University of Nottingham
UK | CHINA | MALAYSIA

Capsule-based Image Translation and Quality Analysis

Fei Yang

A thesis submitted to the school of computer science
for the degree of Doctor of Philosophy
University of Nottingham

July 2021

To My Family

Abstract

The research community has witnessed a great success of computer vision for past decades, benefiting from the rapid development of deep learning technologies. Among a great number of research topics, image translation plays an important role in computer vision, which aims to synthesize image B conditioning on image A. The translation of A to B is achieved by well-designed generative models, which takes image A as the input and generates image B. It is regarded as domain translation or style translation, such as horse-to-zebra, image rendering, noise removal and etc.

The development of deep learning boosts the research of image translation. Advanced network structures have been designed as the translator of models. Effective objective functions have been proposed to supervise the generation of images. Advanced training strategies have been explored to optimise the training procedure. Theses techniques bring improvements to translating models in generating realistic images.

However, the research of image translation still has a long way to go. The applicable scenarios of image translation are diverse with a great number of types. The source domain and the target domain can be defined arbitrarily, which means a horse can be translated into any another object as we want or a painting can be translated to any other style of image as we define. It is hard for one general model to process various translating characteristics. Besides this, existing translation models cannot completely avoid noisy marks that are introduced by the convolution kernels during the translation process.

My contributions are summarised as follow. 1) To push forward the research of image translation, algorithms are developed to enhance the model

performance. A capsule-based framework is built on the structure of image-conditioned generative adversarial network, in which the capsule units are responsible for improving the ability of learning part-to-whole relationship and strengthening the feature learning ability in a global view. 2) Multiple applications are discussed in this report, among which image rendering is a new one and others such as de-raining, de-snowing and de-hazing are traditional noise removal topics. Considering the specific characteristics of each application, various techniques are proposed. A preservation loss is proposed for image rendering. A two-branch structure with a rain component loss is designed for de-raining. A multi-scale structure is proposed for de-snowing. A depth encoding method is developed for de-hazing. 3) Image quality has much correlation with the model performance, especially when assessing the quality of translated images. The ways of estimating the quality levels are explored in this report. Based on the level estimation, a task-oriented image quality assessment method is developed to calculate quality scores.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. Qian Zhang and Prof. Guoping Qiu, for their essential guidance, advice and support throughout my PhD study life. As my main supervisor, Dr. Qian Zhang had great patience to lead me to the final PhD. She gave me much help on examining the experiments, revising the papers, and finding ways to overcome the difficulties. She encouraged me all the time to go through the hard time and lighted me the road ahead.

I would also like to thank to Dr. Zheng Lu and Dr. Jianfeng Ren for the great discussions and the support to my work. They gave me many helpful suggestions in conducting the experiments and taught me how to develop a research idea into a complete work. I would like to thank Prof. Linlin Shen, who guided me the road to begin a PhD journey. I would like to thank Prof. Ruibin Bai, our school head, who believes in me and gave me much support.

Thanks to other members of the Artificial Intelligence Lab of Nottingham Ningbo for great discussions of the research and the enjoyable times we shared.

I would like to thank to my family for their great support to my PhD. My parents give me help unreservedly and support me all the time. My young brother, helps me a lot to overcome the difficulties. Thanks to my girl friend, Miss Wang, who believes me all the time and brightens my life. Without these, it is impossible for me to get through the hard time of the journey.

List of Publications

Conference papers for this thesis:

1. **Fei Yang**, Zheng Lu, Guoping Qiu, Jing Lin and Qian Zhang. Capsule based image synthesis for interior design effect rendering. *14th Asian Conference on Computer Vision (ACCV)*, pages 1830-198, 2018. (Chapter 4)

2. **Fei Yang**, Qian Zhang, Miaohui Wang and Guoping Qiu. Quality classified image analysis with application to face detection and recognition. *24th International Conference on Pattern Recognition (ICPR)*, pages 2863-2868, 2018. (Chapter 8)

3. **Fei Yang**, Zheng Lu, Guoping Qiu and Qian Zhang. Capsule based image translation network. *IET Doctoral Forum on Biomedical Engineering, Healthcare, Robotics and Artificial Intelligence(BRAIN)*, pages 3-8, 2018. (Chapter 3)

Journal papers for this thesis:

1. **Fei Yang**, Jialu Zhang and Qian Zhang. Multi-scale capsule generative adversarial network for snow removal. *IET Computer Vision*, pages 1-9, 2020. (Chapter 7)

2. **Fei Yang** and Qian Zhang. Depth aware image dehazing. *The Visual Computer*, pages 1-9, 2021. (Chapter 6)

Manuscripts under peer review for this thesis:

1. Rain-component-aware capsule-GAN for single image de-raining, *Pattern Recognition*, 2021. (Chapter 5)

2. Capsule based image synthesis for interior design effect rendering. (Chapter 4)

3. Task-oriented image quality assessment for synthesised images. (Chapter 9)

Papers with related ideas but not included in this thesis

1. **Fei Yang**, Qian Zhang, Chi Zheng and Guoping Qiu. In-the-wild facial expression recognition in extreme poses. *9th International Conference on Graphics and Image Processing (ICGIP 2017)*, pages 10615, 2018.

2. Chao Zhang, **Fei Yang**, Guoping Qiu and Qian Zhang. Salient object detection with capsule-based conditional generative adversarial network. *IEEE International Conference on Image Processing (ICIP)*, pages 81-85, 2019.

3. Dan Wang, Yu-Ting Tang, Jun He, **Fei Yang** and Darren Robinson. Generalized models to predict the lower heating value (LHV) of municipal solid waste (MSW). *Energy*, pages 119279, volume 216, Elsevier, 2020

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Great progress has been being made for computer vision over past decades, benefiting from the development of deep learning technologies. Among various artificial neural networks, convolutional neural network (CNN) [99] shows great performance in processing vision-based data, boosting researches such as object detection [168, 171], face recognition [182, 216, 225], image segmentation [58, 152], image translation [78] and etc. This report will discuss the topic of **image translation** with effective algorithms on certain applications.

1.1 Research Topic

1.1.1 Research Problem

Image translation plays an important role in computer vision, which translates images from a domain (A) to another domain (B). It takes an image as input and generates the corresponding targeting image. This research is adapted to a number of applications, such as day-to-night [100], winter-to-summer [78], picture inpainting [154], black-white colourization [76, 253], apple-to-orange [275], horse-to-zebra [275], scratch-to-image [274], image style transfer [82] and so forth. Some conventional topics, such as image super-resolution [263] and image de-noising [278], can be regarded as image translation as well, which take low-resolution or noisy images as input and generate high-resolution or clear images. Fig. 1.1 presents some image translation examples.

This report discusses image rendering and noise removal. Fig. 1.2 shows an

Figure 1.1: *Some image translation examples. The translation is from left-to-right. For example, "Day-to-night" means the left image is captured in day and is translated into the right image in night view. The example of "Style-transfer" is to translate a normal photo into an oil painting.*

application of image translation, image rendering. The images of domain A are plain without light effect, while domain B contains the corresponding rendered images. Image rendering tries to add light effect onto plain images. In this report, noise removal is discussed as a special type of image translation.

Diverse applications are explored by researchers, since the source domain and the target domain are defined almost arbitrarily. Some examples are shown in Fig. 1.1. Good performance makes it suitable for more actual scenarios. Researchers develop different techniques to fit for the characteristics of various scenarios.

The methodology of image translation follows the structure of encoder-decoder, which encodes images into latent representations and decodes that into desired images. The frameworks are trained with pairs of samples by minimiz-

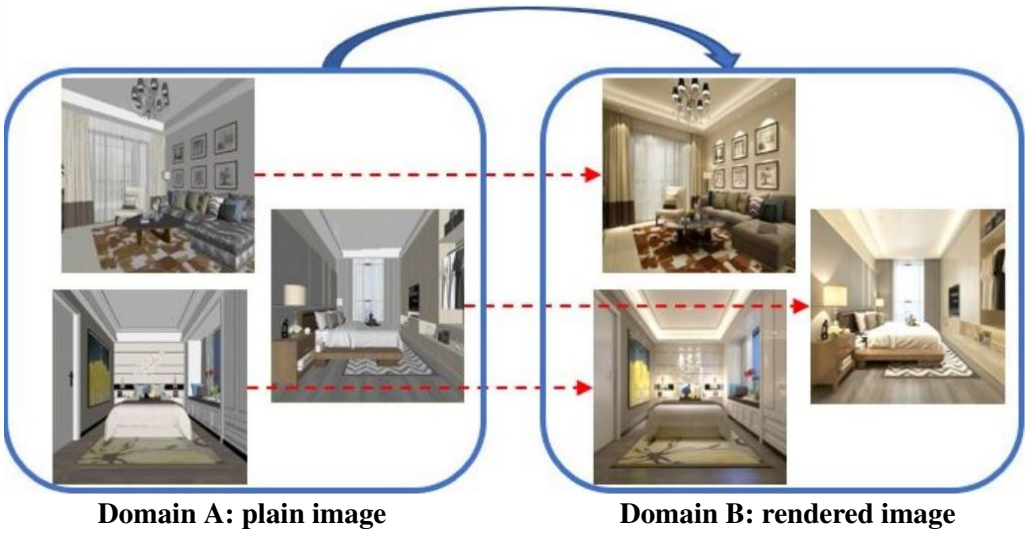


Figure 1.2: *Example of an application for image translation from domain A to domain B.*

ing the difference of outputs and targets. After millions of iterations of training, the model is able to learn the mapping from domain A to domain B. Classical models are like auto-encoder (AE) [229], variational auto-encoder [158] and U-net [175]. Along with the development of network structure, the definition of encoder and decoder are not separate clearly. Advanced networks connect a layer to many other layers with multiple skip connections [78, 175] and identity connections [60], so that encoder and decoder are merged into one network. In recent years, the generative adversarial network (GAN) [48] shows great ability in synthesizing data samples. GAN consists of a generator and a discriminator. The data distribution is learned by an adversarial learning process. The generator of a typical GAN generates images in desired style from random data. The generator can be constructed as an image translator, if it takes images as inputs instead of random data. In this way, image-conditioned GAN is built to process image translation problems. Advanced structures of generator (translator) are proposed by researchers. Better cost functions are explored to stabilise the training procedure along with the discriminator signal [78, 214, 240].

This report focuses on how to translate images with high-quality translation effect while reserving clear image content. New algorithms are proposed to enhance the performance in certain applications. In the following sub-section, the applications are introduced briefly and the detailed work is discussed in the

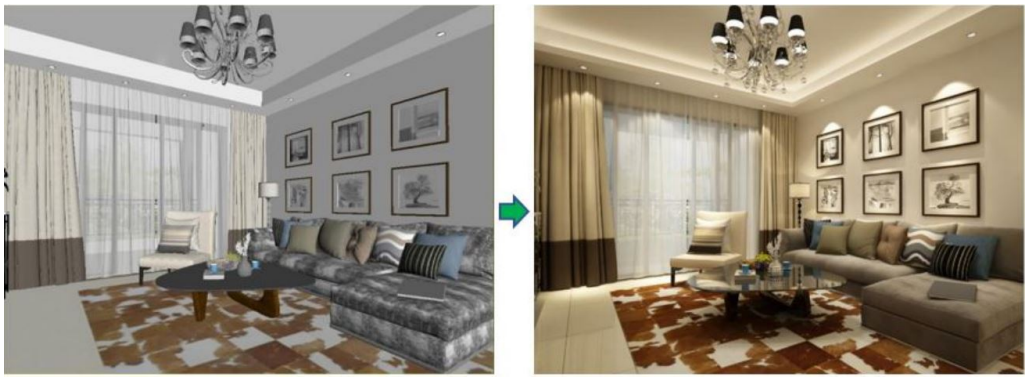


Image A: plain **Image B: rendered**

Figure 1.3: *An example of image rendering.*

following chapters.

1.1.2 Specified Applications

This report discusses algorithms on applications of image rendering, image de-raining, image de-hazing and image de-snowing. Image rendering for house interior decoration is a new topic for image processing. It has a research value that how artificial intelligence learns the complicated light effect, and an industrial value that how the work efficiency of designers is improved by reducing the rendering time. The other three applications belong to noise removal, which has been being a hot research for a long time, since images obtained in real scenarios suffer the quality degradation commonly from bad weather. In this work, new algorithms are proposed to enhance the model performance for each application.

Image Rendering

House interior image rendering aims to add light effect on plain images. An example is shown in Fig. 1.3. Image A is plain without light effect and image B is the rendered one. As an essential step of house interior decoration design, image rendering is to render 3D models into visual friendly images to present the decoration design to customers. The existing method is to calculate the light reflection pixel by pixel using the 3D software, such as $3DMax$ [6], which costs much computing resources and occupies incredible valuable time.

One challenge of image rendering is that the light reflection is complicated between various object surface and multiple different kinds of lights. It is hard

Image A: rainy **Image B: clear**

Figure 1.4: *An example of image de-raining.*

for a translation model to synthesize rendered images perfectly. A well designed model is expected to learn the spatial relationship between lights and objects so that the light reflection can be imitated. Both the local texture and the global spatial relationship are important in the rendering process.

To tackle the house-interior-decoration rendering problem with image processing techniques, a rendering dataset is constructed, named HIDER (House Interior Decoration Effect Rendering).

Image De-raining

Image de-raining is a classical noise removal problem, which aims to restore clear image content from single rainy image by identifying and removing rain. An example is shown in Fig. 1.4. Images captured on rainy days have low visual quality than clear ones, on which some image contents are covered by rain marks. Image de-raining improves the visual quality and benefits other vision-processing systems. For a long time, researchers have been developing de-raining algorithms. Recent deep-learning based models have achieved great results, but the problem remains unsolved.

Different from image rendering, image de-raining is to recover unknown objects that are occluded by rain. Rain is diverse in terms of extent, direction and transparency, which increases the difficulty for existing models. It is hard to separate rain and image content completely. In this report, the proposed method tackles the difficulties with a two-branch framework, which handles rain and image content simultaneously. A rain component aware module is designed to

Image A: hazy **Image B: clear**

Figure 1.5: *An example of image de-hazing.*

extract rain feature effectively.

Image De-hazing

Haze or fog removal has a long research history from the beginning of noise removal. De-hazing is to remove haze and recover image content, shown in Fig. 1.5. In recent years, many researchers have been trying to use CNN models to recover the clear content from a hazy image.

Base on the observation, a drawback of existing de-hazing models is that the models meet the difficulty in removing different extent of haze, especially with both close and distant scenes on the same image. The haze is in a low level with better transparency for close scenes, the opaque for distant scenes.

In this report, the feasibility of solving de-hazing problem as an image translation one is explored. A depth encoding algorithm is developed to link haze appearance with image depth information. By estimating the depth from hazy images, the de-hazing model gains additional effective information when processing haze removal task and produces state-of-the-art performance.

Image De-snowing

Snow removal tries to remove snowflakes from snowy images. An example is shown in Fig. 1.6. Traditionally, researchers consider snow removal as an extension of de-raining, which is transferred to process de-snowing by changing the training dataset. However, snowflakes are different from rain streaks or rain drops, which indicates that a de-raining model might not be appropriate for snow

Image A: snowy **Image B: clear**

Figure 1.6: *An example of image de-snowing.*

removal. From recent, researchers started to prepare larger snowy datasets and trying to develop specific algorithms for snow removal.

A big problem for snow removal is the diversity of snowflakes in terms of size, shape and transparency. Single size-fixed kernel based models are difficult to learn effective feature within one framework. Existing works are easily confused by large snowflakes against white image content and small snowflakes are more likely to be neglected in processing local details. Therefore, a multi-scale branch framework is built for snow removal with different scaling sub networks that focus on the corresponding size of snowflakes.

1.2 Detailed Research Points

1.2.1 Diverse Scenarios

Though researchers have developed image translation models for many scenarios such as those described in Section 1.1.1, there are still new applications that remain untouched. Due to the flexibility of the domain definition, many problems can be transferred as image translation. Image rendering introduced in Chapter 4 is a new one that has not been discussed in the previous literature. The image rendering problem is regarded as a type of image translation. The source domain contains plain images and the desired domain indicates rendered images.

To solve an applicable problem with proper computer techniques, it needs a great mount of work on the question interpretation, problem modelling, methodology analysis, data preparation, framework design, experiments and deploy-

ment debugging. Repeated trials are needed for each process until the best method is found. These explorations are inevitable in transforming an application into a technically solvable question.

Further, the lack of data is a vital problem commonly when a new problem is to be tackled with advanced technologies. To train and evaluate the framework proposed, high-quality datasets are needed. Since deep learning models rely on training data greatly, it is necessary for researchers to spend great effort in designing and building proper datasets. In this thesis, a house interior decoration rendering dataset is constructed for the rendering application in Chapter 4.

1.2.2 Generality

The generality of a model is an essential concern. Researchers have been trying to develop a more general model to fit as many scenarios as possible.

However, the difference between specific applications is nonnegligible, which indicates to apply the same model for all applications is difficult. It is hard for the same model to learn diverse features well at the same time for various applications, since the characteristics of translating domains from different applications many vary greatly. For example, the light reflection feature for image rendering and the noise feature from the de-noise work are much different, though both the two tasks could be processed as image translation problems. To apply the same framework without any modification on both tasks is hard to obtain the best performance for each. Further, even for the de-noising task, a general de-noising model is hard to build, since the types of image noise vary greatly due to the diversity of noise causes, such as weather condition, low resolution, motion blur, image compression and etc.

Therefore, a trade-off of generality and specificity is balanced for actual applications. For specific research problems, the corresponding algorithms are necessary.

1.2.3 Challenges of Each Application

For each application discussed in this report, there exist challenges that are hard to tackle with, such as the light reflection in image rendering, the haze extent in image de-hazing and the shape diversity in image de-snowing. The detailed dis-

cussions are present in the previous sub-section 1.1.2 along with the introduction of each application.

1.3 Thesis Overview

1.3.1 Chapter Contents

Chapter 2 presents the background knowledge about fundamental concepts and existing methods for image translation. The structure of image-cGAN used in many works is introduced. The development of capsule is discussed as the background knowledge of the proposed capsule-based framework.

Chapter 3 proposes the capsule-based cGAN framework. The capsule units are constructed in both the generator and the discriminator, which consider more global content and learn part-to-whole information to generate pleasing images. Especially, the two-branch discriminator with a capsule branch and a patchGAN branch [78, 235] distinguish the images in both the global and local aspects. This framework is used in applications discussed in the following chapters.

Chapter 4 presents the work on image rendering, which intends to add light effect to plain images. Considering the space correlation of objects within images, the proposed capsule-based cGAN framework shows great effectiveness. Based on the observation that house interior images contain many straight lines that affect the visual quality, a line preservation loss is proposed to supervise the image generation with good line shapes. Comparing with other famous image translation models, the proposed model shows great improvement. The ablation experiments examine the effectiveness of the capsule module and the line preservation loss. A dataset is proposed to for the house interior image rendering work.

Chapter 5 describes a de-raining framework. To enhance the model ability of identifying the rain feature, the framework is optimized with cost functions from two aspects, the image content and the rain component. A rain component aware (RCA) loss is designed to extract rain feature from synthesized images and back-propagated as a training signal. The model performance is experimented on different image sizes to examine the model robustness. The proposed model outperforms the state-of-the-art approaches.

Chapter 6 introduces a depth aware framework for haze removal, which estimates the depth map and encodes the depth feature into the haze removal model. The motivation is that the haze has close relation with the image depth, especially for outdoor scenes. The experiments shows the effectiveness of the depth awareness module.

Chapter 7 proposes a multi-scale framework for snow removal. Since the sizes and shapes of snowflakes are in a large range of diversity, the de-snowing framework is built with various scales of sub-nets to extract feature in different scales. By learning feature of various snowflakes better, the proposed model outperforms other de-snowing methods.

Chapter 8 discusses the analysis of image quality. The experiments show that simple convolutional neural works are effective in estimating quality types and quality levels, which provides useful information to a joint system for the final prediction.

Chapter 9 discusses the task oriented IQA (image quality assessment) for synthesized images from image-cGAN framework. The synthesized images of previous works (image rendering, de-raining, de-hazing and de-snowing) are evaluated with the task oriented IQA as the comparison of conventional metrics, PSNR and SSIM.

Chapter 10 is for the conclusion, in which a summary is concluded on the contributions, the limitations and the future work.

1.3.2 Correlation of Chapters

Chapter 3 presents the proposed capsule-based image-cGAN framework, which is a base structure in this report. From Chapters 4 to 7, the detailed works on image rendering, de-raining, de-hazing and de-snowing are introduced. Chapters 8 and 9 discuss the work on image quality which is applied to evaluate the results of translated images.

Image rendering is a typical image translation problem that aims to add light effect. Comparing with other image translation problems, image rendering is more related to the global image content, due to the close relationship of the lights and the reflection from the object surface. The contribution of the capsule module for image-cGAN framework is designed exactly for learning the part-to-whole relationship within an image. Therefore, the specific design of the capsule

suits for the application of image rendering.

The proposed capsule image-cGAN is a general framework that suits for paired image translation as long as there is meaningful correlation for each pair. Image noise removal is another type of image translation. This report discusses weather-affected noise removal such as rain removal, haze removal and snow removal. By examining the characteristics of rain, haze and snow, various methods are proposed to improve the model performance on each certain application.

Chapters 8 and 9 discuss the image quality assessment on synthesized images, especially for images translated from other domains. The experiments of Chapter 8 show that CNN is able to estimate the quality properties effectively by training blindly without referencing images. Thus, a task-oriented IQA method is proposed to estimate the quality of translated images, in which the quality feature is extracted from a level-pretrained feature extraction network.

1.4 Contributions

In this report, a capsule-based cGAN framework is proposed for image translation. For different scenarios, various techniques are proposed. Further, a method to evaluate the quality of synthesized images is introduced. The technical contributions can be summarised as follow.

1.4.1 Technical Contributions

* A capsule-based image-conditioned generative adversarial network (capsule-based image-cGAN) is proposed for image translation. The experiments demonstrate its effectiveness on various applications.

* A line preservation loss is proposed to maintain the line shapes during the learning of image rendering, since lines play an important role in the visual quality of the synthesized images. In the process of image rendering, light effect relies much on the lines to estimate light reflection, which can be grasped by the line preservation loss.

* A RCA (rain component aware) module is proposed to extract effective rain feature, which is used to supervise the training of de-raining model.

A two-branch optimised framework is designed to identify rain in two aspects, image content and rain component. The model performance is improved with removing rain accurately and producing visual friendly images.

* A depth-aware de-hazing framework is developed. Since depth information affects haze extent greatly, a depth-aware module is designed to learn depth feature to help de-hazing. In the experiments, the depth prediction is proved to be effective in the process of de-hazing, especially for outdoor images.

* A multi-scale branch framework is designed to remove various sizes of snowflakes from snowy images. Different branches are designed to process various scales of images so that both big and small snowflakes are learned by the convolution kernels.

* The experiments show that computer vision tasks are sensitive to image quality, which can be estimated by CNNs. A task-oriented image quality assessment (IQA) method is proposed for synthesized images from generative networks. The proposed approach is more fit for translated images than other IQA methods.

1.4.2 Dataset Contributions

* **HIDER**: House Interior Decoration Effect Rendering dataset is built, consisting of 1,174 pairs of plain and rendering images, rendered from 453 house interior decoration design models. All images are in the size of 1300×939 rendered from the software *3D Max* [6]. The images covers four kinds of rooms, living room, bedroom, study room and dining room, in four styles, Chinese style, European style, modern style and post-modern style. https://yang-fei.github.io/tf-capsule-rendering/

* **SnowySet**: A de-snowing dataset, consisting a synthesized snowy dataset and a real-world snowy dataset, is developed. The synthesized set consists of 52,760 snowy images by adding various types of snowflakes on 5,276 clean images. The clean images are selected from BSDS500 [5],

UDIC.v2 and Snow100K [129] by removing the images that are meaningless to the snowy weather (e.g. indoor, underwater, close-view or water sports images). In addition, 100 real-world snowy images are downloaded from Internet for subjective evaluation. https://yang-fei.github.io/capsule-deraining-RCA-cGAN/.

* **Real-world rainy dataset**: A real-world rainy dataset is built with 828 rainy images, which are downloaded from Internet and captured in rainy days. Various image contents are covered, such as buildings, trees, street views and etc. The rain is in different types, rain drops and rain streaks. https://yang-fei.github.io/caps-multiscale-desnowing/.

Chapter 2

Background

2.1 Pre-knowledge of Deep Learning

The research community has witnessed the great development of deep learning for the past decade, due to the amazing performance of deep learning models. This attracts an increasing number of researchers to develop advanced deep learning models, which apply layers of artificial neural networks to simulate the non-linear functions mapping from the input data to the output data. Specially, the convolutional neural network (CNN) based models are quite good at dealing with computer vision problems by processing the image data with layers of convolution kernels. In recent years, advanced technologies have been being explored by deep learning professionals in aspects such as the framework architecture, the training algorithm, the data transforming, the optimization function and so forth. Based on these, the development of computer vision has been being boosted in a high speed.

From the AlexNet [99] that was applied to image classification in the ImageNet competition firstly in 2012, great CNN-based architectures have been proposed for computer vision, such as VGG (16, 19) [191], InceptionNet (v1, v2, v3, v4) [77, 202–204], ResNet (layer-18, layer-34, layer-50, layer-101, layer-150, layer-1000) [60], DenseNet [74]. These famous networks expand the depth and width by structuring more convolutional layers and applying various sizes of convolution kernels. Started from VGG, researchers found that a deeper network shows better performance than a shallow one, but is easier to suffer the problems of gradient vanishing and explosion [50, 271]. InceptionNet discussed the

implementation of multiple convolution kernels parallel. The identity connection introduced in ResNet and the dense connection proposed in Densenet make networks with each layer connected by multiple layers, reducing the side effect of gradient vanishing or explosion by building multiple back-propagating paths between layers. This kind of "skip" connections are used in many later works to improve performance [23, 110, 113, 247, 248]. Since deeper networks require more computing resource, the light networks, MobileNet (v1, v2) [69] and ShuffleNet (v1, v2) [134, 259] are explored for light networks with fewer parameters and less computation so that some light platforms or mobile devices are able to run the models.

The improvement of architecture has stimulated numerous improved networks for various kinds of specific scenarios. According to the structure style, the U-net series (U-net+, U-net++) [57, 175, 272] were proposed and improved with an encoder-decoder structure for image segmentation. The recurrent neural networks (RNN) [183] and the long-short time memory model (LSTM) [65] are good at processing language or video like sequential data. The generative adversarial network (GAN) [48] is suitable for data generation. According to the research topic, some famous frameworks have been proposed in recent years, such as Faster RCNN [171], SSD [122], YOLO (v1, v2, v3) [166–168] for object detection, Facenet [182] and Sphereface [123] for face recognition, Pix2pix [78] for paired image translation, CycleGAN [275] and DualGan [242] for unpaired image translation and etc.

Since learning-based models rely on training data, data preprocessing is an vital step in training models. Commonly, the raw values are mapped into the range of $[-1, 1]$ or $[0, 1]$ by the processing of scaling, zero-mean or standard deviation [151]. Occasionally, The illumination normalization [157] is applied as well in scenarios with a large extent of illumination changes [84]. Complex networks demand a huge mount of data, which inspires researchers to explore methods for data augmentation such as randomly cropping, flipping, randomly rotation, or generating fake training samples using the synthesis methods (GAN) [4]. Data augmentation is much useful, especially data limited application, where the data is hard to collect or label. Another problem of the data preparation is the balance of sample distribution on categories. A training friendly dataset has the same number of training samples for every category so that the training objective

is balanced without stress on any category.

To train the network effectively and efficiently, researchers develop various optimizer such as batch gradient descent (BGD) [176], stochastic gradient descent (SGD) [176], mini-batch gradient descent (MBGD) [176], nesterov accelerated gradient (NAG) [15], adaptive gradient algorithm (Adagrad) [193], Adadelta [246], RMSprop [9] and adaptive moment estimation (Adam) [93]. Theses optimizers try to update the gradients on the weights with different strategies that may fit various problems. Except for the optimizer, other techniques are proposed such as the drop out [196] to overcome the over-fitting, the batch normalization [77] to enhance the training speed and reduce the affects of hyper-parameters, and the momentum [15] to stabilize the training procedure.

The training objective function (loss function) has great influence on the model performance. The absolute distance ($L1$) and the mean squared error ($L2$) of the output and the ground-truth are applied conventionally. The function of softmax [16, 49, 138] is effective for classification problem. The cross entropy loss is applied to measure the distance of two sets of data distribution, which is applicable for multi-class prediction and data distribution estimation. To further enhance the performance, the centre loss [225], the L-margin loss [124], the additive margin loss [215] and the cosine loss [216] are proposed. In capsule network, the margin loss and the spread loss are developed. For image generation, the perceptual loss computes the feature distance by forwarding two images through a pre-trained network. Since SSIM [270] performs closer to human visibility than pixel-error based evaluation metrics, researchers apply SSIM as a loss to supervise the generator to generate high-visual-quality images.

In this report, a general framework is constructed for image translation. And the model performance is further improved by proposing task-oriented techniques for certain application scenarios.

2.2 Image Domain Translation

The concept of image domain translation was addressed firstly in 2016 by Taigman et al. [205], who adopted generative adversarial network (GAN) and variational auto-encoder (VAE) as the mapping function to accomplish the image domain translation [18]. But the image domain translation is indeed a wide re-

search concept that covers a large number of applications from a long time ago, since the source domain and the target domain could be defined according to the scenario almost arbitrarily. Except for those applications introduced in Chapter 1, image translation includes more such as image noise removal and human face transformation. The image noise removal with a longer research history aims to recover clear image content from noisy images such as rain removal, snow removal, haze removal, blur removal and super-resolution. The human face transformation indicates applications like expression transformation, smile addition, eye glass removal, makeup removal, gender transfer, hair transformation, face generation on age changes and so on. Among these, the research of image noise removal started from long ago as an conventional problem of image processing. But it is boosted by the development of CNN. The report discusses the related literature in Chapter 5 for rain removal, Chapter 6 for haze removal and Chapter 7 for snow removal.

Along with the great development of CNN for computer vision, a great number of complicated image translation tasks have attracted researchers' attention by proposing advanced techniques. Cordts et al. [29] proposed the cityscape dataset, which provides the semantic labels as the label domain. Zhang et al. [253] focused the research of black-white colourization. The edge-to-image or sketch-to-image attracts much attention in literature [37, 244, 274]. The day-night transformation was discussed by Laffont in 2014 [100]. The image style transfer was proposed by Johnson et al. [82].

Most recently, the new era of the research for image translation started from the work of Isola et al. [78] in 2017, which proposed an effective image-conditioned GAN on many translation applications with an aerial map dataset contributed to the research community. They developed the framework by setting U-net [175] as the image translator/generator and applying patchGAN [78] as the discriminator. Under the joint supervision from the pixel loss and the patchGAN discriminator, the generator is able to learn the mapping from the input to the output well. Since then, much attention has been attracted on this kind of structure. Following researchers developed advanced algorithms based on this [21, 75, 83, 159, 219, 237]. To train the image translation framework, a paired dataset is needed with paired images in the source domain and the target domain correspondingly. Thus, researchers proposed new datasets when exploring ap-

plications [29, 78, 235, 274]. Though the image-conditioned GAN is powerful, it needs a significant mount of training data with paired input and target, which requires great labour for the data preparation.

The image translation described above is called paired translation, which needs paired samples to train the model with each pair of input-target images on pixel level, for which a lot of human labour is needed to prepare the dataset. According to the training strategies for learning-based algorithms used in existing works commonly, the image translation can be categorized into paired translation [78] and unpaired translation [92, 121, 186, 256, 275, 276]. It was proposed in CycleGAN [275], DiscoGAN [92] and DualGAN [242]. The unpaired translation models are trained on two subsets of images, source domain set and target domain set, with no need to make sure images paired. The supervision signal is achieved by comparing the original image with the synthesized image during the bi-directional translations of input-to-target and target-to-input. Nevertheless, since the training procedure lacks the straightforward description of the target domain during each iteration of the bi-directional translation, the inherent properties of the original images may not be learned well by the model, which will cause the model to produce unexpected marks or artifacts [256]. The unpaired translation attracts an increasing number of researchers, because it is more challenging and needs less labour for data preparation.

Since unpaired translation is with more uncertain factors that decrease performance, this report discusses paired image translation with proposing novel techniques. The applications introduced in this report contain datasets with paired images.

2.3 GAN and Image Conditioned GAN

2.3.1 Generative Adversarial Network

Recent years, Generative Adversarial Networks (GANs), proposed by Goodfellow in 2014 [48], have attracted tremendous amount of attention, which train a generator and a discriminator in an adversarial way for the purpose of generating non-existent samples with the same distribution of the training set. A fundamental GAN consists of a generator G and a discriminator D playing a min-max

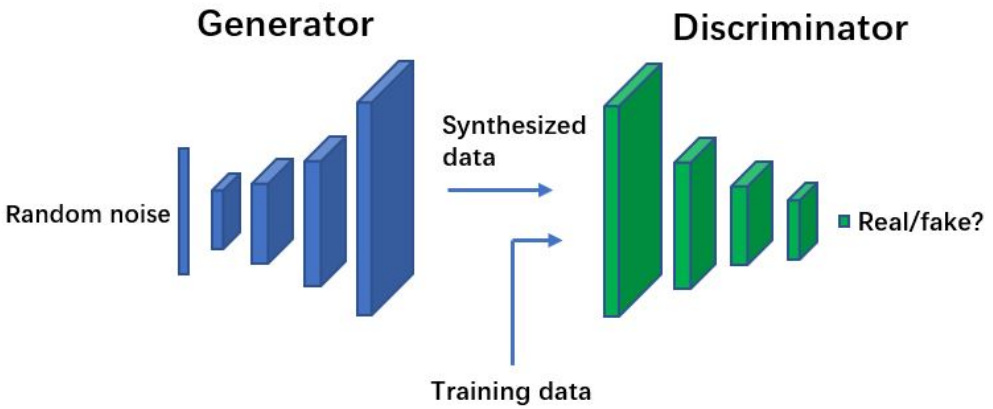


Figure 2.1: *Example for the structure of generative adversarial network.*

optimization game to train the two models simultaneously, where the "adversarial" means G tries to produce data to fool D and D tries to gain its ability to distinguish the data sample as real or fake. An example for the structure of GAN is shown in Fig. 2.1, where the generator consists of four deconvolutional layers and the discriminator is designed with four convolutional layers. After iterative training, the optimization arrives at a stable status, when G is able to produce quite "real" like data to fool D successfully even though D has been trained well with quite a good distinguishing ability. The formulation of the min-max function is show in Eq. (2.1).

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}}[\log D(x)] + E_{z\sim P_{noise}}[\log(1 - D(G(z)))] \quad (2.1)$$

where $P_{data}(x)$ is the real data distribution and $P_{noise}(z)$ represents the random input noise data. The $G(\cdot)$ and the $D(\cdot)$ indicate the forward calculation of G and D. D and G are optimized by maximizing and minimizing Eq. (2.1) iteratively.

According to the principles of GAN, the distribution of target data is learned by the generator by taking random data as input. A well-trained generator could generate realistic samples no matter which exact random data is input. There is no control on data generation once the model is trained over. Thus, GAN is regarded as an unconditioned generative model.

The simple optimizing algorithm without any calculation of likelihood or probability for the data distribution makes GAN awfully flexible for image gen-

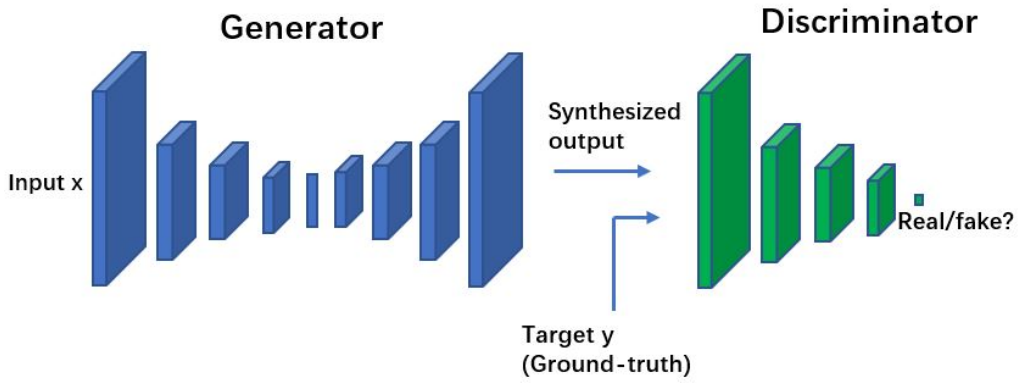


Figure 2.2: *Example for the structure of the image conditioned generative adversarial network.*

eration [149, 161, 180]. The flexibility has enabled various extensions from GAN such as the support structured prediction [149], energy based models [264] and inforgan model [24]. For the past three yeas, there are numerous papers proposing new structures for various applications with a variety of improved techniques. Avinash summarized more than five hundred GANs and listed them in [62]. Different GANs are developed for various applications by modifying the network structures that form the generator and the discriminator, the cost functions that supervise the training process and the frameworks composed by multiple generators and discriminators. This report presents novel network structures and new cost functions to enhance the performance on certain applications.

2.3.2 Image-conditioned Generative Adversarial Networks

Based on the generative adversarial network (GAN) [48] used for the imitation of data distribution, conditioned generative adversarial network (cGAN) learns to generate data samples on conditioned signals such as class labels, data properties or data from different modality [139]. The conditioned signals are encoded into the generator as latent variables to control the diversity of generation [150, 212]. The data generation is directed by the conditioned signals to synthesize data samples with a specific property. For example, a face generator is able to synthesize a smile face by setting the "smile" condition if it is trained with a series of expression labels. CGAN expands the application scenarios with defined labels.

Image-conditioned GAN(image-cGAN) gains its popularity in image trans-

lation [78] by encoding an inputting image as the conditioned labels. For certain applications, the noise input of GAN is discarded to restrict the unique output for an inputting image. Therefore, the model takes images as the inputting condition to generate the corresponding images. An example for the structure of image-cGAN is shown in Fig. 2.2, where the generator is composed by stacked convolutional and deconvolutional layers. This kind of framework is good for solving image translation problems. A famous image-cGAN framework was proposed by Isola et at. [78] with promising results for a variety of applications. It attempts to output sharp and realistic local image patches by proposing the patchGAN [78] to restrict the discriminating attention in local patches [109]. The min-max optimization of image-cGAN is shown in Eq. (2.2).

$$\min_{G}\max_{D} V(D,G) = E_{y\sim P_{data}}[\log D(y)] + E_{x\sim P_{data}}[\log(1 - D(G(x)))] \quad (2.2)$$

where the x and y indicate the inputting and targeting data.

To improve the distinguish ability of D, the inputting image is sent to D as well [78]. The formulation of Eq. (2.2) is written as Eq. (2.3).

$$\min_{G}\max_{D} V(D,G) = E_{x,y\sim P_{data}}[\log D(x,y)] + E_{x,y\sim P_{data}}[\log(1 - D(x,G(x)))] \quad (2.3)$$

The conventional losses used for image synthesis frameworks such as L1-norm and L2-norm distances focus on differences at pixel level between synthesised and ground-truth images. This makes such losses too sensitive to pixel noises and lack the ability to capture high-level information [104]. Researchers explored new loss functions for image reconstruction. Johnson et al. [82] proposed the idea of perceptual loss by minimizing feature differences between synthesised and ground-truth images. It was proved to be effective in image-cGAN [78]. Many works benefit from it such as image super-resolution[56, 104], colourization [76, 253], style transfer [82] and etc. But intuitively, perceptual loss focuses on the content structure and grasp the content feature, which is less sensitive to the noise component, since the noise component is a relatively low-level feature with less structural information. Better cost functions are proposed to supervise the learning for specific applications.

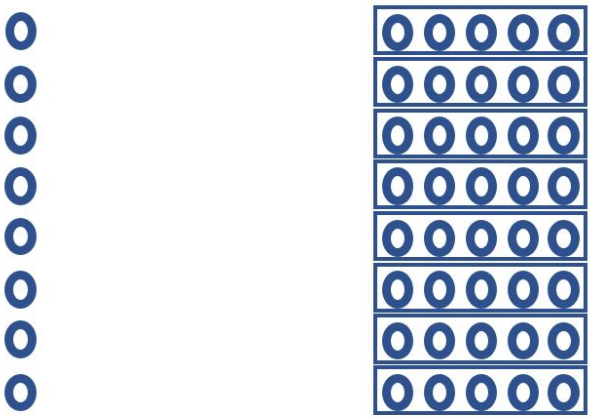
2.4 Capsule

2.4.1 The Concept of Capsule

Conventional convolutional neural network (CNN) based models extract the pose invariant feature by applying layers of convolutional filters to ensure the robustness to scale, translation and rotation. Due to the limitation of kernel size, the feature of each layer is formed as a bunch of isolated values. CNN uses the pooling layers to achieve the local translational invariance by down-sampling the feature of local pools of translated replicas of the same kernel. By these processes, CNN is designed to be view-invariant so that objects can be recognised in any kind of scale, rotated angle and shape. For a long time, researchers have been trying to develop methods to improve the invariance ability to extract useful information from content-rich images.

While, Hinton et al. argued that CNN might be misguided in what they are trying to achieve [63]. In contrary to seeking the purpose of viewpoint invariance in CNN that applies scalar neurons to summarize the feature for a local pool, intelligent models should aim for strong structural feature representation including the relative spatial relationship by using more complex computation for each neuron. For instance, the separate features of eyes, mouth and nose are important to face recognition, but the precise relative positions of the facial organs are also quite vital for identifying the face. Comparing with the scalar neuron, a vectorized neuron learns more properties including the precise pose, position and lighting from the implicitly defined visual entity. The properties are able to be represented by the "directions" of the vector, while the characteristic of viewpoint invariance is reserved by expressing the existence of the entity according to the length of the vector. The vectorized neuron is named a capsule by Hinton et al. [63]. An example is shown in Fig. 2.3.

A capsule is a vectorised neuron with a number of values, replacing the scalar neuron of conventional artificial neural networks. Thus, the conventional neural network is able to be transformed into the capsule network with layers of capsules. Each connection between two neurons becomes the matrix transformation instead of the conventional scaler multiplication. According to the introduction of [63], the capsules are computed to activate the high-level capsules by compare the right spatial relationship of visual entities learned within



(a). Eight conventional scalar neurons (b). Eight capsules with each capsule of five values

Figure 2.3: *The comparison of conventional scalar neurons with capsules.*

each capsule therefore make the capsule network to gain the part-to-whole learning ability. Hinton et al. proved the effectiveness of capsule with auto-encoder experiments on MNIST digit images. Much attractiveness of capsule was not raised by the research community until Sabour et al. proposed the dynamic routing in 2017 [179], in which they proposed an effective method to implement the capsule structure in the conventional CNN successfully.

2.4.2 Dynamic Routing Between Capsules

Though the capsule structure was proposed by Hinton et al.[63] to enhance the capability of feature representation in deep learning models, the usage of capsule is limited only for the conventional fully-connected auto-encoder. The capsule was used in matrix multiplication between fully-connected layers. It is hard to apply it in CNN to process image-based data, since the computation would becomes greatly large if all neurons are replaced by capsules.

To overcome the problem of computation limitation, Sabour et at. implemented capsule in the CNN successfully by proposing the PrimaryCaps layer [179] to transform the conventional convolutional layer maps into capsules layer maps. The implementation of PrimaryCaps layer is shown as Fig. 2.4. The PrimaryCaps layer includes a conventional convolution to set a reception field and a reformation to transfer the scalars into capsules. The reformation is achieved by grouping the values along with the channel dimension into capsules. A convo-

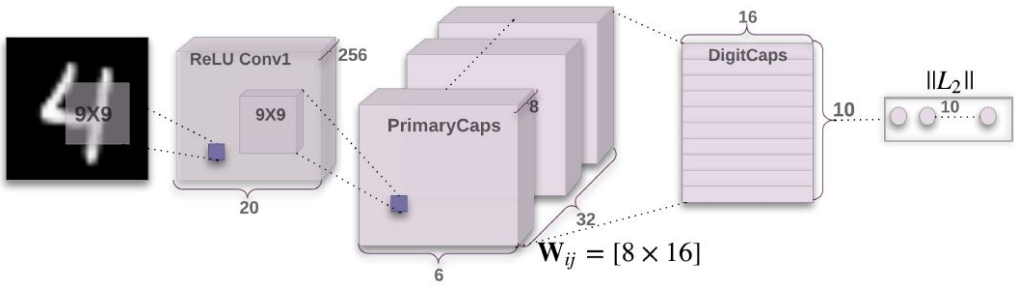


Figure 2.4: *The capsule network introduced in [179]. After the 256 convolutional layer maps "ReLU Conv1", within the PrimaryCaps layer the values are processed by convolution and then grouped into* 8×32 *along the channel dimension therefore form into 32 capsule maps in the size of* 6×6 *with each capsule of 8 values. Then, all capsules from the PrimaryCaps layer are fully connected with the ten capsules of the DigitCaps layer.*

Table 2.1: Dynamic Routing Algorithm

Routing($\hat{u}_{i|j}, r, l$)**:**
for all capsule i in layer l and capsule j in layer $l+1$: $b_{ij} \leftarrow 0$.
for r iterations **do**
for all capsule i in layer l: $c_i \leftarrow softmax(b_i)$ as Eq. (2.4)
for all capsule j in layer $(l+1)$: $s_j \leftarrow \sum_j c_{ij}\hat{u}_{j|i}$
for all capsule j in layer $(l+1)$: $v_j \leftarrow squash(s_j)$ as Eq. (2.5)
for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}v_j$
return v_j

lution layer map with M channels is regrouped into $\frac{M}{N}$ capsule maps with each capsule of N values (defined capsule dimension of N).

Sabour et al. introduced the dynamic routing algorithm within capsule layers [179]. The connection of a capsule with each of its connected capsule from next layer is assigned with a weight according to the similarity of two connected capsules. A large similarity wins a large connection weight c_{ij} from the capsule i of layer l the capsule j of layer $(l+1)$. The detailed routing algorithm is described in Table 2.1. The iteration number r is set as 3 to obtain the best results according to the experiments [179].

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{2.4}$$

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{2.5}$$

After r iterations of routing, c_{ij} is assigned with the proper number, according to which the capsule "finds" the best rout to activate the capsules of next layers. With the proper c_{ij}, the capsule network is able to learn the relationship of its entities and the sub-entities as well as the properties of the entities.

Sabour et al. demonstrated the effectiveness of capsule structure with the dynamic routing algorithm. Since then, the capsule attracted more attention. They provided a practical method to implement capsule in conventional convolutional neural networks. An increasing number of researchers started to advocate into the capsule research field.

2.4.3 Optimization of Capsule Network

The output of capsule is activated through Eq. (2.5). According to [63, 179], the length of the instantiation vector represents the existing probability of a capsule. Thus, the conventional softmax loss or cross entropy loss is proper to be applied to the capsule layer by calculating the lengths of the outputting capsules.

Though cross entropy loss is proper for multiple predictions, Sabour et al. proposed a separate margin loss, which calculates each capsule loss according to Eq. (2.6). The total loss is the sum of the losses of all outputting capsules.

$$L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k)\max(0, \|v_k\| - m^-)^2 \tag{2.6}$$

where $T_k = 1$ if the class k is present; m^+ and m^- are margins set as 0.9 and 0.1; λ is the down-weighting of the loss for the absent classes to avoid the initial learning shrinking the lengths of the vectors of all capsules.

Since an improper λ of Eq. (2.6) may cause the shrinking of all capsules resulting in the failure of training, Hinton et al. proposed the "spread loss" [64] to make the training less sensitive to the initialization and hyper-parameters. The spread loss directly maximize the gap between the activation of the target class (a_t) and the activation of the other classes. If the activation of a wrong class a_i is too large, closer to the right activation a_t, the spread loss is calculated by the squared distance according to Eq. (2.7).

$$L_i = (\max(0, m - (a_t - a_i)))^2, L = \sum_{i \neq t} L_i \tag{2.7}$$

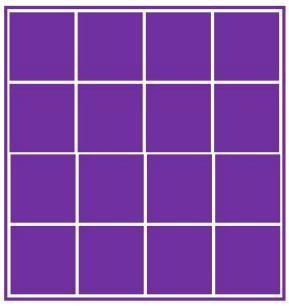


Figure 2.5: *A capsule with* 4×4 *values.*

where m is the loss margin set according to the training stage. To avoid dead capsules in the earlier layers, the the margin begins with 0.2 initially and increases to 0.9 during the training. Specially, spread loss is equivalent to squared Hinge loss [52] with $m = 1$.

2.4.4 Development of Capsule

Hinton's research group proposed the concept of capsule in 2011 [63] and have been doing the related research in recent years. Sabour et al. [179] discussed the dynamic routing method in 2017. Hinton et al. [64] proposed another routing method, EM (Expectation-Maximization) routing algorithm. The recent literature [97] introduced the stacked capsules. Besides Hinton's group, there are other researchers who have been studying capsule. Some advanced capsule structures are proposed, such as graph capsule [213], self-routing capsule [54] and self-attention capsule [66].

Vector to Matrix

Sabour et al. [179] developed the capsule network with the vectorized capsule structure proposed by Hinton et al. [63]. The structure of capsule is not limited in one dimension. Hinton et al. proposed the matrix capsule by designing the capsule unit as a matrix. A 4×4 capsule is shown in Fig. 2.5. The matrix-based capsule stores much pose information while the presentation of the capsule entity remains the same definition of the vectorized capsule, the norm of all its values.

Routing Method

Hinton et al. proposed EM routing method with the Expectation-Maximization (EM) procedure by adjusting the means, variances and activation probabilities of the capsules in the layer $L + 1$ and the assignment probabilities between all $i \in L, j \in (L + 1)$ iteratively [64]. The detailed routing process is described in [64]. Compared with the Dynamic Routing method [179], the EM routing fits matrix-based capsule better with computing efficiency.

The dynamic routing and EM routing still cause high computational complexity. Hahn et al. [54] proposed a self-routing strategy to improve the routing computation by applying the routing on each capsule with its subordinate routing network. Each capsule determines its routing weights by itself without coordinating the agreement with other capsules. But, a subordinate routing network is maintained to provide the routing weight to each capsule. They prove the effectiveness and efficiency of self-routing better than dynamic routing and EM routing with experiments on CIFAR-10 [98] and SVHN [145] and additionally SmallNORB [103].

2.4.5 Application of Capsules

Due to the effectiveness of capsule network, researchers apply capsules to many applications by implementing the capsule structure into conventional models. As Sabour et al. only introduce the capsule structure on MNIST [102] experiments in [179], Xi et al. [228] build a network with more capsule neurons to test the performance in processing complex data. The experiments are done on CIFAR-10 [98] with reconstruction modules. The results show that capsule networks gain improvement than conventional convolutional neural networks, but the computation increases greatly if a larger number of capsules are placed. LaLonde et al. [101] apply capsules in object segmentation with proposing the locally-connected routing and the concept of deconvolutional capsules. The proposed SegCaps are proved to process object segmentation well with substantial decrease in parameter space. Jaiswal et al. [79] used the capsule network as the discriminator to improve the classification performance. They raise the idea that capsule could be implemented into the discriminator to enhance its distinguishing ability. The early research of capsule focuses on checking the effectiveness

of capsules on different tasks. The capsule implementation is similar to that in Sabour's paper [179].

Along with the development of capsule, researchers try to apply capsule into complex networks to tackle more applications. Wang et al. [217] implement capsule into a GAN for image inpainting, in which the generator is from the fundamental capsule network introduced by [179] and the discriminator is similar to that of Jaiswal's work [79]. Though the capsule implementation is the same as before, the combination of generator and discriminator makes the model achieves good results for image inpainting. Verma et al. [213] design the capsule-based graph CNN to solve the graph classification problem, which provides a way to built a model combining capsule and graph structure. The experimental results of Verma is promising with great improvement. Zhang et al. [231] build a capsule graph neural network, which adopts the concept of capsules to address the weakness in existing GNN-based graph embedding algorithms. They use the routing mechanism to capture important information at the graph level so that the graph properties can be captured from different aspects by the multiple embeddings of routed capsules. Bass et al. [8] introduce the capsule-based image synthesis network for data augmentation of biomedical datasets. The capsules are built on each convolutional layer maps to maintain an extra layer computation before being connected to the next convolutional layer. Zhao et al. [265] construct capsule structure into 3D auto-encoders to process sparse 3D point clouds. The use capsule to preserve spatial arrangements of the input data so that the model gains better ability in understanding the spatial relationship in processing point clould-related substantiated tasks. Rajasegaran et al. [164] construct a deep capsule network architecture which uses a novel 3D convolution based dynamic routing algorithm. They use skip connection to connect different capsule-based layers. Hoohi et al. [66] proposed the self-attention capsule network for image classification. Based on the self-attention network, they appended the capsule layers before the final outputting layer. They evaluate the performance within and across different datasets to prove the effectiveness. Nguyen et al. [148] use a capsule network to detect various kinds of spoofs from forged images or videos. Their approach can be a preprocessing step in real-world application scenarios.

2.5 Evaluation Metrics

Commonly the quantitative evaluation for image generation is to compare the similarity of synthesised images and ground-truth. The absolute error (L_1 norm) and the Mean Squared Error (MSE or L_2 norm) are basic ways to calculate the pixel difference between two images. Because the two do not fit the human visual sense well, there are two other better metrics, the Peak Signal-to-Noise Ratio (PSNR) and Structure SIMilarity Index (SSIM) ([270]), which are commonly used in existing image comparison works ([17, 104, 235]).

The formulations are shown in Eq. (2.8), (2.9), (2.10) and (2.11).

$$L_1(m,n) = E_{m,n} \left\| m - n \right\|_1 \tag{2.8}$$

$$MSE = E_{m,n} \left\| m - n \right\|_2 \tag{2.9}$$

$$PSNR(m,n) = 20 * \log_{10}(\frac{MAX}{\sqrt{MSE}}) \tag{2.10}$$

$$SSIM(m,n) = \frac{(2\mu_m\mu_n + c_1)(\sigma_{mn} + c_2)}{(\mu_m^2 + \mu_n^2 + c_1)(\sigma_m^2 + \sigma_n^2 + c_2)} \tag{2.11}$$

where m and n represent two images; $E_{m,n}$ indicates the mean on all pixels; MAX is the maximum value of the image; μ_m and μ_n are the means of the two images; σ_m and σ_n are the standard deviations; σ_{mn} is the covariance; and, c_1 and c_2 are constants to avoid the instability of the equation.

2.6 Summary

This Chapter introduces the background of deep learning technology as the pre-knowledge of this report. The development of image translation and its approaches are introduced. The basic way to tackle image translation is to encode input and generate the corresponding output. The structure of translator determines the performance of overall framework. The background of capsule is introduced as well, which will be constructed in the translating model to synthesise pleasing images. The specific literature review on each topic will be carried

out in subsequent chapters (from Chapter 4 to 9)

Chapter 3

Framework: Capsule-Based Image-Conditioned Generative Adversarial Network

3.1 Introduction

In this chapter, a capsule-based image-conditioned generative adversarial network is introduced, which is the fundamental framework of this report. A shortcoming of the Convolutional Neural Network (CNN) is that the scalar representation and the additive nature for each neuron are ineffective to capture precise spatial relationships, such as part-to-whole [63]. To overcome this, Hinton et al. [63] proposed the concept of capsule, using a vector to replace scalar for each neuron in traditional neural networks. These vectorized neurons construct a capsule layer with each of them connected to the capsules in the next layer through a *dynamic routing* algorithm [179]. Dynamic routing computes the similarity between a capsule and each of its connected capsules, while assigning corresponding weights. A larger connecting weight indicates a higher similarity. The capsule structure is shown as the yellow blocks in Fig. 3.1 and 3.2. Compared with common recognition tasks, the image translation process, like effect rendering, relies significantly on spatial relationships in the image, such as the relative positions between lights and furniture. Hence, I incorporate capsules and dynamic routing because of better capability of capturing and representing such information.

3.2 Framework: Capsule-based Image-cGAN

3.2.1 Overview

As the name suggests, a typical GAN trains both a generator G and a discriminator D iteratively with two adversarial objectives: 1) G aims to generate synthetic data very similar to the real data that D has trouble to distinguish; 2) D aims to discriminate the real data from the synthetic ones generated from G. The optimization of GAN has been introduced in Chapter 2, shown as Eq. (2.1). Specifically, the image conditioned GAN takes an image as the input for G and the following min-max function (Eq. (3.1)) concerning both G and D are optimized.

$$\begin{aligned}\min_G \max_D V(D,G) = E_{x,y\sim P_{data}(x,y)}[logD(x,y)] \\ + E_{x\sim P_{data(x)}}[log(1-D(x,G(x)))]\end{aligned} \tag{3.1}$$

where x represents an inputting image and y represents a targeting image from data distribution P_{data}.

The capsule-based image-cGAN is constructed by involving the capsule modules in the image-cGAN framework. The generator G learns a mapping $x \rightarrow y$ to produce a synthetic image, whereas discriminator D learns to distinguish the ground-truth images from the ones generated by G while observing the inputting image. The overall framework is shown in Fig. 3.1. To better show the work flow of the framework, the images of image rendering are used as examples to represent the inputting, outputting and ground-truth images in Fig. 3.1. The capsule modules in detail are explained in the following sections.

3.2.2 Capsule-based Generator

Consisting of a pair of encoder and decoder, generator G only requires a single forward pass to generate an synthetic image with rendering effects. Initially, the encoder maps the input plain image into a latent space by stacking several 2-step stride convolutional layers, capturing the information that is independent of lighting effect. This is followed by a capsule block which learns precise spatial relationship. Then, the decoder synthesizes the images with lighting effects by several deconvolutional layers according to the learned feature from the capsule block.

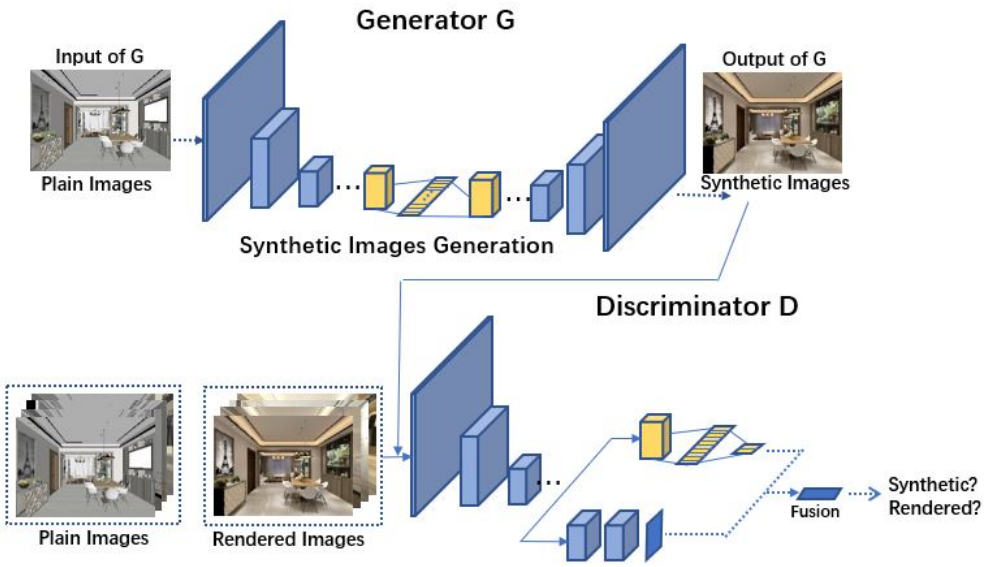


Figure 3.1: *The framework of the capsule based image conditioned generative adversarial network. Take the image rendering images as an example.*

The capsule block is composed of three layers, PrimaryCaps layer, FullConnectionCaps layer, and DePrimaryCaps layer, as shown in the yellow boxes in Fig. 3.2 (a). The PrimaryCaps layer is a convolutional capsule layer [179] that uses N convolutional filters to output a N-dimension capsule map for each channel. Multiple capsule maps with the same dimension of N are calculated. The FullConnectionCaps layer is a set of capsules fully connected with capsules from the PrimaryCaps layer with weights calculated by dynamic routing. The DePrimaryCaps layer is a set of capsule maps, which contain the same number of capsules from PrimaryCaps layer, fully connected to the FullConnectionCaps layer with dynamic routing again. The capsule maps of this layer are calculated by deconvolutional filters after stacking all of them according to the capsule dimension into the form of conventional CNN feature maps.

In this case, it is important that the contribution of every pixel matters and adjacent pixels should contribute the same way. This is achieved by applying convolution kernels with stride of 2 to remap the feature maps, instead of using max-pooling and upsampling layers. Batch normalization (BN) and ReLU nonlinear activation are applied after each convolutional layer and drop-out to avoid overfitting.

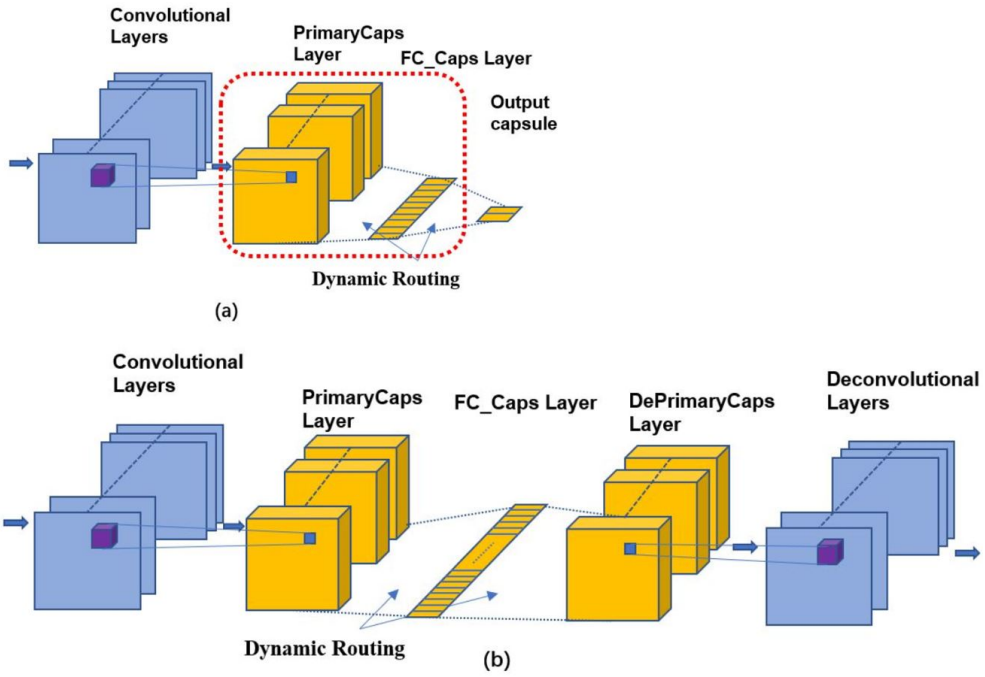


Figure 3.2: *(a) Capsule blocks in the Generator; (b) Capsule block in the discriminator. The red dotted area shows the original capsule structure introduced by [179].*

3.2.3 Capsule-based Discriminator

Discriminator D learns to distinguish whether an image x is from real rendered data or synthesized data from G by calculating the probability of being from real data. Denoting the distribution of plain images and rendered image as $x \sim P_p(x)$ and $y \sim P_r(y)$, ultimately the distribution of synthetic images from G ($G(x) \sim P_g$ conditioned on $P_p(x)$) should be equivalent to $P_r(y)$ when the training reaches the optimum.

After several layers of convolutional filters, the network is separated into two branches, the capsule based one and the patchGAN [78] based one. The capsule based branch is expected to distinguish the "real" and "fake" rendering effects based on information learned from the whole image emphasizing on spatial part-to-whole relationship. The patchGAN branch tries to distinguish the input of the discriminator at local level by observing each patch from the feature map of the last convolutional layer. The aim is to make the discriminator distinguish the images in both the global rendering effects and the local details.

The discriminator structure is shown in Fig. 3.1 and a closer view of the capsule branch is shown in Fig. 3.2 (a). The capsule branch consists of a Pri-

maryCaps Layer, a FullConnectionCaps layer and two output capsules, which represent the instances of "rendered" and "synthetic". The cost function is calculated as the margin loss [179] out of the two capsules. The patchGAN branch stacks three more convolutional layers and calculates the average patch-wise binary cross entropy classification loss on the last feature map. Specifically, the loss functions of the capsule branch, $\mathcal{L}_{caps\_D}$ and $\mathcal{L}_{caps\_G}$, for the optimization of D and G respectively are written as in Eq. (3.2) and (3.3)

$$\mathcal{L}_{caps\_D} = E_{x\sim P_p(x), y\sim P_r(y)}[\mathcal{L}_M(D_{caps}(x,y),[1,0]) + \mathcal{L}_M(D_{caps}(x,G(x)),[0,1])] \tag{3.2}$$

$$\mathcal{L}_{caps\_G} = E_{x\sim P_p(x)}\mathcal{L}_M(D_{caps}(x,G(x)),[1,0]) \tag{3.3}$$

where $\mathcal{L}_M$ is the margin loss as in [179], shown in Eq. (2.6), and D_{caps} is the output of capsule branch.

The loss functions of patchGAN branch, $\mathcal{L}_{patchG\_D}$ and $\mathcal{L}_{patchG\_G}$, optimizing D and G respectively, are written as in Eq. (3.4) and (3.5)

$$\mathcal{L}_{patchG\_D} = E_{x\sim P_p(x), y\sim P_r(y)} Aver(log(1-D_{patchG}(x,y)) + log(D_{patchG}(x,G(x)))) \tag{3.4}$$

$$\mathcal{L}_{patchG\_G} = E_{x\sim P_p(x)} Aver(log(1 - D_{patchG}(x,G(x)))) \tag{3.5}$$

where D_{patchG} is the output of patchGAN branch, and $Aver(\cdot)$ calculates the mean of the data. Thus, the overall discriminator loss functions, $\mathcal{L}_{disc\_D}$ and $\mathcal{L}_{disc\_G}$, optimizing D and G respectively, are written as in Eq. (3.6) and (3.7)

$$\mathcal{L}_{disc\_D} = \mathcal{L}_{caps\_D} + \lambda_1 \mathcal{L}_{patchG\_D} \tag{3.6}$$

$$\mathcal{L}_{disc\_G} = \mathcal{L}_{caps\_G} + \lambda_1 \mathcal{L}_{patchG\_G} \tag{3.7}$$

where λ_1 is the weight balancing the two branches.

3.2.4 Optimization

The adversarial training of GAN is to optimize the G and D iteratively using Eq. (3.1). To show the cost functions more explicitly for the proposed framework, the optimization of Eq. (3.1) is to minimize Eq. (3.6) and (3.7), shown in Eq.

Table 3.1: The training algorithm of the proposed capsule based image conditioned generative adversarial network.

Input:
Hyper-parameter setting
Training samples X_{input}, Y_{target}
for $i = 1; i <= training\_iterations$ **do**
Forward $G(X_{input})$ and $D(X_{input}, Y_{target})$
Update the D by updating the gradient of Eq. (3.8)
Update the G by updating the gradient of Eq. (3.9)

(3.8) and (3.9).

$$Obj_D = \arg\min_D \mathcal{L}_{disc\_D} \tag{3.8}$$

$$Obj_G = \arg\min_G \mathcal{L}_{disc\_G} \tag{3.9}$$

The detailed optimizing process is shown in Table 3.1. With each training sample, the framework is optimized iteratively with two steps, one for D and the other for G. When D or G is optimized, the other remain unchanged.

3.2.5 Implementation

The overall framework is implemented in python on the platform of Tensorflow [226]. The capsule layers are packed as modules, which can be applied in deep neural networks conveniently. The framework is trained with Adam optimizer [93]. Other hyper-parameters such as learning rate are set according to each specific experiment in following chapters.

3.3 Summary

The proposed capsule-based image-conditioned generative adversarial network is applied as a base framework of following works in Chapter 4, 5, 6 and 7. The framework is not simply applied on each application. It is re-constructed into different structures to better fit each application. Besides, for each application, new techniques are proposed with concerning the characteristics of each scenario.

Chapter 4

Image Rendering: Capsule-based Image Rendering for House Interior Decoration

4.1 Introduction of the Rendering Problem

Interior design concerns how to enhance the interior of a building for a healthier and more aesthetically pleasing environment. Traditional way for a interior designer to show his/her work to the customer is to design the interior 3D model with desired layout and decoration in professional software such as 3D Max. And then the designer use the software to render the model with desired lighting effect from various viewing angles to several static indoor images for ease of viewing from customer's perspective. The rendering step is very important as a plain 3D model without any lighting effect would significantly hinder the customers' appreciation of the functionality and aesthetics of the original design.

While being able to produce the best rendering result, the conventional way of interior effect rendering using professional software requires a significant amount of human labour work and very time consuming. Interior designers need to manually adding textures of the indoor objects, such as walls and furniture, and adjusting lighting effects by setting different light types, intensity and direction. This is usually done through a trial-and-error approach with a lot of rounds of trials. In addition, each round of trial usually requires 60 to 90 minutes for the software to render on an average PC. On average it takes about two

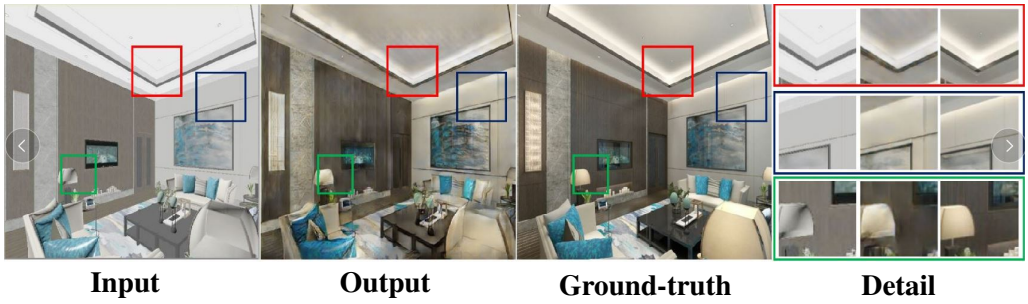


Figure 4.1: *Demonstration of my interior design effect rendering results. Details are shown on the right.*

working days for an experienced interior designer to render several images with desired effects using software.

In recent years, image generation, concerning synthesizing new images based on input and training data, has attracted tremendous attention in the research community. Representative works include synthesizing fake but realistic images using deep generative networks with various loss functions [47, 48, 78]. These approaches aim to translate the input image into the output with certain styles as target general images, instead of focusing on rendering effects of indoor images, hence disregard useful information such as lighting, layouts of walls and furniture, during the learning process. Another line of work focuses specifically on indoor images such as indoor navigation or VR/AR [206, 261], depth estimation [195, 206], or 3D to 2D rendering [197], etc. Despite focusing on indoor images, these works do not concern the rendering effect at all.

The goal is to generate a synthetic indoor image with rendering effect from a plain image rendered from a interior 3D model using professional rendering software. What makes a good rendering effect for interior design? While there is no right or wrong answers to this question, one can be sure that such knowledge has already been embedded in the existing professionally rendered images. In other words, a properly devised data-driven method should be sufficient to learn useful information such as relationship between lighting effect and light style/positions, how indoor layouts effect lighting effect and the colours of indoor objects such as walls and furniture, what is the best colouring effect for certain layouts and view angles. For the sake of notability, in this work *plain image* and *rendered image* are referred to as images without and with rendering effect from a rendering software. *synthetic image* is referred to as image generated by the translation

method.

This work devises a novel capsule-based conditional Generative Adversarial Network (cGAN) approach to synthesise images for interior design effect rendering, with prior domain knowledge of pairs of plain images and well rendered images. The proposed method can generate an image in less than one second automatically. The use of cGAN enables my framework to learn a better generator by simultaneously training a discriminator to distinguish the generated synthetic images from rendered images. Noting the advantage of using capsules [179] instead of scalar based neurons in deep learning networks, my approach incorporates capsules blocks inside both generator and discriminator to encourage strong part-to-whole relationship for better light effect. Hence, my network is able to capture a much better light effect compared with existing image translation techniques. In addition, a multi-way loss discriminator including both capsule blocks and conventional fully connected neurons (patchGAN [78]) is designed to simultaneously capture low and high frequencies during the training process. This ensures the generator can synthesize images with better rendering effects at all detail levels. To facilitate generating realistic lighting effect, a line preservation loss is introduced to constraint the layout of the plain image using line detection. Together with pixel level loss, the line preservation loss not only helps preserve the properties that are independent of lighting effect, but also improves the lighting effect along those lines which are crucial to the effect of interior design. As a result, images with more realistic and aesthetically pleasing rendering effect are synthesized from plain input images (see Fig. 4.1).

The main contributions are summarised as follow. 1) A cGAN is adapted for synthesizing rendering effect from a training set of plain and rendered images; 2) The capsule blocks are incorporated inside both generator and discriminator (multi-way) for more realistic rendering effects; 3) A novel line preservation loss is introduced to help preserve content layout of indoor images while improve lighting effect; 4) An extensive evaluation and analysis are performed to compare the proposed approach to several baselines on the Home Interior Design Effect Rendering Dataset, HIDER. The experiment results confirm that the proposed method produces synthetic images with much better rendering effects. At last, a dataset called HIDER is published to contribute to the research community for further exploration of image synthesis techniques. One publication is in ACCV,

"Capsule based image synthesis for interior design effect rendering" and another is in BRAIN2018, "Capsule based image translation network".

4.2 Pre-knowledge: Rendering and In-door Image Understanding

In the initial explorations of indoor image understanding, the majority of works have put effort in traditional segmentation and recognition tasks, contributing to applications like indoor navigation, virtual/augmented reality (VR/AR), robotic vision, and etc. [206, 261]. A key part to achieve these subsequent intention is to identify the object types, boundaries and learn the corresponding spatial relationship between them. Therefore, object depth estimation and RGB-D image based research take a great part in this field [195, 206]. To facilitate the data-driven based approaches, some of these works [140, 195, 197, 261] attempt to render 2D indoor images from 3D models, so as to enlarge the diversity of training dataset. This work differs in that the emphasis is put into the visual effect of synthetic images, targeting efficient interior design effect rendering, instead of modelling the visual variations caused by viewpoint changes [41, 141, 197]. Due to the high-level requirement of professionalism and extreme dependency of labour/computing cost, there is quite little literature discussing to synthesise interior design rendering images.

Zhdanov et al. propose a photon mapping method for image rendering, which allows to speed up the process of luminance calculation by storing less data in the photon maps [266]. Alexander et al. [95] present an efficient high-quality image segmentation method by analogising segmentation to image rendering. They develop a module that performs point-based segmentation predictions at adaptively selected locations based on an iterative subdivision algorithm. Thies et al. [210] use imperfect 3D content to produce photo-realistic renderings. The noise and incomplete surface geometry cause difficulties to synthesise high-quality images. They build a module to learn neural textures to improve the rendering effect. Aizawa et al. [2] build a model to tackle the viewpoint agnostic rendering problem. The model learns any-viewpoint image generation by manipulating a viewpoint in 3D space where the reconstructed instance shape

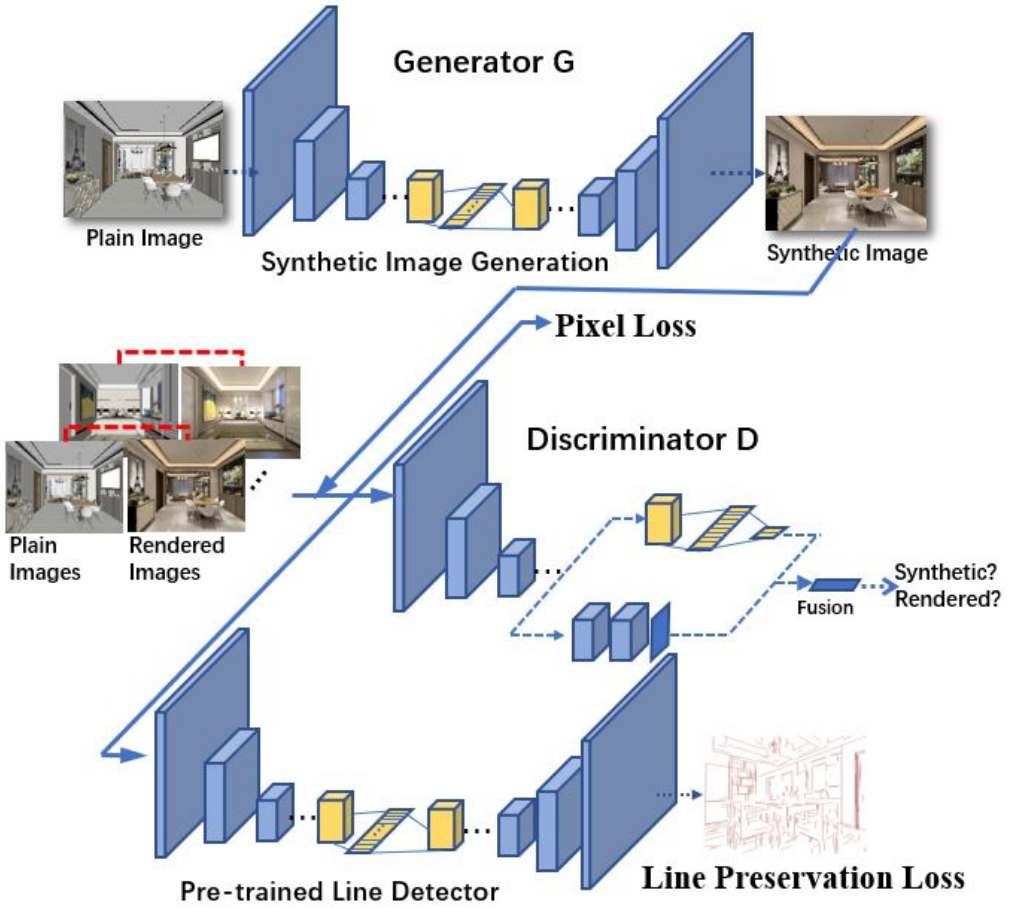


Figure 4.2: *The proposed image synthesis framework. The generator G, with capsule blocks built in, synthesizes the rendering effect images conditioned on the input plain images. A two-branch discriminator distinguishes the rendered ("real") and synthetic ("fake") images. The line preservation loss helps preserve the properties that are independent of lighting and colouring effect and improve the lighting effect along those lines.*

is arranged.

4.3 Methodology

The proposed capsule-based image-conditioned GAN introduced in Chapter 3 is applied to the image rendering. To facilitate realistic rendering effect, a combined loss is applied in the objective function to guide the training so that the framework can generate colouring and lighting effect while keeping the content/structure information, such as base colours, shapes, and edges, from the original plain images. In particular, generator G embedded with *capsule blocks*

attempts to encourage the condition in a part-to-whole manner. The *two-branch discriminator* with two branches forces G to produce images maintaining a consistent appearance (colour, light, texture, and style) distribution throughout the image while keeping each local patch as real as possible. A simple L1 loss in the image space, ensures the low-frequency difference between synthetic images and their targeting rendered images as small as possible. The *line preservation loss* encourages the preservation of the indoor layout by minimizing the L1 distance in a high-level feature space that embedding line and lighting effect information. The overall structure is shown as Fig. 4.2.

4.3.1 Line Preservation Loss

Unlike general images, the line shapes of the objects in indoor images, such as wall, decorations, or furniture, are massive and important in reflecting the overall and local lighting effects. Due to the variety of light directions, some line shapes are omitted and some "light lines"(lighting effect along the lines) appear. Therefore, a line preservation loss on G is introduced to improve lighting effects by preserving line shapes. In particular, besides generator G described in Section 3.2.2, another generator (with the same structure) is added as the line detector L that takes a plain image as input and outputs an image with lines that are important to effect rendering. The dedicated line detector L is specially trained on millions of indoor images [262] with the ground-truth generated using FastLine [105]. Compared to simply applying the state-of-the-art line detection techniques, the line detector can capture lines that appears in and are important to indoor images with rendering effects. The same discrimination loss introduced in Section 3.2.3 and L1 distance loss are used to model the mapping in L.

In particular, the line detector L is appended to G as the *Line Preservation Loss* formulated as $\mathcal{L}_{line}$ in Eq. (4.1). The *Line Preservation Loss* ensures the preservation of the line shapes of the synthetic image under the rendering lighting effects and guides the learning process in this way.

$$\mathcal{L}_{line} = E_{x\sim P_p(s), y\sim P_r(y)}[\|L(y) - L(G(x))\|_1] \tag{4.1}$$

where $L(\cdot)$ is the output of L and $\mathcal{L}_{line}$ is passed back to G without updating the parameters of L, while performing the adversarial training.

4.3.2 Objective and Optimization Functions

Previous research show that image cGANs benefit from extra supervision signals during the adversarial learning [78, 154]. Therefore, I use pixel loss to penalize the difference between $G(x)$ and the groundtruth of $y \sim P_r$, see Eq. (4.2). Instead of L2 distance, L1 distance is used as L1 works the best during the optimization in the work, especially when D finds improper distinguishable features.

$$\mathcal{L}_{pixel} = E_{x\sim P_p(x), y\sim P_r(y)}[\|y - G(x)\|_1] \tag{4.2}$$

By combining Eq. (3.6), (3.7), (4.1) and (4.2), the final objective functions, Obj_G and Obj_D, for G and D respectively are formulated as in Eq. (4.3) and (4.4).

$$Obj_G = \arg\min_G [\mathcal{L}_{disc\_G} + \lambda_2 \mathcal{L}_{line} + \lambda_3 \mathcal{L}_{pixel}] \tag{4.3}$$

$$Obj_D = \arg\min_D \mathcal{L}_{disc\_D} \tag{4.4}$$

where λ_2 and λ_3 are balancing weights to control the loss contribution from the line preservation and pixel loss.

4.4 Dataset - HIDER

There is few works to talk about the dataset dedicated to interior design effect rendering, which maps the plain to the rendered images. Therefore, this work proposes the Home Interior Design Effect Rendering dataset, HIDER, containing image pairs of plain and rendered images. 453 home interior design models using the software *3D Max* [6] are built. The models are designed for various rooms including living room, study room, bedroom, and dinning room, and various styles including modern, post-modern, European, and Chinese. $1-3$ pairs of images were rendered from a single model from different view angles. Various light types, illumination, and lit angles are also adjusted for best rendering effects. In total, 1174 pairs of 1300×939 plain and rendered images are generated in my dataset. The number of each category is shown in Table 4.1. A 512×512 version is prepared as well to benefit current state-of-the-art frameworks. Examples are shown in Fig. 4.3 and 4.4. The dataset is downloaded from the project website. https://yang-fei.github.io/tf-capsule-rendering/

Table 4.1: Sample numbers of HIDER.

	Modern	Post-modern	European	Chinese	Total
Living	143	207	129	194	673
Bedroom	57	138	48	56	299
Dining room	9	9	39	41	98
Study room	21	30	16	37	104
Total	230	384	232	328	**1174**

Image A: plain **Image B: rendered**

Figure 4.3: *A pair of examples from the HIDER dataset. Each pair consists of a plain image and a rendered image.*

4.5 Experimental Verification

4.5.1 Data Preparation and Implementation

Data Preparation The 512-sized version of the HIDER dataset is used to train the proposed network. The dataset is randomly separated into training and testing sets with a ratio 8:2. Prior to feeding the images into the network, data augmentation is adopted by performing cropping and mirror flipping on both plain and rendered images. In the experiments, to ensure each input image maintains majority visual content, the images are resized into 600×600 pixels and then cropped randomly into 512×512 pixels patches for data augmentation.

Implementation Details As demonstrated in Fig. 4.2, the architecture of G starts with eight 2-step stride convolutional layers, followed by the capsule block and the corresponding deconvolutional layers. In the capsule block inside G, 64 channels of 8-D convolutional capsule layers are used in both the PrimaryCaps and DePrimaryCaps layers, fully connecting to 64 16-D capsules in the FullCon-

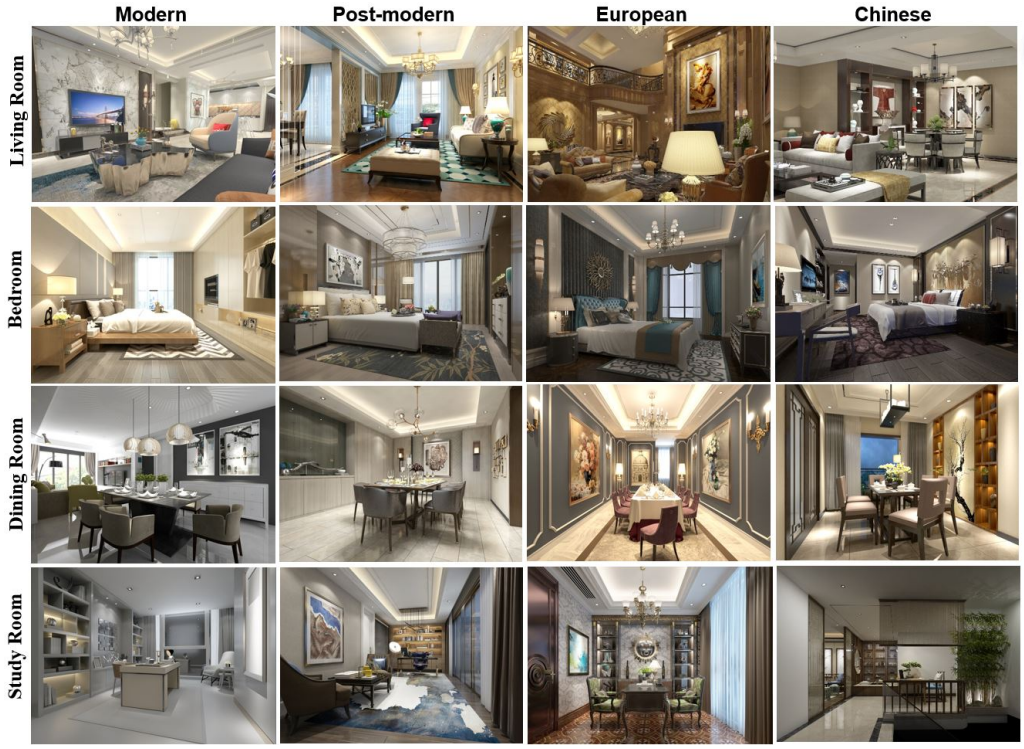


Figure 4.4: *Examples from the HIDER dataset in four room types and four styles.*

nectionCaps layer. Similar to the experimental setting in [179], I set the number of iterations for dynamic routing as three.

Discriminator D is constructed by three 2-step stride convolutional layers, followed by a two-branch stacking layers. The capsule branch consists of a PrimaryCaps layer with 32 channels of 8-D convolutional capsule and a FullConnectionCaps layer with 10 8-D capsules, outputting two 8-D capsules representing the probability of rendered image (real) and synthetic image (fake). The patchGAN branch stacks three convolutional layers and outputs a 1-D feature map with size of 30×30 for patch-wise discrimination.

Both G and D are trained iteratively according to objective Obj_G and Obj_D specified in Section 4.3.2. The line detector L and pixel losses only contribute to the training of G without updating the parameters of L. The loss balancing weights are empirically set as $\lambda_1 = 0.2$, $\lambda_2 = 10$, $\lambda_3 = 10$. I use Adam optimizer with a learning rate of 0.0001. All the experiment setups are trained for 200 epochs on the HIDER dataset with batch size 4. All the experiments were conducted in Tensorflow (python) under the same system environment using a NVIDIA GPU, GTX1080TI or TITAN X (Pascal).

Table 4.2: Quantitative evaluation results using various metrics.

Models	nrmse(L2)	L1	psnr	ssim
AE[229]	0.4135	0.1956	13.74	0.5889
VAE[158]	0.4293	0.2092	13.23	0.5907
U-net[175]	0.3327	0.1552	14.17	0.6537
cGAN[78]	0.3160	0.1626	14.71	0.6271
cGAN+Caps(my method)	0.2779	0.1273	15.85	0.7184
cGAN+Caps^2(my method)	0.3081	0.1375	15.08	0.6440
cGAN+Caps+line(my method)	**0.2063**	**0.1073**	**16.91**	**0.7356**
cGAN+Caps^2+line(my method)	0.2600	0.1160	15.77	0.6542

Table 4.3: Quantitative evaluation results using FCN-score.

Models	Per-pixel acc.	Per-class acc.	Class IOU
AE[229]	0.8728	0.2849	0.2081
VAE[158]	0.8912	0.2615	0.2231
U-net[175]	0.9031	0.3276	0.2737
cGAN[78]	0.9126	0.3901	0.3200
cGAN+Caps(my method)	**0.9202**	0.3653	0.3056
cGAN+Caps^2(my method)	0.9157	0.3733	0.3087
cGAN+Caps+line(my method)	0.9166	0.3807	0.3105
cGAN+Caps^2+line(my method)	0.9123	**0.4308**	**0.3368**

4.5.2 Performance Comparison and Analysis

The approach is evaluated by analysing the performance of each proposed component and comparing with the state-of-the-art methods. The comparisons include image Autoencoder(AE) [229], Variational Autoencoder (VAE) [94], U-net [78, 175], and cGAN (pix2pix) [78]. For the proposed network architecture, I perform ablation studies to interpret the performance of line preservation term, capsule block term, together with the discriminator using a single branch against the discriminator using two branches, respectively. The experimental model settings are denoted by [$cGAN+Caps$], [$cGAN+Caps+line$], [$cGAN+Caps^2$] and [$cGAN + Caps^2 + line$], where [$cGAN$] is the base image-cGAN model, [$* + Caps$] means the network using capsule blocks, [$* + Caps^2$] shorts for the two-branch-discriminator implementation, and [$* + line$] represents the use of line preservation loss. In all these settings, I keep the pixel loss since this is not considered as the contribution.

Quantitative Evaluation Quantitative evaluation is challenging in the field of

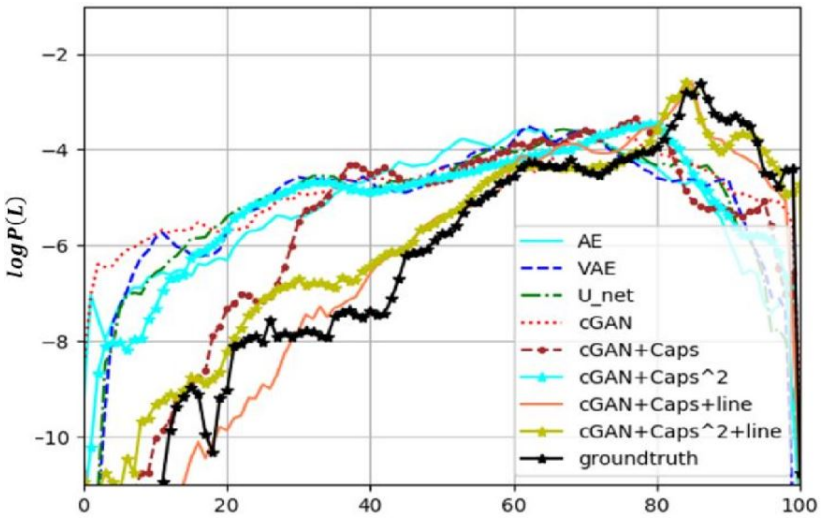


Figure 4.5: *Colour distribution of L channel in Lab space.*

image synthesis. In this experiment, the proposed generative models are evaluated from three perspectives: 1) using mean-squared error (mse) and L1 distance to measure the similarity between synthesized and target image at the pixel level; 2) adopting the Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity index (SSIM) metric to measure the quality of synthesized image at signal level; 3) applying FCN-score to measure the discriminability of the generated image at semantic level, as suggested by [78]. For the FCN-score, the pre-trained FCN-8s [30] classifiers are used to to segment both the rendered and the synthetic images, and compute the per-pixel accuracy, per-class accuracy and class IOU.

Tables 4.2 and 4.3 show the evaluation results. Note that baseline AE, VAE and U-net are generative models without discriminator, while the cGAN [78] having a patchGAN discriminator contributing to high frequency visual content. By comparing cGAN [78] against U-net [175], I can observe the trade-off between pixel and semantic level metrics results, where cGAN shows much better FCN-score results with a lower ssim and L1 scores.

The quantitative results show that all of the proposed models outperform the three generative models in both tables and the $cGAN + Caps^2 + line$ model boosts the per-class accuracy and class IOU scores greatly compared to previous best cGAN based [78] model, indicating that the proposed approach can simultaneously encourage low and high frequency correctness throughout the image at both pixel and semantic level. In addition, the adoption of capsule blocks alone can boost the pixel and signal level scores while maintaining good results

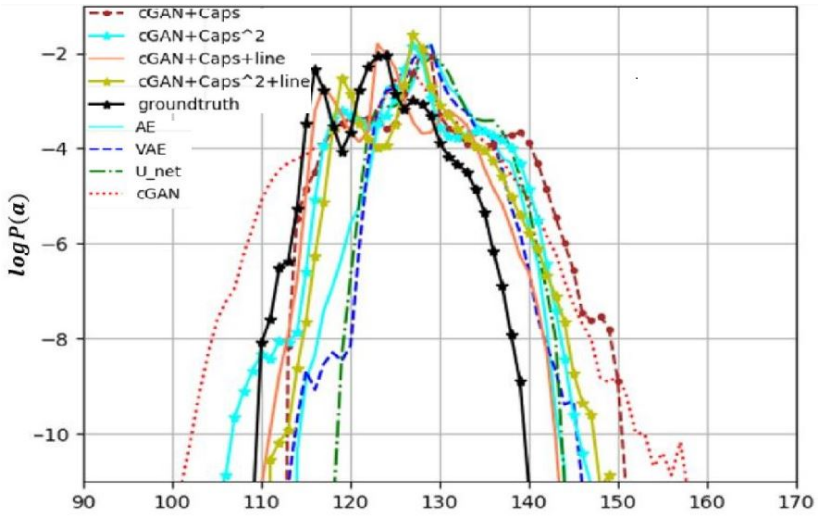


Figure 4.6: *Colour distribution of a channel in Lab space.*

Table 4.4: Histogram interaction of Lab colour space against ground-truth.

Models	L	a	b
AE[229]	0.7358	0.6590	0.6485
VAE[158]	0.7307	0.6416	0.6381
U-net[175]	0.7274	0.6561	0.6437
cGAN[78]	0.7301	0.5425	0.6565
cGAN+Caps(my method)	0.7228	0.6636	0.6638
cGAN+Caps^2(my method)	0.7368	0.7227	0.6862
cGAN+Caps+line(my method)	**0.8255**	**0.8127**	**0.7745**
cGAN+Caps^2+line(my method)	0.8140	0.7561	0.7319

in FCN-Score. The line preservation loss can further improve the result at both the pixel and semantic level by comparing the proposed model with/without the line preservation loss. Compared with a single capsule-block-branch discriminator, a decrease in pixel/signal level metrics while a performance boosting in semantic level metric can be observed with if the two-branch discriminator is applied. This is because the two-branch discriminator also penalize the synthetic images in a patch-wise fashion, guiding the generator producing images with fine details. The results is consistent with the comparison between U-net and cGAN [78]. Combining all quantitative results, it can be seen that the $cGAN+Caps+line$ model outperforms all the other methods in the pixel/signal level, while the $cGAN + Caps^2 + line$ model obtains a better performance in a semantic level.

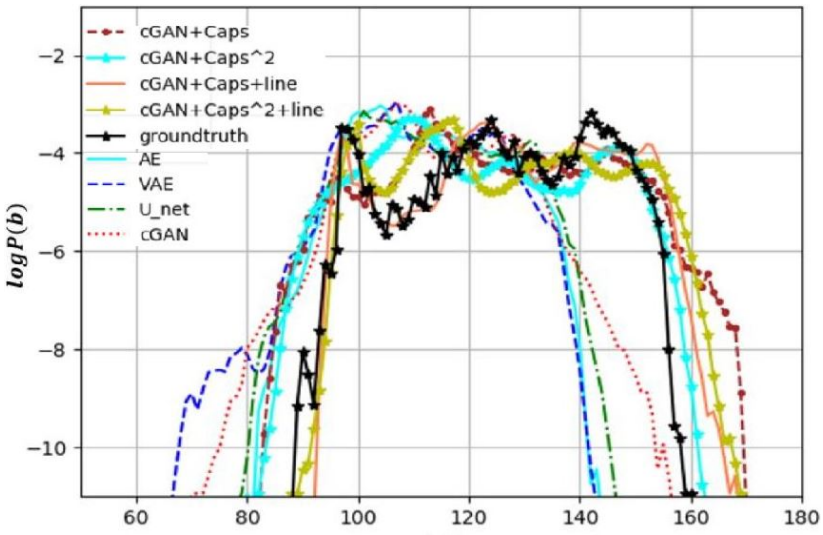


Figure 4.7: *Colour distribution of b channel in Lab space.*

Lighting Fidelity Evaluation Lighting fidelity plays an important role in synthesizing images, especially for effects rendering. To further investigate the performance from a visual perspective, the colour distribution is checked in the Lab space and compute the histogram intersection scores between the synthetic and ground truth images. The result is shown in Fig. 4.5, 4.6 and 4.7. The ground-truth distribution are shown as black lines. It can be seen that both the $cGAN + Caps + line$ and $cGAN + Caps^2 + line$ models obtain very similar distribution compared with the ground-truth in all channels, and they outperform all the other models greatly in the histogram score results, shown in Table 4.4.

The colour distribution shows the use of capsule blocks encourages ground-truth alike lighting effect while avoiding grayish pattern caused by the averaging attempt guided by uncertainty in AE, VAE and U-net. Unsurprisingly, the lighting effect encouraged by the line preservation loss can also be observed in Fig. 4.5. In Fig. 4.6 and 4.6, It can be seen that the pure generative model based methods (AE, VAE and U-net) tend to produce a narrower colour distribution in the green-red space. On the other hand, the cGAN [78] tends to produce much wider distribution, resulting unrealistic visual pattern. In addition, models from AE, VAE, U-net, and cGAN [78] all tend to synthesize images with blue (cool-toned) pattern closer to plain images while producing less red alike (warm-toned) pattern that is closer to ground truth images.

Rendering Effect Fidelity Evaluation It is acknowledged that the quantitative

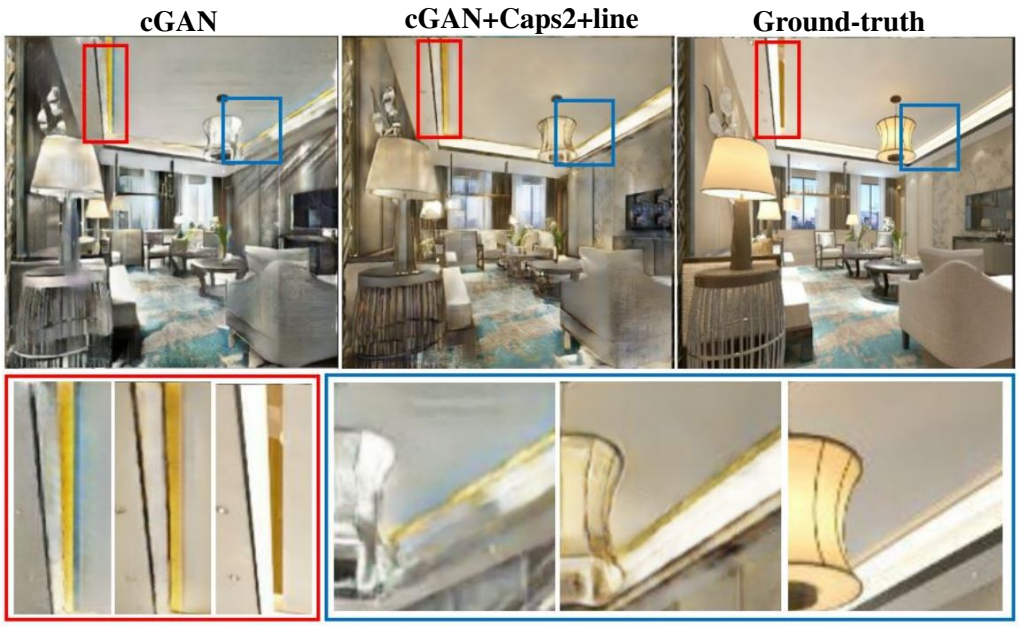


Figure 4.8: *A detailed visual comparison of my result with the basic cGAN framework.*

results and colour distribution is not enough to measure whether the synthetic images contain aesthetically pleasing rendering effect. Therefore, a more comprehensive evaluation is provided with visual analysis to match the visual sense of human perception.

The synthesized images from different methods are shown in Fig. 4.10 and the visual details are demonstrated in Fig. 4.8 and 4.9, starting with input plain images and ending with the ground truth images. As shown in the Fig. 4.10, AE, VAE and U-net tend to generate greyish and blurry images because of the absence of discriminator, resulting in relatively low pixel-wise distance but hard to recognize structures at semantic level. The results are unacceptable in the context of interior design effect rendering. Compared to pure generative models, the output of cGAN [78] is able to preserve more local details by using patchGAN discriminator and hence producing sharper images. The similar visual effect is also observable in the proposed models, where the two-branch discriminator encourages more detailed local features. However, the overall colouring effect of the results from cGAN [78] tend to be distorted and cool-tone instead of warm-tone. For the cGAN [78] output, both the background wall of the TV in the second example and floor from the third example seem locally real but appear abruptly from the whole image point of view. This is because, while being able

to generate fine details, cGAN lacks the capability of capturing precise spatial relationship, and hence tends to produce incoherent rendering effects.

In contrast, the proposed capsule-based method ensures capturing the part-to-whole relationships among the whole images. It is very obvious that all of the proposed capsule block based output obtain a closer colouring compared with the ground truth image. Although both $cGAN + Caps^2$ and $cGAN + Caps^2 + line$ models contain the patchGAN branch in the discriminator that might bring in local artifacts, the part-to-whole relationships tend to guide whether if a local lighting effect is reasonable globally. For example, unlike the cGAN [78] result in the second example, the artifacts in TV is kept in my results while fake lighting effect on the background wall is restricted. Similarly, the floor texture of the third example is maintained consistent throughout all of the proposed methods' output.

The visual results also demonstrate that the line preservation loss not only helps preserve the properties that are independent of rendering effect, but also improves these effects along the lines. It is clearly observable that light and shade is more distinctive and the lighting effect is well captured in the results. For example, in the images synthesized by $cGAN + Caps + line$ and $cGAN + Caps^2 + line$, the chandelier is sharper and the lighting effect along the light belt is more realistic in all three examples. In addition, the lighting reflection is reasonably generated around the mirror and on top of chair/table/floor in the third example.

4.6 Summary

This work brings the idea of image translation to automatically synthesise rendering effect for interior design with the need for manual rendering from software in a trial-and-error manner. Towards this goal, a novel capsule-based image-conditioned generative adversarial network with a two-branch discriminator and a novel line preservation loss are introduced. A dataset, HIDER, is proposed to evaluate the proposed method. The extensive experiments show that the framework is able to generate more realistic and aesthetically pleasing rendering effect at both detail and semantic levels.

I have two publications about this work, "Capsule based image synthesis

for interior design effect rendering" at Asian Conference on Computer Vision in 2018 and "Capsule based image translation network" at IET Doctoral Forum on Biomedical Engineering, Healthcare, Robotics and Artificial Intelligence in 2018. A journal version is under review.

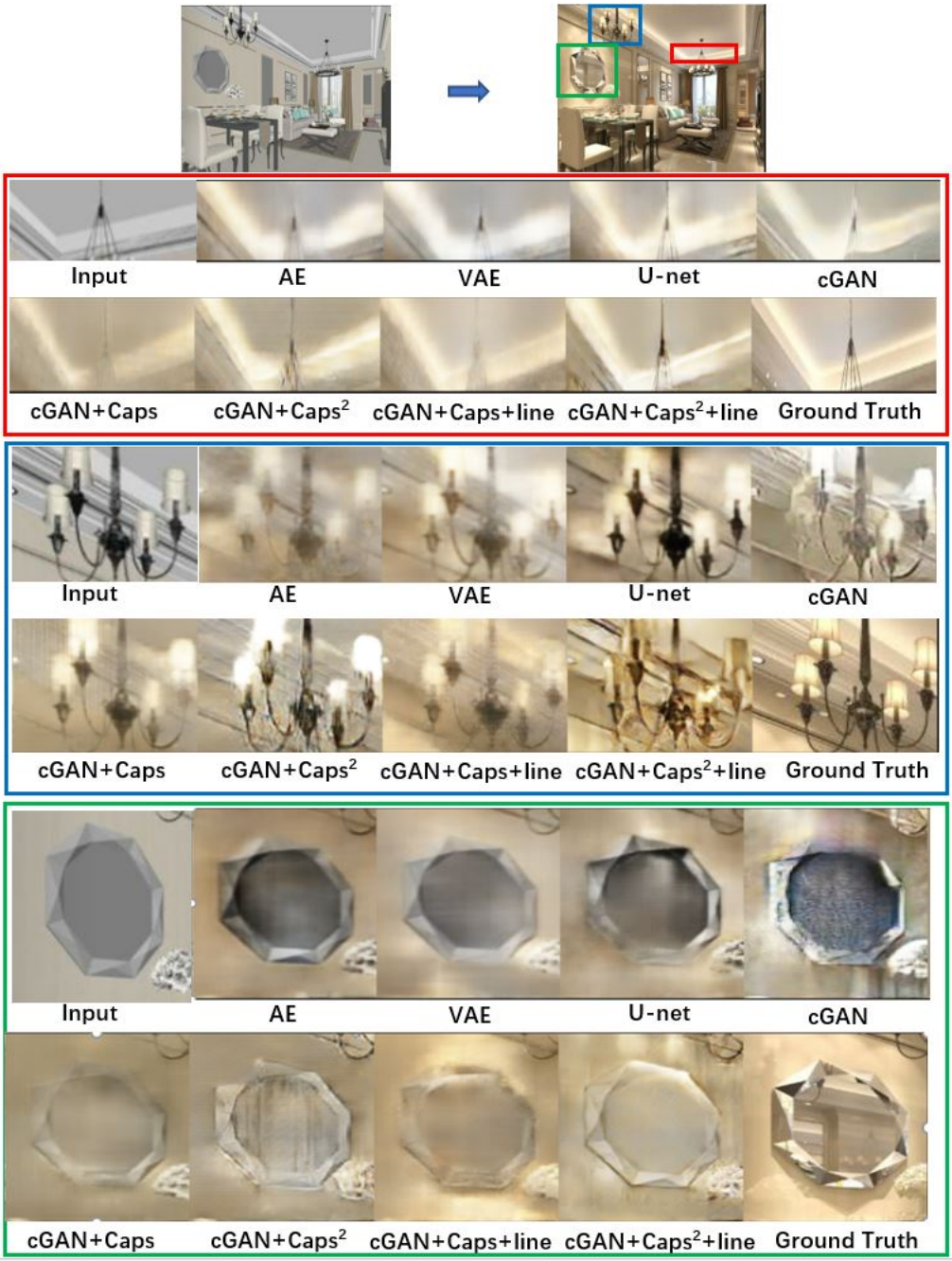


Figure 4.9: *A detailed view of the visual results and various baselines.*

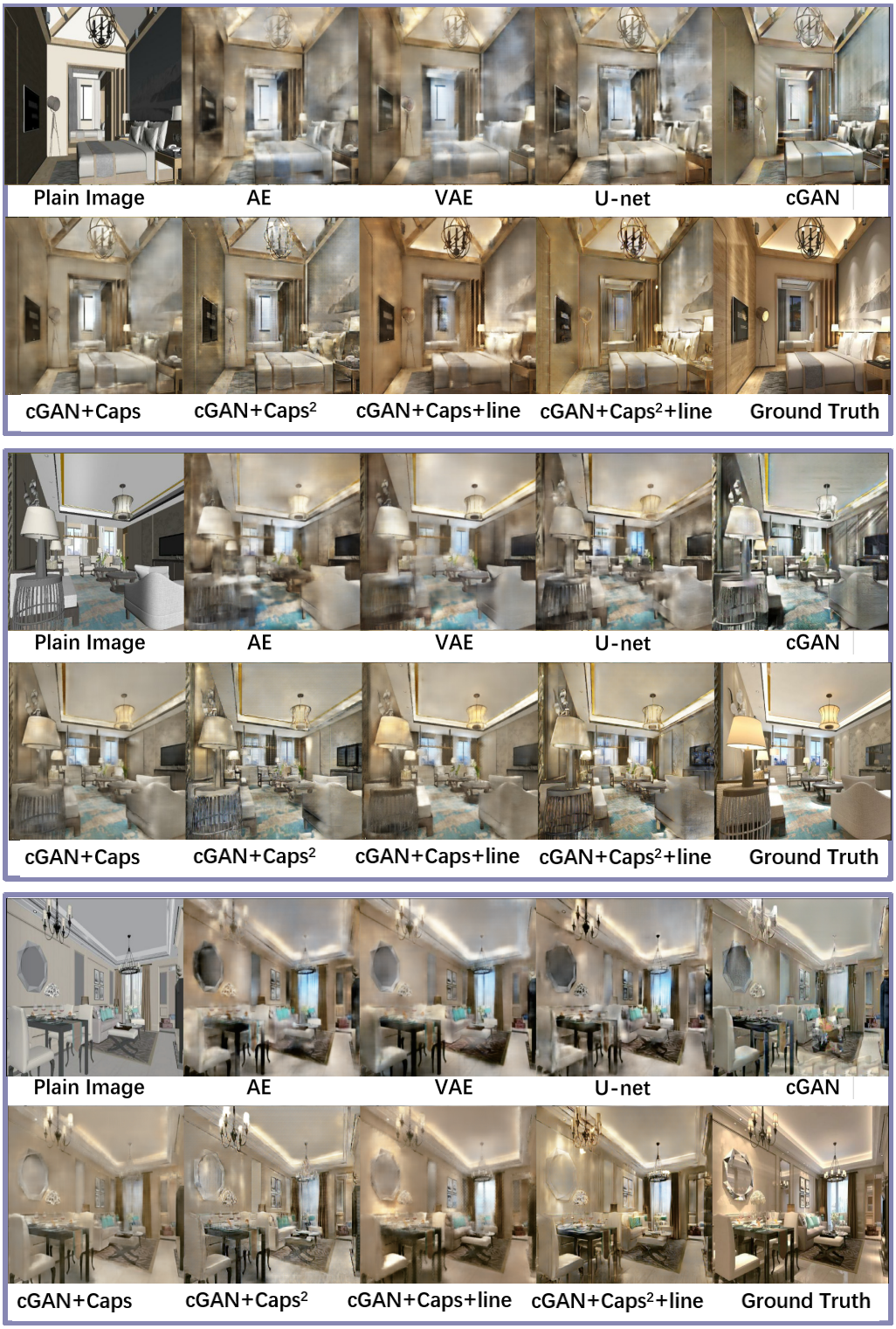


Figure 4.10: *Visual comparison of the proposed approach and various baselines.*

Chapter 5

De-raining: Rain-Component-Aware Capsule-GAN for Single Image De-raining

5.1 Introduction

Image de-raining aims to recover the clear content from a rainy image, which can be regarded as a special type of image translation by defining the rainy and rain-free domains as the source and target domains. Comparing with image rendering (Chapter 4), image de-raining is different in terms of image contents, essential features, and translating purpose. Image rendering is sensitive to lights and other indoor objects, on which the light effect is more important than local texture to provide the stereoscopic feeling, while image de-raining focuses on the rain noise of outdoor images, which concerns much about the consistency of image contents in detail. Therefore, new techniques are proposed in this chapter instead of applying the image rendering framework simply. A successful de-raining model works perfectly to identify the rain component (rain drops or rain streaks) and generate the clear image content by removing rain noise.

Outdoor images will deteriorate due to weathers or air conditions, such as haze, fog, or rain. The irradiance from the scene point to the camera under such conditions has gone through scattering and deflection [142, 155, 243]. Such

phenomenon is especially prominent in the images with rain, and leads to scene obstruction and blur that can significantly reduce image quality. It will also degrade the performance of visual systems, such as object detection [27], scene recognition [131], face recognition [227] and etc., as most of the state-of-the-art computer-vision techniques heavily rely on the low level details of the input image and the mid level cues built upon it.

Image de-raining can be regarded as a preprocessing step to fill in the missing details and improve the overall quality of input images before feeding to other computer-vision models. However, de-raining is a challenging under-constrained problem because of the variability of rain features and the hardness to identifying rain components from image content. Over the years, a lot of efforts have been put into the research of single image de-raining. Traditional methods consider rain components as white streaks and aim to separate it in the frequency domain, but these methods are only effective to certain types of rain streaks [51, 132, 174, 245]. Recent deep-learning-based methods [44, 45, 113, 239, 248, 277] attempt to generate de-raining images by constructing various artificial neural networks with different structures, cost functions and training strategies. While still less than optimal, the results shed light on the promising direction using deep neural networks. An example of the de-raining results obtained by our model is shown in Fig. 5.1

Most existing work focuses on generating rainless images by minimizing the visual differences between the generated images and the target clean images, which only concerns the image content without supervising the learning of rain features. In [113], Li et al. explored the feasibility of deriving the rainless image by subtracting the rain component map from the rainy image. But it is inaccurate to obtain the image content by the simple subtraction, since the rainy image is not simply the summation of the two, but the coverage of the rain over the content with nonuniform transparency. To identify the rain patterns more accurately, a Rain-Component-Aware (RCA) network is proposed to guide the training of the overall de-raining framework. The proposed RCA network could well remove the rain components, as demonstrated later in the experiments.

As a special type of image enhancement, de-raining can be approached using deep Convolutional Neural Networks (CNN), given their successes in applications including de-noising [106], de-hazing [107], super-resolution [263] and

colourization [78]. Traditional CNNs usually use relatively small kernel size and have limited depth. The networks with fewer layers fail to extract adequate global information for rain removal. In addition, traditional techniques for spatial feature extraction, such as pooling, do not focus on the part-to-whole relationship that has been shown critical for image enhancement [179, 235]. The part-to-whole relationship indicates the relations between partial details and global objects, which can be learnt better by the capsule structure [179] than CNN, as demonstrated in [235]. In this work, capsule units are applied to de-raining, to better capture the part-to-whole spatial relationship, and hence identify the rain components more accurately.

The Image-conditioned Generative Adversarial Network (image-cGAN) generates target images by learning the patterns encoded in images with discriminative loss. In this chapter, a two-branch model based on image-cGAN framework is devised. For the branch of the image content, the supervision signal from the discriminator is used to guide the generation of rainless images while minimizing the pixel loss and the SSIM (Structure SIMilarity) loss [270]. For the RCA branch, a RCA network is built to capture the characteristics of rain components, which provides effective optimisation signals to the generator. We refer the proposed RCA guided capsule-based image-conditioned generative adversarial network as **RCA-cGAN**.

Experiments are conducted on both synthetic and real-world rainy images. Comparing with existing image de-raining models, the proposed RCA-cGAN achieves a significant performance improvement on both objective evaluation and subjective visual inspection. The rain components are better removed and more details of the image content are preserved.

Contributions: The contributions of image de-raining are summarized as follow. 1) I design a de-raining framework with two optimisation branches for the image content and the rain components. The de-raining performance benefits from the combination of the two optimisation branches within one joint framework. 2) A RCA network is proposed to better capture rain patterns from rainy images. The proposed RCA network is incorporated into the de-raining framework as the RCA loss to effectively identify the rain components from rainy images to guide the generation of rainless images. 3) Capsule units are adopted in the de-raining work to better model the part-to-while information in the rainy

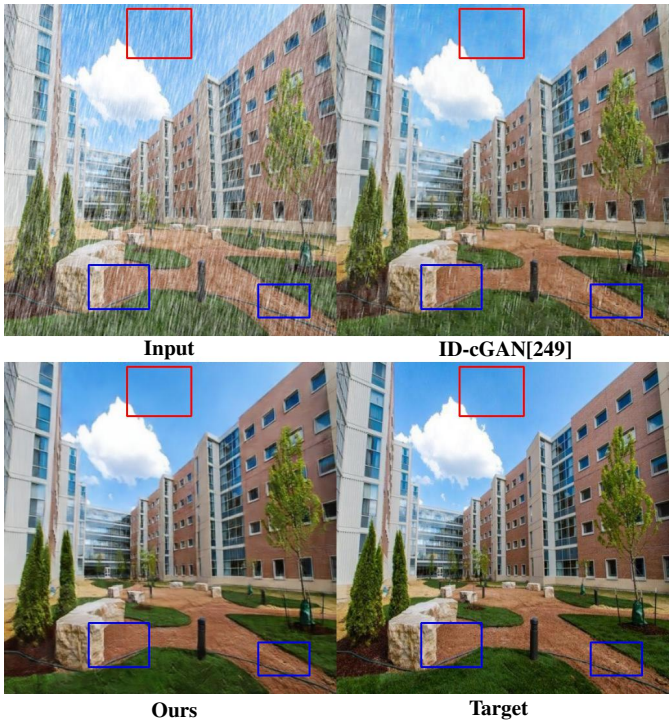


Figure 5.1: *De-raining results on Test1 of synthetic dataset. The right-bottom is the ground truth (target).* **Best viewed on screen.** *My model produces the most smooth images on the "sky" content. The regions pointed by the blue rectangles contain more content and our model can better restore the details.*

image, and hence better remove the rain components. 4) An extensive evaluation and analysis is performed to compare the proposed approach to several state-of-the-art deep learning techniques. The experimental results show that the proposed method produces rainless images of significantly better quality using either synthetic or real-world rainy images. I have a paper submitted to *Pattern Recognition* which is under review, "Rain-component-aware capsule-GAN for single image de-raining".

5.2 Pre-knowledge

5.2.1 Rain Removal

Video-based Rain Removal

A rainy image can be considered as a blending of a clear content layer and a rain component mask. The temporal information between adjacent frames makes

it easy for a de-raining model to identify the rain components and recover the clean contents [45, 46, 113, 181, 211]. Garg et al.[46] analysed the visual effects of rain on imaging systems and developed a correlation model to capture the dynamic characteristics of rain. Zhang et al. [260] considered the temporal and chromatic properties of rain together. Then, researchers tried to explore new methods from indirect aspects, except for the RGB values. Barnum et al. [7] found the dynamic weather has a predictable global effect in frequency space and built the detection model in frequency space. Varun et al. [14] explored de-raining from the histogram of orientation of rain streaks. Zhang et al. [260] considered the temporal and the chromatic properties of rain together. Kim et al. [91] detected rain streaks more accurately using more temporal correlation of adjacent frames and removed them by employing a low-rank matrix completion technique. Readers may refer to [211] for a more comprehensive review on video de-raining.

Single Image Rain Removal

Single image de-raining is to recognize the rain components and recover the clean image content from a single image only without introducing any artifacts. It is more challenging than video based, due to the lack of temporal information between frames and the spatial information from other neighbouring frames. After years of the research, researchers moved the attention to image-based de-raining. Kang et al. [87] developed the first single image de-raining framework, which decomposed the rainy image into different frequency components and identify the rain streaks against the image content using sparse coding. The methods of sparse representation were developed by Sun et al. [199] with an incremental dictionary learning strategy, Huang et al. [72] with a self-learning mechanism, Luo et al. [245] with a discriminative approach, Son et al. [194] with a shrinkage-based technique, and Chen et al. [20] with a hybrid feature set. But the sparse coding methodology is likely to introduce strip-like artifacts and over-smooth the image content through handcrafted representations [129, 239]. Apart from the sparse coding methodology, Chen et al.[25] built a generalized matrix-ranking model to characterize the appearance of rain streaks. Kim et al. [90] performed an adaptive non-local means filter on the detected rain streak regions. Li et al. [114] proposed a layer-prior approach for de-raining based on the

Gaussian Mixture Models(GMM), where the priors for both the background and the rain streaks layers are predicted. And Du et al. [35] separated the rain with the background in the gradient domain.

Methods based on deep learning have demonstrated their effectiveness in single image de-raining [45, 239]. Yang et al. [239] combined the detection and the removal into a joint network by proposing multi-branch of convolutional layers. Fu et al. [45] developed a "deep details network" for removing naturally high frequency rain components. Later, Li et al. [113] built a recurrent structure to detect rain streaks in multiple stages, known as RESCAN. A similar idea was adopted by Ren et al. [169]. The multi-stage framework is effective to remove rain streaks layer by layer [113], which have been demonstrated in [28, 80] as well. Zhang et al. [248] applied multiple dense blocks to estimate rain density in different scales and efficiently removed the corresponding rain streaks using the estimated rain density, known as DID-MDN. The final refinement process in [248] renders the images with enhanced visual quality but with information loss on local texture. Du et al. [36] built separated modules to learn the density information.

Indeed, the de-raining translator, or named generator, could be followed by a discriminator to form the cGAN structure. Zhang et al. [249] developed the ID-cGAN to generate clear images directly and a discriminator to distinguish the clean and rainy images. Their cGAN only considers the perceptual loss, which could produce a visually satisfactory image, but could not fully capture the characteristics of rain streaks. The rain streaks may still be visible after de-raining. Besides, Bi et al. [12] implemented the multi-scale structure and the attention module into the cGAN framework for de-raining. Jin et al. [81] proposed an asynchronous interactive GAN to optimise the de-raining process. In this work, we propose an entirely different structure by incorporating capsule units into both the generator and the discriminator. The proposed RCA loss is more effective than perceptual loss for de-raining as well.

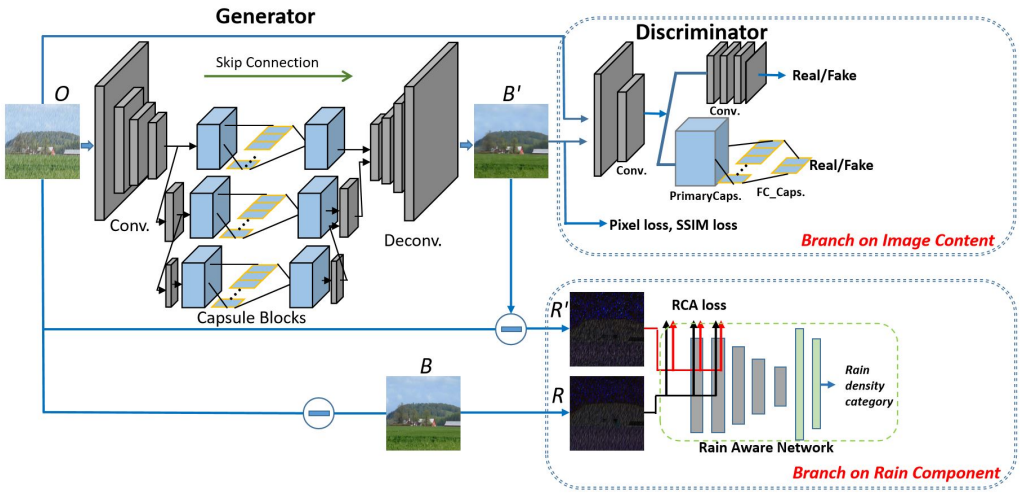


Figure 5.2: *The overall framework of the proposed capsule-based de-raining model. O and B represent the rainy and clear image from the dataset. $R^{'}$ is the mask of predicted rain components. $B^{'}$ is the de-raining image. The generator (G) and the discriminator (D) are trained iteratively. The rain aware network is used to calculate the RCA loss to optimise G.*

5.3 Proposed Method

5.3.1 Overall Framework - RCA-cGAN

The overall structure of RCA-cGAN is shown in Fig. 5.2. De-raining is considered as a procedure of image translation from the rainy image O to the clean image B. Image-cGAN is suitable to achieve the translation task. The generator G takes O as input and generates the clean image $B^{'}$ to fool the discriminator D, whose responsibility is to distinguish B and $B^{'}$ as real or fake.

A rainy image O is regarded as the composition of a rain component map R and a clean image B, i.e.

$$O = R + B, \tag{5.1}$$

which is stated in the form of Eq. (5.2) as well.

$$R = O - B \quad or \quad B = O - R. \tag{5.2}$$

Thus, existing de-raining works [113, 170, 248] optimise the frameworks by minimizing the cost functions on either B or R, noted as $\mathcal{L}_B$ and $\mathcal{L}_R$ respectively. B is obtained according to $B = O - R$ when $\mathcal{L}_R$ is optimized. However, $\mathcal{L}_B$ is equivalent to $\mathcal{L}_R$ only when the optimisation process is ideally achieved

without visual difference affected by $\mathcal{L}_B$ and $\mathcal{L}_R$. In this work, to separate the rain components and image content more accurately, a two-branch-based optimisation framework is developed with $\mathcal{L}_B$ and $\mathcal{L}_R$ being optimized at the same time, as

$$\mathcal{L} = \mathcal{L}_B + \mathcal{L}_R. \tag{5.3}$$

The image content branch consists of pixel loss, SSIM [270] loss and the discriminative loss. The discriminator D aims to distinguish the fake clean image $B^{'}$ and the real clean image B. Different from the conventional D, the discriminator used in this work is constructed with capsule units to further improve the discriminant ability. Indeed, the capsule units are constructed in both G and D to improve the ability of capturing the rain characteristics in a global view. This will be discussed in detail in the next section. After iterative training of G and D, a well trained G is able to generate realistic rainless images so that even a well trained D is hard to distinguish whether it is real or fake. The overall framework is trained by iteratively minimizing the loss of G and D.

To optimise the rain component branch, a **R**ain **C**omponent **A**ware (**RCA**) loss is introduced on the synthesized rain component map $R^{'}$ and the ground truth R by forwarding them through the pre-trained network, named RCA network. The rain components and the image content should be presented by different features at both pixel and semantic pattern levels. The RCA network to learn the rain feature from rain component map R works better than that to learn it from the rainy image O. The RCA loss is used to better remove rain components, which is pre-trained on the rain component maps (rain residuals) of rainy images to estimate the rain densities.

Therefore, based on the structure of image-cGAN, the RCA-cGAN is proposed, shown in Fig. 5.2. To generate a clean image $B^{'}$ (fake image), the learning of the image translator (also regarded as the generator G) is supervised in two branches, the image content branch and the rain component branch. The former aims to improve the image quality after de-raining while the latter focuses on capturing the rain characteristics to better remove the rain. The joint optimisation provides a better supervision signal and guides the training of G in a more effective and efficient way.

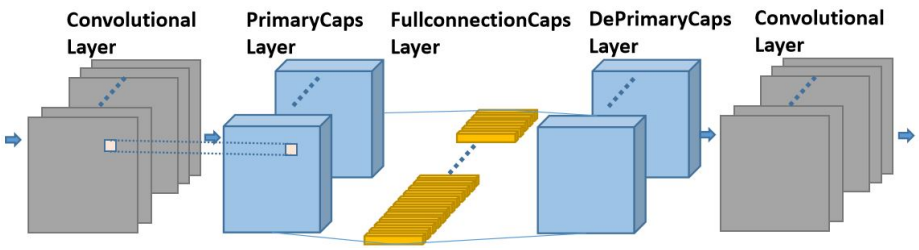


Figure 5.3: *The capsule layers include PrimayCaps, FullconnectionCaps and DePrimaryCaps layers, which can implement the capsule units into convolutional neural networks.*

5.3.2 Generator with Capsule

An end-to-end image-cGAN framework with capsule structure embedded in both generator and discriminator is introduced by building three kinds of layers, PrimayCaps, FullconnectionCaps and DePrimaryCaps, shown in Fig. 5.3. The detailed structure of capsule layers are introduced in Chapter 3. Compared with the implementation of DePrimaryCap in [235], the routing process within this layer is removed to save computation and in this way more capsule units could be applied to model the rain components. In the generator, three capsule blocks are placed within the 4×4, 8×8 and 16×16 layers of the convolution-deconvolution network, shown in Fig. 5.2. The kernel stride of each convolution/deconvolution filter is set as 2 so that the layer maps are encoded/decoded into size by 0.5/2. The skip connections [78] are implemented on the corresponding layers.

5.3.3 Discriminator with Capsule

In this work, a capsule-embedded D is placed in the image content branch. It is hard to discriminate a single rain drop from the adjacent local region, but the rain components from the whole image construct a rain pattern that could be identified. Similar rain drops may present different characteristics on different contents, as shown in Fig. 5.1. The capsule-based discriminator distinguishes the real/fake images at both the local and global level, as the capsule units could well encode the part-to-whole relations. In Fig. 5.2, the discriminator consists of two sub-branch, a capsule-based and a patchGAN [78]. The capsule sub-branch contains a PrimaryCaps and two FullConnectionCaps layers, with the second FullConnectionCaps layer having two capsule units to represent “real” and “fake” rainless images. The patchGAN sub-branch contains only convolutional layers,

of which the last one produces a single-channel map. As demonstrated in [78], the patchGAN-based discriminator encourages the generator to produce sharper details, because of its sensitivity to local details. The two sub-branches share two convolutional layers, as shown in Fig. 5.2.

Formally, the loss function of the discriminator hence consists of the capsule part and the patchGAN part. The formulation is shown in Eq. (3.6).

5.3.4 Rain Component Aware Loss

In the proposed framework, a rain-component-aware branch is designed targeting at capturing the rain characteristics and removing them later on. The rain components vary in terms of the shapes, direction, density and etc., but clearly have some patterns which are significantly different from the image content. The variety of rain pattern is mainly caused by rain density, e.g. heavy rain contains dense and thick rain streaks, while light rain is represented by small white marks. Since image features can be learned by networks through task-oriented pre-training [82], rain patterns can be learned in a similar way. In this work, the RCA branch aims to learn the rain feature from the synthesized images effectively and back propagated the information to the generator. A RCA network is proposed to learns rain features from the rain component maps R, which is implemented in the framework as the RCA optimisation branch.

Training a RCA Network

To acquire the ability of effectively extracting rain features, the RCA network is pre-trained as a classification task on a rain dataset with 4 classes ($\{$no-rain, light rain, medium rain, heavy rain$\}$). Various kinds of rain (rain patterns) need to be identified if we want to predict the rain density accurately. The training dataset consists of 12,000 synthesized rainy images with different rain orientations and scales, similar as in [248]. The RCA network is trained on the rain map R obtained by the subtraction of the clean image B from the rainy image O, instead of training on the rainy image O directly. Since the proposed RCA network classifies the rain levels nearly perfectly, it could well capture the characteristics of the rain components.

The RCA network is built with five convolutional layers and two fully con-

nected layers. ReLU is used in each layer and the 7×7 kernel filter with stride of 3 is applied for the first three convolutional layers. The two subsequent convolutional layers are processed by 3×3 kernel filter with stride of 1. Each of the two fully connected layers consists of 500 neurons and the appended output layer has four nodes representing the four classes. The softmax with cross entropy loss is used for training. Compared with other networks, the RCA network is constructed with a simpler structure for two reasons. One is that a simpler network converges faster and occupies less computing resource. The other is that the rain components are relatively local and low-level features benefit less from the growth of the network depth.

RCA Loss Implementation

The RCA loss is computed by applying L2-norm distance on the feature maps while forwarding the ground truth $R = O - B$ and the predicted rain components $R^{'} = O - B^{'}$ through the well-trained RCA network ϕ, shown as

$$\mathcal{L}_{RCA} = \sum_{j} \left[\frac{1}{C_j \cdot H_j \cdot W_j} \left\| \phi_j(R^{'}) - \phi_j(R) \right\|_2 \right], \tag{5.4}$$

where $j \in \{0, 1, 2\}$, C_j, H_j, W_j represent the number of channels, height, width of the j_{th} feature map; $\phi_j(\cdot)$ indicates to extract the activations of the j_{th} layer feature map; $R^{'}$ is the predicted rain component maps. Since the rain feature is a low-level feature extracted mainly from relatively local areas, the $RCA\_residual$ loss is calculated from relatively low-level layers, the original rain maps ($j = 0$) and the first two convolutional layers $\{conv1, conv2\}$.

Comparison with Perceptual Loss

In recent GAN-based work, besides low-level pixel loss ($L1$ or $L2$ distance on pixels), the perceptual loss [82] is commonly applied for optimizing G along with the discriminative loss. The perceptual loss is able to capture semantic features by computing high-level representations extracted from a pre-trained network (Commonly VGG16 pre-trained on the ImageNet dataset). But for de-raining, the perceptual loss provides limited help, since the rainy image and the target clean image have similar image content. Very similar high-level features could

be generated by the perceptual loss network even if the rain components are not removed completely. This is due to the fact that the network lacks the ability of effectively identifying and representing the rain components.

5.3.5 Objective Functions

Similar to many synthesis tasks [78, 112, 263], the pixel loss is applied in the model to reduce the average pixel error between the outputs and the targets. In this work, the pixel loss is measured by the combination of L_1 and L_2 distances, shown as

$$\mathcal{L}_{pixel} = \frac{1}{C \cdot H \cdot W}(\left\|B^{'} - B\right\|_1 + \left\|B^{'} - B\right\|_2), \tag{5.5}$$

where $B^{'}$ and B are the outputs and the targets; C, H, W indicates the channel, height, width of the image.

Besides the pixel loss, SSIM loss is applied to improve the structural similarity of the outputs and the targets, which has been experimented in [170, 218]. SSIM loss is formulated as

$$\mathcal{L}_{SSIM} = 1 - SSIM(B^{'}, B), \tag{5.6}$$

where $SSIM(\cdot)$ means to calculate the SSIM score of two images. The loss from the discriminator to update the generator is shown as Eq. (3.7) Combining Eq. (5.5), (5.6) and (3.7), the loss function of the image content optimisation branch is defined as

$$\mathcal{L}_B = \mathcal{L}_{pixel} + \mathcal{L}_{SSIM} + \mathcal{L}_G. \tag{5.7}$$

Considering the two optimisation branches, the overall loss function to optimise the generator is written in Eq. (5.8), where λ_2 is the balancing weight.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_B + \mathcal{L}_R \\ &= \mathcal{L}_B + \lambda_2 \mathcal{L}_{RCA}. \end{aligned} \tag{5.8}$$

5.4 Experimental Results

5.4.1 Experiment Settings

Besides PSNR and SSIM, Universal image Quality Index (UQI) [223] and Visual Information Fidelity (VIF) [185] are used to evaluate the algorithms to compare with state-of-the-art models. In view of the difficulty in obtaining the ground truth of real-world rainy images, the de-raining results of real-world images are measured through visual inspection. Besides, the model performance is evaluated on different rain densities and an ablation study is conducted to evaluate the effectiveness of the capsule structure and the RCA loss. A comparison of the proposed RCA loss and the perceptual loss is set in the experiment. Furthermore, the methods are measured indirectly by checking the segmentation results on the de-raining images, following the concept that a good de-raining image should have the same segmentation result as the ground truth has. In addition, the performance of the proposed RCA loss and the perceptual loss is compared.

5.4.2 Datasets

Synthetic Dataset

The proposed method is compared with the state-of-the-art methods on four synthetic datasets with input image sizes ranging from 64×64 to 512×512. The training images are randomly cropped into fixed-size patches before being fed into the network.

64×64 synthesized: Two datasets are used to evaluate the models in the size of 64×64. The Rain800 [113] contains 800 randomly selected outdoor images along with their synthesized rainy ones. 700 pairs are used for training and 100 pairs for testing. The Rain100H [239] is a heavy-rain dataset with 1800 training pairs and 100 testing pairs. They are both challenging datasets, since the rain shapes are various and the density is large [239].

512×512 synthesized: The synthesized dataset [248] consists of $12,000$ training pairs with three rain density levels (light, medium and heavy) generated in different orientations and scales. The testing subsets consist of two parts: the Test1 contains 1,200 image pairs synthesized in the same way as the training set; while the Test2 contains $1,000$ image pairs from another synthetic dataset [45].

Table 5.1: Category of the real-world dataset.

Category	Street	Wild	People-close	Object-close	Total
Heavy	83	44	61	77	265
Medium	127	70	93	128	418
Light	36	18	41	50	145
Total	246	132	195	255	828

Table 5.2: Network settings for 64×64 and 512×512

Inputting size	64×64	512×512
conv/deconv layer number	4	7
conv/deconv kernel size	6	9
caps dim GP/GF/DP/DF	8/16/16/8	8/16/8/8
caps num GP/GF/DP/DF	32/8/16/8	16/8/16/10
kernel size GP/DP	4/7	4/9
batch size	60	4
λ_1/λ_2	0.1/10	0.1/10

Real-world Dataset

Real-word rainy images are prepared as well. 828 images are downloaded from Internet or captured in rainy days. The image contents include street views, wild views, people in close views and other objects in close views. The rain density is in category of {Heavy, Medium, Light}. The statistics of the dataset is summarised in Table 5.1. They are used to evaluate whether the proposed approach could well handle real-world rainy images with diverse rain marks in terms of densities and orientations. The proposed real-world dataset can be downloaded from https://yang-fei.github.io/capsule-deraining-RCA-cGAN/.

5.4.3 Implementation Details

The strides of all the convolutional/deconvolutional filters of both G and D are set to 2. The output images are sent to compute the RCA loss directly without rescaling. The iteration of dynamic routing between two capsule-based layers is set to 3 [179]. Each convolutional/deconvolutional layer is followed by a batch normalization layer and then activated by ReLU. Drop out is applied with a rate of 0.5 for the first three deconvolutional layers. The learning rate is

Table 5.3: Evaluation comparison on the 64×64 synthetic dataset

Dataset	Rain800 [113]				Rain100H [239]			
Models	PSNR	SSIM	UQI	VIF	PSNR	SSIM	UQI	VIF
ID-cGAN [249]	20.14	0.7886	0.8015	0.3957	15.54	0.6187	0.8021	0.2555
PAN [214]	21.13	0.7828	0.8512	0.2941	22.27	0.7544	0.8477	0.2655
DetailsNet[45]	21.16	0.7320	0.8312	0.3941	22.26	0.6928	0.8521	0.3211
DID-MDN[248]	22.11	0.8680	0.8694	0.4112	18.76	0.7723	0.8507	0.2602
RESCAN[113]	24.09	0.8410	0.9377	0.3911	26.45	0.8458	0.9110	0.3745
PReNet[170]	23.69	0.8812	0.9362	0.3925	23.49	0.8542	0.9023	0.2966
RCA-cGAN	**26.86**	**0.8965**	**0.9446**	**0.4295**	**26.78**	**0.8749**	**0.9431**	**0.4111**

Table 5.4: Evaluation comparison on the 512×512 synthetic dataset

Dataset	Test1[248]				Test2[45]			
Models	PSNR	SSIM	UQI	VIF	PSNR	SSIM	UQI	VIF
ID-cGAN [249]	25.86	0.8657	0.8817	0.4023	23.58	0.7997	0.8565	0.3241
PAN [214]	27.43	0.8637	0.9329	0.3591	24.88	0.8093	0.9080	0.3215
DetailsNet[45]	27.33	0.8978	0.9349	0.4137	25.63	0.8851	0.9176	0.3728
RESCAN[113]	28.32	0.8638	0.9278	0.4021	24.70	0.8126	0.9001	0.3703
DID-MDN[248]	27.95	0.9087	0.9299	0.3966	26.07	**0.9092**	0.9206	0.4159
PReNet[170]	30.31	0.9360	0.9427	0.4726	24.34	0.8617	0.8887	0.4009
RCA-cGAN	**32.03**	**0.9468**	**0.9484**	**0.4938**	**27.11**	0.8984	**0.9306**	**0.4293**

initialized with 0.0002 and decayed with a rate of 0.1 after every $60,000$ iterations. All the networks are trained using Adam optimizer in Tensorflow (python) on a NIVIDA GPU, TITAN X(Pascal). Table 5.2 presents more network settings, where GP/GF/DP/DF represent Generator PrimaryCaps, Generator FullConnectCaps, Discriminator PrimaryCaps and Discriminator FullConnectCaps layers respectively. The Generator DePrimaryCaps layer has the same setting as GP, which is not shown in this table.

5.4.4 Comparisons with the State-of-the-art

On Synthetic Images

Table 5.3 summarizes the de-raining results on the $\mathbf{64 \times 64}$ synthesized datasets. It can be seen that our proposed method noticeably outperforms all other methods on both datasets. The proposed method improves the PSNR and the SSIM scores significantly over RESCAN[113], DID-MDN[248] and PReNet[170]. Both datasets contain very heavy rain marks and the image patches in the size of $\mathbf{64 \times 64}$ cover quite limited image content. These cause much difficulty for the models, such as

DetailsNet[45] and DID-MDN[248], to recover the image content. Despite the challenges, the proposed method uses the RCA loss to encourage the model to capture the characteristics of heavy rain and utilizes the capsule units to grasp the information from small image patches, and hence achieve a significant better performance compared with others.

Tabel 5.4 summarizes the experimental results on the two datasets of larger size, Test1 [248] and Test2 [45]. It is obvious that the proposed RCA-cGAN achieves a superior performance compared with the state-of-the-art approaches. DID-MDN [248] has a better SSIM on Test2, due to the learning ability of content information from the multi-scale structure. Because of the proposed RCA loss, RCA-cGAN could better model the rain components. Comparing with PReNet [170], a significant improvement is achieved.

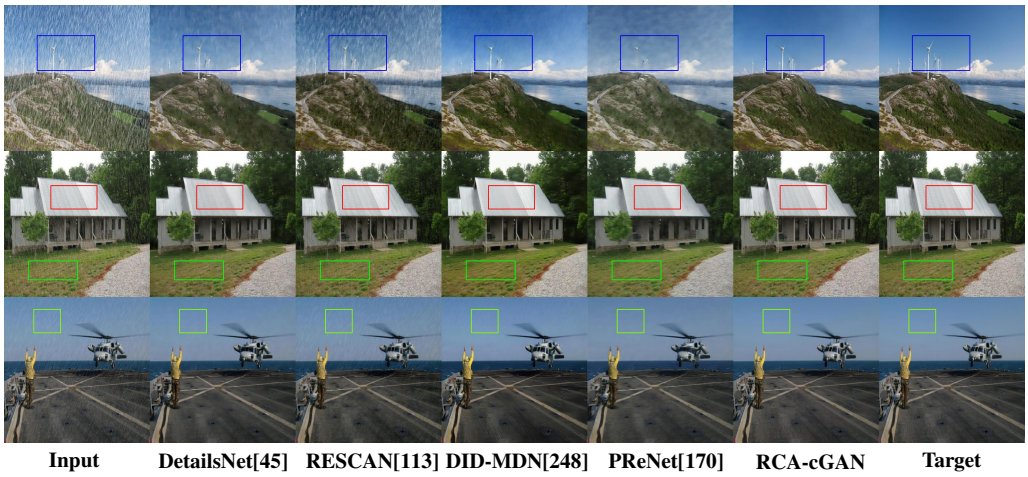


Figure 5.4: *De-raining results on Test1 of synthetic dataset. The last column is the ground truth (target).* **Best viewed on screen.** *The sky regions pointed by the rectangles shows that the proposed model produces the most smooth images. The roof and the grass from the second sample indicate that the proposed model preserves the most details.*

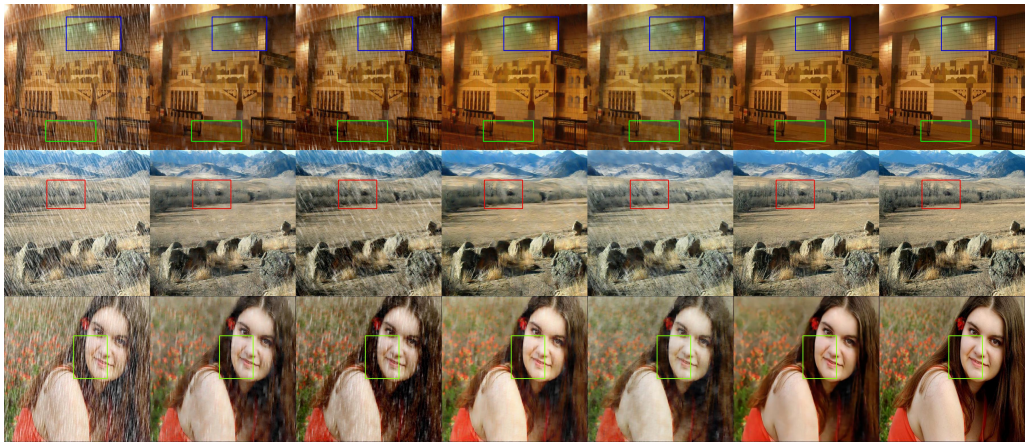


Figure 5.5: *De-raining results on Test2 of synthetic dataset. The last column is the ground truth (target). The image regions in the rectangles show that the proposed model produces the best images with precise details.* **Best viewed on screen.**

To visually inspect the proposed method, it is compared against the state-of-the-art methods using both the synthetic (Fig. 5.4 and 5.5) and the real-world (Fig. 5.6) images. As shown in Fig. 5.4 and 5.5, previous models either tend to under de-raining (see the sky from Fig. 5.4) or blurring image components with artifacts. The compared models, such as DID-MDN [248] and PReNet [170], introduce artifacts (the dark facts from the wall on Fig. 5.5), blurry image content (the roof of the house in Fig. 5.4, the yellow grass, the red flower and the mountains from Fig. 5.5) and other artifacts (see the girl face from Fig. 5.5). In comparison, the de-raining images from the proposed model preserve the original colour distribution with more content details.

On Real-world Images

Fig. 5.6 shows the de-raining results on real rainy images. In general, the proposed RCA-cGAN produces the best visual results. From the red umbrella and the blue clothes of the first image, the dark clothes of the second and the third images, the pale clothes of the sixth image, the proposed model removes the rain almost completely, while other models still contain some rain components. From the wall region of the second image, the proposed model not only well removes the rain but also recovers the details of image content successfully. The fourth and fifth images contain different kinds of rain components, which other models fail to remove but RCA-cGAN successfully eliminates.

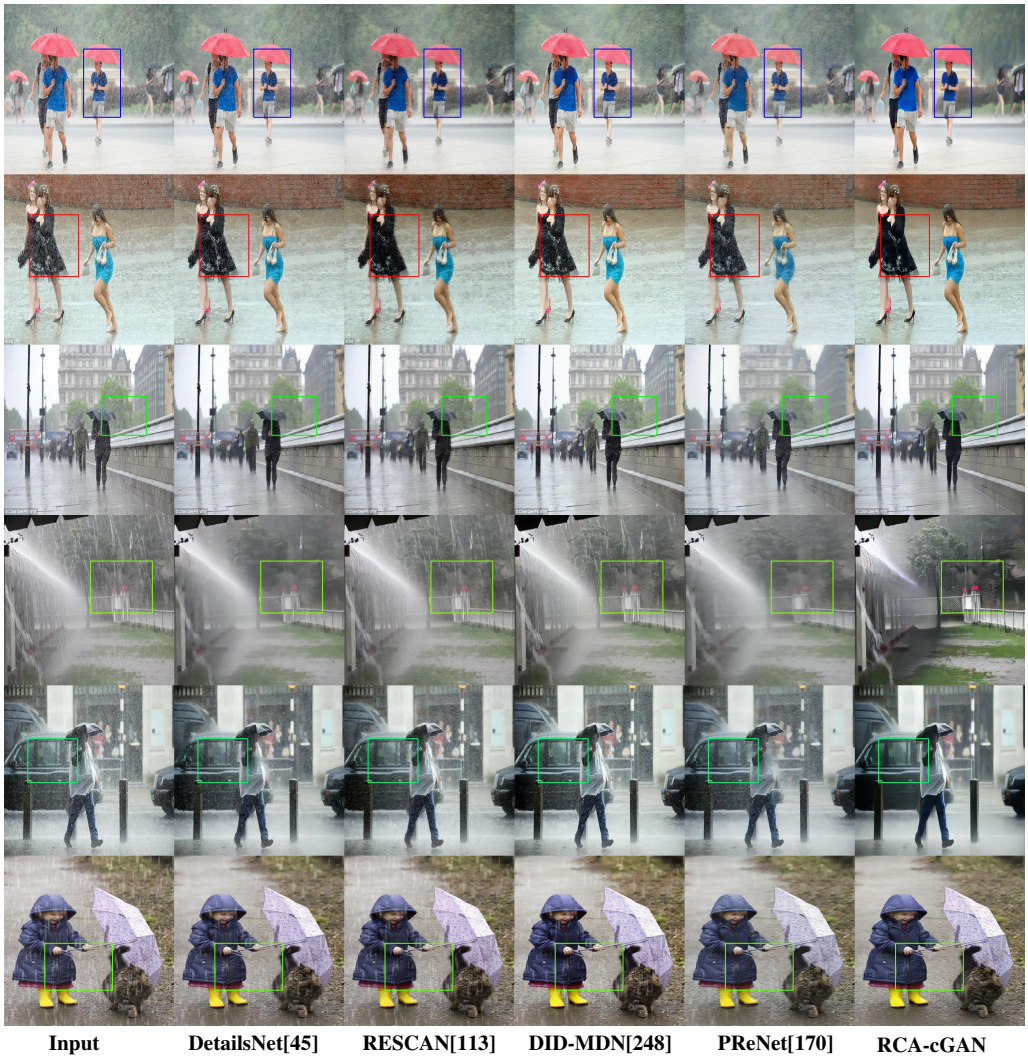


Figure 5.6: *De-raining results on real-world rainy images (no ground truth). By comparing the image regions in the rectangles, RCA-cGAN removes the rain completely and recovers the image details best.* **Best viewed on screen.**

5.4.5 Analysis of RCA-cGAN on Different Rain Densities

The density of the rain components affects the de-raining performance greatly. To verify this, a detailed analysis is conducted on handling rainy images with different densities as shown in Table 5.5. RCA-cGAN is compared with the state-of-the-art-models, PReNet.

Table 5.5: The comparisons of PReNet [170] and the proposed RCA-cGAN on three rain densities of Test1 [248].

models	Subset-Heavy (PSNR/SSIM)	Subset-Medium (PSNR/SSIM)	Subset-Light (PSNR/SSIM)	Overall (PSNR/SSIM)
PReNet [170]	27.49/0.9061	29.36/0.9337	34.11/0.9682	30.32/0.9360
RCA-cGAN	**29.60/0.9215**	**30.93/0.9456**	**35.57/0.9736**	**32.03/0.9468**

Figure 5.7: *Examples of rain component maps. The feature maps of rain component branch are presented.* **Best viewed on screen.** *The rain component maps are contrast enhanced for better visualization.*

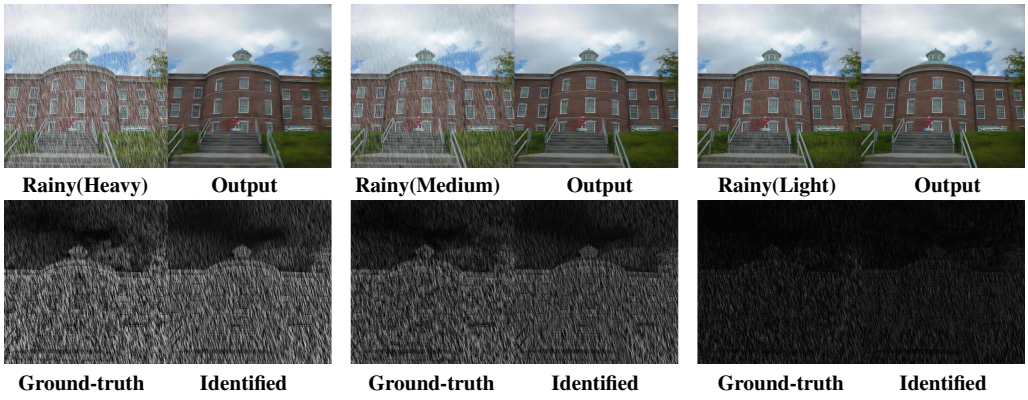


Figure 5.8: *Examples of rain component maps of de-raining results on different rain density images. Our model identifies rain well in various density.* **Best viewed on screen.** *The rain component maps are contrast enhanced for better visualization.*

The table shows that the proposed RCA-cGAN produces better PSNR and SSIM results for all three density levels. Especially on the Subset-Heavy, RCA-cGAN increases the PSNR from 27.49 to 29.60 and the SSIM from 0.9061 to 0.9215. With heavy rain, the image content is hard to recover when most parts of the image are covered by rain. RCA-cGAN could better remove the rain components using the RCA loss and capture the relationship between different image parts using the capsule units, and hence produce better results.

Table 5.6: Ablation study of RCA loss in ID-cGAN on Rain800 [113], Test1 [248], Test2 [45] and Test3 [249].

Dataset	Rain800 [113]		Test1 [248]		Test2 [45]		Test3 [249]	
Models	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ID-cGAN [249]	20.14	0.7886	25.86	0.8657	23.58	0.7997	22.73	0.8133
ID-cGAN[249]+RCA	21.51	0.8182	29.78	0.9235	26.07	0.8489	23.15	0.8371
Proposed RCA-cGAN	**26.86**	**0.8965**	**32.03**	**0.9468**	**27.11**	**0.8984**	**24.97**	**0.8831**

Table 5.7: Ablation study of rain component branch and capsule structure in cGAN on Rain800 [113], Test1 [248] and Test2[45].

Dataset	Rain800 [113]		Test1 [248]		Test2[45]	
Models	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
cGAN [78] (base network)	23.17	0.8408	25.86	0.8492	25.04	0.8294
cGAN [78] + caps	26.04	0.8875	29.21	0.9194	26.17	0.8634
cGAN [78] + caps+RCA	**26.86**	**0.8965**	**32.03**	**0.9468**	**27.11**	**0.8984**

An example of rain component estimated by the proposed RCA-cGAN is shown in Fig. 5.7. The proposed RCA-cGAN removes most of the rain components, with only a few marks left on the contours of the image. Fig. 5.8 shows the de-raining results of our model on different rain densities of images. By comparing the rain component maps of inputs and those of de-raining results, it can be seen that the proposed model removes rain well in various densities.

5.4.6 Ablation Study

Ablation of RCA Loss

To validate the effectiveness of the RCA loss, an experiment of applying the RCA loss is conducted on a GAN-based de-raining framework, ID-cGAN [249], which utilizes the fundamental structure of the image-cGAN. To my knowledge, ID-cGAN was the first to adopt cGAN framework for de-raining. The experiment is conducted on the dataset of ID-cGAN, Test(ID-cGAN) in Table 5.6. For comparison, in this ablation study the results of a cGAN [78] is listed, which is often discussed as a basic image-generation framework. The experiments are conducted using images in Test1 and Test2.

The evaluation results are shown in Table 5.6 and 5.7. The "caps" means the

capsule structure is used. In Table 5.6, the RCA loss improves the performance on both datasets. The PSNR result increases from 25.86 to 29.78 for Test1 and from 23.58 to 26.07 for Test2, respectively. The SSIM also increases greatly. The proposed model shows much better performance on Test(ID-cGAN). Regarding the results of using cGAN as the backbone method in Table 5.7, both the capsule structure and the RCA loss enhance the model performance significantly.

Ablation of Rain Component Branch

A ablation study is set by removing the rain component branch from RCA-cGAN. The experimental results are shown in Table 5.7. RCA-cGAN is built with "cGAN", "caps" and "RCA", which represent the base framework, capsule structure and rain component branch, respectively. By comparing the results of {cGAN + caps} and {cGAN + caps + RCA}, we can observe the great performance increase from rain component branch.

Ablation of Capsule

Using cGAN [78] as a base network, the capsule structure is implemented to perform experiments on Test1 and Test2. The results are listed in Table 5.7. By comparing the results of cGAN and {cGAN + caps}, we can see that the capsule structure improves the performance significantly.

Comparison of RCA Loss and Perceptual Loss

The RCA loss and the perceptual loss [82] follow the same principle that the loss is calculated by passing images through a pre-trained network, whereas the RCA network has a different structure from the perceptual loss network, and these two are pre-trained on different images with opposite objectives. The RCA loss is sensitive to the rain feature while the perceptual network focuses on the image content. The rain components can be seen as one kind of noise to the image content and be ignored by the perceptual loss network indeed.

An experiment is conducted to compare the contributions of the two kinds of losses by applying each of them on the base framework. The results are shown in Table 5.8, where we can see that the effect of the perceptual loss is very limited. The PSNRs for both Rain800 and Test1 decrease and the SSIM on Rain800

Table 5.8: Comparison of RCA and perceptual loss on Rain800 [113], Test1 [248] and Test2 [45] datasets.

Dataset	Rain800[113]		Test1 [248]		Test2 [45]	
Models	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
cGAN [78] (base network)	23.17	0.8408	25.86	0.8492	25.04	0.8294
cGAN [78] + Percep [82]	22.33	0.7192	25.30	0.8634	25.67	0.8299
cGAN [78] + RCA	**24.41**	**0.8559**	**29.21**	**0.9094**	**26.07**	**0.8499**

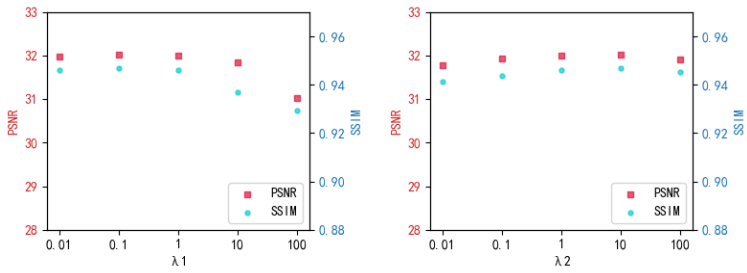


Figure 5.9: *Results of RCA-cGAN with different λ_1 (left) and λ_2 (right) on Test1 [248]*

.

becomes much worse. The results of {cGAN+RCA} show that the RCA loss provides much help to produce better results.

5.4.7 Hyper-parameters λ_1 and λ_2

The proposed RCA-cGAN has two hyper-parameters λ_1 and λ_2 balancing the patchGAN loss in D (Eq. (3.4)) and the RCA loss in the overall optimisation (Eq. (5.8)). In the experiments, the parameters are set as $\lambda_1 = 0.1, \lambda_2 = 10$, as listed in Table 5.2. Experiments are done on Test1 [248] by changing the parameter from 0.01 to 100. λ_1 is set as 0.1 when changing λ_2 and λ_2 is set as 0.1 when changing λ_1. The results are shown in Fig. 5.9. The performance keeps stable when λ_1 is small and starts to decrease when λ_1 changes from 1 to larger. The results become better when λ_2 changes from 0.01 to 10, but worse when changes from 10 to 100. The best settings of $\lambda_1 = 0.1, \lambda_2 = 10$ are set for the proposed RCA-cGAN.

Table 5.9: Segmentation results of different model outputs on Test1 [248] datasets.

Models	Per pixel accu.	Per class accu.	Class IOU
Rainy images	0.9074	0.5446	0.4888
DetailsNet [45]	0.9150	0.6265	0.5074
PAN [214]	0.9073	0.5673	0.4990
ID-CGAN [249]	0.9070	0.5915	0.5144
RESCAN [113]	0.9024	0.5269	0.4592
DID-MDN [248]	0.9378	0.6370	0.5801
PReNet [170]	0.9377	0.6655	0.5693
RCA-cGAN(ours)	**0.9531**	**0.7398**	**0.6387**

5.4.8 Experiments of De-raining for Segmentation

De-raining aims to remove the rain and recover the image content, benefit other visual recognition or image analysis systems. It is reasonable to use an indirect way to evaluate the de-raining models by examining the results of a noise sensitive task on the de-raining images. Similar as in [78], the segmentation work of FCN-8s [30] is applied. The segmentation results of clean images are considered as the ground truth. The segmentation results of the de-raining images from different models are compared with the ground truth to calculate the per pixel accuracy, per class accuracy and the class IOU [30], shown in Table 5.9.

The proposed model gives the best results and improves the performance greatly, especially the per class accuracy and the class IOU. It considers both the pixel-level and the object-level synthesis and tries to generate more meaningful image content, which produces better results in terms of per-class accuracy and the class IOU. The results show that the images generated by the proposed RCA-cGAN are much similar to the clean images and the de-raining process benefits the segmentation task.

5.5 Summary

A novel rain-component-aware image-conditioned generative adversarial network is proposed for rain removal. By incorporating both the RCA optimisation branch and the modified capsule structure, the proposed RCA-cGAN outperforms the state-of-the-art methods over four de-raining datasets in both quanti-

tative evaluation and visual inspection. Through the detailed experimental evaluation and analysis, it can be concluded that the proposed RCA-cGAN can effectively remove the rain components while preserving more image details with visually appealing image appearance, which can benefit image visual enhancement and image analysis systems applied thereafter. The rain patterns could be well captured by the proposed RCA network. The part-to-whole information is modelled by capsule units well. Though the proposed RCA-cGAN achieves good performance, the construction of capsule units and CNN needs to be carefully designed and examined. The experimental results show that the proposed framework outperforms the state-of-the-art models and the proposed RCA loss can be easily applied in other image de-raining framework.

I have one publication about this work, "Rain-component-aware capsule-GAN for single image de-raining" in Pattern Recognition [J], which is under review after the major revision.

Chapter 6

De-hazing: Image-depth-aware Haze Removal

6.1 Introduction

Haze is a common atmospheric phenomenon, caused by the light scattering of the fog, smoke, dust and other floating particles. It lowers the image quality and leads to the performance degradation of vision recognition systems. The research of haze removal plays an important part in visual signal processing and has attracted much attention for a long time.

Haze removal is to remove the haze noise and restore the clear image content from a single hazy image. Conventional de-hazing models formulate the haze image as an atmospheric scattering model described by [136, 143], formed as Eq. (6.1).

$$I(z) = J(z)t(z) + A(z)(1 - t(z)) \tag{6.1}$$

where I is the observed image with haze, J is the clear image content, A is the global light intensity, t is the transmission map and z represents the location of each pixel. The key process of recovering J from I is the accurate estimation of t and A. Comparing with A, t is the primary factor that causes the hazy noise. Thus, except several works that discuss the prediction of atmospheric light A, like [11, 198], most of the de-hazing works pay much attention to the estimation of the transmission map t. Before the great development of CNN, the de-hazing methods were mostly the prior-based approaches, such as [10, 42,

Figure 6.1: *The model generates the de-hazing image and estimates the depth map at the same time*

59, 250]. Among these, dark channel prior [59] was the commonly used method and achieved good results on certain extent of haze. These prior-based methods are not stable in real scenes because of their sensitiveness to the distribution of the haze noise [17]. With the evolution of deep learning techniques [187, 188], the learning-based models [189, 208] have been much applied for the de-hazing problem in various approaches[31], such as predicting the transmission map t [17, 107, 163], estimating the atmospheric light A [240, 247, 273], or synthesizing the clear image content J directly [23, 160, 247].

Although some good results have been achieved, the de-hazing problem is still unsolved with unexpected noise-like artifacts [160, 247]. The essential factor to prevent the model from obtaining the optimal result is the inaccurate prediction of the transmission map t. The learning of t is difficult, since the image background of different images varies much and the distribution of the haze noise is greatly unbalanced. Indeed, t can be expressed as $t(z) = e^{-\beta d(z)}$, where β is the coefficient of the atmosphere and d is the scene depth [108, 247]. The formula indicates the close relationship between the haze extent and the image depth. However, existing de-hazing models [23, 111, 160, 173, 247] discussed little about the depth information specially. These works focused on designing better network structures to learn the mapping from the hazy domain to the clear domain. Considering the relationship of haze and depth, an idea comes out that the depth information may also be estimated from the hazy image directly and provides help in identifying the extent of the haze.

Based on the discussion above, a depth aware framework is proposed to restore the clear image content from hazy images, named Depth De-hazing Network (DDN). By implementing two pairs of generators and discriminators, the

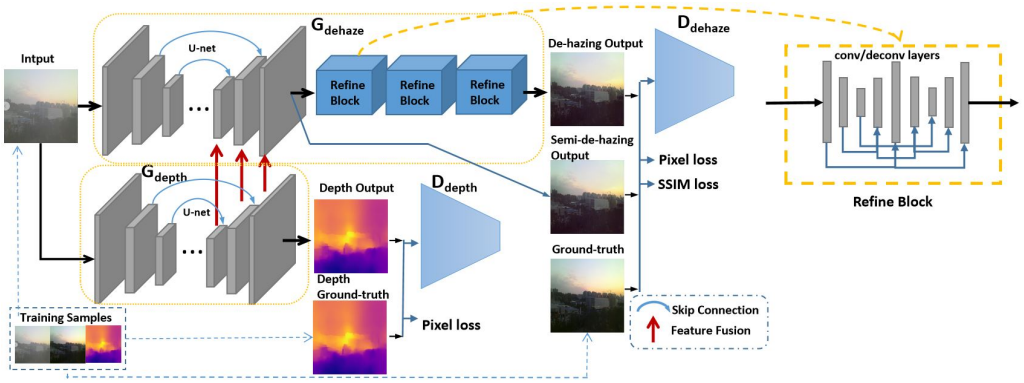


Figure 6.2: *The overall framework of the proposed DepathDehazeNet. The "dehazed Output" of the G_{dehaze}, as the synthesized image expected, is discriminated by D_{dehaze} and calculated for the Pixel loss and SSIM loss. The depth prediction network consists of the G_{depth} and the D_{depth}, which aim to learn the depth feature and distinguish the depth maps, respectively. Specially, the "semi-dehazed output", the output of U-net within G_{dehaze} (before feeding into the refine blocks), contributes to the pixel loss and the SSIM loss partially.*

model is able to recover the image content and predict the depth at the mean time. The depth feature learned by the depth generator is fused into the de-hazing network for better haze removal. The overall framework is trained jointly in an iterative procedure with multiple losses.

Therefore, the contributions are summarised as follow. 1) An end-to-end training framework to estimate depth and haze is proposed and it achieves the state-of-the-art performance for de-hazing. 2) The proposed depth estimation module is proved to be effective for haze removal. 3) A feature fusion method is introduced to embed the depth information into the haze removal network as 1×1 convolution filtering, which is straightforward and effective. I have a publication in *The Visual Computer*, "Depth aware generative adversarial network".

6.2 Pre-knowledge

6.2.1 Prior-based Haze Removal

Most of dehazing models tackle the haze removal problem with the physical scattering model [136, 143], shown in Eq. (6.1). Early models are mostly prior-based methods. Tan et al. [207] focused on the contrast with Markov random field with assuming that the image content has different effects on the contrast

with the haze. Tarel et al. applied the white balanced technique to identify the haze [209]. He et al. [59] proposed a dark channel prior (DCP) method the separate the haze mask.

Later, a method with the boundary constraint was implemented by Meng et al. [137]. A colour attenuation prior method was proposed by Zhu et al. [278]. More recently, Berman et al. [10] proposed a non-local method and Chen et al. [19] used gradient residual minimization to remove the haze noise and suppress the visual artifacts. Kahma et al. [89] apply second-generation wavelets to accelerate the haze estimation. These prior-based methods relied much on the assumption of haze distribution, resulting in the low robustness of the model performance. The drawback is obvious that the prior-based approaches often fail to process images with unbalanced noise distribution. The robustness to tackle complex image contents is also quite low.

6.2.2 Learning-based Haze Removal

With the improvement of deep learning, the learning-based methods with CNN showed much superiority than prior-based methods in terms of the model robustness. Different network structures have been discussed in recent works. Cai et al. [17] proposed the DehazeNet, which firstly introduced a fundamental CNN structure for de-hazing. Later, researchers found that the learning of the image feature benefits from the multi-scale structure, Ren et al. [172] built an multi-scale network (MSCNN) to learn the estimation of the transmission map t. Li et al. [107] improved the de-hazing performance by building a multi-output net with predicting the transmission map t and the atmospheric light A at the mean time. Zhang et al. [247] proposed a densely connected pyramid de-hazing network (DCPDN) to predict t, A and J jointly. Chen et al. [23] proposed a patch map selection method to estimate t and J more accurately. These works showed that to estimate parts of the haze scattering model is effective to estimate haze-free images.

More advanced deep learning techniques motivate researchers to build unified networks to predict de-hazing images. Qu et al. [160] applied the multi-resolution nets and multi-discriminators to synthesize the haze-free images. Fan et al. [39] applied Gaussian process regression into the haze learning. In these works, more advanced network structures were proposed, which also indicated

that the advanced networks makes the training more easily and directly, without considering too much on the scattering model parameters. Based on these techniques, Guo et al. [53] made use of the depth information for de-hazing with the fact that a distant scene contains much haze. Zhang et al. [258] used advanced networks to estimate the transmission map and recover the image content. They then proposed a deep residual network to remove haze [257]. Chen et al. [26] proposed an novel non-local network to improve the robustness of de-hazing.

These de-hazing frameworks learn the haze-free images by the end-to-end training without the consideration of the image content. While, the image content has a close relationship with the haze distribution. In this work, instead of exploring new network structures for the de-hazing, an idea is introduced that the depth information can be estimated from the hazy images directly and is able to enhance the de-hazing performance significantly. The experimental results demonstrate that the proposed model outperforms existing de-hazing methods and the depth aware module can be easily implemented by a feature fusion method.

6.3 Proposed Method

A generative adversarial network-based model is built for haze removal, which involves a depth aware module to learn the depth maps along with the de-hazing process. The depth aware module aims to provide the depth information to the de-hazing module through learning the depth maps from the hazy images directly. The de-hazing performance is much improve with the awareness of depth. The overall framework is trained on well-prepared depth maps and clear images. Since a common sense is presented that a hazy image and its corresponding clear image should have the same depth information, the ground-truth of the depth maps are obtained by applying a depth estimation tool, DenseDepth [3], on clear images during the datasets preparation. To involve the depth feature into the generator of de-hazing model, the generator structure is constructed with convolutional layers without capsules.

Table 6.1: The Training Algorithm.

Input:
Hyper-parameter setting,
Training samples X_{hazy}, X_{clear}, X_{depth},
for $i = 1; i <= training\_iterations$ **do**
Forward $G_{depth}(X_{hazy})$ and $D_{depth}(X_{hazy}, X_{depth})$,
Update D_{depth} by applying the gradient of Eq. (6.2),
Update G_{depth} by applying the gradient of Eq. (6.3),
Forward $G_{dehaze}(X_{hazy})$ and $D_{dehaze}(X_{hazy}, X_{clear})$,
Update D_{dehaze} by applying the gradient of Eq. (6.4),
Update G_{dehaze} by applying the gradient of Eq. (6.5).

6.3.1 The Overall Structure of the Proposed Method

The overall framework, DDN, consists of four parts, the generator (G_{dehaze}) and the discriminator (D_{dehaze}) for de-hazing, the generator (G_{depth}) and the discriminator (D_{depth}) for depth estimation, shown in Fig. 6.2. G_{depth} aims to estimate the depth maps and G_{dehaze} is responsible for generating the expected haze-free images. The depth feature of G_{depth} is fused into G_{dehaze} to provide the depth information for haze removal, shown as the red arrows in Fig 6.2. G_{depth} is in a structure of U-net [175] while G_{dehaze} is with the same U-net structure and three additional refine blocks. D_{depth} and D_{dehaze} are discriminators introduced by [235] to distinguish the real or fake samples. The detailed structure of D_{depth} and D_{dehaze} is not described here, since it is not my main contribution. The U-net is constructed by nine 2-stride convolutional layers for the encoder and nine corresponding 2-stride deconvolutional layers for the decoder. LeakyReLU is applied as the activation of each layer. Each layer is followed by a batch normalization layer. The dropout algorithm with a rate of 0.5 is done for the first four layers of the decoder.

6.3.2 Depth Feature Fusion

The proposed framework consists of two generating parts, the depth aware part and the de-hazing part. The purpose is to fuse the depth feature into the de-hazing process so that the haze and the image content can be separated more effectively.

By the training of the depth module, the decoder layers of the U-net in

G_{depth} should contain the information to predict the depth maps, whose activations are regarded as the depth features to be provided to the de-hazing part. The depth feature fusion can be achieved by concatenating the layer maps of G_{depth} to the corresponding layers of G_{dehaze}, shown in Fig. 6.2. During the training process, G_{depth} and G_{dehaze} are trained separately in an iterative way.

6.3.3 The Refine Block

Commonly, the output of the U-net within G_{dehaze} is not perfect with some unexpected artifacts, which are considered as a failure of the de-hazing optimization, which can be named as semi-de-hazing output. Inspired by DID-MDN ([248]), to further improve the de-hazing performance, three refine blocks are stacked to enhance the image quality. Each refine block is composed of nine convolutional/deconvolutional layers with the filter kernel size of 3×3, shown in Fig. 6.2. The layer maps remain the same size with the input. The kernel stride is set 2 for all layers to achieve the down-sampling and up-sampling processes. Skip connections are set for all the layers to concatenate the layers in the same sizes. Each layer is activated by LeakyReLU and followed with a batch normalization layer. Compared with the refine layers used by DID-MDN, I set more refine blocks and more layers within each to accomplish the refining process; skip connection is used to connect the same-size layers to transmit the features more directly.

6.3.4 Optimization Functions

SSIM is a common evaluation metric to compare the similarity of two images, which is closer to human visibility than other evaluation metrics. Therefore, intuitively, to formulate SSIM as part of the training loss would increase the performance on SSIM scores while enhancing the visual quality of synthesized images. Such an idea was also discussed in recent works [170, 218].

The proposed DDN consists of multiple generative adversarial networks, optimized by a weighted combination of multiple loss functions. D_{depth}, G_{depth}, D_{dehaze} and G_{dehaze} are trained with Eq. (6.2), (6.3), (6.4) and (6.5).

$$\begin{aligned}\mathcal{L}_{D_{depth}} = E_{x,d}[log(1 - D_{depth}(x, d)) \\ + log(D_{depth}(x, G_{depth}(x)))],\end{aligned} \tag{6.2}$$

$$\begin{aligned}\mathcal{L}_{G_{depth}} = E_{x,d}[log(1 - D_{depth}(x, G_{depth}(x)))] \\ + \lambda_1 \mathcal{L}_{pixel},\end{aligned} \tag{6.3}$$

$$\begin{aligned}\mathcal{L}_{D_{dehaze}} = E_{x,y}[log(1 - D_{dehaze}(x, y)) \\ + log(D_{dehaze}(x, G_{dehaze}(x)))],\end{aligned} \tag{6.4}$$

$$\begin{aligned}\mathcal{L}_{G_{dehaze}} = E_{x,y}[log(1 - D_{dehaze}(x, G_{dehaze}(x)))] \\ + \lambda_1 \mathcal{L}_{pixel} + \lambda_2 \mathcal{L}_{SSIM},\end{aligned} \tag{6.5}$$

where x, y, d represent the haze, clear, depth images; $G_*(\cdot)$, $D_*(\cdot)$ indicate the forwarding calculation of the corresponding networks; $\mathcal{L}_{pixel}$, $\mathcal{L}_{SSIM}$ are the pixel loss calculated as the L_1 distances (Eq. (6.6)) and the SSIM loss [270] (Eq. (6.7)). λ_1, λ_2 are the balancing weights for the losses. The overall training algorithm is shown in Table. 6.1

$$\begin{aligned}\mathcal{L}_{pixel} = \lambda_3 \left\| G_{semi\_dehaze}(x) - y \right\|_1 \\ + \lambda_4 \left\| G_{dehaze}(x) - y \right\|_1,\end{aligned} \tag{6.6}$$

$$\begin{aligned}\mathcal{L}_{SSIM} = 1 - \lambda_3 SSIM(G_{semi\_dehaze}(x), y) \\ - \lambda_4 SSIM(G_{dehaze}(x), y),\end{aligned} \tag{6.7}$$

where $SSIM(\cdot)$ is the SSIM score of two images; λ_3 and λ_4 are balancing weights.

6.4 Experiments

6.4.1 Experiments Setting

Experiments are conducted on both the synthetic and the real-world datasets for the comparison with the state-of-the-art de-hazing models. The ablation study is deployed to check the effectiveness of the proposed depth aware module. MSE, L1, PSNR and SSIM [270] are used to evaluate the model performance.

6.4.2 Dataset

The model performance is examined on the dataset of RESIDE proposed by [108], a large-scale hazy image dataset proposed in recent years. It consists of five subsets: ITS (Indoor Training Set), OTS (Outdoor Training Set), SOTS (Synthetic Objective Testing Set), RTTS (Real World task-driven Testing set) and HSTS (Hybrid Subjective Testing Set). ITS, OTS, SOTS are synthetic and can be used for the training and testing as the quantitative evaluation. ITS contains 10,000 clear images with each one generating 10 hazy images, resulting 100,000 hazy images. OTS generates 313,950 hazy images in various haze extent from 8,970 images. SOTS contains two testing sets for indoor and outdoor images, SOTS-indoor and SOTS-outdoor, respectively. SOTS-indoor has 500 hazy images generated from 50 clear images, while SOTS-outdoor contains 500 hazy ones synthesized from 492 clear ones. Images of RTTS and some others downloaded from Internet are used for the visual evaluation in the real-world de-hazing experiment. The clear images of ITS and OTS are processed by DenseDepth [3] to generate the corresponding depth maps.

6.4.3 Implementation Details

To have a fair comparison with other models, the training images (ITS and OTS) are resized into 640×640 and randomly cropped into 512×512 patches for data augmentation. During the testing, the images are resized to 512×512 and fed into the de-hazing model. The models are trained in the batch size of 5 using Adam optimizer with a initializing learning rate of 0.0002 for the generators, 0.0005 for the discriminators. The decay rate of the learning rate is set as 0.1 after every 200,000 iterations. The loss balancing weights λ_1,λ_2, λ_3, λ_4 are set as 10, 1, 0.2, 0.8. The implementation is done in Tensorflow with python.

6.4.4 Quantitative Comparison Results

The comparison results are shown in Table 6.2 and 6.3 for datasets SOTS-indoor and SOTS-outdoor, respectively. On both datasets, the proposed DDN achieves the best results. Comparing with the second best, DA_dehazing, DDN improves the performance slightly on SOTS-indoor, shown in the last two lines of Table 6.2. On SOTS-outdoor in Table 6.3, DDN improves the performance greatly

Table 6.2: Comparison results on SOTS-indoor

Model	PSNR	SSIM	MSE	L1	infer_time (s)
DCP [59]	16.72	0.8232	0.2897	0.2101	0.821
DehazeNet [17]	21.15	0.8459	0.1489	0.3352	0.654
AOD-NET [107]	19.14	0.8522	0.2173	0.1462	0.741
GFN [173]	22.28	0.8812	0.1787	0.1011	**0.547**
DCPDN [247]	15.82	0.8153	0.2811	0.1222	0.651
EPDN [160]	25.17	0.9227	0.1152	0.0553	0.947
DDRL [53]	24.35	0.9017	0.1501	0.0854	0.998
DA_dehazing [184]	25.78	0.9541	0.1198	0.0383	0.857
DDN	**28.43**	**0.9773**	**0.0432**	**0.0210**	0.995

Table 6.3: Comparison results on SOTS-outdoor

Model	PSNR	SSIM	MSE	L1
DCP [59]	19.14	0.8153	0.3141	0.1941
DehazeNet [17]	22.49	0.8517	0.1422	0.3465
AOD-NET [107]	20.33	0.8767	0.2949	0.1500
GFN [173]	21.59	0.8448	0.2737	0.1036
DCPDN [247]	20.14	0.8454	0.2429	0.1436
EPDN [160]	22.59	0.8632	0.1981	0.0782
DDRL [53]	22.21	0.8589	0.2101	0.1192
DA_dehazing [184]	27.23	0.9392	0.0750	0.0461
DDN	**31.29**	**0.9809**	**0.0531**	**0.0114**

from 27.21 to 31.25 for PSNR, and from 0.9389 to 0.9803 for SSIM. The reason why DDN shows better performance on SOTS-outdoor is that the haze distribution has more meaningful relationship with the depth information on outdoor images than indoor ones. The depth feature extracted from outdoor images provides much help for haze removal. The inference time of each model is also listed. They are tested on the same hardware platform with an I7 4-core CPU and GTX1080TI GPU. Though DDN does not cost the shortest time, it needs reasonable inference time to achieve a good performance.

6.4.5 Visual Comparison Results

The visual results are shown in Fig. 6.3, 6.4 and 6.5. The comparable results are presented in Table 6.2 and 6.3. Fig. 6.3 shows that the proposed model removes the haze without adding any other artifacts. "DCPDN" changes the

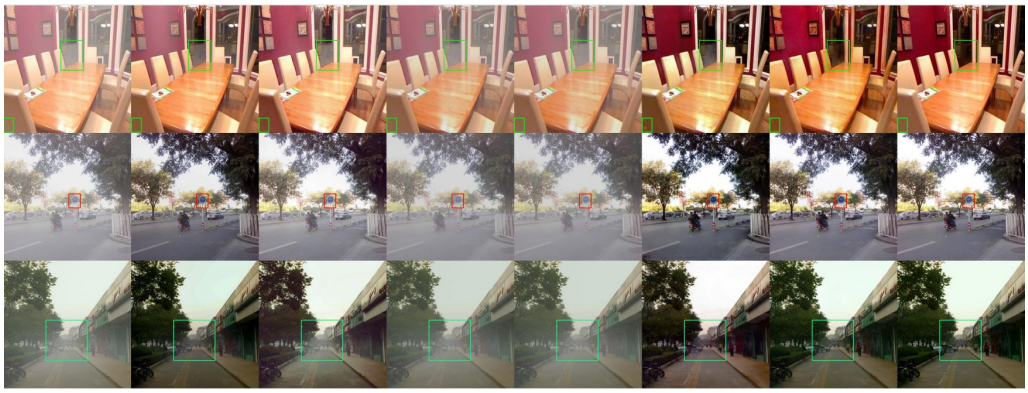


Figure 6.3: *Comparisons on SOTS-door and SOTS-outdoor images. The proposed model generates clear de-hazing images that are much close to ground-truth.*

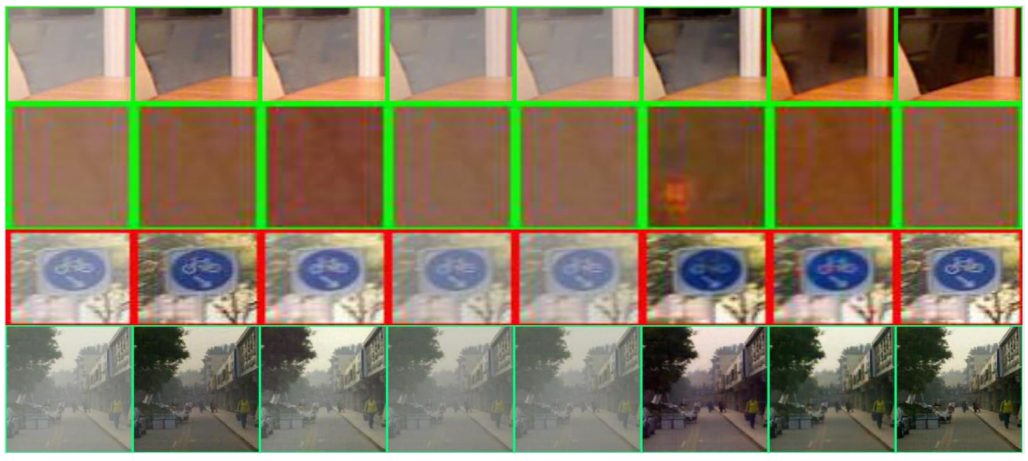


Figure 6.4: *The enlarged details from Fig. 6.3.*

Figure 6.5: *The de-hazing results of real hazy images. The proposed model synthesizes the images with more details. Better to view it on screen.*

Table 6.4: SSIM on different haze densities of SOTS-outdoor.

Model	Heavy	Slight_heavy	Medium	Slight_small	small
DDRL [53]	0.8014	0.8212	0.8761	0.8941	0.9014
DA_dehazing [184]	0.7981	0.9001	0.9562	0.9714	0.9802
DDN	**0.9541**	**0.9795**	**0.9910**	**0.9912**	**0.9914**

overall colour distribution of the image. "DehazeNet" produces too dark images. "EPDN" fails to remove the haze of distant scenes. The details within the rectangles are enlarged, shown in Fig. 6.4. Fig. 6.5 shows that the results of DDN keep more details of the image content, such as the texture of the "tree" on the "road side" and the outlines of the "building". The third testing sample of Fig. 6.3 shows the superiority of DDN greatly that removes the haze from near to far thoroughly. DA_dahzing also gives quite good results, it slightly changes the overall colour distribution. DehazeNet produces sharp images in quite good visual quality, which is adaptive its comparable SSIM scores from Table 6.3. But the second and the third samples show that DehazeNet over-dehazes the image content with producing more dark images and eliminating some local details.

6.4.6 Analysis on Different Haze Densities

The density of haze affects the de-hazing performance greatly. An experiment is conducted to analyse the effect of haze density on de-hazing performance. The testing images of SOTS-outdoor are separated into five haze densities and SSIM is calculated for each one. The proposed model is compared with DA_hazing and the depth-based DDRL. The results are shown in Table 6.4. The proposed DDN outperforms the other two models with a large margin. On heavy haze dataset, the performance of DDRL and DA_dehazing decreases greatly with SSIM about 0.8. Though DDRL contains depth part, the implementation is relatively simple with binary mask for depth regions. The proposed DDN still achieves a quite high score with SSIM of 0.9541 on Heavy set. The proposed DDN identifies haze well on heavy hazy images, benefiting from the effective feature from depth estimation module.

Table 6.5: Ablation Study on SOTS-outdoor.

Model	PSNR	SSIM	MSE	L1
without depth	25.61	0.8747	0.1185	0.0942
with depth	**31.25**	**0.9803**	**0.0543**	**0.0121**

6.4.7 Ablation Study

To examine the effectiveness of the depth awareness, an ablation study is conducted by removing the depth module (both the generator and the discriminator) and the depth feature fusion from the overall framework without modifying any other part. Since the depth information is much related to the haze on outdoor images, the ablated framework is tested on SOTS-outdoor, shown in Table 6.5. It is obvious that the performance decreases greatly after removing the depth module from the overall framework, which demonstrates the importance of the depth module in my framework. But the ablated framework still achieves slightly better results than EPDN, comparing with the last second line of Table 6.3. Besides the depth feature fusion, the reasonable performance improvement of my framework comes from the good design of the overall framework and the balancing combination of the multiple losses.

6.5 Summary

A depth aware de-hazing network is proposed and it is demonstrated that the depth information is essential to the identification of haze noise, especially for outdoor images. The depth information is able to be extracted from the hazy image and provides effect help in haze removal.

The experimental results show that the proposed method outperforms the state-of-the-art models. I have one publication about this work, “Depth aware generative adversarial network” in The Visual Computer [J] in 2021.

Chapter 7

De-snowing: Multi-scale Snow Removal

7.1 Introduction

Similar to rain and haze, snow is another kind of image noise that exists in real-world image commonly. While, different from the rain and the haze, the snowflakes vary much in the shapes, the sizes and the transparency. Simply to apply the de-raining or de-hazing model for the snow removal problem is hard to identify the snowflakes of different sizes at the same time. Thus, in this chapter, I discuss my work on de-snowing separately from the de-raining and the de-hazing.

Vision tasks in surveillance systems are vulnerable to weather conditions, such as rain, hail, and snow, whose atmospheric particles may impede the normal interpretation, resulting in a disaster possibly. An experiment supported by Cascade-DilatedNet [268] illustrates this in Fig. 7.1. Due to the snowstorm, the Cascade-DilatedNet fails to label the contents correctly or segment the objects clearly.

To counter this effect, various rain and snow removal techniques have been proposed to obtain clear images [33, 45, 73, 115, 132, 249, 267]. In the early stages of this research, prior-based approaches dominated. The atmospheric particles are detected and removed by well-designed handcrafted features, such as edge orientations [20, 86, 132], shapes [194] and streak patterns [115]. However, these methods not only heavily depend on the researcher's experience, but

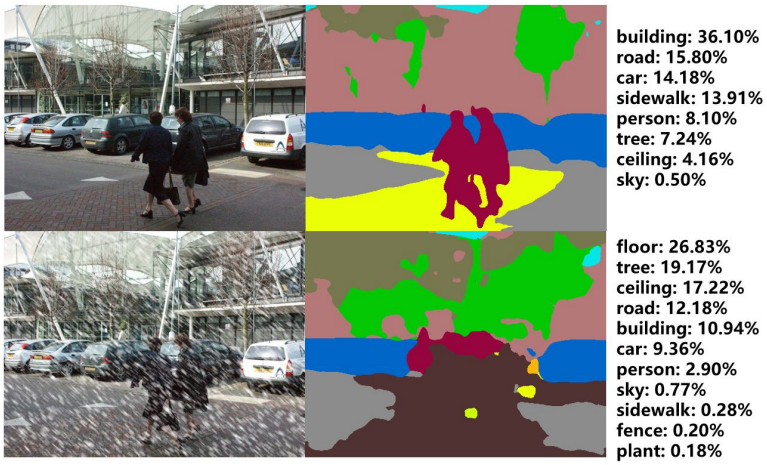


Figure 7.1: *Comparison of segmentation results for clean and snowy images. The segmentation results and the corresponding labels are predicted by [268].*

also greatly limit the generality capabilities of the models. With the development of deep learning, learning-based techniques [44, 45, 170, 239] have become the focus of research due to their greater efficiency and better generalization capabilities.

Some researchers regard snowflakes as a special kind of noise and apply common de-noising models for de-snowing. However, snowflakes have more complex characteristics, such as various morphological structures, irregular trajectories, diversified distribution and non-uniform transparencies, which make de-snowing more difficulty than other noise removal tasks. It is necessary to develop specific de-snowing models taking into account the characteristics of snowflakes.

A snowy image contains various sizes of snowflakes, while hand-crafted feature-based models are weak to learn them accurately and show quite poor generalization abilities. As for deep-learning based frameworks [128], where features are obtained by learning, the performance is improved by the accurate learning of the snow feature. However, the existing approaches still have difficulties in handling snowflakes of various sizes and shapes. Thus, a multi-scale structure is designed so that snowflakes of different sizes can be processed at different scales at the same time for feature learning. Fig. 7.2 shows the entire framework. The overall structure follows the idea of the image-conditioned generative adversarial network (image-cGAN), in which the de-snowing is achieved by generating completely snow-free images from snowy images, which can bet-

ter restore the details of blocked parts. Furthermore, with the help of capsule, a powerful structure proposed by Hinton et al. [64] which enables the feature learning process to be carried out under the comparison with the global image content, the proposed framework can detect and remove snowflakes of diverse shapes more accurately and efficiently.

Learning-based models, especially convolutional neural network based ones, have shown a great power on particle removal tasks. But the performance of these models are heavily dependent on the quality of prepared training datasets. For the snow removal task, Liu et al. constructed a snow dataset named Snow100K [128] by adding synthesized snow masks onto clean images using Photoshop [1]. But they did not examine the image content carefully when synthesizing the snowy images, resulting in some inappropriate training samples. It is improper to add snowflakes to some snow-meaningless pictures, such as indoor or underwater images. Having these inappropriate samples in training may lead to introducing unreasonable artifacts to the outputs when dealing with the real-world snowy images. Thus, the work proposes another large-scale snowy-image dataset, SnowySet, where meaningless samples are removed. Concurrently, a real-world snowy image dataset is collected for evaluation as well. All the details of SnowySet are presented in Section 7.4.2.

The main contributions are summarized as follow. 1) A multi-scale image conditional generative adversarial network (GAN) framework is proposed for single-image de-snowing, within which three-scaled branches are designed to effectively remove various snowflakes. 2) Capsule units are used to combine multi-scale branches in the encoder for image generation, which is proved to be effective for learning snow features. 3) To improve the stability and efficiency of the training process, a selective training method is introduced, where a pre-check mechanism is applied to the discriminative loss to avoid unstable loss signals. 4) A dataset, SnowySet, is constructed by adding snowflakes to snow-free images. The snowflakes vary greatly in shapes, sizes, transparencies and densities. To make the dataset more reliable and meaningful, indoor, underwater and other images that are inappropriate for being snowy have been removed. I have a publication in *IET Computer Vision*, “Multi-scale capsule generative adversarial network for snow removal”.

7.2 Pre-knowledge

Compared with de-raining, de-snowing does not attract so much attention. Some Researchers applied the de-raining models on snow removal directly [7, 33, 220, 267], while ignoring the unique characteristics of snowflakes such as various shapes, opaque and etc. Nevertheless, these special features make snowflakes more complicated than rain drops. In earlier stage, researchers believed that snow inherits certain rain characteristics and regarded the snow as a special form of rain. Hence, referencing to de-raining models, de-snowing models are designed to remove snow particles with the similar handcrafted features as well. Bossu et al. [14] separated the foreground and background by using Gaussian Mixture Model and constructed the snow features in foreground with the Histogram of Orientations of Snow Streaks (HOS). Rajderkar et al. [165] proposed an image decomposition approach based on Morphological component analysis, where the image would be decomposed into low and high frequency (LF/HF) parts by bilateral filters firstly and then applying dictionary learning and sparse coding methods to identify snow components later. Xu et al. [233] modelled snow particles by using colour assumptions and removed snow with a guidance image. As this approach may result in losing detailed information of local regions, they improved their framework with a refined guidance image in [232]. However, these prior-based methods can only model limited features and are weak in generalization. The hyper-parameters of guided filter in [232] may suits low transparency snowflakes, but failed for opaque ones.

Due to the limitation of handcrafted feature-based models, researchers developed learning-based models to learn effective features from existing noisy data distribution. The image translation frameworks have been applied for desnowing [45, 128]. Now, more and more researchers start to notice the differences between snowflakes and rain streaks, and realize that these differences may significantly affect the removing effect. The frameworks specifically designed for snow removal begin to attract more attention [110, 128].

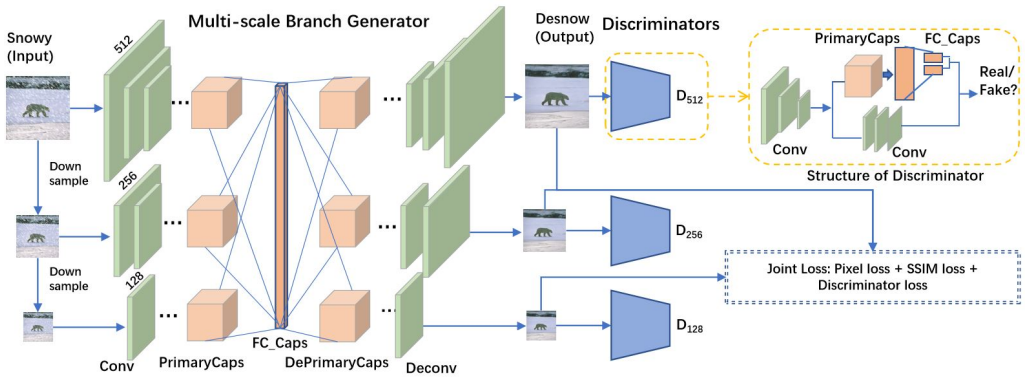


Figure 7.2: *The overall framework of the proposed multi-scale de-snowing model. The generator consists of three branches, at scales* $512 \times 512, 256 \times 256, 128 \times 128$. *Three discriminators are placed to discriminate the synthesized three scales of images. The capsule structure is implemented in both the generator and the discriminators. "Conv" and "Deconv" represent the convolutional and deconvolutional layers.*

7.3 Methodology

7.3.1 Overall Framework

In this work, a multi-scale model is built based on the image conditioned generative adversarial network (image-cGAN), which has been successfully applied on image translation [78, 235]. The overall framework is shown in Fig. 7.2. The generator (G) is constructed with multiple branches to tackle different sizes of snowflakes, since the snowflakes vary greatly in sizes, shapes, densities, orientations and trajectories. The discriminating part consists of three separate discriminators (Ds) with each targeting to distinguish one scale of generated image. Each branch of G takes a certain size of snowy image as its input, which is down-sampled from the original snowy image. After layers of convolutional filters for each branch, the encoded features are jointly connected to the decoder to synthesize the corresponding sizes of clean images.

The idea of the multi-scale structure came from the feature pyramid network [117], which enhances feature learning ability by setting pyramid-like multi-scale layers of filters. Thus, the multi-scale structure is designed for both G and D to improve their feature learning ability for various sizes of snowflakes. It is not the first to apply multi-scale receptive fields to noise removal. [110, 248] introduced the similar idea by implementing different sizes of convolutional filters to expand the receptive fields for de-raining, where the branch with larger convo-

lutional filters is designed for the feature learning of bigger raindrops. Compared with their approaches, the proposed method obtains better results and the computational burden is reduced, since the input images is rescaled for the small-scale branches.

In addition, inspired by the research of capsules [63, 179, 235], capsule units are implemented in both G and D to improve the feature representation ability of the overall framework. According to previous research, capsule units are able to learn the part-to-whole relationship [179], and therefore enhance a model's ability to identify objects on a global view. In the proposed model, the capsule units help to learn the 'hard samples' of the snowflakes by checking them against the image background with global information. This implementation benefits both G and D and improve the generating and discriminating abilities.

7.3.2 Multi-Scale Branches

The conventional generative adversarial network with a single network-based G is weak at tackling the de-snowing problem, because of the monotonicity of convolutional kernels and the variety of snowflakes. My solution enlarges the receptive fields by applying multiple scales of filters, as shown in Fig. 7.2. Taking the model of [248] for reference, a multi-scale structure is designed with three scaled branches, with scales of 512×512, 256×256 and 128×128. The snowy images are resized into the three scales, with each being processed by one branch of G. Each branch includes a PrimaryCaps layer to transform the layer maps into the capsule structure. All the PrimaryCaps capsules of the three branches are densely connected with an FC_Caps layer, which is responsible for merging features from three branches. The capsule structure is described in Section 7.3.3. The FC_Caps layer is connected with three decoding branches, aiming to synthesize three scales of images, 512×512, 256×256, and 128×128. Each branch consists of a DePrimaryCaps layer and several deconvolutional layers. Each deconvolutional layer enlarges the feature maps by a 2-stride filter. The 512×512 image is the de-snowing result, while the outputs from the branches of the 256×256 and 128×128 images are used to help the overall training of the framework.

To distinguish the three scales of synthesized images, three separate discriminators are placed, 512×512 (D_{512}), 256×256 (D_{256}), and 128×128

(D_{128}), each of which is constructed by several convolutional layers and a two-discriminating part (named two-branch D in [235]). The responsibility of each scale discriminator is to distinguish the synthesized de-snowing image from the snow-free images (ground truth) at the corresponding scale. The discriminating loss is back propagated to G to improve G's generating ability on every scale. In particular, the generated 512×512 images are rescaled into 256×256 and 128×128, and send them into D_{256} and D_{128} for multi-scale discriminations. In this way, the decoding branch of 512×512 could have useful supervision signals from all scales of Ds.

7.3.3 Capsule-based Generative Adversarial Network

Generally, three kinds of capsule layers compose the capsule implementation of the framework: the PrimaryCaps layer [179], the FC_Caps layer [179, 235] and the DePrimaryCaps layer [235]. The PrimaryCaps and DePrimaryCaps layers are responsible for the transformation between the conventional convolution based layer maps and the capsule layers. The FC_Caps layer learns and stores the features in capsule units. The connections between two capsule layers are weighted by dynamic routing [179], which finds the optimal way to flow the capsule information from the previous layer to the next one. In the multi-branch model, the FC_Caps layer within G is responsible for merging the features learnt by the three scale encoding branches and passing the features to different-scale decoders.

Capsule-implemented Generator

Shown as Fig. 7.2, the capsule based multi-scale branch generators are constructed with conventional convolutional/deconvolutional layers and capsule based layers. On each scale branch of G, the input image is processed by layers of 2-stride convolutional filters with each layer reducing the feature map size by half. Three PrimaryCaps layers are placed after the three scale convolutional encoders to transform the scaler neurons into the capsules with the same dimension of N. Then, all capsules from these three PrimaryCaps layers are able to be concatenated and fully connected to the FC_Caps layers, a set of capsules, via dynamic routing [179]. The layer maps are transformed by conventional convolutional

layers after the DePrimaryCaps layer [235]. Three DePrimaryCaps layers are placed to transform the FC_Caps layer into three decoding branches. After layers of 2-stride deconvolutional filters, de-snowed images are generated. In addition to the capsule layers, skip connections are placed on the corresponding convolutional and deconvolutional layers [78].

As shown in Fig. 7.2, of the three scales of outputs generated by the generator, the largest one (512×512) is the expected de-snowing result. The two smaller scales of outputs (256×256 and 128×128) are used in the calculation of the joint loss and provide training signals to the corresponding branches. The generator is trained branch by branch according to the optimization functions. The overall training procedure is presented in Table 7.1.

Capsule-implemented Discriminator

Three scales of discriminators with the same structure are placed to distinguish the three scales of synthesized images. The images are processed by several convolutional layers and then are discriminated by two parts, the capsule part [235] and the conventional patchGAN part [78], as introduced in Chapter 3. The two parts focus on discriminating the images in the global and the local view respectively. A balancing weight is placed to summarize the loss from the two discriminating parts, formulated as Eq. (3.6). Three discriminators are placed in the overall framework to distinguish the three scales of images generated by the three branches of Gs. Thus, the $\mathcal{L}_{disc}$ for the three discriminators are written as $\mathcal{L}_{disc\_512}$, $\mathcal{L}_{disc\_256}$, $\mathcal{L}_{disc\_128}$, formulated as $\mathcal{L}_{disc\_*}$ in Eq. (7.1)

$$\begin{aligned}\mathcal{L}_{disc\_*} = E_{x\sim P_{snowy}(x), y\sim P_{clean}(y)}[&\mathcal{L}_M(D_{*\_caps}(x,y),[1,0]) \\ &+ \mathcal{L}_M(D_{*\_caps}(x,G(x)),[0,1]) \\ &+ \lambda_1(log(1-D_{*\_patchG}(x,y)) \\ &+ log(D_{*\_patchG}(x,G_*(x))))],\end{aligned} \tag{7.1}$$

where $G_*(\cdot)$ and $D_*(\cdot)$ mean the forward calculation of the corresponding branch and $*$ represent the corresponding scales, 128, 256 and 512; $P_{snowy}(x)$ and $P_{clean}(y)$ represent the distribution of snowy and clean images in the training set, respectively.

The loss from D to optimize G is written in Eq. (7.2) with training signals

from both the discriminating parts.

$$\begin{aligned}\mathcal{L}_{genGAN} = E_{x\sim P_{snowy}(x)}[\mathcal{L}_M(D_{caps}(x, G(x)), [1, 0]) \\ + \lambda_1 log(1 - D_{patchG}(x, G(x)))].\end{aligned} \tag{7.2}$$

For the three scale branches of Gs, Eq. (7.2) is written separately as $\mathcal{L}_{genGAN\_512}$, $\mathcal{L}_{genGAN\_256}$ and $\mathcal{L}_{genGAN\_128}$, listed in Eq. (7.3) and (7.4).

As mentioned above, to better enhance the synthesis performance, the generated 512×512 images are down-sampled into 256×256 and 128×128, which are sent to D_{256} and D_{128} for down-scaling discrimination. A large size of snowflake, that is hard to be identified by D_{512}, could be discriminated more easily by D_{256} and D_{128} after down-scaling.

$$\begin{aligned}\mathcal{L}_{genGAN\_512} = E_{x\sim P_{snowy}(x)}[\mathcal{L}_M(D_{512\_caps}(x, G_{512}(x)), [1, 0]) \\ + \lambda_1 log(1 - D_{512\_patchG}(x, G_{512}(x))) \\ + \mathcal{L}_M(D_{256\_caps}(x, Down_{256}(G_{512}(x))), [1, 0]) \\ + \lambda_1 log(1 - D_{256\_patchG}(x, Down_{256}(G_{512}(x)))) \\ + \mathcal{L}_M(D_{128\_caps}(x, Down_{128}(G_{512}(x))), [1, 0]) \\ + \lambda_1 log(1 - D_{128\_patchG}(x, Down_{128}(G_{512}(x))))],\end{aligned} \tag{7.3}$$

where $Down * (\cdot)$ represents the down-sampling process into size $*$.

$$\begin{aligned}\mathcal{L}_{genGAN\_*} = E_{x\sim P_{snowy}(x)}[\mathcal{L}_M(D_{*\_caps}(x, G_*(x)), [1, 0]) \\ + \lambda_1 log(1 - D_{*\_patchG}(x, G_*(x)))],\end{aligned} \tag{7.4}$$

where $*$ represent the corresponding scales, 128 and 256.

7.3.4 Overall Optimization Functions

In the training phase, the generator is optimized by a joint loss of pixel loss and SSIM loss [270], along with the supervision signal from the multiple discriminators. The pixel loss is to reduce the overall difference between the synthesized image and the ground truth by calculating the L_1 norm distance. The SSIM loss is to minimize the structural error of the generated outputs from the ground truth. Different losses contribute to the final loss differently with a series of loss

balancing weights.

The supervision signal for the Gs is from the combination of multiple losses, shown in Eq. (7.5).

$$\mathcal{L}_{gen} = \mathcal{L}_{genGAN} + \lambda_2 \mathcal{L}_{pixel} + \lambda_3 \mathcal{L}_{SSIM} \tag{7.5}$$

Here, $\mathcal{L}_{pixel}$ is the pixel loss, which is the L_1 distance; $\mathcal{L}_{SSIM}$ is the SSIM loss calculated as $1 - SSIM$, shown as

$$\mathcal{L}_{pixel} = \mathcal{L}_1(G(x), y), \tag{7.6}$$

and

$$\mathcal{L}_{SSIM} = 1 - SSIM(G(x), y), \tag{7.7}$$

where $\mathcal{L}_1$ is the L_1 distance and $SSIM(\cdot)$ means the calculation of the SSIM score for the two images.

The three branches within the G are optimized in terms of three overall loss functions, shown in Eq.

$$\begin{aligned} \mathcal{L}_{gen\_*} = \mathcal{L}_{genGAN\_*} + \lambda_2 \mathcal{L}_1(G_*(x), y) \\ + \lambda_3(1 - SSIM(G_*(x), y)). \end{aligned} \tag{7.8}$$

where $*$ represent the corresponding scales, 128, 256 and 512.

G and Ds are trained iteratively, shown in detail in Table 7.1. The parameters of the FC_Caps layer within G are updated along with G_{512}. When one branch is being optimized, the parameters of the other branches remain unchanged. Eq. (7.8) is calculated in the form of Eq. (7.9).

7.3.5 Selective Training

A typical way to train a GAN is to iteratively update the gradients for the weights (parameters) of G and D. It is obvious that both G and D are not trained well during the early iterations. Indeed, the discriminator loss fluctuates over the whole training procedure. When G generates "better images" after several optimization steps, the loss of D increases. At this moment, D needs some "time" (more training iterations) to optimize itself and learns to distinguish the generated "better image" and the ground truth.

Table 7.1: The training algorithm of the multi-scale capsule-cGAN.

Input:
Hyper-parameter setting
Training samples snowy images x and clean images y
for $i = 1; i <= training\_iterations$ **do**
Forward samples through Gs and Ds;
for j **in** $\{128, 256, 512\}$ **do**
Optimize D_j by calculating gradients on Eq. (7.1);
for j **in** $\{128, 256, 512\}$ **do**
Optimize G_j by calculating gradients on Eq. (7.9);

The discriminating signal passed backward to G is not always effective, especially when D is not trained well with a larger discriminating loss. This gives G the wrong signal, which leads G to learn improper feature representations and produce low-quality images. After trials of training, I found that the loss fluctuation of D has a big probability of causing the training to fail sometimes by encouraging G to produce a black or a meaningless image. To reduce the side effect of these loss fluctuations, one solution is to use extra training signals (L_1 loss) to guide the learning of G, which is applied in the most recent research [78, 235]. Another solution is to set a bigger learning rate for D to speed up the optimization of D, thus shortening the ineffective learning time of G. Both methods can help the training, but do not solve the problem completely. In addition, the second method has an obvious drawback: it causes an instability of D and brings obstacles to D in arriving at the optimum. In the present paper, except for the extra training signals (L_1 loss and SSIM loss described in Section 7.3.4), a selective training method is introduced to have a pre-check on the loss of D, named selective training. The training signal from D can be applied to G only when D is checked with a low D loss.

The judgement criterion is to check whether the discriminative loss, $\mathcal{L}_{disc}$, is below a reasonable value. Set an empirical value ξ_{dis} as the threshold, and $\mathcal{L}_{genGAN}$ is back-propagated when $\mathcal{L}_{disc}$ is lower than ξ_{dis}. Otherwise, the gradients calculated from $\mathcal{L}_{genGAN}$ will not be applied. Therefore, Eq. (7.8) are rewritten in the form of Eq. (7.9).

$$\mathcal{L}_{gen*} = \begin{cases} \mathcal{L}_{genGAN*} + \lambda_2 \mathcal{L}_{pixel*} + \lambda_3 \mathcal{L}_{SSIM*}, if \mathcal{L}_{disc*} < \xi_{dis} \\ \lambda_2 \mathcal{L}_{pixel*} + \lambda_3 \mathcal{L}_{SSIM*}, \quad if \mathcal{L}_{disc*} \geq \xi_{dis} \end{cases} \tag{7.9}$$

where the $*$ means the corresponding scale (128, 256 or 512).

7.4 Experiment

7.4.1 Experiment Setting

The proposed model (MC-DS) is compared with auto-encoder(AE) [230], Pix2pix [78], cycleGAN [275], DID-MDN [248], DesnowNet [128], LSTM-GAN [61] and ComposGAN [116] on two datasets, Snow100K [128] and the constructed SnowySet. Due to the lack of the source code of DesnowNet, LSTM-GAN and ComposGAN, their frameworks are re-implemented in tensorflow with their recommended hyper-parameters. The experiments are conducted on both synthetic and real-world snowy images. MSE, L1, PSNR and SSIM [270] are used for quantitative evaluation. The visual de-snowing samples of both the synthetic and the real-world datasets are represent. To check the effectiveness of the multi-scale structure, an ablation study is set by examining a framework with a single-scale generator constructed.

7.4.2 Dataset

Snow100K

Snow100K [128] contains 100,000 synthesized snowy images, 50,000 for training and 50,000 for testing. The snow masks used for synthesizing the snowy images are also provided. According to the density of snowflakes, the testing set is separated into three subsets, Snow100K-S, Snow100K-M and Snow100K-L, representing Small, Medium and Large, respectively. Besides the synthesized images, [128] also provides 1,329 realistic snowy images downloaded via the Flickr api.

SnowySet

To increase the diversity of snowflakes, I make another dataset, SnowySet, which is synthesized with snowflakes in great variations of the shapes, sizes, density, transparency and floating trajectories. The clean images are selected from BSDS500 [5], UDIC.v2 and Snow100K [128] by removing the images that are meaningless to the snowy weather (e.g. indoor, underwater, close-view or water sports images), 4,236 for training and 1,040 for testing. The snow masks synthesized in Photoshop [1] are added onto the clean images to synthesize the snowy images, resulting in 42,360 training samples and 10,400 testing samples. Similar to Snow100K and inspired by [236], the dataset is prepared with three density levels, SnowySet-L (Large), SnowySet-M (Medium) and SnowySet-S (Small), to simulate different densities of snowflakes in reality.

100 real-world snowy images are collected from Internet for testing with different image contents and various snowflakes. Comparing with the realistic snowy images provided by Snow100K, the ones in SnowySet are with snowflakes of more variations. The datasets can be downloaded from https://yang-fei.github.io/caps-multiscale-desnowing/.

7.4.3 Implementation Details

For data augmentation, the images are resized into 584×584 before being randomly cropped into 512×512 patches. The testing images are resized into 512×512 before being forwarded through the well-trained generator.

The kernel sizes and strides in all convolutional/deconvolutional layers are fixed into 5×5 and 2. The height and width of feature maps are reduced into half after each convolution layer. The number of the convolution layers for each branch varies according to the inputting scale so that 16×16 feature maps are obtained before being connected to the PrimaryCaps layer. The filter number (layer channels) of each convolutional layer doubles that of the previous layer, with the first layer channel of 32. The layers of the decoder branches of G (deconvolutional layer) have the same settings corresponding to the encoder branches. According to [179, 235], the capsule structure is designed with the dimension of 8 for all capsules and set capsule numbers in PrimaryCaps, FC_Caps, DePrimaryCaps layers of G to 16, 48, and 16, respectively.

The first several convolutional layers of Ds are similar to the encoder branches of G, outputting the feature maps in the size of 32×32 before being sent into the two-sub-branch discriminating part. The PrimaryCaps and the FC_Caps layers of Ds contain 32 and 16 capsules appended by another two capsules as the output. The patchGAN branch of Ds consists three convolutional layers with the last one outputting a 9×9 single channel feature map.

The training is conducted with a batch size of 8 for 100 epochs where the learning rate is decayed by 0.1 per 20 epochs from a initialization of 0.0002. The loss balancing weights are $\lambda_1 = 0.1, \lambda_2 = 10, \lambda_3 = 5, \lambda_4 = 0.1$. The experiments are executed in Tensorflow (python) on NVIDIA GPU Tesla V100 (32GB).

7.4.4 Quantitative Results

The quantitative results of the two synthetic datasets are shown in Tables 7.2 and 7.3. The subsets are evaluated separately. The overall results are calculated on the whole datasets, which are equivalent to the average of the subsets. The boldface values indicate the best results.

Table 7.2: Evaluation comparison on Snow100K [128].

Dataset	Models	PSNR	SSIM	nrmse	L1
Snow100K-L	AE[230]	21.50	0.7528	0.1938	0.1773
	Pix2pix[78]	25.83	0.8490	0.1170	0.1534
	cycleGAN[275]	19.21	0.6575	0.3596	0.2876
	DID-MDN[248]	26.34	0.8969	0.1097	0.0897
	DesnowNet [128]	27.17	0.8983	0.1101	0.0398
	LSTM-GAN [61]	26.12	0.8616	0.1214	0.1102
	ComposGAN [116]	**29.54**	0.9021	0.1001	**0.0301**
	MC-DS	28.14	**0.9032**	**0.0995**	0.0304
Snow100K-M	AE[230]	23.38	0.8422	0.1865	0.1298
	Pix2pix[78]	29.29	0.9261	0.0798	0.0243
	cycleGAN[275]	20.30	0.7377	0.2215	0.2654
	DID-MDN[248]	28.79	0.9423	0.0841	0.0285
	DesnowNet [128]	30.87	0.9409	0.0775	0.0210
	LSTM-GAN [61]	30.84	0.9384	0.0721	0.0212
	ComposGAN [116]	31.21	0.9431	0.0622	0.0218
	MC-DS	**31.59**	**0.9444**	**0.0619**	**0.0208**
Snow100K-S	AE[230]	23.60	0.8444	0.2015	0.1472
	Pix2pix[78]	29.90	0.9232	0.1201	0.0785
	cycleGAN[275]	21.40	0.7825	0.2012	0.2043
	DID-MDN[248]	30.15	0.9521	0.0765	0.0265
	DesnowNet [128]	32.33	0.9500	0.0599	0.0210
	LSTM-GAN [61]	30.09	0.9411	0.0914	0.0255
	ComposGAN [116]	30.43	0.9612	0.0641	0.0207
	MC-DS	**32.49**	**0.9674**	**0.0557**	**0.0174**
Overall	AE[230]	22.83	0.8131	0.1939	0.1514
	Pix2pix[78]	28.34	0.8994	0.1056	0.0854
	cycleGAN[275]	20.30	0.7259	0.2607	0.2524
	DID-MDN[248]	28.42	0.9304	0.0901	0.0482
	DesnowNet [128]	30.11	0.9296	0.0825	0.0272
	LSTM-GAN [61]	29.01	0.9137	0.0949	0.0523
	ComposGAN [116]	30.39	0.9355	0.0754	0.0242
	MC-DS	**30.74**	**0.9383**	**0.0724**	**0.0229**

Table 7.3: Evaluation comparison on SnowySet.

Dataset	Models	PSNR	SSIM	nrmse	L1
SnowySet-L	AE[230]	20.50	0.7433	0.2021	0.1141
	Pix2pix[78]	24.88	0.8520	0.1221	0.0864
	cycleGAN[275]	19.92	0.6124	0.2569	0.1335
	DID-MDN[248]	26.12	0.8922	0.1088	0.0878
	DesnowNet [128]	26.35	0.8821	0.1102	0.0874
	LSTM-GAN [61]	25.11	0.8641	0.0998	0.0841
	ComposGAN [116]	26.48	0.8972	0.0947	0.0695
	MC-DS	**27.40**	**0.8974**	**0.0932**	**0.0613**
SnowySet-M	AE[230]	23.33	0.8376	0.1965	0.1294
	Pix2pix[78]	28.20	0.9201	0.0987	0.0879
	cycleGAN[275]	21.22	0.7122	0.2543	0.2231
	DID-MDN[248]	28.21	0.9401	0.0879	0.0483
	DesnowNet [128]	27.98	0.9342	0.0754	0.0531
	LSTM-GAN [61]	27.14	0.9274	0.0911	0.0784
	ComposGAN [116]	28.95	0.9379	0.0841	0.0399
	MC-DS	**31.49**	**0.9549**	**0.0621**	**0.0283**
SnowySet-S	AE[230]	26.32	0.8897	0.1120	0.0909
	Pix2pix[78]	29.23	0.9123	0.0889	0.0456
	cycleGAN[275]	24.76	0.8212	0.1876	0.1534
	DID-MDN[248]	30.77	0.9512	0.0721	0.0219
	DesnowNet [128]	30.98	0.9512	0.0731	0.0201
	LSTM-GAN [61]	29.88	0.9341	0.0712	0.0265
	ComposGAN [116]	31.78	0.9599	0.0645	0.0204
	MC-DS	**33.42**	**0.9620**	**0.0596**	**0.0162**
Overall	AE[230]	23.38	0.8235	0.1702	0.1114
	Pix2pix[78]	27.44	0.8948	0.1032	0.0733
	cycleGAN[275]	21.96	0.7152	0.2329	0.1700
	DID-MDN[248]	28.36	0.9274	0.0896	0.0526
	DesnowNet [128]	28.43	0.9225	0.0862	0.0535
	LSTM-GAN [61]	27.95	0.9085	0.0873	0.0630
	ComposGAN [116]	29.07	0.9316	0.0811	0.0432
	MC-DS	**30.77**	**0.9381**	**0.0716**	**0.0353**

In overall, the proposed MC-DS performs the best and ComposGAN [116] ranks the second. DID-MDN [248] and DesnowNet [128] produce comparable results, which are still better than LSTM-GAN [61]. The comparisons are observed more obviously on the two heavy-snow subsets, Snow100K-L and SnowySet-L. The performance of AE [230] and cycleGAN [275] cannot meet the state-of-the-art, due to the simpleness of AE structure and the lack of paired information for cycleGAN [275]. Pix2pix [78] gives quite comparable results on Medium and Small snow subsets, which shows that an U-net generator with a patchGAN discriminator [78] is capable to handle the lightweight work of de-snowing.

In Table 7.2, DID-MDN [248] produces quite good SSIM scores on Snow100K-M and Snow100K-S, benefiting from its refinement layers to improve the image quality. But it suffers the degradation for heave-snow images on Sonw100K-L with worse scores than DesnowNet [128] and ComposGAN [116]. ComposGAN [116] gives the best PSNR on Snow100K-L with 29.54, but the SSIM is still lower than ours.

In Table 7.3, DesnowNet [128] and DID-MDN [248] give comparable results with ComposGAN [116], but still perform worse than ours. SnowySet contains snowflakes of more variations, which cause more difficulties to de-snowing models. DID-MDN [248] with density estimation contains the ability to learn the feature of multi-density snowflakes. But the proposed MC-DS with multi-scale branches shows better ability of learning the variations of snowflakes. On SnowySet-L, LSTM-GAN [61] gives quite low SSIM of 0.8641, showing that the LSTM structure may not help a lot to learn heavy-snow features. The patchGAN-based discriminator encourages Pix2pix [78] to learn more local details and texture features, resulting in a reasonable L1 score on SnowySet-L.

7.4.5 Visual Comparison Results

The visual inspection of the de-snowing results is shown in Fig. 7.3, 7.4, 7.5 and 7.6. Samples with diverse image contents are selected from both the synthetic and the real-world datasets.

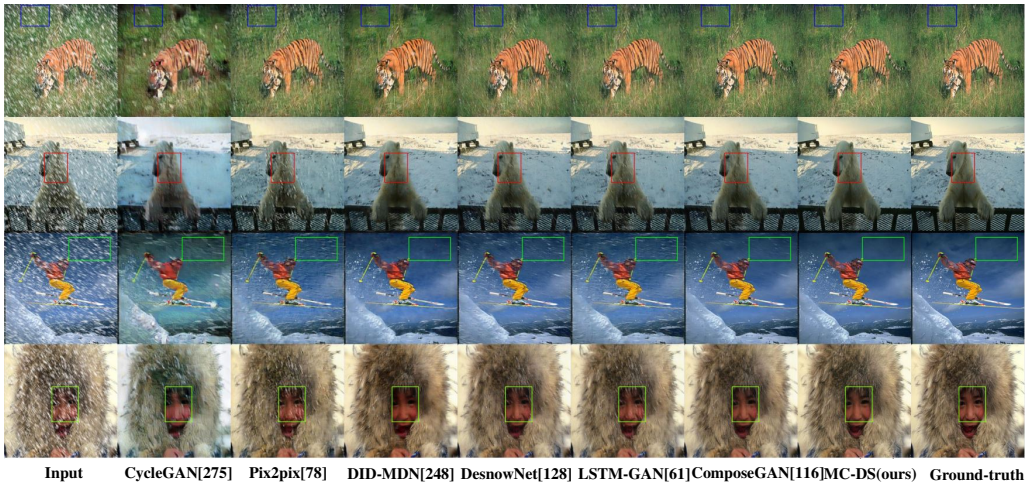


Figure 7.3: *The de-snowing results of different frameworks on testing images from SnowySet. The proposed model produces the best clear sky and recovers more details, such as the plant and the face. Larger patches are shown in Fig. 7.4.* **Best view on screen.**

Synthetic Images

Fig. 7.3 and 7.5 show the de-snowing results on synthetic images from the testing datasets of SnowySet and Snow100K [128], respectively. Fig. 7.4 shows the enlarged details of images in Fig. 7.3.

CycleGAN [275] and Pix2pix [78] fail to remove the snowflakes and CycleGAN even changes the global colour into Green effects. The fourth columns of the Fig. 7.3 and 7.5 present better de-snowing results, but some local details are also eliminated. The images are over smoothed by DID-MDN [248], which is quite obvious in the first image of Fig. 7.4. For a comparison, ComposGAN [116] and MC-DS restore more details of the grass. The same phenomenon can be seen in the fourth image of Fig. 7.3, the first and the second images of Fig. 7.5. Compared with DesnowNet [128] and LSTM-GAN [61], the proposed MC-DS removes the snowflakes more completely, which can be examined by checking the sky regions of the third sample in Fig. 7.4. ComposGAN's result is close to ours, but some snowflakes are still left. The sky regions from the first sample in Fig. 7.5 presents the similar conclusion. There are snow marks that are not removed on the bear face and the human face from the second and forth images of Fig. 7.3 and 7.4.

The proposed MC-DS gives the best de-snowing results close to the ground

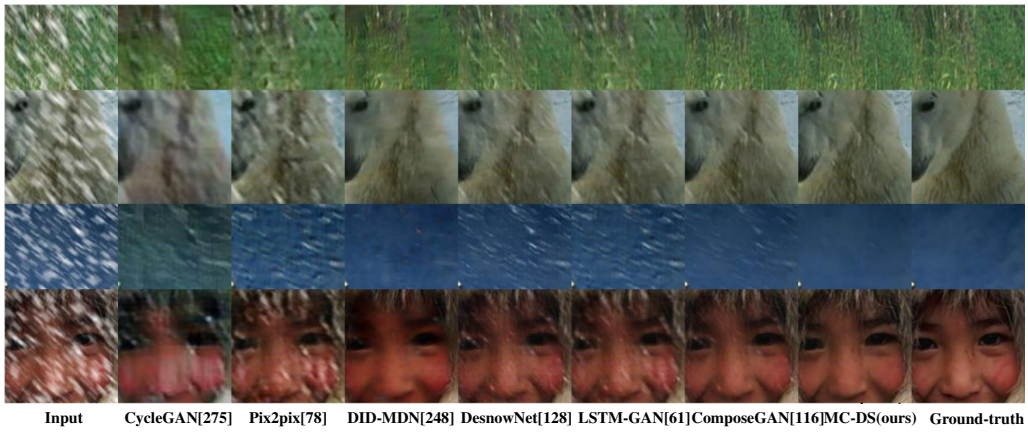


Figure 7.4: *The detail comparison of different frameworks on testing images from SnowySet.* **Best view on screen.**

truth by successfully removing the snowflakes on various kinds of background and recovering most details of the image content.

Real-world Images

Fig. 7.6 shows some de-snowing results on real-world snowy images. With the similar phenomenon to that on synthetic images, CycleGAN [275] produces blurry images with Green effects. Pix2pix [78] leaves some snowflakes undetected apparently. Compared with DID-MDN [248], DesnowNet [128], LSTM-GAN [61] and ComposGAN [116] from the first and second images of Fig. 7.6, the proposed MC-DS produces better images by removing more snowflakes and generating smoother sky. The down-left corner of the third sample shows two people getting on the black car, which are synthesized blurry by DID-MDN [248], DesnowNet [128] and LSTM-GAN [116]. ComposGAN's result is much worse. On the forth sample, DID-MDN [248] considers the yellow tie on the human in black clothes as a snowflake and removes it totally from the outputting result. For the comparison, MC-DS recovers it much better. From these samples we can see the superiority of MC-DS in distinguishing the image contents against various snowflakes.

7.4.6 Ablation Study

To check the effectiveness of the multi-scale structure, a ablation study is set with a single-scale structure by removing the branches of scale_128 and scale_256,

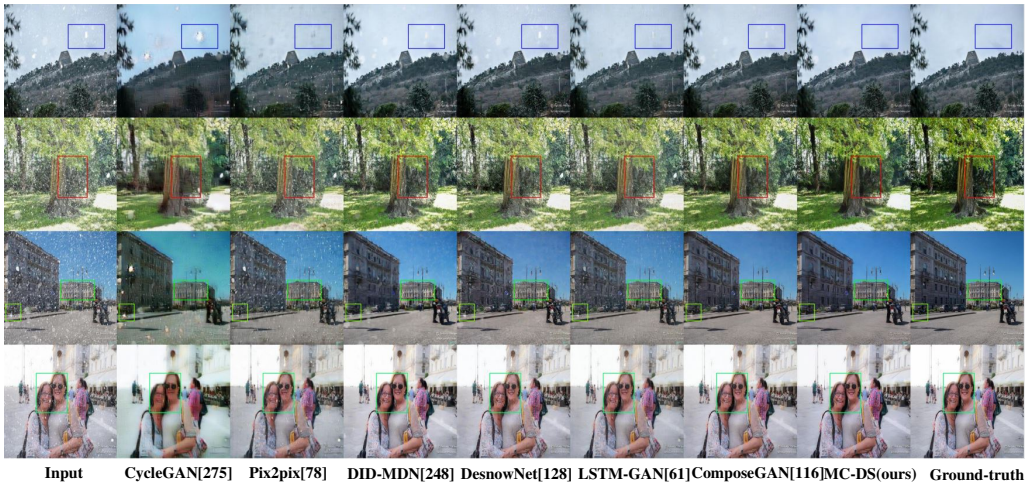


Figure 7.5: *The de-snowing results of different frameworks on testing images from Snow100K. The proposed model recovers the best image details by observing the tree of the second sample and the building of the third sample.* **Best view on screen.**

Table 7.4: Ablation Study on Multi-scale Structure

Dataset	Models	PSNR	SSIM	nrmse	L1
Snow100K-L	Multi_scale	**27.54**	**0.9012**	**0.0995**	**0.0304**
	Single_scale	26.31	0.8910	0.1129	0.1009
SnowySet-L	Multi_scale	**27.40**	**0.8974**	**0.0932**	**0.0613**
	Single_scale	25.31	0.8803	0.1143	0.0899

marked as Single_scale. The proposed MC-DS is marked as Multi_scale here. Since the heavy-snow images contain snowflakes of more variations, the ablation study is conducted on Snow100K-L and SnowySet-L. The results are shown in Table 7.4.

Without the multi-scale structure, the performance of MC-DS-single_scale decreases with all the evaluation metrics on the two subsets. The single-scale model faces difficulties in processing diverse sizes of snowflakes from different image contents. The Single_scale still obtains comparable results with DID-MDN [248] and DesnowNet [128] of Table 7.2 and 7.3, which benefits by the contributions of the SSIM loss and the capsule-based structure.

7.5 Summary

A multi-scale image-cGAN is built to remove snowflakes from snowy images. Compared with existing de-snowing models, the proposed MC-DS outperforms

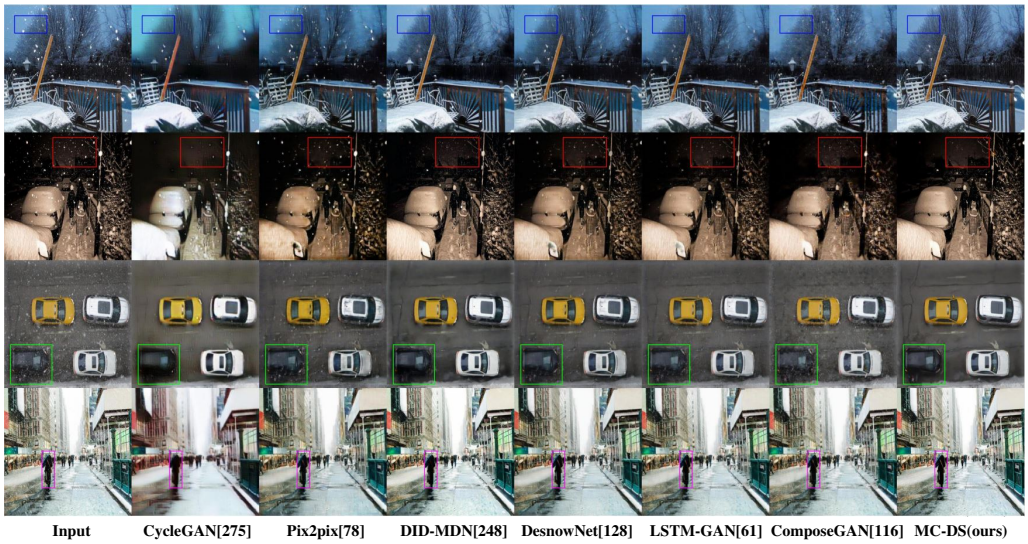


Figure 7.6: *The de-snowing of different frameworks on real-world images. The first two samples show that the proposed model removes the snowflakes best. The car of the third sample and the human with bag of the fourth sample show that the proposed model recovers more image detail.* **Best view on screen.**

the state-of-the-art. Capsule layers are implemented to fuse the features of different branches and learn the part-to-whole relationship between local regions and the global image content.

Although MC-DS performs well in the experiments, there are more to be explore and discussed. Firstly, better methods to connect the capsule block with convolutional/deconvolutional layers can be explored by designing new structures of PrimaryCaps and DePrimaryCaps layers [179, 235]. Secondly, effective routing algorithms might be developed for the feature fusion of the three branches if better capsule structures are constructed.

The experimental results demonstrate the effectiveness of the multi-scale structure in removing various sizes of snowflakes. And the propose method outperforms state-of-the-art models. I have a publication about this work, "Multi-scale capsule generative adversarial network for snow removal" in IET Computer Vision [J] in 2020.

Chapter 8

Quality Type and Quality Level Prediction

8.1 Introduction

Previous chapters have discussed the work of image synthesis on applications of image rendering, de-raining, de-hazing and de-snowing. The following two chapters will introduce the work on image quality estimation, which are used for the assessment of translated images. In this chapter, the quality level estimation is discussed.

Object detection and recognition have achieved significant progress in recent years. In real-world application scenarios, motion blur, lossy image compression, insufficient spatial resolution caused by out of focus or objects being too far away and other factors can all result in poor image quality problems. Fig. 8.1 shows two typical low-quality versions of an image caused by lossy image compression and low spatial resolution. This work will explicitly show that image quality is an important factor that will affect the performances of object recognition and detection algorithms.

The discussion begins by introducing a motivating experiment. Firstly take a publicly available face detection dataset [96], and then downscale each image in the dataset from 512×512 into 40×40 pixels, resulting in a low-quality dataset. I then take one of the latest deep learning based face detection techniques [200] and train a high image quality detector (using the original resolution images) and a low image quality detector (using the 40 by 40 pixels images). Then the

Clear **JPEG compression** **Low resolution**

Figure 8.1: *A good quality image (left), its low quality JPEG compressed version (middle) and its low spatial resolution version (right)*

.

two detectors are tested on both high and low-quality images. The results are shown in Fig. 8.2. It is seen that the high image quality detector works very well on the high-quality testing images; however, its performance is much poorer for the low-quality testing images. Similarly, it can be seen that the low image quality detector performs very well on the low-quality testing images, but its performance deteriorates significantly for the high-quality images. This example tells us, it is not the image quality itself that is the most important in designing a good face detector, but rather the quality of the images used in training the detector should be similar to that of images in the testing set. In many ways, this is to be expected and also makes good sense, nevertheless, this motivating experiment has confirmed that image quality needs to be considered in designing an image analysis solution.

Using face detection and recognition as specific applications, a quality classified image analysis framework is developed. To give this study a better focus, two specific types of image quality issues are considered, one caused by image compression (specifically JPEG compression), and the other caused by low spatial resolution.

Partly motivated by the results in Fig. 8.2, which suggests that it is not the image quality itself that is the most important, but rather the quality of the images used to train the analyser should be similar to that of those on which the analyser will be tested. Based on this observation, my strategy is first to classify the input images into different quality classes and then designs a suitable image analyser for an individual image quality class.

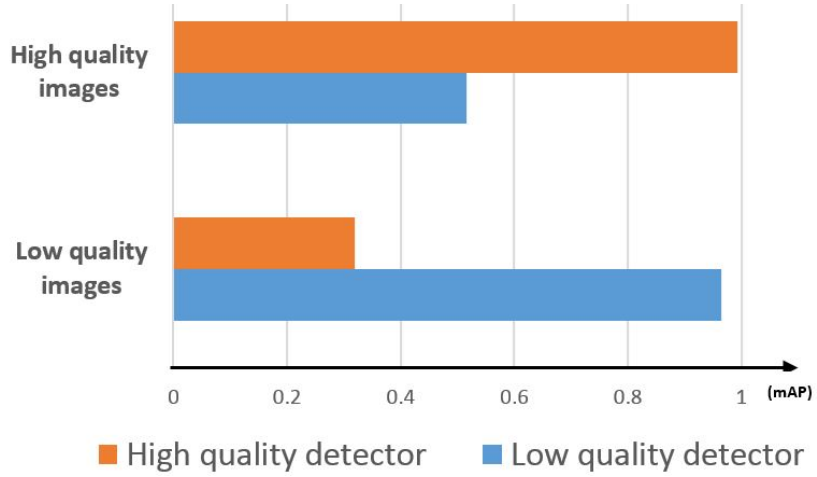


Figure 8.2: *Face detection performances (mAP - mean average precision) for detectors trained and tested on images of different qualities.*

The contributions are summarised as follow. First, an image quality classified framework is proposed which can better handle images of mixed qualities. Second, it is shown that convolution neural network based image quality classifier can first determine if an image is of good or poor quality; and then for poor quality images. It can not only determine whether it is caused by JPEG compression or by low spatial resolution, but also the severity of compression and down-sampling. Third, a method is proposed that first designs separate object detectors or classifiers for different classes of image quality in the training stage, and then in the testing stage, automatically sends an image to the first few most suitable individual detectors or classifiers whose outputs are then fused together to improve performances.

8.2 Pre-knowledge

The image quality estimation problem has been studied for a long time in the area of image processing. It learns the visual difference caused by image quality, such as lossy compression, brightness, sharpness, and resolution. Several convolutional neural networks based methods have been developed to assess the quality of whole image [43, 85]. In the research of Image Quality Assessment (IQA)[135], a rating score is obtained by solving a regression problem. However, all the quality scores are labelled by human beings [67, 68], which is subjective. And all the images are labelled discarding their specific quality classes, resulting

in the images with the same quality score containing different quality classes and visual appearance, which also makes the network hard to converge. Meanwhile, researchers are also conducted in the field of JPEG compression related assessment. [156] focused on detecting whether an image is compressed with JPEG. Further research including estimating the quantization table [130], and removing blocking artifacts [40]. These methods either rely on external information from header file or special designed hand-crafted features for detecting blocking artifacts. To my knowledge, no one has ever used CNN based methods to estimate the detailed quality information, concerning the quality types and levels.

There are only a few works that concern the effects of image quality in solving detection or recognition problems [34, 55, 88, 126]. Two types of research work are summarized. One is to analyse how much the image quality can affect the performance of standard object or face problems[34, 88]. They compare the model robustness by testing the models on manually decreased low-quality images.

The other is to develop methods to overcome the low-quality problem in real application scenarios through identifying the low-quality images discarding their quality classes and the corresponding severity. [126] proposed a quality assessment network within an end-to-end training framework in human re-id and face recognition problems. Instead of a single image, the network regards a set of images or a sequence of images as a recognition subject entity and handles the set to set recognition by predicting the quality score of the image within each set. A low-quality image is given a small score and, hence, reducing its impact on the whole set. Similarly, [55] concerned the image quality problem in facial landmark detection by selecting the high-quality image in a video sequence. The low-quality frame problem is addressed by locating and replacing with high-quality face in the previous video frames. They also narrowed the quality causes within face patches, which assumes the face can be correctly detected under poor image quality.

The proposed method belongs to the second type. However, instead of purely identifying and discarding the low-quality images, the main strength is to predict the quality classes as well as their severity explicitly and handle them differently with the specific prior knowledge. It can be widely adopted to handle unknown quality images any image-based detection and recognition problems,

Figure 8.3: *Overall framework for quality level prediction applied in vision systems.*

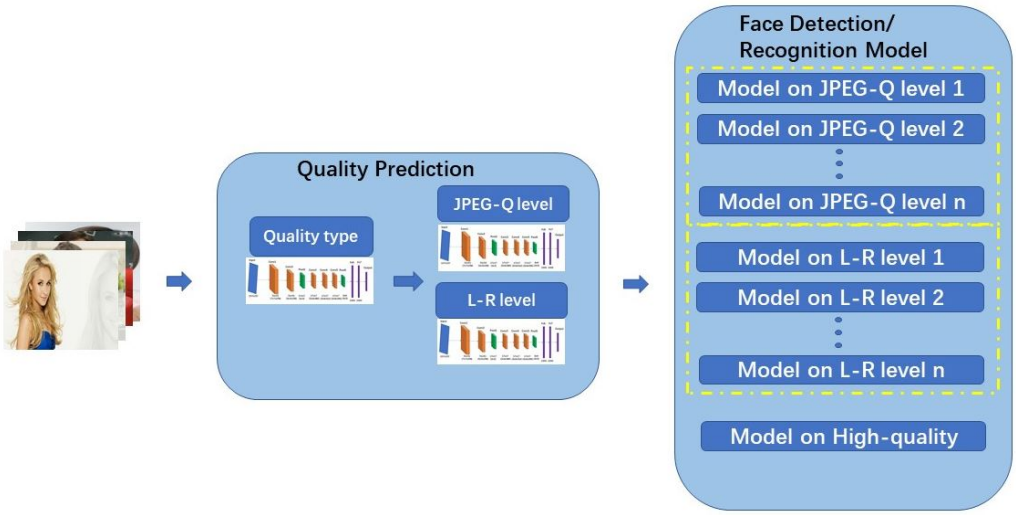


Figure 8.4: *Overall framework for quality level prediction and quality classified image face detection/recognition. The test image is sent into the quality prediction module to estimate the degradation types and the corresponding levels. Then k detection/recognition models are selected for prediction.*

without an additional requirement for manually labelled data.

8.3 Image Quality Analysis

8.3.1 Overall framework

The proposed framework is shown in Fig. 8.3. The images are estimated for the quality type and quality level before being sent into other vision processing systems. This work uses face detection/recognition as specific case studies. In particular, consider two types of image quality problems, JPEG compression, and low-resolution. For each type of quality issue, I also consider the severity of the quality issue, and call this the quality level. Firstly, three main image quality classes are defined: Good Quality (G), Bad Quality JPEG compression (BJ) and Bad Quality low resolution (BL). For the BJ and BL class, two subsets are defined based on the level of severity of compression or low-resolution, $\{BJ_i, i = 1, 2, 3, ..., m\}$ and $\{BL_j, j = 1, 2, 3, ..., n\}$. Therefore, an image can

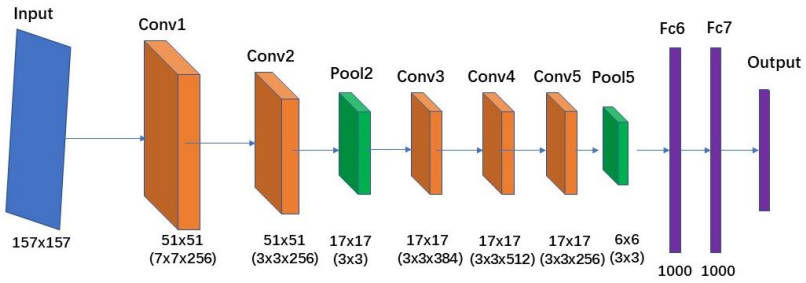


Figure 8.5: *Quality prediction network architecture.*

be classified based on its quality into one of the classes in the following quality class set $C = \{G, BJ_i, BL_j; i = 1, 2, ...m, j = 1, 2, ...n\}$.

As demonstrated in the preliminary study (see Fig. 8.2), image quality is an essential issue in image analysis. The question is how should I deal with it.

A quality classification approach is discussed and an image is classified into one of the quality classes defined. Once the quality class of an image is determined, an image analyser can be designed, which is suitable for that image quality class. The solution framework is shown in Fig. 8.4. An image is firstly estimated by a neural network into one of the three first-level quality classes $\{G, BJ, BL\}$, then, it is classified into the subclasses $\{BJ_i, i = 1, 2, 3, ..., m\}$, and a third deep learning network to classify those in the BL class into their subclasses $\{BL_j, j = 1, 2, 3, ..., n\}$.

Multiple detection/recognition models are trained using images of different quality levels. For a given input image, the first few most likely quality classes the input belongs are selected for the results fusion.

8.3.2 Quality prediction network

All the three prediction networks share similar network architecture. Fig. 8.5 illustrates the details of my proposed quality prediction convolutional neural network. Image patches are randomly cropped from the input image. The image patch size is set as 157×157. Similar to a typical CNN, five convolutional and two fully connected layers are stacked. To reduce the feature dimension, a 3-stride pooling layer is put after the second and fifth convolutional layers, respectively. Two fully connected layers (Fc6 and Fc7), containing 1000 neurons each, are followed by the final pooling layer. A Softmax output is used to gener-

ate the first-level class scores $\{p(G), p(BJ), p(BL)\}$, as well as the second-level class scores, $\{p(BJ_i), i = 1, 2, 3, ..., m\}$ and $\{p(BL_j), j = 1, 2, 3, ..., n\}$.

8.3.3 Target model selection and result fusion

The class score vectors $p(\cdot)$, which come from the output of the three quality prediction networks, is fused to generate a single quality score vector P_c with

$$P_C = \{p(G) * 1, p(BJ) * p(BJ_i), p(BL) * p(BL_j), \\ i = 1, 2, 3, ..., m, j = 1, 2, 3, ..., n\}$$

P_C indicates the probability of the input image patch belonging to each quality class, according to which, I select the top K corresponding trained models to form the final image analyser.

In a quality classified face detection application, the series of face bounding boxes produced by the top K face detection models are merged, where the models are trained on different quality-level datasets using an existing face detection method. A Non Maximum Suppression [146] method is applied to locate the redundant boxes.

Similarly, in a quality classified face recognition application, face identity scores coming from the top K face recognition models are aggregate with their weight calculated according to P_C.

8.4 Experiments

Two sets of experiments are conducted. The first one presents the quality prediction results to show how well my proposed network can learn quality feature and accurately predict the quality classes. In the second one, I evaluate my proposed quality classified image analysis framework to demonstrate its effectiveness in face detection and recognition applications. The experiments are done with Caffe (Matlab).

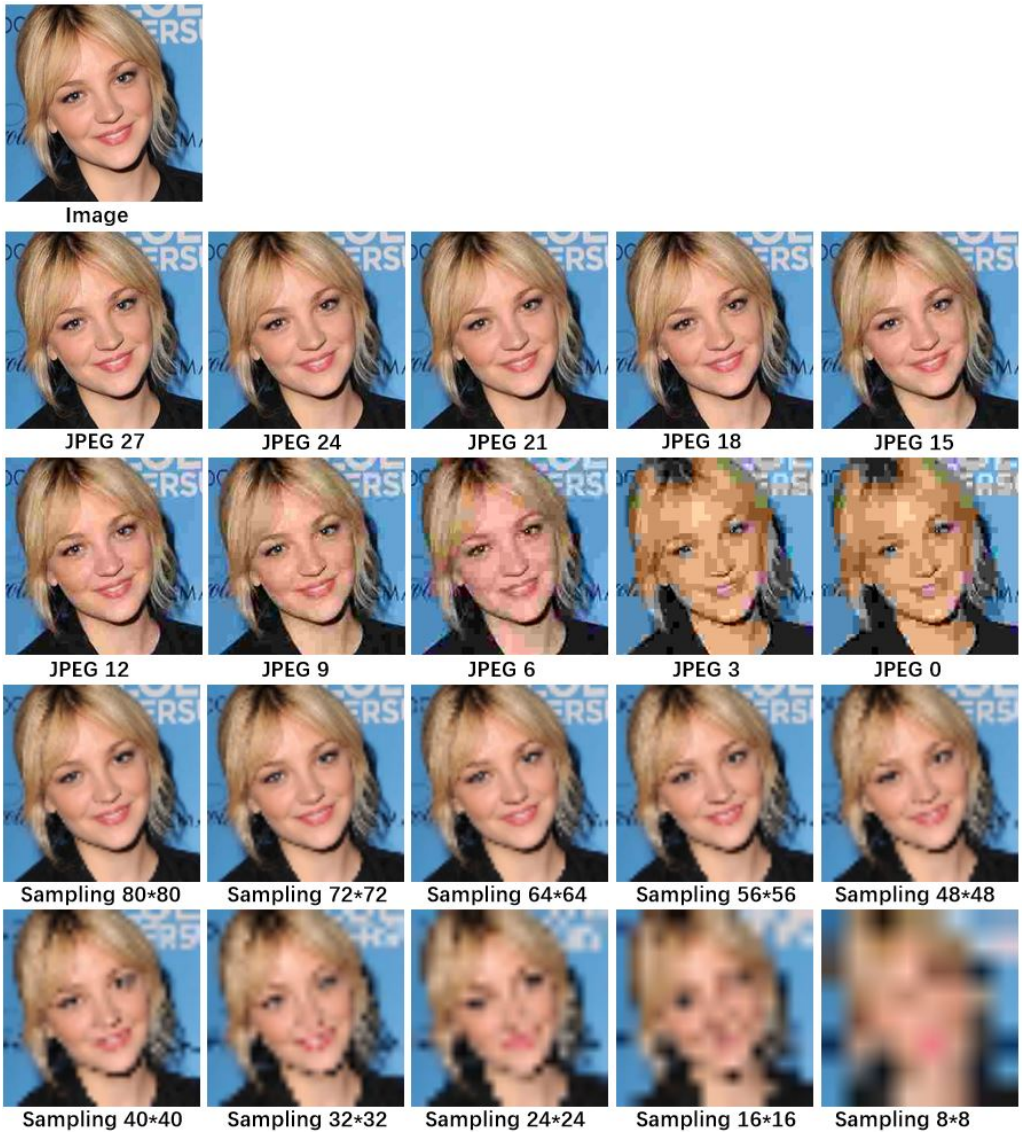


Figure 8.6: *The first row contains the JPEG compression level samples in the setting of* $\{uncompressed, 27, 24, 21, 18, 15, 12, 9, 6, 3, 0\}$. *The second row contains down-sampling level samples in the setting of* $\{unresized, 80*80, 72*72, 64*64, 56*56, 48*48, 40*40, 32*32, 24*24, 16*16, 8*8\}$.

8.4.1 Quality Prediction

Dataset

10,000 images are randomly selected from COCO [118] and MegaFace [144] separately and processed them into different quality classes with JPEG compression or down-sampling. The quality prediction network is trained on COCO images and finetuned on MegaFace images. Each image is compressed by JPEG standard with 11 quality factors = $\{27, 24, 21, 18, 15, 12, 9, 6, 3, 0\}$ and down sampled each image into 11 classes with sizes = $\{80 * 80, 72 * 72, 64 * 64, 56 * 56, 48 * 48, 40 * 40, 32 * 32, 24 * 24, 16 * 16, 8 * 8\}$, respectively.

Quality type prediction

The first quality prediction network is responsible for predicting the low-quality types, which is trained on a set with three classes, good images G, JPEG compressed low-quality images BJ and down-sampling low-quality images BL, denoted as $\{G, BJ, BL\}$. The dataset is split into training(80%) and testing(20%) sets, the testing result reached a very high accuracy of 99.9%. Note that instead of image's quality type, more emphasis are placed on the resulting probability, which indicates the relative weightings of each quality type contributing to the final results during fusion.

Quality level prediction

The exact quality severity is predicted according to the predefined quality class levels, JPEG compressed levels and down-sampling levels. As a comparison, other popular CNN architectures are also tested, including AlexNet, Inception, VGG, and ResNet. The overall results are shown in Table 8.1.

From Table 8.1, it is seen that the network built with five Convolutional layers can obtain a reasonably high accuracy while costing little computation time. Some other well-known networks are tested as well. The best one is ResNet50 with a accuracy of 91.5% for JPEG level and 98.2% for down sampling level. While for the the quality problem, a relatively simpler network is able to achieve a good result. By checking the confusion matrix in Tables 8.2 and 8.3, it is seen that all the predictions are classified into the correct classes or the quality level

Table 8.1: Quality level prediction accuracy and model testing time

Network	JPEG compression level prediction	Down sampling level prediction	Feed-forward time
AlexNet	77.2%	84.6%	58.4ms
Inception V3	87.3%	94.2%	95.9ms
VGG-16	90.6%	96.8%	112.9ms
ResNet-50	91.5%	98.2%	72.7ms
ResNet-101	Not Converge	Not Converge	177.8ms
Proposed net (Two Conv. layers)	76.4%	84.3%	58.0ms
Proposed net (Five Conv. layers)	87.8%	95.2%	18.5ms

Table 8.2: Confusion matrix of quality level estimation on JPEG compression using the proposed model (Five Conv. layers). "G" and "P" indicate the "Ground-truth" and "Prediction".

G \ P	1	2	3	4	5	6	7	8	9	10	11
1	2000	0	0	0	0	0	0	0	0	0	0
2	11	1688	194	105	0	0	0	0	0	0	0
3	0	153	1653	157	37	0	0	0	0	0	0
4	0	9	346	1580	65	0	0	0	0	0	0
5	0	9	31	102	1649	209	0	0	0	0	0
6	0	0	0	11	105	1651	233	0	0	0	0
7	0	0	0	0	8	159	1694	139	0	0	0
8	0	0	0	0	0	0	64	1910	26	0	0
9	0	0	0	0	0	0	0	35	1798	167	0
10	0	0	0	0	0	0	0	21	105	1853	21
11	0	0	8	4	0	4	0	1	0	143	1840

very close to the correct classes, which shows the CNN model has learned the quality features well and would work well in the overall framework.

8.4.2 Face Detection and Recognition with Image Quality Analysis

Evaluation protocol and datasets

Face detection and recognition are adopted as specific applications to evaluate the proposed image quality classified image detection and recognition frame-

Table 8.3: Confusion matrix of quality level estimation on down-sampling using the proposed model (Five Conv. layers). "G" and "P" indicate the "Ground-truth" and "Prediction".

G \ P	1	2	3	4	5	6	7	8	9	10	11
1	2000	0	0	0	0	0	0	0	0	0	0
2	9	1901	64	20	6	0	0	0	0	0	0
3	3	69	1891	31	6	0	0	0	0	0	0
4	0	0	64	1853	76	5	0	0	0	0	0
5	0	1	0	41	1899	58	1	0	0	0	0
6	0	0	0	5	61	1907	21	6	0	0	0
7	0	0	0	0	0	62	1913	25	0	0	0
8	0	0	0	0	0	26	54	1867	53	0	0
9	0	0	0	0	0	0	0	59	1895	46	0
10	0	0	0	0	0	0	0	0	61	1897	42
11	0	0	0	0	0	0	0	0	37	42	1921

work. Different methods are tested on each low-quality dataset to compare their performance and then test my framework on a mix-quality dataset, which consists of images that are randomly decreased into one of the quality classes.

AFLW [96] is used as the face detection dataset, which contains 25,993 faces in 21,997 images. The face recognition is tested on the CASIA-Webface [241] dataset, which consists of 494,414 faces from 10,575 subjects. The datasets are separated into training and testing sets with a ratio of 0.8:0.2. In the first experiment, all images are processed into all the quality classes I previously defined. In the second experiment, the **mix-quality** set is prepared by randomly decreasing each image into one of the quality classes. It is then separated into a mix-quality training set and a mix-quality testing set.

Performance on low quality sets

In this experiment, how image quality affects face detection and recognition is examined.

In the face detection application, three settings are adopted. **Setting 1:** train on high quality, test on low-quality level images. The **baseline** is defined as Faster RCNN [200] implementation for face detection, two other popular face detection tools are applied as well, MTCNN [251] and TinyFace [70]. For all the three methods, I train these face detectors on the unprocessed original data

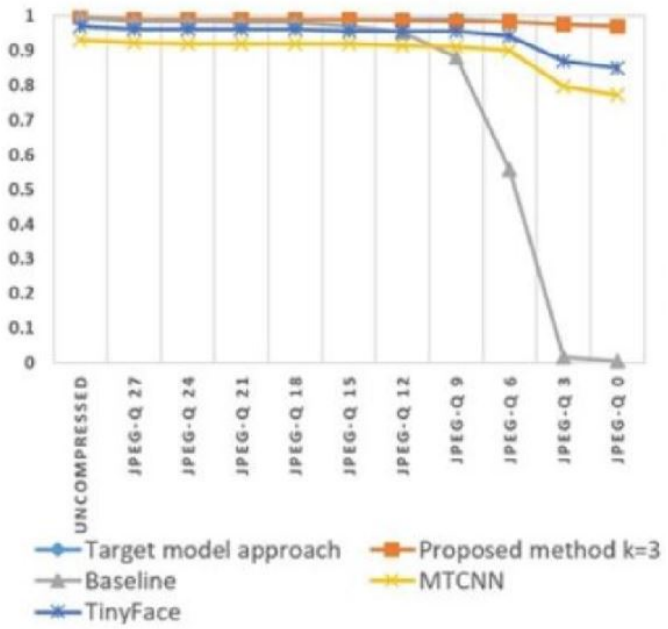


Figure 8.7: *Performance reduction on low-quality images from face detection results with JPEG compression distortion.*

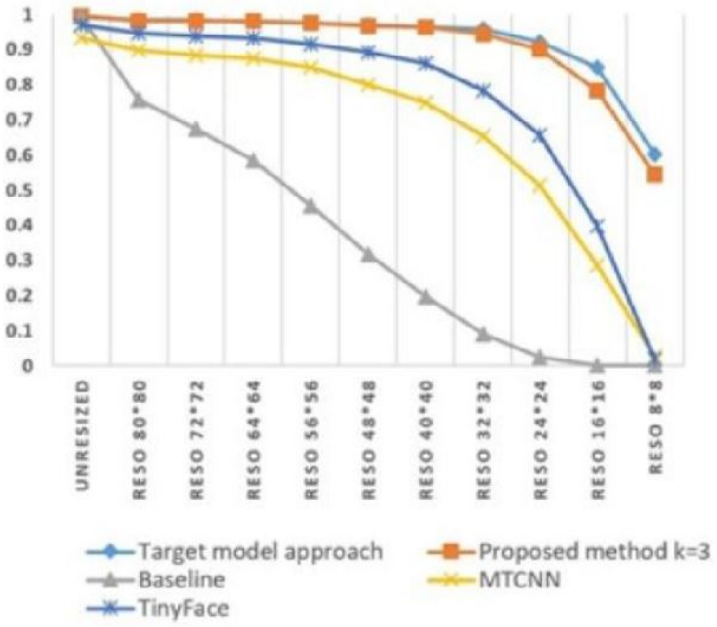


Figure 8.8: *Performance reduction on low-quality images from face detection results with down-sampling images.*

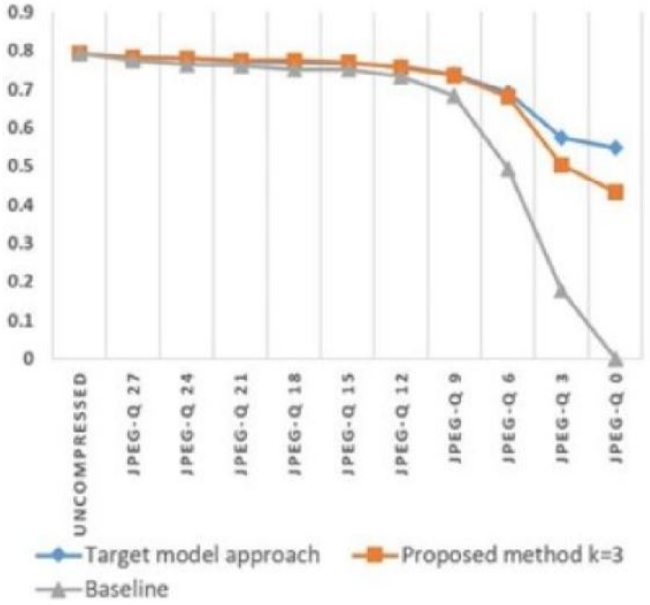


Figure 8.9: *Performance reduction on low-quality images from face recognition results with JPEG compression distortion.*

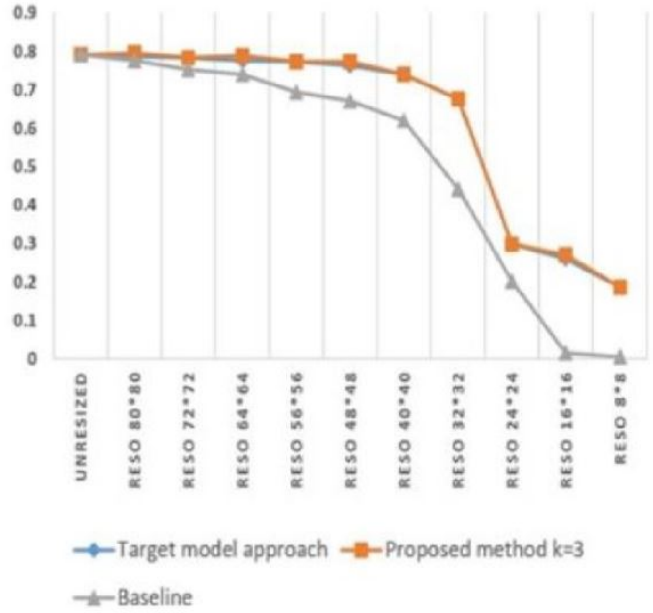


Figure 8.10: *Performance reduction on low-quality images from face recognition results with down-sampling images.*

and test them on each level of the processed data, either JPEG compression or down-sampling. **Setting 2:** train and test on the same quality class dataset. I define **target models approach** as the Faster RCNN approach that train and test on each quality class dataset separately, i.e., train 11 target detectors and test them on the corresponding quality class testing data. **Setting 3 (proposed framework):** predict the quality type and severity class, fuse detection results coming from corresponding detectors. The proposed framework is tested on each of the 11 classes testing data. Again, the Faster RCNN approach is chosen to train the 11 models separately. After quality prediction, $K = 3$ models are chosen for fusion. The results are denoted as **proposed method** and plotted in the Fig.8.7 and 8.8.

From the results, it is seen that the performance of all three methods, baseline, MTCNN and TinyFace in setting 1, drop dramatically when the corresponding JPEG compression and down-sampling levels reach to a certain level. In setting 2, if the quality class is the given prior knowledge, i.e., the method can select the correct model to analyse the image, the results can be improved significantly. Again, it proved that it is not the image quality itself that is the most important, but rather the quality of the images used to train the analyser should be similar to that of those on which the analyser will be tested. In the last setting, even without the quality information, the proposed framework can estimate the quality well and fuse the right target models to achieve a promising result.

For the face recognition application, VGG_Face [153] is chosen as the baseline evaluation method. The same three settings are followed as previously, and denoted as baseline, target model approach and proposed method, respectively. As shown in Fig. 8.9 and 8.10, the face recognition application obtain similar results. Fig 8.10 shows the proposed method has a quite close result with the target model approach.

Results on the mix-quality dataset

It is simulated how the face detection/recognition techniques perform in a real-world scenario by applying different training settings and test on the mix-quality dataset. These training settings involve the standard dataset (high-quality images) training, the mix-quality dataset training, the target model training, and my proposed framework fusing results coming from K predicted target models.

Table 8.4: Face detection accuracy on mix-quality dataset. My method * denotes Faster RCNN is applied within the proposed framework.

Training setting	Face detection methods	Accuracy(mAP)
Standard (quality unknown)	MTCNN	0.7541
	TinyFace	0.7310
	Faster RCNN	0.7292
Mixed-quality (quality unknown)	Faster RCNN	0.9095
Target model (qulity known)	Faster RCNN	0.9557
Target model (quality predicted)	My method* (K=1)	0.9216
	My method* (K=3)	0.9512
	My method* (K=5)	0.9602

Table 8.5: Face recognition accuracy on mix-quality dataset. My method# denotes VGG_face is applied within the proposed framework.

Training setting	Face recognition methods	Accuracy
Standard (quality unknown)	VGG_face	61.4%
Mixed-quality (quality unknown)	VGG_face	63.43%
Target model (quality known)	VGG_face	65.65%
Target model (quality predicted)	My method# (K=1)	62.63%
	My method# (K=3)	65.02%
	My method# (K=5)	65.81%

As shown in Table 8.4, the models are trained in different settings and tested on the mixed-quality testing dataset. Faster RCNN is applied as the baseline face detection method in my proposed image quality classified framework, which predicts the quality of the image and fuse the results coming from target models trained on each class level separately. As a comparison, I also present results of MTCNN, TinyFace and Faster RCNN in different settings. In the real-world scenario, the result shows, if the models are trained on a good quality dataset, bad results are got in all three state-of-the-art face detection methods. If the training dataset is switched with a mixed-quality one, the model improves the accuracy from around 0.73 to 0.9. Ideally, if the quality of each testing image is known precisely and selecting target model training on the corresponding quality level data, it can increase the result to 0.955. However, the proposed method could help to predict the quality classes as well as their severity, together with model fusion, it further improves the result to 0.96 if top 5 models are chosen to fuse. Table 8.5 shows the proposed framework achieves similar performance improvements in face recognition.

Hence, through extensive experimental results, it shows that image quality has a great impact on the face detection/recognition applications. Through carefully designed quality prediction network, it could recognize poor quality image caused by either JPEG compression or low resolution with high accuracy and efficiency. Using the proposed model fusion framework, it can significantly boost the accuracy of face detection/recognition in the real-world scenario.

8.5 Summary

In this work, the image quality is estimated before image-based object recognition and detection. An image quality classified image analysis framework is presented to reduce the effect of image quality factor on the performances of object detection and recognition systems. It is shown that deep learning neural networks can recognize the type and severity image quality degradations.

The experimental results on face detection and recognition show that the visual processing system's performances can be improved effectively with the image quality analysis. I have a publication about this work, "Quality classified image analysis with application to face detection and recognition" at Interna-

tional Conference on Pattern Recognition in 2018.

Chapter 9

IQA: Task Oriented Image Quality Assessment for Synthesized Images

9.1 Introduction

Conventional image quality assessment for synthesized images is to compare the similarity of synthesized image and its corresponding ground-truth. Though it is workable for translated images, the similarity comparison method does not fit image translation perfectly, since the assessment of image translation should concern the inputting images to assess how much the translation process has been completed. In this report, an image quality assessment method is proposed specially for image translation with the consideration of inputting, outputting and targeting images.

Assessing the image quality with manually labelling is inappropriate as an formal evaluation method, but it is labour-consuming and individually biased. The testing set usually contains ground-truth images, which are considered as the reference images to estimate the quality of translated images. The Mean Squared Error (MSE) of the distorted and the referencing images is commonly used to compare the image difference. However, the pixel errors of two images cannot exactly represent their differences in terms of human visual sense ([270]). Furthermore, although two blurry images may have a large MSE because of some local differences, their visual quality might be similar. Consequently, researchers have been trying to design better comparison metrics to fit the human sense of visibility.

Traditionally, IQA is classified into three types depending on the availability of the referencing image, full reference IQA (FR-IQA), reduced reference IQA (RR-IQA), and the non-reference IQA (NR-IQA) or blind IQA (BIQA). FR-IQA has clear images as the reference. Meanwhile, RR-IQA uses part of the information of the referencing image, usually some extracted features. Comparing with FR-IQA and RR-IQA, NR-IQA does away with referencing images completely, instead it outputs the quality scores by only observing the distorted images. In this work, the IQA for the translated images is FR-IQA because the ground-truth images (referencing images) are provided in the testing sets. This work develops better FR-IQA methods for image translation, considering its particularity of involving the source inputting images.

9.1.1 Preliminary Knowledge on IQA Metrics

Non-deep-learning-based Metrics

FR-IQA is to compute the difference of the generated image (the distorted image) and the corresponding ground-truth (the referencing image). As introduced in Chapter 2, the absolute error (L_1 norm), the Mean Squared Error (MSE or L_2 norm), the Peak Signal-to-Noise Ratio (PSNR) and Structure SIMilarity Index (SSIM) ([270]) are commonly used in existing image comparison works ([17, 104, 235]). The formulations are shown in Eq. (9.1), (9.2), (9.3) and (9.4).

$$L_1(m,n) = E_{m,n} \left\| m - n \right\|_1 \tag{9.1}$$

$$MSE = E_{m,n} \left\| m - n \right\|_2 \tag{9.2}$$

$$PSNR(m,n) = 20 * \log_{10}(\frac{MAX}{\sqrt{MSE}}) \tag{9.3}$$

$$SSIM(m,n) = \frac{(2\mu_m\mu_n + c_1)(\sigma_{mn} + c_2)}{(\mu_m^2 + \mu_n^2 + c_1)(\sigma_m^2 + \sigma_n^2 + c_2)} \tag{9.4}$$

where m and n represent two images; $E_{m,n}$ indicates the mean on all pixels; MAX is the maximum value of the image; μ_m and μ_n are the means of the two images; σ_m and σ_n are the standard deviations; σ_{mn} is the covariance; and, c_1

Figure 9.1: *Two pairs of examples from the results in the image rendering experiments. The upper pair is with acceptable rendering result but very low SSIM score. The lower pair is rendered badly but with a quite high SSIM score.*

and c_2 are constants to avoid the instability of the equation.

Among the four metrics, SSIM is the most widely used because its evaluation performance is much closer to human visibility than the other three ([270]). For the last decade, better perceptual-motivated metrics have been introduced, such as MS_SSIM ([222]), IW_SSIM ([221]), FSIM ([252]), and HDR-VDP ([162]). Although these metrics have improved the performance in some aspects, their aims are the same (i.e., to extract better visibility-based structural information for image comparison).

Deep-learning-based Metrics

Motivated by the great success of deep learning in vision works, researchers have started to use neural networks to solve the IQA problem. For FR-IQA, DeepIQA [13] and LPIPS [254] are network-based approaches. DeepIQA built an end-to-end cascaded network to predict the quality score with the distorted-reference image pair as the input. Like the other learning-based models [119, 125], it needs a well-labelled dataset, which contains the manually labelled score for each distorted image. LPIPS explored the effectiveness of network-based features for IQA, which are extracted from networks that are pre-trained on the common object datasets, such as the ImageNet object recognition dataset ([178]). These authors proved that this network-based feature can effectively represent the image quality. LPIPS does not need the quality-labelled datasets, but it does need a quite large recognition dataset. The final score of LPIPS is obtained by calculating the L_2 error distance of the feature vectors.

9.1.2 Problem Analysis of Existing Metrics

The pixel-based metrics, L_1, MSE and PSNR, compare two images by computing the intensity difference on pixel values, which are mathematically convenient and have clear physical meanings. However, the error (or difference) on pixels is not equated with the loss of visibility quality because some obvious low-quality distortions have little affect on the average pixel difference and some individual noise points cause a great difference on pixels but with acceptable visual quality. Previous research has found that the correlation between image fidelity and image quality is only moderate ([190, 224, 270]). Therefore, [270] proposed SSIM to compare the image similarity on structural information, which is closer to human visibility.

However, SSIM does not perform well in all situations. For the translation task, SSIM focuses on the general structure similarity of the synthesized $B^{'}$ and the targeted B, without consideration of the application. The SSIM score (Eq. (9.4)) has no information of the source image A and it lacks the ability to express the translation extent, such as the light effects added for an image rendering task. For a de-hazing work, the fog or the haze is the vital element. Furthermore, SSIM is a fixed equation, according to which different kinds of samples are calculated

Figure 9.2: *The detail comparison of the synthesized images and the ground-truth.*

by the same equation without any emphasis on the image characteristics. Fig. 9.1 shows two examples, of which Example A has an acceptable visual quality but a low SSIM score, and in contrast Example B is worse visually but it has a higher SSIM score. SSIM does not represent the visual quality precisely for the two samples. From Eq. (9.4), we know that the pixel correlation of two images, σ_{mn} as Eq. (9.5), occupies an important part of SSIM. The differences on local regions cause a low covariance of the two images, which results in a low value of the SSIM score. However, the visual sense of the synthesized image of Example A is not so bad because of its visually reasonable light effect, which presents a good stereoscopic feeling of the room. For the rendering work, the light effect is the most essential factor that affects the synthesizing quality. Consequently, better evaluation methods are needed to concentrate on the light effect for the rendering work

Fig. 9.1 shows two examples, of which example A is with acceptable visual quality but a low SSIM score while example B is worse visually but with a higher SSIM score. From Eq. (9.4) I know that the pixel correlation of two images σ_{mn} plays an important part of the SSIM score, which is sensitive to local changes. Comparing with that of example B, the synthesis of A contains greater differences on local image regions with the ground-truth, shown in Fig. 9.2. From Eq. 9.4, the differences on local regions cause a low covariance of the two images, σ_{mn} (Eq. (9.5)), which results in the low value of SSIM score. The visual feeling

of the synthesized image of example A is not so bad, due to the reasonable light effect visually, which presents us a good stereoscopic feeling of the room. For example B, the inadequate rendering effect causes a bad stereoscopic feeling on the synthesis.

$$\sigma_{mn} = E[(m - \mu_m)(n - \mu_n)] \tag{9.5}$$

The neural networks-based FR-IQA models show better performance than pixel-based metrics. However, it still only considers the target image as the referencing image, which does not fit the application of image translation. There are three kinds of images within a translation task, the inputting image (source domain A), the translated image, and the ground-truth/targeting image (target domain B image). Besides the ground-truth, the inputting image is another essential reference to measure how well the translation is achieved. A two-referencing IQA method is proposed that takes both the inputting and the targeting images as the reference to estimate the quality of the generated images.

9.1.3 Contributions

The contributions are summarised as follows. 1) An image quality assessment method is proposed specially for translated images, which fits image translation better than conventional image similarity comparison methods. 2) A task-oriented feature extraction network is built to extract effective features from images. 3) A quality score formula is proposed and extensive experiments show that the proposed method outperforms other famous image similarity comparison metrics.

9.2 Task-oriented Image Quality Assessment

9.2.1 Method Overview

In this work, a task-oriented approach is proposed to evaluate the image quality of translated images, following the principle that to assess the similarity of two images is to calculate their feature distance.

A common CNN-based feature is obtained by extracting the layer activa-

tions of a pre-trained network, such as the second last layer of VGG ([192]) pre-trained on ImageNet ([177]) classification. This dataset contains a large range of categories with plenty of samples for each class. A well-trained network is able to learn effective feature if it can classify a variety of objects. Although [254] showed that this kind of feature is quite effective for IQA, it does have a number of drawbacks. This kind of feature lacks the information of certain translation tasks, which is likely to ignore special information, such as haze-like and rain-like noise, or the light reflection effect. Unusual distortions have a significant affect on visual quality but they cannot be well caught by the feature extraction networks.

In this work, better features are explored to represent the information for both image content and certain translation task. The image content is a fundamental factor in measuring visual quality. However, the translation task determines which kind of information affects the model's performance (e.g. the haze in a de-hazing work, the light effects in a rendering work, etc.). A two-task pre-training method is proposed to train the network so that it obtains the ability to extract the expected feature. The network is optimised by two loss functions iteratively with a well-prepared task-oriented dataset.

However, to only have effective feature representations is not enough. Therefore, the score formulation is designed by involving the inputting image, outputting image (distorted image) and the targeting image (referencing image) in the quality estimation. The inputting and targeting images are both regarded as the references of FR-IQA, which is different from previous works that only take the targeting as the referencing image.

9.2.2 Feature Network

The training of the feature network focuses on two aspects, the image content and the certain translation task. Therefore, a translation-classification joint-training framework is designed, which contains two optimization branches: the translation branch and the classification branch. The translation branch decodes the learned feature into an input-like image in an encode-decode manner, while the classification branch classifies the inputting image into one level category. Here, each level category indicates that one extent level the translation is achieved because the training set is prepared with various translation extents of generated

images for different extent levels. The two optimization branches share the same encoding base net, whose last layer is used for feature extraction.

The dataset must be well prepared to achieve the goal of two-task training. Taking the image rendering task as an example, the rendering process adds light effect on the plain image, while preserving the image content unchanged. A dataset with different rendering levels between the plain image domain and the rendered image domain is needed. Suppose that a rendering level i contains semi-rendered images R_i, which is linearly related to the plain image I and the rendered image R, as shown in Eq. (9.6).

$$f(R_i) = \lambda_1 f(I) + \lambda_2 f(R) \tag{9.6}$$

where $f(\cdot)$ represents the rendering extent assessment on images, and λ_1 and λ_2 are balancing weights to define the rendering extent. Because the shapes and the lines are mostly unchanged between the plain and the rendering images, $f(\cdot)$ is regarded as a linear transform on pixels, shown in Eq. (9.7).

$$f(R_i) = f(\lambda_1 I + \lambda_2 R) \tag{9.7}$$

Indeed, λ_1 and λ_2 have a relationship of $\lambda_1 + \lambda_2 = 1$, if R_i represent i_th level of intermediate rendering images. The linear $f(\cdot)$ is removed and get Eq. (9.8):

$$R_i = (1 - \alpha_i)I + \alpha_i R \tag{9.8}$$

where α_i is the balancing weight to replace λ_1 and λ_2, ranging from 0 to 1. By setting different α_i, images in different categories are obtained with each one representing a rendering level i. The prepared dataset is used to train N as the translation-classification task, where the ground-truth of the translation is the same as the input image and the classification labels are set along with the rendering levels.

Softmax with cross entropy loss is applied for the classification branch to classify the level category and the L_2 loss is implemented on the decoding branch to minimize the error of the decoding image and the inputting image. The two kinds of loss are used to optimize the network in an iterative way with error back propagation.

The well-trained N is able to capture the features of the image content and the translation extent. The second last layer of the classification branch and the last layer of the encoder are concatenated to extract the image representation when forwarding an testing image through it.

The network structure is shown in Fig. 9.3. The inputting image is processed by an eight-convolution-based encoder and encoded into a 1×1-sized feature map. Each convolutional layer downsizes the feature maps into half with a kernel stride of two. It is then connected into two branches: the decoding branch with eight two-stride deconvolution layers and the classification branch with three dense and connected layers. Note that the skip connection ([78]) that is commonly used in the encoder-decoder structure is not applied here because the skip connections would build multiple "routes" from the inputs to the outputs and the unique encoding feature vector is obtained as the image representation. The activation function, ReLU, is set on every layer except the last layer of each branch and the batch normalization is applied on each convolutional or deconvolutional layer except the last layer.

After training, N is used for the feature extraction by obtaining the activations of the last layer of the encoder, denoted as $feat1$, and the second last layer of the classification branch, denoted as $feat2$. The concatenated features are flattened into a vector as the final feature f_{image} with a weighted factor m, as shown as Eq. (9.9):

$$f_{image} = Concat(m * \frac{feat1}{\|feat1\|_2}, (2-m) * \frac{feat2}{\|feat2\|_2}) \tag{9.9}$$

where the weighted factor m is used to balance the importance of the two features, which will be examined in detail in the experiments. The weight m is necessary because the two features are in different value levels and they have different importance in the final quality estimation. The f_{image} becomes the direct concatenation of $feat1$ and $feat2$ when $m = 1$. In this report, it is set as $m = 1$.

Based on the motivation that both the source domain image and the targeting domain image are considered as the references of IQA, the trained N is used to extract the features for the synthesized $B^{'}$ (distorted image), the ground-truth B (referencing image) and the source A.

9.2.3 Quality Score Formulation

According to Eq. (9.9), the feature is extracted from the pre-trained network with a certain m. By forwarding the images, the image representations of f_A, f_B and $f_{B'}$ are obtained for the calculation of the quality score t_{score}. Obviously, the t_{score} relies on the negative correlation of the feature distances of f_B and $f_{B'}$, such as the Euclidean distance. The cosine distance was also used for computing the feature distance in the quality assessment research ([255]).

In this work, the quality score t_{score} is calculated based on the Euclidean distance because of its convenience when used to compare distances between multiple vectors. The feature distance of the distorted image $f_{B'}$ and the referencing image f_B are compared, which is called the "content score". For the consideration of the certain image translation task, the model performance is expressed by how well the generated image is synthesized following the mapping of f_A to f_B, which can be measured by the comparison of $f_A \rightarrow f_{B'}$ and $f_{B'} \rightarrow f_B$. Thus, the calculation of t_{score} is completed by adding the ratio of the two vectors, $f_{B'} - f_A$ and $f_{B'} - f_B$, which is called the "translation score". A balancing weight is added on the combination of the content score and the translation score, as Eq. (9.10):

$$\begin{aligned} t_{score} = \beta_1 * d_s(1, f_{B'} - f_B) \\ + (1 - \beta_1) * d_s(f_{B'} - f_A, f_{B'} - f_B) \end{aligned} \tag{9.10}$$

in which,

$$d_s(A, B) = \frac{\|A\|_2}{\|B\|_2 + \epsilon}, \epsilon = e^{-9} \tag{9.11}$$

where d_s is the score according to distance, shown in Eq. (9.11); β_1 is the balancing weight to control the importance proportion of the two scores; ϵ is a small constant to avoid the fraction infinite. When set $\{\beta_1 = 1\}$, t_{score} becomes the Euclidean distance-based FR-IQA score; and when set $\{\beta_1 = 0\}$, t_{score} depends on the translation score completely.

It is seen that the weight β_1 controls the participation ratio of the inputting image A so that the formulation can fit different tasks with different values of β_1. Some translation tasks may focus more on the image content if the targeting image has much different content with the inputting image, while some other translation tasks may have two domain images with very similar image content

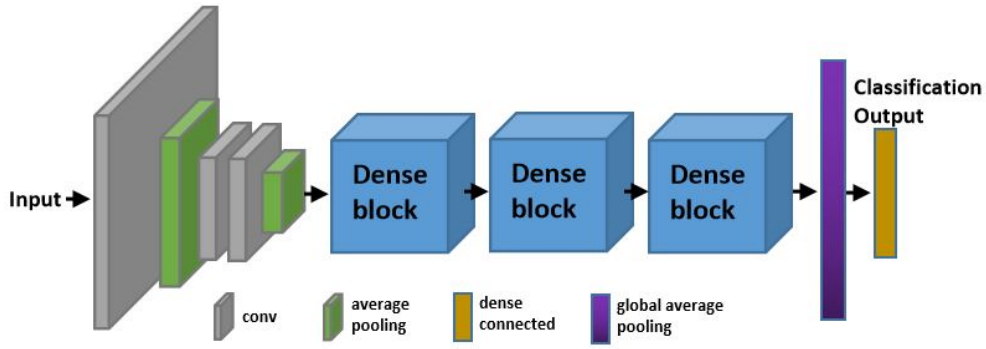


Figure 9.3: *The structure of the feature extraction network N.*

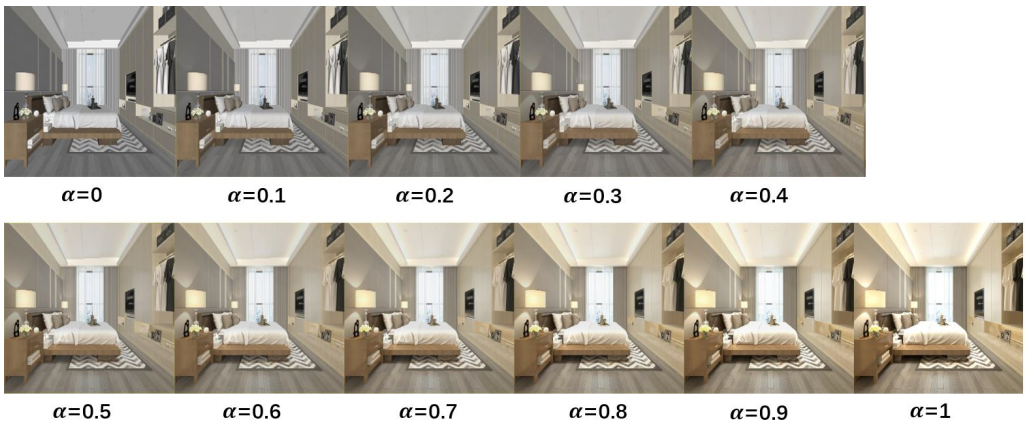


Figure 9.4: *One example of the data set prepared for the training of feature extraction net in the image rendering task.*

but only slightly difference in other aspects, such as the texture, the colour or the image noise. In this report, β_1 is set as 0.5 to have an equal contribution from the FR-IQA score and the translation score.

9.3 Experiments

9.3.1 IQA Evaluation

The proposed task-oriented IQA is applied to evaluate the testing set of each work in this report, image rendering, de-raining, de-hazing and de-snowing. The net used for extracting the task-oriented feature is pre-trained on the corresponding dataset for each work. The details of the experiments for each evaluation task are discussed and the results are presented in the following sections. The experiments are conducted in Tensorflow (python).

Figure 9.5: *Examples with high SSIM scores, but with low t_scores.*

Evaluation on Image Rendering

The network N for extracting the rendering feature is trained on images prepared according to Eq. (9.8) by setting α_i as $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. After 200,000 iterations of training, the testing accuracy is 0.7548. Table 9.1 is the confusion matrix, showing that most of the predictions are accurate with some wrong predictions classified into the close rendering levels. From the confusion matrix, it is seen that N gains an ability of learning the rendering feature from the pre-training.

The pre-trained N is used to extract rendering features for images, defined as f_A, f_B, $f_B^{'}$, which is used to calculate the IQA score t_{score} according to Eq. (9.10). The proposed method is applied to evaluate the results of image rendering and the comparisons with conventional metrics are shown in Table 9.2. Generally, t_{score} presents the same conclusion with that of PSRN and SSIM. the result of $cGAN+Caps+line$ has the highest t_{score} of 1.82, and its PSNR and SSIM are the highest too. Table 9.2 is the average evaluation results of the testing dataset. Some examples are shown in Fig. 9.5 and 9.6. Fig. 9.5 shows four examples whose SSIM scores are relatively high, but with bad visual quality. Fig. 9.6 presents another four examples whose SSIM scores are low but the visual quality is good. From Fig. 9.5 and 9.6, it is observed that t_{score} is more consistent with visual quality, especially about the rendering effect. For rendering images,

Figure 9.6: *Examples with low SSIM scores, but with high t_scores.*

the light effect plays an important role in visual quality. The Images from Fig. 9.6 have acceptable light effect, which present good visual quality. But the local details are a little blurry with some unclear boundaries, which leads to low SSIM scores. The proposed t_score concerns more about the overall light effect than local details and produces more consistent scores.

Evaluation on De-raining

The dataset proposed by Zhang et al. [248] (Chapter 5), contains the subsets of $\{Heavy, Medium, Slight\}$. It is used to train N without data augmentation. After 10,000 iterations of training, the classification accuracy is nearly 100%, indicating that N has learned feature effectively from rainy images. The feature vectors ,f_A, f_B, $f_B^{'}$, represent rainy image, clear image, synthesized de-raining image. They are used to compute the t_{score} according to Eq. (9.10).

The results are in Table 9.3. Comparing with other de-raining models, the proposed RCA-cGAN obtains the highest t_{score}. This is consistent with PSNR and SSIM, which means the proposed method is workable to evaluate the de-raining results.

Table 9.1: Confusion matrix of the pre-training net for image rendering with each class for the corresponding α_i. "G" and "P" indicate the "Ground-truth" and "Prediction".

G \ P	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	56	0	0	0	0	0	0	0	0	0	0
0.1	1	55	0	0	0	0	0	0	0	0	0
0.2	0	20	36	0	0	0	0	0	0	0	0
0.3	0	0	12	44	0	0	0	0	0	0	0
0.4	0	0	1	10	40	5	0	0	0	0	0
0.5	0	0	0	1	12	41	2	0	0	0	0
0.6	0	0	0	0	0	5	38	8	4	1	0
0.7	0	0	0	0	0	0	6	41	9	0	0
0.8	0	0	0	0	0	0	2	7	37	10	0
0.9	0	0	0	0	0	0	1	2	15	38	0
1	0	0	0	0	0	0	0	1	2	14	39

Evaluation on De-hazing

The dataset used to train N is prepared by weighted combination of clear and hazy images according to Eq. (9.12).

$$H_i = (1 - \alpha_i)O + \alpha_i H \tag{9.12}$$

where O and B represent clear and hazy images; α_i is set as

$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

An 11-category dataset is made by defining 11 α values. $\alpha = 0$ indicates the clear images without any haze and $\alpha = 1$ represents the hazy images. N is trained to identify the haze levels. The feature vectors for hazy images (f_A), clear images (f_B) and synthesized image ($f_B^{'}$) are used to compute the IQA T_{score} according to Eq. (9.10).

t_{score} is calculated on the testing sets, SOTS-indoor and SOTS-outdoor described in Chapter 6, shown in Tables 9.4 and 9.5. The proposed DDN produces the highest t_{score} on both datasets. It is seen that DDN outperforms others with a large margin on the observation of t_{score} results, which are consistent with PSNR and SSIM.

Table 9.2: Quantitative evaluation t_{score} on rendering images comparing with PSNR and SSIM.

Models	t_{score}	PSNR	SSIM
AE[229]	1.02	13.74	0.5889
VAE[158]	1.10	13.23	0.5907
U-net[175]	1.64	14.17	0.6537
cGAN[78]	1.67	14.71	0.6271
cGAN+Caps(my method)	1.70	<u>15.85</u>	<u>0.7184</u>
cGAN+Caps^2(my method)	1.74	15.08	0.6440
cGAN+Caps+line(my method)	**1.82**	**16.91**	**0.7356**
cGAN+Caps^2+line(my method)	<u>1.75</u>	15.77	0.6542

Table 9.3: Quantitative evaluation t_{score} on de-raining images from Chapter 5 comparing with PSNR and SSIM.

Dataset	Test1[248]			Test2[248]		
Models	t_{score}	PSNR	SSIM	t_{score}	PSNR	SSIM
DetailsNet[45]	2.95	27.33	0.8978	<u>3.18</u>	25.63	0.8851
PAN [214]	3.11	27.43	0.8637	2.18	24.88	0.8093
RESCAN[113]	3.41	28.32	0.8638	2.16	24.70	0.8126
DID-MDN[248]	<u>3.96</u>	28.28	0.9218	2.57	<u>26.36</u>	<u>0.8886</u>
ID-cGAN[249]	3.02	25.86	0.8657	1.97	23.58	0.7997
PReNet[170]	3.89	<u>30.31</u>	<u>0.9360</u>	2.44	24.34	0.8617
RCA-cGAN	**4.51**	**32.03**	**0.9468**	**3.47**	**27.11**	**0.8984**

Evaluation on De-snowing

The dataset proposed in Chapter 7 contains three subsets of {Heavy, Medium, Slight}, which is used to train N as a 4-category classification problem with $clear$ images as the fourth category. After training, the classification accuracy reached almost 99%.

T_{score} are calculated and shown in Table 9.6. The testing set is divided into "Snow-100K-L", "Snow-100K-M" and "Snow-100K-S" to represent {Heavy, Medium, Slight}. In overall, the proposed MC-DS produces the best results on all evaluation metrics, including t_{score}. On the subsets of "Snow-100K-M" and "Snow-100K-S", MC-DS gives the second best SSIM score, and t_{score} shows that MC-DS is better than other models on these two subsets. The snow on images from "Snow-100K-M" and "Snow-100K-S" affects little about the im-

Table 9.4: Comparison results t_{score} with PSNR and SSIM on SOTS-indoor from Chapter 6.

Model	t_{score}	PSNR	SSIM
DCP [59]	1.98	16.62	0.8179
DehazeNet [17]	2.01	21.14	0.8472
AOD-NET [107]	2.21	19.06	0.8504
GFN [173]	2.47	22.30	0.8800
DCPDN [247]	3.95	15.85	0.8175
EPDN [160]	4.85	25.06	0.9232
DDRL [53]	4.21	24.35	0.9017
DA_dehazing [184]	<u>5.11</u>	<u>25.78</u>	<u>0.9541</u>
DDN	**5.84**	**28.43**	**0.9733**

Table 9.5: Comparison results t_{score} with PSNR and SSIM on SOTS-outdoor from Chapter 6.

Model	t_{score}	PSNR	SSIM
DCP [59]	1.94	19.13	0.8148
DehazeNet [17]	2.21	22.46	0.8514
AOD-NET [107]	2.69	20.29	0.8765
GFN [173]	2.98	21.55	0.8444
DCPDN [247]	3.11	19.93	0.8449
EPDN [160]	3.24	22.57	0.8630
DDRL [53]	3.45	22.21	0.8589
DA_dehazing [184]	<u>4.53</u>	<u>27.23</u>	<u>0.9392</u>
DDN	**6.54**	**31.29**	**0.9809**

age content so that the SSIM is insensitive to the differences of snowy images (the input) and clear images (ground-truth). t_{score} of MC-DS is better than other models obviously.

9.3.2 Comparison with State-of-the-art

To evaluate the proposed IQA method, an evaluation dataset is needed. In this report, the evaluation dataset is prepared by labelling quality scores for two testing sets, the synthesized images from the rendering work ([235]) and the de-hazing set from the haze removal work ([108]). The inputting, outputting, targeting images are shown group by group to ten non-expert interviewers, who are asked to give a score for each group to express the translating quality. Each group is

present to the same viewer twice to make sure the score is labelled unbiased. The final score are obtained by averaging the scores from the ten viewers, as the mean opinion score (MOS). The images and the labelled scores will be made publicly available.

Pearson linear correlation coefficient (PLCC) and Spearman rank order coefficient (SROCC) are used to estimate the correlation of predicted scores and the labelled ground-truth. The two coefficients are commonly used to evaluate the performance of IQA methods in literature ([119, 125, 133, 234, 254]). The score is in the range of 0 to 1 with a higher number indicating a better IQA method. For comparison, the results are compared with famous metrics, PSNR, SSIM ([270]), MS_SSIM ([222]), IW_SSIM ([221]), FSIM ([252]), DeepIQA ([13]) and LPIPS ([254]).

The PLCC and SROCC results on rendering dataset are shown in Table 9.7. Among the conventional metrics, SSIM performs relatively worse and IW_SSIM gives better results with PLCC of 0.4579 and SROCC of 0.4567. The deep learning-based metrics, DeepIQA and LPIPS, produced much different results. DeepIQA performs quite badly, because its training datasets have no images on light effects or related rendering tasks. LPIPS produces quite good results, as it is trained on large common-object datasets and contains the ability of learning the feature of usual objects, whose visibility is much affected by the light reflection and the rendering effect. Our method produces the better results than LPIPS, as we designed the decoding branch that is trained on the rendering dataset in the encoder-decoder manner, which supervises the model to learn the feature of the image content well. The classification branch for the rendering extent estimation also helps a lot in the feature learning to identify the light effect.

The PLCCs and SROCCs results on haze dataset are shown in Table 9.8. My method achieves the best performance. PSNR gives a slightly better result than SSIM, due to the fact that haze noise has a big affect on the pixel difference. DeepIQA has a better performance on this dataset than the rendering dataset, since the haze noise is closer to the distortion types of what DeepIQA was trained on. LPIPS performs much worse than the rendering dataset, only with PLCC of 0.1655 and SROCC of 0.1541, the worst of all the metrics in Table 9.8. It is because of the non-correlation of the noise and the image content, as different hazy images contains the similar, indeed the same, image content, so that LPIPS

would learn the similar content feature even from different noisy images. The changes of the haze distribution are not represented on the feature difference that LPIPS tries to learn. A light-hazy image has the quite similar image content with the middle-hazy one.

9.4 Summary

The task-oriented IQA (Image Quality Assessment) method is proposed for translated image, which is used to evaluate the translated images in this report.

The experimental results show that the proposed method fits image translation task better than other evaluation metrics. I have a paper manuscript, “Task-oriented image quality assessment for synthesised images”.

Table 9.6: Evaluation comparison t_{score} on Snow100K from Chapter 7. The results with bold font and underline are the best and the second best.

Dataset	Models	t_{score}	PSNR	SSIM	nrmse	L1
Snow-100K-L	AE[230]	1.59	21.50	0.7528	0.1938	0.1773
	Pix2pix[78]	2.42	25.83	0.8490	0.1170	0.1534
	cycleGAN[275]	1.96	19.21	0.6575	0.3596	0.2876
	DID-MDN[248]	3.97	26.34	0.8969	0.1097	0.0897
	DesnowNet [129]	3.67	27.17	0.8983	0.1101	0.0398
	LSTM-GAN [61]	3.11	26.12	0.8616	0.1214	0.1102
	ComposGAN [116]	<u>4.02</u>	**29.54**	<u>0.9021</u>	<u>0.1001</u>	**0.0301**
	MC-DS	**4.16**	<u>28.14</u>	**0.9032**	**0.0995**	<u>0.0304</u>
Snow-100K-M	AE[230]	2.14	23.38	0.8422	0.1865	0.1298
	Pix2pix[78]	4.25	29.29	0.9261	0.0798	0.0243
	cycleGAN[275]	1.83	20.30	0.7377	0.2215	0.2654
	DID-MDN[248]	<u>4.95</u>	28.79	0.9423	0.0841	0.0285
	DesnowNet [129]	4.19	30.87	0.9409	0.0775	<u>0.0210</u>
	LSTM-GAN [61]	4.21	30.84	0.9384	0.0721	0.0212
	ComposGAN [116]	<u>4.95</u>	<u>31.21</u>	<u>0.9431</u>	<u>0.0622</u>	0.0218
	MC-DS	**5.21**	**31.59**	**0.9444**	**0.0619**	**0.0208**
Snow-100K-S	AE[230]	2.34	23.60	0.8444	0.2015	0.1472
	Pix2pix[78]	4.66	29.90	0.9232	0.1201	0.1011
	cycleGAN[275]	1.79	21.40	0.7825	0.2012	0.2043
	DID-MDN[248]	5.33	30.15	0.9521	0.0765	0.0265
	DesnowNet [129]	<u>5.64</u>	<u>32.33</u>	0.9500	<u>0.0599</u>	0.0210
	LSTM-GAN [61]	4.54	30.09	0.9411	0.0914	0.0255
	ComposGAN [116]	5.11	30.43	<u>0.9612</u>	0.0641	<u>0.0207</u>
	MC-DS	**5.91**	**32.49**	**0.9674**	**0.0557**	**0.0174**
Over-all	AE[230]	2.06	22.83	0.8131	0.1939	0.1514
	Pix2pix[78]	3.77	28.34	0.8994	0.1056	0.0929
	cycleGAN[275]	1.84	20.30	0.7259	0.2607	0.2524
	DID-MDN[248]	4.69	28.42	0.9304	0.0901	0.0482
	DesnowNet [129]	4.48	30.11	0.9296	0.0825	0.0272
	LSTM-GAN [61]	3.95	29.01	0.9137	0.0949	0.0523
	ComposGAN [116]	<u>4.70</u>	<u>30.39</u>	<u>0.9355</u>	<u>0.0754</u>	<u>0.0242</u>
	MC-DS	**5.09**	**30.74**	**0.9383**	**0.0724**	**0.0229**

Table 9.7: Comparison with other IQA metrics on the image rendering set [235]

IQA metrics	PLCC	SROCC
PSNR	0.3312	0.3002
SSIM	0.3087	0.2597
IW_SSIM	0.4579	0.4567
FSIM	0.4086	0.3857
MS_SSIM	0.4304	0.4151
DeepIQA	0.1374	0.0942
LPIPS	<u>0.4984</u>	<u>0.4830</u>
t_score	**0.5621**	**0.5258**

Table 9.8: Comparison with other IQA metrics on the image de-hazing set [108]

IQA metrics	PLCC	SROCC
PSNR	0.3978	<u>0.3419</u>
SSIM	0.3698	0.2886
IW_SSIM	0.3365	0.2837
FSIM	0.3606	0.2876
MS_SSIM	<u>0.4136</u>	0.3217
DeepIQA	0.2242	0.2519
LPIPS	0.1655	0.1541
t_score	**0.5457**	**0.6249**

Chapter 10

Conclusion

This thesis discusses the research on image translation. With deep learning techniques, the capsule-based image-conditioned generative adversarial network is proposed to translate images from a domain to another. Specially, various methods are developed for certain applications, image rendering, de-raining, de-hazing and de-snowing. Image quality affects vision analysis systems greatly. A task-oriented image quality assessment method is proposed to evaluate the synthesised images. The following sections summarise contributions, publications, limitations and future work.

10.1 Contributions

The contributions are summarised as follows with a review of Chapters 3 to 9.

Chapter 3 introduces a capsule-based image-cGAN framework for image translation. Different from conventional CNN-based networks, capsule-based networks are proved to be more effective in learning the part-to-whole relationship, such as the light effect for rendering (Chapter 4). With capsule units, the generator and the discriminator gain better ability to learn image contents from both global and local views.

Chapter 4 introduces the work of image rendering. Besides the capsule implementation, a line preservation loss is designed to supervise the learning of line shapes. The rendering effect relies much on the relationship of lights and other objects, which can be well learned by capsule units. The line shapes play an important role in the visual quality of indoor images. They can be maintained

by the line preservation loss. A dataset is contributed to the research community. The proposed capsule-based image conditioned generative adversarial network and the rendering work (Chapter 3 and 4) have been published in IET Doctoral Forum on Biomedical Engineering, Healthcare, Robotics and Artificial Intelligence(BRAIN), 2018 and Asian Conference on Computer Vision with the title of "Capsule based image translation network" and "Capsule based image synthesis for interior design effect rendering". A journal version of this work is preparation.

Chapter 5 proposes a two-branch framework for de-raining. A rain component aware module is designed to better identify the rain from image contents. The overall framework is supervised with two optimizing aspects, the image content and the rain component. The de-raining work (Chapter 5) has been submitted into *Patter Recognition* under reviewing.

Chapter 6 discusses a depth aware de-hazing framework. By estimating the depth map and the haze-free image jointly at the same time, the model is able to identify haze better and recover clearer images. An effective encoding module is designed to involve the depth feature into de-hazing model. The de-hazing work (Chapter 6) has been published in *The Visual Computer* with the title of "Depth aware generative adversarial network for real haze removal".

Chapter 7 devises a multi-scale branch framework for snow removal. Each branch is to learn specific scale of snowflakes with pre-defined scaling kernels. By this, the multi-branch model is able to learn various scales of snowflakes. The capsule units are implemented as well to merge the feature from different scale branches. The de-snowing work (Chapter 7) has been published in *IET Computer Vision* with the title of "Multi-scale capsule generative adversarial network for snow removal".

Chapter 8 discusses the image quality levels caused by JPEG compression and down-sampling. The quality levels are predicted with convolutional neural networks. The quality information is used for face detection/recognition. The experiments prove that image quality affects the model performance greatly but could be estimated by CNN accurately. With quality level pre-prediction, a joint framework of multiple detection models is effective than a single one. The work of quality level prediction (Chapter 8) has been published in *International Conference on Pattern Recognition* 2018.

Chapter 9 proposes an image quality assessment method specially for image translation. With the effective feature extracted from well-trained network, an equation is formulated with concerning the input, the target and the output images. Comparing with conventional image similarity evaluation metrics, the proposed method fits better certain image translation tasks. The work of task-oriented image quality assessment (Chapter 9) is under preparation for a journal paper.

10.2 Limitations and Future Work

One limitation of the proposed capsule-based frameworks is the training stableness. In the experiments, the training dose not converge sometimes under some sets of hyper-parameters, due to the inconsistency of supervision signals from multiple loss functions. More time is needed to find the best hyper-parameters such as the loss balancing weights. Further work is needed about how to find the best hyper-parameters or how to reduce the affect from the hyper-parameter settings.

There is another limitation about the network pre-training. In this report, the proposed method often needs pre-training for the networks or modules such the RCA network in de-raining framework and the feature extraction network in task-oriented IQA. The pre-training is donw on specific datasets, which is less convenient. A trade-off solution is to apply a fixed network pre-trained on a common dataset, like the classification dataset from ImageNet competition [178], for the feature extraction. It is obvious that the network trained on common datasets lacks the ability to extract effective feature for certain tasks, such as the light reflection feature and the noise feature. How to pre-trained the network for different scenarios more conveniently is in future work.

The capsule with the fundamental vectorized structure introduced by [63] and [179] is implemented. Many new techniques of capsules have been proposed by the research community. The matrix capsule [64] and new routing algorithms [54, 64] have been developed in recent years. Further research is to be done on the combination of image translation framework and new capsule techniques, such as matrix capsule [64], spread loss [64], EM Routing [64], convolution formed capsule [101], graph capsule [213], self-routing capsule [54],

self-attention capsule [66].

In the image rendering work from Chapter 4, the images are rendered blindly without extra information, such as the room style, the room type, the light types, the light illumination and etc. The rendering model should be trained well if targeting one specific condition, a living room in Chinese style with 4 lights for example. The rendering model might also be trained well if these extra information is encoded into the model. To achieve this, a large amount of training data is needed. The labour resource is too limited to accomplish the huge of work. The future work could be set as image rendering with manual conditions if a large of rendering dataset is available. It can be achieved by building the image translator with some pre-defined inputting nodes to encode the manual condition settings.

In de-hazing work, a joint framework is built with haze removal and depth estimation at the same time. It is expected to remove haze by being aware of the depth feature. Another idea should be reasonable if the depth estimation is purpose with being aware of haze. The correlation of haze and depth inspires me to develop a jointly training framework in which one task contributes to the other. In fact, a variety of computer vision tasks have correlations with each other, such as segmentation and depth, segmentation and edge detection [127], colourization [76, 253] and object detection, style transfer [82] and scene understanding [38], motion estimation [269] and human detection [147], face detection [71, 201, 238] and human detection, face recognition [32, 123] and age estimation [22, 120], and so forth. Indeed, it is believed that all those tasks have correlations, because these tasks are different representations of the same image understanding. Currently, it may be impossible to build a complicated system that involves too many tasks, though a well-designed joint system is expected to have good performance with multiple tasks. But the research of correlation between multiple computer vision tasks is valuable so that a better joint system could be built by finding groups of tasks that have much correlations with each other.

Bibliography

[1] Adobe. Adobe photoshop. https://www.adobe.com/products/photoshop.html.

[2] Hiroaki Aizawa, Hirokatsu Kataoka, Yutaka Satoh, and Kunihito Kato. Agnostic image rendering. *Proceedings of Winter Conference on Applications of Computer Vision*, pages 3803–3812, 2021.

[3] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv e-prints*, abs/1812.11941, 2018.

[4] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[5] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[6] Autodesk. 3d max. https://www.autodesk.com/products/3ds-max/overview.

[7] Peter C. Barnum, Srinivasa Narasimhan, and Takeo Kanade. Analysis of rain and snow in frequency space. *International Journal of Computer Vision*, 86(2-3):256, 2010.

[8] Cher Bass, Tianhong Dai, Benjamin Billot, Kai Arulkumaran, Antonia Creswell, Claudia Clopath, Vincenzo De Paola, and Anil Anthony Bharath. Image synthesis with a convolutional capsule generative adversarial network. *Medical Imaging with Deep Learning*, 2019.

[9] Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *corr abs/1502.04390*, 2015.

[10] Dana Berman, Tali Treibitz, and Shai Avidan. Non-local image dehazing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[11] Dana Berman, Tali Treibitz, and Shai Avidan. Air-light estimation using haze-lines. *Proceedings of the IEEE International Conference on Computational Photography*, 2017.

[12] Xiaojun Bi and Junyao Xing. Multi-scale weighted fusion attentive generative adversarial network for single image de-raining. *IEEE Access*, 8:69838–69848, 2020.

[13] Sebastian Bosse, Dominique Maniry, Klaus-Robert Muller, Thomas Wiegand, and Wojciech Samek. Deep neural networks for no-reference and full-reference image quality assessment. *IEEE Transactions on Image Processing*, pages 1–1, 2017.

[14] Jérémie Bossu, Nicolas Hautière, and Jean Philippe Tarel. Rain or snow detection in image sequences through use of a histogram of orientation of streaks. *International Journal of Computer Vision*, 93(3):348–367, 2011.

[15] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1899–1903, 2017.

[16] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[17] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.

[18] Huiwen Chang, Jingwan Lu, Fisher Yu, and Adam Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 40–48, 2018.

[19] Chen Chen, Minh N Do, and Jue Wang. Robust image and video dehazing with visual artifact suppression via gradient residual minimization. *Proceedings of the European Conference on Computer Vision*, pages 576–591, 2016.

[20] Duan Yu Chen, Chien Cheng Chen, and Li Wei Kang. Visual depth guided color image rain streaks removal using sparse coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(8):1430–1455, 2014.

[21] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520, 2017.

[22] Shixing Chen, Caojin Zhang, Ming Dong, Jialiang Le, and Mike Rao. Using ranking-cnn for age estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5183–5192, 2017.

[23] Wei-Ting Chen, Jian-Jiun Ding, and Sy-Yen Kuo. Pms-net: Robust haze removal based on patch map for single images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11681–11689, 2019.

[24] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[25] Yi Lei Chen and Chiou Ting Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[26] Zhihua Chen, Zhuoliang Hu, Bin Sheng, Ping Li, Jinman Kim, and Enhua Wu. Simplified non-locally dense network for single-image dehazing. *The Visual Computer*, 36(10):2189–2200, 2020.

[27] Zongli Chen, Yiyue Wang, Wei Han, Ruyi Feng, and Jia Chen. An improved pretraining strategy-based scene classification with deep learning. *IEEE Geoscience and Remote Sensing Letters*, 17(5):844–848, 2020.

[28] Jaehoon Cho, Seungryong Kim, Dongbo Min, and Kwanghoon Sohn. Single image deraining using time-lapse data. *IEEE Transactions on Image Processing*, 29:7274–7289, 2020.

[29] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[30] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[31] Bijaylaxmi Das, Joshua Peter Ebenezer, and Sudipta Mukhopadhyay. A comparative study of single image fog removal methods. *The Visual Computer*, pages 1–17, 2020.

[32] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[33] Xinghao Ding, Liqin Chen, Xianhui Zheng, Yue Huang, and Delu Zeng. Single image rain and snow removal via guided l0 smoothing filter. *Multimedia Tools and Applications*, 75(5):2697–2712, 2016.

[34] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. *2016 Eighth International Conference on Quality of Multimedia Experience*, pages 1–6, 2016.

[35] Shuangli Du, Yiguang Liu, Mao Ye, Zhenyu Xu, Jie Li, and Jianguo Liu. Single image deraining via decorrelating the rain streaks and background scene in gradient domain. *Pattern Recognition*, 79:303–317, 2018.

[36] Yingjun Du, Jun Xu, Xiantong Zhen, Ming-Ming Cheng, and Ling Shao. Conditional variational image deraining. *IEEE Transactions on Image Processing*, 29:6288–6301, 2020.

[37] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics*, 31(4):1–10, 2012.

[38] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[39] Xin Fan, Xianxuan Tang, Minjun Hou, and Zhongxuan Luo. Fast example searching for input-adaptive data-driven dehazing with gaussian process regression. *The Visual Computer*, 35(4):565–577, 2019.

[40] Z. Fan and R. L. de Queiroz. Identification of bitmap compression history: Jpeg detection and quantizer estimation. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 12(2):230–235, 2003.

[41] Kayvon Fatahalian. *Enolving the real-time graphics pipeline for micropolygon rendering*. Stanford University, 2011.

[42] Raanan Fattal. Dehazing using color-lines. *ACM Transactions on Graphics*, 34(1):1–14, 2014.

[43] Jie Fu, Hanli Wang, and Lingxuan Zuo. Blind image quality assessment for multiply distorted images via convolutional neural networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1075–1079, 2016.

[44] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017.

[45] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1715–1723, 2017.

[46] Kshitiz Garg and S. K. Nayar. Detection and removal of rain from videos. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[47] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[48] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[49] Edouard Grave, Armand Joulin, Moustapha Cissé, Hervé Jégou, et al. Efficient softmax approximation for gpus. *Proceedings of the 34th International Conference on Machine Learning*, 70:1302–1310, 2017.

[50] Roger Grosse. Lecture 15: Exploding and vanishing gradients. *University of Toronto Computer Science*, 2017.

[51] Liu Guangcan, Lin Zhouchen, Yan Shuicheng, Sun Ju, Yu Yong, and Ma Yi. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

[52] Yann Guermeur and Emmanuel Monfrini. A quadratic loss multi-class svm for which a radius–margin bound applies. *Informatica*, 22(1):73–96, 2011.

[53] Tiantong Guo and Vishal Monga. Reinforced depth-aware deep learning for single image dehazing. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8891–8895, 2020.

[54] Taeyoung Hahn, Myeongjang Pyeon, and Gunhee Kim. Self-routing capsule networks. *Advances in Neural Information Processing Systems*, pages 7656–7665, 2019.

[55] Mohammad A. Haque, Kamal Nasrollahi, and Thomas B. Moeslund. Quality-aware estimation of facial landmarks in video sequences. *Applications of Computer Vision*, pages 678–685, 2015.

[56] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[57] SM Kamrul Hasan and Cristian A Linte. U-netplus: A modified encoder-decoder u-net architecture for semantic and instance segmentation of surgical instruments from laparoscopic images. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 7205–7211, 2019.

[58] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[59] Kaiming He, Sun Jian, and Xiaoou Tang. Single image haze removal using dark channel prior. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *Proceedings of the European Conference on Computer Vision*, pages 630–645, 2016.

[61] Li He and Jie Zhang. Snowflakes removal for single image based on model pruning and generative adversarial network. *2019 IEEE 4th International Conference on Image, Vision and Computing*, pages 172–176, 2019.

[62] Avinash Hindupur. Gan zoo. https://github.com/hindupuravinash/the-gan-zoo.

[63] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. *Proceedings of the International Conference on Artificial Neural Networks*, pages 44–51, 2011.

[64] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. *Proceedings of the International Conference on Learning Representations*, 2018.

[65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[66] Assaf Hoogi, Brian Wilcox, Yachee Gupta, and Daniel L Rubin. Self-attention capsule networks for image classification. *arXiv preprint arXiv:1904.12483*, 2019.

[67] Y Horita, S Arata, and T Murai. No-reference image quality assessment for jpeg/jpeg2000 coding. *Signal Processing Conference, European*, pages 1301–1304, 2004.

[68] Weilong Hou, Xinbo Gao, Dacheng Tao, and Xuelong Li. Blind image quality assessment via deep learning. *IEEE Transactions on Neural Networks & Learning Systems*, 26(6):1275–1286, 2015.

[69] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and H Adam Mobilenets. Efficient convolutional neural networks for mobile vision applications. *arXiv preprint ArXiv:1704.0486*, 2017.

[70] Peiyun Hu and Deva Ramanan. Finding tiny faces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[71] Peiyun Hu and Deva Ramanan. Finding tiny faces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–959, 2017.

[72] De An Huang, Li Wei Kang, Y. C. F. Wang, and Chia Wen Lin. Self-learning based image decomposition with applications to single image denoising. *IEEE Transactions on Multimedia*, 16(1):83–93, 2013.

[73] De-An Huang, Li-Wei Kang, Min-Chun Yang, Chia-Wen Lin, and Yu-Chiang Frank Wang. Context-aware single image rain removal. *2012 IEEE International Conference on Multimedia and Expo*, pages 164–169, 2012.

[74] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[75] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[76] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4):1–11, 2016.

[77] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[78] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5967–5976, 2017.

[79] Ayush Jaiswal, Wael AbdAlmageed, and Premkumar Natarajan. Capsulegan: Generative adversarial capsule network. *arXiv preprint arXiv:1802.06167*, 2018.

[80] Kui Jiang, Zhongyuan Wang, Peng Yi, Chen Chen, Baojin Huang, Yimin Luo, Jiayi Ma, and Junjun Jiang. Multi-scale progressive fusion network for single image deraining. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8346–8355, 2020.

[81] Xin Jin, Zhibo Chen, and Weiping Li. Ai-gan: Asynchronous interactive generative adversarial network for single image rain removal. *Pattern Recognition*, 100:107143, 2020.

[82] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European Conference on Computer Vision*, pages 694–711, 2016.

[83] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

[84] Ioannis A Kakadiaris, George Toderici, Georgios Evangelopoulos, Georgios Passalis, Dat Chu, Xi Zhao, Shishir K Shah, and Theoharis Theoharis. 3d-2d face recognition with pose and illumination normalization. *Computer Vision and Image Understanding*, 154:137–151, 2017.

[85] Le Kang, Peng Ye, Yi Li, and David Doermann. Simultaneous estimation of image quality and distortion via multi-task convolutional neural networks. *Proceedings of the IEEE International Conference on Image Processing*, pages 2791–2795, 2015.

[86] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE transactions on image processing*, 21(4):1742–1755, 2011.

[87] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE Transactions on Image Processing*, 21(4):1742, 2012.

[88] Samil Karahan, Merve Kilinc Yildirum, Kadir Kirtac, Ferhat Sukru Rende, Gultekin Butun, and Hazim Kemal Ekenel. How image degradations affect deep cnn-based face recognition? 2016.

[89] Asem Khmag, SAR Al-Haddad, Bahareh Kalantar, et al. Single image dehazing using second-generation wavelet transforms and the mean vector l2-norm. *The visual computer*, 34(5):675–688, 2018.

[90] Jin Hwan Kim, Chul Lee, Jae Young Sim, and Chang Su Kim. Single-image deraining using an adaptive nonlocal means filter. *Proceedings of the IEEE International Conference on Image Processing*, 2014.

[91] Jin-Hwan Kim, Jae-Young Sim, and Chang-Su Kim. Video deraining and desnowing using temporal correlation and low-rank matrix completion. *IEEE Transactions on Image Processing*, 24(9):2658–2670, 2015.

[92] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning*, 70:1857–1865, 2017.

[93] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[94] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[95] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9808, 2020.

[96] Martin Koestinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2144–2151, 2011.

[97] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. *Advances in Neural Information Processing Systems*, pages 15486–15496, 2019.

[98] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[99] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[100] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics*, 33(4):149, 2014.

[101] Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. *Conference on Medical Imaging with Deep Learning*, 2018.

[102] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[103] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II–104, 2004.

[104] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[105] Jin Han Lee, Sehyung Lee, Guoxuan Zhang, Jongwoo Lim, Wan Kyun Chung, and Il Hong Suh. Outdoor place recognition in urban environments using straight lines. *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, pages 5550–5557, 2014.

[106] Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3587–3596, 2017.

[107] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. *Proceedings of the IEEE International Conference on Computer Vision*, 1(4):7, 2017.

[108] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. *IEEE Transactions on Image Processing*, 28(1):492–505, 2018.

[109] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *Proceedings of the European Conference on Computer Vision*, pages 702–716, 2016.

[110] Pengyue Li, Mengshen Yun, Jiandong Tian, Yandong Tang, Guolin Wang, and Chengdong Wu. Stacked dense networks for single-image snow removal. *Neurocomputing*, 367:152–163, 2019.

[111] Runde Li, Jinshan Pan, Zechao Li, and Jinhui Tang. Single image dehazing via conditional generative adversarial network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8202–8211, 2018.

[112] Ruoteng Li, Loong-Fah Cheong, and Robby T Tan. Single image deraining using scale-aware multi-stage recurrent network. *arXiv preprint arXiv:1712.06830*, 2017.

[113] Xia Li, Jianlong Wu, Zhouchen Lin, Hong Liu, and Hongbin Zha. Recurrent squeeze-and-excitation context aggregation net for single image

deraining. *Proceedings of the European Conference on Computer Vision*, pages 262–277, 2018.

[114] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2736–2744, 2016.

[115] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2736–2744, 2016.

[116] Zhi Li, Juan Zhang, Zhijun Fang, Bo Huang, Xiaoyan Jiang, Yongbin Gao, and Jenq Neng Hwang. Single image snow removal via composition generative adversarial networks. *IEEE Access*, PP:1–1, 2019.

[117] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[118] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *Proceedings of the European Conference on Computer Vision*, pages 740–755, 2014.

[119] Wang G . Lin K Y. Hallucinated-iqa: No-reference image quality assessment via adversarial learning. *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[120] Hao Liu, Jiwen Lu, Jianjiang Feng, and Jie Zhou. Group-aware deep feature learning for facial age estimation. *Pattern Recognition*, 66:82–94, 2017.

[121] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[122] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *Proceedings of the European Conference on Computer Vision*, pages 21–37, 2016.

[123] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220, 2017.

[124] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. *Proceedings of the International Conference on Machine Learning*, 2(3):7, 2016.

[125] Xialei Liu, Van De Weijer Joost, and Andrew D Bagdanov. Rankiqa: Learning from rankings for no-reference image quality assessment. *Proceedings of the International Conference on Computer Vision*, 2017.

[126] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality aware network for set to set recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[127] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3000–3009, 2017.

[128] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *IEEE Transactions on Image Processing*, 27(6):3064–3073, 2018.

[129] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *IEEE Transactions on Image Processing*, 27(6):3064–3073, 2018.

[130] Jan Lukáš and Jessica Fridrich. Estimation of primary quantization matrix in double compressed jpeg images. *Proceedings of the Digital Forensic Research Workshop*, pages 5–8, 2003.

[131] Fulin Luo, Liangpei Zhang, Bo Du, and Lefei Zhang. Dimensionality reduction with enhanced hybrid-graph discriminant learning for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(8):5336–5353, 2020.

[132] Yu Luo, Yong Xu, and Hui Ji. Removing rain from a single image via discriminative sparse coding. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3397–3405, 2015.

[133] Kede Ma, Wentao Liu, Kai Zhang, Zhengfang Duanmu, Zhou Wang, and Wangmeng Zuo. End-to-end blind image quality assessment using deep neural networks. *IEEE Transactions on Image Processing*, 27(3):1202–1213, 2018.

[134] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European Conference on Computer Vision*, pages 116–131, 2018.

[135] Luca Marchesotti and Rodrigue Nkoutche. Image quality assessment, 2012.

[136] E. J. Mccartney and Freeman F. Hall Jr. Optics of the atmosphere: Scattering by molecules and particles. 1976.

[137] Gaofeng Meng, Ying Wang, Jiangyong Duan, Shiming Xiang, and Chunhong Pan. Efficient image dehazing with boundary constraint and contextual regularization. *Proceedings of the IEEE International Conference on Computer Vision*, pages 617–624, 2013.

[138] Tomáš Mikolov. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80, 2012.

[139] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[140] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? *Proceedings of the European Conference on Computer Vision*, pages 202–217, 2016.

[141] Yair Movshovitz-Attias, Yaser Sheikh, Vishnu Naresh Boddeti, and Zijun Wei. 3d pose-by-detection of vehicles via discriminatively reduced ensembles of correlation filters. *Proceedings of the British Machine Vision Conference*, 2014.

[142] Lawrence Mutimbu and Antonio Robles-Kelly. A factor graph evidence combining approach to image defogging. *Pattern Recognition*, 82:56–67, 2018.

[143] Srinivasa G. Narasimhan and Shree K. Nayar. Vision and the atmosphere. *International Journal of Computer Vision*, 48(3):233–254, 2002.

[144] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[145] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *Advances in Neural Information Processing Systems*, 2011.

[146] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. *Proceedings of the International Conference on Pattern Recognition*, pages 850–855, 2006.

[147] Duc Thanh Nguyen, Wanqing Li, and Philip O Ogunbona. Human detection from images and videos: A survey. *Pattern Recognition*, 51:148–175, 2016.

[148] Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. *ICASSP 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2307–2311, 2019.

[149] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1609.02781*, 2016.

[150] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *Proceedings of the 34th International Conference on Machine Learning*, 70:2642–2651, 2017.

[151] Kuntal Kumar Pal and KS Sudeep. Preprocessing for image classification by convolutional neural networks. *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology*, pages 1778–1781, 2016.

[152] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[153] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. *Proceedings of the British Machine Vision Conference*, 1(3):6, 2015.

[154] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[155] Zhao Pei, Min Jin, Yanning Zhang, Miao Ma, and Yee-Hong Yang. All-in-focus synthetic aperture imaging using generative adversarial network-based semantic inpainting. *Pattern Recognition*, 111:107669, 2020.

[156] Tomás¿ Pevny and Jessica Fridrich. Detection of double-compression in jpeg images for applications in steganography. *IEEE Transactions on Information Forensics & Security*, 3(2):247–258, 2008.

[157] P Jonathon Phillips and Yehuda Vardi. Efficient illumination normalization of facial images. *Pattern Recognition Letters*, 17(8):921–927, 1996.

[158] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in Neural Information Processing Systems*, pages 2352–2360, 2016.

[159] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial

animation from a single image. *Proceedings of the European Conference on Computer Vision*, pages 818–833, 2018.

[160] Yanyun Qu, Yizi Chen, Jingying Huang, and Yuan Xie. Enhanced pix2pix dehazing network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8160–8168, 2019.

[161] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[162] Rafat, Mantiuk, Kil, Joong, Kim, Allan, G., Rempel, Wolfgang, and Heidrich. Hdr-vdp-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM Transactions on Graphics*, 2011.

[163] Suresh Chandra Raikwar and Shashikala Tapaswi. Tight lower bound on transmission for single image dehazing. *The Visual Computer*, 36(1):191–209, 2020.

[164] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10725–10733, 2019.

[165] Dhanashree Rajderkar and PS Mohod. Removing snow from an image via image decomposition. *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology*, pages 576–579, 2013.

[166] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[167] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[168] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[169] Dongwei Ren, Wei Shang, Pengfei Zhu, Qinghua Hu, Deyu Meng, and Wangmeng Zuo. Single image deraining using bilateral recurrent network. *IEEE Transactions on Image Processing*, 29:6852–6863, 2020.

[170] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: a better and simpler baseline. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2019.

[171] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[172] Wenqi Ren, Si Liu, Hua Zhang, Jinshan Pan, Xiaochun Cao, and Ming-Hsuan Yang. Single image dehazing via multi-scale convolutional neural networks. *Proceedings of the European Conference on Computer Vision*, pages 154–169, 2016.

[173] Wenqi Ren, Lin Ma, Jiawei Zhang, Jinshan Pan, Xiaochun Cao, Wei Liu, and Ming-Hsuan Yang. Gated fusion network for single image dehazing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2018.

[174] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1-3):19–41, 2000.

[175] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[176] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[177] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[178] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[179] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[180] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[181] Varun Santhaseelan and Vijayan K. Asari. Utilizing local phase information to remove rain from video. *International Journal of Computer Vision*, 112(1):71–89, 2015.

[182] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[183] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[184] Yuanjie Shao, Lerenhan Li, Wenqi Ren, Changxin Gao, and Nong Sang. Domain adaptation for image dehazing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2808–2817, 2020.

[185] Hamid R Sheikh and Alan C Bovik. Image information and visual quality. *IEEE Transactions on image processing*, 15(2):430–444, 2006.

[186] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2107–2116, 2017.

[187] Xiangbo Shu, Jinhui Tang, Guojun Qi, Wei Liu, and Jian Yang. Hierarchical long short-term concurrent memory for human interaction recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[188] Xiangbo Shu, Liyan Zhang, Guo-Jun Qi, Wei Liu, and Jinhui Tang. Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction. *arXiv preprint arXiv:1909.13245*, 2019.

[189] Xiangbo Shu, Liyan Zhang, Yunlian Sun, and Jinhui Tang. Host-parasite: Graph lstm-in-lstm for group activity recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[190] D Amnon Silverstein and Joyce E Farrell. The relationship between image fidelity and image quality. *Proceedings of the IEEE International Conference on Image Processing*, pages 881–884, 1996.

[191] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014.

[192] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[193] Bikesh Kumar Singh, Kesari Verma, and AS Thoke. Adaptive gradient descent backpropagation for classification of breast tumors in ultrasound imaging. *Procedia Computer Science*, 46:1601–1609, 2015.

[194] Chang Hwan Son and Xiao Ping Zhang. Rain removal via shrinkage of sparse codes and learned rain dictionary. *IEEE International Conference on Multimedia and Expo Workshops*, 2016.

[195] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 190–198, 2017.

[196] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[197] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[198] Matan Sulami, Itamar Glatzer, Raanan Fattal, and Mike Werman. Automatic recovery of the atmospheric light in hazy images. *Proceedings of the IEEE International Conference on Computational Photography*, 2014.

[199] S. H. Sun, S. P. Fan, and Y. C. F. Wang. Exploiting image structural similarity for single image rain removal. *Proceedings of the IEEE International Conference on Image Processing*, 2015.

[200] Xudong Sun, Pengcheng Wu, and Steven C. H Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 2017.

[201] Xudong Sun, Pengcheng Wu, and Steven CH Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 299:42–50, 2018.

[202] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-first AAAI conference on artificial intelligence*, 2017.

[203] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[204] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[205] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[206] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018.

[207] Robby T Tan. Visibility in bad weather from a single image. *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[208] Jinhui Tang, Zechao Li, Hanjiang Lai, Liyan Zhang, Shuicheng Yan, et al. Personalized age progression with bi-level aging dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):905–917, 2017.

[209] Jean-Philippe Tarel and Nicolas Hautiere. Fast visibility restoration from a single color or gray level image. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2201–2208, 2009.

[210] Justus Thies, Michael Zollhfer, and Matthias Niener. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics*, 38(4):1–12, 2019.

[211] Abhishek Kumar Tripathi and Sudipta Mukhopadhyay. Removal of rain from videos: a review. *Signal Image and Video Processing*, 8(8):1421–1430, 2014.

[212] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[213] Saurabh Verma and Zhi-Li Zhang. Graph capsule convolutional neural networks. *Joint ICML and IJCAI Workshop on Computational Biology*, 2018.

[214] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. Perceptual adversarial networks for image-to-image transformation. *IEEE Transactions on Image Processing*, 27(8):4066–4079, 2018.

[215] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[216] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[217] Ruoyu Wang, Guangyao Li, and Dongsheng Chu. Capsules encoder and capsgan for image inpainting. *Proceedings of the IEEE International Conference on Artificial Intelligence and Advanced Manufacturing*, pages 325–328, 2019.

[218] Tianyu Wang, Xin Yang, Ke Xu, Shaozhe Chen, Qiang Zhang, and Rynson WH Lau. Spatial attentive single-image deraining with a high quality real rain dataset. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12270–12279, 2019.

[219] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[220] Y. Wang, S. Liu, C. Chen, and B. Zeng. A hierarchical approach for rain or snow removing in a single color image. *IEEE Transactions on Image Processing*, 26(8):3936–3950, 2017.

[221] Z Wang and Q Li. Information content weighting for perceptual image quality assessment. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 20(5):p.1185–1198, 2011.

[222] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. *Signals, Systems and Computers, IEEE*, 2004.

[223] Zhou Wang and Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002.

[224] Zhou Wang, Alan C Bovik, and Ligang Lu. Why is image quality assessment so difficult? *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4:IV–3313, 2002.

[225] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. *Proceedings of the European Conference on Computer Vision*, pages 499–515, 2016.

[226] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. *Proceedings of the European Conference on Computer Vision*, pages 499–515, 2016.

[227] Wenqi Wu, Yingjie Yin, Xingang Wang, and De Xu. face detection with different scales based on faster r-cnn. *IEEE Transactions on Cybernetics*, 49(11):4017–4028, 2019.

[228] Edgar Xi, Selina Bing, and Yang Jin. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

[229] Guo-Sen Xie, Xu-Yao Zhang, and Cheng-Lin Liu. Efficient feature coding based on auto-encoder network for image classification. *Proceedings of the Asian Conference on Computer Vision*, pages 628–642, 2014.

[230] Guo Sen Xie, Xu Yao Zhang, and Cheng Lin Liu. Efficient feature coding based on auto-encoder network for image classification. *Proceedings of the Asian Conference on Computer Vision*, 2014.

[231] Zhang Xinyi and Lihui Chen. Capsule graph neural network. *International Conference on Learning Representations*, 2018.

[232] Jing Xu, Wei Zhao, Peng Liu, and Xianglong Tang. An improved guidance image based method to remove rain and snow in a single image. *Computer and Information Science*, 5(3):49, 2012.

[233] Jing Xu, Wei Zhao, Peng Liu, and Xianglong Tang. Removing rain and snow in a single image using guided filter. *2012 IEEE International Conference on Computer Science and Automation Engineering*, 2:304–307, 2012.

[234] Q. Yan, D. Gong, and Y. Zhang. Two-stream convolutional networks for blind image quality assessment. *IEEE Transactions on Image Processing*, 28(5):2200–2211, 2019.

[235] Fei Yang, Zheng Lu, Guoping Qiu, Jing Lin, and Qian Zhang. Capsule based image synthesis for interior design effect rendering. *Proceedings of the Asian Conference on Computer Vision*, 2018.

[236] Fei Yang, Qian Zhang, Miaohui Wang, and Guoping Qiu. Quality classified image analysis with application to face detection and recognition. *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 2863–2868, 2018.

[237] Qingsong Yang, Pingkun Yan, Yanbo Zhang, Hengyong Yu, Yongyi Shi, Xuanqin Mou, Mannudeep K Kalra, Yi Zhang, Ling Sun, and Ge Wang. Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE Transactions on Medical Imaging*, 37(6):1348–1357, 2018.

[238] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5533, 2016.

[239] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1357–1366, 2017.

[240] Xitong Yang, Zheng Xu, and Jiebo Luo. Towards perceptual image dehazing by physics-based disentanglement and adversarial training. *Thirty-second AAAI conference on artificial intelligence*, 2018.

[241] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch. *Computer Science*, 2014.

[242] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2849–2857, 2017.

[243] Shibai Yin, Yibin Wang, and Yee-Hong Yang. A novel image-dehazing network with a parallel attention block. *Pattern Recognition*, 102:107255, 2020.

[244] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.

[245] Luo Yu, Xu Yong, and Ji Hui. Removing rain from a single image via discriminative sparse coding. *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[246] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[247] He Zhang and Vishal M Patel. Densely connected pyramid dehazing network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2018.

[248] He Zhang and Vishal M Patel. Density-aware single image de-raining using a multi-stream dense network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 695–704, 2018.

[249] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[250] Jiawan Zhang, Liang Li, Guoqiang Yang, Yi Zhang, and Jizhou Sun. Local albedo-insensitive single image dehazing. *The Visual Computer*, 26(6):761–768, 2010.

[251] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[252] L Zhang, L Zhang, X Mou, and D Zhang. Fsim : a feature similarity index for image quality assessment. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 20(8):2378–2386, 2011.

[253] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. *European Conference on Computer Vision*, pages 649–666, 2016.

[254] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[255] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[256] Rui Zhang, Tomas Pfister, and Jia Li. Harmonic unpaired image-to-image translation. *Proceedings of the International Conference on Learning Representations*, 2019.

[257] Shengdong Zhang and Fazhi He. Drcdn: learning deep residual convolutional dehazing networks. *The Visual Computer*, 36(9):1797–1808, 2020.

[258] Shengdong Zhang, Fazhi He, Wenqi Ren, and Jian Yao. Joint learning of image detail and transmission map for single image dehazing. *The Visual Computer*, 36(2):305–316, 2020.

[259] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[260] Xiaopeng Zhang, Hao Li, Yingyi Qi, Wee Leow, and Teck Ng. Rain removal in video by combining temporal and chromatic properties. *proceedings of the IEEE International Conference on Multimedia and Expo*, 2006.

[261] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5057–5065, 2017.

[262] Yinda Zhang, Fisher Yu, Shuran Song, Pingmei Xu, Ari Seff, and Jianxiong Xiao. Large-scale scene understanding challenge: Room layout estimation. *accessed on Sep*, 15, 2015.

[263] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[264] Junbo Zhao, Michael Mathieu, and Yann Lecun. Energy-based generative adversarial network. *arXiv preprint arXiv:1511.06434*, 2016.

[265] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1009–1018, 2019.

[266] A Zhdanov and D Zhdanov. The backward photon mapping for the realistic image rendering. *Proceedings of Conference on Computer Graphics and Machine Vision*, 2744:1–12, 2020.

[267] Xianhui Zheng, Yinghao Liao, Wei Guo, Xueyang Fu, and Xinghao Ding. Single-image-based rain and snow removal using multi-guided filter. *International Conference on Neural Information Processing*, pages 258–265, 2013.

[268] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.

[269] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[270] Wang Zhou, Bovik Alan Conrad, Sheikh Hamid Rahim, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans on Image Process*, 13(4):600–612, 2004.

[271] Zhiming Zhou, Weinan Zhang, and Jun Wang. Inception score, label smoothing, gradient vanishing and-log (d (x)) alternative. *arXiv preprint arXiv:1708.01729*, 2017.

[272] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, 2018.

[273] Hongyuan Zhu, Xi Peng, Vijay Chandrasekhar, Liyuan Li, and Joo-Hwee Lim. Dehazegan: When image dehazing meets differential programming. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1234–1240, 2018.

[274] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. *Proceedings of the European Conference on Computer Vision*, pages 597–613, 2016.

[275] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[276] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

[277] Lei Zhu, Chi-Wing Fu, Dani Lischinski, and Pheng-Ann Heng. Joint bilayer optimization for single-image rain streak removal. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2526–2534, 2017.

[278] Qingsong Zhu, Jiaming Mai, and Ling Shao. A fast single image haze removal algorithm using color attenuation prior. *IEEE Transactions on Image Processing*, 24(11):3522–3533, 2015.

Appendix A

List of Abbreviations

AE	Auto-encoder
BIQA	Blind Image Quality Assessment
VAE	Variational Autoencoder
CapsDR	Capsule De-Raining
CapsGAN	Capsule Generative Adversarial Network
cGAN	Conditioned Generative Adversarial Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DCP	Dark Channel Prior
DDN	Depth De-hazing Network
DePrimaryCaps	DePrimary Capsule layer
FC	Fully Connection
FC_Caps	Fully Connected Capsule layer
FN	False Negative
FP	False Positive
GAN	Generative Adversarial network
GCN	Graph Convolutional Network
GCNN	Graph Convolutional Neural Network
GNN	Graph Neural Network
HIDER	Home Interior Design Effect Rendering
HOG	Histogram of Oriented Gradients
IDIC	International Doctoral Innovation Centre
image-cGAN	Image-conditioned Generative Adversarial Network
IoU	Intersection over Union
IQA	Image Quality Assessment
LBP	Local Binary Pattern
LSTM	Long-Short Term Memory
mAP	mean Average Precision
mAR	mean Average Recall
MAE	Mean Average Error
MC-DS	Multi-scale Capsule DeSnowing

NR-IQA	Non-Referenced Image Quality Assessment
PCA	Principal Component Analysis
PR	Precision-Recall
PrimaryCaps	Primary Capsule layer
PSNR	Peak Signal-to-Noise Ratio
RBM	Restricted Boltzmann Machine
RCA	Rain Component Aware
RCAC	Rain Component Aware Capsule
RNN	Recurrent Neural Network
RR-IQA	Reduced Referenced Image Quality Assessment
R-IQA	Referenced Image Quality Assessment
SSIM	Structural Similarity Index Measure
SVM	Support Vector Machine
TP	True Positive
TN	True Negative