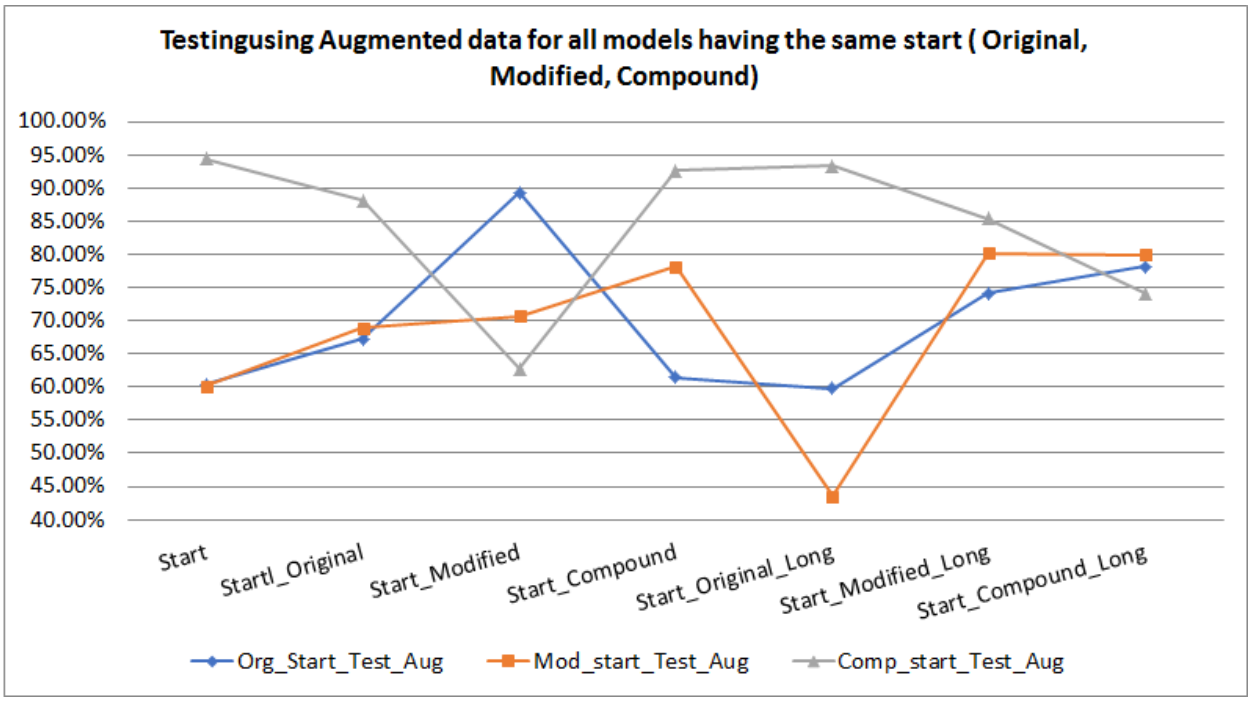


Figure 4-61 Testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with/without long connections that start with Original Versus start with Modified versus start with Compound connections between the first two U-Nets and variant connections between the second and third U-Net

- **Compare all Models have the same end connections between the second and third U-Net**

In this section, the models based on 2U-Nets structure with filters 32-512 would be compared with all other models based on 3U-Net models where the second and third U-Nets having the same connections as the 2U-Net based model with/without long connections. For example, the 2_U-Net model with Modified connections would be compared with all 3U-Net models end with Modified connections between the second and the third U-Net (Modified, Original_Modified, Modified_Modified, Compound_Modified, Original_Modified_Long, Modified_Modified_Long, Compound_Modified_Long).

❖ **Models end with Original connections**

All the models included in this section consist of 3U-Nets where the second and third U-Nets are connected based on the **Original** structure while the first and the second U-Net connected with one of three structures (Original, Modified, Compound). The other models have the same structure as the previous ones in addition to the Long connections.

The Compound_Original_Long model recorded the maximum accuracy for Test_Avg and Test_Avg_Aug factors while the maximum for training and validation recorded for Original_Original and Modified_Original models respectively. The minimum accuracy for Validation and Test_Avg_Aug recorded for the Modified_Original_Long model while the Original and Original_Original_Long models recorded the minimum accuracy for Training and Test_Avg respectively. Table 4-32

For Training and validation factors, all models used Long connections recorded lower accuracy than the models without long connections. Figure 4-62. Test_Avg and Test_Avg_Aug having the same curve behavior, the models with long connections recorded

lower accuracies than the same model structure without long connections except for Compound_Original_Long model which achieved accuracy higher than Compound_Original model. Figure 4-63

The two U-Net based model with Original connections achieved the minimum accuracy over all of three U-Nets with/without long connections for Training, while recorded the second minimum accuracy model for Validation after Modified_Original_Long model and the third minimum for Test_Avg after Original_Original_Long and Modified_Original_Long models and third minimum for Test_Avg_Aug after Modified_Original_Long and Original_Original_Long in sequence. The original model recorded accuracy lower than all 3U-Net models without long connections for Test_Avg and Test_Avg_Aug. Table 4-32

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	**97.55%**	70.65%	73.21%	60.31%
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Modified_Original	97.67%	**84.87%**	76.36%	68.83%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Compound_Original_Long	97.65%	83.51%	**93.18%**	**93.29%**

Table 4-32 All models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)

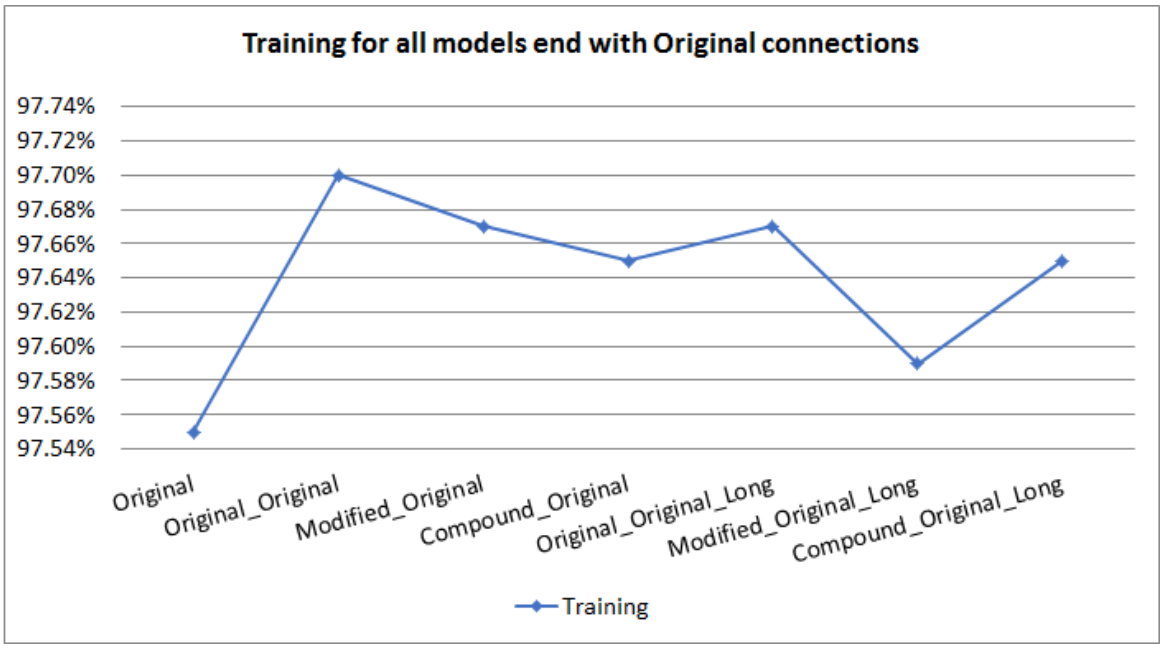


Figure 4-62 Training accuracy for all models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets

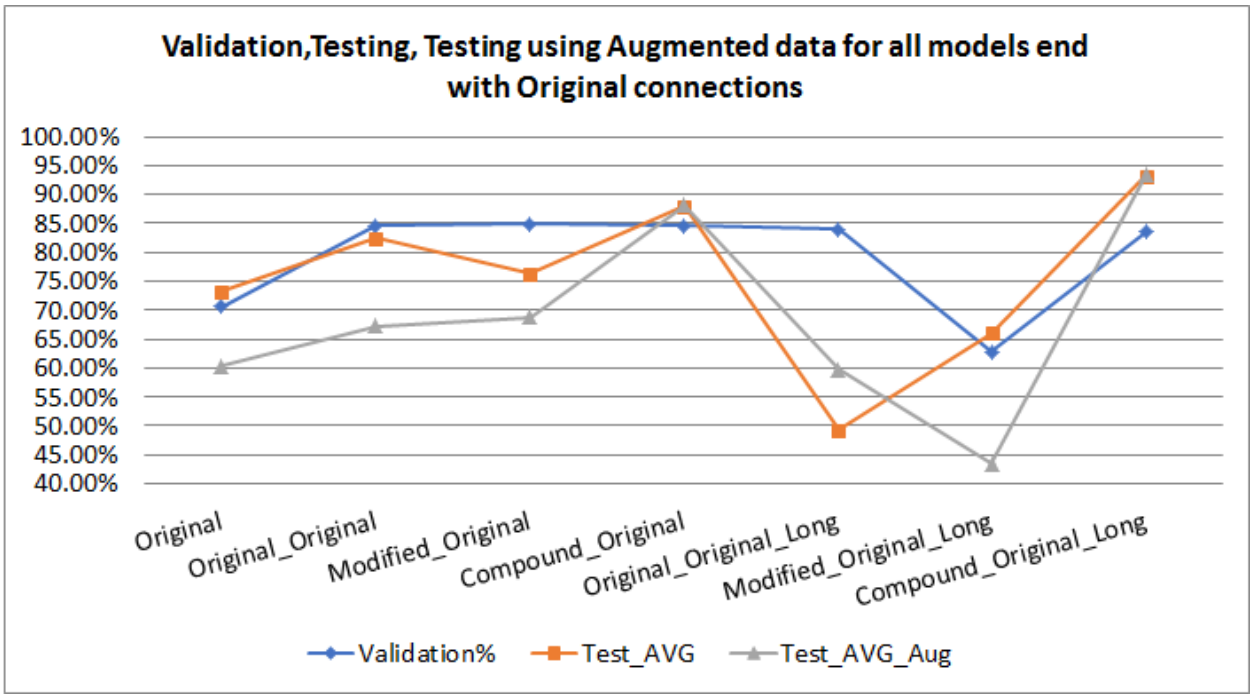


Figure 4-63 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Original connections between the second and third U-Nets and variant connections between the first and second U-Nets

- **Models end with Modified connections**

All the models included in this section consist of 3U-Nets where the second and third U-Nets are connected based on the **Modified** structure while the first and the second U-Net connected with one of three structures (Original, Modified, Compound). The other models have the same structure as the previous ones in addition to the Long connections.

The Original_Modified model recorded the maximum accuracy for Test_Avg and Test_Avg_Aug factors while the maximum for training and validation recorded for Modified and Compound_Modified_Long models respectively. The minimum accuracy for Validation and Test_Avg recorded for the Compound_Modified model while the Modified and Original_Modified models recorded the minimum accuracy for Training and Test_Avg_Aug respectively. Table 4-33

For Training factor, all models with Long connections recorded higher accuracy than the models without long connections except for Modified_Modified_Long model which recorded accuracy lower than Modified_Modified model. Figure 4-64

For Validation factor, all models with Long connections recorded lower accuracy than the models without long connections except for Compound_Modified_Long model which recorded accuracy higher than Compound_Modified model. Figure 4-65

Test_Avg and Test_Avg_Aug having the same curve behavior, the models with long connections recorded higher accuracies than the same model structure without long connections except for Original_Modified_Long model which achieved accuracy lower than Original_Modified model. Figure 4-65

The two U-Net based model with Modified connections achieved the maximum accuracy over all of three U-Nets with/without long connections for Training and minimum for Test_Avg_Aug, while recorded the second minimum accuracy model for Validation after

Compound_Modified model and the third minimum for Test_Avg after Compound_Modified and Modified_Modified models. Table 4-33

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Modified	**97.85%**	67.02%	79.12%	**60.10%**
Original_Modified	**97.58%**	85.45%	**91.52%**	**89.26%**
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Compound_Modified	97.63%	**59.82%**	**60.16%**	62.77%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Compound_Modified_Long	97.68%	**89.47%**	87.56%	85.41%

Table 4-33 All models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)

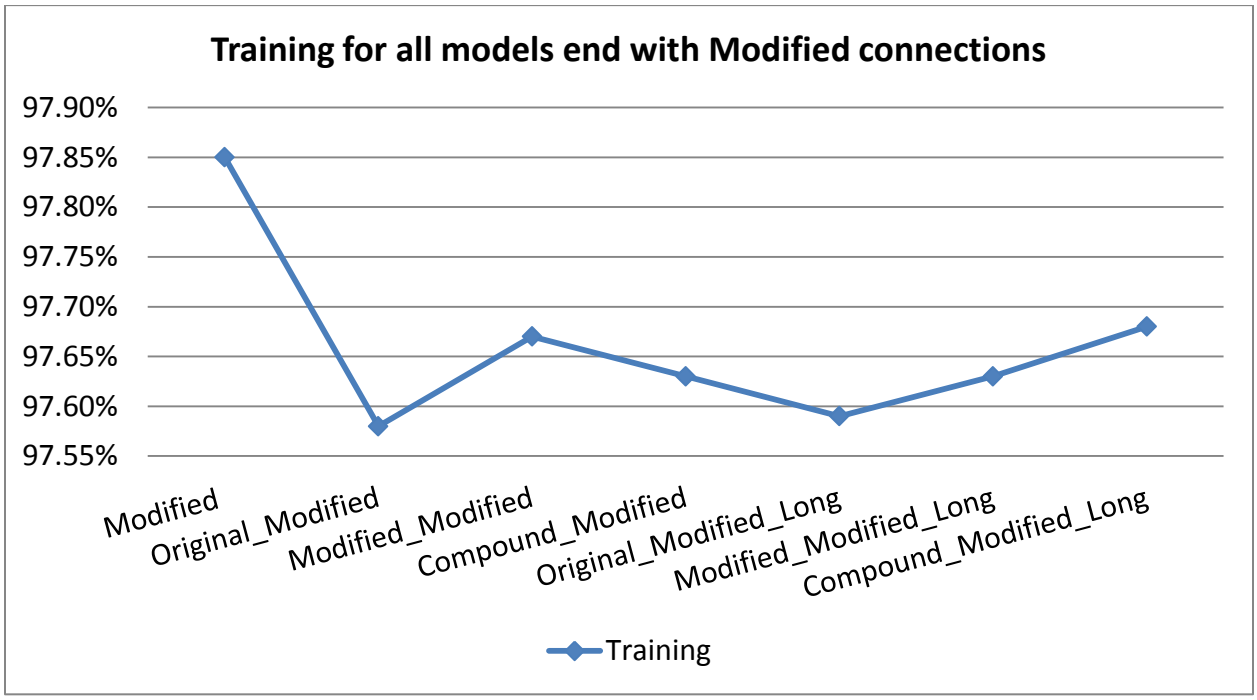


Figure 4-64 Training accuracy for all models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets

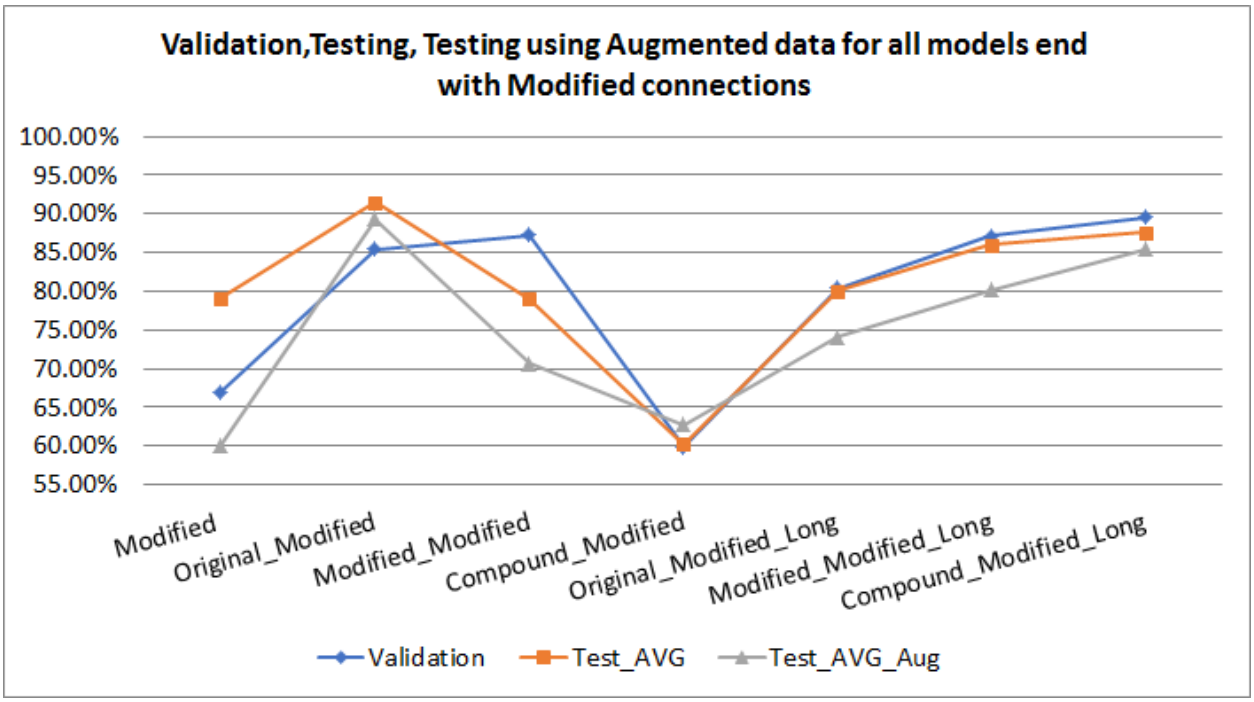


Figure 4-65 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Modified connections between the second and third U-Nets and variant connections between the first and second U-Nets

- **Models end with Compound connections**

All the models included in this section consist of 3U-Nets where the second and third U-Nets are connected based on the **Compound** structure while the first and the second U-Net connected with one of three structures (Original, Modified, Compound). The other models have the same structure as the previous ones in addition to the Long connections.

The Compound model recorded the maximum accuracy for Validation, Test_Avg and Test_Avg_Aug factors while the maximum for training recorded for Compound_Compound_Long models. The minimum accuracy for Test_Avg_Aug and Test_Avg recorded for the Original_Compound model while the Compound and Modified_Compound models recorded the minimum accuracy for Training and Validation respectively.

For Training factors, all models used Long connections recorded higher accuracy than the models without long connections except for Original_Compound_Long model which recorded accuracy lower than Original_Compound model. For Training, all models with long connections recorded higher accuracy than the respective models without long connections except for model Original_Compound_Long that recorded accuracy lower than Original_Compound. Figure 4-66

For Validation, all models with long connections recorded higher accuracy than the respective models without long connections.

For Test_Avg and Test_Avg_Aug, the models with long connections recorded higher accuracies than the same model structure without long connections except for Compound_Compound_Long model which achieved accuracy lower than Compound_Compound model. Figure 4-67

The two U-Net based model with Compound connections achieved the maximum accuracy over all of three U-Net models with/without long connections for Validation, Test_Avg and Test_Avg_Aug, while recorded the minimum accuracy for Training. Table 4-34

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**
Original_Compound	97.64%	86.84%	**69.09%**	**61.48%**
Modified_Compound	97.59%	**80.71%**	78.08%	78.13%
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Original_Compound_Long	97.56%	89.68%	81.91%	78.19%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-34 All models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red)

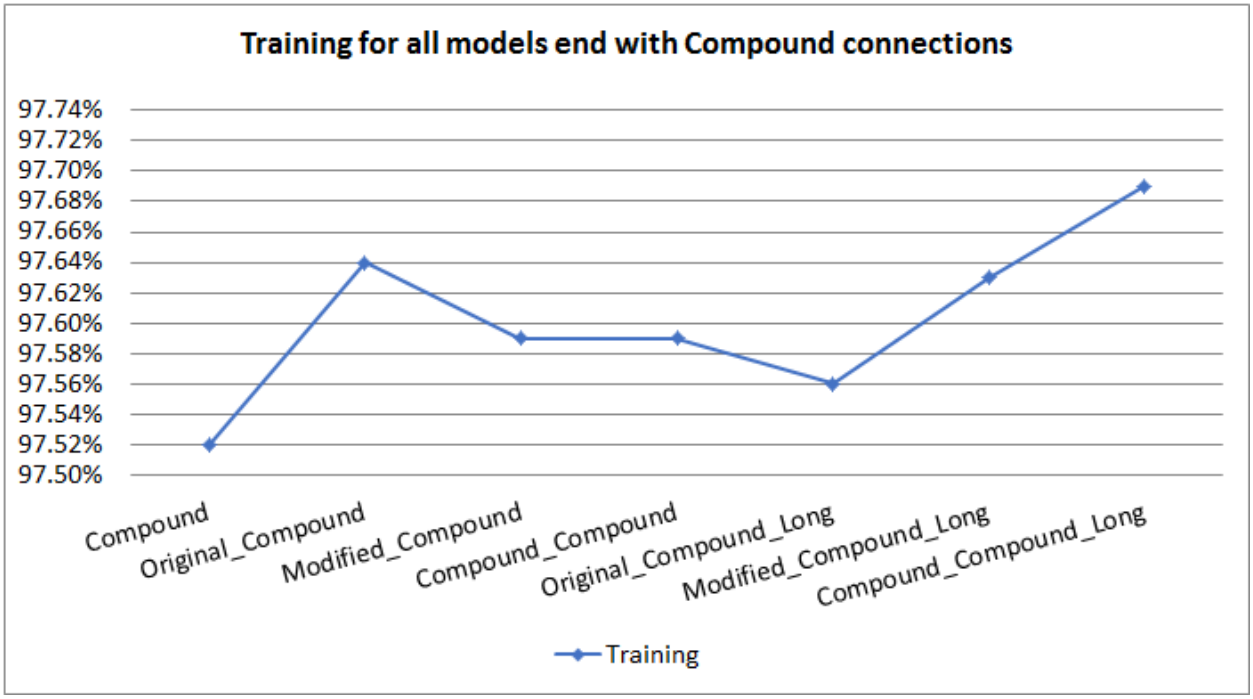


Figure 4-66 Training accuracy for all models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

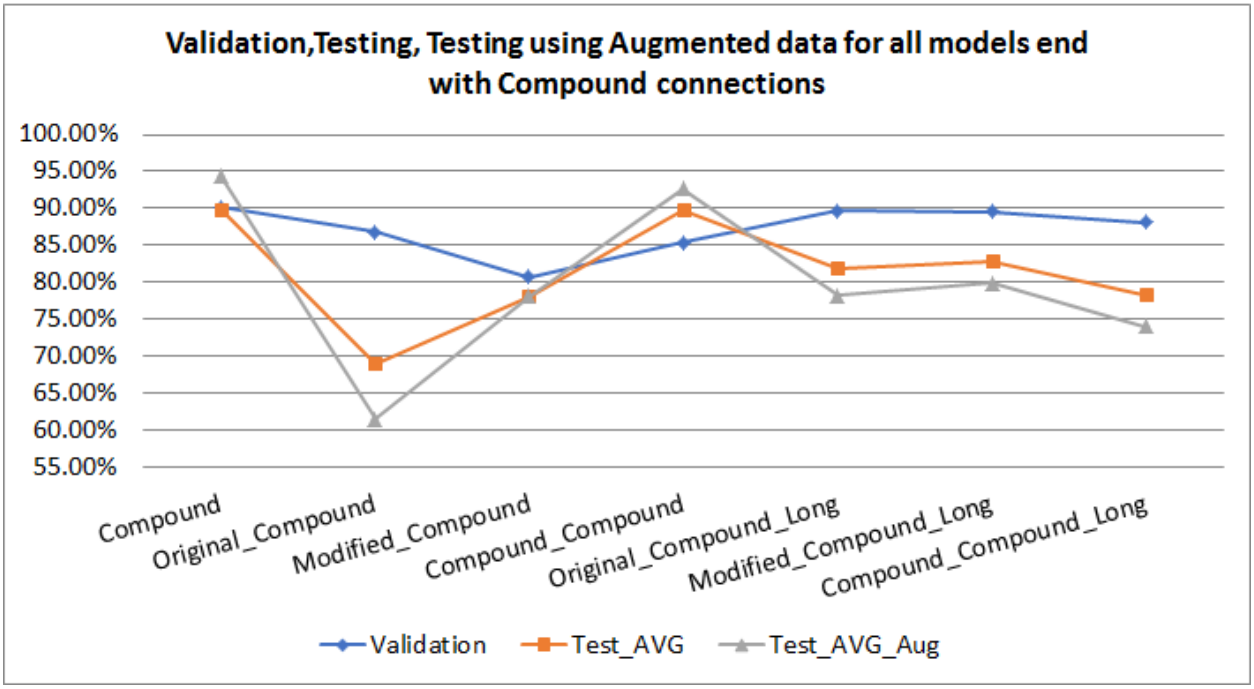


Figure 4-67 Validation, Testing (Test_Avg), testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

- **All models end with the same connections (Original, Modified, Compound)**

In this section, the curve of the accuracies resulted from the models that ended with the same connections between the second and third U-Net (for Ex, ended with Compound connections) is compared with curve of results of the models ended with the other two connections (Original, Modified). For example, each point on any curve will be compared with the point of the other curves for the same models with the other ends. For example, the model ends with Original will be compared with the respective models ended with Modified and Compound connection (Compound_Original, Compound_Modified, Compound_Compound).

The **Training** factor accuracies show that, the models ended with Original connections recorded higher training accuracy than the models ended with Compound for the two U-Net model and 4 of 6 models based on 3U-Net while the remaining 2 models (Modified_Original_Long, Compound_Original_Long) recorded accuracy lower than the same models ended with Compound (Modified_Compound_Long, Compound_Compound_Long). The models ended with Original connections have a slightly different behavior when compared with the models ended with Modified connections where the models ended with original connections recorded higher accuracy with 3 of 6 models based on 3U-Net while recorded same accuracy with the model (Modified_Original = Modified_Modified) but recorded accuracy with the remaining 2 models (Modified_Original_Long, Compound_Original_Long) lower than (Modified_Modified_Long, Compound_Modified_Long) as in the comparison with the models ended Compound models. The Original Model recorded accuracy lower than the Modified Model when using 2U-Net mode. Figure 4-68.

The models ended with Modified connections recorded accuracy higher than 4 of 7 of the similar models ended with Compound connections including the 2U-Net based model same accuracy with model (Modified_Modified_Long and Modified_Compound_Long) while the accuracy for the models Original_Modified, Compound_Modified_Long were lower than models (Original_Compound and Compound_Compound_Long) in sequence.

For **Validation** factor, for 5 of 7 models including the 2U-Net based models, the models ended with Compound connections recorded higher Validation accuracy than both the models ended with Modified and Original connections except for models Modified_Compound where the accuracy was lower than both (Modified_Original, Modified_Compound) and model Compound_Compound_long where the accuracy was Lower than the model Compound_Modified_long and higher than the model Compound_Original_long. Figure 4-69.

The models ended with Modified connections recorded accuracy higher than 4 of 6 of the similar models ended with Original connections while recorded lower accuracy for the remaining 2 of 6 models and the model based on 2U-Net where the models Modified, Compound_Modified, and Original_Modified_Long recorded accuracy lower than the models Original, Compound_Original, Original_Original_Long in sequence.

For **Testing** factor (**Test_Avg**), The models ended with Compound connections recorded higher Test_Avg accuracy than the models ended with Original connections except for models Original_Compound and Compond_Compond_Long where recorded accuracy lower than the models Original_Original and Compond_Original_Long. The models ended with Compound connections recorded accuracy lower than the models ended with Modified connections except for the models Compound, Compound_Compound, Original_Compound_Long where the accuracies are higher than the models Modified, Compound_Modified, Original_Modified_Long. Figure 4-70.

The models ended with Modified connections recorded accuracy higher than the similar models ended with Original connections except for two models Compound_Modified and Compound_Modified_Long where the recorded accuracies lower than models Compound_Original and Compound_Original_Long in sequence.

The **Test_Avg_Aug (Testing using augmented data) shows that**, The models ended with Compound connections recorded higher Test_Avg_Aug accuracy than the models ended with Original connections except for models Original_Compound and Compond_Compond_Long where recorded accuracy lower than the models Original_Original and Compond_Original_Long.

The models ended with Compound connections recorded accuracy Higher than the models ended with Modified connections except for the models Original_Compound and Compond_Compond_Long and where the accuracies are lower than the models Original_Modified and Compound_Modified_Long in sequence. Figure 4-71

The models ended with Modified connections recorded accuracy higher than the similar models ended with Original connections except for two models Compound_Modified and Compound_Modified_Long where the recorded accuracies lower than models Compound_Original and Compound_Original_Long in sequence while the original model slightly higher than the Modified model with accuracy 60.31% and 60.10% respectively. Table 4-35

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original	**97.55%**	70.65%	73.21%	60.31%
Original_Original	**97.70%**	84.62%	82.42%	67.22%
Modified_Original	97.67%	**84.87%**	76.36%	68.83%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Compound_Original_Long	97.65%	83.51%	**93.18%**	**93.29%**

Modified	**97.85%**	67.02%	79.12%	**60.10%**
Original_Modified	**97.58%**	85.45%	**91.52%**	**89.26%**
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	**59.82%**	**60.16%**	62.77%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Compound_Modified_Long	97.68%	**89.47%**	87.56%	85.41%
Compound	**97.52%**	**90.11%**	**89.88%**	**94.42%**
Original_Compound	97.64%	86.84%	**69.09%**	**61.48%**
Modified_Compound	97.59%	**80.71%**	78.08%	78.13%
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Original_Compound_Long	97.56%	89.68%	81.91%	78.19%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Compound_Long	**97.69%**	88.14%	78.30%	74.07%

Table 4-35 All models of three U-Nets with the same connections between the second and third U-Nets and variant connections between the first and second U-Nets, Highest accuracy (Green) lowest accuracy (Red) within each group

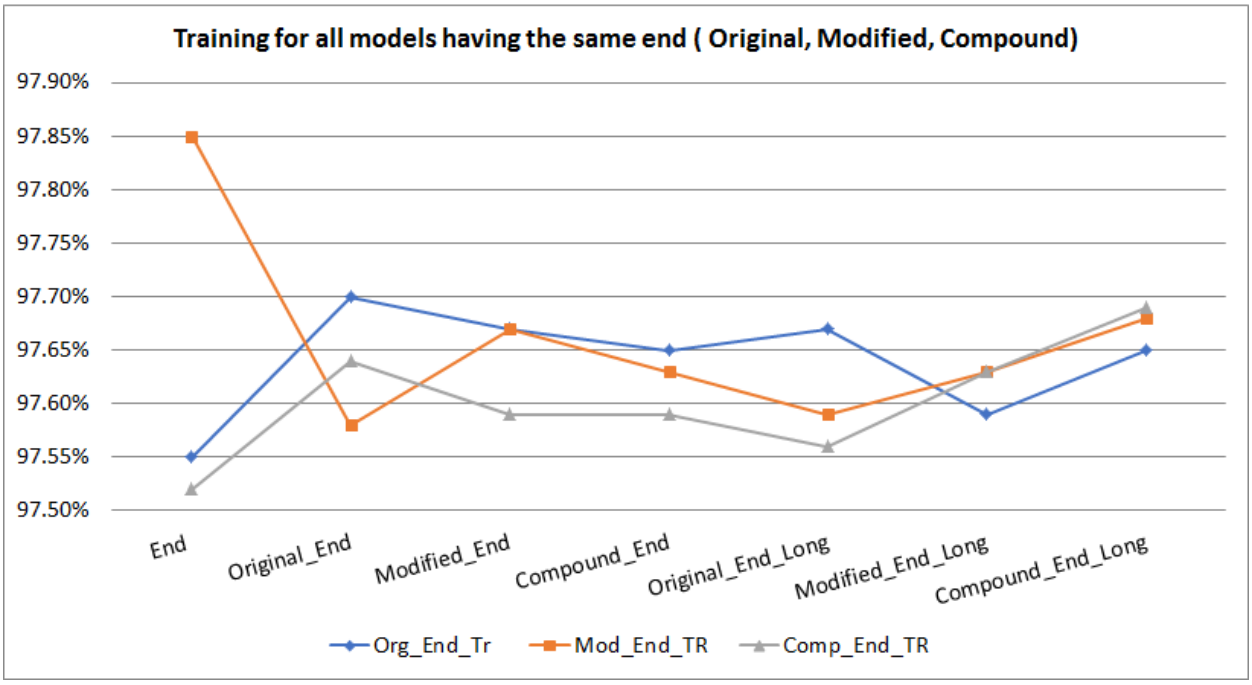


Figure 4-68 Training accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

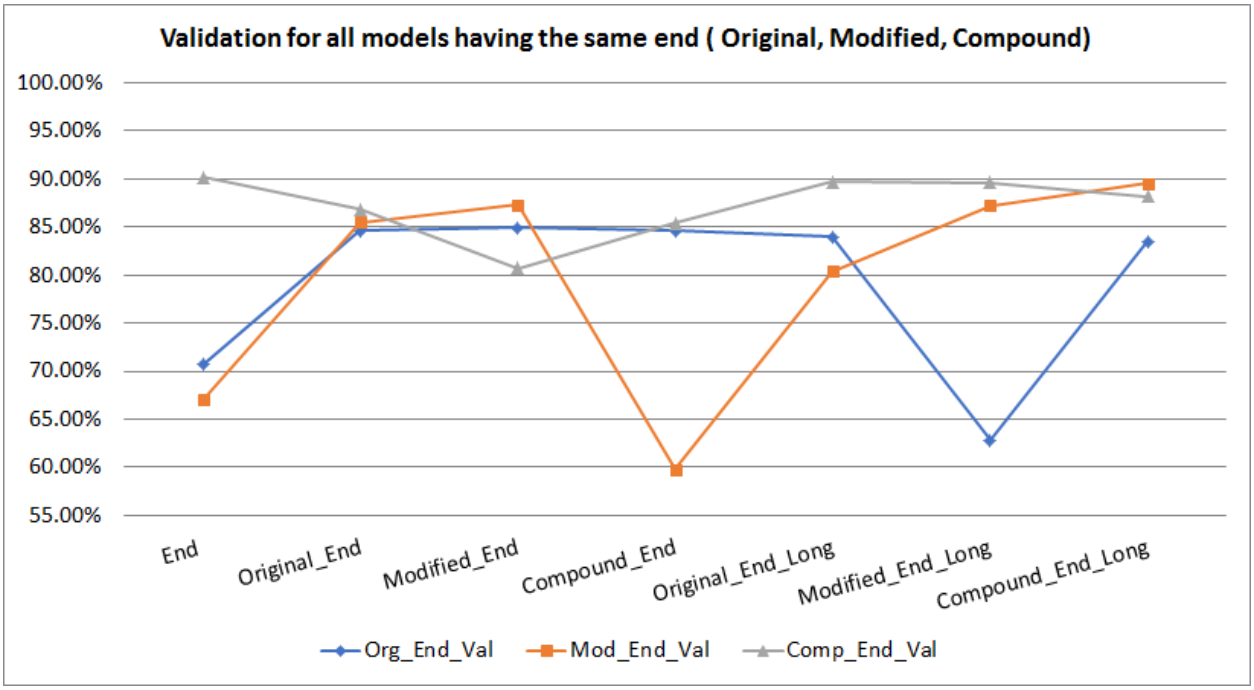


Figure 4-69 Validation accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

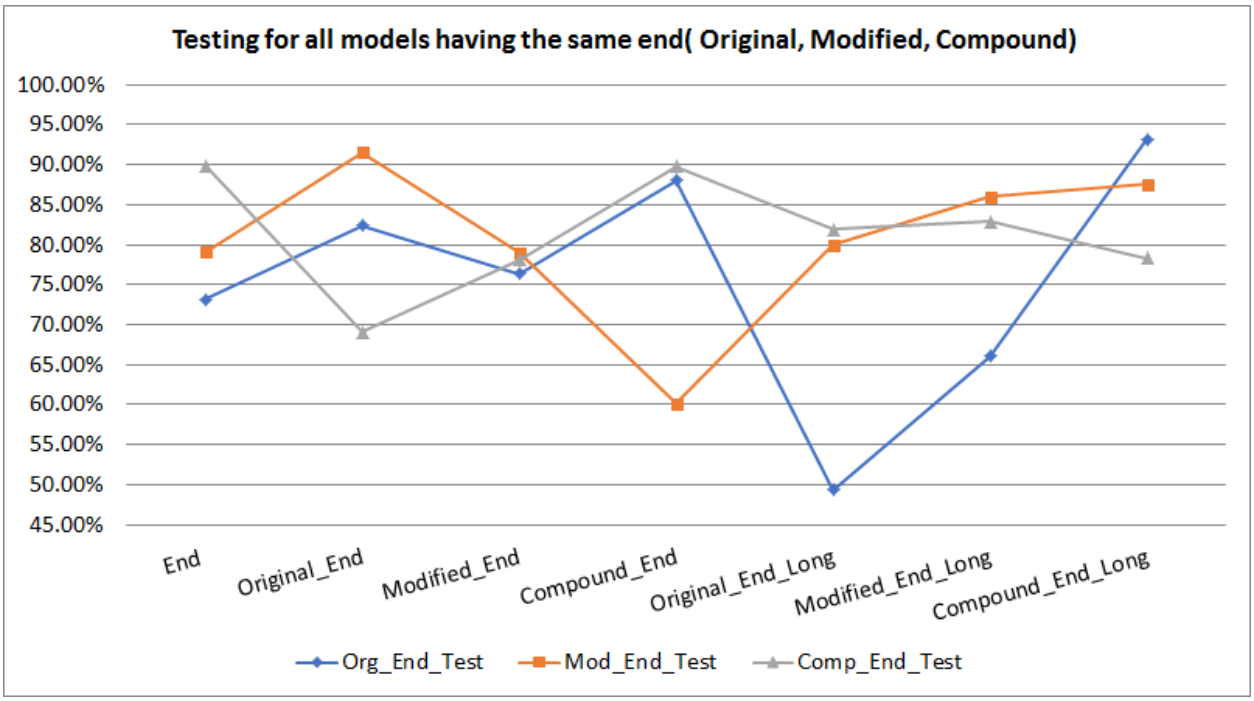


Figure 4-70 Testing (Test_Avg) accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

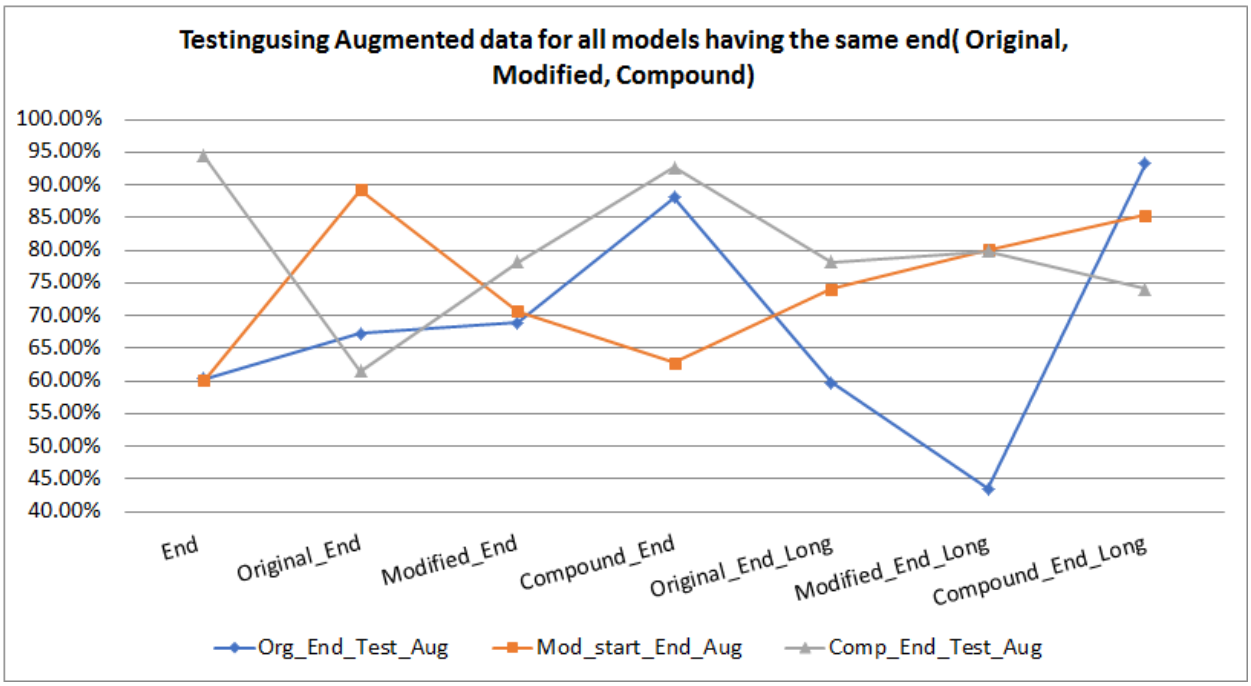


Figure 4-71 Testing using augmented data (Test_Avg_Aug) accuracy for all models of three U-Nets with/without long connections that end with Original Versus end with Modified versus end with Compound connections between the second and third U-Nets and variant connections between the first and second U-Nets

4.1.3.4 Compare all models

The **Training** accuracy for all models is very close and varies from 97.85% to 97.52% Table 5.2. While the Compound model achieved the maximum **Test_Avg_Aug** (Testing using augmented data) (94.42%) and maximum **Validation** (90.11%), Compound model also got the minimum **Training** accuracy (97.52%).

The modified model recorded the maximum **Training** accuracy (97.85%), while the maximum **Test_Avg** recorded for the Compound_Original_Long model (93.18%). The minimum accuracy for validation (59.82%), minimum Test_Avg (49.34%), and minimum Test_Avg_Aug (43.52%) are recorded for Compound_Modified model, Original_Original_Long model, and Modified_Original_Long model respectively. Table 4-36

Model	Training	Validation	Test_Avg	Test_Avg_Aug
Original_Original	97.70%	84.62%	82.42%	67.22%
Original_Modified	97.58%	85.45%	91.52%	89.26%

Original_Compound	97.64%	86.84%	69.09%	61.48%
Modified_Original	97.67%	84.87%	76.36%	68.83%
Modified_Modified	97.67%	87.27%	78.99%	70.67%
Modified_Compound	97.59%	80.71%	78.08%	78.13%
Compound_Original	97.65%	84.60%	87.99%	88.08%
Compound_Modified	97.63%	59.82%	60.16%	62.77%
Compound_Compound	97.59%	85.39%	89.76%	92.64%
Original_Original_Long	97.67%	83.95%	**49.34%**	59.73%
Original_Modified_Long	97.59%	80.35%	80.01%	74.08%
Original_Compound_Long	97.56%	89.68%	81.91%	78.19%
Modified_Original_Long	97.59%	**62.84%**	66.02%	**43.52%**
Modified_Modified_Long	97.63%	87.17%	85.95%	80.10%
Modified_Compound_Long	97.63%	89.59%	82.84%	79.89%
Compound_Original_Long	97.65%	83.51%	**93.18%**	93.29%
Compound_Modified_Long	97.68%	89.47%	87.56%	85.41%
Compound_Compound_Long	97.69%	88.14%	78.30%	74.07%
Original	97.55%	70.65%	73.21%	60.31%
Modified	**97.85%**	67.02%	79.12%	60.10%
Compound	**97.52%**	**90.11%**	89.88%	**94.42%**

Table 4-36 All models of 2U-Nets, 3U-Nets, and 3U-Nets with long connection. Max accuracy (Green), Min accuracy (Red)

For the **Training** Factor, The training accuracy for all models with the range of 0.33% between 97.85% and 97.52%. The 2U-Net model using Modified connection recorded the maximum accuracy 97.85% while the compound and Original recorded the minimum and second minimum over all the models. All 3U-Net based models recorded accuracy higher than Original and Compound model. While the maximum three models based on 3U-Nets are Original_Original, Compound_Compound_Long, and Compound_Modified_Long(97.70, 97.69, 79.68) in sequence, the minimum 2 accuracies recorded for Original_Compound_Long, and Original_modified (97.56%, 97.58%)in sequence. The remaining models categorized into 3 groups, **Group 1** contains the next best 3 models with same accuracy (97.67%) (Modified_Original, Modified_Modified, Original_Original_Long). **Group2** contains 6 models (Compound_Original, Compound_Original_Long, Original_Compound, Compound_Modified, Modified_Modified_Long, Modified_Compound_Long) with accuracies (97.65%, 97.65%,

97.64%, 97.63%, 97.63%, 97.63%), Group3 contains 4 models with the same accuracy 97.59% (Modified_Compound, Compound_Compound, Original_Modified_Long, Modified_Original_Long). Figure 4-72

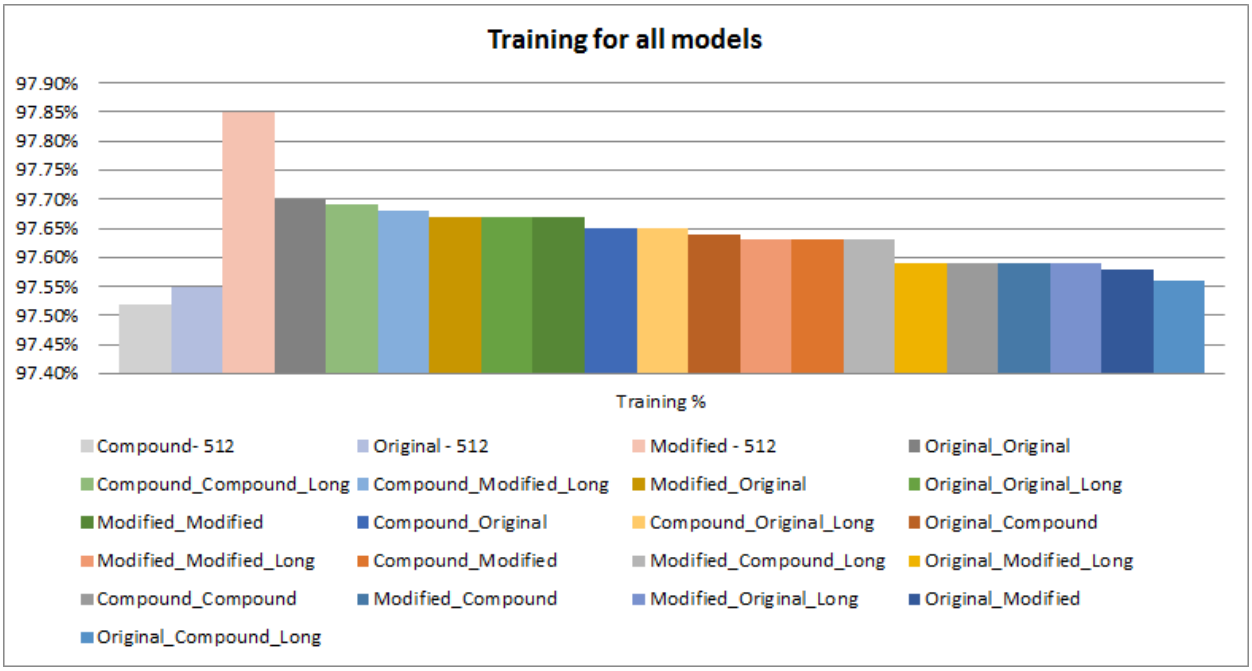


Figure 4-72 Training accuracy for all models

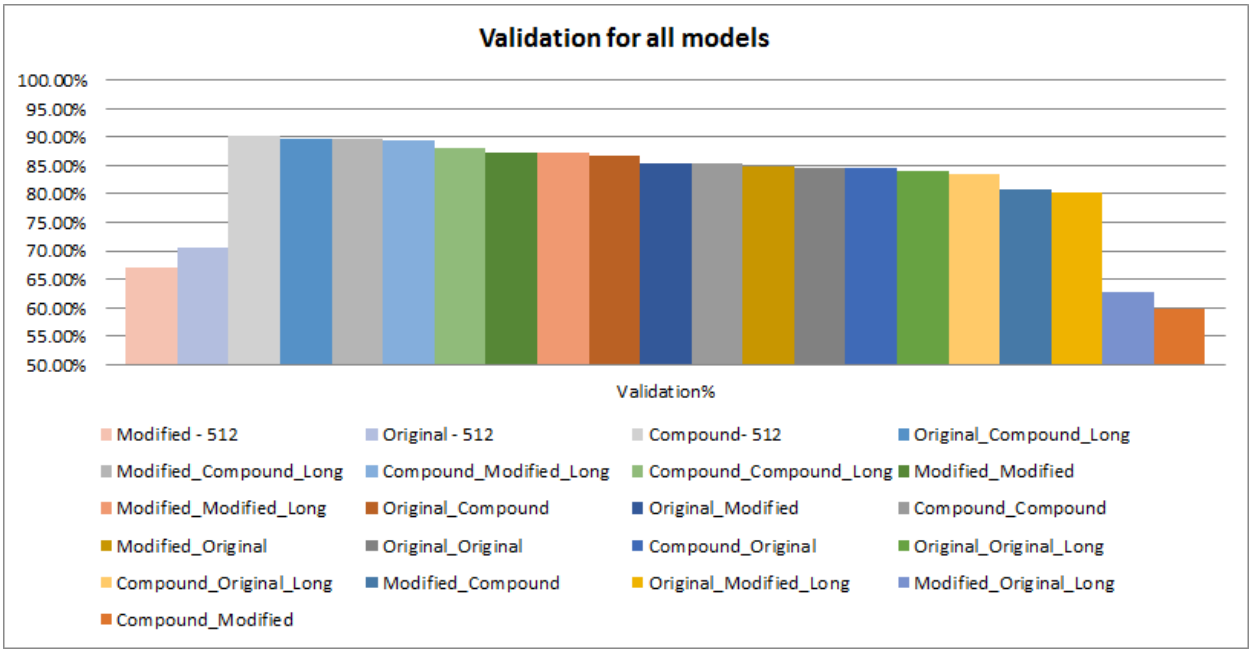


Figure 4-73 Validation accuracy for all models

The **validation** accuracy shows that, the compound model recorded the maximum accuracy (90.11%) while the modified and Original models recorded the third and fourth minimum (67.02%, 70.65%) respectively.

For the 3U-net based models, the first and second minimum recorded for Compound_Modified and Modified_Original_Long with values 59.82% and 62.84% which are the two minimums over all models while the first three maximum validation accuracies recorded for the models Original_Compound_long, Modified_Compound_Long, and Compound_Modified_Long with values 89.68%, 89.59%, and 89.47% respectively. The remaining models can be categorized into two groups based on the accuracy. Group 1 contains 3 models with accuracies within the range 88% to 86%, these models are (Compound_Compound_Long, Modified_Modified, Modified_Modified_Long, Original_Compound) with accuracies 88.14%, 87.27%, 87.17%, 86.84% in sequence while Group2 contains 7 models where the accuracies within the range from 85% to 83%, the

models are (Original_Modified, Compound_Compound, Modified_Original , Original_Original , Compound_Original , Original_Original_Long , Compound_Original_Long) with the respective accuracy (85.45%, 85.39%, 84.87%, 84.62%, 84.60%, 83.95%, 83.51%). Figure 4-73

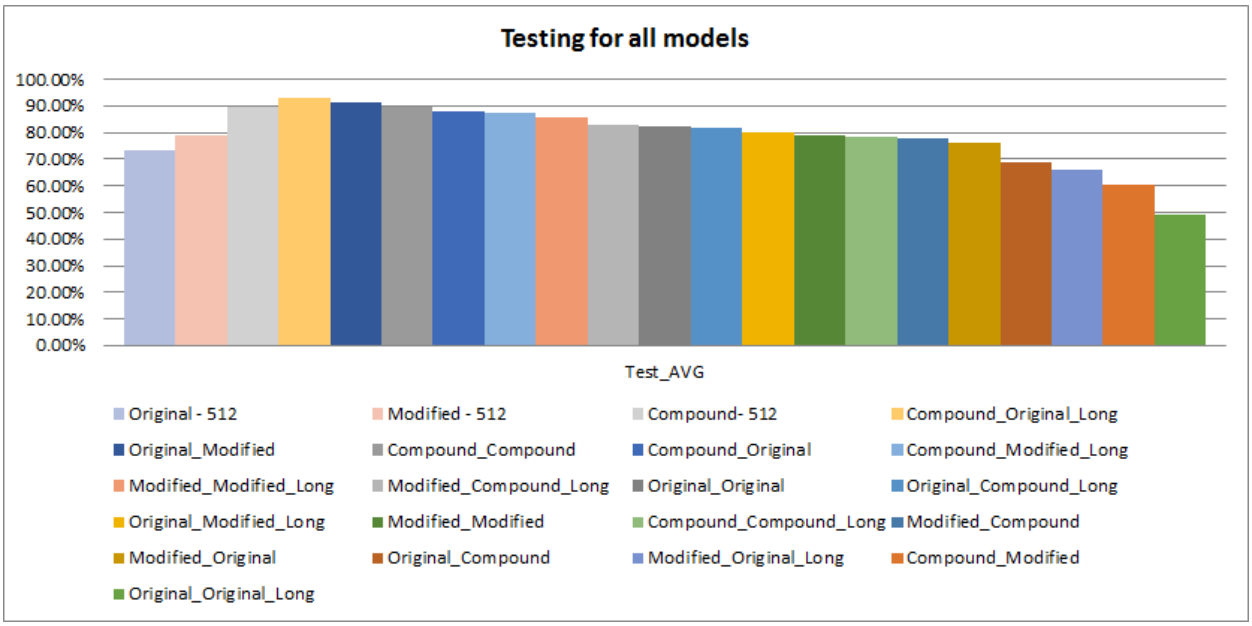


Figure 4-74 Testing accuracy for all models

The **Testing** accuracy for all models shows that, for the models based on 2U-Net, Compound, Original, and Modified reached the 3rd maximum, 5th minimum, and 10th minimum respectively with values (89.88%, 73.21%, and 79.12%). The overall maximum four accuracies recorded of the models The models Compound_Original_Long, Original_Modified, Compound, and Compound_Compound with values 93.18%, 91.52%, 89.88% and 89.76% respectively while the minimum five accuracies recorded for models Original_Original_Long, Compound_Modified, Modified_Original_Long, Original_Compound, and Original with values 49.34%, 60.16%,66.02%,69.09%, and 73.21% respectively. The remaining models can be categorized to 3 groups. Group1 contains the models Compound_Original ,Compound_Modified_Long with accuracies 87.99% ,87.56% while

Group2 contains 5 models with accuracy in range from 80% to 86% Modified_Modified_Long, Modified_Compound_Long , Original_Original, Original_Compound_Long, Original_Modified_Long, 85.95%, 82.84%, 82.42%, 81.91%, 80.01%). Group3 contains 5 models where the accuracy range between 79% and 76% (Modified, Modified_Modified, Compound_Compound_Long, Modified_Compound, Modified_Original 79.12%, 78.99%, 78.30%, 78.08%,76.36%) respectively. Figure 4-74

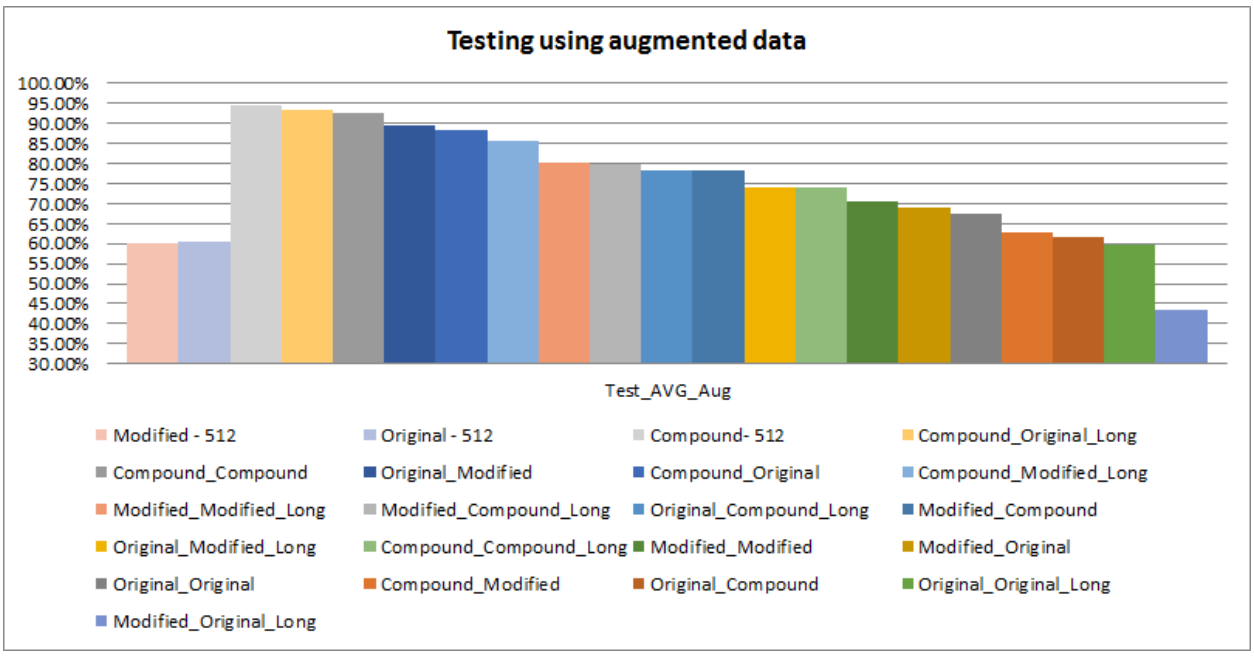


Figure 4-75 Testing with augmented data accuracy for all models

For **Test_Avg_Aug**, the 2U-Net models Compound, Modified and Original recorded the maximum accuracy and the 3rd and 4th minimum over all the models with values 94.42%, 60.10%, and 60.31% respectively while the 1st and 2nd minimum recorded for Modified_Original_Long and Original_Original_Long with values 43.52% and 59.73% respectively.For the 3U-Net based models, the maximum 3 accuracies over 90% recorded for models Compound_Original_Long, Compound_Compound with values 93.29%, 92.64%. The remaining models could be categorized based on accuracy range with 5% step from 60% to 90% into 6 groups. Group1 contains 2 models Original_Modified, Compound_Original with accuracies 89.26%, 88.08%. Group2 contains 2 models

Compound_Modified_Long, Modified_Modified_Long with accuracies 85.41%, 80.10% respectively. Group3 consists of 3 models Modified_Compound_Long, Original_Compound_Long, Modified_Compound with values 79.89%, 78.19%, 78.13% respectively. Group4 contains 3 models Original_Modified_Long, Compound_Compound_Long , Modified_Modified with values 74.08%, 74.07%, 70.67% respectively. Group5 consists of 2 models Modified_Original Original_Original with accuracies 68.83%, 67.22%. Group6 contains 2 models Compound_Modified, Original_Compound with values 62.77%, 61.48% respectively. Figure 4-75

For the models based on 2U-Net, The validation accuracy recorded 1 of 3 the models >= 90% and 1of 3 between 70% and 80% and 1 of 3 between 60% and 70% while for Testing 1 of 3 models between 80 and 90% and 2of 3 models between 70% and 80%. Testing using augmented data recorded 1 of 3 models > 90% and 2 of 3 models between 60%and 70%. Table 4-37

For 3U-Net models, the **validation** accuracy recorded 88.88% of the models recorded accuracy between 80% and 90% for both with/without long connections while 11.11% with accuracy from 50% to 60% with long connection and 11.11% of the models with accuracy 60% to 70% without long connection. The **Testing** Factor, 44.44% of the 3U-Net models recorded accuracy >= 80% including 11.11% of them > 90% while the remaining models divided into 33.33% with accuracy between 70% and 80% and 22.22% between 60% and 70%. Using 3U-Net models with long connection showed better performance for 66.66% of the models >80% including 11.11% > 90% while the remaining models recorded accuracy within ranges (from 40% to 50%) and (from 60% to 70%) and (from 70% to 80%) with 11.11% of the models for each of them. The accuracies for **Test_Avg_Aug** (Testing using augmented data) shows that, 33.33% of the 3U-Net models recorded accuracy > 80% including 11.11% > 90% while 22.22% of the models with accuracy between 70% and 80% and 44.44% of the models with accuracy between

60% and 70%. 33.33% of the model based on 3U-Net with long connection recorded accuracy > 80% including 11.11% > 90% (same as the models without long connections) while 44.44% of the models recorded accuracy between 70% and 80% and 11.11% recorded accuracy in both ranges(from 50% to 60%) and (from 40% to 50%). Table 4-37

Accuracy Range	Models type	Number of Models		
		Validation	Test_Avg	Test_Avg_Aug
Accuracy >= 90%	3U-Net	0	1	1
	3U-Net Long	0	1	1
	2U-Net	1	0	1
90% > Accuracy >= 80%	3U-Net	8	3	2
	3U-Net Long	8	5	2
	2U-Net	0	1	0
80% > Accuracy >= 70%	3U-Net	0	3	2
	3U-Net Long	0	1	4
	2U-Net	1	2	0
70% > Accuracy >= 60%	3U-Net	0	2	4
	3U-Net Long	1	1	0
	2U-Net	1	0	2
60% > Accuracy >= 50%	3U-Net	1	0	0
	3U-Net Long	0	0	1
	2U-Net	0	0	0
50% > Accuracy >= 40%	3U-Net	0	0	0
	3U-Net Long	0	1	1
	2U-Net	0	0	0

Table 4-37 number of models versus the accuracy ranges for Validation, Test_Avg, Test_Avg_Aug for all models (2U-Net, 3U-Nets, 3U-Nets with long connection)

4.1.3.5 Discussion for skip connections

This section will focus on the discussion of the findings for the results of the experiments related to the models based on 3 connected U-Nets with/without long connections while the 2U-Net models discussed in more details in section 4.1.3.2

The study of the 3U-Net based models can concluded the findings into, 1- 3U-Nets model with/without long connections have higher performance than the 2U-Nets models with original and Modified. 2- All 3U-Net models started with Compound recorded better performance with/without long connections. 3- The 3U-Net model with Compound connections between the three U-Nets achieved the best accuracy without long connections.

4- All the models that contains Compound model either at the beginning or at the end showed better accuracy when using long connections except Compound_Compound. 5- All models contain the same structure e.g. Compound_Compound recorded lower accuracy when using long connection. 6- Adding long connection to the 3U-Net models did not show specific pattern on the performance in terms of increasing or decreasing.

The findings illustrate the correlation between the model accuracy and the existing of the Compound connections where 90% of the models that contain compound connections got higher accuracy over the other models. A similar correlation exists when using 3U-Net models with/without long connections where it shows higher accuracy than the Original and Modified 2U-Net models.

The results in line with the hypotheses of adding the compound connections to the models based on 2U-Nets and 3U-Nets will minimize the loss that might happen during the convolutional and pooling layers. The compound model contains 3 bridging connections between the 2U-Nets while the original and modified models contain only two bridges. In a

similar way, the 3U-Net model with compound connections will contain 6 bridges while the models using original or modified connections will contain only 4 bridges.

The unexpected results appeared while adding the long connections with the 3U-Net model where not all the models recorded better performance by adding the long connection. Adding a long connection recorded the best performance with the models started with compound connection between the 1st and 2nd U-Nets. The reason of that is because the global features maps that generated from the high levels of the first U-Net have less effect when added once to the last expansion path of the 3rd U-Net with Original or modified connections between the first two U-Nets, while with the compound connections these feature-maps added three times in accumulative style and two convolutional processes cycles applied on it during the second and third U-Net plus the final concatenation to the last expansion path through the long connection.

The study could not exceed 3 U-Net especially with image size 256*256 or more because of the significant number of parameters and the limitations of the computational resources.

4.1.3.6 Recommendations for bridge and skip connections

Based on the analysis of the results, the following recommendation would be useful for designing the U-Net based model for liver segmentation.

- When using only 2Bridged U-Net, avoid using the original connections. The compound connections model is the first recommended model and the modified connections come in the second place.
- The 2U-Net bridge model with compound connections is the best when using the recommended structure of 5 levels start with 64 filters and end with 1024 filters at the deepest level for image size 256*256.

- 3U-Net models with/without long connections are recommended over 2U-Net with original or modified connections.
- When using 3U-Net based models, it is recommended to use compound connections between 1st and 2nd U-Nets or modified connections will be the second recommended.
- When using 3U-Net based models, it is recommended to use compound or modified connections between 2nd and 3rd U-Nets, but better to use original connections between 2nd and 3rd U-Net in case long connection between 1st and 3rd U-Nets will be used.
- In case of using 3U-Net with long connections, the Compound_Original_Long model is the most recommended.
- In case of using 3U-Net without long connections, the Compound_Compound model is the most recommended then Original_Modified model in the second place.
- Using long connection between 1st and 3rd U-Net is not recommended when using 3U-Net model while the connection between the 1st and 2nd U-Net is the same as the 2nd and 3rd U-Net e.g. Original_Original
- With 3U-Net models using Compound connection at the beginning (Between 1st and 2nd U-Net) or at the end (between 2nd and 3rd U-Net), it is recommended to use long bridge connection between 2st and 3rd U-Net.

According to the previous recommendations and the results, the most recommended model is (Compound_Original_Long) then 2U-Net with compound connections, and Compound_Compound and Original_modified ranked at the 3rd and 4th place.

4.2 Loss functions to solve the flipping issues

Based on all the previous recommendations, the recommended model that will be used in this section is 2U-Net with compound connections. The model consists of 2U-Net stacked with 32 filters applied at the first level and 512 filters applied at the deepest 5th level (32-512) using 256*256 image size. The model used the compound bridge connection to connect the 2 U-Nets. The detailed structure of the model explained in a previous section. (3.4.2 3.4.2 Bridge-Net)

This section will explain the results of the experiments conducted to solve the new examined flipping issue. Using augmented data based on rotation and flipping techniques increased the number of available samples that needed to train the models, while the results of testing showed that, the liver detected in both sides of the image at the same time for significant number of testing samples. Two groups of new loss functions introduced to solve the issue. The first group implemented a new Centroid distance between the plops and the second function and integrated it with Dice Similarity coefficient function. The second group based on Deep-supervision approach with integrated DSC and Centroid functions.

4.2.1 Regression Loss functions

In this section, two loss functions (DSC, Centroid) are implemented and integrated in order to solve the flipping issue.

4.2.1.1 Dice Similarity Coefficient (DSC)

The main loss function in the model is Dice Similarity Coefficient (DSC) that measure the percentage of difference between the predicted mask and the ground truth in terms of the absolute values of the intersection between the prediction and ground truth over the summation.

$$DSC = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

Model	Filters	Training %	Validation%	Test	Aug - Test
Original	32-512	97.55%	70.65%	73.21%	60.31%
Modified	32-512	**97.85%**	67.02%	79.12%	60.10%
Compound	32-512	97.52%	**90.11%**	**89.88%**	**94.42%**
Original	64-1024	97.38%	87.75%	79.89%	75.34%
Modified	64-1024	98.12%	**92.50%**	81.37%	**83.68%**
Compound	64-1024	**98.12%**	91.13%	**83.03%**	78.36%

Table 4-38 Training, Validation, Testing (Test_Avg) and testing using augmented data (Test_Avg_Aug) accuracies for all 2U-Net models with Dice Similarity coefficient (DSC) loss function

The flipping issue appeared with all models U-Net, 2U-Net with Original and Modified connections. Although the Compound model enhanced the overall testing accuracy for the model from 60.31% to 94.42% using DSC loss function and the issue is partially decreased, the issue still exists and need extra enhancement Table 4-38 Figure 4-76.

The compound W-Net model recorded significant enhancement toward solving the flipping issue because the additional bridging connections from each level of the first U-Net to the second U-Net concatenated the out features maps from the contraction path of the first U-Net with inputs of the expansion path of the second U-Net which leads to minimize the features that might be lost during the convolutions and pooling operations on the contraction paths. Figure 4-76.

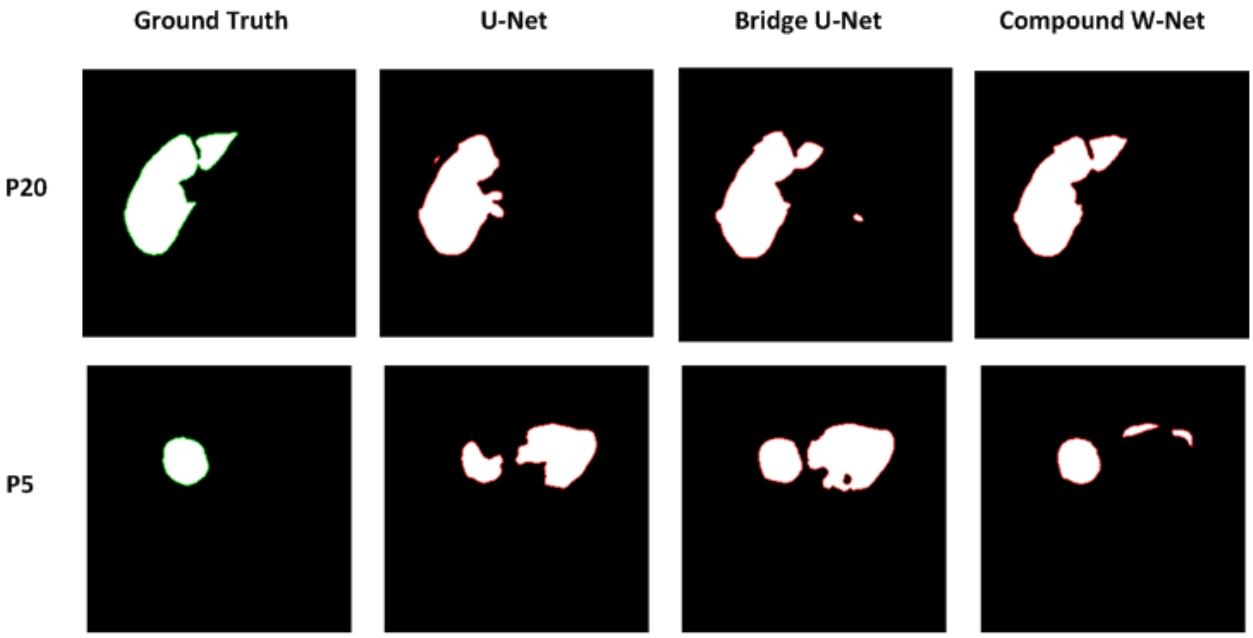


Figure 4-76 The results for U-Net, Bridge U-Net and 2U-Nets with Compound connection. Compound model have the best performance for (P20) and enhanced the flipping issue for P5

4.2.1.2 Weighted sum of DSC and Centroid functions

The Centroid distance loss function calculates the distance between the center of mass of each plop and working to minimize the distance. The new function calculates the total loss as the sum of both DSC and Centroid distance with different weights for each function. The experiment repeated 9 times to find the best weights combination for each function.

All the weights where DSC is less than Centroid recorded almost the same low accuracy for all factors including 14.8%, 18.23% for test_Avg and Test_Avg_Aug respectively.

Patient 20 (P20) recorded higher accuracy than Patient 5 (P5). In general the constant weights ratio of (3: 1) for (DSC: Centroid) recorded the highest accuracy with 88.20%, 87.58%, 90.23%, 87.89% for (P5, P20, P20_Aug, Average (Test)) respectively. While (2:1) is better for validation, p5_Aug, Average (Aug_Test) Table 4-39. The Dice: Centroid weights with (3:1) recorded the best accuracy for 4 of 6 factors and followed by second and third best accuracy for (5:1) and (2:1) in sequence.

Although using Dice:Centroid percentage with (3:1) enhanced the output mask for the flipping issue with number of slices e.g. (100, 104, 106) and solved the issue for significant number of images e.g.(121, 125, 128), the issue still exists. While 5:1 for the Dice:Centroid ratio enhanced the issue more than 3:1 for number of samples e.g. (100, 104, 106), the ratio 3:1 recorded much better performance for other samples e.g. (121, 125, 128). Using weighted loss function for DSC and Centroid distance enhanced the output to solve the issue and the output mask is significantly improved. Figure 4-77

The results indicated that, the DSC function should have higher weight over the centroid function because it works to increase the intersection between the prediction and the ground truth for all samples while the centroid function mainly activated when the predicted output contains more than on plop. If the predicted output contains only one plop, the distance between the center points will evaluated to zero as x1 = x2 and y1 = y2 and the total loss will equal to the DSC only.

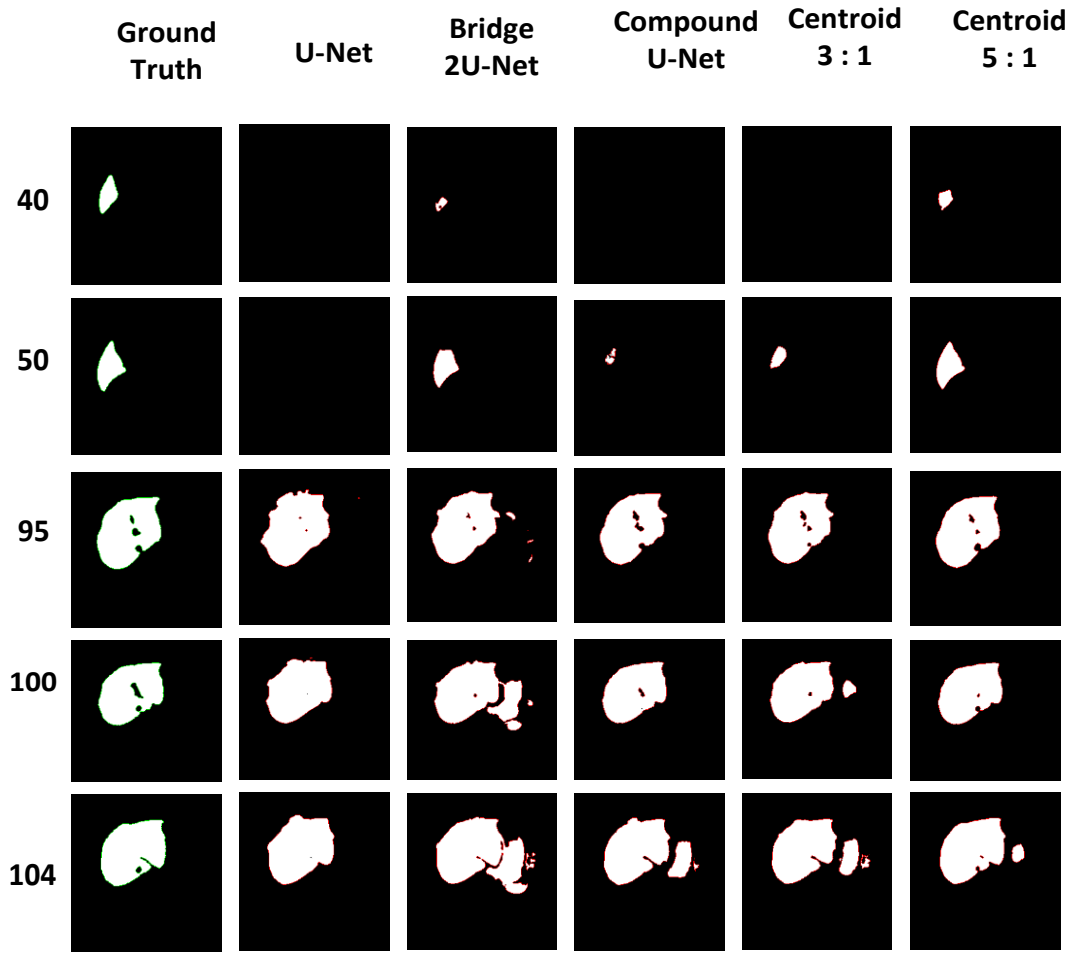
$$\boldsymbol{Dist} = \left(\sqrt{(\boldsymbol{x_1} - \boldsymbol{x_2})^2 + (\boldsymbol{y_1} - \boldsymbol{y_2})^2}\right) = \boldsymbol{Zero}$$

$$L = DSC * \beta 1 + Dist * \beta 2 = DSC * \beta 1$$

Where $\boldsymbol{(x1, y1)}$ and $\boldsymbol{(x2, y2)}$ are the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates of the centroid points of blob 1 and blob 2 respectively. L is the total loss, DSC is Dice Similarity Coefficient; $Dist$ is the distance between centroid points of the blobs, $\beta 1$ and $\beta 2$ are constant weights for DSC and $Dist$ respectively.

Weights (DSC-Centroid)	P5	P20	P5_Aug	P20_Aug	Training	Validation	Test_Avg	Test _Avg _Aug
1--2	1.41%	28.19%	28.19%	14.56%	0.01%	0.01%	14.80%	21.38%
1--3	1.41%	28.19%	21.90%	14.56%	0.00%	0.00%	14.80%	18.23%
1--4	1.41%	28.19%	21.90%	14.56%	0.00%	0.00%	14.80%	18.23%
1--5	1.41%	28.19%	21.90%	14.56%	0.00%	0.00%	14.80%	18.23%
1--1	83.10%	93.39%	54.37%	85.99%	97.21%	74.14%	88.25%	70.18%
2--1	83.56%	84.02%	**87.32%**	71.42%	**97.54%**	80.38%	83.79%	**79.37%**
3--1	**88.20%**	**87.58%**	66.89%	**90.23%**	**97.54%**	70.99%	**87.89%**	78.56%
4--1	74.52%	70.26%	84.47%	66.98%	97.52%	76.18%	72.39%	75.72%
5--1	82.15%	86.09%	59.50%	74.05%	97.53%	**85.24%**	84.12%	66.77%

Table 4-39 Different weights for DSC and Centroid distance function showed the best ratio is (DSC:Centroid) is (3:1), then 2:1 then 5:1



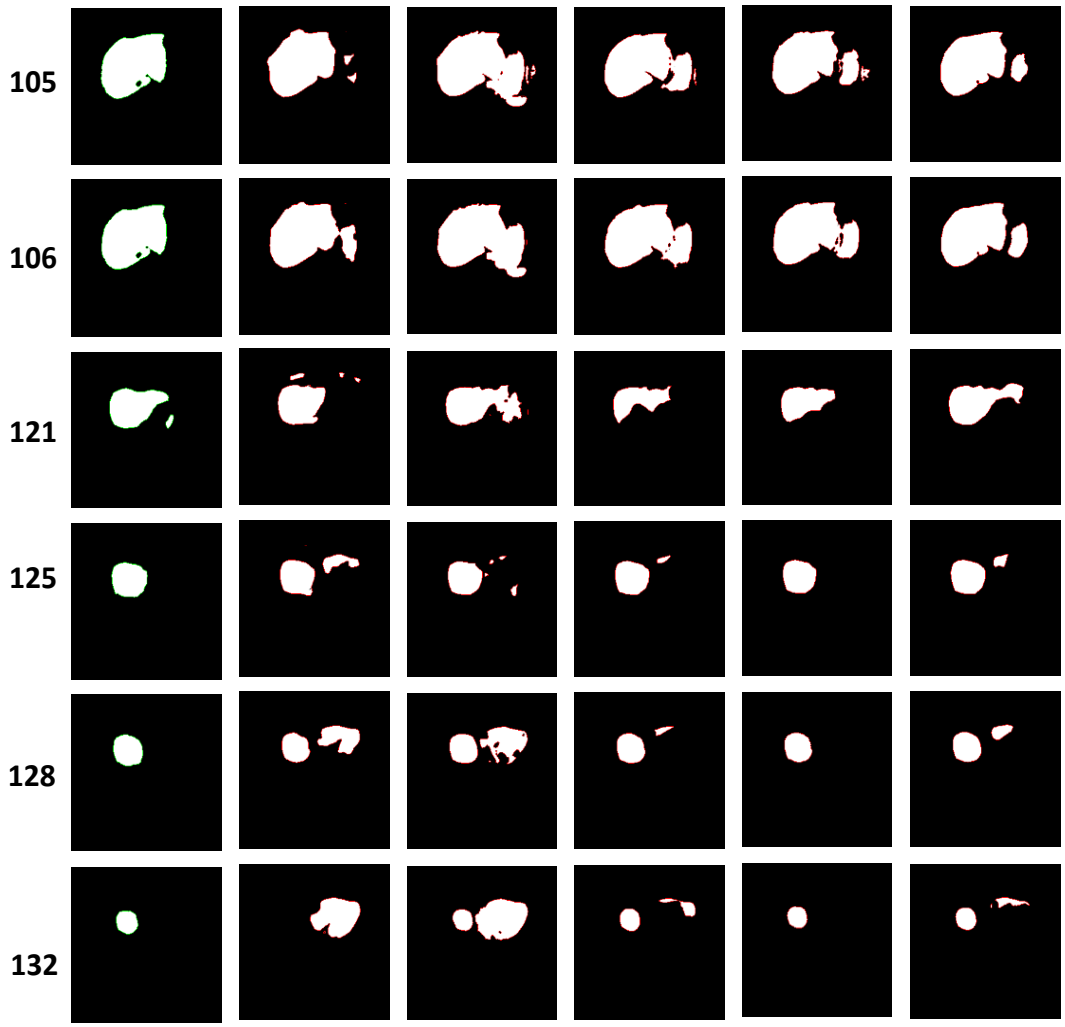


Figure 4-77 Using (DSC : Centroid) weights with 5:1recorded accuracy better than 3:1 for the first half of samples while for the second half the accuracy decreased even worse than using DSC loss function

4.2.2 Deep- supervision with different loss functions

Deep- supervision approach is based on applying loss function on deeper levels of the model not only at the last level to add more constraints to the learning process. Two different types of deep-supervision introduced to enhance the model performance with the flipping issue. The first approach used the ground truth masks with its original resolution 256*256 with DSC loss function while the second approach used different resolution masks according to the supervision level.

4.2.2.1 Deep supervision with De-Convolution using DSC loss function

In this approach, a single scale mask with the original resolution 256*256 will be used within the DSC loss function. Because the size of the feature maps is increased by 50% from the deeper level to the higher level until reach the original size at the topmost level, a number of De-Convolutional processes applied to the output feature maps from each level to increase the size to the original mask size. The number of De-Convolutional process applied at each level depends on the size of the feature maps at this level e.g. at level1 (The deepest level of U-Net with 32-512 filter and image size 256*256) feature maps size after applying the original concatenation and up-sampling of the U-Net will be 32*32, then three De-Convolution processes will be applied to increase the feature maps size to reach the original image size (32*32 → 64*64) , (64*64 → 128*128), (128*128 → 256*256) . The number of De-Convolutional process needed to resize the feature maps to the original size will be decreased with the higher levels of U-Net. The loss function will be calculated at each level of the 4 levels of the expansion path of second U-Net. The total loss is calculated as the weighted sum of all 4 loss functions.

$$\text{Loss} = \sum_{i=1}^{n} L_i * \omega_i$$

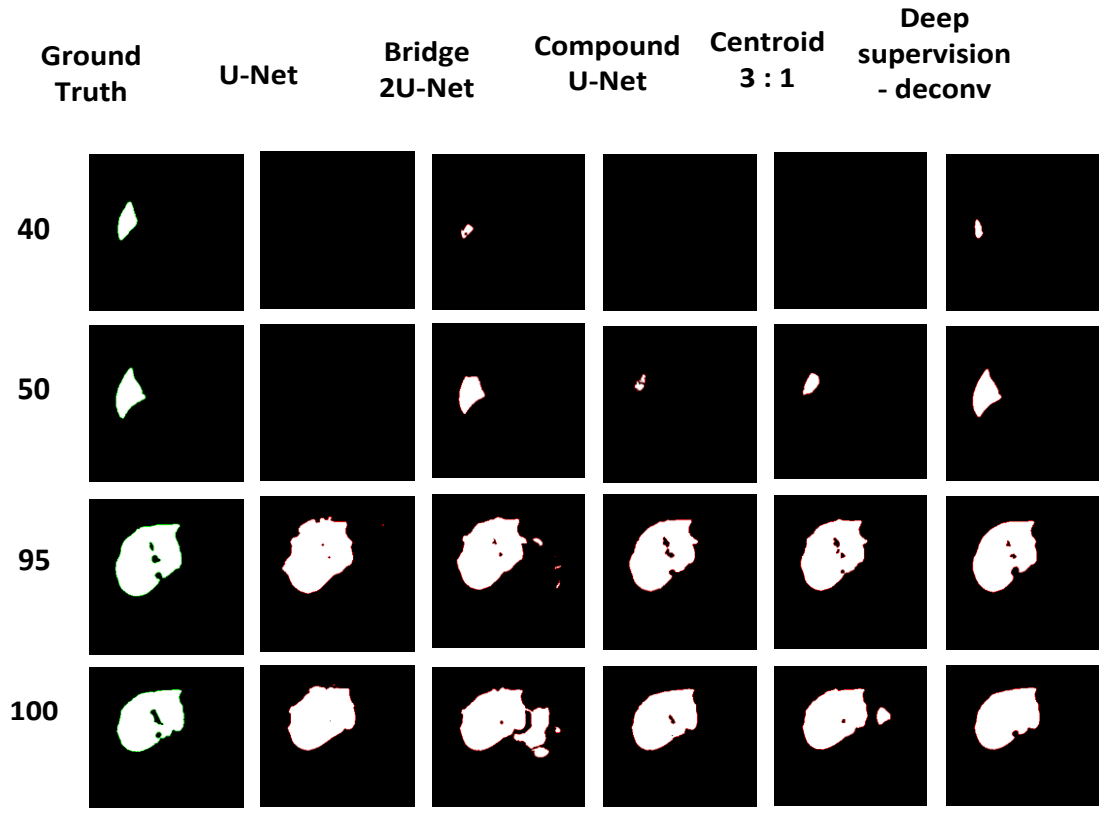
Where n is the number of levels in the model, w is the constant weight for each loss function, L is the loss at level i.

The best accuracy recorded for the weights combination (10:2:2:10) for all 4 outputs of the U-Net levels where the output organized from deeper to final level (L1, L2, L3, L4) L4 represent the final output at the topmost level of the 2nd U-Net. Figure 4-79 Test_Avg is the average for all samples in P5 and P20 where patient number 5 (P5) contains most of the images with the flipping issue. Table 4-40

The deep-suppression approach with De-convolutional process showed accuracy higher than the Compound model with DSC and higher than the model using DSC and Centroid distance loss functions (3:1) because it added loss functions (DSC) on each deeper level of the model to the total loss. The flipping issue minimized for most of the images although the issue appeared in some images that solved using Centroid function e.g. (128, 130) because the deep-supervision implemented only DSC as a loss function. Figure 4-78 we assume that, integrating Centroid loss with DSC loss within the deep-supervision will enhance the model accuracy and minimize the flipping.

Weights of 4 outputs	P5	P20	Training	Validation	Test_Avg
OUT_1-1-1-1	73.69%	93.45%	97.29%	87.81%	83.57%
OUT_1-2-3-10	78.15%	**96.89%**	96.79%	**90.32%**	87.52%
OUT_10-2-2-10	**80.37%**	95.99%	**97.34%**	89.33%	**88.18%**

Table 4-40 Deep-Supervision approach using De-Convolutional layers with DSC loss function showed that the best performance recorded for the weight combination (10:2:2:10 for the four outputs of the model.



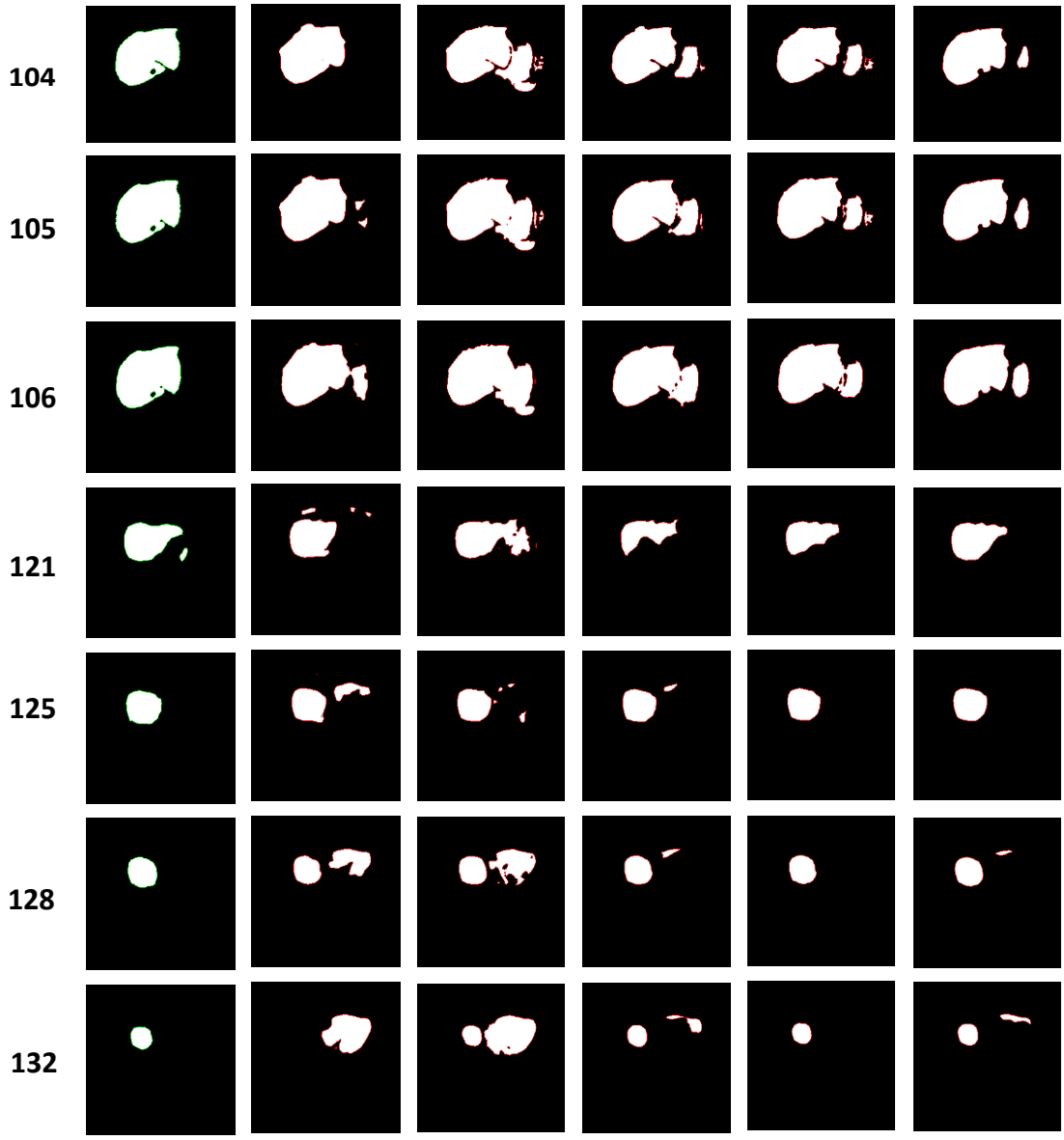


Figure 4-78 Deep-Supervision using De-Convolutional layers with DSC loss function achieved accuracy better than the Compound model, and better than the centroid function with 3:1 ratio for the first half of samples for (P5) but the Centroid function is better for the second half of the samples.

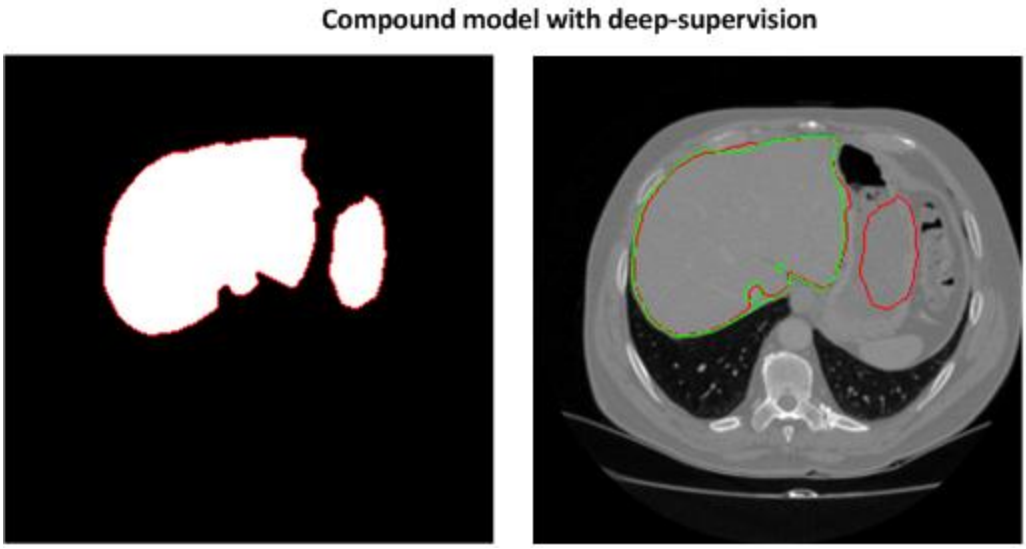


Figure 4-79 sample output for Compound model with Deep-Supervision approach using De-Convolutional layers and loss function is sum of weighted loss of the four outputs using DSC function. Predicted mask at left, mapped ground truth (green) and predicted mask (Red) at right

4.2.2.2 Deep supervision with multi-resolution mask using DSC loss function

The second approach of Deep-Supervision used multiple resolution versions of the same mask and feed it to the loss functions at different levels with the respective mask version the De-Convolutional process is NOT needed anymore. In the experiment with 256*256 image size and 2U-Net with filters applied 32-512, the four levels of 2-Unet (L1, L2, L3, L4) will use a different mask resolution (32*32, 64*64, 128*128, 256*256) respectively. The maximum accuracy for Testing is recorded with the weights combination (10:2:2:10) for loss function (L1, L2, L3, L4) respectively while the total loss is the sum of weighed loss.

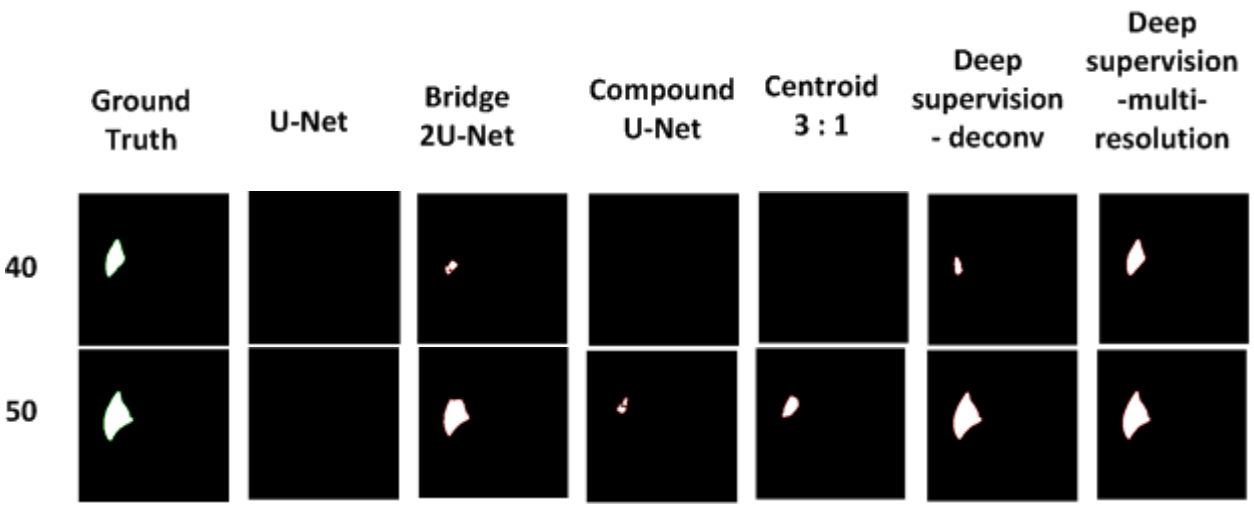
Similar to the results of Deep-Supervision with De-Convolutional processes, The Deep-Supervision using Multi-Resolution masks recorded the best accuracy for weights combinations (10:2:2:10) for the four outputs Table 4-41. The model with weights ration (10:2:2:10) recorded the highest accuracy for Training, Testing for P5 and P20 (90.04% for

testing) while the best validation recorded with model with output weights (1:1:3:5) Figure 4-81.

Deep-supervision with Multi-resolution masks model showed significant improvement on the model accuracy over all the previous approaches (Compound with DSC, compound with DSC and Centroid, and compound with Deep-Supervision using De-convolutions using DSC) although the flipping issue is not solved completely. Using multi-resolution masks instead of applying multiple de-convolution processes to increase the size of the feature maps to be equal to the original mask enhanced the accuracy and decreased the flipping as showed in images (40, 104, 105, 128) in Figure 4-80. The results indicated that, there might be some lost features due to applying multiple de-convolutional processes on the feature-maps at each level when using deep-supervision with de-convolution approach.

Weights of 4 outputs	P5	P20	Training %	Validation%	Test_Avg
OUT_1-1-1-1	78.06%	94.07%	97.47%	63.19%	86.07%
OUT_1-2-3-10	76.99%	91.62%	97.27%	77.12%	84.31%
OUT_10-2-2-10	**83.82%**	**96.27%**	**97.40%**	82.93%	**90.04%**
OUT_7_1_1_1	82.56%	95.06%	97.11%	87.53%	88.81%
OUT_1_1_1_7	80.31%	95.08%	97.32%	86.60%	87.69%
OUT_1_1_3_5	81.38%	95.12%	97.36%	**89.12%**	88.25%
OUT_4_1_1_4	76.75%	94.73%	97.22%	86.30%	85.74%

Table 4-41 Deep-Supervision using Multi-Resolution masks showed the best accuracy recorded for the output weights (10:2:2:10)



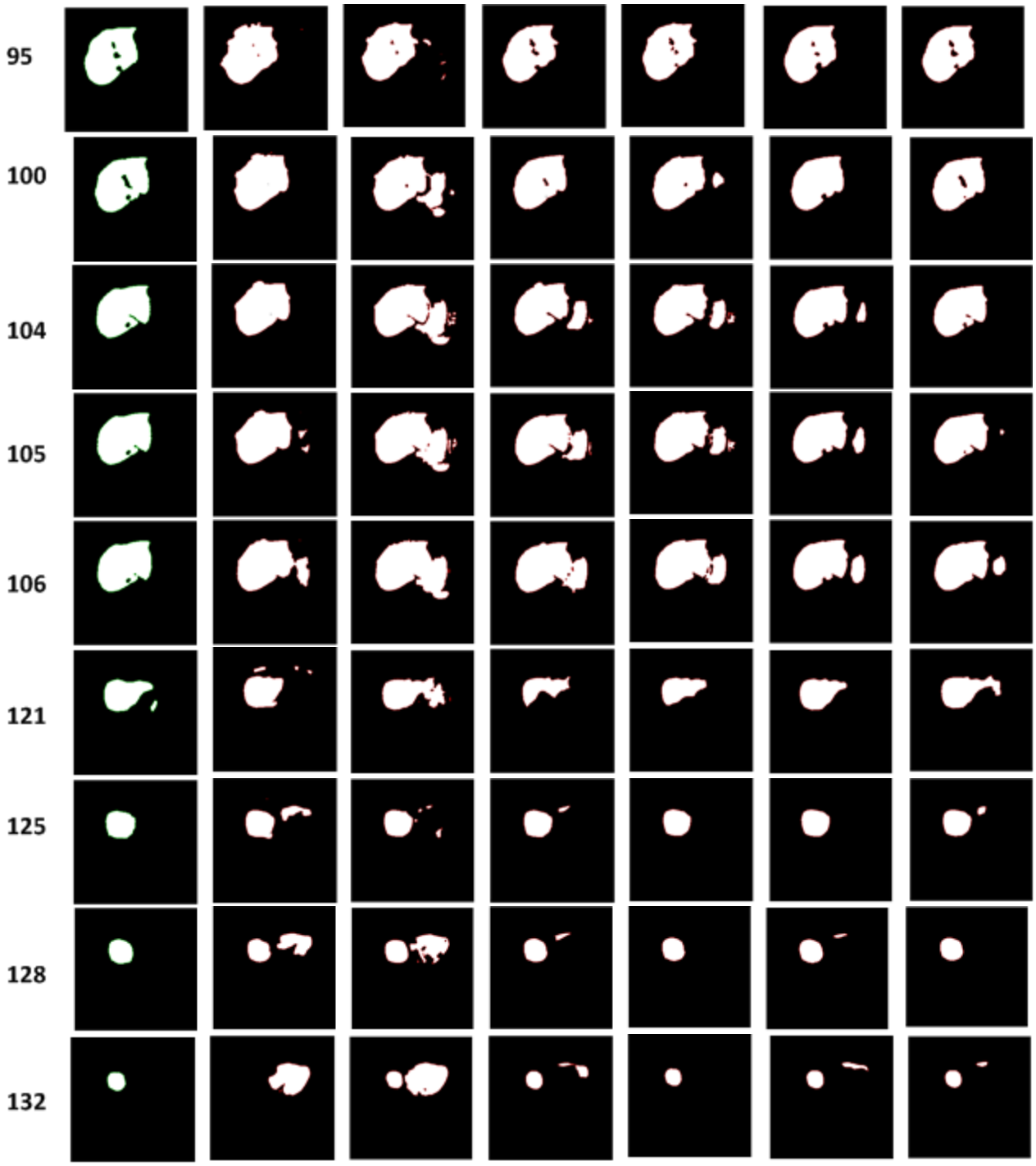


Figure 4-80 Deep-Supervision using Multi-Resolution masks with output weights (10:2:2:10) using the loss function DSC recorded the best output over Deep-Supervision using De-Convolutional layers, and better than compound model with Centroid function with weight (3:1) in the first half while the accuracy become lower during the second half of the samples.

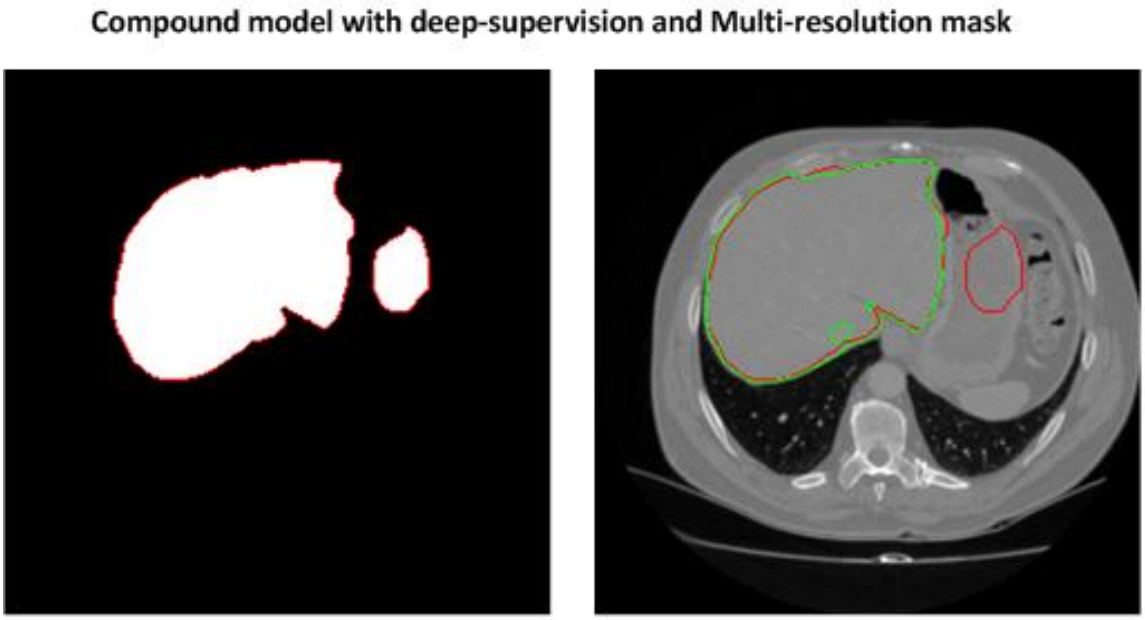


Figure 4-81 Example for Deep-Supervision using Multi-Resolution masks. Predicted mask (Left), mapping the predicted mask (red) and Ground truth (Green) on the original image

4.2.2.3 Deep supervision with De-Convolution using DSC and Centroid loss functions

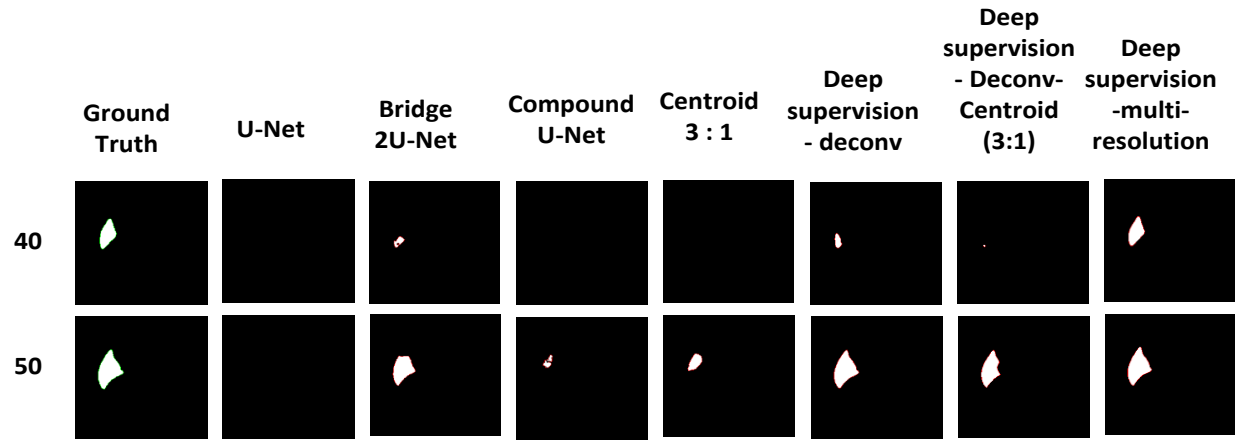
This approach used a hybrid approach using Deep-Supervision with De-Convolutional approach and the loss function is the weighted loss of DSC and Centroid with weight ratio (3:1). Using the weighted loss function of DSC and Centroid with weighted ratio (3:1) achieved the best accuracy over the other ratios of (2:1 and 5:1) Table 4-42. The model used the weights ratio between the 4 outputs of the model that recorded the best accuracy (10:2:2:10) with values (0.4166, 0.0833, 0.0833, 0.4166).

The model with DSC:Centroid Ratio equal (3:1) recorded accuracy **89.14%** that is higher than the model used only Centroid function and higher than the model used Deep-supervision with DSC where the accuracies were (**88.18%** and **87.89%)** respectively while the model with ratio (2:1) ranked in the 2nd place with accuracy (**88.34%)** Table 4-42**.** Although the model enhanced the output mask for significant number of images over e.g. (40, 50, 95, 100, 104, 105, 106, 132) while the out mask was worse than the other models for small number of images e.g. (121,125,128).Figure 4-82

The result of applying deep-supervision using de-convolution and weighted loss function of DSC and centroid proved that combining the indicators from section 4.2.1.2 and section 4.2.2.1 will increase the accuracy and enhance the solution of the flipping issue, where the first indicator from section 4.2.1.2 stated that, combining DSC and Centroid loss functions in one weighted loss function with weighted ratio (3:1) for DSC and Centroid enhanced the model performance rather than using only one loss function while section 4.2.2.1 illustrated that, integrating deep-supervision approach with the model increased the model accuracy more than the compound model. The best DSC:Centroid functions ration is 3:1 the 5:1 based on the empirical experiments, the same with the ratio between the four outputs of the deep-supervision within the total loss where 10:2:2:10 is the best empirical results which gives more weight to the loss at the deepest and highest level in the model.

Weights of 4 outputs	P5	P20	Training %	Validation%	Test
OUT-10-2-2-10	83.71%	83.71%	97.35%	78.35%	83.71%
(3—1)	78.38%	96.51%	97.24%	92.06%	87.44%
	78.72%	94.40%	97.31%	91.79%	86.56%
	81.40%	**96.88%**	97.25%	**91.66%**	**89.14%**
Average (3--1)	**80.55%**	**92.87%**	**97.30%**	**87.40%**	**86.71%**
5—1	**79.21%**	**91.58%**	**97.29%**	**81.68%**	**85.40%**
2—1	81.12%	95.55%	97.32%	91.10%	88.34%

Table 4-42 Deep-Supervision with De-Convolutional layers using weighted outputs (10:2:2:10) using weighted loss function for (DSC:Centroid) equal (3:1) , Highest accuracy (Green) lowest accuracy (Red) within each group



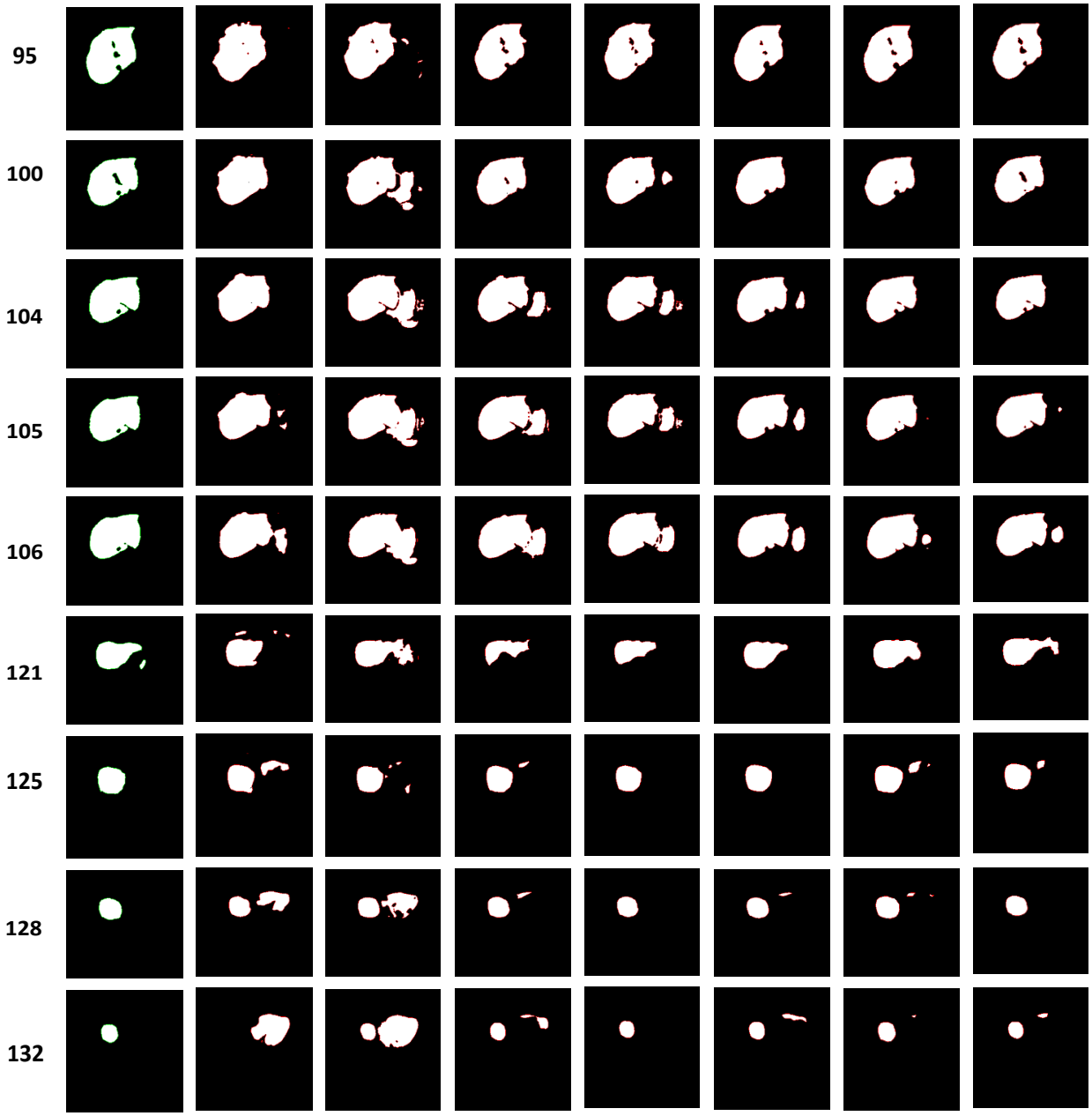


Figure 4-82 Deep-Supervision with De-Convolutional layers using weighted outputs (10:2:2:10) using weighted loss function for (DSC: Centroid) equal (3:1). The model recorded the best accuracy for the first half samples over all the previous loss functions while for the second half it recorded the worst accuracy over all the previous proposed loss functions

4.2.2.4 Deep supervision with multi-resolution mask using DSC and Centroid loss functions

This approach used a hybrid approach based on Deep-Supervision with Multi-Resolution masks approach with weighted loss function of DSC and Centroid with weighted

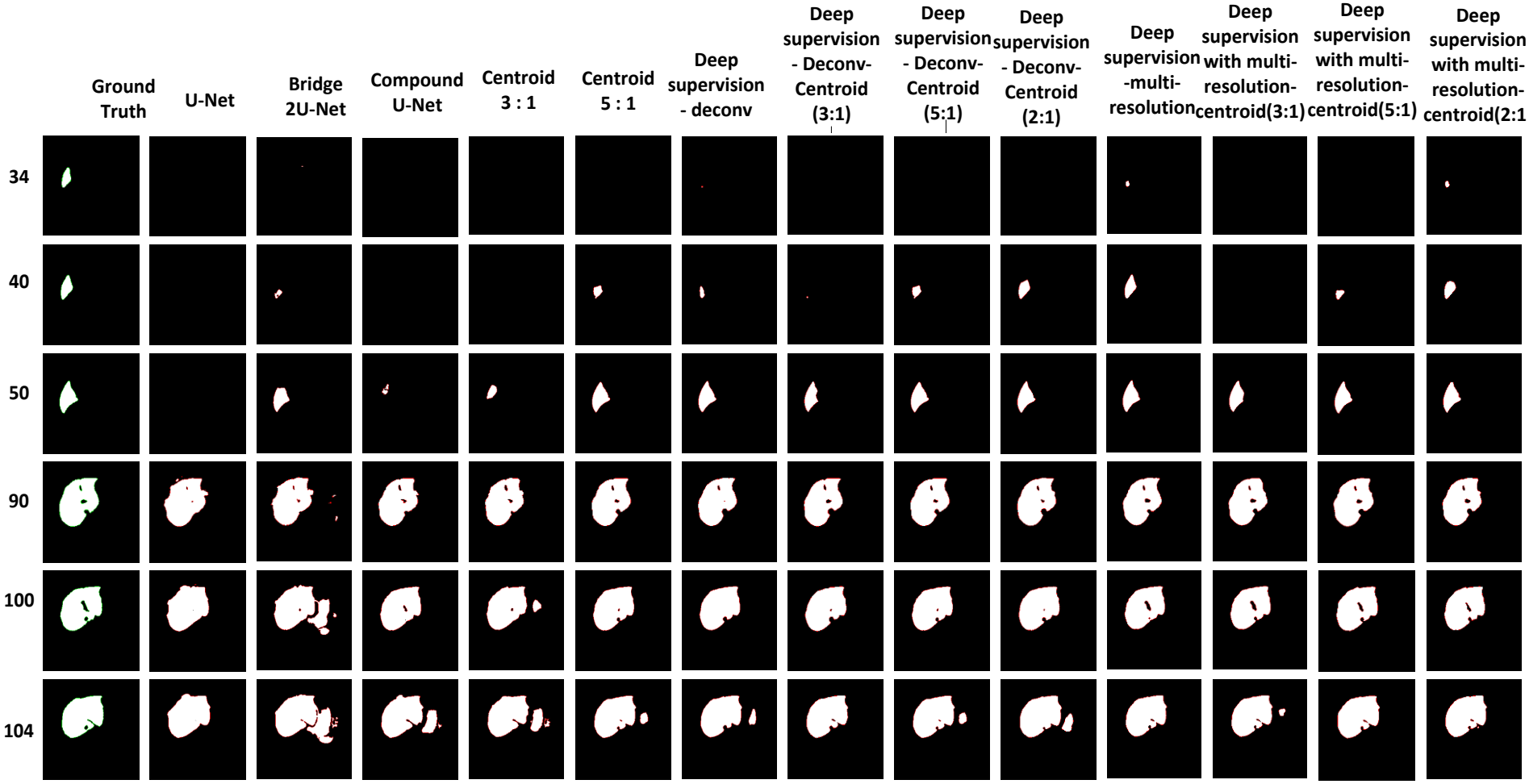
ratio in (2:1, 3:1, 5:1). The model used the weights ratio between the four outputs of the model that recorded the best accuracy (10:2:2:10) with values (0.4166, 0.0833, 0.0833, 0.4166).

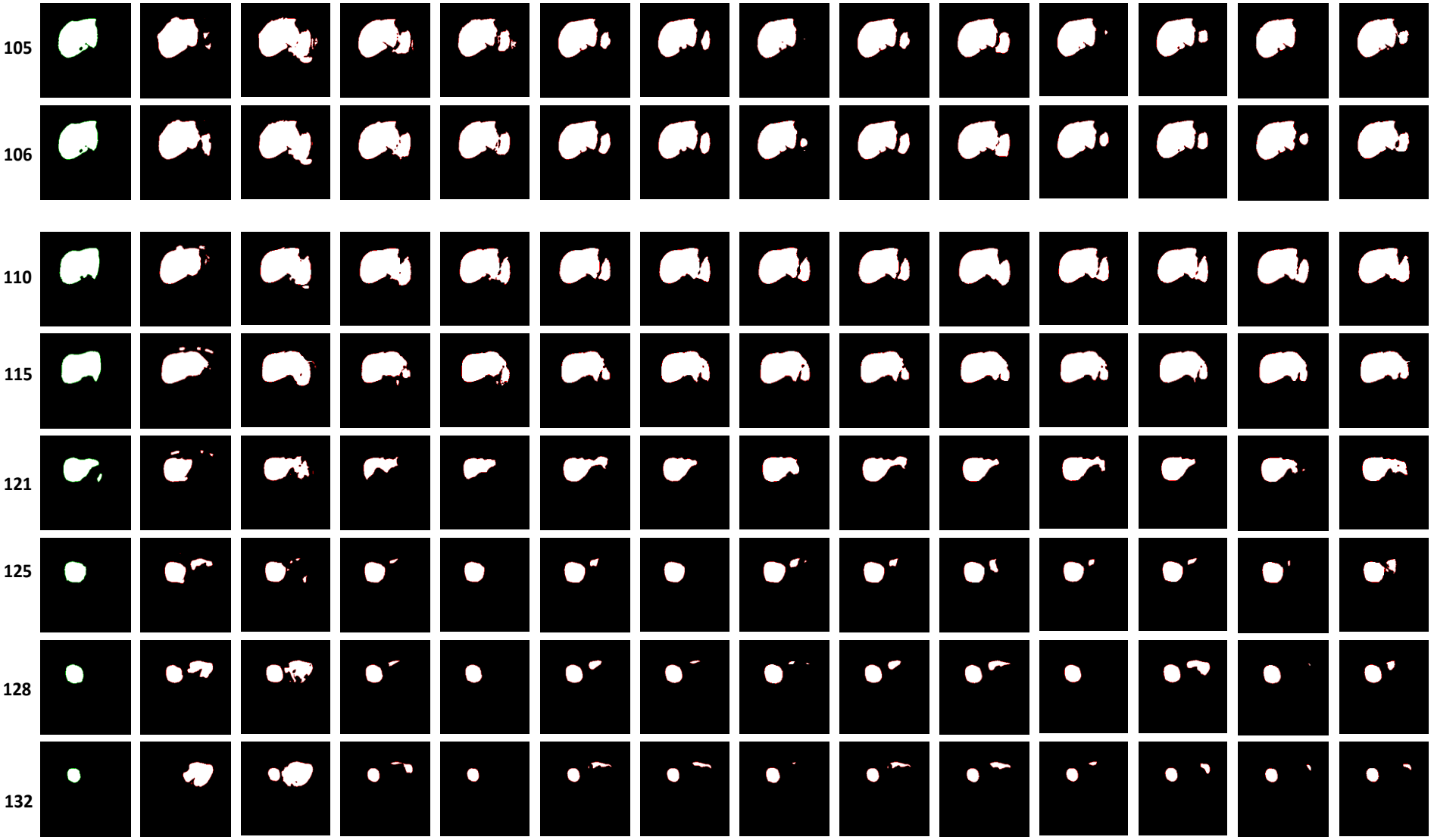
The model with DSC:Centroid ratio equal (3:1) recorded accuracy **88.05%** that is lower than the compound model used Centroid function and lower than the model used Deep-supervision with multi-Resolution and DSC loss function where the accuracies were (**88.18%** and **90.04%)** respectively but higher than using for DSC:Centroid ratio (2:1 or 5:1) Table 4-43.

When the model used only DSC as the loss function, it recorded output better than the models (Compound, and compound with Centroid function) for most of the samples except for samples (121, 125, and 128) where the centroid model was better. Replacing the DSC loss function for the Deep-Supervision with Multi-Resolution masks by the weighted loss function of DSC and Centroid with weights ratio (3:1 and 2:1) decreased the accuracy of the model while using DSC:Centroid ration of (5:1) recorded almost the same accuracy as the same model with only DSC. The results indicated that the DSC:centroid ratio should be higher equal or higher than 5:1 in order to get better accuracy than the model using only DSC which is comply with the same indicator from sections 4.2.1.24.2.2.2and 4.2.2.2. The model shows output masks almost the same the models used Deep-Supervision with De-convolutional (using DSC, using DCS + Centroid). Figure 4-83

Weights (DSC-Centroid)	P5	P20	Training %	Validation%	Test
OUT-10-2-2-10	78.17%	95.02%	97.29%	88.78%	86.60%
(3—1)	81.02%	95.07%	97.34%	70.93%	88.05%
	75.16%	96.10%	97.29%	89.97%	85.63%
Average(3—1)	**78.11%**	**95.40%**	**97.31%**	**83.23%**	**86.76%**
5--1	80.99%	95.10%	97.34%	75.65%	88.04%
2--1	81.96%	93.51%	97.35%	62.73%	87.73%

Table 4-43 Deep-Supervision using Multi-Resolution masks with weighted outputs (10:2:2:10) and loss function is weighted loss for (DSC:Centroid) functions with weights (3:1), Highest accuracy (Green)


Ground Truth
U-Net
Bridge 2U-Net
Compound U-Net
Centroid 3 : 1
Centroid 5 : 1
Deep supervision - deconv
Deep supervision - Deconv-Centroid (3:1)
Deep supervision - Deconv-Centroid (5:1)
Deep supervision - Deconv-Centroid (2:1)
Deep supervision -multi-resolution
Deep supervision with multi-resolution-centroid(3:1)
Deep supervision with multi-resolution-centroid(5:1)
Deep supervision with multi-resolution-centroid(2:1
34
40
50
90
100
104


105
106
110
115
121
125
128
132

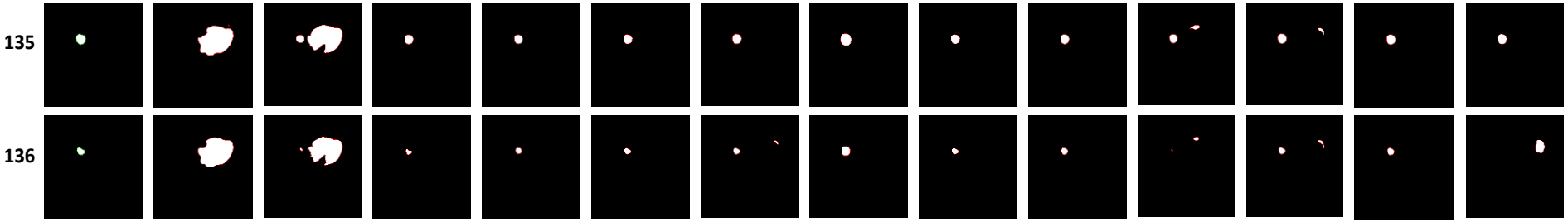


Figure 4-83 All models using Deep-Supervision with de-convolution and Multi-Resolution masks with weighted outputs (10:2:2:10) and loss functions are DSC and weighted loss for (DSC:Centroid) functions with weights (3:1), (2:1) and (5:1)

4.2.3 Results Summary

Using **DSC and Centroid** function with weights (3:1) solved the flipping issue for 50% of the samples that contains the issue and enhanced the remaining 50% of the samples ~ 20% than the compound model.

Using **Deep-Supervision with de-convolution** approach using DSC as a loss function enhanced the output of the first half of the samples than using Centroid but the last half of the samples that solved by Centroid started to get 10% of flipping issue to appear again. While replacing the DSC function with the combined DSC and Centroid loss functions with 3:1 weight to the deep-supervision model enhance the overall output masks over Centroid alone and DSC with deep-Supervision using de-convolution, the flipping issue still exists with a very small size in a small number of samples. The weighted loss for (DSC:Centroid) equal to (3:1) recorded the best accuracy over the other two ratios (2:1 and 5:1).Table 4-44

The model based on **Deep-Supervision with Multi-Resolution** masks using DSC as a loss function recorded significant improving in the output masks where the size of the duplicate plops got smaller than all the previous models. Although the model started to detect the liver at slice number 34 while the ground truth detected it at slice 19, it is considered as the best performance of all models where it detected the liver starting from slices number 40 and 50. Replacing the DSC loss function with the weighted DSC and Centroid loss function with weights (3:1 or 2:1 or 5:1) decreased the overall accuracy of the model. Using weighted DSC and Centroid loss function showed the flipping issue again for the last half of the slices with larger size than the models (Bridge with Centroid, Deep-supervision with De-convolution and DSC, Deep-supervision with De-convolution and Centroid, Deep-supervision with Multi-Resolution and DSC). For the first half of slices, Deep-Supervision with Multi-Resolution masks using DSC and Centroid (3:1) loss function almost

has similar accuracy as Deep-Supervision with De-Convolutional model but lower than the models (Deep-Supervision with Multi-Resolution masks using DSC, and , Deep-Supervision with De-Convolution using DSC and Centroid) in sequence. Table 4-44

Model	Outputs ratio	DSC:Centroid ratio	P5	P20	Training	Validation	Test_Avg
Compound	N/A	N/A	85.98%	93.79%	**97.52%**	90.11%	89.88%
Centroid	N/A	**3:1**	**88.20%**	87.58%	97.45%	70.99%	87.89%
Deep-Supervision with De-Convolution	10:2:2:10	N/A	80.37%	95.99%	97.34%	89.33%	88.18%
Deep-Supervision with De-Convolution	10:2:2:10	3:1	81.40%	**96.88%**	97.25%	**91.66%**	89.14%
Deep-Supervision with De-Convolution	10:2:2:10	2:1	81.12%	95.55%	97.32%	91.10%	88.34%
Deep-Supervision with De-Convolution	10:2:2:10	5:1	79.21%	91.58%	97.29%	81.68%	85.40%
Deep-Supervision with Muti-Resolution	10:2:2:10	N/A	83.82%	96.27%	97.40%	82.93%	**90.04%**
Deep-Supervision with Muti-Resolution	10:2:2:10	3:1	81.02%	95.07%	97.34%	70.93%	88.05%
Deep-Supervision with Muti-Resolution	10:2:2:10	2:1	81.96%	93.51%	97.35%	62.73%	87.73%
Deep-Supervision with Muti-Resolution	10:2:2:10	5:1	80.99%	95.10%	97.34%	75.65%	88.04%

Table 4-44 The accuracy for all loss functions used to solve the flipping issue with different approachs and multiple ouput weights and DSC:Centroid weights, highest accuracy (Green)

4.2.4 Qualitative analysis for flipping issue's proposed solutions

This section covers the qualitative metric that had been used to rank the proposed solutions for the flipping issue. The approach based on the comparison between the segmentation output mask from each model for the same image. The study compares the output of 13 models over 139 images that belong to the patient number 5 (P5) where most of the flipping issue exists.

Evaluations steps:

1- List the models' names in a tabular style
2- List the output masks from all models for on image together with the ground truth for comparison.
3- Figure 4-84
4- Rank the models based on the existence and size of the flipping issue (1→13)
5- Multiply the rank by fixed weight for each rank. Figure 4-85
6- Calculate the summation of the total weight over all images for each model.
7- Ranking the models based on the total weights. Figure 4-86

Results:

- Deep-Supervision with multi-resolution masks with weighted loss function Dice:Centroid ratio 5:1 share the first rank with Deep-Supervision with de-convolution masks with weighted loss function Dice:Centroid ratio 3:1
- Deep-Supervision with de-convolution masks with DSC loss function achieved the 3rd rank
- Deep-Supervision with de-convolution masks with weighted loss function Dice:Centroid ratio 5:1 came in the 4th place
- Deep-Supervision with multi-resolution masks with DSC loss function achieved the 5th place
- The model with weighted loss function Dice:Centroid ratio 5:1 came in the 6th place

And the full list in Figure 4-86 and A full datasheet and ranking for all models over the images in details in section 6.5

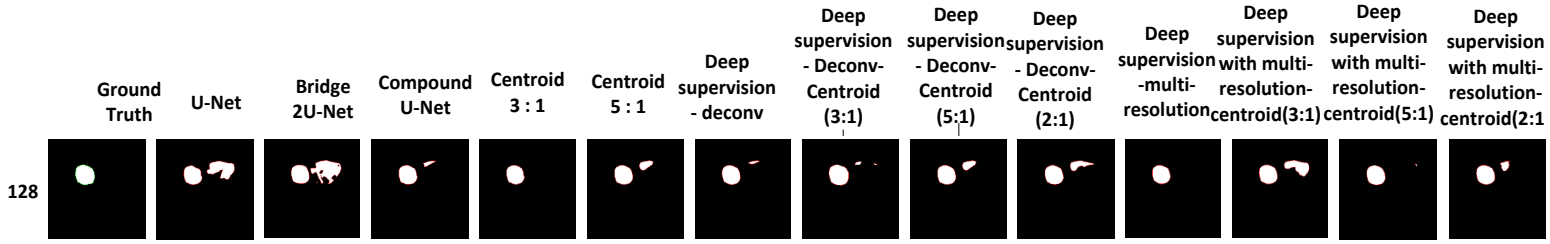


Figure 4-84 Example of models' output comparison

	Ground Truth		U-Net		Bridge 2U-Net		Compound U-Net		Centroid 3 : 1		Centroid 5 : 1		Deep supervision - deconv		Deep supervision - Deconv-Centroid (3:1)		Deep supervision - Deconv-Centroid (5:1)		Deep supervision - Deconv-Centroid (2:1)		Deep supervision -multi-resolution		Deep supervision with multi-resolution-centroid(3:1)		Deep supervision with multi-resolution-centroid(5:1)		Deep supervision with multi-resolution-centroid(2:1)	
	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight	Rank	weight
133	13	1	12	2	11	3	1	13	9	5	4	10	1	13	7	7	10	4	6	8	4	10	3	11	7	7		
134	13	1	12	2	5	9	1	13	4	10	8	6	1	13	6	8	7	7	11	3	10	4	1	13	8	6		
135	13	1	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	11	3	10	4	1	13	1	13		
136	13	1	12	2	1	13	1	13	1	13	8	6	1	13	1	13	1	13	10	4	9	5	1	13	11	3		
137	13	1	12	2	1	13	1	13	1	13	7	7	1	13	1	13	8	6	10	4	9	5	1	13	11	3		
138	13	1	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	10	4	9	5	1	13	11	3		
Total		2229		687		799		691		624		730		667		882		651		699		548		887		35		

Figure 4-85 Calculating the weights and total weights for each model over all the images

1588	808	1579	1552	1613	1641	1662	1625	1512	1622	1583	1662	1514
7	13	9	10	6	3	1	4	12	5	8	1	11

Figure 4-86 calculating the final ranking for the models based on the total weights

4.2.5 Discussion for flipping issue

The results of the proposed approaches to solve the flipping issue highlighted the following key findings. 1- The proposed model of 2 U-Net with compound bridged connections suffers from the flipping issue with low percentage than U-Net or original bridged U-Net. 2- Using the weighted loss function of DSC and Centroid loss function enhanced the models performance, even if the weights ratio is empirical. 3- Integrating the deep-supervision approaches with the Compound connection model significantly helps to minimize the flipping issue. 4- Integrating the weighted loss function with the deep-supervision approaches with the compound model achieved a significant improvement on the model performance to solve the flipping issue. 5- The weights for the loss functions and the weights for the different outputs of the deep-supervision need more investigation to find a standard formula.

The study highlights the correlation between the performance and the weights of the loss functions where the DSC loss function should have equal or higher weight than the Centroid function because DSC is working over all the image pixels to minimize the difference in predictions where the centroid only work when there is more than one predicted blob to minimize the distance between the centers of mass.

The compound model has 5 levels of convolutional processes that lead to have 4 sub-outputs when integrate the deep-supervision approach. The total output loss is the weighted sum of the outputs. The empirical study emphasize that the first and last output should be at least 3 times weight of the middle ones because the last output work over the final up-sampled features which is the final predictions and increase its weight will add more constrained to minimize the loss and the same for the first output at the deepest level where it works on the smallest size of the feature-maps before any propagated loss.

The results in line with the hypotheses mentioned in the methodology section where assumed that, the centroid function as a new novel contribution with the deep-supervision will enhance the model accuracy. The results came in line with the previous studies of implementing deep-supervision with de-convolution where it enhances the models performance[115] [223] [216] [150].

The literature shows some studies implementing deep-supervision with de-convolutional and some other studies used weighted loss functions but or study proposed a novel approach where it integrate a newly introduced Centroid loss function and newly modified deep-supervision approach with multi-resolution masks together with Dice Similarity coefficient (DSC) loss function. The approach integrates the deep-supervision with the weighted loss functions to the novel Compound 2U-Nets model.

The solution achieved high liver segmentation accuracy with minimized effect of flipping due to augmentation techniques while the computation cost is equal to the normal 2 stacked U-Net even with shorter training time compared to the large training dataset.

The study could not formulate the ratio of the weights between the loss function and the multiple outputs of the deep-supervision but only used fixed weights based on the empirical experiments. Formulation of the weights would be an important part of the future study.

4.2.6 Flipping issue recommendations

- To decrease the flipping issue effect, it is highly recommended to use one of the deep- supervision approaches.
- Two recommended model can solve the flipping issue with almost 93%. The first model is 2Bridged U-Net with deep-supervision based on multi-resolution masks

using weighted loss function combined DSC and Centroid loss with ratio (5:1) and the weighted loss for the 4 output of the deep supervision is (10:2:2:10).

- The second recommended model based on deep supervision using de-convolution and weighted loss ratio for DSC and Centroid is (3:1).
- The 3rd recommended model is the same as the first recommended model except that it used DSC alone as the loss function.
- The last recommended model is the same as the first model except that it use DSC:Centroid ratio equal (3:1).
- For the images where the liver size is very small, the 2U-Net with compound bridge connection using withed loss function combined DSC and centroid by ratio (3:1) has the same recommendation as the first with deep-supervision based on multi-resolution masks and weighed loss function (3:1) for DSC and centroid.

4.3 Segmentation results validation by Institut Kanser Negara (IKN)

The research for liver segmentation that included within the thesis is part of a collaboration project between University of Nottingham, Monash University and (IKN). IKN provided data samples of CT scans for 15 patients with total number of 26,948 DICOM image. The images can't be used for model training because it didn't contain any annotated masks for the liver. Our proposed model for liver segmentation trained and tested using the public dataset (3Dircadb1). In order to verify our model accuracy, the trained model tested using IKN dataset which is completely different dataset in terms of the source scanning machine and setting with different image intensity range and resolution and some images are scanned during the treatment procedure which make it not suitable for testing because the metal needles inside the image caused a lot of noise, distortion, very light spots (high intensity needle) and very dark region around the needle. Figure 4-87

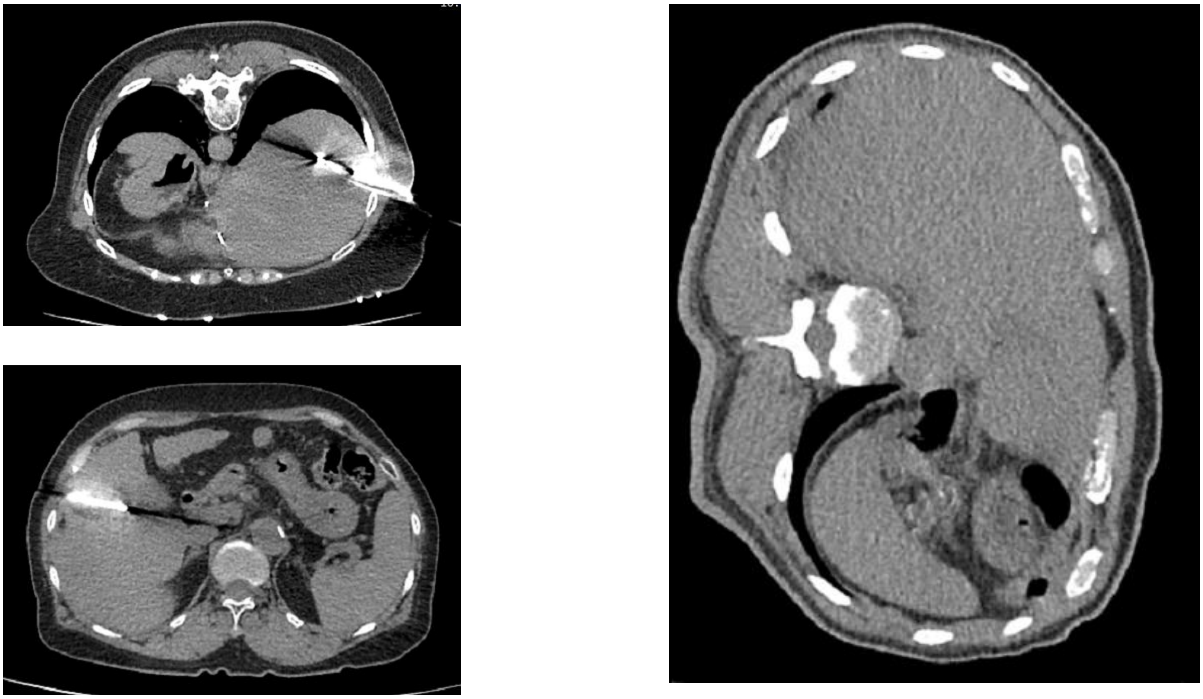


Figure 4-87 Examples of IKN dataset, images with treatment needle appeared as a light beam in addition to black line (left top and left bottom), patient lies down on the left side (right)

4.3.1 Validation guidelines and criteria

The tested samples had been sent to IKN with a suggested validation criteria. Only patients (CTY02, CTY03, CTY05, CTY06, CTY07) had been tested, Some images are excluded because it doesn't contain liver, or it is very dark, or it is scanned during the treatment procedure. The total tested samples are 1549 image. Three types of images are supplied as in Figure 4-88 , original image (right), generated liver mask (middle), mask mapped on the original image in red (left).

Validation criteria:

- The accuracy of the model could be the accumulation of the accuracy of all slides of the same patient, and then accumulated over all patients.
- Ideally, the accuracy of liver segmentation of a particular slide is the ratio of the overlapping area between the detected liver region and the real liver region, to the area of real liver region. This would require a domain expert to delineate the true liver region slide by slide.
- However, if step-2 is not viable at this stage, a comprised choice is to determine the accuracy per slide based on a range rather than a specific value. For example, the accuracy could be as follow:
    - between 0% to 20%
    - from 21% to 40%
    - from 41% to 60%
    - from 61% to 75%
    - more than 75%

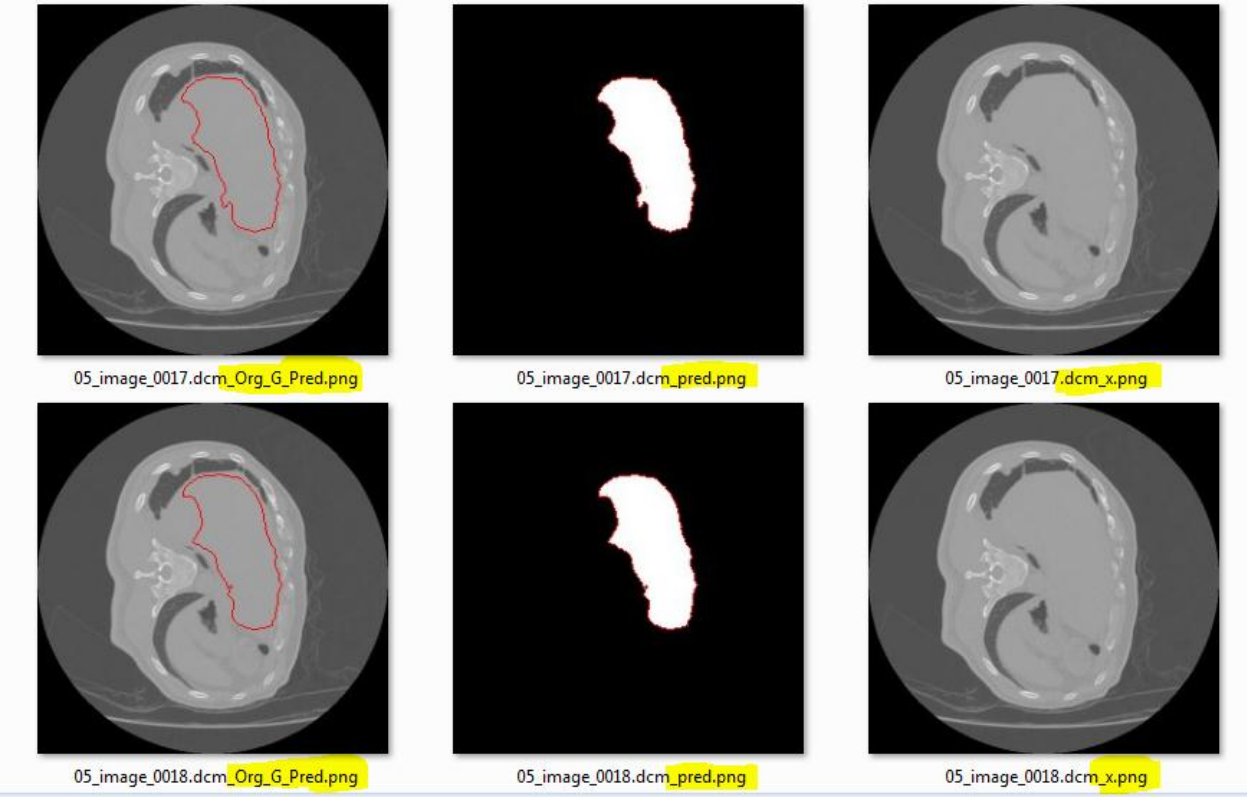


Figure 4-88 Example of tested data for IKN validation, original image (right), generated liver mask (middle), mask mapped on the original image in red (left).

4.3.2 IKN Validation results

The results based on IKN verifications show that, 95.93% of the images recorded more than 75% accuracy while 2.969% with accuracy between 61 and 75% and only 17 images (~1%) recorded accuracy less than 60% Table 4-45. The results indicated that, the model succeeded to segment the liver for most of the images with high accuracy although the model was trained on different dataset. The proposed model for compound connections between two U-Nets with deep-supervision and weighted loss function can be used for transfer learning between different datasets. According to the validation results, the accuracy could be more 90% for significant number of images but it will be time consuming for the clinical experts

from IKN to carry out this fine detailed validation or generating a manual annotated mask for the dataset.

Patient	CTY02	CTY06	CTY07	CTY03	CTY05	Total Images	%
Accuracy More than 75%	564	8	233	292	389	1486	95.9329%
Accuracy From 61 to 75%	16	0	21	1	8	46	2.9697%
Accuracy From 41 to 60%	3	0	0	6	2	11	0.7101%
Accuracy From 21 to 40%	0	0	0	2	2	4	0.2582%
Accuracy From 0 to 20%	0	0	0	1	1	2	0.1291%
Total images	583	8	254	302	402	1549	

Table 4-45 statistical results based on IKN validation

Chapter 5: Conclusions and Future work

Deeper models

The To-Down and Bottom-up approaches are alternatives to each other's as they represent the same U-Net based models with different approach of sorting and grouping. In 90% of the models, going deeper with the models inhanced the accuracy until certain levels then the accuracy will start to decrease. Over all maximum values for Validation, Test_Avg, Test_Avg_Aug factors recorded for models (32-512, 8-512, 64-2048) with values (98.00%, 78.56%, 80.39%) while the maximum accuracy for Training is (97.81%) that achieved using 2 models (64-204, 128-1024). The model (64-2048) recorded the maximum accuracy for 2 of 4 factors (Training, Test_Avg_Aug). The maximum accuracy for all 4 factors (Training, Validation, Test_Avg, Test_Avg_Aug) over all the 27 models recorded for models that ended with number of filters at the deepest level are one of three values(512, 1024, 2048) and the topmost levels started with number of filters (8,16, 32, 64). From 24 maximum values achieved for 4 factors, the recommended model depth could be (4, 5, 6, and 7) levels with minimum image size at the deepest level are (32*32, 16*16, 8*8, 4*4).

Wider Models

When using image size 256*256 with U-Net model, the deepest level will apply convolutional process over the minimum image size 16*16 because each level has maxpooling layer with size 2*2 which decrease the image size by 50% after each level. The accuracy results showed that, **stacked 2U-Nets** recorded the best accuracy (77.96%, 77.95%) for (Avg_Test, and Avg_AUG_test) over the original U-Net (58.73%, 75.93%) and over all other structures using 3 or 4 stacked U-Nets, While using 4 stacked U-Nets recorded the minimum accuracy. the results indicates that, increasing the number of stacked U-Net which increased the number of trainable parameters can enhance the model accuracy until certain values which limited to 2 stacked U-Nets with around 62millions of

parameters. increasing the number of U-Nets and parameters over 2 stacked will decrease the accuracy. Train stacked U-Net based models using image size 128*128 will decrease the image size at the deepest level to 8*8 which shows instability of the accuracy results compared with image size 256*256. The results indicate that, original U-Net achieved maximum accuracy for validation and testing (91.91%, 77.20%) and minimum for training, while 4 U-Nets recorded the maximum in training and Avg_AUG_Test (97.21%, 87.00%).

Skip connections variations

The two bridged U-Net model with the **Compound** skip connections proved enhancement of the model accuracy over the U-Net model and higher accuracy than Original and Modified skip connections. the Compound model concatenate the output feature maps from all previous 3 paths of the two U-Nets to the inputs of the last expansion path of the second U-Net before the de-convolution process. The concatenation minimizes the loss of features that may happen during the previous convolutions and de-convolutions processes.

Adding a third U-Net and adding long connection between the 1st and 3rd U-Net didn't reach the maximum accuracy recorded by the 2U-Net compound model but in general recorded better accuracy than Original and Modified 2U-Net for model of the 3U-Net based models.

The maximum accuracy for 2U-Net models is greater than 3U-Net models except for Test_Avg where 2U-Net models is lower than the maximum of 3U-Net models (Compound < Original_Modified) while the minimum for 3U-Net is lower than the minimums for 2U-Net. All 3U-Net models with long connections have the same behavior as with 3U-Net model. The maximum accuracy recorded for 2U-Net models always greater than the maximum for 3U-Net models with long connections except for testing while the minimum

for all factors for models with long connections is lower than the minimums for 2U-Net models.

Adding Long connections to the 3U-Net models slightly increased the accuracy for 5 of 9 models for all training, validation, testing, and testing using augmented data. These models are Compound_Modified_Long, Modified_Compound_Long, Original_Compound_Long, Compound_Original_Long, and Modified_Modified_Long. The rest 4 of 9 models showed the opposite trend, where the normal models recorded better performance than using long connections with all training, validation, testing, and testing using augmented data. These models are Original_Modified, Modified_Original, Compound_Compound, and Original_Original. All the models that contains Compound model either at the beginning or at the end showed better accuracy when using long connections except Compound_Compound better accuracy without using long connections. All models contain the same structure (original_original, Compound_Compound) recorded lower accuracy when using long connection except Modified_Modified which showed better accuracy with long connection. Original_modified or Modified_Original models showed that, the accuracy decreased when using long connections.

Adding Long connections to the 3U-Net models slightly increased the overall maximum accuracies. The maximum accuracy for (validation, testing, and testing using augmented data) for models with long connections (**89.68%, 93.18%, 93.29%)** are higher than the normal models (**87.27%, 91.52%, and 92.64%) ,** while for training the maximum for normal models is greater than the maximum for long connection models with **0.01% (97.70% -- 97.69%).**

For both **Test_Avg** and **Test_Avg_Aug** factors, most of **the models started with Compound** connections recorded higher accuracy than both the models started with Modified and original connections. The models started with Modified connections recorded accuracy higher than the similar models started with Original connections.

For both **Test_Avg** and **Test_Avg_Aug factors**, **most of the models ended with Compound** connections recorded higher accuracy than the models ended with Original connections .The models ended with Modified connections recorded accuracy higher than the similar models ended with Original connections. While Test_Avg_Aug factor for the models ended with Compound connections recorded accuracy Higher than the models ended with Modified connections, for Test_Avg the models ended with Compound connections recorded accuracy lower than the models ended with Modified connections.

The **Training** accuracy for all models is very close and varies from 97.85% to 97.52%. While the Compound model achieved the maximum **Test_Avg_Aug** (Testing using augmented data) (94.42%) and maximum **Validation** (90.11%) and second maximum for Test_Avg (89.88%), Compound model also got the minimum **Training** accuracy (97.52%).

Loss Function for flipping issue

Using centroid function with weight (3:1) solved the flipping issue for 50% of the samples that contains the issue and enhanced the remaining 50% of the samples ~ 20% than the compound model.

Using deep-Supervision with de-convolution approach using DSC as a loss function enhanced the output of the first half of the samples than using Centroid but the last half of the samples that solved by Centroid started to get 10% of flipping issue to appear again. While replacing the DSC function with the combined DSC and Centroid loss functions with 3:1 weight to the deep-supervision model enhance the overall output masks over Centroid alone and DSC with deep-Supervision using de-convolution, the flipping issue still exists with a very small size in a small number of samples.

The model based on Deep-Supervision with Multi-Resolution masks recorded significant improving in the output masks where the size of the duplicate plops got smaller than all the

previous models. Although the model started to detect the liver at slice number 34 while the ground truth detected it at slice 19, it is considered as the best performance of all models where it detected the liver starting from slices number 40 and 50. Replacing the DSC loss function with the weighted DSC and Centroid loss function with weights (3:1) decreased the overall accuracy of the model. Using weighted DSC and Centroid loss function showed the flipping issue again for the last half of the slices with larger size than the models (Bridge with Centroid, Deep-supervision with De-convolution and DSC, Deep-supervision with De-convolution and Centroid, Deep-supervision with Multi-Resolution and DSC).

Loss functions can be ordered based on the percentage of solving the flipping issue to, 1- Deep-Supervision with Multi-Resolution masks using weighted loss function combining DSC and centroid with ratio (5:1). 2- Deep-Supervision with De-Convolution using weighted loss function for DSC and Centroid with ratio (3:1), 3- Deep-Supervision with De-Convolution using DSC loss function, 4- Deep-Supervision with Multi-Resolution masks using weighted loss function of DSC and centroid with ratio (3:1). 5- W-Net with compound bridged connection using weighted loss function DSC and Centroid (3:1).

Future work

There are some topics I could not explore and I wish I have the chance to make research related to it.

- Applying image pre-processing on the training data to enhance the model performance for segmentation.
- Build a model the catch the advantages from both deeper and wider model in a new ensemble models.
- Extend the compound model connections to catch all the missing features during all the convolutional layers on the previous U-Net paths to enhance the segmentation accuracy.
- Design a theoretical formula for the weighted loss functions ratio and the weighted multiple output of the deep-supervision.
- Apply my new model (compound bridged 2 U-Net) for other organs e.g. kidney, tumors and mandible, I started that but the result is not good enough yet.
- Designing a model that can segment multiple organs at the same time with high speed performance and hopefully it can be instantly using a visual technology like hologram.

Appendices

6.1 The implemented U-Net model

```
def  get_model(optimizer, loss_metric, metrics, lr=1e-4):
   Inputs = Input ((IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS))
##################################################
   conv1 = Conv2D (64, (3, 3), activation='relu', padding='same') (inputs)
   conv1 = Conv2D (64, (3, 3), activation='relu', padding='same') (conv1)
   pool1 = MaxPooling2D (pool_size= (2, 2))(conv1)
   drop1 = Dropout (0.5) (pool1)
  ##################################################
   conv2 = Conv2D (128, (3, 3), activation='relu', padding='same') (drop1)
   conv2 = Conv2D (128, (3, 3), activation='relu', padding='same') (conv2)
   pool2 = MaxPooling2D (pool_size= (2, 2))(conv2)
   drop2 = Dropout (0.5) (pool2)
##################################################
   conv3 = Conv2D (256, (3, 3), activation='relu', padding='same') (drop2)
   conv3 = Conv2D (256, (3, 3), activation='relu', padding='same') (conv3)
   pool3 = MaxPooling2D (pool_size= (2, 2))(conv3)
   drop3 = Dropout (0.5) (pool3)
  ##################################################
   conv4 = Conv2D (512, (3, 3), activation='relu', padding='same') (drop3)
   conv4 = Conv2D (512, (3, 3), activation='relu', padding='same') (conv4)
   pool4 = MaxPooling2D (pool_size= (2, 2)) (conv4)
   drop4 = Dropout (0.5) (pool4)
  ##################################################
  conv5 = Conv2D (1024, (3, 3), activation='relu', padding='same') (drop4)
```

```
    conv5 = Conv2D (1024, (3, 3), activation='relu', padding='same') (conv5)

#################################################

   up6 = concatenate ([Conv2DTranspose (512, (2, 2), strides= (2, 2), padding='same')
(conv5), conv4], axis=3)

    conv6 = Conv2D (512, (3, 3), activation='relu', padding='same')(up6)

    conv6 = Conv2D (512, (3, 3), activation='relu', padding='same')(conv6)

    conv6 = Dropout (rate=0.4) (conv6)

#################################################

    up7 = concatenate ([Conv2DTranspose (256, (2, 2), strides=(2, 2), padding='same')
(conv6), conv3], axis=3)

    conv7 = Conv2D (256, (3, 3), activation='relu', padding='same') (up7)

    conv7 = Conv2D (256, (3, 3), activation='relu', padding='same') (conv7)

    conv7 = Dropout (rate=0.4) (conv7)

   #################################################

   up8 = concatenate ([Conv2DTranspose (128, (2, 2), strides= (2, 2), padding='same')
(conv7), conv2], axis=3)

    conv8 = Conv2D (128, (3, 3), activation='relu', padding='same') (up8)

    conv8 = Conv2D (128, (3, 3), activation='relu', padding='same') (conv8)

    conv8 = Dropout (rate=0.4) (conv8)

#################################################

    up9 = concatenate ([Conv2DTranspose (64, (2, 2), strides= (2, 2), padding='same')
(conv8), conv1], axis=3)

    conv9 = Conv2D (64, (3, 3), activation='relu', padding='same') (up9)

    conv9 = Conv2D (64, (3, 3), activation='relu', padding='same') (conv9)

    conv9 = Dropout (rate=0.4) (conv9)

   conv10 = Conv2D (1, (1, 1), activation='sigmoid')(conv9)

    Model = Model (inputs= [inputs], outputs= [conv10])

    model.compile (optimizer=Nadam (lr=1e-5), loss=dice_coef_loss, metrics= [dice_coef])
```

Return model

6.2 The implemented 2 bridged U-Net with original connection

```
def  get_model(optimizer, loss_metric, metrics, lr=1e-4):
   Inputs = Input((IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS))
##########################################################
   conv1 = Conv2D (32, (3, 3), activation='elu', padding='same') (inputs)
   conv1 = Conv2D (32, (3, 3), activation='elu', padding='same') (conv1)
   pool1 = MaxPooling2D (pool_size= (2, 2))(conv1)
   drop1 = Dropout (0.5) (pool1)
    ################################################
   conv2 = Conv2D (64, (3, 3), activation='elu', padding='same') (drop1)
   conv2 = Conv2D (64, (3, 3), activation='elu', padding='same') (conv2)
   pool2 = MaxPooling2D (pool_size= (2, 2)) (conv2)
   drop2 = Dropout (0.5) (pool2)
 ################################################
   conv3 = Conv2D (128, (3, 3), activation='elu', padding='same') (drop2)
   conv3 = Conv2D (128, (3, 3), activation='elu', padding='same') (conv3)
   pool3 = MaxPooling2D (pool_size= (2, 2)) (conv3)
   drop3 = Dropout (0.5)(pool3)
 ################################################
   conv4 = Conv2D (256, (3, 3), activation='relu', padding='same') (drop3)
   conv4 = Conv2D (256, (3, 3), activation='relu', padding='same') (conv4)
   pool4 = MaxPooling2D (pool_size= (2, 2))(conv4)
   drop4 = Dropout (0.5) (pool4)
 ################################################
   conv5 = Conv2D (512, (3, 3), activation='relu', padding='same') (drop4)
   conv5 = Conv2D (512, (3, 3), activation='relu', padding='same') (conv5)
```

```
 ################################################

   up6 = concatenate ([Conv2DTranspose (256, (2, 2), strides= (2, 2), padding='same')
(conv5), conv4], axis=3)

   conv6 = Conv2D (256, (3, 3), activation='elu', padding='same') (up6)

   conv6 = Conv2D (256, (3, 3), activation='elu', padding='same') (conv6)

   conv6 = Dropout (rate=0.4) (conv6)

 ################################################

   up7 = concatenate ([Conv2DTranspose (128, (2, 2), strides= (2, 2), padding='same')
(conv6), conv3], axis=3)

   conv7 = Conv2D (128, (3, 3), activation='elu', padding='same') (up7)

   conv7 = Conv2D (128, (3, 3), activation='elu', padding='same') (conv7)

   conv7 = Dropout (rate=0.4) (conv7)

 ################################################

   up8 = concatenate ([Conv2DTranspose (64, (2, 2), strides= (2, 2), padding='same')
(conv7), conv2], axis=3)

   conv8 = Conv2D (64, (3, 3), activation='elu', padding='same') (up8)

   conv8 = Conv2D (64, (3, 3), activation='elu', padding='same') (conv8)

   conv8 = Dropout (rate=0.4) (conv8)

 ################################################

   up9 = concatenate ([Conv2DTranspose (32, (2, 2), strides= (2, 2), padding='same')
(conv8), conv1], axis=3)

   conv9 = Conv2D (32, (3, 3), activation='elu', padding='same') (up9)

   conv9 = Conv2D (32, (3, 3), activation='elu', padding='same') (conv9)

   conv9 = Dropout (rate=0.4) (conv9)



#################### Second U-Net ############

   conv10 = Conv2D (32, (3, 3), activation='elu', padding='same') (conv9)

   conv10 = Conv2D (32, (3, 3), activation='elu', padding='same') (conv10)
```

```
    pool10 = MaxPooling2D (pool_size= (2, 2)) (conv10)
    drop10 = Dropout (0.5) (pool10)
  ################################################
    bridge_1 = concatenate ([(drop10), conv8], axis=3)
    conv11 = Conv2D (64, (3, 3), activation='elu', padding='same') (bridge_1)
    conv11 = Conv2D (64, (3, 3), activation='elu', padding='same') (conv11)
    pool11 = MaxPooling2D (pool_size= (2, 2))(conv11)
    drop11 = Dropout (0.5) (pool11)
     ################################################
    bridge_2 = concatenate ([(drop11), conv7], axis=3)
    conv12 = Conv2D (128, (3, 3), activation='elu', padding='same') (bridge_2)
    conv12 = Conv2D (128, (3, 3), activation='elu', padding='same') (conv12)
    pool12 = MaxPooling2D (pool_size= (2, 2))(conv12)
    drop12 = Dropout (0.5)(pool12)
  ################################################
    bridge_3 = concatenate ([(drop12), conv6], axis=3)
    conv13 = Conv2D (256, (3, 3), activation='relu', padding='same') (bridge_3)
    conv13 = Conv2D (256, (3, 3), activation='relu', padding='same') (conv13)
    pool13 = MaxPooling2D (pool_size= (2, 2))(conv13)
    drop13 = Dropout (0.5) (pool13)
  ################################################
    bridge_4 = concatenate ([(drop13), conv5], axis=3)
    conv14 = Conv2D (512, (3, 3), activation='relu', padding='same') (bridge_4)
    conv14 = Conv2D (512, (3, 3), activation='relu', padding='same') (conv14)
  ################################################
    up15 = concatenate ([Conv2DTranspose (256, (2, 2), strides= (2, 2), padding='same')
(conv14), conv13, conv4], axis=3)
```

conv15 = Conv2D (256, (3, 3), activation='elu', padding='same') (up15)

conv15 = Conv2D (256, (3, 3), activation='elu', padding='same') (conv15)

conv15 = Dropout (rate=0.4) (conv15)

##

up16 = concatenate ([Conv2DTranspose (128, (2, 2), strides= (2, 2), padding='same') (conv15), conv12, conv3], axis=3)

conv16 = Conv2D (128, (3, 3), activation='elu', padding='same') (up16)

conv16 = Conv2D (128, (3, 3), activation='elu', padding='same') (conv16)

conv16 = Dropout (rate=0.4) (conv16)

##

up17 = concatenate ([Conv2DTranspose (64, (2, 2), strides= (2, 2), padding='same') (conv16), conv11, conv2], axis=3)

conv17 = Conv2D (64, (3, 3), activation='elu', padding='same') (up17)

conv17 = Conv2D (64, (3, 3), activation='elu', padding='same') (conv17)

conv17 = Dropout (rate=0.4) (conv17)

##

up18 = concatenate ([Conv2DTranspose (32, (2, 2), strides=(2, 2), padding='same') (conv17), conv10 , conv1], axis=3)

conv18 = Conv2D (32, (3, 3), activation='elu', padding='same') (up18)

conv18 = Conv2D (32, (3, 3), activation='elu', padding='same') (conv18)

conv18 = Dropout (rate=0.4) (conv18)

##

conv19 = Conv2D (1, (1, 1), activation='sigmoid') (conv18)

Model = Model (inputs= [inputs], outputs=[conv19])

model.compile (optimizer=Nadam (lr=1e-5), loss=dice_coef_loss, metrics= [dice_coef])

Return model

6.3 Models summary

6.3.1 2 Bridge U-Net with Original connections

Layer (type)	Output Shape	Parameter	Connected to
input_1 (InputLayer)	(None, 256, 256, 1)	0	
conv2d_1 (Conv2D)	(None, 256, 256, 32)	320	input_1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 32)	9248	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 128, 128, 32)	0	conv2d_2[0][0]
dropout_1 (Dropout)	(None, 128, 128, 32)	0	max_pooling2d_1[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 64)	18496	dropout_1[0][0]
conv2d_4 (Conv2D)	(None, 128, 128, 64)	36928	conv2d_3[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_4[0][0]
dropout_2 (Dropout)	(None, 64, 64, 64)	0	max_pooling2d_2[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 128)	73856	dropout_2[0][0]
conv2d_6 (Conv2D)	(None, 64, 64, 128)	147584	conv2d_5[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 128)	0	conv2d_6[0][0]
dropout_3 (Dropout)	(None, 32, 32, 128)	0	max_pooling2d_3[0][0]
conv2d_7 (Conv2D)	(None, 32, 32, 256)	295168	dropout_3[0][0]
conv2d_8 (Conv2D)	(None, 32, 32, 256)	590080	conv2d_7[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 256)	0	conv2d_8[0][0]
dropout_4 (Dropout)	(None, 16, 16, 256)	0	max_pooling2d_4[0][0]
conv2d_9 (Conv2D)	(None, 16, 16, 512)	1180160	dropout_4[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 512)	2359808	conv2d_9[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 32, 32, 256)	524544	conv2d_10[0][0]
concatenate_1 (Concatenate)	(None, 32, 32, 512)	0	conv2d_transpose_1[0][0]
			conv2d_8[0][0]
conv2d_11 (Conv2D)	(None, 32, 32, 256)	1179904	concatenate_1[0][0]
conv2d_12 (Conv2D)	(None, 32, 32, 256)	590080	conv2d_11[0][0]
dropout_5 (Dropout)	(None, 32, 32, 256)	0	conv2d_12[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 64, 64, 128)	131200	dropout_5[0][0]
concatenate_2 (Concatenate)	(None, 64, 64, 256)	0	conv2d_transpose_2[0][0]
			conv2d_6[0][0]
conv2d_13 (Conv2D)	(None, 64, 64, 128)	295040	concatenate_2[0][0]
conv2d_14 (Conv2D)	(None, 64, 64, 128)	147584	conv2d_13[0][0]
dropout_6 (Dropout)	(None, 64, 64, 128)	0	conv2d_14[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 128, 128, 64)	32832	dropout_6[0][0]
concatenate_3 (Concatenate)	(None, 128, 128, 128)	0	conv2d_transpose_3[0][0]

			conv2d_4[0][0]
conv2d_15 (Conv2D)	(None, 128, 128, 64)	73792	concatenate_3[0][0]
conv2d_16 (Conv2D)	(None, 128, 128, 64)	36928	conv2d_15[0][0]
dropout_7 (Dropout)	(None, 128, 128, 64)	0	conv2d_16[0][0]
conv2d_transpose_4 (Conv2DTrans	(None, 256, 256, 32)	8224	dropout_7[0][0]
concatenate_4 (Concatenate)	(None, 256, 256, 64)	0	conv2d_transpose_4[0][0]
			conv2d_2[0][0]
conv2d_17 (Conv2D)	(None, 256, 256, 32)	18464	concatenate_4[0][0]
conv2d_18 (Conv2D)	(None, 256, 256, 32)	9248	conv2d_17[0][0]
dropout_8 (Dropout)	(None, 256, 256, 32)	0	conv2d_18[0][0]
conv2d_19 (Conv2D)	(None, 256, 256, 32)	9248	dropout_8[0][0]
conv2d_20 (Conv2D)	(None, 256, 256, 32)	9248	conv2d_19[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 128, 128, 32)	0	conv2d_20[0][0]
dropout_9 (Dropout)	(None, 128, 128, 32)	0	max_pooling2d_5[0][0]
concatenate_5 (Concatenate)	(None, 128, 128, 96)	0	dropout_9[0][0]
			dropout_7[0][0]
conv2d_21 (Conv2D)	(None, 128, 128, 64	55360	concatenate_5[0][0]
conv2d_22 (Conv2D)	(None, 128, 128, 64)	36928	conv2d_21[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_22[0][0]
dropout_10 (Dropout)	(None, 64, 64, 64)	0	max_pooling2d_6[0][0]
concatenate_6 (Concatenate)	(None, 64, 64, 192)	0	dropout_10[0][0]
			dropout_6[0][0]
conv2d_23 (Conv2D)	(None, 64, 64, 128)	221312	concatenate_6[0][0]
conv2d_24 (Conv2D)	(None, 64, 64, 128	147584	conv2d_23[0][0]
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 128)	0	conv2d_24[0][0]
dropout_11 (Dropout)	(None, 32, 32, 128)	0	max_pooling2d_7[0][0]
concatenate_7 (Concatenate)	(None, 32, 32, 384)	0	dropout_11[0][0]
			dropout_5[0][0]
conv2d_25 (Conv2D)	(None, 32, 32, 256)	884992	concatenate_7[0][0]
conv2d_26 (Conv2D)	(None, 32, 32, 256)	590080	conv2d_25[0][0]
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 256)	0	conv2d_26[0][0]
dropout_12 (Dropout)	(None, 16, 16, 256)	0	max_pooling2d_8[0][0]
concatenate_8 (Concatenate)	(None, 16, 16, 768)	0	dropout_12[0][0]
			conv2d_10[0][0]
conv2d_27 (Conv2D)	(None, 16, 16, 512)	3539456	concatenate_8[0][0]
conv2d_28 (Conv2D)	(None, 16, 16, 512)	2359808	conv2d_27[0][0]
conv2d_transpose_5 (Conv2DTrans	(None, 32, 32, 256)	524544	conv2d_28[0][0]
concatenate_9 (Concatenate)	(None, 32, 32, 768)	0	conv2d_transpose_5[0][0]
			conv2d_26[0][0]

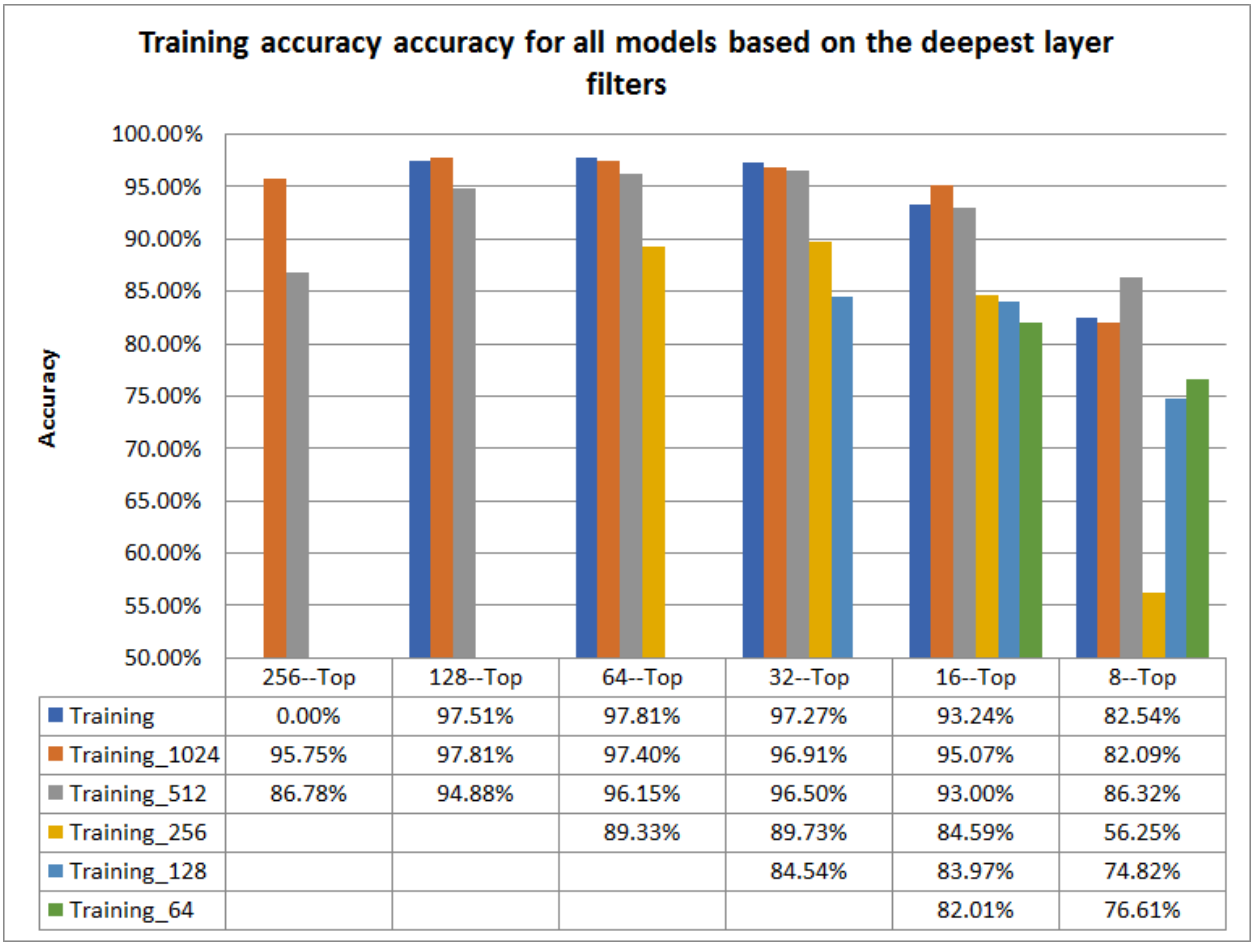
			conv2d_8[0][0]
conv2d_29 (Conv2D)	(None, 32, 32, 256)	1769728	concatenate_9[0][0]
conv2d_30 (Conv2D)	(None, 32, 32, 256)	590080	conv2d_29[0][0]
dropout_13 (Dropout)	(None, 32, 32, 256)	0	conv2d_30[0][0]
conv2d_transpose_6 (Conv2DTrans	(None, 64, 64, 128)	131200	dropout_13[0][0]
concatenate_10 (Concatenate)	(None, 64, 64, 384)	0	conv2d_transpose_6[0][0]
			conv2d_24[0][0]
			conv2d_6[0][0]
conv2d_31 (Conv2D)	(None, 64, 64, 128)	442496	concatenate_10[0][0]
conv2d_32 (Conv2D)	(None, 64, 64, 128)	147584	conv2d_31[0][0]
dropout_14 (Dropout)	(None, 64, 64, 128)	0	conv2d_32[0][0]
conv2d_transpose_7 (Conv2DTrans	(None, 128, 128, 64)	32832	dropout_14[0][0]
concatenate_11 (Concatenate)	(None, 128, 128, 192)	0	conv2d_transpose_7[0][0]
			conv2d_22[0][0]
			conv2d_4[0][0]
conv2d_33 (Conv2D)	(None, 128, 128, 64)	110656	concatenate_11[0][0]
conv2d_34 (Conv2D)	(None, 128, 128, 64)	36928	conv2d_33[0][0]
dropout_15 (Dropout)	(None, 128, 128, 64)	0	conv2d_34[0][0]
conv2d_transpose_8 (Conv2DTrans	(None, 256, 256, 32)	8224	dropout_15[0][0]
concatenate_12 (Concatenate)	(None, 256, 256, 96)	0	conv2d_transpose_8[0][0]
			conv2d_20[0][0]
			conv2d_2[0][0]
conv2d_35 (Conv2D)	(None, 256, 256, 32)	27680	concatenate_12[0][0]
conv2d_36 (Conv2D)	(None, 256, 256, 32)	9248	conv2d_35[0][0]
dropout_16 (Dropout)	(None, 256, 256, 32)	0	conv2d_36[0][0]
conv2d_37 (Conv2D)	(None, 256, 256, 1)	33	dropout_16[0][0]
Total params:	19,444,737		
Trainable params:	19,444,737		
Non-trainable params:	0		

6.4 Analysis of all deeper models based on the number of filters applied at the deepest level

In this section, all models included in deeper U-Net experiments will be compared and full dataset in Table 6-1

Name	Test_Avg	Test_Avg_Aug	Training	Validation	Parameters	Levels	Min Image size	Batch size
256--2048	0.00%	0.00%	0.00%	0.00%	N/A	3	32	N/A
128--2048	69.61%	74.63%	97.51%	87.99%	124,107,393	4	16	4
64--2048	60.24%	**80.39%**	**97.81%**	**93.68%**	**124,361,025**	**5**	**8**	**8**
32--2048	68.59%	**64.62%**	97.27%	86.96%	124,424,353	6	4	16
16--2048	**45.36%**	67.05%	93.24%	85.31%	124,440,145	7	2	32
8--2048	**74.50%**	73.20%	**82.54%**	**82.93%**	**142,444,073**	**8**	**1**	**32**
256--1024	53.23%	70.20%	95.75%	83.64%	29,761,793	3	64	4
128--1024	60.98%	66.26%	**97.81%**	88.82%	30,776,961	4	32	8
64--1024	58.73%	**75.93%**	**97.40%**	91.50%	31,030,593	5	16	16
32--1024	66.42%	70.45%	96.91%	**92.72%**	31,093,921	6	8	32
16--1024	**69.81%**	70.80%	95.07%	90.23%	**31,109,713**	**7**	**4**	**32**
8--1024	**34.82%**	**29.09%**	**82.09%**	**57.89%**	31,113,641	8	2	32
256--512	**28.74%**	41.02%	**86.78%**	78.33%	6,427,393	2	128	4
128--512	50.86%	52.50%	94.88%	85.72%	7,442,561	3	64	8
64--512	49.96%	44.23%	**96.15%**	83.52%	7,696,193	4	32	16
32--512	46.49%	60.07%	**96.50%**	**98.00%**	7,759,521	5	16	16
16--512	**40.79%**	**39.95%**	93.00%	**77.85%**	7,775,313	6	8	32
8--512	**78.56%**	**73.89%**	**86.32%**	89.89%	**7,779,241**	**7**	**4**	**32**
64--256	45.57%	39.25%	**89.33%**	72.55%	1,861,697	3	64	32
32--256	**41.84%**	**34.11%**	**89.73%**	**64.30%**	1,925,025	4	32	32
16--256	**52.60%**	**67.13%**	84.59%	**85.00%**	**1,940,817**	**5**	**16**	**32**
8--256	42.16%	46.11%	**56.25%**	76.52%	1,944,745	6	8	32
32--128	**45.08%**	38.81%	**84.54%**	**80.05%**	**465,953**	**3**	**64**	**32**
16--128	40.41%	**54.83%**	83.97%	78.58%	481,745	4	32	32
8--128	**34.09%**	**22.44%**	**74.82%**	**48.72%**	485,673	5	16	32
16--64	**44.47%**	**35.87%**	**82.01%**	**71.83%**	**116,753**	**3**	**64**	**32**
8--64	**41.33%**	**36.97%**	**76.61%**	**75.02%**	120,681	4	32	32

Table 6-1 All models with same start and different number of levels compared with all possible starts



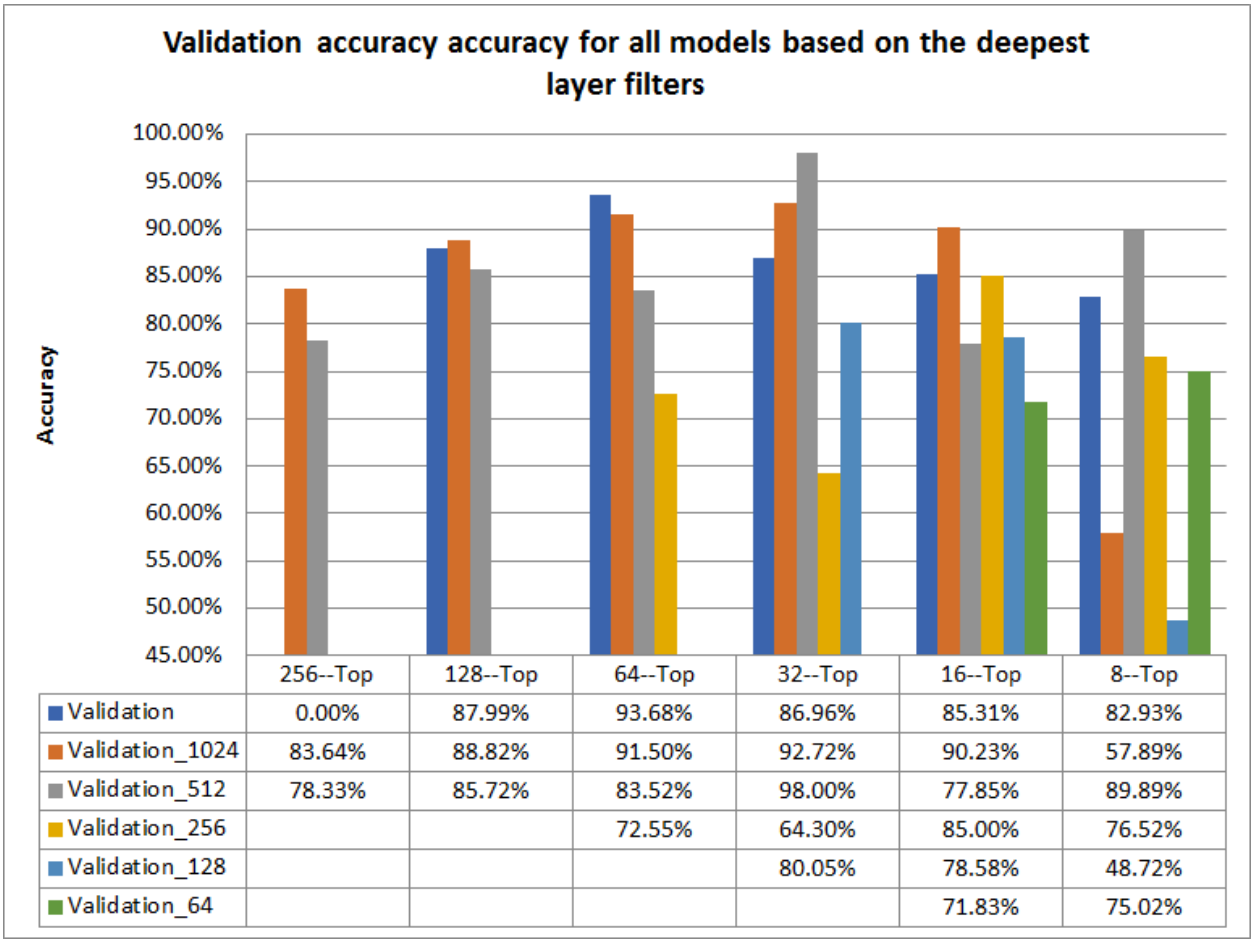
	256--Top	128--Top	64--Top	32--Top	16--Top	8--Top
Training	0.00%	97.51%	97.81%	97.27%	93.24%	82.54%
Training_1024	95.75%	97.81%	97.40%	96.91%	95.07%	82.09%
Training_512	86.78%	94.88%	96.15%	96.50%	93.00%	86.32%
Training_256			89.33%	89.73%	84.59%	56.25%
Training_128				84.54%	83.97%	74.82%
Training_64					82.01%	76.61%

Figure 6-1 Training accuracy for all models based on the deepest layer filters

For **training** factor, 4 of 6 categories of models that started with filters (256, 512, 1024, 2048) at the deepest level increased the accuracy with increasing the depth (number of levels on top of the deepest level) to reach the maximum at the top most applicable model with filters (32, 32, 128, 64) in sequence. The remaining 2 of 6 categories that start with filters (64, 128) started with the maximum accuracy at models (16-64, 32-128) and decreased with increasing the depth of the model.

The highest 5 maximum values achieved for models (128-1024, 64-2048, 128-2048, 64-1024, 32-2048) in sequence while the 5 minimum values achieved for models are (8-256, 8-12, 8-64, 16-64, 32-2048)in sequence. The number of models achieved the

maximum accuracy divided into 1 model with 16 filters at the top level, 3 models with 32 on the top level, 1 with 64 and 1 with 128 filters on the top level Figure 6-1 Table 6-1.

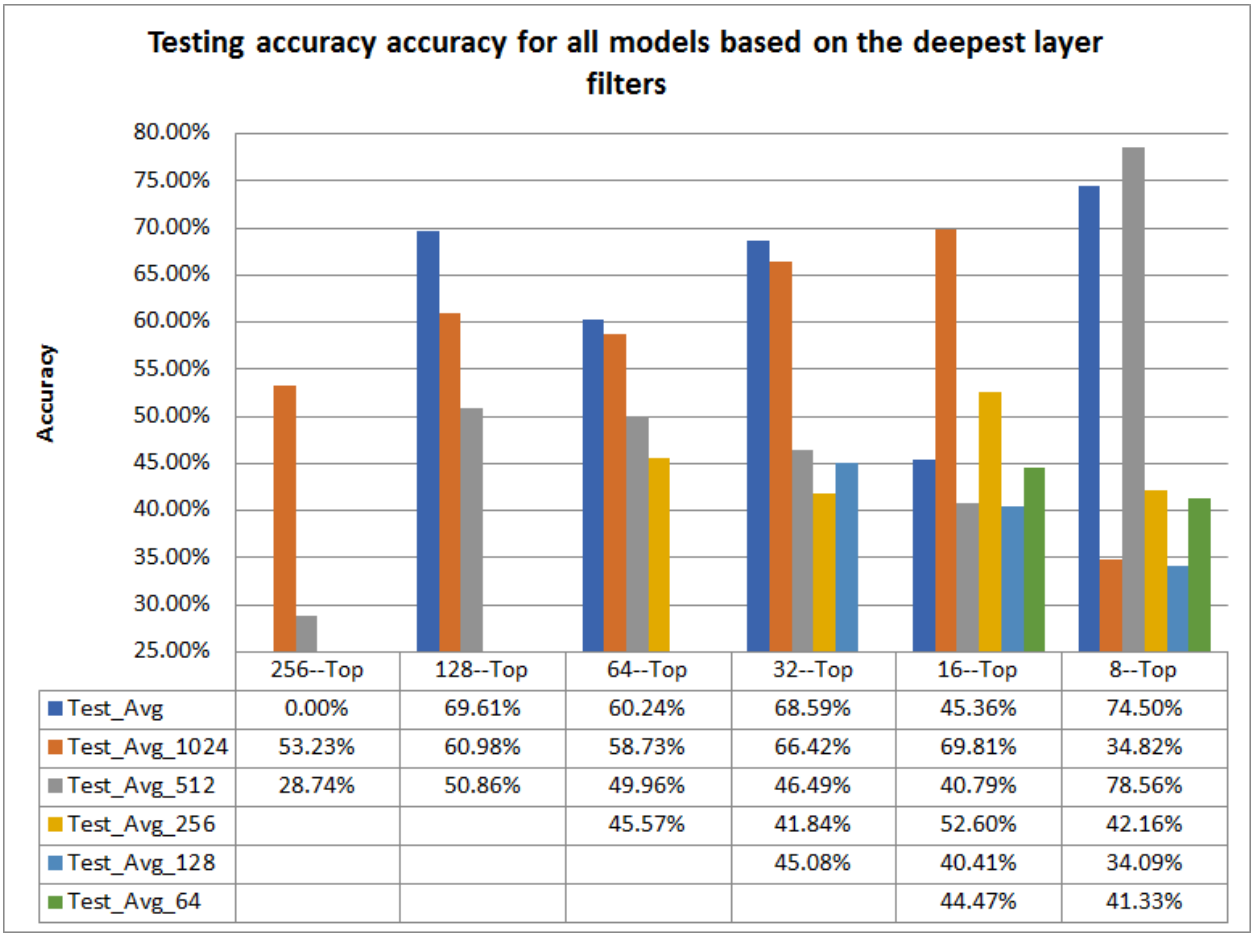


	256--Top	128--Top	64--Top	32--Top	16--Top	8--Top
Validation	0.00%	87.99%	93.68%	86.96%	85.31%	82.93%
Validation_1024	83.64%	88.82%	91.50%	92.72%	90.23%	57.89%
Validation_512	78.33%	85.72%	83.52%	98.00%	77.85%	89.89%
Validation_256			72.55%	64.30%	85.00%	76.52%
Validation_128				80.05%	78.58%	48.72%
Validation_64					71.83%	75.02%

Figure 6-2 Validation accuracy for all models based on the deepest layer filters

For **Validation** factor, 2 of 6 categories that ended with filters (2048, 1024)at the deepest level increased with the models getting deeper to reach the maximum with models (64-2048, 32-1024)then decreased while the models getting deeper. The next 2 categories that ended with filters (256,512) also increased with the models getting deeper to reach the maximum at models (16-256, 32-512) then decreased while these two categories had a drop in the accuracy during the increasing at models (32-256, 64-512) in sequence. The category of models that ended with filters 128 at the deepest level started with the maximum accuracy at model (32-128) then decreased while the remaining category ended

with 64 filters at the deepest level recorded only increasing of the accuracy with deeper models and reached the maximum at model (8-64). The highest 5 maximum values achieved for models (32-512, 64-2048, 32-1024, 64-1024, 16-1024) in sequence while the 5 minimum values achieved for models (8-12, 8-1024, 32-256, 16-64, 64-256)in sequence. The number of models achieved the maximum accuracy divided into 1 model with 8 filters at the top level, 1 model with 16 on the top level, 3 with 32 and 1 with 64 filters on the top level Table 6-1 Figure 6-2

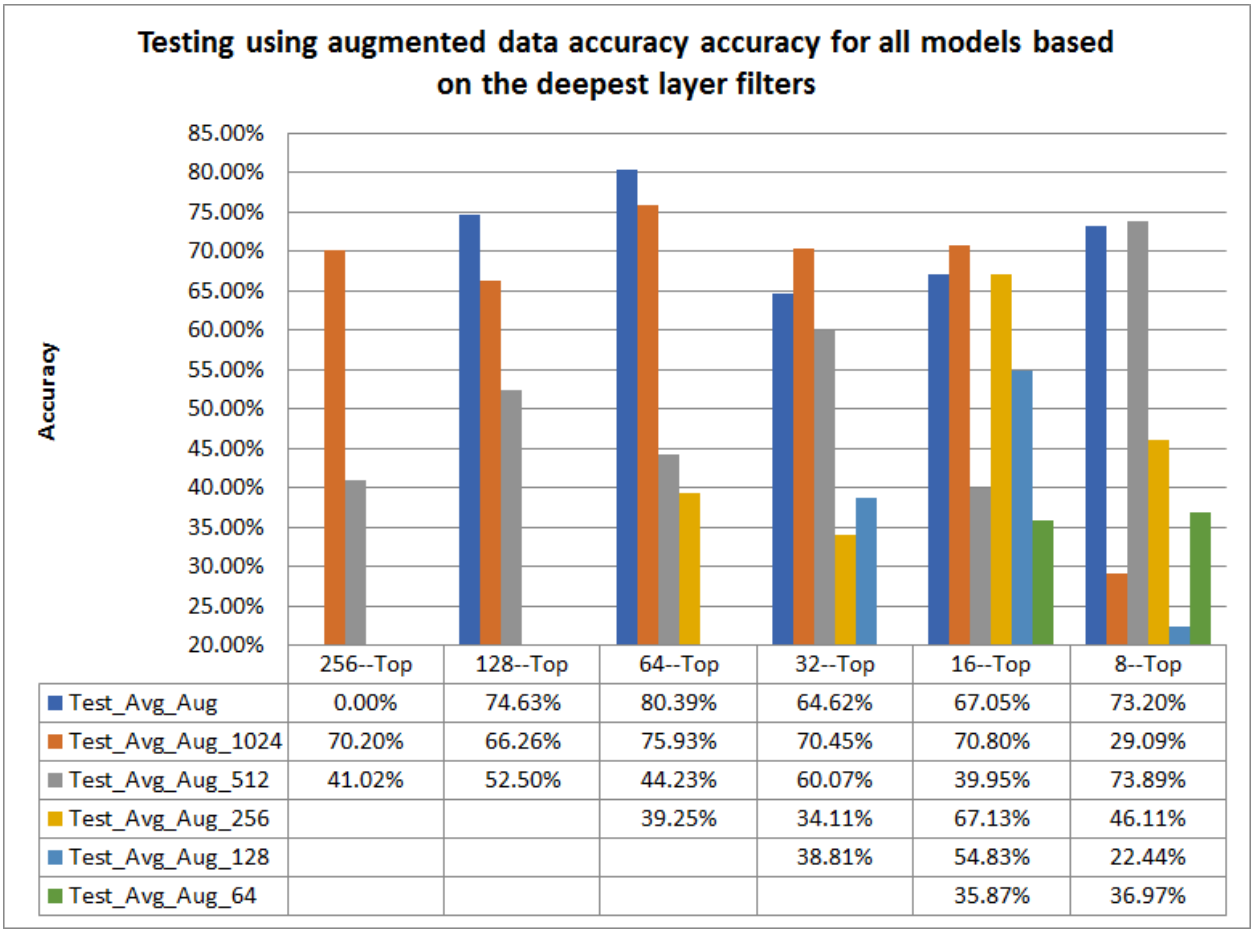


	256--Top	128--Top	64--Top	32--Top	16--Top	8--Top
Test_Avg	0.00%	69.61%	60.24%	68.59%	45.36%	74.50%
Test_Avg_1024	53.23%	60.98%	58.73%	66.42%	69.81%	34.82%
Test_Avg_512	28.74%	50.86%	49.96%	46.49%	40.79%	78.56%
Test_Avg_256			45.57%	41.84%	52.60%	42.16%
Test_Avg_128				45.08%	40.41%	34.09%
Test_Avg_64					44.47%	41.33%

Figure 6-3 Testing accuracy for all models based on the deepest layer filters

For **Testing** (Test_Avg) factor, 2 of 6 categories that ended with filters (1024, 215)at the deepest level increased with the models getting deeper to reach the maximum with models (16-1024, 16-256)then decreased while the models getting deeper while these two categories had a drop in the accuracy during the increasing at models (64-1024, 8-256)

in sequence. The 2 categories of models that ended with filters 64 and 128 at the deepest level started with the maximum accuracy at model (16-64, 16-128) then decreased. The remaining 2 categories ended with 512 and 2048 filters at the deepest level recorded fluctuation of the accuracy with deeper models until reached the maximum at deepest models (8-512, 8-2048). The highest 5 maximum values achieved for models (8-512, 8-2048, 16-1024, 128-2048, 32-2048) in sequence while the 5 minimum values achieved for models (256-512, 8-128, 8-1024, 16-128, 16-512)in sequence. The number of models achieved the maximum accuracy divided into 2 models with 8 filters at the top level and 4 models with 16 on the top level Figure 6-3 Table 6-1.



	256--Top	128--Top	64--Top	32--Top	16--Top	8--Top
Test_Avg_Aug	0.00%	74.63%	80.39%	64.62%	67.05%	73.20%
Test_Avg_Aug_1024	70.20%	66.26%	75.93%	70.45%	70.80%	29.09%
Test_Avg_Aug_512	41.02%	52.50%	44.23%	60.07%	39.95%	73.89%
Test_Avg_Aug_256			39.25%	34.11%	67.13%	46.11%
Test_Avg_Aug_128				38.81%	54.83%	22.44%
Test_Avg_Aug_64					35.87%	36.97%

Figure 6-4 Testing using augmented data (Test_Avg_Aug) accuracy for all models based on the deepest layer filters

For **Testing using augmented data** (Test_Avg_Aug) factor, for 2 of 6 categories that ended with filters (2048, 128) at the deepest level, the accuracy increased with the models getting deeper to reach the maximum with models (64-2048, 16-128) then decreased with the models are going deeper while one category that ended with 64 filters increase with deeper models to reach the maximum with model (8-64). The 2 categories of models that ended with filters (1024, 256) at the deepest level increased with the models are getting deeper to reach the maximum at models (64-1024, 16-256) then decreased with deeper models and had one drop at models (128-1024, 32-256)during the increasing while the category of models ended with 512 filters at the deepest level recorded two drops during the increasing at models (64-512, 16-512) before reaching the maximum at the deepest model (8-512).The highest 5 maximum values achieved for models (64-2048, 64-1024, 128-2048, 8-512, 8-2048) in sequence while the 5 minimum values achieved for models (8-128, 8-1024, 32-256, 16-64, 8-64)in sequence. The number of models achieved the maximum accuracy divided into 2 models with 8 filters at the top level and 2 models with 16 on the top level and 2 models with 64 filters at the top level. Table 6-1 Figure 6-4

6.5 Qualitative analysis for flipping issue's proposed solutions

Ground Truth	U-Net	Bridge 2U-Net	Compound U-Net	Centroid 3 : 1	Centroid 5 : 1	Deep supervision - deconv	Deep supervision - Deconv-Centroid (3:1)	Deep supervision - Deconv-Centroid (5:1)	Deep supervision - Deconv-Centroid (2:1)	Deep supervision -multi-resolution	Deep supervision with multi-resolution-centroid(3:1)	Deep supervision with multi-resolution-centroid(5:1)	Deep supervision with multi-resolution-centroid(2:1)

Model	1		2		3		4		5		6		7		8		9		10		11		12		13	
1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
3	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
4	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
5	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
6	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
7	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
8	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
9	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
10	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
11	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
12	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
14	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
15	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
16	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
17	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
18	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
19	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
20	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
21	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
22	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13

23	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
24	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
25	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
26	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
27	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
28	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
29	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
30	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
31	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
32	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
33	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
34	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
35	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
36	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
37	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
38	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
39	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
40	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
41	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
42	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
43	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
44	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
45	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
46	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
47	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
48	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
49	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
50	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
51	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	13	1	1	13	1	13	1	13	1	13
52	1	13	1	13	1	13	1	13	1	13	1	13	13	13	1	13	12	2	1	13	1	13	1	13	1	13

53	1	13	1	13	1	13	1	13	1	13	1	13	13	13	1	13	1	13	1	13	1	13	1	13	1	13
54	1	13	12	2	1	13	1	13	1	13	1	13	13	13	1	13	1	13	1	13	1	13	1	13	1	13
55	1	13	13	1	1	13	1	13	1	13	1	13	12	13	1	13	1	13	1	13	1	13	1	13	1	13
56	1	13	13	1	1	13	1	13	1	13	1	13	12	13	1	13	1	13	1	13	1	13	1	13	1	13
57	1	13	13	1	1	13	1	13	1	13	1	13	12	13	1	13	1	13	1	13	1	13	1	13	1	13
58	1	13	13	1	1	13	1	13	1	13	1	13	12	13	1	13	1	13	1	13	1	13	1	13	1	13
59	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
60	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
61	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
62	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
63	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
64	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
65	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
66	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
67	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
68	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
69	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
70	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
71	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
72	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
73	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
74	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
75	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
76	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
77	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
78	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
79	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
80	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
81	1	13	13	1	1	13	1	13	12	2	1	13	1	13	12	2	1	13	1	13	1	13	1	13	1	13
82	1	13	13	1	1	13	1	13	12	2	1	13	1	13	12	2	1	13	1	13	1	13	1	13	1	13

83	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
84	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
85	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
86	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
87	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
88	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
89	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
90	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
91	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
92	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
93	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
94	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
95	1	13	13	1	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
96	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
97	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
98	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
99	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
100	1	13	13	1	1	13	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
101	1	13	13	1	12	2	11	3	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13	1	13
102	1	13	13	1	12	2	11	3	10	4	8	6	1	13	10	4	1	13	1	13	1	13	1	13	1	13
103	6	13	13	1	11	3	12	2	8	6	10	4	1	13	8	6	9	5	1	13	1	13	1	13	1	13
104	1	13	13	1	11	3	12	2	8	6	9	5	1	13	8	6	10	4	1	13	6	8	1	13	1	13
105	4	13	13	1	12	2	11	3	7	7	9	5	2	12	7	7	10	4	3	11	6	8	1	13	5	9
106	5	13	13	1	12	2	11	3	5	9	8	6	1	13	5	9	10	4	3	11	4	10	2	12	9	5
107	3	13	13	1	12	2	11	3	3	11	3	11	8	6	3	11	10	4	2	12	3	11	1	13	9	5
108	3	13	13	1	12	2	11	3	4	10	8	6	7	7	4	10	10	4	1	13	6	8	2	12	9	5
109	1	13	13	1	12	2	9	5	2	12	6	8	5	9	2	12	12	2	2	12	8	6	7	7	10	4
110	1	13	13	1	11	3	8	6	2	12	5	9	2	12	2	12	12	2	5	9	5	9	9	5	10	4
111	1	13	13	1	12	2	7	7	2	12	6	8	8	6	2	12	11	3	6	8	6	8	9	5	10	4
112	1	13	13	1	7	7	4	10	2	12	4	10	4	10	2	12	12	2	8	6	9	5	10	4	11	3

113	1	13	13	1	2	12	5	9	2	12	6	8	7	7	2	12	12	2	8	6	10	4	11	3	9	5
114	1	13	13	1	5	9	12	2	2	12	6	8	4	10	2	12	9	5	6	8	6	8	10	4	11	3
115	1	13	13	1	4	10	7	7	1	13	11	3	12	2	1	13	5	9	5	9	9	5	10	4	8	6
116	1	13	13	1	1	13	6	8	1	13	7	7	5	9	1	13	10	4	8	6	9	5	10	4	10	4
117	5	9	13	1	1	10	7	7	2	12	6	8	2	12	2	12	10	4	9	5	8	6	11	3	12	2
118	5	9	13	1	2	12	8	6	2	12	5	9	2	12	1	13	9	5	10	4	7	7	11	3	12	2
119	11	3	13	1	1	13	6	8	4	10	1	13	1	13	4	10	6	8	10	4	8	6	8	6	12	2
120	10	4	13	1	4	10	3	11	6	8	1	13	5	9	6	8	1	13	11	3	9	5	8	6	12	2
121	5	9	13	1	7	7	6	8	11	3	1	13	9	5	10	4	1	13	8	6	1	13	4	10	12	2
122	2	12	13	1	10	4	9	5	5	9	2	12	11	3	5	9	5	9	5	9	1	13	4	10	12	2
123	5	9	12	2	3	11	1	13	11	3	4	10	6	8	10	4	9	5	8	6	7	7	2	12	13	1
124	9	5	7	7	4	10	1	13	10	4	1	13	12	2	10	4	8	6	6	8	5	9	3	11	13	1
125	13	1	7	7	4	10	1	13	9	5	1	13	9	5	8	6	11	3	6	8	5	9	3	11	12	2
126	13	1	12	2	5	9	1	13	9	5	1	13	11	3	6	8	6	8	3	11	8	6	4	10	10	4
127	12	2	13	1	5	9	1	13	9	5	4	10	7	7	6	8	8	6	3	11	11	3	1	13	10	4
128	12	2	13	1	6	8	1	13	8	6	5	9	4	10	8	6	10	4	1	13	1	13	3	11	7	7
129	12	2	13	1	7	7	1	13	9	5	4	10	5	9	9	5	11	3	1	13	8	6	3	11	6	8
130	12	2	13	1	6	8	1	13	8	6	4	10	7	7	9	5	10	4	3	11	11	3	5	9	1	13
131	12	2	13	1	6	8	1	13	8	6	7	7	2	12	8	6	10	4	3	11	11	3	3	11	5	9
132	13	2	12	2	11	3	1	13	9	5	8	6	2	12	6	8	10	4	4	10	7	7	3	11	5	9
133	13	1	12	2	11	3	1	13	9	5	4	10	1	13	7	7	10	4	6	8	4	10	3	11	7	7
134	13	1	12	2	5	9	1	13	4	10	8	6	1	13	6	8	7	7	11	3	10	4	1	13	8	6
135	13	1	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	11	3	10	4	1	13	1	13
136	13	1	12	2	1	13	1	13	1	13	8	6	1	13	1	13	1	13	10	4	9	5	1	13	11	3
137	13	1	12	2	1	13	1	13	1	13	7	7	1	13	1	13	8	6	10	4	9	5	1	13	11	3
138	13	1	12	2	1	13	1	13	1	13	1	13	1	13	1	13	1	13	10	4	9	5	1	13	11	3
		1588		808		1579		1552		1613		1641		1662		1625		1512		1622		1583		1662		1514

	1588	808	1579	1552	1613	1641	1662	1625	1512	1622	1583	1662	1514
flip	7	13	9	10	6	3	**1**	4	12	5	8	1	11

287

rank	weight
1	13
2	12
3	11
4	10
5	9
6	8
7	7
8	6
9	5
10	4
11	3
12	2
13	1

References

[1] "Our World In Data," 2020. [Online]. Available: https://ourworldindata.org/cancer#deaths-from-cancer.

[2] "Global Cancer Observatory, International Agency for Research on Cancer, World Health Organization," 2020. [Online]. Available: https://gco.iarc.fr/.

[3] "Cancer.Net, Liver cancer statistics," 2020. [Online]. Available: https://www.cancer.net/cancer-types/liver-cancer/statistics.

[4] "World Health Ranking," 2020. [Online]. Available: https://www.worldlifeexpectancy.com/cause-of-death/liver-cancer/by-country/.

[5] S. V Vanmore, "Survey on Automatic Liver Segmentation Techniques from Abdominal CT Images," no. Iciccs, pp. 1030–1035, 2019.

[6] M. Moghbel, S. Mashohor, R. Mahmud, and M. I. Bin, "Review of liver segmentation and computer assisted detection / diagnosis methods in computed tomography," *Artif. Intell. Rev.*, vol. 50, no. 4, pp. 497–537, 2018.

[7] C. Science and I. Technology, "Liver Segmentation : A Survey of the State-of-the-art," pp. 4–9, 2017.

[8] S. Luo, X. Li, and J. Li, "Review on the Methods of Automatic Liver Segmentation from Abdominal Images," vol. 2014, no. January, pp. 1–7, 2014.

[9] S. Priyadarsini, "Survey on Segmentation of Liver from CT Images," no. 978, pp. 234–238, 2012.

[10] A. M. Mharib, A. Rahman, S. Mashohor, and R. Binti, "Survey on liver CT image segmentation methods," pp. 83–95, 2012.

[11] T. Heimann *et al.*, "Comparison and Evaluation of Methods for Liver Segmentation From CT Datasets," vol. 28, no. 8, pp. 1251–1265, 2009.

[12] Y. Xu *et al.*, "3D-SIFT-Flow for atlas-based CT liver image segmentation," *Med. Phys.*, vol. 43, no. 5, pp. 2229–2241, 2016.

[13] B. He *et al.*, *Fast automatic 3D liver segmentation based on a three-level AdaBoost-*

guided active shape model, vol. 43. 2016.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," pp. 1–8, 2015.

[15] P. Hu, F. Wu, J. Peng, P. Liang, and D. Kong, *Automatic 3D liver segmentation based on deep learning and globally optimized surface evolution*, vol. 61. 2016.

[16] W. Chen *et al.*, "Prostate Segmentation using 2D Bridged U-net," 2018.

[17] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. 2002.

[18] O. Marques, *Practical Image and Video Processing Using MATLAB*. 2011.

[19] D. Kaur and Y. Kaur, "Various Image Segmentation Techniques: A Review," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 5, p. 809–814, date accessed: 18/05/2016, 2014.

[20] D. L. Pham, C. Xu, and J. L. Prince, "A survey of current methods in medical image segmentation," *Annu. Rev. Biomed. Eng.*, vol. 2, pp. 315–337, 2000.

[21] G. S. Chandel, R. Kumar, D. Khare, and S. Verma, "Analysis of Image Segmentation Algorithms Using MATLAB," *Int. J. Eng. Innov. Res.*, vol. 1, no. 1, 2012.

[22] E. V. C. Chijindu, "Medical Image Segmentation Methodologies – A Classified Overview," vol. 5, no. 5, pp. 100–108, 2012.

[23] G. Läthén, *Segmentation Methods for Medical Image Analysis : Blood vessels, multi-scale filtering and level set methods*, no. 1434. 2010.

[24] A. Li, Y. Li, T. Wang, and W. Niu, "Medical image segmentation based on maximum entropy multi-threshold segmentation optimized by improved cuckoo search algorithm," *Image Signal Process. (CISP), 2015 8th Int. Congr.*, no. Cisp, pp. 470–475, 2015.

[25] C. M. Lewandowski, N. Co-investigator, and C. M. Lewandowski, *Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies - Volume 1*, vol. 1. 2015.

[26] C. M. Smith, J. Cole Smith, S. K. Williams, J. J. Rodriguez, and J. B. Hoying,

"Automatic thresholding of three-dimensional microvascular structures from confocal microscopy images," *J. Microsc.*, vol. 225, no. 3, pp. 244–257, 2007.

[27] A. Mcandrew, "An Introduction to Digital Image Processing with Matlab Notes for SCM2511 Image Processing 1," *Image Rochester NY*, vol. 1, no. 1, pp. 1–13, 2005.

[28] R. Muthukrishnan and M. Radha, "Edge Detection Techniques for Image Segmentation.," *Int. J. Comput. Sci. …*, vol. 3, pp. 259–267, 2011.

[29] S. Angelina, L. P. Suresh, and S. H. K. Veni, "Image segmentation based on genetic algorithm for region growth and region merging," *2012 Int. Conf. Comput. Electron. Electr. Technol. ICCEET 2012*, pp. 970–974, 2012.

[30] W. Wiharto and E. Suryani, "The Comparison of Clustering Algorithms K-Means and Fuzzy C-Means for Segmentation Retinal Blood Vessels," *Acta Inform. Medica*, vol. 28, no. 1, p. 42, 2020.

[31] W. G. Bradley, "History of Medical Imaging," *Proc. Am. Philos. Soc.*, vol. 152, no. 3, pp. 349–361, 2008.

[32] "Imaging Technology News ITN." [Online]. Available: http://www.itnonline.com/article/eclectic-history-medical-imaging. [Accessed: 01-Jan-2017].

[33] "World Health Organization." [Online]. Available: http://www.who.int/diagnostic_imaging/en/. [Accessed: 01-Jan-2017].

[34] P. Kršek, M. Španěl, P. Krupa, I. Marek, and P. Černochová, "Teeth and jaw 3D reconstrucion in stomatology," *Proc. - 4th Int. Conf. Med. Inf. Vis. Biomed. Vis. MediViz 2007*, no. MediViz, pp. 23–28, 2007.

[35] D. Iurp *et al.*, "7hhwk /deholqj iurp &%&7 'dwd xvlqj wkh &lufxodu +rxjk 7udqvirup," pp. 5–8, 2016.

[36] R. Peterková *et al.*, "Different morphotypes of the tabby (EDA) dentition in the mouse mandible result from a defect in the mesio-distal segmentation of dental epithelium.," *Orthod. Craniofac. Res.*, vol. 5, no. 4, pp. 215–26, 2002.

[37] P.-L. Lin and Y.-H. Lai, "An Effective Classification System for Dental Bitewing Radiographs Using Entire Tooth," *2009 WRI Glob. Congr. Intell. Syst.*, pp. 369–373, 2009.

[38] D. A. Atwood, "Postextraction changes in the adult mandible as illustrated by microradiographs of midsagittal sections and serial cephalometric roentgenograms," *J. Prosthet. Dent.*, 1963.

[39] S. Rueda, J. A. Gil, R. Pichery, and M. Alcañiz, "Automatic Segmentation of Jaw Tissues in CT Using Active Appearance Models and Semi-automatic Landmarking," *Med. Image Comput. Comput. Interv. – Miccai 2006*, pp. 167–174, 2006.

[40] C. Spampinato, C. Pino, D. Giordano, and R. Leonardi, "Automatic 3D segmentation of mandible for assessment of facial asymmetry," *MeMeA 2012 - 2012 IEEE Symp. Med. Meas. Appl. Proc.*, pp. 247–250, 2012.

[41] M. Nakao *et al.*, "Volumetric Fibular Transfer Planning with Shape-Based Indicators in Mandibular Reconstruction," *IEEE J. Biomed. Heal. Informatics*, vol. 19, no. 2, pp. 581–589, 2015.

[42] S. Liao, R. Tong, and J. Dong, "3D Human Mandible Reconstruction from CT Data *," no. 2002.

[43] M. Nakao, S. Aso, Y. Imai, and N. Ueda, "Statistical Analysis of Interactive Surgical Planning Using Shape Descriptors in Mandibular Reconstruction with Fibular Segments," vol. 63, no. 12, pp. 1–16, 2016.

[44] R. Enciso, A. Memon, and J. Mah, "Three-dimensional visualization of the craniofacial patient: volume segmentation, data integration and animation," *Orthod. Craniofac. Res.*, vol. 6 Suppl 1, no. September, pp. 66-71-182, 2003.

[45] A. Naik, S. Tikhe, S. Bhide, K. P. Kaliyamurthie, and T. Saravanan, "Algorithm to Detect Fracture from OPG Images Using Texture Analysis," *Proc. - 6th Int. Adv. Comput. Conf. IACC 2016*, vol. 382, pp. 382–385, 2016.

[46] H. P. Ng, S. H. Ong, P. S. Goh, and W. L. Nowinski, "Automatic Segmentation of

Muscles of Mastication from Magnetic Resonance Images Using Prior Knowledge," *Pattern Recognit.*, pp. 18–21, 2006.

[47] "Timetoast." [Online]. Available: https://www.timetoast.com/timelines/history-of-medical-imaging. [Accessed: 01-Jan-2017].

[48] A. Norouzi *et al.*, "Medical image segmentation methods, algorithms, and applications," *IETE Tech. Rev.*, vol. 31, no. June, pp. 199–213, 2014.

[49] "Science Learning." [Online]. Available: http://sciencelearn.org.nz/Contexts/See-through-Body/Timeline. [Accessed: 01-Jan-2017].

[50] "Open MRI of Connectcut." [Online]. Available: http://www.openmrict.com/a-brief-historical-timeline-of-medical-imaging-technology/. [Accessed: 01-Jan-2017].

[51] "US National Institute of Health." [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/24663131. [Accessed: 01-Jan-2017].

[52] M. Rastgarpour and J. Shanbehzadeh, "The Problems, Applications and Growing Interest in Automatic Segmentation of Medical Images from the Year 2000 till 2011," *Int. J. Comput. Theory Eng.*, vol. 5, no. 1, pp. 1–4, 2013.

[53] A. T. Submitted, F. O. R. The, D. Of, and D. Of, "Segmentation of the Oral and Facial Regions from Imaging Modalities with Reduced or No Ionizing Radiation JI DONG XU," 2013.

[54] V. Shrimali, R. S. Anand, and V. Kumar, "Current Trends in Segmentation of Medical Ultrasound B-mode Images: A Review," *IETE Tech. Rev.*, vol. 26, no. 1, pp. 8–17, Jan. 2009.

[55] S. S. Al-amri, N. V. Kalyankar, and S. D. Khamitkar, "A Comparative Study of Removal Noise from Remote Sensing Image," vol. 7, no. 1, pp. 32–36, 2010.

[56] A. A. Abubaker, R. S. Qahwaji, M. J. Aqel, and M. H. Saleh, "Average Row Thresholding Method for Mammogram Segmentation," pp. 3288–3291, 2005.

[57] J. Li, S. Zhu, and H. Bin, "Medical image segmentation techniques," *Shengwu Yixue Gongchengxue Zazhi/Journal Biomed. Eng.*, vol. 23, no. 4, pp. 891–894, Aug. 2006.

[58] C. H. Bindu, K. S. Prasad, and A. Pradesh, "An Efficient Medical Image Segmentation Using Conventional OTSU Method An Efficient Medical Image Segmentation Using Conventional OTSU Method ECE Department , QIS College of Engineering & Technology ," no. January 2012, 2014.

[59] Y. Feng, H. Zhao, X. Li, X. Zhang, and H. Li, "A multi-scale 3D Otsu thresholding algorithm for medical image segmentation," *Digit. Signal Process.*, vol. 60, pp. 186–199, 2017.

[60] W. Haider and M. Raza, "Achieving Accuracy in Early Stage Tumor Identification Systems based on Image Segmentation and 3D Structure Analysis Achieving Accuracy in Early Stage Tumor Identification Systems based on Image Segmentation and 3D Structure Analysis," no. January, 2011.

[61] J. Shan, H. D. Cheng, and Y. Wang, "A completely automatic segmentation method for breast ultrasound images using region growing," 2006.

[62] H. Guan, D. Li, J. Lin, and T. Wang, "Segmentation of Ultrasound Medical Image Using A Hybrid Method," in *2007 IEEE/ICME International Conference on Complex Medical Engineering*, 2007, pp. 644–647.

[63] N. Senthilkumaran and R. Rajesh, "Edge Detection Techniques for Image Segmentation – A Survey of Soft Computing Approaches," *Int. J. Recent Trends Eng.*, no. May 2014, 2007.

[64] R. Hegadi, "A Survey on Deformable Model and its Applications to Medical Imaging A Survey on Deformable Model and its Applications to Medical Imaging," *IJCA*, no. January, 2010.

[65] C. Xu and J. L. Prince, "Image Segmentation Using Deformable Models," in *Image Segmentation Using Deformable Models*, 2000, pp. 129–174.

[66] P. Yan and A. A. Kassim, "Medical Image Segmentation Using Minimal Path Deformable Models With Implicit Shape Priors," *IEEE Trans. Inf. Technol. Biomed.*, no. November, 2006.

[67] X. Chen, S. Member, and L. Pan, "A Survey of Graph Cuts / Graph Search Based Medical Image Segmentation," *IEEE Rev. Biomed. Eng.*, vol. 11, pp. 112–124, 2018.

[68] K. Fritscher, T. Gmbh, M. Nemoto, Y. Masutani, and N. Hayashi, "3-D Graph Cut Segmentation with Riemannian Metrics to Avoid the Shrinking 3-D Graph Cut Segmentation with Riemannian Metrics to Avoid the Shrinking Problem," in *Medical Image Computing and Computer Assisted Interven- tion { MICCAI 2011, Lecture Notes in Computer Science, vol. 6893, 554{561, Springer, Berlin, 2011.*, 2011, no. December 2013.

[69] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1240–1251, 2016.

[70] L. E. F. Ang *et al.*, "Automatic segmentation of nine retinal layer boundaries in OCT images of non-exudative AMD patients using deep learning and graph search," *Biomed. Opt. EXPRESS 2732*, vol. 8, no. 5, pp. 2732–2744, 2017.

[71] X. Sui, Y. Zheng, B. Wei, H. Bi, and J. Wu, "Neurocomputing Choroid segmentation from Optical Coherence Tomography with graph- edge weights learned from deep convolutional neural networks," *Neurocomputing*, vol. 237, no. December 2016, pp. 332–341, 2017.

[72] F. Lu, F. Wu, P. Hu, Z. Peng, and D. Kong, "Automatic 3D liver location and segmentation via convolutional neural network and graph cut," pp. 171–182, 2017.

[73] Z. Mishra, A. Ganegoda, J. Selicha, Z. Wang, S. R. Sadda, and Z. Hu, "Automated Retinal Layer Segmentation Using Graph-based Algorithm Incorporating Deep-learning-derived Information," *www.nature.com/scientificreports*, pp. 1–8, 2020.

[74] D. Vanisri, "RESEARCH ARTICLE A Novel Kernel Based Fuzzy C Means Clustering With Cluster Validity Measures," vol. 3, no. 12, pp. 254–260, 2014.

[75] T. H. Lee, M. Faizal, A. Fauzi, and R. Komiya, "Visualisation Segmentation of CT Brain Images Using K-means and EM Clustering Faculty of Engineering , (I (i , j) − I L) (

I U − I L),” pp. 339–344, 2008.

[76] T. Z. T. Muda and R. Abdul, “Blood Cell Image Segmentation using Hybrid K-means and Median-cut Algorithms,” pp. 237–243, 2011.

[77] K. Kasiri, K. Kazemi, M. J. Dehghani, and M. S. Helfroush, “Atlas-based Segmentation of Brain MR Images Using Least Square Support Vector Machines,” *Image Process. Theory, Tools Appl.*, pp. 0–4, 2010.

[78] S. Egmentation, D. L. Pham, C. Xu, and J. L. Prince, “CURRENT METHODS IN MEDICAL IMAGE SEGMENTATION,” 2000.

[79] T. F. Cootes, “Statistical Models of Appearance for Computer Vision,” *Direct*, vol. M, pp. 1–124, 2004.

[80] T. Heimann and H. P. Meinzer, “Statistical shape models for 3D medical image segmentation: A review,” *Med. Image Anal.*, vol. 13, no. 4, pp. 543–563, 2009.

[81] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active Shape Models-Their Training and Application,” *Comput. Vis. Image Underst.*, vol. 61, pp. 38–59, 1995.

[82] D. J. Kroon, *Segmentation of the mandibular canal in cone-beam CT data*. 2011.

[83] S. T. Gollmer and T. M. Buzug, “Fully automatic shape constrained mandible segmentation from cone-beam CT data,” *Proc. - Int. Symp. Biomed. Imaging*, pp. 1272–1275, 2012.

[84] M. Brandariz, N. Barreira, M. G. Penedo, and M. Suárez-Cunqueiro, “Automatic segmentation of the mandible in Cone-Beam Computer Tomography images,” *Proc. - IEEE Symp. Comput. Med. Syst.*, pp. 467–468, 2014.

[85] T. Albrecht, T. Gass, C. Langguth, and L. Marcel, “Multi Atlas Segmentation with Active Shape Model Refinement for Multi-Organ Segmentation in Head and Neck Cancer Radiotherapy Planning,” pp. 1–6.

[86] K. Y. E. Leung, M. Van Stralen, G. Van Burken, N. De Jong, and J. G. Bosch, “Automatic active appearance model segmentation of 3D echocardiograms,” *2010 7th IEEE Int. Symp. Biomed. Imaging From Nano to Macro, ISBI 2010 - Proc.*, pp. 320–

323, 2010.

[87] G. Tzimiropoulos and M. Pantic, "Optimization problems for fast AAM fitting in-the-wild," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 593–600, 2013.

[88] S. C. Mitchell, J. G. Bosch, B. P. F. Lelieveldt, R. J. Van der Geest, J. H. C. Reiber, and M. Sonka, "3-D active appearance models: Segmentation of cardiac MR and ultrasound images," *IEEE Trans. Med. Imaging*, vol. 21, no. 9, pp. 1167–1178, 2002.

[89] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Proc. Eur. vonference Comput. Vis.*, vol. 23, no. 6, pp. 484–498, 1998.

[90] T. F. Cootes, G. Edwards, and C. J. Taylor, "A Comparative Evaluation of Active Appearance Model Algorithms," *Procedings Br. Mach. Vis. Conf. 1998*, p. 68.1-68.10, 1998.

[91] G. J. Edwards, C. J. Taylor, and T. F. Cootes, "Interpreting face images using active appearance models," *Autom. Face Gesture Recognit.*, pp. 300–305, 1998.

[92] R. Beichel, H. Bischof, F. Leberl, and M. Sonka, "Robust active appearance models and their application to medical image analysis," *IEEE Trans. Med. Imaging*, vol. 24, no. 9, pp. 1151–1169, 2005.

[93] K. Babalola and T. Cootes, "AAM Segmentation of the Mandible and Brainstem," *MIDAS J. - Head Neck Auto-Segmentation Challenge.*, 2009.

[94] R. Mannion-haworth, M. Bowes, A. Ashman, G. Guillard, A. Brett, and G. Vincent, "Fully Automatic Segmentation of Head and Neck Organs using Active Appearance Models," vol. im.

[95] X. Han *et al.*, "Atlas-based auto-segmentation of head and neck CT images," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5242 LNCS, pp. 434–441, 2008.

[96] M. Lorenzo-Vald??s, G. I. Sanchez-Ortiz, A. G. Elkington, R. H. Mohiaddin, and D. Rueckert, "Segmentation of 4D cardiac MR images using a probabilistic atlas and the EM algorithm," *Med. Image Anal.*, vol. 8, no. 3, pp. 255–265, 2004.

[97] M. E. Hartnett, N. Tinkham, L. Paynter, P. Geisen, G. Koch, and K. L. Cohen, "NIH Public Access," vol. 148, no. 6, pp. 895–901, 2010.

[98] X. Han, L. Hibbard, and N. O'Connell, "Automatic segmentation of head and neck CT images by GPU-accelerated multi-atlas fusion," *3D Segmentation*, 2009.

[99] K. D. Fritscher, M. Peroni, P. Zaffino, M. F. Spadea, R. Schubert, and G. Sharp, "Automatic segmentation of head and neck CT images for radiotherapy treatment planning using multiple atlases, statistical appearance models, and geodesic active contours," *Med. Phys.*, vol. 41, no. 5, p. 51910, 2014.

[100] X. Zhang, J. Tian, Y. Wu, J. Zheng, and K. Deng, "Segmentation of Head and Neck CT Scans Using Atlas-based Level Set Method," pp. 1–9, 2009.

[101] X. Han, L. S. Hibbard, N. P. O'Connell, and V. Willcut, "Automatic Segmentation of Parotids in Head and Neck CT Images using Multi-atlas Fusion," *Med. Image Anal. Clin. A Gd. Chall.*, pp. 297–304, 2010.

[102] V. Fortunati *et al.*, "Hyperthermia critical tissues automatic segmentation of head and neck CT images using atlas registration and graph cuts," *Proc. - Int. Symp. Biomed. Imaging*, pp. 1683–1686, 2012.

[103] A. Chen and B. M. Dawant, "A Multi-atlas Approach for the Automatic Segmentation of Multiple Structures in Head and Neck CT Images."

[104] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," *Nat. Methods*, vol. 13, no. 1, pp. 35–35, 2015.

[105] Ian Goodfellow and Yoshua Bengio and Aaron Courville, "Deep Learning Book," *MIT Press*, 2016. [Online]. Available: http://www.deeplearningbook.org/.

[106] Y. Lecun, "Lecun-98," 1998.

[107] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[108] G. Litjens *et al.*, "A Survey on Deep Learning in Medical Image Analysis," no. 1995, 2017.

[109] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12–June, pp. 1–9, 2015.

[110] M. Lin, Q. Chen, and S. Yan, "Network In Network," pp. 1–10, 2013.

[111] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimed. Tools Appl.*, pp. 1–17, 2017.

[112] Y. Liu *et al.*, "Detecting Cancer Metastases on Gigapixel Pathology Images," pp. 1–13, 2017.

[113] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, p. 115, Jan. 2017.

[114] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA - J. Am. Med. Assoc.*, vol. 316, no. 22, pp. 2402–2410, 2016.

[115] E. Hosseini-Asl, G. Gimel'farb, and A. El-Baz, "Alzheimer's Disease Diagnostics by a Deeply Supervised Adaptable 3D Convolutional Network," no. 502, 2016.

[116] A. Payan and G. Montana, "Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks," pp. 1–9, 2015.

[117] J. Kawahara *et al.*, "BrainNetCNN: Convolutional neural networks for brain networks; towards predicting neurodevelopment," *Neuroimage*, vol. 146, no. September 2016, pp. 1038–1049, 2017.

[118] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," 2014.

[119] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," *Nips*, pp. 1–9, 2012.

[120] R. Korez, B. Likar, F. Pernuš, and T. Vrtovec, *Model-Based Segmentation of Vertebral Bodies from MR Images with 3D CNNs*. 2016.

[121] P. Moeskops *et al.*, "Deep Learning for Multi-Task Medical Image Segmentation in

Multiple Modalities," no. October, pp. 478–486, 2017.

[122] M. Shakeri *et al.*, "Sub-cortical brain structure segmentation using F-CNN's," 2016.

[123] Y. Song, L. Zhang, S. Chen, D. Ni, B. Lei, and T. Wang, "Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 10, pp. 2421–2433, 2015.

[124] Y. Guo, Y. Gao, and D. Shen, "Deformable MR Prostate Segmentation via Deep Feature Learning and Sparse Patch Matching," *Deep Learn. Med. Image Anal.*, vol. 35, no. 4, pp. 197–222, 2017.

[125] Y. Gordienko, P. Gang, J. Hui, W. Zeng, Y. Kochura, and O. Alienin, "Deep Learning with Lung Segmentation and Bone Shadow Exclusion Techniques for Chest X-Ray Analysis of Lung Cancer."

[126] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen, "Combining Fully Convolutional and Recurrent Neural Networks for 3D Biomedical Image Segmentation," *arXiv:1609.01006*, no. Ii, pp. 3036–3044, 2016.

[127] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, "Deep Learning Techniques for Medical Image Segmentation : Achievements and Challenges," vol. 3, 2019.

[128] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-net: Learning dense volumetric segmentation from sparse annotation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9901 LNCS, pp. 424–432, 2016.

[129] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," pp. 1–11, 2016.

[130] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10008 LNCS, pp. 179–187, 2016.

[131] K. Kamnitsas *et al.*, "Efficient multi-scale 3D CNN with fully connected CRF for

accurate brain lesion segmentation," *Med. Image Anal.*, vol. 36, pp. 61–78, 2017.

[132] C. Wang *et al.*, "A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks," *2015 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, pp. 2415–2418, 2015.

[133] T. Brosch, L. Y. W. Tang, Y. Yoo, D. K. B. Li, A. Traboulsee, and R. Tam, "Deep 3D Convolutional Encoder Networks With Shortcuts for Multiscale Feature Integration Applied to Multiple Sclerosis Lesion Segmentation," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1229–1239, 2016.

[134] Y. Xie, Z. Zhang, M. Sapkota, and L. Yang, "Spatial Clockwork Recurrent Neural Network for Muscle Perimysium Segmentation," *Med. Image Comput. Comput. Assist. Interv.*, vol. 9901, pp. 185–193, 2016.

[135] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber, "Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation," vol. di, pp. 1–9, 2015.

[136] P. Doetsch, M. Kozielski, and H. Ney, "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition," *Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR*, vol. 2014–Decem, pp. 279–284, 2014.

[137] R. P. K. Poudel, P. Lamata, and G. Montana, "Recurrent fully convolutional neural networks for multi-slice MRI cardiac segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10129 LNCS, pp. 83–94, 2017.

[138] F. Milletari, N. Navab, and S. A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," *Proc. - 2016 4th Int. Conf. 3D Vision, 3DV 2016*, pp. 565–571, 2016.

[139] X. Xia and B. Kulis, "W-Net: A Deep Model for Fully Unsupervised Image Segmentation," 2017.

[140] M. Jun, "Segmentation Loss Odyssey," *arXiv Prepr. arXiv2005.13449*, 2020.

[141] S. Jadon, "A survey of loss functions for semantic segmentation," 2020.

[142] M. A. Vi-de and L. I. U. Qing, "Automated Image Segmentation Using Improved," pp. 743–746, 2004.

[143] V. Pihur, S. Datta, and S. Datta, "Gene expression Weighted rank aggregation of cluster validation measures : a Monte Carlo cross-entropy approach," vol. 23, no. 13, pp. 1607–1615, 2007.

[144] Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function : Modeling the Costs of Mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[145] S. Xie, "Holistically-Nested Edge Detection University of California , San Diego," 2015.

[146] T. Lin, F. Ai, and P. Doll, "Focal Loss for Dense Object Detection," 2018.

[147] F. Caliv, C. Iriondo, and A. M. Martinez, "Distance Map Loss Penalty Term for Semantic Segmentation," no. 1, pp. 1–5, 2019.

[148] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, M. J. Cardoso, and T. I. Group, "Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations," pp. 1–8, 2017.

[149] S. Sadegh, M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3D fully convolutional deep networks," 2017.

[150] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," 2018.

[151] I. SEYED RAEIN HASHEMI, SEYED SADEGH MOHSENI SALEHI[Student Member, S. K. W. DENIZ ERDOGMUS [Senior Member, IEEE], SANJAY P. PRABHU1, and I. [Senior Member, IEEE], ALI GHOLIPOUR1 [Senior Member, "Asymmetric Loss Functions and Deep Densely Connected Networks for Highly Imbalanced Medical Image Segmenta," no. September 2018, pp. 1–31, 2020.

[152] M. Berman, A. Rannen, and T. Matthew, "The Lov ´ asz-Softmax loss : A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," pp. 4413–4421, 2018.

[153] Hayder Zeeshan;Xuming He; Mathieu Salzmann, "Shape-aware Instance Segmentation," no. December, 2016.

[154] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware Instance Segmentation," 2017.

[155] J. Ribera, D. Güera, Y. Chen, and E. Delp, "Weighted Hausdorff Distance: A Loss Function For Object Localization," *ArXiv*, vol. abs/1806.0, 2018.

[156] J. Ribera, G. David, Y. Chen, and E. J. Delp, "Locating Objects Without Bounding Boxes," 2019.

[157] D. Karimi and S. E. Salcudean, "Reducing the Hausdorff Distance in Medical Image Segmentation With Convolutional Neural Networks," vol. 39, no. 2, pp. 499–513, 2020.

[158] S. Asgari *et al.*, "Computerized Medical Imaging and Graphics Combo loss : Handling input and output imbalance in multi-organ segmentation," *Comput. Med. Imaging Graph.*, vol. 75, pp. 24–33, 2019.

[159] K. C. L. Wong, M. Moradi, H. Tang, and T. Syeda-mahmood, "3D Segmentation with Exponential Logarithmic Loss for Highly Unbalanced Object Sizes," vol. 2018, 2018.

[160] S. Zhao, B. Wu, W. Chu, Y. Hu, and D. Cai, "Correlation Maximized Structural Similarity Loss for Semantic Segmentation," 2019.

[161] "Optimizers for Convolutional neural networks," 2020. [Online]. Available: https://keras.io/api/optimizers/.

[162] J. Antony, K. McGuinness, N. E. O. Connor, and K. Moran, "Quantifying Radiographic Knee Osteoarthritis Severity using Deep Convolutional Neural Networks," 2016.

[163] E. Kim, M. Corte-Real, and Z. Baloch, *A deep semantic mobile application for thyroid cytopathology*. 2016.

[164] A. Menegola, M. Fornaciali, R. Pires, S. Avila, and E. Valle, "Towards Automated Melanoma Screening: Exploring Transfer Learning Schemes," no. Ic, pp. 1–4, 2016.

[165] S. Urbán, L. Ruskó, and A. Nagy, "A Self-learning Tumor Segmentation Method on DCE-MRI Images BT - Image Analysis and Recognition," 2016, pp. 591–598.

[166] J. Delmoral, "Deep learning methods for multimodal segmentation: Fusing nuclear and structural medical image," 2016.

[167] K. A. Saddi and C. Chefd, "Global-to-Local Shape Matching for Liver Segmentation in CT Imaging," *MICCAI (October 2007)*, pp. 207–214, 2007.

[168] L. Rusko and G. Bekes, "Fully automatic liver segmentation for contrast-enhanced CT images," *Int. Conf. Med. Image Comput. Comput. Interv. Segmentation Clin. a Gd. challenge, 2007*, pp. 143–150, 2007.

[169] G. Schmidt, M. Athelogou, R. Schönmeyer, R. Korn, and G. Binnig, "Cognition network technology for a fully automated 3d segmentation of liver," *MICCAI Work. 3D Segmentation Clin. A Gd. Chall.*, pp. 125–133, 2007.

[170] R. Susomboon, "A hybrid approach for liver segmentation," *… Segmentation Clin. …*, vol. i, pp. 151–160, 2007.

[171] Y. Chi, P. M. M. Cashman, O. Bello, and R. I. Kitney, "A Discussion on the Evaluation of A New Automatic Liver Volume Segmentation Method for Specified CT Image Datasets." .

[172] D. Seghers *et al.*, "Landmark based liver segmentation using local shape and local intensity models," *Proc. Work. 10th Int. Conf. MICCAI, Work. 3D Segmentation Clin. A Gd. Chall.*, pp. 135–142, 2007.

[173] D. Kainmüller, T. Lange, and H. Lamecker, "Shape constrained automatic segmentation of the liver based on a heuristic intensity model," *MICCAI Work. 3D Segmentation Clin. A Gd. Chall.*, pp. 109–16, 2007.

[174] T. Heimann, H. Meinzer, and I. Wolf, "A Statistical Deformable Model for the Segmentation of Liver CT Volumes Using Extended Training Data," *Proc. MICCAI Work. 3-D Segmentat. Clin. A Gd. Challenge,* p. 161–166., 2007.

[175] D. Furukawa, S. Akinobu, and H. Kobakate, "Automatic liver segmentation method based on maximum a posterior probability estimation and level set method," *Proc. MICCAI Work. 3-D Segmentat. Clin. A Gd. Chall.*, pp. 117–124, 2007.

[176] J. Lee *et al.*, "Efficient Liver Segmentation exploiting Level-Set Speed Images with 2 .5D Shape Propagation," *3D Segmentation Clin. A Gd. Chall.*, pp. 189–196, 2007.

[177] E. Van Rikxoort, Y. Arzhaeva, and B. Van Ginneken, "Automatic segmentation of the liver in computed tomography scans with voxel classification and atlas matching," *Proc. MICCAI …*, pp. 101–108, 2007.

[178] R. Beichel, C. Bauer, A. Bornik, E. Sorantin, and H. Bischof, "Liver Segmentation in CT Data : A Segmentation Refinement Approach," *Methods*, no. c, pp. 235–245, 2007.

[179] A. Beck and V. Aurich, "Hepatux–a semiautomatic liver segmentation system," *3D Segmentation Clin. A Gd. Chall.*, pp. 225–233, 2007.

[180] B. M. Dawant, R. Li, B. Lennon, and S. Li, "Semi-automatic segmentation of the liver and its evaluation on the MICCAI 2007 grand challenge data set," *3D Segmentation Clin. A Gd. Chall.*, pp. 215–221, 2007.

[181] A. Wimmer, G. Soza, and J. Hornegger, "Two-stage semi-automatic organ segmentation framework using radial basis functions and level sets," *3D segmentation Clin. a Gd. challenge, LNCS, Springer Berlin, Heidelb.*, pp. 207–214, 2007.

[182] P. Slagmolen, A. Elen, D. Seghers, D. Loeckx, F. Maes, and K. Haustermans, "Atlas based liver segmentation using nonrigid registration with a B-spline transformation model," *Proc. MICCAI Work. 3D segmentation Clin. a Gd. Chall.*, pp. 197–206, 2007.

[183] X. Zhang, J. Tian, K. Deng, Y. Wu, and X. Li, "Automatic liver segmentation using a statistical shape model with optimal surface detection," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 10 PART 2, pp. 2622–2626, 2010.

[184] A. Wimmer, G. Soza, and J. Hornegger, "Organ Segmentation," *Image (Rochester, N.Y.)*, pp. 26–33, 2009.

[185] X. Chen, J. K. Udupa, U. Bagci, Y. Zhuge, and J. Yao, "Medical image segmentation by combining graph cuts and oriented active appearance models.," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2035–46, 2012.

[186] L. Massoptier and S. Casciaro, "A new fully automatic and robust algorithm for fast segmentation of liver tissue and tumors from CT scans," *Eur. Radiol.*, vol. 18, pp. 1658–1665, 2008.

[187] L. Ruskó, G. Bekes, and M. Fidrich, "Automatic segmentation of the liver from multi- and single-phase contrast-enhanced CT images," *Med. Image Anal.*, vol. 13, no. 6, pp. 871–882, 2009.

[188] S. Tomoshige, E. Oost, A. Shimizu, H. Watanabe, and S. Nawano, "A conditional statistical shape model with integrated error estimation of the conditions; Application to liver segmentation in non-contrast CT images," *Med. Image Anal.*, vol. 18, no. 1, pp. 130–143, 2014.

[189] L. Grady, "Random walks for image segmentation," *Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.

[190] T. Okada *et al.*, "Automated Segmentation of the Liver from 3D CT Images Using Probabilistic Atlas and Multilevel Statistical Shape Model," *Acad. Radiol.*, vol. 15, no. 11, pp. 1390–1403, 2008.

[191] C. Platero and M. C. Tobar, "A multiatlas segmentation using graph cuts with applications to liver segmentation in CT scans," *Comput. Math. Methods Med.*, vol. 2014, 2014.

[192] J. Stawiaski *et al.*, "Interactive Liver Tumor Segmentation Using Graph-cuts and Watershed To cite this version : HAL Id : hal-01445751 Graph-cuts and Watershed," 2017.

[193] D. Wong *et al.*, "A semi-automated method for liver tumor segmentation based on 2D region growing with knowledge-based constraints," *Midas J.*, no. January 2008, 2008.

[194] J. H. Moltz, L. Bornemann, V. Dicken, and H.-O. Peitgen, "Segmentation of liver metastases in CT scans by adaptive thresholding and morphological processing," *MICCAI Work.*, vol. 41, no. 43, p. 195, 2008.

[195] A. Choudhary and N. Moretto, "An entropy based multi-thresholding method for semi-

automatic segmentation of liver tumors," *Miccai …*, no. January 2008, pp. 1–12, 2008.

[196] T. Kubota, "Efficient automated detection and segmentation of medium and large liver tumors: CAD approach," *MICCAI Work.*, 2008.

[197] H. Nugroho, "Contrast enhancement for liver tumor identification," *… Liver Tumor …*, no. May 2014, 2008.

[198] Y. Häme, "Liver Tumor Segmentation Using Implicit Surface Evolution," *Liver*, pp. 1–10, 2008.

[199] M. Goryawala, S. Gulec, R. Bhatt, A. J. McGoron, and M. Adjouadi, "A low-interaction automatic 3D liver segmentation method using computed tomography for selective internal radiation therapy," *Biomed Res. Int.*, vol. 2014, no. 1, 2014.

[200] A. Shimizu and T. Narihira, "Ensemble segmentation using AdaBoost with application to liver lesion extraction from a CT volume," *… Segmentation (…*, no. May 2014, 2008.

[201] J. Zhou *et al.*, "Semi-automatic segmentation of 3D liver tumors from CT scans using voxel classification and propagational learning," *2008 MICCAI Work. Gd. Chall. Liver Tumor Segmentation*, vol. d, no. May, 2008.

[202] X. Zhang, J. Tian, D. Xiang, X. Li, and K. Deng, "Interactive liver tumor segmentation from ct scans using support vector classification with watershed.," *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2011, pp. 6005–8, 2011.

[203] W. Wu, Z. Zhou, S. Wu, and Y. Zhang, "Automatic Liver Segmentation on Volumetric CT Images Using Supervoxel-Based Graph Cuts Automatic Liver Segmentation on Volumetric CT Images Using Supervoxel-Based Graph Cuts," *Comput. Math. Methods Med.*, vol. 1, no. 4, pp. 1–14, 2016.

[204] M. A. Selver, A. Kocaoğlu, G. K. Demir, H. Doğan, O. Dicle, and C. Güzeliş, "Patient oriented and robust automatic liver segmentation for pre-evaluation of liver transplantation," *Comput. Biol. Med.*, vol. 38, no. 7, pp. 765–784, 2008.

[205] M. Suhuai Luo; Qingmao Hu; Xiangjian He; Jiaming Li; Jin, J.S.; Park, "Automatic liver parenchyma segmentation from abdominal CT images using support vector machines," *Complex Med. Eng. 2009. C. ICME Int. Conf.*, 2009.

[206] Y. Zheng *et al.*, "Feature learning based random walk for liver segmentation," *PLoS One*, vol. 11, no. 11, pp. 1–17, 2016.

[207] F. Lu, F. Wu, P. Hu, Z. Peng, and D. Kong, "Automatic 3D liver location and segmentation via convolutional neural networks and graph cut," 2016.

[208] L. Bi, J. Kim, A. Kumar, and D. Feng, "Automatic Liver Lesion Detection using Cascaded Deep Residual Networks," 2017.

[209] L. Zhang and L. Xu, "An Automatic Liver Segmentation Algorithm for CT Images U-Net with Separated Paths of Feature Extraction," *2018 3rd IEEE Int. Conf. Image, Vis. Comput. ICIVC 2018*, pp. 294–298, 2018.

[210] H. Seo, C. Huang, M. Bassenne, R. Xiao, and L. Xing, "Modified U-Net (mU-Net) with incorporation of object-dependent high level features for improved liver and liver-tumor segmentation in CT images," *arXiv*, vol. 39, no. 5, pp. 1316–1325, 2019.

[211] I. Astono, J. S. Welsh, and S. Chalup, "Adjacent network for semantic segmentation of liver CT scans," *Proc. - 2018 IEEE 18th Int. Conf. Bioinforma. Bioeng. BIBE 2018*, pp. 35–40, 2018.

[212] K. Saito, H. Lu, H. Kim, S. Kido, and M. Tanabe, "ROI-based fully automated liver registration in multi-phase CT images," *Int. Conf. Control. Autom. Syst.*, vol. 2018–October, no. ICCAS, pp. 645–649, 2018.

[213] H. S. Hoang, C. Phuong Pham, D. Franklin, T. Van Walsum, and M. Ha Luu, "An Evaluation of CNN-based Liver Segmentation Methods using Multi-types of CT Abdominal Images from Multiple Medical Centers," *Proc. - 2019 19th Int. Symp. Commun. Inf. Technol. Isc. 2019*, pp. 20–25, 2019.

[214] A. A. Albishri, S. J. H. Shah, and Y. Lee, "CU-Net: Cascaded U-Net Model for Automated Liver and Lesion Segmentation and Summarization," *Proc. - 2019 IEEE*

Int. Conf. Bioinforma. Biomed. BIBM 2019, pp. 1416–1423, 2019.

[215] Y. Liu, N. Qi, Q. Zhu, and W. Li, "Cr-u-net: Cascaded u-net with residual mapping for liver segmentation in ct images," *2019 IEEE Int. Conf. Vis. Commun. Image Process. VCIP 2019*, pp. 6–9, 2019.

[216] Z. H. Wang, Z. Liu, Y. Q. Song, and Y. Zhu, "Densely connected deep U-Net for abdominal multi-organ segmentation," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2019–September, pp. 1415–1419, 2019.

[217] W. Xu, H. Liu, X. Wang, and Y. Qian, "Liver segmentation in CT based on ResUNet with 3D Probabilistic and Geometric Post Process," *2019 IEEE 4th Int. Conf. Signal Image Process. ICSIP 2019*, pp. 685–689, 2019.

[218] Y. C. Liu, D. S. Tan, J. C. Chen, W. H. Cheng, and K. L. Hua, "Segmenting Hepatic Lesions Using Residual Attention U-Net with an Adaptive Weighted Dice Loss," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2019–September, pp. 3322–3326, 2019.

[219] V. Giannini *et al.*, "Deep learning to segment liver metastases on CT images: Impact on a radiomics method to predict response to chemotherapy," *IEEE Med. Meas. Appl. MeMeA 2020 - Conf. Proc.*, 2020.

[220] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: Redesigning skip connections to exploit multiscale features in image segmentation," *arXiv*, vol. 39, no. 6, pp. 1856–1867, 2019.

[221] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019.

[222] R. Gu, Shanqing; Pednekar, Manisha; and Slater, "Improve Image Classification Using Data Augmentation and Neural Networks," *SMU Data Sci. Rev.*, vol. 2, no. 2, pp. 1–43, 2019.

[223] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Sci. Rep.*, vol. 10, no. 1, pp. 1–7, 2020.

[224] "3D-IRCADb (3D Image Reconstruction for Comparison of Algorithm Database)." [Online]. Available: https://www.ircad.fr/research/3dircadb/.

[225] "ISBI challenge 2015," 2015. [Online]. Available: http://brainiac2.mit.edu/isbi_challenge/.

[226] "Computer-Automated Detection of Caries in Bitewing Radiography," 2015. [Online]. Available: http://www-o.ntust.edu.tw/~cweiwang/ISBI2015/challenge2/index.html.

[227] "U-Net Achivements," 2015. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/.

[228] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," pp. 1–12.