University of Nottingham
UK | CHINA | MALAYSIA

New Approaches for the Integration of the Discrete Cosine Transform in Neural Networks for Fine-Grained Image Classification

Submitted 15 March 2023, in partial fulfilment of the conditions for the award of the degree **Doctor of Philosophy.**

KELVIN TAN SIM ZHEN

Student ID: 18023285

Supervised by: **Dr Tomas Maul**
Secondary Supervisor: **Dr Nafizah Goriman Khan**
External Supervisors: **Dr Khor Jeen Ghee**
Dr Wong Yee Wan
Dr Thomas Ooi Wei Min

Department of Electrical and Electronic Engineering
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature: 

Date 15 / 03 / 2023

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Public access to this dissertation is restricted until: DD/MM/YYYY

Declaration

I declare that all the research work conducted in this thesis was completed in the Department of Electrical and Electronic Engineering at the University of Nottingham Malaysia between March 2019 and September 2022. The work completed in this thesis is, to the best of my knowledge, novel and has not been submitted for any other degree of any other universities.

Kelvin Tan Sim Zhen
University of Nottingham Malaysia
15 March 2023

Abstract

A convolutional neural network (CNN) is a popular neural network architecture that excels in its ability to capture patterns in tasks with grid-structured inputs (e.g. visual recognition). Fine-grained visual classification (FGVC) uses CNN to categorise images of high intra-class and low inter-class variance. According to the literature, the 2D Discrete Cosine Transform (DCT) is one of the well-known transformations used in compression for its robustness and high data compaction properties. In compressed domain image classification, many works have focused on extracting features from the low DCT coefficients (L-DCTCs) through a fully pointwise vanilla CNN. Here, the abundant medium to high DCTCs have typically been discarded. Although pointwise convolution is capable of complex transformations, the spatial context and representation are limited. The area of compressed domain FGVC remains a relatively inactive field. It is therefore essential to explore compressed domain FGVC under DCT conditions to investigate the relationship between fine-grained features and the full spectrum of DCTCs. More specifically, this thesis intends to adopt and extend DCT techniques in compressed domain FGVC to address three topics: (1) the usability and inclusive learning of mid-band DCTCs; (2) the adaptive learning of DCT basis functions on composing the pointwise convolutional kernels; (3) the interaction between DCT channel groups in feature representations. The first contribution introduces the 'Skipped Medium DCT CNN'. The M-DCTCs were processed via a skipping branch with a shallow convolutional block alongside the L-DCTCs which were passed through the main branch of the CNN. This architecture achieved a classification error drop of up to 7% over the standard model without the skipping branch. It highlights the importance of combining higher-frequency DCTCs with lower ones for improved robustness. The second contribution enhances the prior network by adaptively weighting the DCT basis functions to form a pointwise convolutional kernel. The spatial details were considered when constructing the pointwise convolutional kernel apart from the frequency contents. The adaptive weights are referred to as the 'Adaptive DCT (Adapt-DCT)' kernel. This network achieved up to 8% classification error drop on small-scale FGVC datasets and a top-5 testing accuracy of 73.93% on mini-ImageNet. The third contribution investigates the significance of DCT feature groups in the compressed domain FGVC. The modified attention mechanism that prioritises the channel interactions within the DCT group is referred to as the 'Hybrid Modified Efficient

Channel Attention' (HyMod-ECA). It reduces the classification error by up to 3.5% over the original ECA. The optimised Adapt-DCT CNN with HyMod-ECA achieves a substantial parameter reduction of up to 73%. It is shown that the interactions among the DCT feature groups are one of the promising mechanisms to ease compressed domain FGVC. To conclude, this thesis discusses novel contributions in the context of combining the higher frequency DCTCs via a DCT-oriented convolutional kernel with an attention mechanism to address compressed domain FGVC.

Acknowledgements

First of all, I would like to convey my deepest gratitude to my main supervisor Dr Tomas Maul for his endless support along my PhD journey. Dr Tomas had definitely guided me mentally and practically to complete my research work, through a lot of ups and downs. His continuous care on my mental health and life besides PhD is impeccable. I would also love to thank my secondary supervisor Dr Nafizah for her guidance on teaching me the technical writing style of a journal. Her diligent on every bit of writing details had certainly turned me into a better academic writer. I also wish to thank my internal examiner Dr Hermawan for his valuable feedback during the annual review. Besides, I wish to appreciate my ex-supervisors Dr Khor Jeen Ghee and Dr Wong Yee Wan for providing me an interesting research topic and supporting my first year in PhD to kickstart the research work. In addition, I am indebted to my industry supervisor Dr Thomas Ooi from Intel Corporation. He provided me a one-year industry placement to be equipped with tons of technical skillset that are helpful throughout my PhD journey. Finally, I am blessed to have parents that can offer me an opportunity to pursue my PhD dream. I am also truly blessed to have a kind, diligent and sympathetic girlfriend who has helped me to achieve my every milestone along the PhD journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Full Term
AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalisation
CBAM	Convolutional Block Attention Mechanism
CDA	Compressed Domain Analytics
CDIA	Compressed Domain Image Analytics
CNN	Convolutional Neural Network
CODEC	Compression Decompression
CPU	Central Processing Unit
DCT	Discrete Cosine Transformation
DCT-BF	Discrete Cosine Transform Basis Functions
DCTC	Discrete Cosine Transform Coefficients
DFT	Discrete Fourier Transformation
DL	Deep Learning
DNN	Deep Neural Network
DWT	Discrete Wavelet Transformation
ECA	Efficient Channel Attention
FC	Fully Connected (layers)
FFT	Fast Fourier Transformation
FGVC	Fine-Grained Visual Classification

FM	Feature Maps
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
H-DCTC	High frequency Discrete Cosine Transform Coefficients
HyMod	Feature Maps
IOT	Internet of Things
JPEG	Joint Photographic Experts Group
L-DCTC	Low frequency Discrete Cosine Transform Coefficients
LMH-DCTC	Low, Medium, High frequency Discrete Cosine Transform Coefficients
MCP	McCulloch-Pitt
M-DCTC	Medium frequency Discrete Cosine Transform Coefficients
ReLU	Rectified Linear Unit function
SOTA	State of the Art
TLU	Threshold Logic Unit

Chapter 1 Introduction

1.1 Background

Artificial intelligence (AI) is a field that is dedicated to the development of algorithms that are aimed to make smart decisions like a human. The deep learning (DL) approach is a subset of AI that uses complex neural networks to imitate how the human brain works. It is composed of a complex mathematical algorithm that contains adjustable weights. A very deep model can recognise data patterns and construct complex feature representations. It can even match or excel in human-level performance. An AI chatbot (ChatGPT) that implements a deep learning algorithm was launched in November 2022 and accumulated up to 100 million worldwide users in just 2 months [1]. It can interact with users by addressing specific questions through an interactive response. The deep learning approach is also used in the camera application of mobile devices for image processing. The 'photonic engine' is a module developed by Apple that enhances photographs on an iPhone [2] through computational image processing by using deep learning models. In 2023, about 19% of the worldwide population (1.5 billion out of 8 billion) owns an iPhone [3]. In other words, deep learning plays a critical role in modern human life.

A Convolutional Neural Network (CNN) as shown in Figure 1.1 is a type of deep neural network that is formed by stacking sequential layers between input and output layers at both ends. It is commonly found in image classification [4] and object detection [5]. CNNs [6] are biologically inspired by the concept of the human visual cortex. CNNs possess the ability to extract the regular patterns in the input space. It operates by convolving a reduced set of weights, known as kernels, across the input data and producing the corresponding output, often referred to as feature maps. The local receptive field is a small region within the input where the convolutional weights engage. During convolution, the same set of weights is used to receive information across the whole extent of the input space. This weight-sharing technique used by CNN further reduces its number of trainable parameters compared to other architectures like Multilayered Perceptron (MLP).

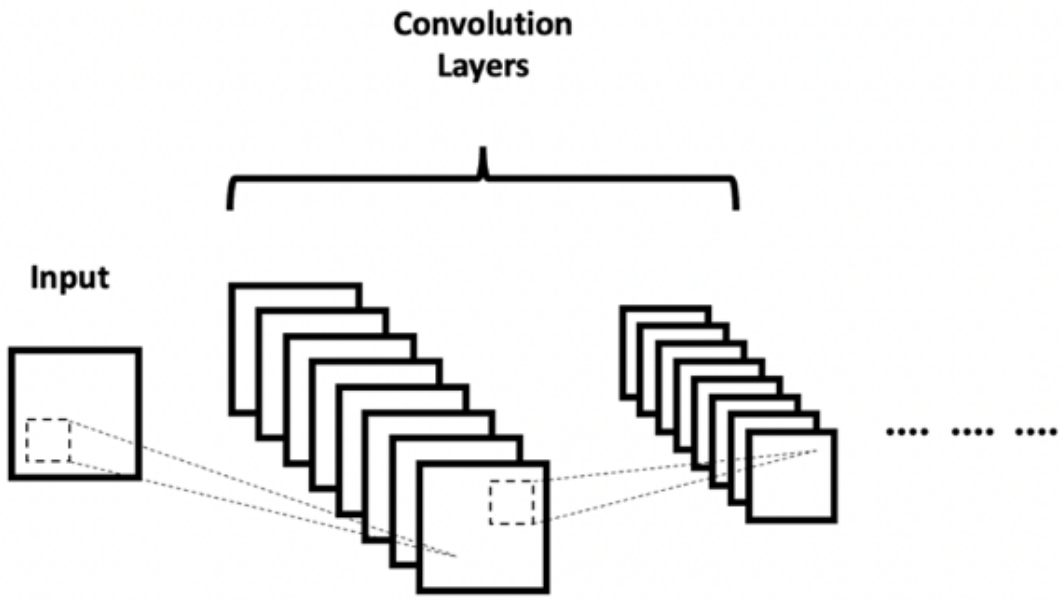


Figure 1.1: An illustration of a CNN.

An RGB image is the spatial representation of static visual content. It is formed by several pixels over the spatial context. A conventional CNN typically receives raw pixels from RGB images as the input [7] for image classification. The working strategy of developing such CNNs is considered to be in the spatial domain CNN. By convolving different kernels over the input space, a collection of distinct patterns is formed. These patterns are combined to form the output feature maps consisting of multiple channels. With the increasing demands on higher resolution visual contents [8], more advanced CNNs are required to process these extensive data. A complex CNN model such as ResNet-152 [9] can take weeks to develop due to its high complexity and computational cost.

In fact, it was shown that an image representation in the spatial domain can carry redundancy [10]. This scenario is particularly prevalent in fine-grained images, where several works have been carried out such as [11][12][13][14][15] to mitigate its negative impact on the model's performance. Fine-grained visual classification (FGVC) is a type of visual recognition task that exhibits low inter-class and high intra-class variance. It corresponds to the problem of classifying between hard-to-distinguish object classes, such as animal species and vehicle models. FGVC emphasises learning the most discriminative pattern, which means only part of the image is useful for describing the subtle differences. When the convolutional kernel processes redundant data, it generates a highly similar pattern, which can be considered as information

redundancy. In other words, the features produced by fine-grained images can contain high redundancy in spatial domain CNNs [16]. The emergence of these trends could lead to a less robust deep learning framework.

To overcome the challenges above, the idea of implementing compressed domain analytics in deep learning has triggered noticeable research interest [17]. Compressed domain analytics refers to transforming the spatial domain information into its relative frequency counterpart when implementing deep learning algorithms. The objective of this technique is to improve compression gain and promote a lightweight and robust CNN development process. This approach also facilitates resource optimisation by reducing the overall memory footprint across the entire deep learning development system. Resource optimisation in deep learning involves minimising computational costs, memory usage, and processing time while maintaining or enhancing model performance. For fine-grained visual classification, resource optimisation is particularly critical because these tasks require intricate feature extraction, leading to large, computationally intensive models. Achieving resource optimisation can involve various strategies such as data preprocessing, dataset management, model design, algorithmic improvements, and the use of hardware accelerators. In this thesis, the focus is on optimising data processing and model algorithms. Techniques like model compression and parameter reduction are employed to decrease the model's memory footprint and boost processing efficiency, directly addressing the challenges of resource optimisation in fine-grained visual classification tasks. In essence, images are commonly stored in a compressed format to reduce redundancy, called the 'Joint Photographic Expert Group' (JPEG). The JPEG CODEC algorithm uses 2D-Discrete Cosine Transformation (2D-DCT) to compress an image. A JPEG image is produced by applying 2D-DCT to an RGB image, followed by several standard encodings. A set of Discrete Cosine Transform Coefficients (DCTCs) are formed along the compression process. It contains different sets of basis functions in both spatial and frequency contexts. It eliminates spatial redundancy and delivers compact features for CDA [10]. In fact, the DCTCs of an image can be obtained by either performing forward 2D-DCT on the RGB image or by decoding a JPEG image without going through the inverse 2D-DCT. The DCTCs represent an image by combining several frequency bands. Its data compaction properties [18][19] allow it to consolidate most of the useful information towards the L-DCTCs. Hence, the DCTCs possess higher compression gain and lower redundancy as compared to an RGB image. The DCTCs are widely used in compressed domain CNNs

[20] as they do not carry imaginary components. Thus, it helps to regulate the interpretability of a CNN model.

Compressed domain analytics have been widely adopted in various applications, especially in machine vision [21]. It is also found in general classification tasks such as CIFAR-10 [22], CIFAR-100 [23], and ImageNet [24][25]. The research community has widely implemented pruning techniques by using only the low DCTCs (L-DCTCs) for general classification tasks [26][27] in compressed domain CNN. The medium to high DCTC frequency bands is discarded under most circumstances as they are believed to exhibit noise contamination. Commonly, the L-DCTCs are sufficient for feature extraction in such cases, where the images carry high inter-class and low intra-class variance. It offers a robust and lightweight model with fewer parameters compared with spatial domain CNN [25][28][29]. Therefore, the working strategy of compressed domain analytics can lead to several advantages, especially in FGVC. However, to detect subtle differences in fine-grained images, solely considering L-DCTCs is challenging. This is because the fine-grained features could be contained within frequency bands beyond L-DCTCs. Therefore, given the established advantages of compressed domain analytics over the spatial domain, and the convergence of these trends, it motivates a study on compressed domain image analytics (CDIA) in the context of small-scale FGVC, an area that has not been previously investigated.

1.2 Problem Statement

To address FGVC problems, conventional CNNs must exhibit exceptional sensitivity to subtle differences, particularly within the spatial context of feature maps. These small changes are analogous to variations in frequency bands in compressed domain analytics, where the recognition of discriminative patterns becomes crucial. Compressed domain analytics, with their ability to analyse features through a broader spectrum of frequency bands, could potentially offer increased robustness compared to spatial domain analysis [30][31][32][33]. While former FGVC research emphasises complexity in the spatial domain [34][35][36][37], limited attention has been given to compressed domain analytics in this context. The existing compressed domain implementations often rely solely on pruning higher frequency bands to focus on L-DCTCs for general classification [38][39]. Although this reduces complexity, it raises concerns about the adequacy of L-DCTCs in fully representing and detecting subtle differences in FGVC. In the representation of compressed domain features through 2D-DCT, the patterns distributed across the spatial context are hierarchically encoded along the channel dimension [24]. In other words, the fine-grained object parts that were originally located across the spatial context are now encoded in the channel dimension. However, relying exclusively on L-DCTCs for classification in FGVC may pose challenges, particularly regarding the loss of critical edge information leading to distortion or artifacts. This information can be essential in forming discriminative patterns, and their absence could hinder accurate classification. To address this limitation and explore comprehensive solutions [40][41], it becomes imperative to consider frequency bands beyond L-DCTCs. In fact, the medium-frequency DCTCs are widely explored in steganography [42][43], whereas a few other papers [44][45] have provided early attempts on mid-high frequency bands for the recognition task, indicating their potential in FGVC. To highlight fine details, one of the strategies is to employ attention mechanisms. It allows the network to focus on individual components carrying unique patterns representing fine-grained details. In compressed domain analytics, a unique pattern could be formed by several frequency bands. It is trivial to consider pieces of individual components, where the collection of patterns can be more profound in representing the most discriminative features. A convolutional kernel is responsible for composing local receptive fields. It is becoming more specific at higher-level intervals, where extended channel depths are present. In the context of spatial

domain FGVC, this implies that a large number of hierarchical kernels across different levels are needed to capture highly specific features. This approach yields less robust outcomes, necessitating a deeper exploration of implementing a compressed domain approach in the convolutional kernel. Although CNNs employ weight-sharing techniques, the inherent redundancy in RGB images in the spatial domain, especially in the context of fine-grained visual classification (FGVC), can lead to inefficient resource usage. Processing this redundant information requires additional parameters, which wastes computational resources and memory. This inefficiency can result in the need for more training data, increased computational complexity, and a longer development timeline. Consequently, the development process can become prohibitively expensive, both in terms of computation and memory, limiting the practical deployment of CNNs in real-world applications. These cumulative challenges highlight the importance of resource optimisation as a key objective of this research, aiming to make deep learning models more efficient and deployable.

In short, the summary of problem statements is:

- The pruning technique in CDA solely considers L-DCTCs, where the frequencies beyond L-DCTCs are mostly discarded. This could lead to suboptimal feature representations and hinder comprehensive frequency information at higher-level frequency bands for FGVC.
- The robustness of kernel analytics is limited at different convolutional block intervals. This limitation can compromise the extraction and synthesis of discriminative features to maintain efficacy across different network depths.
- The conventional attention mechanism predominantly highlights individual components within the feature maps. This approach potentially overlooks the relationships among features to capture feature interactions.

1.3 Research Aim and Objectives

This research aims to employ DCT-based strategies to develop novel deep learning methods for small-scale fine-grained visual classification to achieve resource optimisation.

The objectives are outlined below:

1. To develop a CNN algorithm that integrates different levels of frequency bands in feature maps.
2. To develop a DCT-based strategy to compose the convolutional kernel using DCT basis functions.
3. To develop a hybrid attention mechanism to highlight interactions between channel groups of a frequency nature.

1.4 Contribution of the Research

This research mainly covers the research gap in compressed domain CNNs for FGVC using DCT. The key contributions and accomplishments are:

- A preliminary work on general classification was performed using a fully pointwise compressed domain CNN. This work focused on the novel implementation aspect of the compressed domain CNN by using the OpenVINO inference engine to run the model on the CPU. The work was published in [46] and provided foundational insights for conducting this research work.
- The development of a novel 'M-skipped pointwise convolution branch' aims to motivate the distinctive learning of M-DCTCs alongside L-DCTCs. The "M-Skipped DCT CNN" with the M-Skipped branch exploits various compositions of L-DCTCs and M-DCTCs to construct a rich and robust set of features. This architecture enables the interaction of lower-level features with higher-level features via shallowly convolved M-DCTCs. By adopting the M-Skipped architecture, the model avoids processing the full spectrum of DCT coefficients, reducing computational complexity and processing time, thus achieving optimisation. The M-Skipped DCT CNN serves as the baseline architecture for subsequent algorithms, incorporating the Adapt-DCT kernel and HyMod-ECA into this model. The experiments show that the characteristics of low-level M-DCTC are important in compressed domain FGVC.
- The development of a novel algorithm named 'Adaptive-DCT pointwise convolution kernel' (Adapt-DCT kernel) was implemented on top of the M-Skipped architecture. The technique adaptively weights the individual spatial and frequency bases of the DCT basis functions to form a pointwise convolution kernel. This technique enables the modification of spatial sizes and pruning of frequency bases of the DCT basis functions. It considers both spatial and frequency bases of DCT basis functions in forming a pointwise convolution kernel. This algorithm avoids composing conventional convolutional kernels which may contain redundant filters, which significantly reduces the model's footprint and enhances computational efficiency. The technique emphasizes the spatial context of DCT-based representations in kernels.

- The development of a 'Hybrid Modified Efficient Channel Attention' (HyMod-ECA) mechanism, which is used to investigate cross-DCT channel group interactions. This was achieved by applying fast 1D convolution with a large stride on the intermediate features. Unlike existing work that relies on cross-individual channel interactions, HyMod-ECA considers groupwise DCT attention weights that preserve the linearity and correspondence of the attention weights. The technique is novel for its study of groupwise DCT channel interactions. The integration of HyMod-ECA plays a crucial role in combining the relationships among features in compressed domain analytics. This mechanism is key to resource optimisation because it utilises a smaller number of attention weights compared to conventional methodologies. By employing fast 1D convolution with a larger stride size, the computational complexity is significantly reduced, resulting in fewer parameters and more efficient computing. This reduction in algorithmic complexity directly addresses the problem of resource inefficiency found in existing methods. Through the integration of HyMod-ECA, resource optimisation is achieved while enhancing the relationships among feature groups, which has a direct positive impact on both the overall computational efficiency and model performance.

The significance of this research covers the following notable knowledge:

- A broader implication of incorporating DCT coefficients beyond low frequency at greater network depths extends the understanding of fine-grained image representation and feature synthesis within CNN.
- The advancements in frequency domain kernel formulation, learning capabilities, and optimisation techniques by leveraging the DCT basis function enhances kernel analytics in frequency domain CNNs.
- The importance of interactions and correspondence among DCT channel sets and their corresponding weights along the learning phase of frequency domain CNN provides greater insights into feature relationships across various channel depths.

1.5 Scope of the Research

Figure 1.2 presents a brief overview of the scope of this research work. As there is a wide spectrum of topics under the deep learning umbrella, therefore the topic listings in Figure 1.2 are not necessarily exhaustive, and only the most relevant ones are depicted. Figure 1.2 covers the basic types of ANNs, their applications, a few optimisation methods, and the compressed domain technique used in this research work. The orange boxes indicate the specific scope chosen for the research underlying this thesis. The compressed domain can cover many aspects of implementation in a deep learning algorithm. This research focuses on using the DCT technique to optimise the input and form a convolutional kernel in CNN for small-scale FGVC.

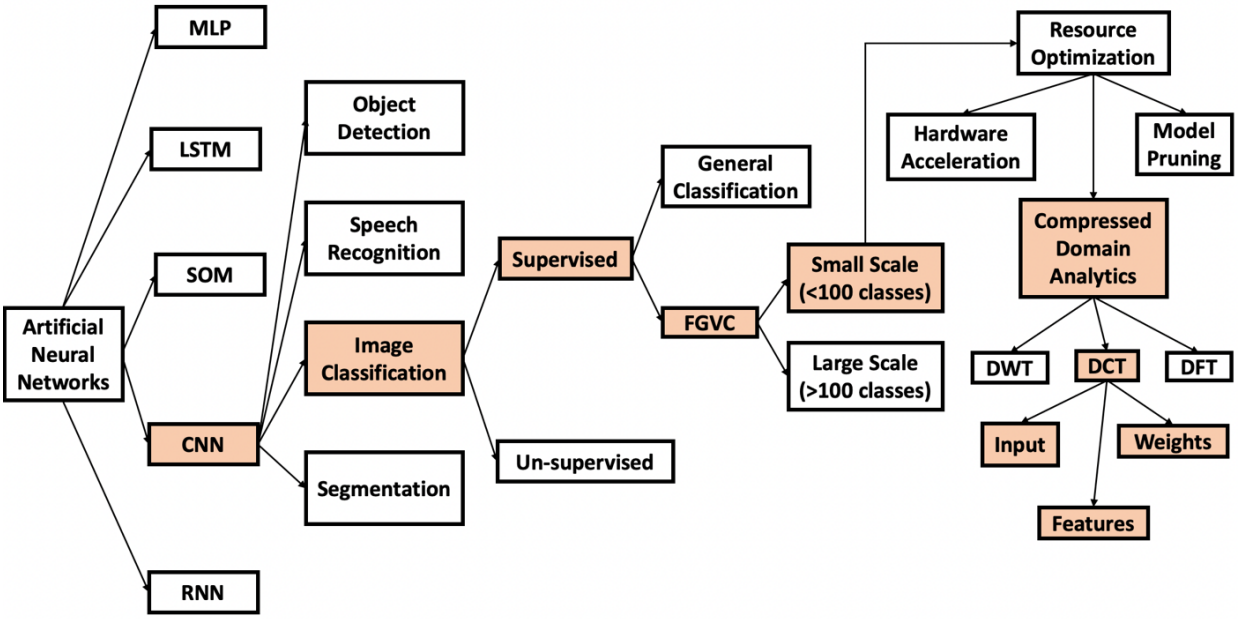


Figure 1.2: The scope of the research work.

Figure 1.3 portrays the high-level block diagram of the novel approaches proposed in this research. Specifically, the Adapt-DCT convolutional kernel is implemented in M-Skipped DCT-CNN, whereas the HyMod-ECA module is added toward the output of each convolutional block across the main network. The integration of all three approaches works seamlessly to address the issues of FGVC. The arrows in Figure 1.3 indicate the directional flow of information along the CNN. A JPEG image is partially decompressed to obtain three partitions of DCTCs as input, specifically L-DCTCs, M-DCTCs, and H-DCTCs. The L-DCTCs will flow through the main convolutional blocks whereby the adaptive weighted convolutional kernel and hybrid attention module are employed. The M-DCTCs will flow through another branch with fewer convolutional layers where the M-Skipped branch is employed. The concluding outputs from the two branches are concatenated and fed through the fully connected layers for classification.

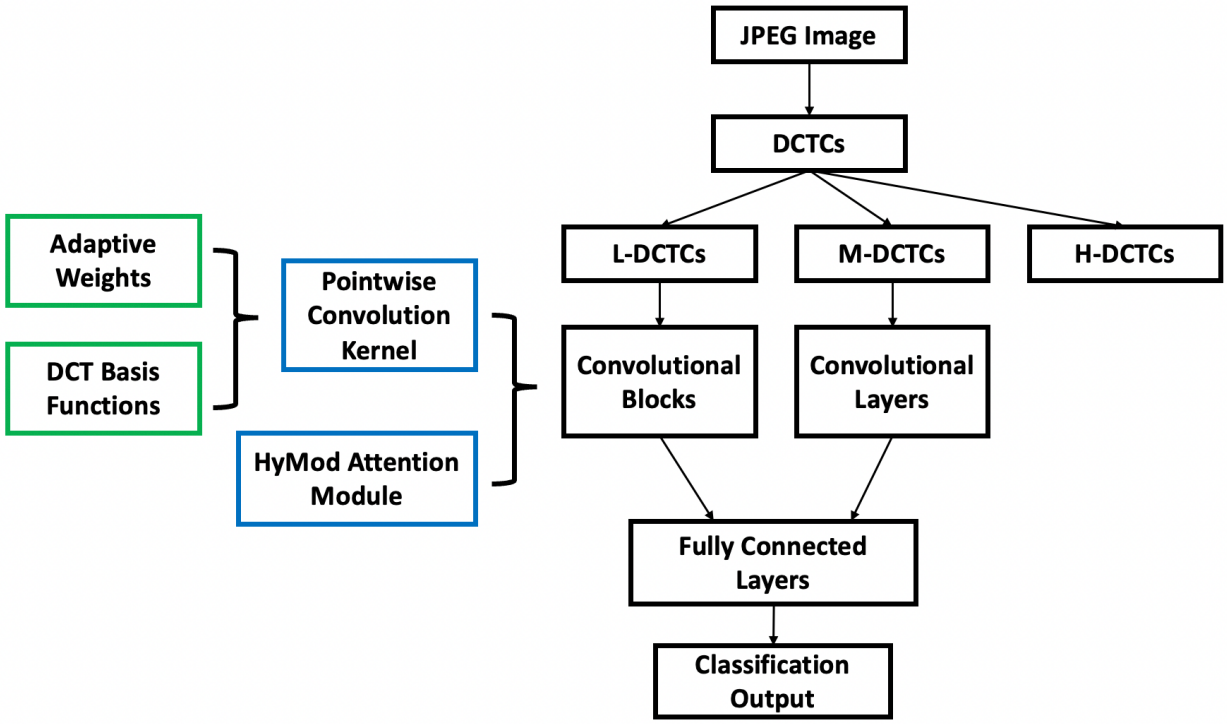


Figure 1.3: A high-level block diagram of the research work.

1.6 Thesis Structure

This thesis consists of 5 chapters and is organised as below:

- Chapter 1 – Introduction: This chapter thoroughly discusses the background of the deep learning approach and the importance of artificial intelligence. It provides the problem statements and the motivations for conducting this research work in the compressed domain FGVC. The aims and objectives, scope, and contribution of this research are outlined.
- Chapter 2 – Literature Review: This chapter presents an overview of CNNs, compression algorithms, the mathematical transformation of compressed domain analytics, compressed domain-related CNNs, FGVC-related modules, and attention mechanisms.
- Chapter 3 – Methodology: This chapter introduces the design and implementation of the skipping branch containing M-DCTC in a CNN. The M-Skipped branch is integrated on a fully pointwise VGG-16 network, called 'M-Skipped DCT CNN'. It also develops and demonstrates the adaptive weighting of DCT basis functions to form a pointwise convolution kernel. It is referred to as the 'Adapt-DCT Pointwise Convolutional Kernel'. The last section modifies the original ECA to form the novel hybrid modified ECA (HyMod-ECA) to exploit cross-DCT channel group interactions.
- Chapter 4 – Results and Discussion: A few variants of the M-Skipped DCT CNN were experimented with, specifically by skipping the M-DCTC over convolutional block 1, block 2, and block 3. Small-scale fine-grained datasets of the frequency domain were used to evaluate the performance of the M-Skipped DCT CNN. Several variations of the adaptive weighting scheme and optimisation were investigated. A few benchmark datasets were tested on the Adapt-DCT CNN, and major results were presented and discussed. Several small-scale FGVC datasets were used to compare the performance of the model with and without this attention module.
- Chapter 5 – Conclusion and Future Works: This chapter recaps the key findings and results of this research work. Several future works are suggested for further improvement of this study.

Chapter 2 Literature Review

2.1 Convolutional Neural Networks

The artificial neural network (ANN) is a type of neural network inspired by the biological neuron [47]. It mimics the working nature of the human brain. A single-layer perceptron probabilistic model [48] can perform decisions based on a set of weighted real value inputs. The weighted inputs are summed and passed through a Threshold Logic Unit (TLU), which can be referred to as an activation function [49] to produce a final output. A simple illustration of the perceptron is shown in Figure 2.1 [50].

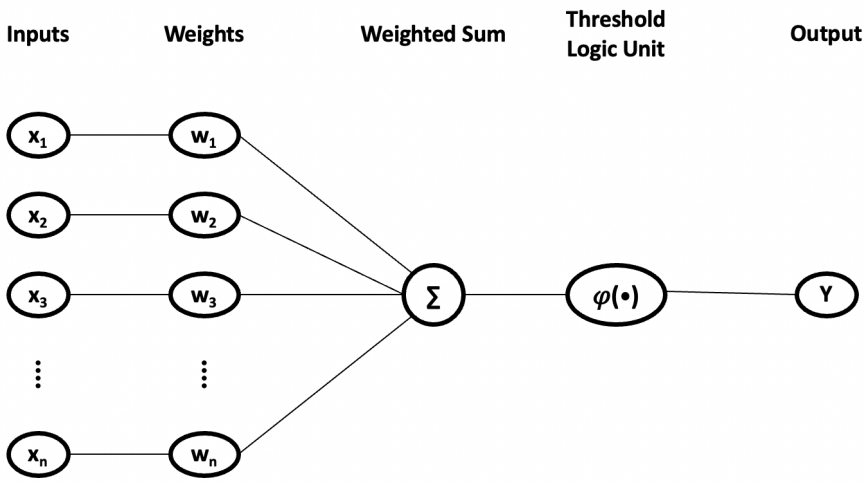


Figure 2.1: A perceptron probabilistic model [50].

By combining multiple hidden layers of perceptron between the input and output, a Multilayered Perceptron (MLP) is formed. The combination of non-linear decision functions in an MLP provides non-linear boundaries in relation to the given inputs, which can thus solve more complex problems. The MLP, when composed of many layers, is a type of Deep Neural Network (DNN) [51]. It is also known as a network with fully connected (FC) layers due to its architecture where all the neurons in the current hidden layer are connected to all neurons in the previous layer. Deep learning [52] is a type of computational algorithms that implements DNN to extract high-level features from data. The number of trainable parameters and the model size of an MLP increase according to the complexity of a problem. This can cause several drawbacks such as an extensive rise in computational requirements and proneness to overfitting [53].

A convolutional layer carries units with receptive fields which involve a localised part of the previous layer. The adaptive weights associated with these units are often collectively referred to as a filter. The down-sampling layer benefits CNN by scaling down the spatial dimension of the previous layer and easing the classification. The neocognitron [54] is the first CNN precursor to use weight weight-sharing technique across multiple units. It addressed the challenges of MLPs regarding optimising the deep learning algorithms with reduced parameters and computational complexity by introducing weight sharing.

With the increasing trend of digital media consumption over the years, images and videos are heavily stored and shared across online platforms. The internet access to media content has also increased especially for adolescents [55][55]. The applications and solutions of computer vision also dominate the market in areas such as automation and advanced manufacturing industries. The fundamental working mechanism of computer vision in deep learning algorithms relies on 2D-CNNs. Yann LeCun published the first work on a fully automated learning of CNN using backpropagation [56]. The backpropagation algorithm is used to optimise the weights by backpropagating the gradient of the loss function for each weight during training [57]. The model is named 'LeNet' which was used to learn handwritten digits. The work is governed by the general convolution theorem forming modern CNNs [58]. A typical CNN consists of a few individual components including a convolutional layer, pooling layer, and fully connected layer [59]. A simple CNN can be visualised in Figure 2.2

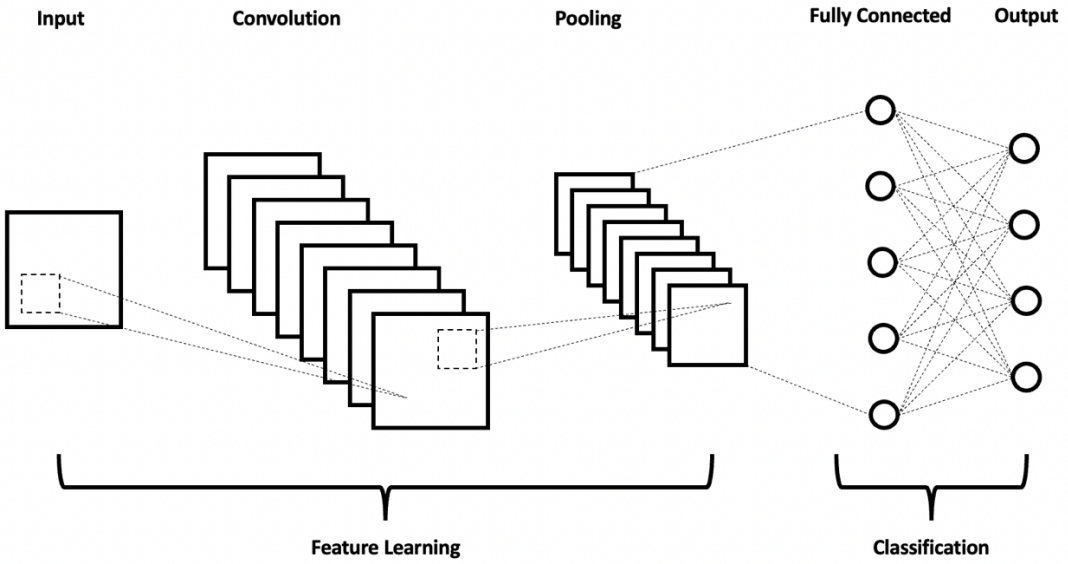


Figure 2.2: A simple illustration of the architecture of a CNN.

Convolution is the core operation of a CNN whereby a filter (kernel) is convolved across the input matrix. In 2D CNNs, the convolution is done by computing the dot product between the filter and the input entries. The kernel size and the corresponding stride size are determined before a convolutional process. Stride size is the number of shifting pixels when the filter matrix is convolved over the input matrix. The mathematical equation of 2D convolution [60] is denoted in Equation 2.1:

$$y[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot x[i-m, j-n] \qquad \text{Eq. 2.1}$$

Where 'h' is the filter matrix, 'x' is the input matrix, and 'y' is the output feature map. The convolutional layer serves as a feature extractor [4]. A small number of artificial neurons with learnable weights are grouped to form a filter in the convolutional layer. These filters are also known as kernels, and they are capable of learning local feature representations. The kernels are convolved across the input matrix during the forward pass to produce the corresponding output. The receptive field is a portion within the local input region where the kernel function is applied to produce the corresponding output feature. The respective outputs are arranged in a spatial configuration consistent with the previous layer to produce a feature map. A variety of feature maps are formed by convolving different kernels with the input matrix. The feature maps are stacked to produce the output of a convolutional layer which will be sent to an activation function to form the final activation maps.

Activation function [49] is a crucial part of a neural network as it serves non-linearity boundaries for the output from an artificial neuron. It also has a major influence on the computational efficiency and performance of a CNN [61]. Dhanasree [62] presented the significance of an activation function along multiple hidden layers in a network for data analytics. The paper concludes that the performance of a general neural network will be affected by the number of hidden layers and types of activation functions used.

The weighted sum of input sets in an artificial neuron delivers a single output proportional to the input sets. The drawback of the linear regression model is that the input and output sets are treated as a linear relationship, whereas the real-world problems exhibit non-linearity in nature. Fortunately, the nonlinear activation functions overcome these issues by adopting non-linearity. This property allows the stacking of multiple layers of artificial neurons to form a DNN which is desired for solving complicated tasks, such as image classification performed by a CNN [63]. The nonlinearity of activation functions is important since it allows the network to form complex non-linear mappings between input and output. Two popular activation functions are highlighted which are found in many DNNs today, namely the Sigmoid function and the Rectified Linear Unit function (ReLU). Their respective behaviours are shown in Figure 2.3 [64].

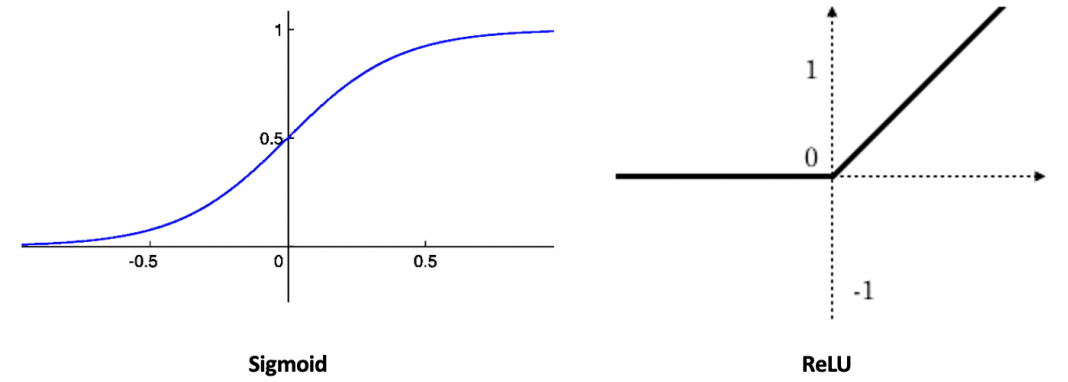


Figure 2.3: An illustration of the activation function and their corresponding response [64].

ReLU is widely used in CNNs as it is computationally efficient [65] to achieve non-linearity and it reduces the vanishing gradient problem. The sigmoid activation function produces a smooth and continuous non-linear output concerning the input; thus, it is useful in probability and prediction algorithms. The feature maps from a convolutional layer that goes through the activation function will form an array of activation maps. In other words, the convolutional layer learns filters that will lead to large node activations when specific features are detected in specific locations of the input. In the same convolutional layer, feature maps contain non-identical weights to detect different features at different input locations. The convolution is done between input images or feature maps with different filters to generate new feature maps, and the output is delivered through nonlinear activation functions to produce the final

activation maps. The pooling layer is also an important element of CNNs. It is used to perform a non-linear down-sampling of the activation maps. The objective of performing the pooling operation is to downsize the spatial dimension of the activation maps, to provide invariance properties, and to reduce computational complexity. 2D spatial max pooling is the commonly adopted method for pooling in CNNs. The example in Figure 2.4 portrays the concept of max pooling [66].

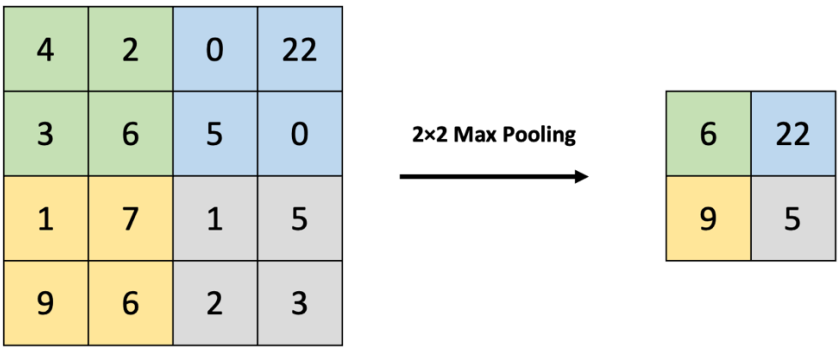


Figure 2.4: The operation of max-pooling in a CNN [66].

The convolutional and pooling layers are often stacked on top of each other progressively to form a convolutional block. The spatial dimension of the feature maps will shrink in size due to the stride operation and the spatial down-sampling while extending depth in the layer dimension. After the pooling layer in the last convolutional block, all the feature maps are flattened into 1D vectors. The 1D vectors are fed towards a few layers of fully connected layers and classification is conducted. For multiclass image classification, the SoftMax function is used for determining the probability class of a given input image. The SoftMax function is also used in other applications such as reinforcement learning in game theory [67].

A CNN is comprised of several convolutional blocks followed by FC layers. It extracts several simpler features primarily and gradually combines them in the later layers forming the required results. The principle of hierarchical feature extraction of the CNN is shown in Figure 2.5 [68]. It shows that the first stage of the neural network model recognises the edges of the representation. The latter stages further combine and map the previous features into a higher-level feature for classification.

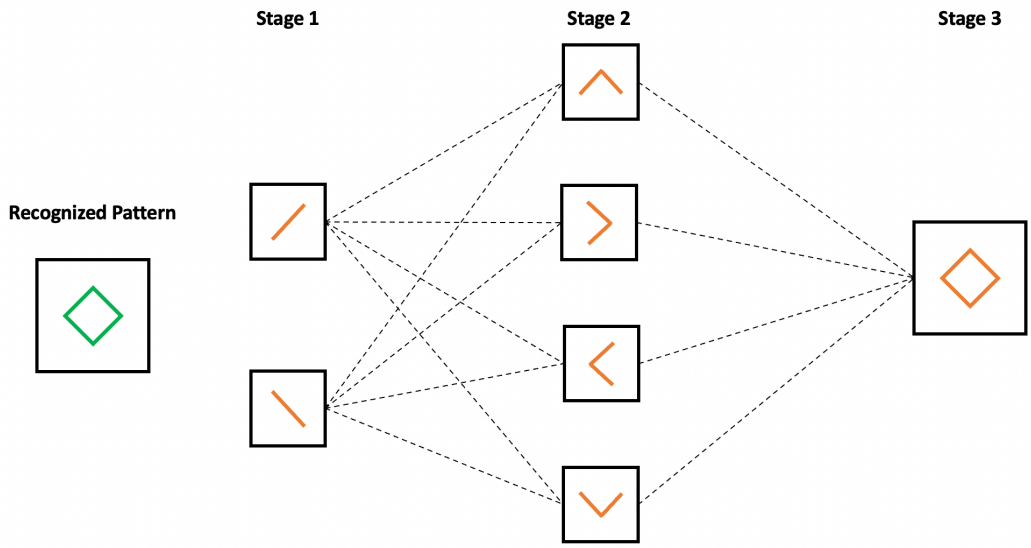


Figure 2.5: Principle of hierarchical feature extraction in a CNN [68].

With the advancement in hardware technologies for training CNNs over the years, more popular architectures [69] have been established and have brought astounding impact to modern computer vision recognition [70]. The motivation to design a better CNN [71] tends to lead to better performance and overcome several issues in CNN development. A few state-of-the-art CNN architectures are covered in this section to appreciate the insights and contributions that resolved the potential gaps in between.

LeNet-5 [72] was the first CNN to perform Optical Character Recognition (OCR) through 3 convolutional layers, 2 pooling layers, and 2 fully connected layers. The major downside of this architecture is that it is not deep enough. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [73] is a well-known computer vision classification benchmark on 1000 categories. The task incorporates object detection [74] and classification [75]. AlexNet as the improved version of LeNet-5 scored a top-5 error rate of 18.2% in the ILSVRC 2012. AlexNet was the first CNN to feature GPU learning for accelerated training. Since AlexNet contains up to 61 million trainable parameters, the DropOut layer [76] was used later to generalise the model to avoid overfitting. Some of the dominant CNN models after AlexNet are the VGG-nets [75], Residual Networks [8], Dense-net [77], and MobileNet [78]. They are widely used among state-of-the-art contemporary architectures.

The Visual Geometry Group (VGG) developed VGG-net [75] with deeper convolutions adapting the same filter size. It features a uniform CNN with a few feed-forward convolutional blocks. The most popular variation is the VGG-16 which is shown in Figure 2.6. It contains a total of 16 layers deep, whereby all the filters carry a spatial dimension of 3 × 3. The very deep VGG-16 consists of 138 million trainable parameters. It was found that very deep networks were subjected to the vanishing gradient problem when the VGG nets extended beyond 20 convolutional layers, which limits the network performance.

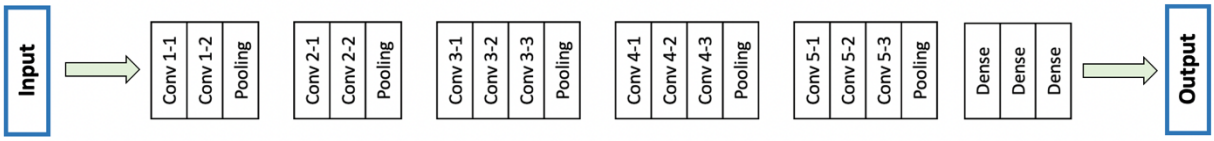


Figure 2.6: The architecture of a VGG-16 net [75].

Deep residual learning was introduced to resolve the vanishing gradient problem by inserting residual connections between convolutional layers. The residual network (ResNet) [8] is the first CNN adopting residual learning which won the ILSVRC in 2015 with a top-5 error rate of 3.57%. The ResNet-152 used in the competition consists of 152 convolutional layers, which is approximately 8 times deeper when compared with VGG-19. Figure 2.7 portrays the bottleneck of the 'identity shortcut connection' from one of the residual blocks. Batch normalisation is also added after every identity block to further reduce the overfitting and vanishing gradient issue.

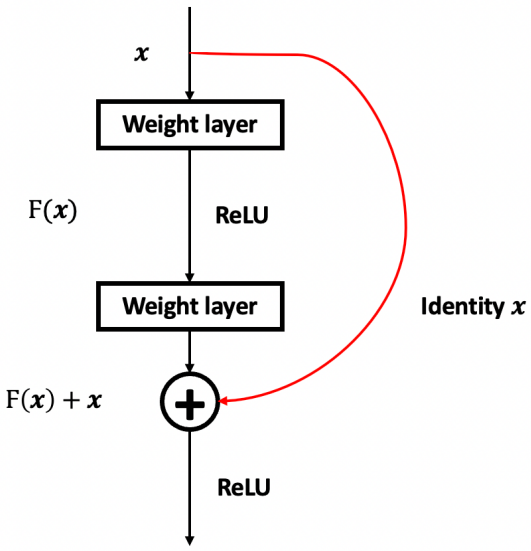


Figure 2.7: A 2-layer identity block from residual network [8].

DenseNet [77] also uses shortcut connections between layers by connecting all previous layers to the current layer. All feature maps of preceding layers are served as a concatenated input towards the current layer. It contains more shortcut connections between the convolutional layers. DenseNet can overcome the vanishing gradient problem while greatly reducing the number of trainable parameters. The Squeeze-Net [79] architecture was modified from Alex-Net [80] with 50 × fewer parameters while attaining similar performance. The key technique for reducing the number of parameters in SqueezeNet is by replacing 3 × 3 filters with 1 × 1 filters.

Mobile-Net [78] is a small CNN architecture that highlights depth-wise separable convolutions. Depth-wise convolution is a convolution process applied spatially across each of the channels independently, producing an output with the same number of channels as the input feature. The output is later applied with 1 × 1 filters. The 1x1 filter is referred to as a pointwise convolution filter. It requires less power to run and is specially designed to fit applications in mobile devices.

From the above literature, it can be recognised that by adopting a greater number of deep convolutional layers in CNN using residual learning, CNN can achieve a good performance. To optimise the CNN while retaining comparable performance, an individual can implement 1 × 1 filters onto the prior baseline models. The remaining sections of the literature will cover the background knowledge underlying the core ideas of this work.

2.2 Compressed Domain Algorithms and Neural Networks

2.2.1 Comparison Metrics of various Compression Algorithms

Compression is applied to an image to reduce redundancy. It is important to understand the fundamentals of the compression algorithms specifically between DFT, DCT, and DWT. A mathematical transformation (MT) is conducted in the compression process to transform the image into another domain that uses fewer bits for image representation. Most image compression is done by converting the image from a spatial to a frequency domain via Fourier-related transformation. The popular MT algorithms are Discrete Wavelet Transformation (DWT) and Discrete Cosine Transformation (DCT) [81].

[82] showed that DCT compression has an overall higher Mean Square Error (MSE) and Compression Ratio (CR) over DWT, and lower Peak Signal-to-Noise Ratio (PSNR) and Bits Per Pixel (BPP) over DWT. [81] used FFT, DCT, and DWT as the base functions for image transformation by applying four different compression percentages (10%, 30%, 50%, and 70%). The author concluded that DCT is a better MT compared with DWT and FFT in image compression. The author also indicated that most of the image quality is preserved when the compression rate is below 10% regardless of the image size. Another paper [83] did a comparative study on image compression between DCT and DWT using JPEG and PNG colour images. The image size stayed constant throughout the experiment and was processed with level-1 DWT. It was shown that image compression using DWT and Inverse DWT performs better than DCT in terms of image devaluation criteria, which were measured in MSE and PSNR. Some other studies [84][85] have shown that both DWT and DCT have comparable image quality preservation while DCT performs better than DWT through less computational complexity.

Fourier-related transformation (DFT, DCT) is efficient in extracting low-level (frequency) features of an image, while high-frequency components are usually quantised thus causing poor edge quality [86]. The transformation usually partitions the image into blocks and applies the transformation to each of the blocks. It was

reported in [87] that DCT performs efficiently at medium bit rates. JPEG uses sequential DCT such that image coding at high and medium bit rates is good. Low compression bit rates with high quantisation will increase the compression ratio, but it can lead to blocking artefacts and spatial decorrelation from the neighbouring blocks. DWT can preserve better image quality as there are no blocks or partitions involved. DWT exhibits multi-resolution properties. It can decompose a frequency signal using sinusoids of very fine resolution. DWT produces high-quality content at lower bit rates. Larger DWT basis functions or wavelet filters can cause blurring effects at image edges. At low compression rates, DWT produces a lower-quality image than DCT and requires a longer compression time. DWT and DCT produce comparable energy compaction properties. DCT has better performance than DWT in terms of lower BPP and higher CR. So, it is a fair trade-off whereby DCT does not match DWT in terms of image quality but outperforms DWT in terms of computational requirements.

The main difference between DCT and DWT coefficients lies in the high pass band. The DCT high pass band provides higher frequency resolution but lower spatial resolution. It has more frequency bands, but it is hard to recognise spatial information. In contrast, the wavelet sub-bands provide higher spatial resolution and lower frequency resolution as the number of sub-bands is smaller, but the spatial resolution is prevalent.

[88] compared DCT- and DWT-based compressed image algorithms to observe their transmission through wireless sensor networks (WSN). The results showed that DWT transform is better than DCT in terms of image quality and execution time, while DCT outperformed DWT in memory space consumption. [89] compared data compression in image processing for medical images. The experiment covered an MRI CT scan of an axial slice of the human brain with a grayscale image size of 512×512. The result showed that JPEG compression outperforms DWT in terms of PSNR and CR. JPEG has higher image quality at a lower CR than DWT. At higher CR, the quality of the JPEG image degrades due to blocking artefacts. Wavelets however provide good image quality at low bit rates due to their overlapping basis functions and the energy compaction property of wavelet transforms.

[90] presented a broad review of various methods used for feature extraction in facial expression recognition. By solely comparing DCT zigzag extraction and DWT feature extraction in [91], DCT achieved 80% accuracy while DWT achieved 81%. DCT

and DWT feature extraction performance is comparable in terms of feature quality preservation. Another paper [92] demonstrated facial recognition using compressed domain data with Principal Component Analysis (PCA) and Independent Component Analysis (ICA). The focus of this paper was to observe the normalised recognition rate (NRR). For PCA NRR comparison, DCT and DWT have a comparable performance level from 0.5 bit-per-pixel onwards. The results from ICA NRR showed that DCT outperforms DWT. [93] demonstrated iris recognition using block-based DWT and DCT for feature extraction. Block-based DWT outperformed conventional DWT and DCT as the false acceptance rate (FAR) and false rejection rate (FRR) are minimal. In [94], FFT, DCT, DWT, and Gabor Filter were used as feature extractors in fingerprint recognition. By comparing only DWT and DCT, both transformations achieved a comparable total success rate (TSR).

Table 2.1 establishes the summary of the discussion. From the literature reviewed, DWT outperforms DCT in image quality, but the difference is minimal. DWT is a type of image CODEC algorithm that is very sensitive to sudden changes in an image. Several scaling and bandpass filter-like processes are employed along the transform for signal coding. Although it is a lossless image compression format, DWT generally requires more effort compared to DCT. In terms of computational complexity, DCT is reported to be more efficient as compared to DWT. By extending the concluding factors towards compressed domain CNNs, DCT will be less expensive to implement. Therefore, DCT will be used in this research as it is intended to focus on compression gain over image quality.

Table 2.1: Comparison of different aspects of each compressed domain algorithms.

Aspects	**DCT**	**DWT**
Image quality	Slightly worse, negligible	Better
Preserve edge information	Slightly worse, negligible	Better
Compression gain	Higher	Lower
Computational complexity	Lower	Higher

2.2.2 Frequency Domain CNNs

The DCT approaches used in CNNs will be reviewed in this section. Table 2.2 summarises the core comparisons of the related works in section 2.2.2 and 2.2.3:

Table 2.2: Summary of literature on frequency domain and DCT-based CNNs.

Literature	Compression Algorithm	Methodology
[95]	DCT	Apply DCT towards the entire face image for recognition.
[96][97]	DCT	Combine DCTC with dynamic weighted discrimination power analysis (DPA) for identity recognition.
[98][99]	DFT	Compute convolution as the pointwise product in the Fourier Domain and reuse the transformed feature maps repeatedly.
[100]	DCT	Train CNN directly on the DCTC partition available within the JPEG codec without full decompression.
[25][101]	DCT	Integrate a DCT module on top of a CNN to train a robust model by accommodating specific distortions.
[102]	DCT	DCT-based spectral pooling replaces normal 2D-MaxPooling to resolve heavy information loss.
[23]	Inverse-DCT	The convolution filters are trained and saved in DCT coefficients. Inverse DCT is applied to the filters to get spatial filters during inference.
[26][103]	DCT	Network consisting of harmonic blocks (learning optimal combinations of spectral filters defined by DCT) and some optionally learned spatial convolution or FC layers.
[104]	Spatial domain, none used	Factorise spatial mixed feature maps at different frequencies and encourage inter- and intra-frequency communication.
[105]	Cosine basis function	Frequency parameters (amplitudes, frequencies, phases) of cosine basis are learned to produce spatial domain filter weights during training.

An early paper [96] demonstrated a statistical analysis by combining DCT coefficients and discrimination power analysis (DPA) for identity recognition. Dynamic weighted DPA (DWDPA) was proposed to enhance the DP of the selected DCTC without a pre-masking window. Mathieu et al. showed an early technique [98] to reduce the CNN's training and inference time by computing convolutions as pointwise products in the Fourier Domain. The transformed feature maps were reused repeatedly. The algorithm is based on the Convolution Theorem which states that circular convolutions in the spatial domain are equivalent to pointwise products in the Fourier domain. This method is efficient when the size of the convolution kernel is close to the input. The need to compute padding or periodic expansion is not easy to implement although it can save up computational power. Besides, a complex number can introduce complexity in calculations. The results in [98] proved that even though FFT is less efficient for a single convolution, with a deep network, the resource optimisation is much more noticeable. Even same memory bank was used to store the Fourier Domain feature maps for different layers, additional memory is still required. The inverse FFT is required to transform the data back to the spatial domain at the end of the network.

[99] opted out of the repeating process of forward and inverse Fourier transformations in the CNN. The training was conducted entirely in the Fourier frequency domain to speed up the entire process. From the Convolution Theorem as shown in Equation 2.2, '$\mathcal{F}$' denotes Fourier Transform, '*' denotes convolution and Θ represents the Hadamard Pointwise Product.

$$\mathcal{F}(\kappa * u) = \mathcal{F}(\kappa) \odot \mathcal{F}(u) \qquad \text{Eq. 2.2}$$

The operation carried out in the Fourier domain is less intuitive as the representation of the filters learned in the frequency domain could not be interpreted directly. FFT was only applied on the input images while the kernels were treated as Fourier filters. The Fourier-CNN (FCNN) in [99] can learn arbitrarily large spatial kernels through the Fourier domain. However, it was limited by the initial image size, as the spatial domain kernel size cannot be larger than the input image size. The pooling operation was carried out alongside the convolution to save up more computational costs. Fourier domain feature maps were distributed differently in FCNN. More features can be preserved in the Fourier domain as compared with the spatial domain. According to the 3D tensor shown in Figure 2.8, high-frequency data is often

concentrated towards the centre of the Fourier tensor while low frequency is towards the boundaries. The pooling operation which discards the high-frequency components is straightforward.

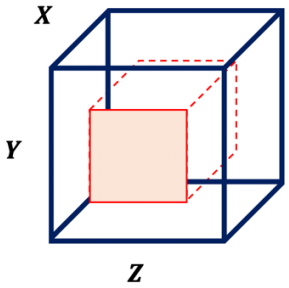


Figure 2.8: A pooling mechanism within a Fourier domain [99].

In the spatial domain, the kernels or feature maps will overload the memory if the input image size reaches a certain limit (2^9 in [99]). It can also cause parallel training to be infeasible, which can be addressed by reducing the batch size. This issue was addressed and resolved by training the model in the Fourier domain as it requires less memory when running in parallel. The computational complexity of spatial convolution increases exponentially as the image size increases, while Fourier convolution scales at a slower rate. The FCNN was assessed based on the computational time and accuracy of 3 datasets (MNIST, CIFAR-10, and Kaggle fundus images). In most of the experiments, FCNN exhibited a run-time advantage. The optimisation technique used in FCNN can be implemented in other network architectures to achieve SOTA accuracies with reduced memory footprint. The advantages exhibited by Fourier-related CNNs encourage more studies to be conducted on compressed domain CNNs.

A combination of spectral and spatial representations was incorporated by [106] in the convolutional block by effectively integrating features from both domains. The combination was done on a channel-shifting mechanism. It was found that this mechanism can reduce information loss as well as encourage model robustness. Since the spatial-spectral convolution learns features in both domains concurrently, it can detect the information from both domains. This includes local correlations in the spatial domain, global features of low frequency, and granularities of high frequency in the spectral domain. The channel division factor on spatial and spectral domains is based on a hyperparameter. The output feature maps from both domains are concatenated and passed through channel shifting before being fed into the next input. The shifted-spectral convolution was evaluated based on CIFAR and SVHN datasets on the VGG

network. Based on the experiments conducted in [106], with an optimal channel-division factor, this module can achieve a lower classification error rate with fewer parameters compared with conventional spatial domain convolutions.

In natural images, higher frequencies encode fine details while lower frequencies encode global structures. Octave Convolution (Oct-Conv) was introduced in [104] to factorise the feature maps at different frequencies to reduce spatial redundancy. Oct-Conv can be used as a direct replacement for vanilla convolutions. The information between adjacent locations is treated as a multi-frequency representation. The high and low-frequency features are stored and processed through different channels. In octave convolution, the aim is to obtain both low- and high-frequency maps (inter- and intra-frequency update) as the input towards the next feature maps. The up-sampling process is applied on the low-frequency maps when performing inter-frequency communication towards the high-frequency output, while the average pooling is imposed on the high-frequency maps. The Oct-Conv was tested on ImageNet using ResNet and DenseNet [104]. It was recognised that Oct-Conv can achieve 82.9% classification accuracy by only using 22.2 GFLOPS.

The pioneering introduction of FCNN marked a major leap in utilising compressed domain-related techniques to achieve computational optimisation without compromising SOTA accuracies. Techniques such as shifted-spectral convolution and octave convolution have emerged as noteworthy approaches for integrating information from various domains or capturing details at different frequency levels. These methodologies provide valuable insights into how combining signals from diverse domains can effectively reduce classification error rates and enhance optimisation. These algorithms were primarily tested on general classification datasets such as MNIST, CIFAR, SVHN, and ImageNet using popular CNN architectures like VGG, ResNet, and DenseNet. They prioritise parameter commonality while using classification error rates and computational complexity as key evaluation metrics. Despite their advantages in facilitating information exchange, these approaches often overlook the global context of features. Nonetheless, they serve a foundational background that motivates further exploration into the integration of different frequency coefficients at multiple scales.

2.2.3 Integration of DCT within CNNs

The authors in [100] presented a method to train the CNN directly on the DCT coefficients available within the JPEG codec without performing full decompression. The frequency information was fed directly into the modified ResNet-50 network. It was reported that the network was 1.77 × faster and more accurate on ImageNet than the original ResNet-50. It was reported in [100] that the 'Deconvolution RFA' network achieved the lowest top-5 error rate with an improved inference speed of 30%. The 'Late-Concat-RFA-Thinner' architecture from [100] has the closest baseline error rate with 77% faster inference. This study contributes to an insight into implementing the DCT coefficients directly in CNN for computer vision tasks. It also showed that it is vital to balance between the model's performance and computational speed.

[25] addressed the issues of modern CNNs which rely heavily on large datasets for training and CNNs' vulnerability to image quality degradation. 'Distortion Robust DCT-Net' was proposed in this research by integrating a DCT module on top of VGG-16. It improved the CNN's invariance by exposing it to more unseen images by fine-tuning the model to accommodate specific distortions. The forward 2D-DCT was performed on the input image, and the high-frequency components were discarded according to a uniformly distributed scale. The remaining coefficients were transformed back to RGB images via IDCT and fed into the network for training. The randomised selection of the threshold value for discarding high-frequency DCT coefficients of the input image was limited. A better scheme is required to produce a deterministic decision to select which coefficients to discard. Besides, the repeating forward and inverse DCT are inefficient. By discarding high-frequency components, noise, and distortion can be rejected. But at the same time, the method also discards edge details within the image. A blindly trained model with image data containing mostly low-frequency DCT coefficients can perform well in image classification with large variations. However, it may lead to potential gaps in fine-grained image classification.

The DCT operation was used in [101] for feature extraction. The authors incorporated the DCT process with convolutional layers after the non-linear activation function and before the pooling layer. The best result was obtained by performing DCT once towards the first convolutional layer to capture low-level information. The DCT process can be found at the 'conv1' layer in the CNN architecture in Figure 2.9.

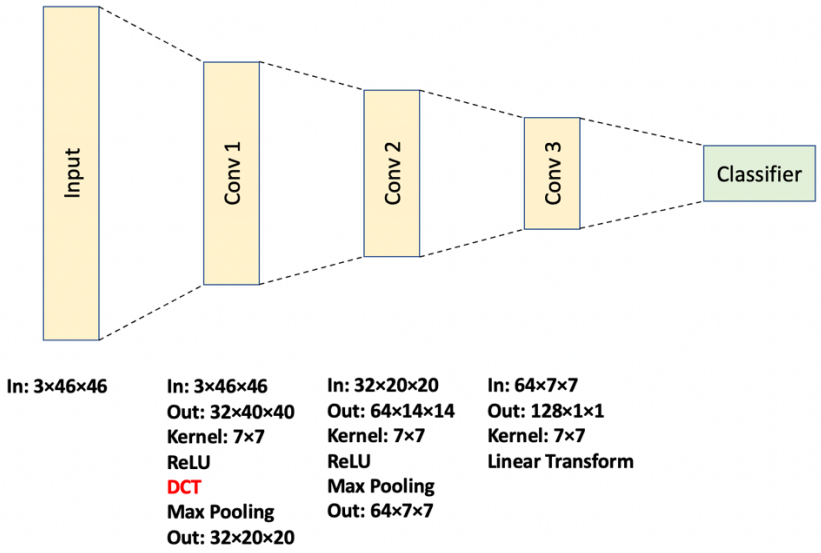


Figure 2.9: A DCT operation in the first convolutional layer [101].

The model was tested on pedestrian detection and object recognition. It was reported that feature extraction in the DCT domain resulted in an accuracy drop of 7.1% and an increase in convergence speed on CIFAR-10 when compared with conventional models. The weight matrices in the final model are sparser thus further optimisation can be achieved. Although the real-time DCT operation exhibited noticeable computational complexity, it helped the model to converge $5\times$ faster with fewer training epochs. A novel DCT-based pooling layer was developed in [102] to mitigate heavy information loss due to max-pooling in the spatial domain. It was proved that DCT-based pooling can preserve more information as compared to DFT due to its high energy compaction. The DCT-transformed matrices were embedded into the linear layer to accelerate the pooling process on GPU. Although the method can preserve features and fast DCT was used, the repeating forward and inverse DCT is still inefficient.

In [107], a DCT perceptron layer based on the DCT convolution theorem was developed. It was aimed to substitute 3×3 convolutional layers in CNN. The DCT perceptron module contains a scaling layer, a pointwise convolutional layer, and a trainable soft-thresholding function. The forward 2D-DCT was initially imposed on the intermediate feature maps before it was fed into the module. The output from the module is transformed back into spatial features via an inverse 2D-DCT. The scaling layer is similar to the concept of spatial domain convolution, where the trainable soft-thresholding layer can capture the positive and negative amplitudes of DCTC. The

trainable parameters in the soft-thresholding layer can be neglected for more parameter savings. This is because the PReLU activation function is sufficient to detect the positive and negative portions of the DCTCs. Although this method exhibited a reduced number of trainable parameters, the forward and inverse DCT processes are still undesirable. Thus, the process shall be addressed with an alternative approach.

Chęciński et al. [23] presented a method to perform IDCT on a set of trained parameters (frequency weights) in the context of spatial convolutional filters. The authors suggested a transformation over the trained parameters (small dimension of kernel weights) to produce the filter. This was done by coding the spatial convolution filters with trained DCT parameters to achieve a smaller model size. In other words, whenever any filters in the convolutional layer were used (inference), these filters were generated via IDCT from the frequency weights. This research showed that sufficiently rich spatial filters can have sparse frequency representations. The IDCT was performed independently for each filter. This can lead to a tremendous increase in computational complexity. Another recent work on using DCT in convolutional kernel can be found in [33].

Harmonic blocks produce features by learning optimal combinations of spectral filters defined by DCT [26]. It replaces the convolutional layers in conventional CNNs to construct a partial or fully harmonic CNN. Spectral features are generated by learning the linear correlations of the spectral filters defined by 2D-DCT. The spectral filter selects or eliminates image contents based on wavelength information. DCT is advantageous in frequency separation and energy compaction. The harmonic networks learn responses by combining 'window-based-DCT' with a small receptive field. Besides, it also learns to combine spectral coefficients at every layer to produce a fixed-size representation defined as the 'weighted sum of responses' of DCT filters.

Harmonic networks consist of a few harmonic blocks with some optionally learned spatial convolutions or FC layers. The harmonic blocks decompose the input features using 'window-based 2D-DCT'. The transformed signal is then combined with learned weights. The input features that have gone through spectral decomposition will produce a block-wise DCT representation. Thus, a new feature map for each channel is formed. The frequency coefficients of the transformed features are mapped along the layer dimension. Each set of feature maps in each layer represents a particular DCT basis function. The harmonic blocks can be treated as a special case of depth-wise

separable convolution with predefined spatial filters. It learns the relative importance of the feature extractors (DCT filter responses) at multiple layers. The spectral decomposition is computationally cheaper when compared with spatial convolution, but it up-samples the number of intermediate features hence increasing the respective memory requirements. This is particularly inefficient for memory management.

Figure 2.10(a) shows a harmonic block that performs channel-wise windowed DCT and recombines the responses with 1×1 convolution. Given that a set of filters carrying size 'K', number of input channels 'N', and output channels 'M', each box shows the corresponding operation, filter size (if applicable), and the number of output channels. The batch normalisation (BN) layer is optional. The compression of harmonic networks limits the visual spectrum of the harmonic blocks. It allows less processing of low-frequency DCT coefficients to reduce the number of parameters and operations. The coefficients were arranged following their importance in the triangle patterns as shown in Figure 2.10(b).

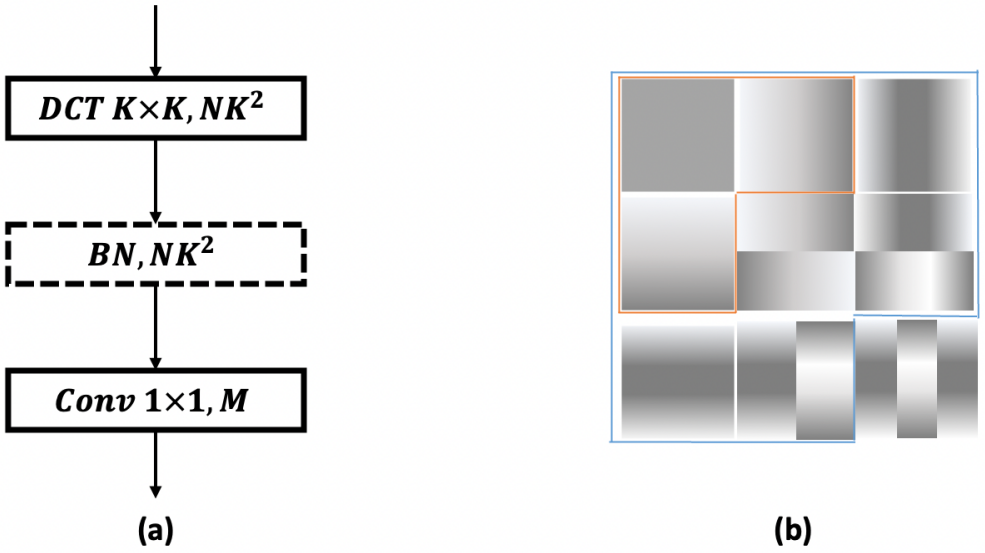


Figure 2.10: (a) Harmonics convolutional block. (b) Harmonics convolutional kernel visualisation [26].

In [26], a hyperparameter will determine the level of DCTC allowed for computation. Equation 2.3 shows that a feature map 'h' at depth 'l' is computed as a weighted linear combination of DCT coefficients across all input channels 'N'. $\varphi_{u,v}$ is the frequency selective DCT filter with a spatial size of K × K, '$**$' represents 2D convolution, and$W^{l}_{n,u,x}$ is the learned weight. The integration of harmonic blocks in CNN showed improvement in classification accuracy and parametric complexity. However, the

spectral decomposition up-samples the number of intermediate features (by a factor of K^2) hence increasing the memory requirements.

$$h^l = \sum_{n=0}^{N-1} \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} w_{n,u,v}^l \, \varphi_{u,v} ** h_n^{l-1} \qquad \text{Eq. 2.3}$$

The work was further studied by Ulicny et al. in [103] to address the need that CNNs have for large amounts of training data. The proposed harmonic blocks with DCT filters in CNNs are computationally cheaper and perform better than wavelet scattering networks (a network that works robustly with wavelet scattering data). [103] relates harmonic convolution with Modified-DCT (MDCT) to consider overlap in convolution. MDCT can reduce artefacts at the window edge. An improved model with higher efficiency was achieved which outperformed the wavelet scattering network.

The former work on harmonic blocks [26] replaced fully learned convolution with multidimensional input features. L1-normalized filters (batch normalization) were found useful for conserving DCTC features along the spatial-frequency spectrum. For parallel execution, extra memory was required to store the DCT filters' responses at each layer. It was realised that DCT and the linear combination operation could be integrated into a single operation. By factorising the filters as a linear combination of DCT basis functions, equivalent features can be obtained. With control over the filters, the author can approximate the signal with reduced computational complexity [103]. It was concluded that with only a small increment in multiply-add operations, it was possible to obtain similar performance and avoid overfitting in the model. [103] studies the effects of window functions in MDCT with a harmonic block by evaluating the classification error on 3 datasets (MNIST, CIFAR, and STL-10). Instead of computing a total DCT computation, using shifting to compute the convolution of filters along the feature maps might achieve a cheaper computation and the shifting weights can be learned dynamically.

Ciurana et al. [105] proposed a method to use the cosine basis function to generate filter weights. The convolutional layer is known as 'Cosine Convolution filter' (CBC). Frequency parameters of cosine bases were learned to produce spatial domain filter weights. The hybrid CBC only uses amplitudes, frequencies, and phases to represent the entire spatial filter. The CBC filter can be known as the frequency decomposition of a convolutional filter. The CBC filter can be derived from spatial

dimensions (x, y) and channel dimension (c). To produce a spatial filter, two cosine basis functions of 'spatial-product (S_P)' and 'spatial-direction (S_D)' were developed. The spatial-product CBC filter only defines the composition of two unique spatial harmonics. On the other hand, a spatial-direction CBC filter defines one harmonic in two different directions, 'w_X' and 'w_Y' which represent the unique frequency in both vertical and horizontal dimensions. The S_P resembles two harmonics in horizontal and vertical dimensions while the S_D uses a single harmonic to define a filter in different directions. The definitions of S_P and S_D are shown in Equations 2.4 and 2.5.

$$S_P(x, y) = \cos(w_x \cdot x + \phi_x) \cdot \cos(w_y \cdot y + \phi_y) \quad \text{Eq. 2.4}$$

$$S_D(x, y) = \cos(w_x \cdot x + w_y \cdot y + \emptyset) \quad \text{Eq. 2.5}$$

the feature dimension, 'feature-direct (F_D)' and 'feature-weight (F_W)' were proposed. Another cosine basis function was initialised, with 'A' representing the amplitude, 'w_C' representing the frequency, and 'c' representing the feature coordinate of the filter weights. It has higher compression as only three parameters (amplitude A, frequency $w_{C,}$ and phase φ_c) are required to define a channel. Both spatial and feature dimension basis functions can combine to create a CBC filter as shown in Equation 2.6. Spatial or feature dimensions can be any one of the two equations proposed earlier.

$$CBC(x, y, c) = \underbrace{S(x, y)}_{\text{Spatial dimensions}} \cdot \underbrace{F(c)}_{\text{Feature dimension}} \quad \text{Eq. 2.6}$$

The hybrid CBC layer combines conventional spatial filters and CBC filters. It is aimed to capture harmonics in the feature maps using a CBC filter while conventional filters are used for complex features. With a total of 'M' filters, 'α' determines the number of CBC filters versus spatial filters. When α is close to zero, a near-to conventional convolutional filter will be produced and vice versa. The hybrid CNC filter bank is shown in Figure 2.11. For a 1×1 filter, the S(x, y) was set to 1 in Equation 2.6 such that only F(c) was used.

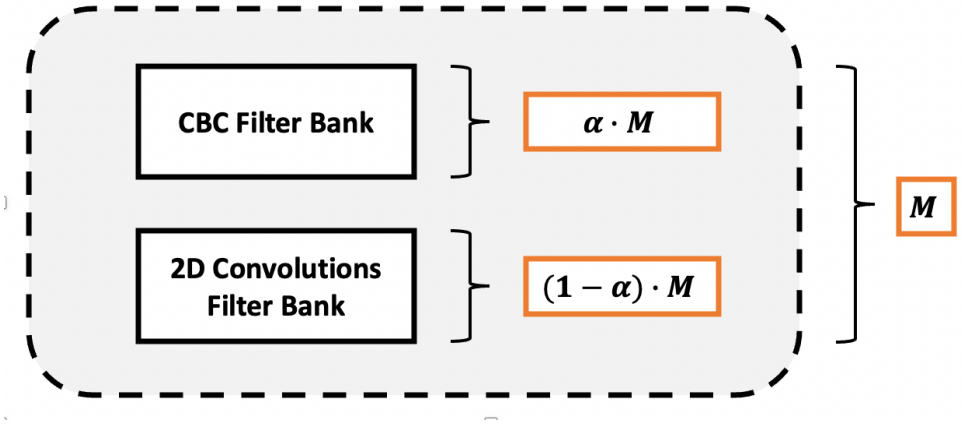


Figure 2.11: An illustration of the Hybrid CBC filter bank [105].

VGG-16 and ResNet-50 were employed to train datasets including CIFAR-10, CIFAR-100, and Monkeys (for fine-grained classification). The evaluation primarily centred on classification accuracy and convergence speed. Remarkably, the experiments revealed that models incorporating hybrid CBC layers exhibited superior performance compared to conventional networks, even when operating with fewer parameters (compression factors ranging between 1.2 and 2). This technique not only facilitates faster convergence speed during training but also allows for incremental increases in the spatial size of filters without incurring additional costs, all while reducing the overall number of parameters.

A notable advantage of the CBC filter is that receptive fields can be increased at no cost as the number of parameters does not change in the cosine basis function. Based on the spatial and feature weighting of the CBC filter, the parameters (A, w_X, w_Y, w_c, φ_x, φ_y, φ_c) forming $S(x,y)$ and $F(c)$ were trained to be consistent across each channel instance. Although fewer parameters were required to form a CBC filter, it is challenging to access and fine-tune each feature channel instance (c) at a particular direction of (x,y) thus limiting the potential of CBC to form the spatial kernel.

From the literature reviewed, the main metrics for evaluating the effectiveness of compressed domain techniques in CNNs typically revolve around classification accuracy or error rate. In addition, the secondary metrics to assess the training speed of compressed domain CNNs are often measured by convergence speed. Other relevant parameters include the number of trainable parameters and computational complexity. For studies targeting general classification tasks, common datasets like MNIST, CIFAR, NORB, and STL-10 are frequently used for small-scale evaluation, while ImageNet serves as a benchmark for SOTA assessments. Provided the focus of this research is on

FGVC, small-scale datasets like Monkeys can be found in the literature to offer a fair comparison. Popular networks such as VGG and ResNet are usually employed due to their ease of implementation and well-established performance. While equipment variations exist across studies, the use of GPU for this research is deemed acceptable. Because the focus of this work is on achieving compression gain and comparable accuracy in compressed domain FGVC, the specific equipment used becomes a secondary concern, as it does not impact the research outcomes or outcomes. Table 2.3 provides a summary of the key metrics, models, and commonly utilised datasets found in the majority of literature related to compressed domain CNNs.

Table 2.3: The quantitative evaluation metrics, data sources, and the model apparatus relating to the compressed domain technique in the corresponding literature.

Literature	**Metrics**	**Model**	**Datasets**
[98][99]	Classification Accuracy Computational Complexity	Basic CNN	MNIST CIFAR
[106]	Test Error Rate Number of Parameters	VGG	CIFAR SVHN
[104]	Classification Accuracy Computational Complexity	ResNet DenseNet	ImageNet
[100]	Classification Accuracy Inference Speed	ResNet	ImageNet
[101]	Classification Accuracy Convergence Speed	Basic CNN VGG ResNet	CIFAR STL-10
[26]	Classification Error Number of Parameters	ResNet	MNIST CIFAR NORB STL-10
[105]	Classification Accuracy Convergence Speed	VGG ResNet	CIFAR Monkeys

2.3 Attention Mechanism and Adaptive Techniques on Compressed Domain FGVC

Table 2.4 summarises the core comparisons of the related works in section 2.3 concerning FGVC.

Table 2.4: Summary of literature on attention mechanism and adaptive technique in compressed domain CNNs.

Literature	Domain	Methodology
[109]	Similarity learning	Use distance metrics to compute discriminative information between subtle features.
[31] [32]	Similarity learning	Compute bilinear functions.
[33]	Similarity learning	Compute channel and spatial correlations.
[34]	Information exchange	Use progressive training to acquire features at different stages to form higher-level features.
[35] [36]	Information exchange	Use dual pathway hierarchy to integrate low-level with high-level features.
[37]	Information exchange	Combine localisation and fine-grained feature learning.
[110][111]	Spatial attention	Produce attention mask to highlight features across spatial locations.
[112]	Channel attention	Highlight features across channel.
[113] [114] [115]	Channel-spatial attention	Study combination of both attention mechanisms.
[116]	Spatial attention	Study spatial interrelationships across semantic regions.
[117]	Channel attention	Study integration of global and local context.
[118]	DCT-related attention	Use DCT to build attention approximation function in NLP.
[119]	Multi-frequency attention	Compute inter- and intra-frequency interactions between low and high spatial features derived from DCT using octave convolution.
[120]	Frequency domain attention	Highlight intra-frequency to retain important frequencies using the frequency domain attention mechanism (FDAM).

2.3.1 Fine-Grained Visual Classification

FGVC is challenging due to the need to recognise subtle differences between classes. The core techniques involved in solving FGVC-related problems involve localising and differentiating subtle features via discriminative methods. Similarity learning and information exchange are reviewed in this section to gather insights that lead to research gaps as proposed in this thesis.

The bi-similarity network (BS-Net) [109] assumes that fine-grained images can be distinguished by two or more distance metrics of feature maps by comparing the query and support images. The two-distance metrics are the Euclidean distance and the cosine distance. This approach packs the information into smaller feature spaces with more discriminative feature maps. This is because the feature embedding module within the network is required to satisfy two different distance metrics. The authors in [31] addressed the issue of the equal treatment of features in FGVC for high-similarity subclasses by integrating bilinear CNN with a subclass similarity measurement. A weakly supervised localisation technique was then used to produce a bounding box around the object, and a fuzzing similarity matrix was used to compute interclass similarities. Finally, weighted triplet loss and classifier loss functions were used to compute the classification. Other similar papers such as [32] also used a bilinear technique. The work in [33] also exploits feature correlations by focusing on channel and spatial correlations without dimensionality increment. Similarity learning utilises domain optimisation and comparative measurements by computing relative metrics to improve FGVC. However, the learning can be optimised further by conducting FGVC in the frequency domain. This is because DCTC naturally provides a clear distinguishable boundary between varying features which can be found in basis functions.

Another method to ease FGVC is to effectively obtain and allow information exchange between global and local features. Progressive multi-granularity (PMG) training [34] focuses on identifying the most discriminative granularities by adopting a progressive training process, whereby granularities learned by earlier stages are used to facilitate the formulation of higher-level fine-grained details at later stages. A jigsaw puzzle generator is used on the fly during training to create images of different granularity levels. The authors in [35] showed how the integration of low-level information with high-level features using a dual pathway hierarchy, can result in a network that performs better with higher accuracy in locating discriminative regions. Another paper [36] extends the work beyond combining localised fine-grained features with global information by leveraging intra-class divergence and similarity. The literature showed that low-level features are critical in conjunction with higher-level features in FGVC for better performance. Identically, this provides valuable insights such that one may expect a combination of different levels of DCTC can ease FGVC in the frequency domain.

Discriminative part localisation and part-based fine-grained feature learning are typically solved independently when their inter-dependence could be exploited for improved performance. In other words, current methods focus on addressing these problems independently, while failing to realise that region detection and fine-grained feature learning are mutually correlated and thus can reinforce each other. In [37], the authors presented an attention object location module (AOLM) and an attention part proposal module (APPM) to forecast informative locations. The approach combines the idea of localising and learning fine-grained object parts in solving the FGVC problem.

The prior works on similarity learning and information exchange encourage a study in the frequency domain FGVC. It identifies a common gap in how one can refine the DCTC into different levels of frequency coefficients to obtain discriminative features and combine this information at different stages of the network. It also provides an opportunity to explore FGVC in the compressed domain, which is an area that is lacking research.

2.3.2 Attention Mechanisms in CNNs

Attention mechanisms [121] are a popular approach used in FGVC. Attention enables the network to selectively focus on salient regions while diminishing irrelevant parts. This mechanism enhances fine-grained feature discrimination by highlighting critical regions within an image. This is particularly important in FGVC due to highly localised and detailed features, which are usually not uniformly distributed across the entire image. The application of conventional CNNs to FGVC often leads to poor generalisation due to uniform feature processing. This can cause difficulty to the model in terms of capturing fine-grained distinctions. Attention mechanisms address this limitation by dynamically adjusting the weight of features based on their importance, allowing for more flexible and focused learning of discriminative patterns. While these mechanisms significantly improve FGVC performance, they also introduce additional parameters. This results in increased computational requirements, which can hinder model deployment. Therefore, it becomes imperative in this study to explore DCT-based attention mechanisms to achieve both enhanced feature discrimination and resource optimisation, aligning with the overall goals of this research. Attention helps the network focus on salient regions and diminish the irrelevant parts. The concepts of spatial and channel attention which were generally adopted in computer vision are initially reviewed. This is followed by a discussion on multi-frequency and frequency domain-related attention. It is essential to review the frequency domain attention solutions besides the spatial attention ones that were implemented in earlier works to compare and consolidate the focus of this thesis.

Spatial attention [110][111] focuses on producing an attention mask to amplify features across spatial locations, while channel attention [112] intensifies certain feature channels. Spatial attention was frequently used in semantic segmentation and object detection while channel attention was found in image classification. A combination of spatial and channel attention was demonstrated in [113] to generate a fusion of ‘what’ and ‘where’ processes, which is reminiscent of computational theories about the human visual cortex. In [114], some parts of the neural architecture were extended with channel-spatial attention to get spatial features on top of prior global features. The global features were included in channel pruning to enhance interdependence between channels within the same layer [115].

Spatial interrelationships among semantic regions were studied in [116] with the attention module to shift the focus of the model towards highly activated regions while diminishing focus in low response areas. [117] links global and local contexts via channel attention to weigh different contexts. Spatial and spectral details were fused adaptively in [122] in addition to input and delivered to a 3D CNN to produce attention masks that highlight spectral and spatial characteristics. From the literature reviewed, many of the recent attention techniques incorporate global and local discriminative features for different tasks.

A lightweight attention module [123] took channel and spatial attention into account simultaneously for better flexibility and adaptability. Cross-channel with spatial interrelationships via two parallel attention modules was used in [124] for COVID-19 detection. [118] employs DCT properties to build an attention approximation module for efficient computing in NLP. The approximation is retrieved by performing IDCT before feeding the resulting features to the SoftMax layer. A similar concept is shown in [125] by treating the global average pooling from channel attention as a special case of DCT function decomposition. Other similar works on spatial channel attention were also implemented on crowd counting [126], cross-channel loss computation for FGVC [127], dense attention network detection [128] to resolve the lack of mutual dependency of features, and wavelet attention on high-frequency and low-frequency information [129].

Multi-frequency attention network [119] generates low and high spatial frequency components from feature decomposition using 2D-DCT. Octave convolution was used to compute inter- and intra-frequency interactions for multi-frequency learning, while frequency channel attention was imposed on each of the high and low-frequency features. The self-supervised attention filtering and multi-scale features network (SA-MFN) [130] was used in FGVC. The multi-scale attention map generator extracts local attention maps and the relative global spatial relationship by providing a prediction score during scanning. To succeed in multi-scale feature learning, it was found that it is beneficial to relate the spatial relationship between global and local details to construct robust local features.

Frequency transformation can be modified to ease pointwise convolution. Classification. [38] proposed activations transformation by computing pointwise convolutions in a DCT-based frequency space. Each pointwise layer was replaced with

truncated DCT to achieve channel-frequency band pruning. The author developed a frequency contiguous mask (FC-Mask) with a new trainable parameter to learn the level of channel-frequency band pruning per layer and per channel during training. This method allows continuous dense pruning at neighbouring channels, losing only a few pruned coefficients while achieving efficiency. It was found that a minimal difference is acquired between this method and the unrestricted channel-coefficient pruning. The drawback of this approach is the need to compute repeating DCT and IDCT operations.

The frequency domain attention mechanism (FDAM) from [120] utilised intra-frequency to retain valuable frequencies and suppress the trivial ones. It removed redundant frequency channels through an understanding of frequency details between classes using a gate module. [38] utilised 1D CNNs to extract local-channel correlations on multi-level features while dual attention was used in [131] to combine features from different branches (locate local and global discriminative features) to produce multi-scale features. A similar method was adopted by [132] to solve inaccurate region localisation caused by discriminative regions spread with overlaying local receptive fields. Two attention modules with depth-wise separable convolutions [133] were used to capture channels and positional information for garbage image classification supported by a residual network for improved discriminative classification.

The literature above portrayed the significance of attention mechanisms from the spatial and channel perspective for image classification. The channel attention mechanism is considered in this thesis as this research work is intended to focus on frequency feature extraction. Besides, channel properties are prioritised over the spatial context in the compressed domain. It was found that the multi-frequency concept and global-local attention integration are vital techniques implemented in contemporary attention related networks. This is particularly compelling to be adopted in the frequency domain as DCT basis functions encode features at explicit frequencies. This work intends to integrate the multi-frequency and channel attention mechanisms in compressed domain FGVC to serve several advantages such as reduced parameters for compression gain. This can build a more robust learning framework that is less likely to overfit.

2.3.3 Adaptive DCT-based CNNs

Several studies have been presented in this section to understand the adaptive concept in various DL applications. [134] exploits spatial details obtained from the input at two scales. These features are then rectified according to their channel importance. It can be seen as an improved pointwise convolution incorporating spatial context information. A few groups of similar spatial-sized tensors were obtained from the spatial domain input according to varying channels. A single filter was used to convolve with these groups to produce a new group of intermediate features. The new group is stacked and further processed with a 'context refinement module (CRM)' to produce a stage 1 output. A stage 2 output is produced corresponding to stage 1 with another CRM to produce a group 2 output. The original pointwise convolved output is compiled with both groups generating a final output. Two 3×3 convolutions are employed in both stages to mimic a 5×5 convolution as it uses fewer parameters. In the ablation study, the author showed that a 2-stage process is the optimal variant of this method. The multiscale learning technique is similar to the method that is intended to be explored in this thesis.

Another paper [135] demonstrated the adaptive learning of frequency domain decomposition and transformation. Frequency attention features were continuously integrated aside from spatial clues for forgery detection. Soft masks with trainable parameters which replaced the fixed frequency transforms were utilised to decompose the frequency features with triplet loss. Inverse frequency transform (inverse DCT) was conducted after the frequency feature was applied with soft masks. The attention mechanism was applied to the resulting features and later fused with the original RGB branch. This technique is similar to the proposed technique, with the difference being that the whole adaptive kernel is trained on the frequency domain, hence no inverse transformation is required. [136] explores subject tracking using discriminative DCTC based on mean estimation of feature distributions.

[137] developed CropNets to acquire feature map patches by cropping at multiple stages. Skipped branches were used from intermediate features toward the network output with different loss functions. The network consisted of a 3-stage coarse-to-fine coordinate regression framework and facial landmark location was refined in each patch obtained from lower-level feature maps. Image super-resolution by processing features at different frequency scales effectively with coarse-to-fine matter was

presented in [138]. This method implemented a progressive frequency domain module (PFDM) and convolution-guided module (CGM) by incorporating frequency features with discriminative properties to compensate for detail loss. Hyperspectral image (HSI) classification [139] used multi-level feature extraction on the spectral-spatial attention model to resolve inhomogeneous pixels or inherent spectral correlation in HSI classification. A similar frequency domain HSI classification approach using a complex value wavelet network was presented in [140] to convert efficient features into frequency domain complex values to enhance the robustness and generalisation of CNNs. A gated RNN architecture forming a small hybrid model for HSI classification was also reported in [141] to fuse spectral and spatial information.

A special spectral rectified linear unit (SReLU) activation function was designed in [142] to perform computation in the frequency domain to avoid domain switching. The solution involves optimisation by using low-frequency coefficients [143] that adaptively fuse features acquired from FFT and low-pass filter weighting (spatial and channel) and aims to generate enhanced discrimination of image representations for better retrieval accuracy. [144] uses a DWT/IDWT layer to replace down-sampling operations in CNN to reduce aliasing effects thus improving noise and adversarial robustness.

Several papers have demonstrated adaptive methods in CNNs. However, research on the adaptive learning of DCT basis functions is currently lacking in DCT domain CNNs, not to mention the possibility of employing adaptive learning algorithms on DCT basis functions for FGVC. Hence, the need to investigate adaptive learning of DCT basis functions forming the frequency domain kernel in CNN for FGVC is strongly supported.

2.4 Fundamental Concepts

2.4.1 Basics of CNN and DCT

The convolution is a mathematical operation that combines two sequences to produce a third sequence by sliding a small filter over the input sequence. It intends to highlight certain patterns or features in that sequence. Often, the mathematical procedure induces multiplying the filter's values with the corresponding portion of the input sequence at each step and summing up the results. In deep learning, 2D convolution is widely used in CNNs for image recognition. A collection of hierarchy representations is formed when multiple deep convolutions are stacked together forming a CNN. Combining higher-level features, a CNN learns local representations efficiently by conducting 2D convolution by sliding a spatial filter (kernel) across the input image. The corresponding output feature maps determine the presence or absence of particular patterns in different spatial locations, forming spatially correlated relationships.

The convolution theorem states that the Fourier transform of a convolution is the pointwise product of the Fourier transform of each function. In other words, it is possible to conduct convolution in the frequency domain according to this theory. This was demonstrated in [28] and later in [101]. The frequency-related CNN is named FCNN. The strategy of implementing the Fourier domain in CNNs is by applying FFT towards the spatial kernel and feature maps, then applying a pointwise product across the two functions. By integrating FFT into CNNs, a frequency domain learning framework is formed, whereby a CNN is trained to capture frequency components instead of spatial contents. In the frequency domain CNN, a frequency representation of kernels and features separates the content into low and high bands, derived from a combination of consistent Fourier-related functions at specific frequencies. This allows multiscale analysis of features composed at different frequency ranges. An image or feature commonly has sparse representation, where several frequency components contribute heavily to the overall content. This is particularly beneficial when conducting convolution in the frequency domain, often shortening the lifecycle of CNN development. Table 2.5 tabulates the basic differences of spatial and frequency domain CNN.

Table 2.5: Comparison of spatial and frequency domain CNN.

Aspects	**Spatial domain CNN**	**Frequency domain CNN**
Theory	Highlight spatial features across the input sequence	Capture frequency components at a specific range
Working strategy	Sliding filter across entire input feature	The dot product between two Fourier transformed functions, i.e. kernel and feature
Significance	Emphasise spatial correlated relationships	Emphasise multiscale frequency compositions

Real-world data, particularly images, often exhibit real-valued and correlated structures in their spatial context. A compressed JPEG image is formed by conducting forward 2D-DCT towards an RGB image, whereby an intermediate form of frequency image representation can be cultivated. The frequency domain image representation is constructed by DCTC. Although DCT is part of the Fourier transform, it focuses on real-valued signals only. While the Fourier transform operates on complex numbers, the DCT produces a set of real coefficients. This property is advantageous when dealing with images that inherently possess real-valued attributes.

Fourier-related CNNs are not widely found in recent CNNs as FFT consists of imaginary parts which can affect the interpretability of a CNN. Moreover, the substitution of FFT and a pointwise product in the convolution layer requires the repeated computation of forward and inverse FFT. The inverse FFT is required to transform the data back to the spatial domain at the end of the network. As such, the implementation of frequency domain CNN by consuming image DCTCs is a more practical option. A DCT-related CNN takes DCTCs as input instead of raw pixel values. It learns features directly from DCTCs to capture information relevant to compression and frequency-related characteristics. The key benefit of DCT over the Fourier Transform is its strategy of handling content using real values. Table 2.6 compares the differences between Fourier- and DCT-related CNN.

Table 2.6: Comparison between Fourier- and DCT-based CNN.

Aspects	**Fourier CNN**	**DCT CNN**
Transform Domain	Complex	Real
Feature representation	Global features	Local features
Computational complexity	Higher	Lower

Besides taking DCTCs into a DCT-related CNN, the DCT technique is also widely applied in many parts across the CNN, such as pooling layers, kernels, and feature maps. Such a process intends to achieve different objectives. In this thesis, the DCT strategy is adopted and modified to be implemented in the kernel and feature maps for its advantages over other frequency strategies, which include a broader feature representation in higher-level frequency bands and kernel analytics at different depths of a CNN. In essence, it is intended to establish the foundational concept of DCT in CNN for FGVC. Table 2.7 summarises the key areas where a DCT-based strategy is implemented along a CNN.

Table 2.7: Implementation of DCT-based strategy and its significance in a CNN.

Aspects	**Significance**
DCT in pooling layers	Resolve heavy information loss
DCT in kernels	Kernels formed by DCTC are sparse hence reducing model size
DCT in feature maps	Convolutional kernels learn frequency compositions instead of local representations

2.4.2 Attention Mechanism involving Frequency Properties

Deep learning attention mechanisms are inspired by human visual attention. They allow a neural network algorithm to focus on specific sections of the sequence rather than the entire input series. The working strategy of the attention mechanism usually involves assigning dynamic weights to different parts of the input sequence, thus allowing the model to pay more attention to relevant information. An early motivation behind attention mechanisms lies in the desire to enhance the ability of neural networks to process and understand sequential data, which is crucial as it improves the interpretability on specific instances within a series of data.

The attention mechanism is particularly useful in neural networks such as RNN and CNN. Conventionally, RNNs such as LSTM and GRU suffer from vanishing gradient and exploding gradient issues. In LSTM, attention weights are associated with the hidden states of the encoder to reflect the relative importance of each state during the decoding process. Similarly, attention weights are computed based on the relevance between each hidden state with the current decoding step in GRU. The weighted sum of the encoder states based on the attention weights is then used to generate a prediction. In both cases, the attention mechanism helps the model to handle long-range dependencies more effectively. It has proven to improve the performance of applications such as NLP and prediction in the early stage of the deep learning NLP renaissance.

Attention mechanisms have also found their way into CNNs in the application of computer vision, especially FGVC. Typically, FGVC contains hard-to-classify objects where the intra-class difference is minimal. The use of attention in CNNs aimed to improve the network's ability to focus on relevant fine-grained object regions for effective feature extraction. The basic approach for integrating attention into CNNs is by applying the attention module at various stages across the network, including before or after convolutional layers. This involves optimising learnable parameters to highlight the most prevalent features.

In DCT-related CNNs, the feature channel which consists of frequency-related information is usually derived from hierarchy-convolved DCTC. By applying attention to feature channels in a DCT-related CNN, the attention module focuses on capturing specific frequency components relevant to the task instead of regional features relating to the spatial context. The working strategy of implementing channel attention in DCT-related CNNs is similar to the one in spatial domain CNNs. This is done by assigning learnable parameters across the convolutional layers. In this thesis, the attention weights are applied to the frequency channels after the convolutional layer. It is intended to ease the model to focus on more significant frequency components relating to the fine-grained features of FGVC. Attention to frequency analytics provides several benefits over the spatial domain such as improved model robustness.

In Table 2.8, a summary of the highlights, significance, and outcome between the attention mechanism applied on spatial domain CNNs and DCT-related CNNs is established.

Table 2.8: Differences between attention mechanism on a spatial- versus DCT-based CNN.

Aspects	**Attention on spatial domain CNN**	**Attention on DCT-related CNN**
Highlights	Focus on regional features relating to spatial context	Focus on frequency-related components
Significance	Relevant parts derived from hierarchy-convolved feature maps of spatial context	Relevant information derived from hierarchically convolved representations relating to DCTC
Outcome	Improved regional feature discrimination involves spatial context	Improved frequency component discrimination and interpretability

2.5 Summary

From the literature reviewed, it is prevalent that network performance can be improved through the right combination of features at different scales. While similarity learning and information exchange provide insights into feature communication in the spatial domain, the exploration of learning different frequency coefficients is notably absent in DCT-related CNNs. It is particularly essential to consider multiple ranges of frequency coefficients in addressing a gap in compressed domain FGVC. While pointwise convolutional filters in residual learning prove efficient and robust in DCT-related CNNs, their limitation lies in the inability to incorporate spatial context on top of the frequency features. This gap prompts an interesting exploration of adopting DCT to form the pointwise convolutional filters that encapsulate both spatial and frequency contexts. Additionally, an attention mechanism has proven its importance in enhancing feature extraction in FGVC, yet the lack of correspondence between the highlighted signals calls for a deeper understanding. Henceforth, it is imperative to study the relationship between the focused frequency components produced by an attention mechanism. The identified research gaps from the literature review can be summarised as follows:

- The limitation of discriminative learning in DCT-related CNNs, especially involving frequencies beyond L-DCTC.
- The absence of spatial and frequency analytics in the pointwise convolutional kernel within DCT-related CNNs.
- The need for interpretability regarding the correspondence and relationship between highlighted signals produced by an attention mechanism.

In essence, this study explores the potential framework of compressed domain CNNs to bridge these research gaps in frequency domain-related CNNs. Furthermore, it aims to underscore the importance of striking a balance between frequency analytics and compression gain. Resource optimisation is a critical focus in deep learning research, aimed at balancing model performance with the efficient use of computational resources. It not only reduces the financial and environmental costs associated with model training and deployment but also democratises access to advanced AI technologies. By making optimised models more accessible, industries and researchers with limited resources can benefit from deep learning advancements. As deep learning scales across diverse applications, resource efficiency becomes essential to ensure sustainability and practicality without compromising model effectiveness. Additionally, resource-optimised models are crucial for operating in environments with limited computational power, such as mobile devices or embedded systems, while still managing complex tasks like fine-grained classification. Achieving this scalability allows the models to be adaptable for both large-scale systems and constrained environments. By reducing the number of parameters, optimising computational operations, and integrating efficient mechanisms like DCT-based strategies, it is possible to develop models that maintain or improve performance while significantly enhancing computational efficiency.

Chapter 3 Methodology

3.1 Overview

Fine-grained visual classification (FGVC) [145] focuses on using deep learning models to classify hard-to-distinguish object classes such as species of animals or identifying certain models of vehicles. FGVC domain typically exhibits low inter-class variance and high intra-class variance. The major challenges include locating fine-grained object parts and emphasising the learning of those fine-grained features. Fine-grained feature localisation is more challenging when the pose of an object changes. Most FGVC solutions follow the process of finding foreground objects or parts (where) to extract discriminative features (what).

In recent years, relatively little research has focused on FGVC. Recent literature often emphasises the localising [146] and attention [33] of the most discriminative features within the fine-grained images [147]. Most of the discriminative methods tend to employ multiple models (two models for bi-similarity networks, and three models for triplet loss function). Although the models use shared weights, network complexity is still a potential issue. Several papers have been working on metric learning [148][149] or similarity learning [109][150][151] for FGVC. The few-shot learning based on metric learning has become more popular in FGVC [33] for its ability to address the problems of differentiating FGVC features from small datasets. Through preserving the relationships embedded in feature space, it allows the model to generalise well and only requires very little instances per class for training. Although the classification performance is on par with benchmarks, the existing methods do not fully address fine-grained feature learning issues.

In FGVC, spatial information is found to be less informative compared to order-less descriptors. Regardless of the learning techniques or modules used, the properties of the input domain and the corresponding feature representation could be a factor hindering the relative ease of fine-grained feature learning. Varying the input domain or feature map representation can potentially ease the network to classify fine-grained images. This is because, with different input domains, the model is encouraged to learn different feature representations, thereby fostering the emergence of different properties with richer higher-level representations.

Motivated by the proven benefits of frequency domain representation, this thesis starts with formulating input representation in the frequency domain to cover the research gap on medium frequencies. In the spatial domain of natural images, higher frequencies encode fine details while lower frequencies encode coarse features. The usage of DCT coefficients for representing images allows higher-level feature representations to be learned at earlier parts of a network. Moreover, DCT excels in energy compaction and frequency separation. Thus, it can ease a network to learn a compact and robust feature when it uses DCTC inputs as compared to RGB.

Inspired by the concept of metric learning on emphasising subtle feature discrimination yet sustaining variability between features, this thesis initially focuses on exploiting low and medium-frequency DCT coefficients via branching architectures in the FGVC domain. Learning the features individually allows the network to extract discriminative content, while combining these features at the latter part of the network bridges the two approaches. More explicitly, it is proposed that by combining the features extracted from low DCTC (L-DCTC) with medium DCTC (M-DCTC), the discriminative feature learning procedure can be significantly improved whilst enhancing the conventional approach that utilises multiple models for discriminative classification. Henceforth, it is hypothesised that the integration of a skipping connection that carries M-DCTC with a deeply convolved L-DCTC for FGVC will outperform the standard solution that does not integrate medium frequency coefficients. This is because the classifier can take advantage of fine-grained features through shallowly convolved M-DCTC.

CNNs are known for their capability to capture local correlations in a feature map. Existing works on compressed domain CNNs have focused on convolving either the input [101] or the feature maps [105][152][23] in the frequency domain. Frequently, the compressed domain CNN involves the forward and/or inverse 2D-DCT (DCT/DCT^{-1}) to obtain the spatial or frequency information. Figure 3.1 shows the 2D-DCT (DCT/DCT^{-1}) being applied in one or more processes along the compressed domain CNN including the input, filter, and output.

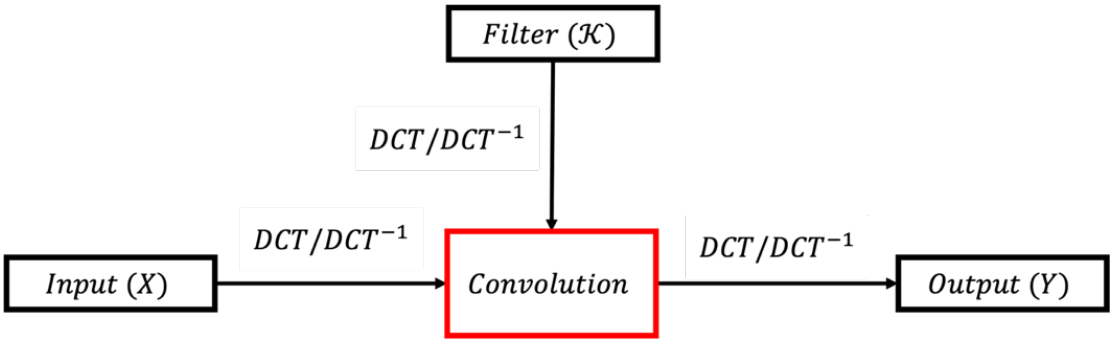


Figure 3.1: Forward and inverse 2D-DCT implemented along the compressed domain CNN.

Figure 3.2 compares the major works on compressed domain CNN. Figure 3.2(a) shows the conventional CNN uses RGB kernel ($\mathcal{K}_{\text{RGB}}$) to convolve with RGB input features (X_{RGB}) to obtain a RGB output feature map (Y_{RGB}). Figure 3.2(b) shows the inverse 2D-DCT (DCT^{-1}) applied on the DCT kernel ($\mathcal{K}_{\text{DCT}}$) during the forward pass to generate the spatial domain kernel ($\mathcal{K}_{\text{RGB}}$) to convolve with the spatial domain features. Figure 3.2(c) shows the forward 2D-DCT (DCT) is used to convert the RGB feature map into the DCT domain before forwarding to the network, and the convolving output is converted back to the spatial domain via inverse 2D-DCT. The spatial kernel in Figure 3.2(d) is produced from cosine bases whereby the trainable parameters are the frequency parameters (amplitudes, frequency, phases).

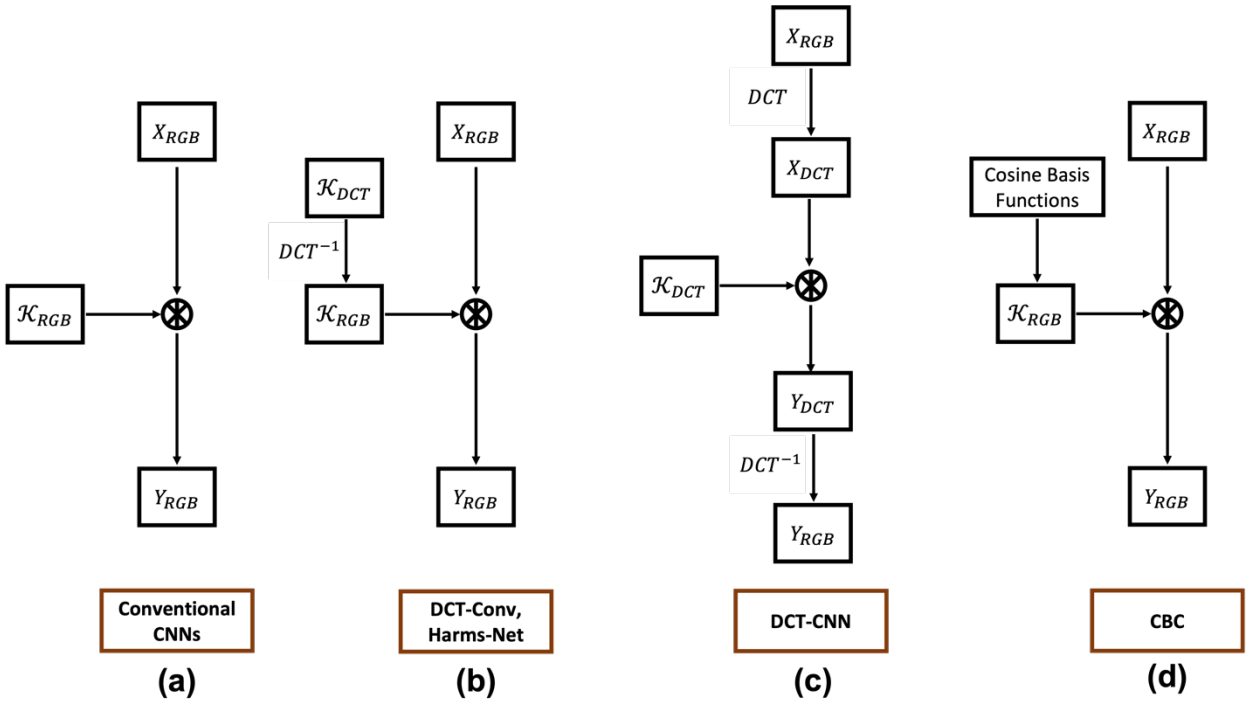


Figure 3.2: Comparison of different DCT-based approaches for compressed domain CNN.

The DCT technique is commonly found in two parts across compressed domain CNNs, particularly the feature map and the convolution kernel. In this thesis, the frequency domain input and feature maps are referred to as the 'DCT input' and the 'DCT feature maps'. The DCT input is obtained by performing forward 2D-DCT-II on the RGB image (commonly with an 8×8 block-wise partition), while the DCT feature maps are the collective convolutional result of the DCT input. It is well known that most compressed domain networks that accept DCT input are constructed using a fully pointwise convolutional block. Pointwise convolution can handle complex feature transformations due to its non-linearities when correctly composed. Although pointwise convolution ignores explicit spatial correlations, it provides advantages in terms of feature channel representations.

However, solely using the pointwise convolution in compressed domain CNN brings limitations in terms of spatial analytics. This is because pointwise convolution falls short of recognising local correlation patterns consisting of spatial context. The spatial combination of DCT basis functions from neighbouring coordinates in a trivial manner is not feasible, although each basis function represents a specific set of spatial correlations. Moreover, the DCT input and the DCT feature maps are the results of block-wise partition pixels, causing decorrelation and block-wise partition disjoints, therefore the learning process is biased towards the feature channel. Thus, it is crucial to identify and leverage the common interval between spatial and frequency context to enhance the robustness of kernel analytics in the compressed domain.

In compressed domain analytics, the conversion between spatial and frequency domains occurs through 2D-DCT, where the DCT basis functions serve as the intermediary for this process. The basis functions govern the fundamental properties and behaviour of the corresponding frequency representation. Particularly, it expresses the representation by incorporating spatial and frequency components. To enhance the robustness of pointwise convolutional kernels for spatial analytics, it is mandatory to consider the modification of basis functions during the intermediate phase of forming the pointwise convolutional kernel. Recognising the importance of DCT basis functions, this research advocates for an innovative approach: the modification of these basis functions to compose a robust pointwise convolutional kernel capable of spatial and frequency analytics. To address this challenge, the proposed technique involves assigning weights to each spatial and frequency base of the DCT-BF during the kernel formulation. This novel approach is called 'Adaptive DCT (Adapt-DCT) Pointwise

Convolution'. It is implemented within M-Skipped DCT-CNN and aims to address the inherent limitations of conventional pointwise convolution in handling both frequency and spatial aspects effectively. The adaptive nature of the Adapt-DCT kernel allows the following distinguishable differences over the prior pointwise convolutional kernels being investigated: (1) a subset of DCT basis functions is emphasized and selected to form the final pointwise convolution kernel; (2) the adaptive kernel ($\mathcal{K}_{\mathrm{TS}}$) serves as trainable weights to multiply (element-wise) with the modified DCT basis functions; (3) the original coefficients for the DCT basis function are no longer available since they are replaced by $\mathcal{K}_{\mathrm{TS}}$.

Attention mechanism has found its way into computer vision [121] for its capability to accentuate discriminative characteristics in various visual recognition tasks, especially fine-grained visual classification (FGVC) [153]. Several contemporary attention-related methods cover channel attention and spatial attention [114][124][125][154][155]. Channel attention is good for image classification while spatial attention is advantageous for segmentation and object detection. Other popular attention-related methods are convolutional block attention module (CBAM) [113][156] and Squeeze and Excitation Net (SE-Net) [112].

The implementation of FGVC in the DCT domain is challenging due to the subtle details encoded in the DCTCs. DCT is the key ingredient in frequency domain-related convolution neural networks (CNN) for its advantage of packing features with a high level of compactness to achieve compression. However, in the DCT domain, most of the DCTC of frequency bases are encoded into the channel dimension. Since DCT naturally destroys the neighbouring spatial correlations of a spatial context, a spatial attention mechanism with convolving or masking properties is less effective in locating fine-grained features. Most of the existing works as reviewed in Chapter 2 relating to DCT attention only deal with spatial domain input and networks, such as [118] uses DCT in self-attention for natural language processing while [125] treats the normal channel attention as the special case of 2D-DCT.

Henceforth, it is arguably appropriate to focus on modifying channel attention as channel properties outweigh spatial properties in the DCT domain CNN. Implementation of direct channel attention towards DCT feature maps not only increases trainable parameters but also further causes non-linear projection between features and attention weights in SE-Net. Therefore, this thesis focuses on exploring the channel attention mechanism in a slightly different approach: Can the channel

attention module in the DCT domain be modified to capture the interaction of discriminative DCT-related properties effectively?

Since the introduction of M-Skipped-DCT CNN, the channel dimension of the DCT feature maps is viewed as a sequential-ordered tensor consisting of low, medium, and high DCT coefficients (LMH-DCTC). Instead of using channel attention to capture depth-wise DCT features directly, an alternative approach is developed. Following the similar perspective of adaptively weighting the DCT basis functions to acquire different DCT bases in Adapt-DCT CNN, a novel idea of exploiting intra-group DCT channel relationships is proposed. This method aims to cultivate attention based on intra-channel LMH-DCTC sets to produce an attention map that encourages the network to emphasize the relationship changes within varying groups of DCT channels. By utilising the attention mechanism on the interaction of LMH-DCT feature maps, it was shown that it is possible to improve the performance without increasing model complexity. In other words, it is exceptionally crucial to incorporate attention mechanisms into the Adapt-DCT CNN with a DCT input image (input image with DCT coefficients) to understand the effects on several FGVC datasets.

The efficient channel attention (ECA) [156] is an improvised attention mechanism originating from SE-Net where it avoids dimensionality reduction and emphasizes cross-channel relationships. The working strategy behind ECA is to conduct fast 1D convolution between the kernel and feature channel to produce a channel attention map. Since DCTC carries a compact feature representation where the redundant feature is filtered, therefore explicit dimensionality reduction could be unnecessary as it may cause undesired information loss. Besides, the avoidance of dimensionality reduction is important for learning channel attention based on [156]. This is relatable as the fine-grained DCT features are deeply encoded in the channel dimension. The cross-channel interaction in DCT features is important as it gathers frequency correlation across channels forming complex patterns and relationships that can ease FGVC. Moreover, the cross-channel interaction can retain the performance and robustness of a model. In Table 3.1, comparison metrics between CBAM, SE-Net, and ECA extracted from [156] are computed to present the reason behind selecting ECA as the baseline module in this research.

Table 3.1: Metrics comparison between three attention modules for selection criteria.

Metrics / Architecture	Dimensionality Reduction	Cross-channel Interaction	Lightweight Model
SE-Net [112]	✓	✓	✗
CBAM [113]	✓	✓	✓
ECA [156]	✗	✓	✓

It is clear that by considering the comparison from Table 3.1, it is particularly relevant to pick ECA owing to its modularity, criteria, and efficiency over the other attention mechanism alternatives. Hence, ECA is selected for its ability to capture individual cross-channel interactions and reduce model complexity, which is aligned with our objective.

The ECA is employed as the baseline attention module on top of the Adapt-DCT CNN with several tweaks. Similar to the original proposed ECA to capture the local cross-channel interaction, the ECA module is further adjusted to obtain the intra-group DCT channel interaction. Specifically, a larger size of the 1D convolution kernel and stride size is used as compared with the former ECA to harvest the intra-group DCT channel interaction and maximize the correspondence of DCT channel sets and weights. The ECA that is altered to suit Adapt-DCT CNN is called 'Hybrid Modified-ECA' (HyMod-ECA) which enhances the former ECA.

Figure 3.3 illustrates the overview of the HyMod-ECA module. Provided the input feature (X) to acquire a 1D vector ($\overline{X}$) via global average pooling (GAP), X is viewed as a sequential order of tensors consisting of different DCTC channel sets (best viewed in colour) separated by multiple thick lines. These grouped channels produce $\overline{X}$, which is referred to as the 'aggregated DCT channel set'. The number of available DCT channel sets is represented by $\frac{C}{f}$ (more details in the following section). The DCT channel set weights ($\overline{A}$) are produced by performing a 1D convolution between $\overline{X}$ and the 1D kernel (K), whereas the 1D kernel size is determined by the number of DCT channels per set (f).

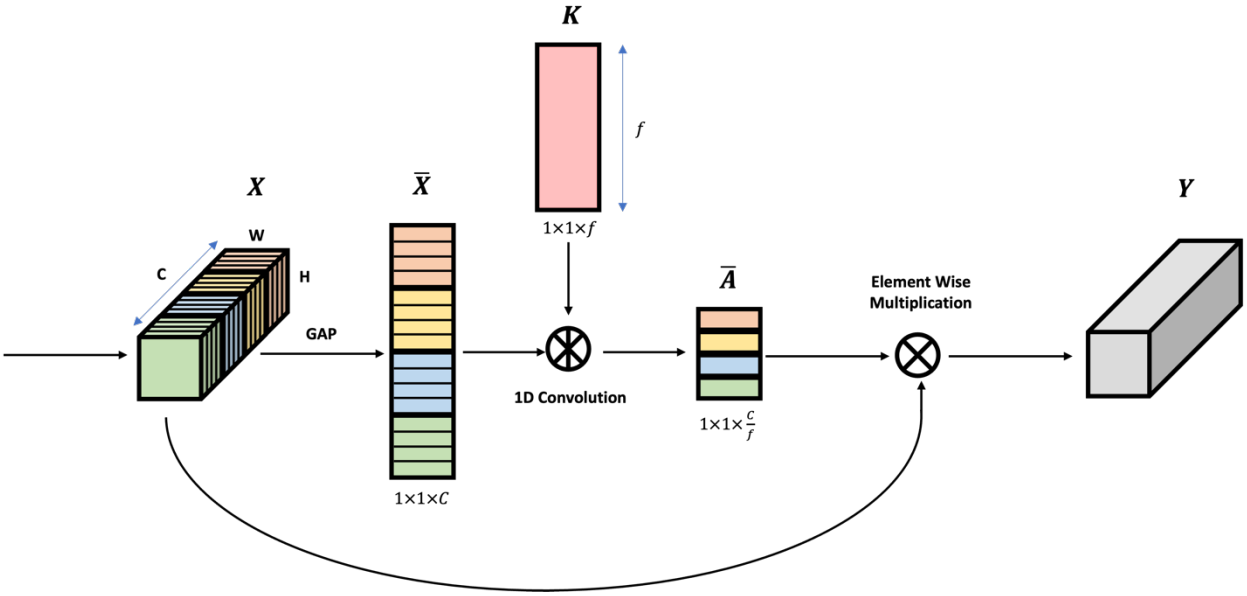


Figure 3.3: Overview of the Hybrid Modified Efficient Channel Attention (HyMod-ECA). The thin line separates individual channels, while the bold line separates channel groups.

In the following section, the methodology and experimental design of the M-Skipped network, the Adapt-DCT convolutional kernel, and the HyMod-ECA module are established. The M-Skipped DCT CNN is designed to integrate frequencies beyond L-DCTC to improve FGVC in the compressed domain. Due to the limitations associated with conventional pointwise convolutional kernels in the context of compressed domain, the Adapt-DCT kernel is introduced and integrated into the prior M-Skipped network. To study the interactions between the convolved frequency features generated by the enhanced network, the HyMod-ECA module is further incorporated into the existing M-Skipped network which already encompasses the Adapt-DCT kernel. The synthesis of each developed algorithm within this framework combines the compressed domain approaches to address the specific challenges encountered in FGVC.

3.2 Skipped Medium-DCT Convolutional Neural Network

The motivation to introduce the skipped M-DCT branch to the baseline CNN model in a compressed domain is to enable fine-grained communication between low- and high-level features. The foundational concept of skipping layer is motivated by [113]. This skipping methodology offers great flexibility in adjusting skip lengths to enable feature exchange at multiple scales. It fits the objective of this research on exploring the effects of integrating M-DCTC with different parts of the baseline network. The shortcut connections from ResNet and DenseNet aimed to address the gradient vanishing problem and enhance feature reuse. These skipping layers are not employed as they do not serve the purpose of this research.

The skipped M-DCT convolutional neural network consists of a baseline CNN network attached to a skipped convolutional branch. The skipped convolutional branch is targeted to extract intermediate fine-grained features. The L-DCTC input is fed through the CNN model based on pointwise convolutional blocks. The M-DCTCs are fed through (bypass) another 1×1 skipped convolutional layer (with batch normalization and activation function). The skipped convolutional layer's output is then concatenated with the intermediate output feature maps from the baseline network. The subsequent combined feature maps are then fed to the classifier. Figure 3.4 shows the basic integration of M-DCTC with the convolutional features. The fine-grained features (M-DCTC) are retained in the elementary form by passing the M-DCTCs through a convolutional layer. It is intended to combine low- and high-level features to improve feature representation and model robustness.

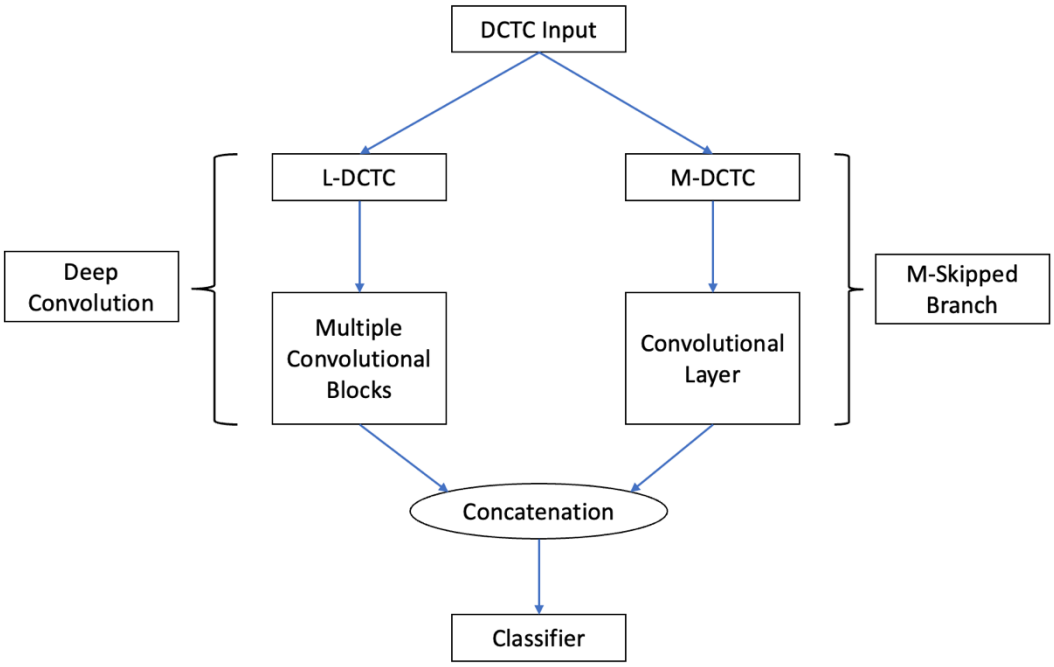


Figure 3.4: Basic illustration of the integration of M-DCTC with baseline convolutional network.

3.2.1 Low, Medium, High DCTC Representations

This work employs partial CODEC on JPEG images during training to obtain DCTC as input to the network. In particular, the datasets comprising raw RGB images undergo a process of partial compression. The formulation below demonstrates the process and the separation of low, medium, and high DCTC (LMH-DCTC).

Let Equation 3.1 denote the tensor comprising raw RGB images in the spatial domain, while Equation 3.2 represents the input tensor in the frequency domain. Equation 3.3 provides a fundamental overview of the conversion of raw RGB image data from spatial to frequency (DCT) domain:

$$X_i \in R^{h \times w \times c} \quad \text{Eq. 3.1}$$

$$X_v \in R^{\frac{h}{p} \times \frac{w}{p} \times p^2 \times c'} \quad \text{Eq. 3.2}$$

$$\mathcal{H}_{2D-DCT}(X_i) = X_v \quad \text{Eq. 3.3}$$

Function H represents a forward 2D DCT with $p \times p$ partition, where h and w represent the height and width of the input tensor respectively, c and c' refer to spatial channels and frequency channels respectively. The conversion of the spatial domain input X_i to the frequency domain input X_v requires 2D forward DCT, which is the core of JPEG compression. The conversion of the input tensor from the spatial domain to the frequency domain will shrink the spatial dimension from $h \times w$ to $\frac{h}{p} \times \frac{w}{p}$, and extend the channel depth to (p^2). The depth-wise coefficients (p^2) are derived from the sum of the DCT basis functions across that partition block.

Regarding the idea of the top left portion of the $p \times p$ partitions representing L-DCTC and the bottom right representing high DCTC, we derived the zigzag encoded $1 \times p^2$ DCTCs are arranged in ascending order from low to high DCTC. Fundamentally, L-DCTC corresponds to coarse and slow varying features while H-DCTC corresponds to rapidly varying features (often noise). DCTC pruning techniques typically suggest that the most useful L-DCTCs are found in the first 12 coefficients. Some early literature [157][158][159][45] performed similar DCTC factorizations based on different ratios but those works were not applied to FGVC. This section takes the initiative to perform a split of four to factorize LMH-DCTC. Specifically, a dedicated factor of two is allocated for M-DCTC which potentially encapsulates more fine-grained features.

Zigzag encoding was used to convert the $p \times p$ partition into a $1 \times p^2$ depth-wise DCTC representation after the 2D forward DCT step. Following the zigzag encoding pattern for each $p \times p$ partition, as shown in Figure 3.5, this method explicitly defines the first $\frac{p^2}{4}$ DCTCs as L-DCTC, the intermediate $\frac{p^2}{2}$ DCTCs as M-DCTC while the last $\frac{p^2}{4}$ coefficients as H-DCTC. Figure 3.5 explains the factorization pattern of a $p \times p$ DCT partition.

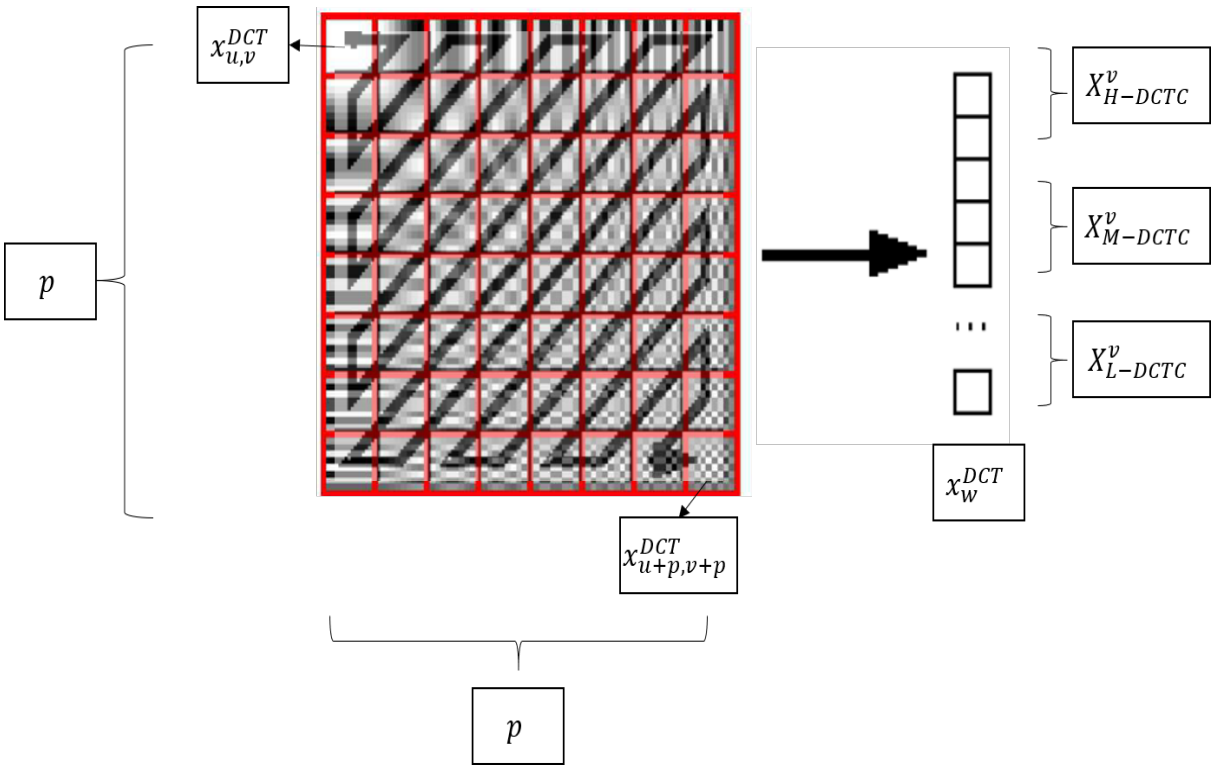


Figure 3.5: Low, medium, and high DCTC factorisation. After 2D-DCT and zigzag encoding, the 1D vector is factorised along the channel direction.

For a single channel of spatial input with $p \times p$ partition at an arbitrary location of (i, j), forward 2D-DCT will map the spatial input of $\{x_{i,j}^{RGB}, \cdots, x_{i+p,j+p}^{RGB}\}$ to a frequency domain of $\{x_{u,v}^{DCT}, \cdots, x_{u+p,v+p}^{DCT}\}$. The zigzag encoding will map the frequency partition block to $\{x_0^{DCT}, \cdots, x_{p^2-1}^{DCT}\}$, with x_w^{DCT} indicating the instantaneous index from the depth-wise vector. The entire partition restructure can be written as:

$$\{x_{i,j}^{RGB}, \cdots, x_{i+p,j+p}^{RGB}\} \leftrightarrow \{x_w^{DCT}, \cdots, x_{p^2}^{DCT}\} \quad \text{Eq. 3.4}$$

From the p^2depth-wise DCTCs, we factorize X_v along the p^2 dimension into low (X_{L-DCTC}^{v}), medium (X_{M-DCTC}^{v}), and high (X_{H-DCTC}^{v}) DCTC representation with the following notation:

$$X_v = \{X_{L-DCTC}^{v}, X_{M-DCTC}^{v}, X_{H-DCTC}^{v}\} \quad \text{Eq. 3.5}$$

Where they share the same spatial tensor of size as X_v, with different depths along the channel dimension. The resulting LMH-DCTC tensor input representation is shown as follows:

$$X^{v}_{L-DCTC} \in \mathbb{R}^{\frac{h}{p}\times\frac{w}{p}\times\frac{\boldsymbol{p^2}}{\boldsymbol{4}}\times c'} \quad \text{Eq. 3.6}$$

$$X^{v}_{M-DCTC} \in \mathbb{R}^{\frac{h}{p}\times\frac{w}{p}\times\frac{\boldsymbol{p^2}}{\boldsymbol{2}}\times c'} \quad \text{Eq. 3.7}$$

$$X^{v}_{H-DCTC} \in \mathbb{R}^{\frac{h}{p}\times\frac{w}{p}\times\frac{\boldsymbol{p^2}}{\boldsymbol{4}}\times c'} \quad \text{Eq. 3.8}$$

With $p = 8$ in a standard JPEG compression, compressing a 8×8 partition will produce a corresponding channel with a depth of 64. An even ratio split of 4 supplies L-DCTC with 16 DCTC, which is more than the fundamental requirement to fully represent the contents within a particular $p \times p$ partition. M-DCTC is given 32 coefficients while H-DCTC is given 16 coefficients. Since an RGB input consists of 3 channels, applying forward 2D-DCT on an $8 \times 8 \times 3$ partition will produce the corresponding channel with a depth of 192.

3.2.2 Baseline Network Setup

This thesis adopts the VGG-16 architecture as the baseline model due to its simplicity and proven effectiveness. The uniformity of its architecture can simplify the design process, rendering it to be a versatile apparatus for this research when compared to other SOTA networks. Notably, the VGG-16 architecture excels in interpretability. Thus, it is convenient to facilitate a thorough analysis of its behaviour across various contexts. By leveraging the VGG-16 as a baseline model, it prompts an early insight and focus on the DCT technique that addresses the objective of this research without considering the need for exhaustive evaluations across an extensive array of SOTA networks. The substitution of VGG-16 with other SOTA networks such as Mobile-Net or Res-Net with M-Skipped-DCT architecture is straightforward.

In the compressed domain, more emphasis is put on DCTC depth-wise feature representations over spatial correlations. In essence, a 1×1 convolution kernel that emphasises depth-wise context can achieve increased parameter savings compared to a 3×3 convolution kernel. Hence, a pointwise convolution filter was used in the experiments given its superior performance. Convolutions based on 3×3 kernels were also explored but it was found that 1×1 performed better. Consider the ratio below to

compare the increment in trainable parameters from using 3×3 kernels over 1×1 kernels:

$$Ratio = \frac{3 \times 3 \times n}{1 \times 1 \times n} = 9 \qquad \text{Eq. 3.9}$$

Where n represents the number of depth-wise filters. With a single convolution layer, a 3×3 kernel has 9 times more trainable parameters compared to a 1×1 kernel. The VGG-16 architecture consists of 3 convolutional layers per block, with a total of 3 convolutional blocks, which translates into significant parameter savings, leading to a reduction of both training time and energy consumption and a lower probability of overfitting.

In most of the general image classification tasks, VGG-16 typically accepts an RGB image input with a spatial size of 224×224 and a channel depth of 3. This thesis processes the spatial input with forward 2D-DCT using an 8×8 partition, resulting in a DCTC input with a spatial size of 28×28 and a channel depth of 192 [100]. To properly fit a DCTC with a smaller spatial size and larger channel dimensionality, the third convolutional block from the original VGG-16 algorithm which accepts an input spatial size of 28×28 was used as the initial block to process this input. This adheres to the VGG algorithm's convention of preserving the spatial size and channel depth in each successive convolutional block, facilitating a systematic and effective extraction of hierarchical features. This also ensured that the spatial size of the final output before flattening the layer was the same as the original RGB version of VGG-16. In summary, in this thesis, a total of 3 convolution blocks from the original VGG-16 network were used instead of 5, i.e. convolution blocks from the third to the fifth stage.

The 'Parametric Rectified Linear Unit' (PReLU) was selected as a key activation function over the original 'Rectified Linear Unit' (ReLU) for all of the convolutional layers in the primary network, while the M-skipped branch implemented ReLU. Figure 3.6 compares the activation responses of both ReLU and PreLU. This is in conjunction with the feature scaling of the compressed domain images into the range from -1 to +1. PReLU also provides a larger activation function region along the negative portion of the input to facilitate the learning process.

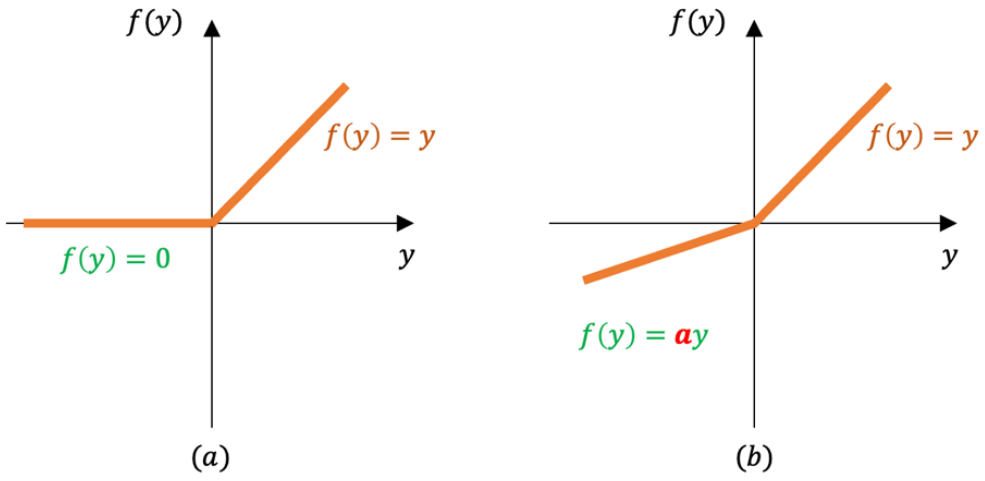


Figure 3.6: Activation function comparison between (a) ReLU vs (b) PreLU). PreLU allows the negative response of the network to continuously learn.

In addition, our model also avoids using fully connected layers at the end of the network. Preliminary experiments showed that the inclusion of fully connected layers sometimes resulted in minimal to no performance improvement, and at other times resulted in worse performance. Moreover, these layers contribute a significant number of additional parameters. Hence, the final concatenated output feature maps are fed directly into a linear Softmax layer after flattening. A more detailed experiment can be found in the ablation study in section 4.1.3.

Table 3.2 shows the modified version of the VGG-16 model in the DCT domain. The abbreviations of $[f, k, s, p]$ indicate the number of filter channels, kernel size, stride size, and padding respectively. The number of layers in a convolutional block is usually in the power of two due to hardware efficiency. Since the corresponding DCTC input from an RGB image consists of 192 channels, thus 256 filters are dedicated to the first convolutional block, and 512 filters are used in the second and last block. The number of baseline filters for each convolutional block was kept the same across all experiments to avoid discrepancy. The M-skipped connection can be concatenated with any of the output feature maps from the original baseline network marked '*'.

Table 3.2: VGG-16 baseline model.

Layer	$[f, k, s, p]$	**Output**	**Details**
Conv1_1, 2, 3	[256, 1, 1, 0]	$28 \times 28 \times 256$	3x layer
2D Max-pooling	[-, 2, 2, 0]	$14 \times 14 \times 256$*	-
Conv2_1, 2, 3	[512, 1, 1, 0]	$14 \times 14 \times 512$	3x layer
2D Max-pooling	[-, 2, 2, 0]	$7 \times 7 \times 512$*	-
Conv3_1, 2, 3	[512, 1, 1, 0]	$7 \times 7 \times 512$	3x layer
2D Max-pooling	[-, 2, 2, 0]	$3 \times 3 \times 512$*	-
Concatenate	-	$14 \times 14 \times 256 + (Mfilter)$[a] $7 \times 7 \times 512 + (Mfilter)$[b] $3 \times 3 \times 512 + (Mfilter)$[c]	Connection from the output of the M-Skipped Branch can be connected to any of the output with *
Classifier	-	Class number	-

a. When M-skipped is connected to the output feature map of Conv1_3, the input towards Conv2_1 will be $14 \times 14 \times 256 + (Mfilter)$.

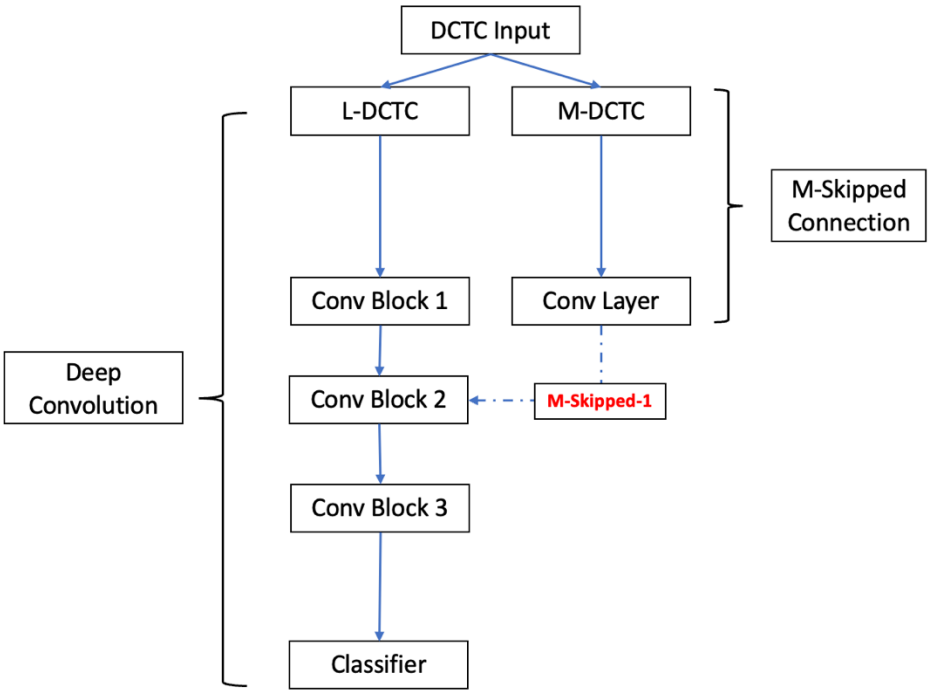
b. When M-skipped is connected to the output feature map of Conv2_3, the input towards Conv3_1 will be $7 \times 7 \times 512 + (Mfilter)$.

c. When M-skipped is connected to the output feature map of Conv3_3, the linear input towards the final classifier will be $3 \times 3 \times 512 + (Mfilter)$.

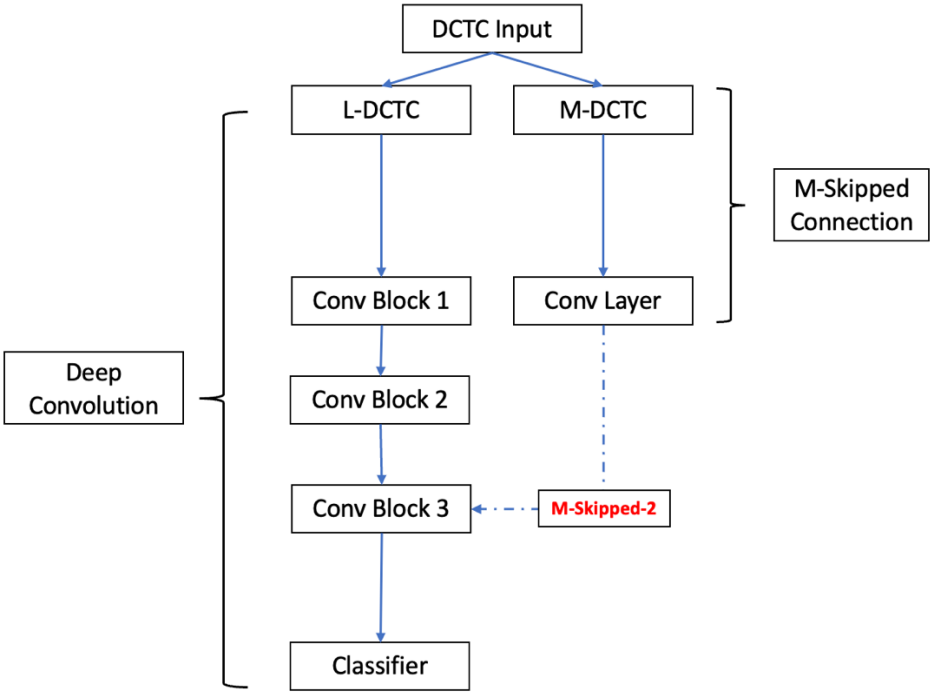
3.2.3 M-Skipped DCTC Branch

This section demonstrates the feature representation between the M-Skipped DCTC and the deeply convolved L-DCTC. The basic idea is to deliver a lower-level feature representation based on M-DCTC to the higher-level feature by using a skip connection. L-DCTC represents coarser features while M-DCTC represents finer features. The full spectrum of L-DCTCs is obtained from the factorisation and fed into the main network instead of using a pruning technique. This is because it provides the convolutional layers the flexibility to select which features to use from the full spectrum of L-DCTCs.

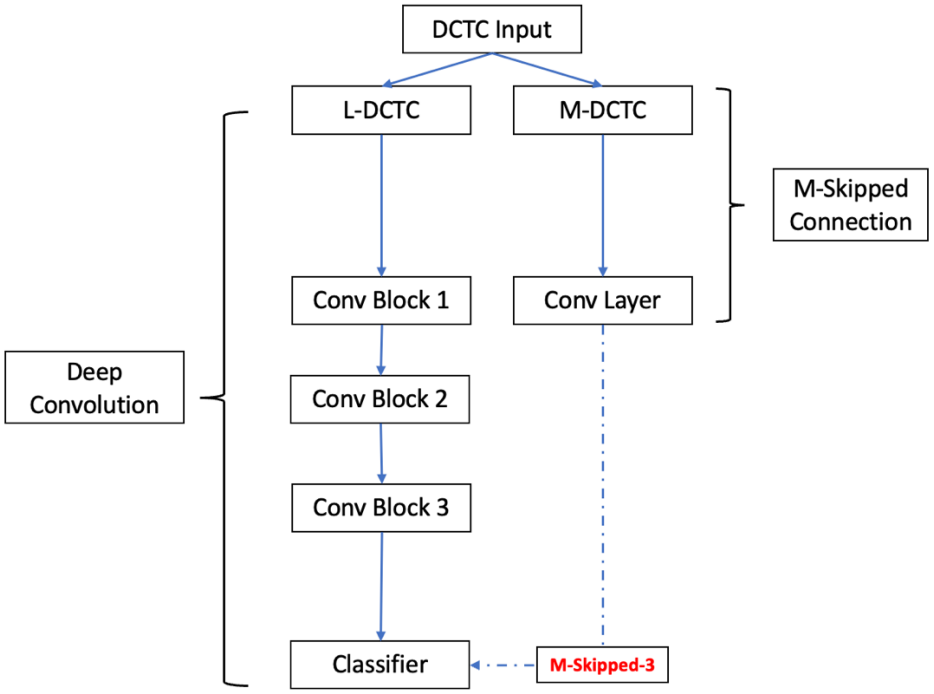
There are a few ways to integrate a single M-DCTC skipped convolution branch with the baseline network. In Figure 3.7, three variations of single M-Skipped branches are established. The L-DCTCs are passed through the baseline network, while the M-DCTCs are passed through a skipping branch. In Figure 3.7 (a), when the skipping branch is connected with the output from the first convolutional block of the baseline network, it is denoted as M-Skipped-1. In Figure 3.7 (b), when it is connected with the output from the second convolutional block, it is denoted as M-Skipped-2. Whereas M-Skipped-3 refers to the variation where the skipping branch is connected to the output from the last convolutional block, shown in Figure 3.7 (c).



(a)



(b)



(c)

Figure 3.7: M-Skipped DCT-CNN for variant (a) M-Skipped-1, (b) M-Skipped-2, (c) M-Skipped-3.

There are several differences and significance for connecting the skipping branch with different parts of the baseline network. When integrating the M-DCTC skipping branch with the early convolutional block, the combined features are significant in producing higher-level hierarchical representations. While integrating the skipping branch directly with the last convolutional block, the concatenated features contribute significantly to the final classification tasks. Therefore, it is crucial to understand the impact of different skipping variations to achieve the objective of this research.

The formulation of the novel M-Skipped feature representation is shown below. Any intermediate output feature map of a deep neural network is denoted as:

$$Y_v^i = X_v^{i+1} = G(X_v^i) \quad \text{Eq. 3.10}$$

Y is the output while X is the input. The superscript 'i' indicates the sequence of the convolution block. G is the function combining the convolutional operation and the activation function along the main network, resulting in a new feature map. In this case, L-DCTC is fed into the deep network such that the model can focus on learning fundamental key features. The new parameter for the M-Skipped DCTC branch is denoted as:

$$Y_v^{M-DCTC} = g(X_{M-DCTC}^v) \quad \text{Eq. 3.11}$$

Where the notation of Y_v^{M-DCTC} represents the output by convolving M-DCTC. In Equation 3.11, different from the main network, 'g' denotes the shallow convolution block in the M-Skipped branch. The skipped connection of shallowly convolved M-DCTC separates the fine-grained features from the main network. A shallow convolution block is applied on M-DCTC to preserve low-level fine-grained features.

$$Y_v^{final} = Y_v^i \cup Y_v^{M-DCTC} = G\left(X_v^i\right) \cup g(X_{M-DCTC}^v) \quad \text{Eq. 3.12}$$

Y_v^{final} is the final integrated output feature map obtained by concatenating the output from the $(i+1)^{th}$ convolutional block with the output of the M-DCTC branch. In this preliminary experiment, it was found that concatenation works better than information exchange such as matrix summation and octave convolution.

$$Classification\ Output = Softmax(Y_v^{final}) \quad \text{Eq. 3.13}$$

The integrated final feature undergoes flattening and a feed-forward layer ending in the final classifier (Softmax). As such, the M-DCTCs which contain low-level fine-grained feature information can be made visible to the classifier with only one shallow convolutional layer in between.

3.3 Adaptive-DCT Pointwise Convolutional Kernel

While the M-Skipped DCT-CNN effectively considers higher frequency bands for improved feature extraction, it has limitations in adaptability and efficiency, particularly in handling complex frequency patterns. These limitations arise from the nature of the convolutional kernel and the underlying algorithms. The current composition of the convolutional kernel demonstrates resilience towards challenging fine-grained features, which can hinder its performance across various tasks. To address this issue, the Adaptive-DCT-based convolutional kernel is introduced. This algorithm enables the model to prioritise more relevant frequencies during kernel formation. This adjustment enhances both the flexibility and accuracy of feature representations by dynamically adapting to the complexity of the data, thereby overcoming the limitations associated with the resilience of the convolutional kernel in the M-Skipped approach.

The Adapt-DCT kernel is initially explained simplistically in section 3.3.1 from a forward 2D-DCT-II process to set up a fundamental understanding. The basic formulation is then extended towards the adaptive DCT tensor in section 3.3.2 with relation to the working of DCT, basis functions, and depth-wise channel mapping. Section 3.3.3 tailored the final frequency domain pointwise convolutional kernel through graphics and a few associated conditions for usage and different circumstances.

Figure 3.8 shows the difference between the original forward 2D-DCT and the modified method employed. A modified 3-dimensional DCT basis function is formed by applying zigzag encoding towards the original 4-dimensional one. The Adapt-DCT pointwise convolutional kernel replaces the coefficients of the modified DCT basis functions with trainable weights to adaptively learn the importance of spatial and frequency bases. The kernel is entirely produced by adaptively weighting the modified DCT basis functions during the training. Figure 3.9 illustrates the formation of the

pointwise convolutional kernel from adaptively weighting the modified 3D DCT basis functions. An adaptive kernel ($\mathcal{K}_{ts}$) is element-wise multiplied (denoted by '⊗') with the basis functions ($\mathbb{B}'$). The spatial summation is applied towards the resulting tensor forming a pointwise convolution kernel ($\mathcal{K}_{fr}$). The kernel is then used to perform pointwise convolution with the DCT features along the compressed domain CNN, as shown in Figure 3.10.

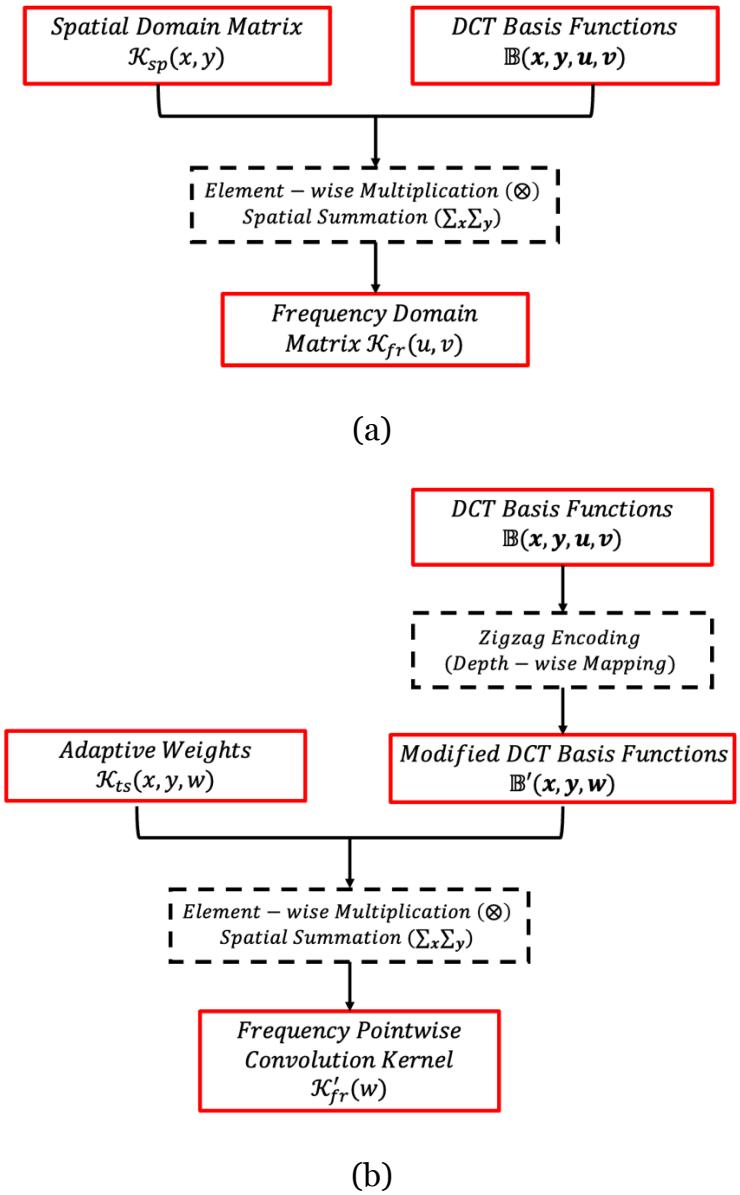


Figure 3.8: Comparison between (a) original forward 2D-DCT and (b) modified forward 2D-DCT.

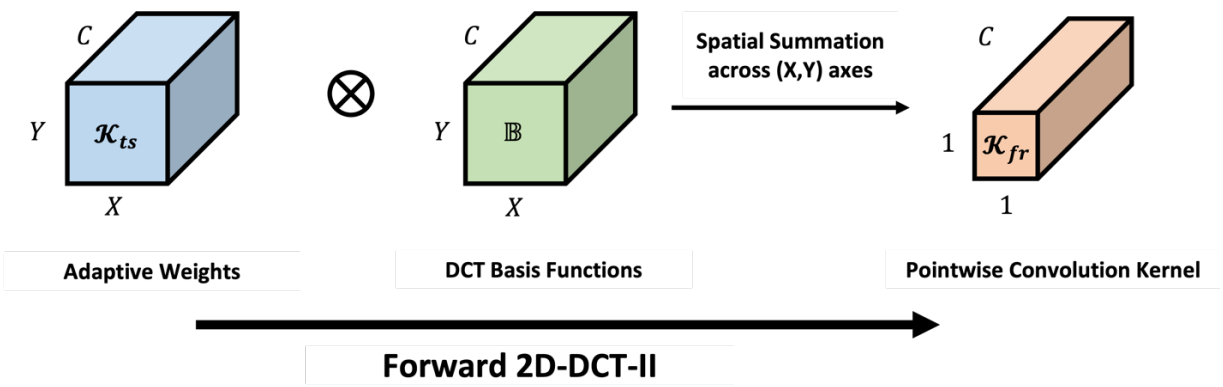


Figure 3.9: The formation of the frequency pointwise convolution kernel from adaptively weighting the modified DCT basis functions.

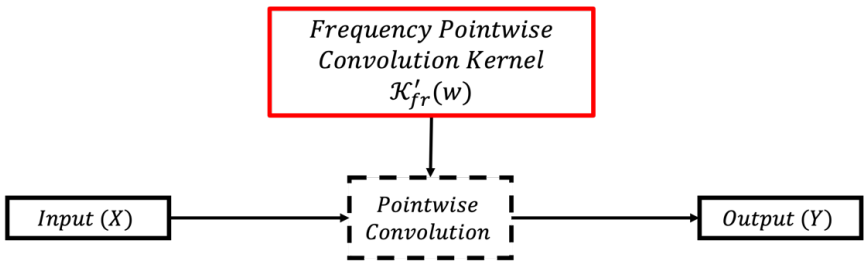


Figure 3.10: Pointwise convolution adopting the Adapt-DCT pointwise convolutional kernel.

The key difference between the Adapt-DCT convolution and the Cosine Basis Convolution (CBC) is that CBC uses cosine basis to produce spatial filter weights by weighting the parameters (frequencies and phases) within basis functions, while the Adapt-DCT convolution weights the modified basis functions directly to generate the frequency domain pointwise convolution kernel. The Adapt-DCT study serves as an initial attempt to explore the usability of the modified DCT basis functions in both spatial and frequency bases.

During the training of the Adapt-DCT CNN, only part of the DCT processes is required. The forward pass consisting of the forward DCT only involves the multiplication of the adaptive weights with the modified DCT basis functions, followed by spatial summation to acquire the DCT pointwise convolutional kernel. While computing the backpropagation algorithm, only the multiplication of the differential kernel weights with the modified DCT basis functions applies. The optimisation of the Adapt-DCT kernel and backpropagation is like a perceptron forward pass algorithm, as DCT is a linear transformation.

3.3.1 Background Motivations and Theories of Adaptive DCT Basis Functions

When a partition of spatial domain image or a feature map is applied with 2D-DCT-II, the frequency bases are mapped towards the layer dimension. The layer dimension comprised different sets of individual frequency bases. A similar idea can be applied to the formation of a pointwise convolution kernel, with the difference being that the spatial dimension of the kernel is smaller than the feature maps. The Adapt-DCT kernel is a 3D matrix. Once multiplied with the modified DCT basis functions followed by the application of spatial summation, a frequency domain pointwise convolution kernel is formed.

The forward 2D-DCT-II is introduced to provide some background knowledge. Each intermediate step along the forward 2D-DCT-II process is elaborated to ease understanding. The full expansion of the DCT basis functions is highlighted to ease the explanation of the adaptive weighting at the later stage. Simplifications and notations of forward 2D-DCT-II are shown to derive the modified DCT basis functions. The objective is to obtain the modified DCT basis function. Generally, the forward 2D-DCT-II is written as:

$$\mathcal{K}_{fr}(u,v) = \mathcal{F}_{f-DCT}\{\mathcal{K}_{sp}(x,y)\}$$

$$= \left[\frac{2}{N} * \xi(u) * \xi(v)\right] \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [\mathcal{K}_{sp}(x,y)] \left[\cos\frac{\pi \cdot u(x+0.5)}{N}\right] \left[\cos\frac{\pi \cdot v(y+0.5)}{N}\right] \quad \text{Eq. 3.14}$$

Where function $\xi(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & otherwise \end{cases}$, $\xi(v) = \begin{cases} \frac{1}{\sqrt{2}}, & v = 0 \\ 1, & otherwise \end{cases}$

$\mathcal{F}_{f-DCT}$ indicates the forward 2D-DCT-II. $\mathcal{K}_{fr}$ is the frequency domain matrix with $\mathcal{K}_{fr}(u,v)$ representing the element with an index at (u,v). $\mathcal{K}_{sp}$ is the spatial domain matrix with $\mathcal{K}_{sp}(x,y)$ representing the element with index at (x,y). N is the spatial dimension of the spatial domain matrix. Rewriting and simplifying the equation above yields:

$$\mathcal{K}_{fr} = \mathcal{F}_{f-DCT}\{\mathcal{K}_{sp}\} = \mathbb{C} \cdot \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}[\mathcal{K}_{sp} \cdot \mathbb{B}] \quad \text{Eq. 3.15}$$

Where: $\mathbb{C} = \frac{2}{N} * \xi(u) * \xi(v)$

$$\mathbb{B} = \left[\cos\frac{\pi \cdot u(x+0.5)}{N}\right]\left[\cos\frac{\pi \cdot v(y+0.5)}{N}\right]$$

$For\ x, y, u, v = 0,1,2, \dots, N-1$

The 4-dimensional tensor $\mathbb{B}$ in Equation 3.15 represents the DCT basis functions with element $\mathcal{B}(x, y, u, v)$ at the index location of (x, y, u, v). The forward 2D-DCT-II in Equation 3.15 converts the spatial coefficients ($\mathcal{K}_{sp}$) directly into frequency coefficients ($\mathcal{K}_{fr}$). An instantaneous tensor is given by the notation $\mathcal{K}'_{sp}$ can be obtained from the forward 2D-DCT-II. It represents the result of multiplying $\mathcal{K}_{sp}$ with the DCT basis functions (Equation 3.16). After applying spatial summation on $\mathcal{K}'_{sp}$, $\mathcal{K}_{fr}$ is obtained as the frequency coefficients (Equation 3.17). Figure 3.11 provides an example of the spatial summation for a 2D tensor; it is not to be confused with the summation of activation outputs in a regular neural network.

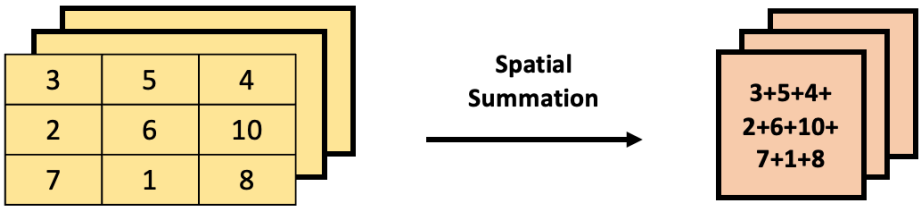


Figure 3.11: Spatial summation of a 2D tensor.

The spatial summation of the 2D tensor can be represented in Equations 3.16 and 3.17:

$$\mathcal{K}'_{sp}(x, y, u, v) = \mathcal{K}_{sp}(x, y) \otimes \mathcal{B}(x, y, u, v) \quad \text{Eq. 3.16}$$

$$\mathcal{K}_{fr}(u, v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathcal{K}'_{sp}(x, y, u, v) \quad \text{Eq. 3.17}$$

The $\mathcal{K}_{sp}$ and $\mathcal{K}_{fr}$ are single layers of a 2D matrix. $\otimes$ indicates element-wise multiplication. A spatial domain matrix of $\mathcal{K}_{sp}$ carrying $\Re^{N\times N}$ will produce a corresponding frequency domain matrix of $\mathcal{K}_{fr}$ carrying $\Re^{N\times N}$ according to 2D-DCT-II principle, whereas the corresponding DCT basis functions $\mathbb{B}$ will carry a shape of $\Re^{N\times N\times N\times N}$. The spatial dimension (height and width) of both $\mathcal{K}_{sp}$ and $\mathcal{K}_{fr}$ is denoted by N. The DCT basis functions follow the same fashion with each of the indexes in (x, y, u, v) ranging from $\{0, 1, 2, \dots, N-1\}$.

The following step is to define the modified DCT basis functions. It is obtained by mapping the DCT basis functions of frequency bases towards the depth channel. In other words, the DCT basis functions will be propagated from a 4D tensor into a 3D tensor. Specifically, the tensor index is mapped from (u, v) to (w), producing $\mathcal{B}'(x, y, w)$ from $\mathcal{B}(x, y, u, v)$. The objective of producing such modified function is to reduce the dimension of the DCT basis functions to ease the adaptive weighting and the kernel composition at the later stage. The 3D DCT basis functions are referred to as the 'modified DCT basis functions'. The depth-wise mapping from $\mathcal{B}(x, y, u, v)$ to $\mathcal{B}'(x, y, w)$ follows the 'horizontal prior' for algorithmic simplicity. It provides easier access to certain frequency coefficients at specific locations with instance (u, v). The spatial context $\mathcal{B}'(x, y, :)$ is referred to as the spatial bases whereas the channel context $\mathcal{B}'(:, :, w)$ is referred to as the frequency bases.

Figure 3.12 illustrates the original 4D DCT basis functions and the modified 3D counterpart.

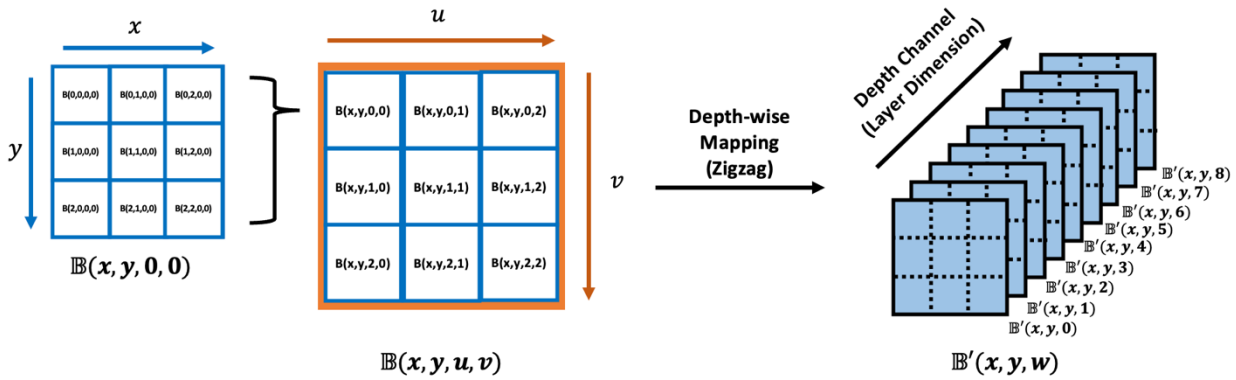


Figure 3.12: Original DCT basis functions $\mathcal{B}(x, y, u, v)$ versus modified DCT basis functions $\mathcal{B}'(x, y, w)$.

The zigzag encoding scheme is shown in Figure 3.13 and the corresponding element mapping is shown in Table 3.3.

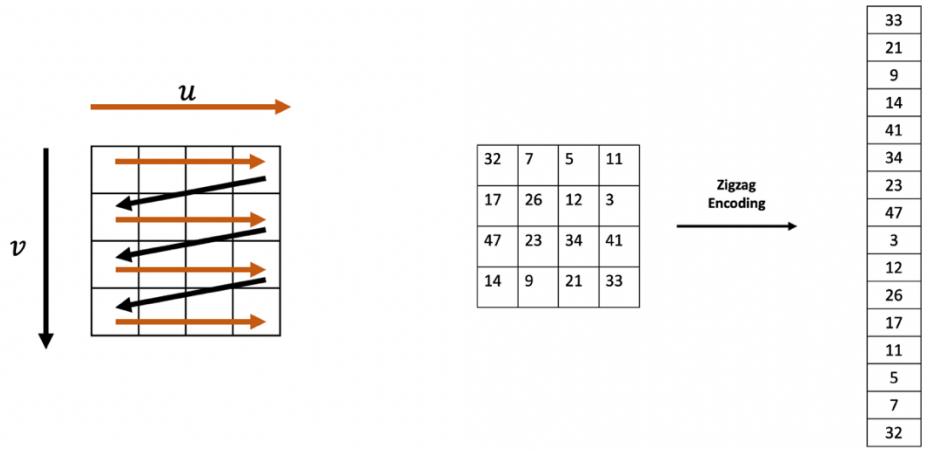


Figure 3.13: Horizontal prior zigzag encoding.

Table 3.3: Depth-wise mapping of elements from (u, v) to (w).

Index (u, v)	*Index* (w)	*Element*
(0,0)	0	32
(0,1)	1	7
(0,2)	2	5
(0,3)	3	11
(1,0)	4	17
(1,1)	5	26
(1,2)	6	12
(1,3)	7	3
(2,0)	8	47
(2,1)	9	23
(2,2)	10	34
(2,3)	11	41
(3,0)	12	14
(3,1)	13	9
(3,2)	14	21
(3,3)	15	33

The mapping procedures can be derived from the following notations:

$$\mathcal{B}(x, y, u, v) \xrightarrow{Depthwise\ Mapping} \mathcal{B}'(x, y, w) \qquad \text{Eq. 3.18}$$

$$\mathbb{B} \xrightarrow{Depthwise\ Mapping} \mathbb{B}'$$

Each set of (u, v) corresponds to a particular (w) in the layer dimension. As each of the index (u, v) is in the range of $\{0, 1, 2, \dots, N-1\}$, thus, the index (w) will range from $\{0, 1, 2, \dots, N^2 - 1\}$. With the introduction of 3D-modified DCT basis functions, both spatial and frequency bases are accessible. This allows flexibility tuning and information filtering along each base. A new tensor $(\mathcal{K}_{ts})$ is introduced to replace the spatial domain matrix of $\mathcal{K}_{sp}$ in conjunction with the modified DCT basis functions. $\mathcal{K}_{ts}$ carries the same shape as the modified DCT basis functions of $\mathbb{B}'$. This simplifies the element-wise multiplication to obtain the intermediate tensor of $\mathcal{K}'_{ts}$. $\mathcal{K}_{ts}$ is called the 'Adaptive DCT Tensor'. The corresponding shape of each tensor is:

$$\mathbb{B} \in \Re^{N \times N \times N \times N}$$

$$\mathbb{B}' \in \Re^{N \times N \times N^2}$$

$$\mathcal{K}_{ts} \in \Re^{N \times N \times N^2}$$

$$\mathcal{K}'_{ts} \in \Re^{N \times N \times N^2}$$

The modified DCT basis functions adhere to the same rules as the original 2D-DCT formula. These basis functions maintain orthogonality among each other, signifying that the inner product between any two distinct basis functions is zero. This characteristic holds a particular significance for the transformed kernel $(\mathcal{K}'_{ts})$. This is because it simplifies the kernel composition of $\mathcal{K}_{fr}$ by expressing it as individual decorrelated components. This composition involves decomposing the initial spatial filter of $\mathcal{K}_{sp}$, ensuring the preservation of kernel properties and enabling analytics for each component. The kernel properties are effectively distributed among these orthogonal components. Hence, the formation of $\mathcal{K}'_{fr}$ is required to adhere to the core basis function rules.

By modifying the earlier Equations 3.16 and 3.17, the illustrations and logic are presented in Figure 3.14:

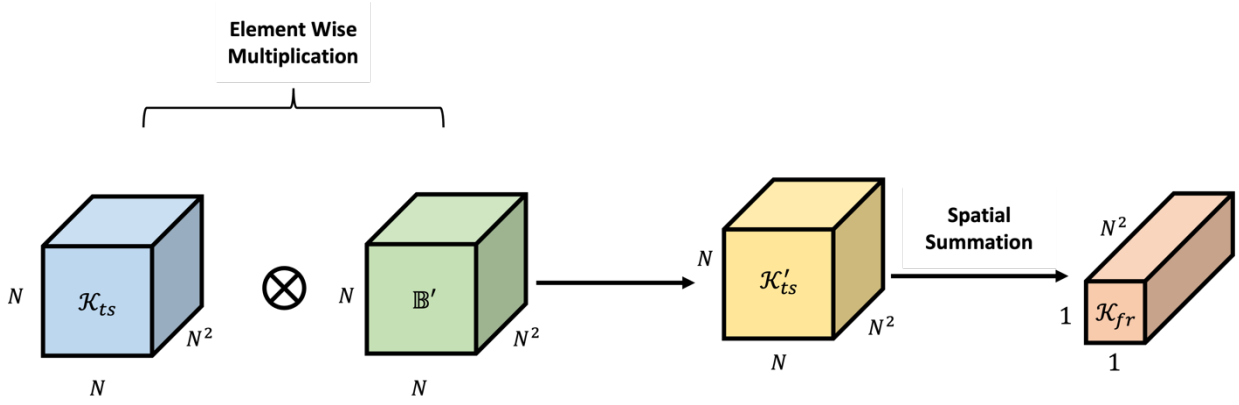


Figure 3.14: Obtaining frequency tensor from the Adaptive DCT Tensor.

$$\mathcal{K}'_{ts}(x,y,w) = \mathcal{K}_{ts}(x,y,w) \otimes \mathcal{B}'(x,y,w) \quad \text{Eq. 3.19}$$

$$\mathcal{K}'_{fr}(w) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathcal{K}'_{ts}(x,y,w) \quad \text{Eq. 3.20}$$

$$\mathcal{K}'_{fr} \in \Re^{1\times 1\times N}$$

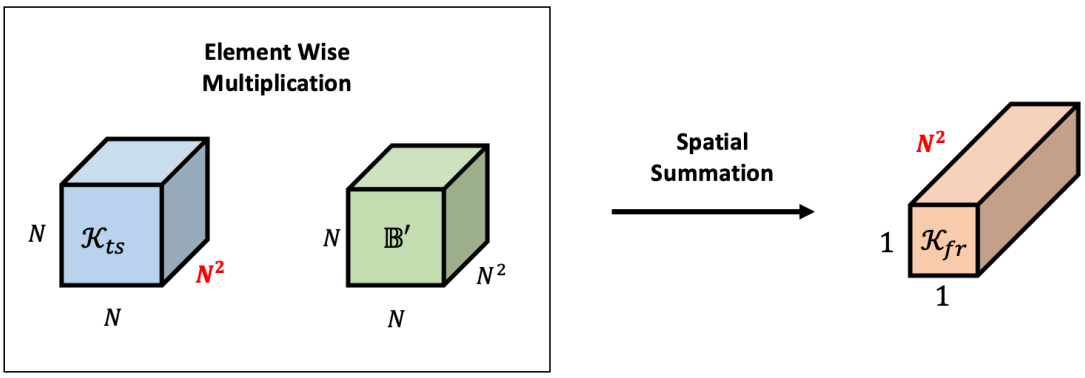
By conducting spatial summation on $\mathcal{K}'_{ts}$ along the (x,y) axes will produce a 1-dimensional frequency vector of $\mathcal{K}'_{fr}$. The intention of producing a 1D frequency vector at this point is to set up a common background for formulating the pointwise convolution kernel in the next section. The overall transformation of the Adaptive DCT tensor can be summarised as:

$$\mathcal{K}_{ts}(x,y,w) \xrightarrow{\substack{Element-wise \\ multiplication\ with\ \mathcal{B}'}} \mathcal{K}'_{ts}(x,y,w) \xrightarrow{Spatial\ Summation} \mathcal{K}'_{fr}(w)$$

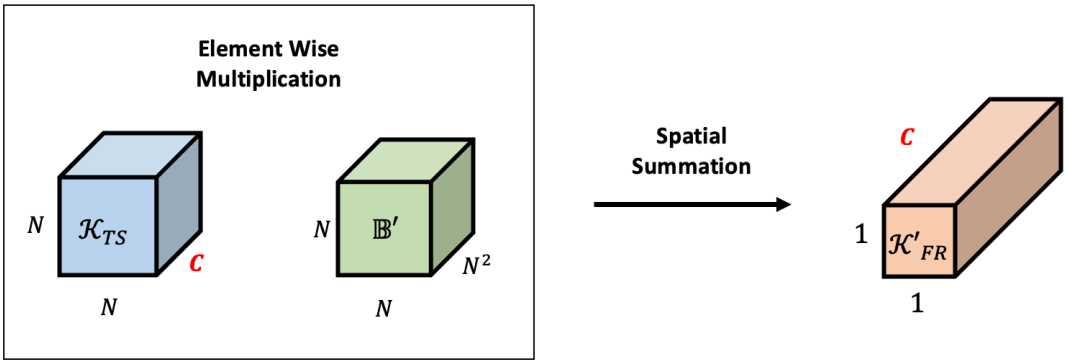
3.3.2 Adaptive DCT Pointwise Convolutional Kernel

In this section, the formulation of the Adapt-DCT kernel and its transformation towards the pointwise convolutional kernel will be established. The pointwise convolution is a convolution process that adopts 1×1 kernel of 1D vector. In the DCT domain, the pointwise convolution is widely implemented. Following the concept established earlier, a 1×1 kernel ($\mathcal{K}'_{fr}$) can be produced by multiplying an adaptive weight with the modified DCT basis functions. In other words, an adaptive tensor can be initialized as trainable weights to compose a 1×1 kernel.

Let the pointwise convolutional kernel be $\mathcal{K}'_{\mathrm{FR}}$ with a shape of $\Re^{1\times1\times\mathrm{C}}$, where C is the total number of channels carried by the 1×1 kernel. In the modified DCT basis functions, the number of frequency bases is derived from its corresponding spatial dimension ($\mathrm{N} \times \mathrm{N}$), denoted as N^2. Under normal circumstances, the number of channels (C) of the pointwise convolutional kernel ($\mathcal{K}'_{\mathrm{FR}}$) will always be greater than the number of frequency bases. Following the principle above, it can be derived that $\mathrm{C} \geq \mathrm{N}^2$. To produce a pointwise convolutional kernel consisting of C channels by inheriting earlier concepts, the desired adaptive tensor ($\mathcal{K}_{\mathrm{ts}}$) shall carry C layer instead of N^2. To achieve learning capabilities, the adaptive tensor can be defined as weights, denoted as $\mathcal{K}_{\mathrm{TS}}$. Thus, $\mathcal{K}_{\mathrm{TS}}$ is named as the 'Adaptive DCT (Adapt-DCT) Tensor', or simply adaptive weights. A simple comparison can be seen from Figures 3.15(a) and 3.15(b).



(a)



(b)

Figure 3.15: (a) Primary concept of adopting Adapt-DCT tensor to produce a 1-dimensional frequency vector. (b) The adaptive weighting of the modified DCT basis functions to produce the frequency pointwise convolution kernel.

In essence, due to the dissimilarities between the number of channels and frequency bases, each DCT basis function within a specific frequency base requires the application of several adaptive weights. This is achieved through a process known as 'channel extension' applied to the initial set of $\mathcal{K}_{\mathrm{ts}}$. The primary goal of channel extension is to establish a correspondence between adaptive weights and the basis functions. The objective is to synchronise their multiplication with the modified DCT basis functions.

The $\mathcal{K}_{\mathrm{ts}}$ in the earlier section multiplies a single layer of $\mathcal{K}_{\mathrm{ts}}(:,:,\mathrm{w})$ with a corresponding layer in the modified DCT basis function $\mathcal{B}'(:,:,\mathrm{w})$. Since $\mathcal{K}_{\mathrm{TS}}$ carries a channel of C as opposed to N^2, multiple layers within $\mathcal{K}_{\mathrm{TS}}$ shall be multiplied with a single layer of frequency base from $\mathcal{B}'$. In other words, each layer of DCT basis functions of frequency base is multiplied with a set of 'channel weights' within $\mathcal{K}_{\mathrm{TS}}$. The set of 'channel weights' is formed by extending each layer of the prior $\mathcal{K}_{\mathrm{ts}}(:,:,\mathrm{w})$ at the index location of w by $\frac{\mathrm{C}}{\mathrm{N}^2}$ times. The calculation for each set of channel weights possesses an equivalent depth of $\frac{\mathrm{C}}{\mathrm{N}^2}$. Each set of channel weights is denoted as $\mathcal{K}_{\mathrm{TS}}(\mathrm{w}')$ (or simplified as $\mathcal{K}_{\mathrm{TS}}^{\mathrm{w}'}$), with index w' referring to the $(\mathrm{w}')^{\mathrm{th}}$ set. An example is given in Figure 3.16, where $\mathrm{C} = 8$ and $\mathrm{N} = 2$. Thus, $\mathrm{N}^2 = 4$ and $\frac{\mathrm{C}}{\mathrm{N}^2} = 2$. $\mathcal{K}_{\mathrm{TS}}$ contains more learnable weights per frequency base as compared to $\mathcal{K}_{\mathrm{ts}}$ with improved learning capabilities.

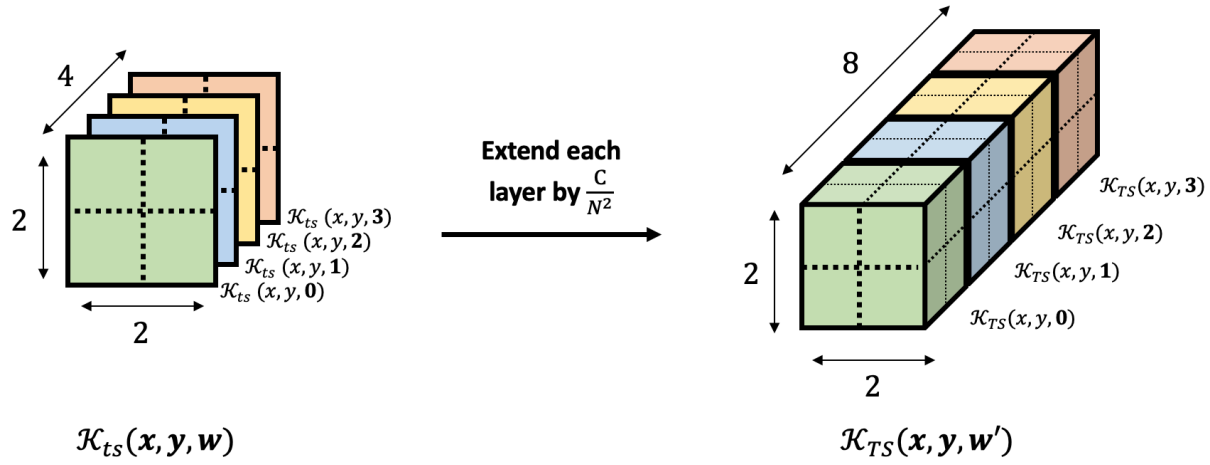


Figure 3.16: The extension of a single layer $\mathcal{K}_{ts}$ by $\frac{C}{N^2}$ times to produce the required $\mathcal{K}_{TS}$ which carries a channel dimension of C.

For each $\mathcal{K}_{TS}(w')$:

$$\mathcal{K}_{TS}(w') \in \Re^{N\times N\times\frac{C}{N^2}}$$

$$\mathcal{K}_{TS}^{w'} = \mathcal{K}_{TS}(w')$$

$$\mathcal{K}_{TS}^{w'} = \left\{\mathcal{K}_{ts}(x,y,0), \mathcal{K}_{ts}(x,y,1), \dots, \mathcal{K}_{ts}(x,y,\boldsymbol{z}), \dots, \mathcal{K}_{ts}\left(x,y,\frac{C}{N^2}-1\right)\right\} \quad \text{Eq. 3.21}$$

z is the channel index within each set of the channel weights ($\mathcal{K}_{TS}^{w'}$) ranging from $\left\{0,1,2,\dots,\frac{C}{N^2}-1\right\}$. The $\mathcal{K}_{TS}$ can also be viewed as equal grouping across the layer dimension to produce N^2 sets of equivalent depth tensor ($\mathcal{K}_{TS}^{w'}$). Each group is a set of channel weights to be multiplied with a frequency base. The final $\mathcal{K}_{TS}$ can be written as:

$$\mathcal{K}_{\mathrm{TS}} \in \Re^{\mathrm{N}\times\mathrm{N}\times\mathrm{C}}$$

$$\mathcal{K}_{TS} = \{\mathcal{K}_{TS}(0), \mathcal{K}_{TS}(1), \mathcal{K}_{TS}(2), \dots, \mathcal{K}_{TS}(w'), \dots, \mathcal{K}_{TS}(N^2-1)\}$$

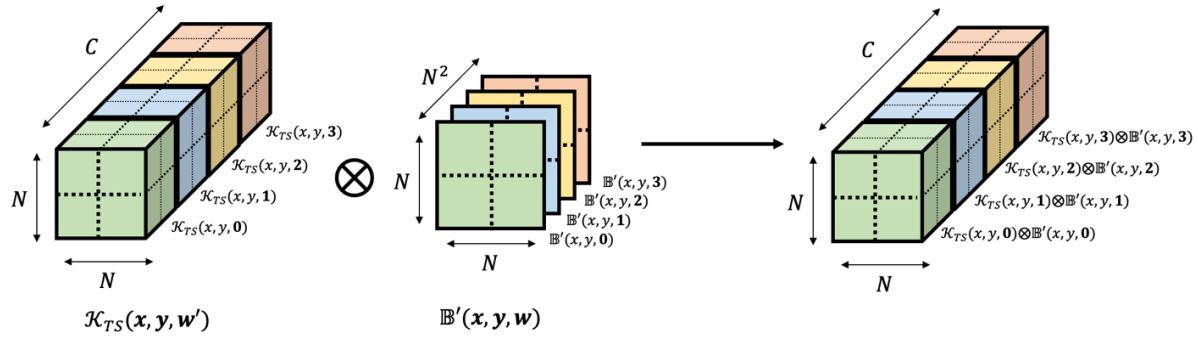
$$\mathcal{K}_{TS} = \left\{\mathcal{K}_{TS}^{0}, \mathcal{K}_{TS}^{1}, \mathcal{K}_{TS}^{2}, \dots, \mathcal{K}_{TS}^{w'}, \dots, \mathcal{K}_{TS}^{N^2-1}\right\} \quad \text{Eq. 3.22}$$

Consecutively, the modified DCT basis functions ($\mathbb{B}'$) can be separated into a single layer of $\mathcal{B}_w'$ to be multiplied with $\mathcal{K}_{TS}^{w'}$. The definition of $\mathbb{B}'$ is defined in the earlier section in Equation 3.18. Each $\mathcal{B}_w'$ is the frequency base instantiation across the layer dimension at w^{th} depth. The modified DCT basis functions can be written as a collection of single layers $\mathcal{B}'(:,:,w)$:

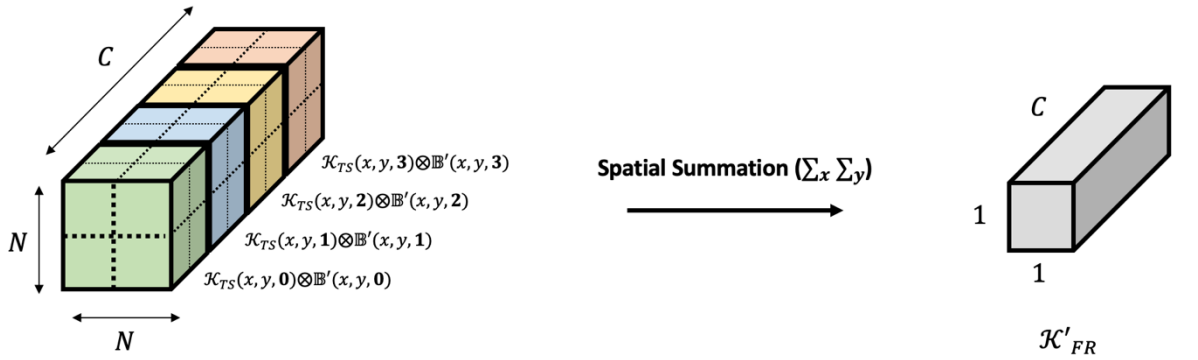
$$\mathbb{B}' = \{\mathcal{B}'(x,y,0), \mathcal{B}'(x,y,1), \dots, \mathcal{B}'(x,y,\mathrm{w}), \dots, \mathcal{B}'(x,y,N^2-1)\}$$

$$\mathbb{B}' = \left\{\mathcal{B}'_0, \mathcal{B}'_1, \mathcal{B}'_2, \dots, \mathcal{B}'_w, \dots, \mathcal{B}'_{N^2-1}\right\} \quad \text{Eq. 3.23}$$

During training, the model will optimize based on $\mathcal{K}_{TS}$ instead of directly on the 1×1 kernel. The modified DCT basis functions can also be pre-computed. Therefore, $\mathcal{K}_{TS}$ and $\mathbb{B}'$ are initialised before the training for computational efficiency. Each set of the tensor $\mathcal{K}_{TS}^{w\prime}$ is multiplied with the corresponding layer of the DCT basis function of $\mathcal{B}'_w$. The process is shown in Figure 3.17(a). Then, the resultant tensor will go through spatial summation and depth-wise concatenation to obtain the final pointwise convolution kernel ($\mathcal{K}'_{FR}$) as shown in Figure 3.17(b). The pointwise convolution kernel is formed via adaptive weighting of modified DCT basis functions followed by spatial summation. During deployment, only the final 1×1 kernel is required for inference. The inference model is like a fully pointwise CNN.



(a)



(b)

Figure 3.17: Full process of acquiring the frequency domain pointwise convolutional kernel from Adapt-DCT kernel.

The formula can be represented as:

$$\mathcal{K}'_{FR}(w') = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}[\mathcal{K}_{TS}(w')\otimes\mathcal{B}'(\mathrm{w})]_{(x,y)} \quad \text{Eq. 3.24}$$

It can be further simplified and denoted as:

$$\boldsymbol{\mathcal{K}'_{FR} = \Psi\{\mathcal{K}^{w\prime}_{TS}, \mathcal{B}'_{w}\}} \quad \text{Eq. 3.25}$$

$$\mathcal{K}'_{FR} = concat\{\Psi[\mathcal{K}^{0}_{TS}, \mathcal{B}'_{0}], \Psi[\mathcal{K}^{1}_{TS}, \mathcal{B}'_{1}], \Psi[\mathcal{K}^{2}_{TS}\cdot\mathcal{B}'_{2}], \dots, \Psi[\mathcal{K}^{w\prime}_{TS}, \mathcal{B}'_{w}], \dots, \Psi[\mathcal{K}^{N^2-1}_{TS}, \mathcal{B}'_{N^2-1}]\}$$

The function 'Ψ' represents element wise multiplication of $\mathcal{K}^{w\prime}_{TS}$ with $\mathcal{B}'_{w}$ followed by spatial summation along the (x, y) axes. For $\mathcal{K}'_{FR}(w')$:

$$\mathcal{K}'_{FR}(w') \in \Re^{1\times1\times\frac{C}{N^2}}$$

$$\mathcal{K}'_{FR} = \{\mathcal{K}'_{fr}(0), \mathcal{K}'_{fr}(1), \mathcal{K}'_{fr}(2), \dots, \mathcal{K}'_{fr}(w'), \dots, \mathcal{K}'_{fr}(N^2-1)\} \quad \text{Eq. 3.26}$$

$$where\ \mathcal{K}'_{FR}(w') = \left\{\mathcal{K}'_{FR}(0), \mathcal{K}'_{FR}(1), \mathcal{K}'_{FR}(2), \dots, \mathcal{K}'_{FR}(\boldsymbol{z}), \dots, \mathcal{K}'_{FR}\left(\frac{C}{N^2}\right)\right\}$$

3.3.3 Principles of Adaptive DCT Pointwise Convolutional Kernel

Several conditions govern the formulation of the Adapt-DCT pointwise convolution kernel, and it must be satisfied in conjunction with the modified DCT basis functions. The conditions are as below:

(i) The number of channels (C) of the final DCT domain pointwise convolution kernel ($\mathcal{K}'_{FR}$) should always be equal to or larger than the number of modified DCT basis functions of frequency bases (N^2).

$$C \geq N^2$$

(ii) The number of channels (C) of the final DCT domain pointwise convolution kernel ($\mathcal{K}'_{FR}$) should always be divisible by the number of modified DCT basis functions of frequency bases (N^2). '%' indicates the modulo operation.

$$C \,\%\, N^2 \equiv 0$$

In the sense that any of the above conditions cannot be satisfied during the initialisation of the Adapt-DCT pointwise convolution kernel, the $\mathcal{K}'_{FR}$ will be undefined.

$$\mathcal{K}'_{FR} = \emptyset \begin{cases} C < N^2 \\ C \,\%\, N^2 \neq 0 \end{cases}$$

3.3.4 Adaptive DCT Pointwise Convolutional Neural Network

In this section, the full process of the adaptive DCT pointwise convolution is established. It covers the forward and backward passes of the Adapt-DCT convolution. The Adapt-DCT convolutional layer can be treated as a plug-and-play layer to replace any convolutional layer in a CNN. Specifically, the Adapt-DCT convolution is used in compressed domain CNN with an input consisting of a DCT image. The forward and backward pass can operate flawlessly since the kernel will not be affected by the DCT process.

The forward pass is computed by performing pointwise convolution between the pointwise convolutional kernel over the DCT input or DCT feature map. The process is shown in Figure 3.18. The kernel is denoted as $\mathcal{K}'_{FR}$. The input feature map is represented by X and the output is Y. The '$H \times W$' is the height and width of the feature map, C is the number of channels, where $C = \mathrm{C}_i \times \mathrm{C}_j$. Since the operation is pointwise convolution, hence the spatial size $(H \times W)$ of the input and output will be the same. Thus, the forward pass is:

$$Y = \mathcal{K}'_{FR} * X \qquad \text{Eq. 3.27}$$

$$\mathcal{K}'_{FR} \in \Re^{1\times1\times C}$$
$$X \in \Re^{H\times W\times \mathrm{C}_i}$$
$$Y \in \Re^{H\times W\times \mathrm{C}_j}$$

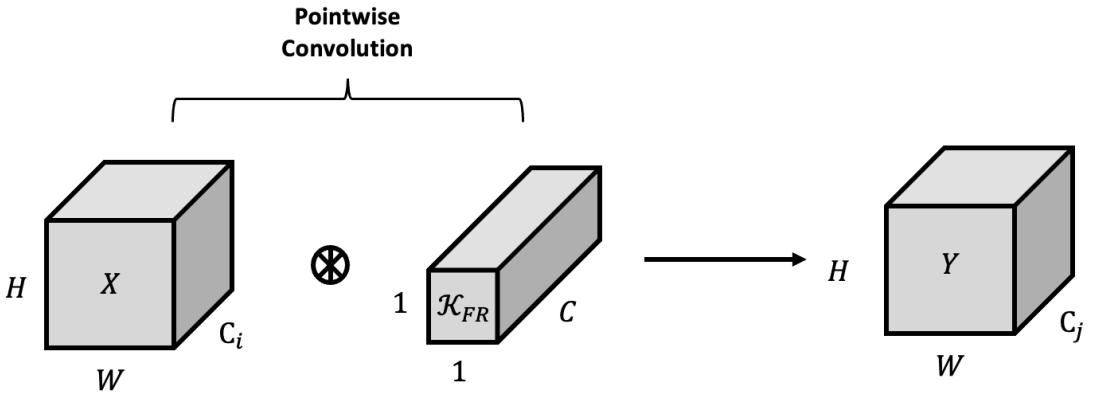


Figure 3.18: Pointwise convolution between a convolutional kernel and the DCT input.

In the forward pass, the input is processed through the CNN and a loss is computed at the end. The chosen loss function for this research is the categorical cross-entropy loss. It is widely implemented in the context of multi-class classification problems. Let $\mathcal{J}$ denote the loss function for the classification prediction with N total number of classes. y_i indicates the true categorical label for class i, encoded as a one-hot vector, whereas $\hat{y}_i$ resembles the predicted probability distribution for the same class. $\hat{y}_i$ is computed by passing Y through a fully connected layer followed by the Softmax activation function. The expression of the loss function is defined in Equation 3.28.

$$\mathcal{J} = -\sum_{i=1}^{N} (y_i) \cdot \log (\hat{y}_i) \qquad \text{Eq. 3.28}$$

To facilitate the update of convolutional filters, it involves minimizing the loss by backpropagating the gradient of the loss function concerning the filers back through the network. From the proven formula for backpropagation in a convolutional layer using the chain rule, the specific gradient of interest is the convolution between the input X and the gradient loss from the subsequent layer $\frac{\partial \mathcal{J}}{\partial Y}$. Consequently, the relationship between the gradient of the loss function concerning the kernel $\frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}}$ in the current layer with X and $\frac{\partial \mathcal{J}}{\partial Y}$ is written as:

$$\frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}} = Convolution \left[X * \frac{\partial \mathcal{J}}{\partial Y} \right] \qquad \text{Eq. 3.29}$$

Since the pointwise convolutional kernel ($\mathcal{K}'_{FR}$) is obtained from the adaptive weights ($\mathcal{K}_{TS}$), to optimize the initial $\mathcal{K}_{TS}$, it is required to backpropagate the loss from $\mathcal{K}'_{FR}$ to $\mathcal{K}_{TS}$. Specifically, the partial derivative of $\frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}}$ is used to find $\frac{\partial \mathcal{J}}{\partial \mathcal{K}_{TS}}$. By using the formula above to obtain $\frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}}$, $\frac{\partial \mathcal{J}}{\partial \mathcal{K}_{TS}}$ can be computed. Provided that the Adapt-DCT kernel is weighting each of the DCT basis functions across the spatial dimension individually, therefore the same spatial size of the Adapt-DCT kernel and the basis function applies. Two intermediate steps in forward 2D-DCT are conducted to produce $\mathcal{K}'_{FR}$ from $\mathcal{K}_{TS}$, specifically element-wise multiplication of Adapt-DCT kernel ($\mathcal{K}_{TS}$) with

the modified DCT basis functions ($\mathbb{B}$') and spatial summation of the resultant tensor $\mathcal{K}'_{TS}$. The representation can be described as below:

$$\mathcal{K}'_{TS} = \mathcal{K}_{TS} \otimes \mathbb{B}' \quad \text{Eq. 3.30}$$

$$\mathcal{K}'_{FR} = \sum_x \sum_y [\mathcal{K}'_{TS}]_{(x,y)} \quad \text{Eq. 3.31}$$

$\mathcal{K}'_{TS}$ is the modulated form of $\mathcal{K}_{TS}$ with the same shape. It is obtained after conducting element-wise multiplication of $\mathcal{K}_{TS}$ with the modified DCT basis function at a specific frequency base. It is important to note that the notation above is the intermediate representation of 2D-DCT whereby the spatial dimension of both the $\mathcal{K}_{TS}$ and $\mathbb{B}'$ exists. The constant term of $\mathbb{C} = \frac{2}{N} \cdot \xi(u) \cdot \xi(v)$ is neglected as it will not affect the backward pass of the loss function. To compute $\mathcal{K}_{TS}$ from $\mathcal{K}'_{FR}$, inverse 2D-DCT is performed without computing the summation.

$$\mathcal{K}_{TS} = \mathcal{K}'_{FR} \otimes \mathbb{B}' \quad \text{Eq. 3.32}$$

This concept is applicable for the backward pass algorithm since 2D-DCT between $\mathcal{K}_{TS}$ and $\mathcal{K}'_{FR}$ is a linear transformation. Hence, the computation of its partial derivatives loss function and backward pass is straightforward. From Equation 3.32, differentiate loss function '$\mathcal{J}$' with respect to the kernel '$\mathcal{K}$' governs the following:

$$\frac{\partial \mathcal{J}}{\partial \mathcal{K}_{TS}} = \frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}} \otimes \mathbb{B}' \quad \text{Eq. 3.33}$$

$$For\ each\ \frac{\partial \mathcal{J}}{\partial \mathcal{K}_{TS}(x,y,w')} = \frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}(w')} \otimes \mathcal{B}'(x,y,w) \quad \text{Eq. 3.34}$$

By substituting $\frac{\partial \mathcal{J}}{\partial \mathcal{K}'_{FR}}$ obtained from Equation 3.29 into Equation 3.33, each of the corresponding backward pass functions $\frac{\partial \mathcal{J}}{\partial \mathcal{K}_{TS}}$ can be computed to optimise the Adapt-DCT kernel ($\mathcal{K}_{TS}$). The layers other than Adapt-DCT convolution such as fully connected layers will be computed as usual.

3.3.5 Spatial Upscaling of Adaptive DCT Kernel

In the spatial domain CNN, generally the spatial dimension of the convolution kernel consists of odd numbers such as 3×3, 5×5, and so on. With the partial forward 2D-DCT to transform the Adapt-DCT kernel ($\mathcal{K}_{TS}$) into the pointwise convolution kernel, it allows the upscaling of the initial Adapt-DCT kernel to an arbitrary spatial size, but there are several principles to abide which concern the DCT partition.

Commonly, the spatial dimension ($N \times N$) of an Adapt-DCT kernel is upscaled in a way such that it is a multiplier of 2. This is because of the need to adhere to the original partition size of the DCT image. Additionally, it is intended to achieve division with zero remainder with the channel (C) over the frequency bases of the DCT basis functions (N^2). All the possible spatial upscaling of the ADAPT-DCT kernel with reference to the DCT partition of the image is provided in Table 3.4.

Table 3.4: Spatial upscaling of Adapt-DCT kernel concerning the DCT partition.

Spatial Size of $\mathcal{K}_{TS}$ / DCT Partition	1×1	2×2	4×4	8×8
2×2	✓	✓	✗	✗
4×4	✓	✓	✓	✗
8×8	✓	✓	✓	✓

As an example, with a DCT partition of 2×2, the maximum spatial upscaling of the $\mathcal{K}_{TS}$ is 2×2. This is because of a spatial size of $\mathcal{K}_{TS}$ larger than 2×2 does not symbolize nor represent the accurate DCT spatial base information. Moreover, it may overlap or cause aliasing effects on the spatial bases with other DCT partitions. From the experiments conducted, a larger spatial size of $\mathcal{K}_{TS}$ with a smaller DCT partition does not bring any performance improvement. The same concept is applicable even though a normal pointwise convolution layer exists before an Adapt-DCT convolution layer in compressed domain CNN. With a particular DCT partition, a larger spatial dimension ($N \times N$) of the Adapt-DCT kernel can accommodate more DCT basis

functions (subjected to $C \geq N^2$). Furthermore, by using more basis functions to represent a pointwise convolution kernel, more sampling provides more adaptive weighting flexibility. Thus, for a specific number of channels (C), an increasing spatial size ($N \times N$) of the initial Adapt-DCT kernel ($\mathcal{K}_{TS}$) will reduce the number of channel sets ($\frac{C}{N^2}$) allocated to each of the DCT basis functions of frequency base.

A few scenarios with a specific DCT partition of 8×8 are elaborated further to pursue an in-depth understanding of the concept of spatial upscaling of the Adapt-DCT kernel. Let the number of channels of the input and output feature maps be C_i and C_j, where $C_i = C_j = 8$. Consequently, the number of channels of the corresponding pointwise convolution kernel will be C. Table 3.5 presents a few variations of the spatial dimension of the Adapt-DCT kernel and their effects on the overall properties of other tensors.

Table 3.5: Variations of spatial upscaling on Adapt-DCT kernel with a DCT partition of 8×8

		Case 1 ($N = 2$)	**Case 2 ($N = 4$)**	**Case 3 ($N = 8$)**
Spatial Dimension of Adapt-DCT kernel	$N \times N$	2×2	4×4	8×8
Number of available frequency bases	N^2	4	16	64
Number of channel weights per frequency base	$\frac{C}{N^2}$	16	8	1
Shape of $\mathcal{K}_{TS}^{w'}$	$\Re^{N \times N \times \frac{C}{N^2}}$	$\Re^{2 \times 2 \times \frac{64}{4}} = \Re^{2 \times 2 \times 16}$	$\Re^{4 \times 4 \times \frac{64}{16}} = \Re^{4 \times 4 \times 8}$	$\Re^{8 \times 8 \times \frac{64}{64}} = \Re^{8 \times 8 \times 1}$
Shape of $\mathcal{K}_{TS}$	$\Re^{N \times N \times C}$	$\Re^{2 \times 2 \times 64}$	$\Re^{4 \times 4 \times 64}$	$\Re^{8 \times 8 \times 64}$
Shape of $\mathcal{K}_{FR}$ after spatial summation on $\mathcal{K}_{TS}$	$\Re^{1 \times 1 \times C}$	$\Re^{1 \times 1 \times 64}$	$\Re^{1 \times 1 \times 64}$	$\Re^{1 \times 1 \times 64}$

The above logic is applicable even with a DCT partition of 4×4 or 2×2. The difference is that with an increasing spatial dimension of the Adapt-DCT kernel, it will reduce the number of channel weights set ($\mathcal{K}_{TS}^{w\prime}$) per DCT basis function of frequency base ($\mathcal{B}'(w)$). From the standard scenarios as presented above, for any Adapt-DCT kernel with spatial dimension equal to any reasonable real number, i.e., $0 < N \leq DCT\ Partition$, computing forward 2D-DCT will produce a frequency domain pointwise convolution kernel with a consistent spatial dimension of 1×1 ($\mathcal{K}_{FR} \in \Re^{1\times1\times C}$). This can be seen in the last row in Table 3.5. Hence, it is safe to deduce that the spatial dimension (N) of the Adapt-DCT kernel will not affect the spatial dimension of the final frequency domain pointwise convolution kernel ($\mathcal{K}_{FR}$). But different N will require a different number of channel weights set ($\mathcal{K}_{TS}^{w\prime}$) per DCT frequency base.

The potential of spatial upscaling in an Adapt-DCT kernel serves several flexibilities in the CNN. The spatial dimension of an Adapt-DCT kernel can be increased to improve the sampling and partition steps of the pointwise convolution kernel. In other words, when more DCT basis functions are used to define a final pointwise convolution kernel, more flexibility in terms of fine-tuning and weighting each spatial and frequency base can be exercised. In contrast, the spatial dimension of an Adapt-DCT kernel can be decreased to reduce trainable parameters.

3.3.6 Depth-wise Level Optimisation of Adaptive DCT Pointwise Convolutional Kernel

This section discusses the depth-wise channel optimisation within the Adapt-DCT kernel. The optimisation strategy draws inspiration from the pruning technique applied to DCT. The underlying concept revolves around the fact that basis functions serve as a common connection between the resultant pointwise convolutional kernel and the adaptive weights. Unlike the approach used in the M-Skipped network, where the modifications took place on the feature map, this method focuses on modifying the DCT basis functions during the kernel composition. It is crucial to note that these two methods are not contradictory; rather, the key distinction lies in the former being focused on the feature map, while the present work is centred on the kernel formulation. By carefully pruning the higher-level frequency bands across the DCT basis functions and specifying the number of resulting channels, this optimisation ensures a particular frequency band will be weighted by similar or more sets of adaptive weights compared to the original case.

Under ordinary circumstances, the DCT basis functions carry a tensor shape of $\mathbb{B}' \in \Re^{N\times N\times N^2}$, where $N \times N$ represents the spatial dimension of the Adapt-DCT kernel as shown in section 3.3.2. The depth-wise channel across the adaptive weights is equally divided according to the number of DCT basis functions of frequency bases. The optimisation is applied on the layer dimension of the frequency base in the DCT basis function, denoted by the instance (w) in $\mathcal{B}(x, y, w)$.

Let the number of frequency bases be η, such that the corresponding DCT basis functions will carry a shape of $\mathbb{B}' \in \Re^{N\times N\times \eta}$. The parameter '$\eta$' represents the shape of the layer dimension of the frequency bases. This parameter is restrained by the conditions below where it specifies the usage boundary of 'η':

(i) The parameter 'η' should be larger than zero and smaller than the original layer shape of N^2.

$$0 < \eta < N^2$$

(ii) The parameter 'η' should be divisible by the total number of channels of the Adapt-DCT kernel 'C'.

$$C \,\%\, \eta \equiv 0$$

An extension example is elaborated from the scenario in the previous section to showcase the working of depth-wise channel optimisation on the Adapt-DCT kernel. With a DCT partition of 8×8, the spatial size of the Adapt-DCT kernel (N) is 8 whereas the number of channels (C) will be 64, such that $\mathcal{K}_{TS}$ carries $\Re^{8\times8\times64}$. Following the original formulation, the number of DCT basis functions of frequency bases along the layer dimension (N^2) shall be 64, and the respective channel weights per DCT basis function ($\mathcal{K}_{TS}^{w\prime}$) will carry a shape of $\Re^{8\times8\times1}$. With $\eta = 4$ replacing the original $N^2 = 64$, the optimised basis function $\mathbb{B}'$ will carry the shape of $\Re^{8\times8\times4}$ ($\Re^{N\times N\times\eta}$) instead of the former $\Re^{8\times8\times64}$ ($\Re^{N\times N\times N^2}$). This optimisation will correspond to an increase in the number of channel weights per DCT frequency base ($\mathcal{K}_{TS}^{w\prime}$) from the original $\Re^{8\times8\times1}$ ($\Re^{N\times N\times\frac{C}{N^2}}$) to $\Re^{8\times8\times16}$ ($\Re^{N\times N\times\frac{C}{\eta}}$).

By adopting the idea of DCT pruning, it involves a selective inclusion of frequency bases in the formation of the resultant kernel that contribute significantly to the major recognition properties. This approach preserves the spatial context of DCT basis functions and adaptive weights while optimizing the frequency bases. Instead of incorporating all available frequency bases, the optimisation selectively utilises a subset to compose the pointwise convolutional kernel. The primary outcome of this optimisation is to investigate the robustness throughout the learning phase by employing a reduced number of basis functions for kernel composition. It is aimed to observe how varying the number of DCT basis functions in kernel composition can influence performance.

3.4 Hybrid Modified Efficient Channel Attention

While the earlier Adapt-DCT CNN improved kernel flexibility for handling complex fine-grained patterns, how each feature channel contributes to FGVC in the compressed domain remains unclear. Therefore, adopting an attention mechanism is crucial to explore this dimension. Fine-grained features along the channel dimension are often better captured through attention mechanisms, which emphasise the most discriminative aspects of the data. However, the attention mechanisms typically introduce additional parameters which directly increases the computational requirements. The introduction of the HyMod-ECA complements the earlier model by integrating feature prioritisation and interaction while reducing the number of parameters. This balance is critical for enhancing resource efficiency, leading to overall performance improvements and model optimisation.

In this section, the main conceptual differences between former ECA on local cross-channel interactions and the proposed HyMod-ECA on intra-group DCT channel interactions are established. Later, the formulation and implementation details of the HyMod-ECA will be elaborated on. Towards the end, a demonstration of implementing the HyMod-ECA on top of the former Adapt-DCT CNN will be explained.

The original ECA highlights individual local cross-channel interactions whereby every local channel and its neighbouring relationship and interaction is considered. However, in the frequency domain, it is suggested that not every single channel and its neighbouring channels correlate. This is because DCT decorrelates spatial context and concentrates the frequency features towards several coefficients in the channel dimension. Therefore, different from the conventional ECA, HyMod-ECA considers the interaction within a larger DCT channel group. Each of the DCT channel groups across the output of a convolution block carries a different level of frequency information in the feature space. By considering the interaction between DCT channel groups at the output of each convolution block, it is suggested that the connection between hierarchical discriminative features can be established to ease FGVC.

Let the output of an intermediate feature map from any convolutional block be $X \in \mathbb{R}^{H \times W \times C}$, where H, W, C are the height, width, and number of channels. To produce an aggregated 1D vector of $\bar{X} \in \mathbb{R}^{1 \times 1 \times C}$ based on X, the original channel-wise GAP (denoted as $GAP(*)$) is used to average each spatial map of X. GAP is utilized over GMP for its ability to emphasize generalization and robustness of the attention mechanism to

avoid overfitting, such that every DCT channel group is equally considered. $\bar{X}_c$ below indicates the aggregated feature after computing GAP at the channel c.

$$\bar{X}_c = GAP(X_c) = \frac{1}{WH}\sum_{i=1,j=1}^{W,H} X_{c,ij} \quad \text{Eq. 3.35}$$

From the original ECA, an effective method to apply weight sharing across channel groups is through computing fast 1D convolution. This technique is further extended in this section by modifying the stride size of the 1D kernel to reduce the computational complexity and to emphasize intra-group DCT channel interactions. Figure 3.19 briefly provides a fundamental concept on the formulation of DCT channel groups. The aggregated DCT vector ($\bar{X}$) is equally split into several DCT channel groups, with each respective group denoted as $\bar{X}_w \in \mathbb{R}^{1\times1\times f}$, where w and f represent the sequence instance of the DCT channel group and the channel depth of each DCT channel group. $\bar{X}_w$ is referred to as an 'aggregated DCT channel group'.

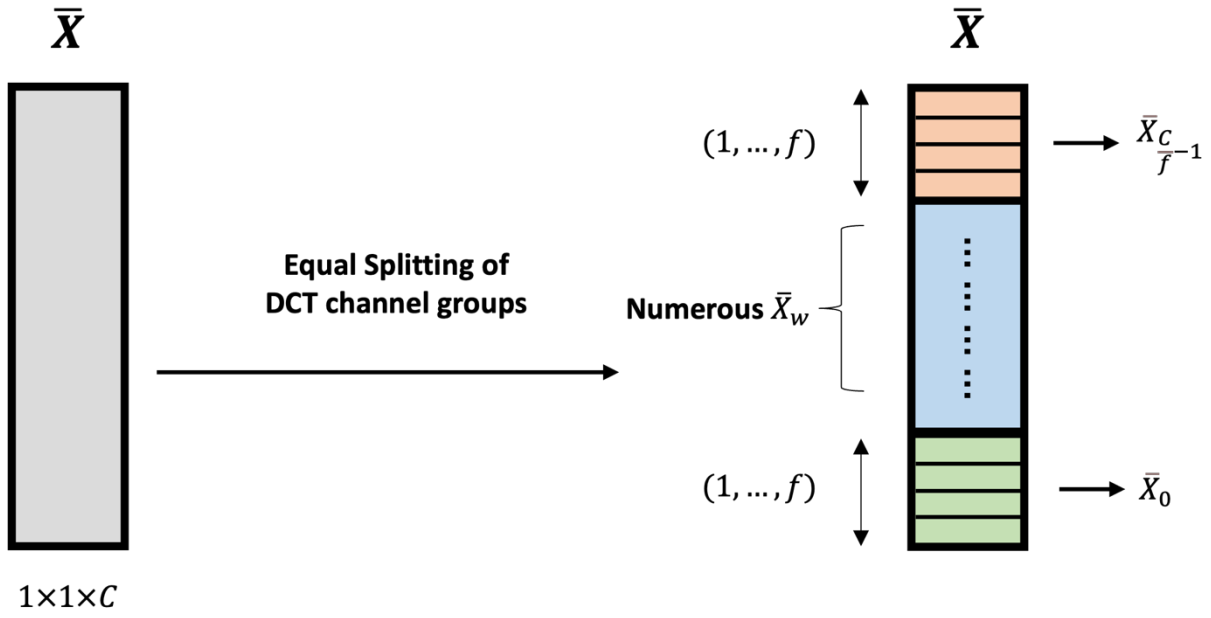


Figure 3.19: Aggregated DCT channel groups (X_w), produced by the original 1D tensor acquired by performing GAP over the input feature map X.

Since the total number of DCT channels available from the aggregated feature $\bar{X}$ is C, by conducting equal channel dissemination on $\bar{X}$, the total number of available DCT channel groups is equal to $\frac{C}{f}$. The same concept is also applied to the input feature map X to obtain the same form of DCT input feature channel groups, denoted as $X_w \in \mathbb{R}^{H\times W\times f}$. X is later used to multiply with the attention weights following the order of

DCT channel groups. Each of the DCT channel groups (X_w) will be multiplied with one of the attention scalar weights acquired from $\bar{A}$. Multiple sets of attention maps are needed to weight each DCT channel group respectively. To acquire an aggregated DCT channel group from the original aggregated feature $\bar{X}$, the below notation can be considered:

$$\bar{X} \in \left\{ \bar{X}_0, \bar{X}_1, \dots, \bar{X}_w, \dots, \bar{X}_{\frac{C}{f}-1} \right\}, where\ w\ elapses\ from\ 0\ to\ \frac{C}{f} - 1.$$

Similarly, the DCT input feature map X can be visualized in the same fashion where each of the $\bar{X}_w$ is produced from $\bar{X}$. The DCT input feature map corresponds to its channel groups and can be written as:

$$X \in \left\{ X_0, X_1, \dots, X_w, \dots, X_{\frac{C}{f}-1} \right\}, where\ w\ elapses\ from\ 0\ to\ \frac{C}{f} - 1.$$

The intention to perform equal channel dissemination on $\bar{X}$ is to obtain the subsequent scalar attention weights $\bar{A}$ produced via fast 1D convolution. It will be used to multiply with the corresponding X_w to achieve a linear correspondence of cross-DCT channel group attention. It also aimed to ease the fast 1D convolution to adopt consistent kernel and stride size.

Let K be the 1D convolution kernel that slides across the channel dimension $\bar{X}$. Figure 3.20 briefly showcases the concept of 'local cross channel' versus 'intra DCT channel groups' interaction by conducting 1D convolution. Figure 3.20 (a) shows the intra-DCT channel group interaction while Figure 3.20 (b) is the original ECA. Due to the difference in the stride size of the 1D convolution, the final produced attention weights for HyMod-ECA ($\bar{A}$) are smaller in terms of the depth-wise channel dimension when compared with the original ECA which carries the same shape ($\bar{A}'$) as the aggregated feature $\bar{X}$.

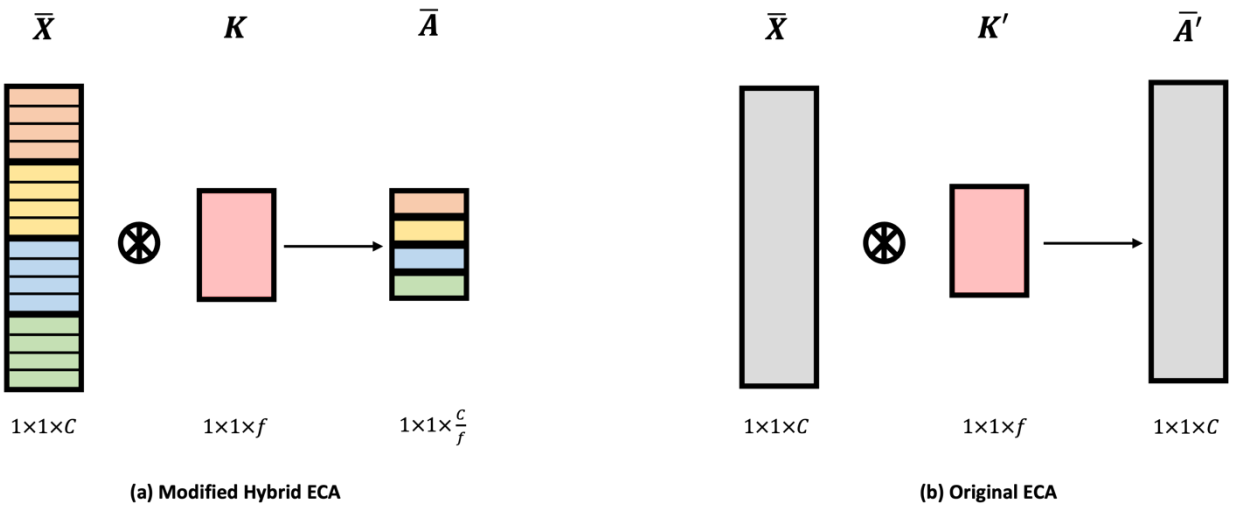


Figure 3.20: Comparison between (a) HyMod-ECA versus (b) the original ECA.

The 1D convolution kernel K carries a set of convolutional properties of $K_{(f,k,s,p)}$, where (f, k, s, p) indicates number of input filters, spatial size of kernel, stride size, padding size respectively. Since the convolution is of 1D nature intending to capture cross DCT channel group relationship, therefore the spatial size is 1 and the stride size is set to the same as the number of kernel filters (f) within the same group. The padding is not applied in the HyMod-ECA to produce the DCT channel group weights; thus, the padding size is set to zero. Rewriting the kernel $K_{(f,k,s,p)}$ with the above criteria will result in $K_{(f,1,f,0)}$, where $K_{(f,1,f,0)} \in \mathbb{R}^{1\times1\times f}$.

Conducting fast 1D convolution followed by the Sigmoid activation function (σ) across the aggregated feature $\bar{X}$, the kernel K will produce the subsequent attention weights $\bar{A} \in \mathbb{R}^{1\times1\times\frac{C}{f}}$, which we referred to as the "DCT channel group attention weights". The interaction of the intra-group DCT channel is governed by performing fast 1D convolution via the following formulation:

$$\bar{A} = \sigma\left[\bar{X} * K_{(f,k,s,p)}\right]_{conv\_1D} = \sigma\left[\bar{X} * K_{(f,1,f,0)}\right]_{conv\_1D} \quad \text{Eq. 3.36}$$

Similar to $\bar{X}$, the attention weights $\bar{A}$ can also be seen as an individual group denoted as $\bar{A}_w$. Each $\bar{A}_w$ is produced by element-wise multiplication and summation between the 1D convolution kernel $K_{(f,1,f,0)}$ and the aggregated DCT channel set $\bar{X}_w$.

Since both the $\bar{A}_w$ and K carries the same shape of $\mathbb{R}^{1\times1\times f}$, the instantaneous convolution between them will produce a single scalar of $\bar{A}_w \in \mathbb{R}^{1\times1\times1}$. $\bar{A}_w$ resembles the attention weightage that is applied towards the respective input feature group of X_w. The relationship between A and $\bar{A}_w$ can be written as:

$$\bar{A} \in \mathbb{R}^{1\times1\times\frac{C}{f}}$$

$$\bar{A} \in \left\{\bar{A}_0, \bar{A}_1, \dots, \bar{A}_w, \dots, \bar{A}_{\frac{C}{f}-1}\right\}, where\ w\ ranges\ from\ 0\ to\ \frac{C}{f}-1.$$

$$For\ each\ \bar{A}_w = \sum[\bar{X}_w \otimes K_{(f,1,f,0)}] \qquad \text{Eq. 3.37}$$

The technique of grouping DCT channels and conducting 1D fast convolution to capture intra-group DCT channel interactions is significant. Through grouping the DCT channel sets, it creates a connection between the frequency coefficients in the feature space. It is more robust to consider several frequency features as opposed to individual ones in discriminative learning. This is because a collective of DCT channel groups can correlate with each other in detecting subtle differences. Convolving across DCT channel groups with a larger stride and kernel size focuses on capturing the interaction within several frequency coefficients in producing higher-level features. With the application of HyMod-ECA towards the lower-level DCT channel group, it can be seen as capturing a collective of interactions between basic frequency features. While the convolution process in the higher-level DCT channel group captures the discriminative relationship between more complex and higher-level abstract frequency feature groups.

Furthermore, the hybrid 1D fast convolution performed in HyMod-ECA involves lesser computational complexity as compared to the original ECA due to its larger kernel and stride size. Given that f denotes the filter depth while C represents the channel depth, the computational complexity and their ratio for 1D fast convolution of both the original and the HyMod-ECA are:

$$Original\ ECA: O_{ECA} = O[(d-k+3)(k-1)k] = O[(C-f+3)\cdot(f-1)\cdot f] \quad \text{Eq. 3.38}$$

$$Hybrid\ Modified\ ECA: O_{HyMod-ECA} = O[d\cdot(k-1)] = O[C\cdot(f-1)] \quad \text{Eq. 3.39}$$

$$Computation\ complexity\ ratio: \frac{O_{ECA}}{O_{HyMod-ECA}} = k - \frac{[k\cdot(k-3)]}{d} = f - \frac{[f\cdot(f-3)]}{C} \quad \text{Eq. 3.40}$$

Full derivations are shown in the appendix. It is clear from the derivation that the HyMod-ECA convolution consists of $f - \frac{[f\cdot(f-3)]}{C}$ fewer computations when compared with the original ECA, thus achieving efficient and hybrid fast 1D convolutions. At this point, it is important to recall that in HyMod-ECA convolution, the DCT channel group depth dimension (f) is the same as the length of the kernel (f) and the convolution stride size (s). Towards the end, to compute the output feature map with HyMod-ECA denoted as $Y \in \mathbb{R}^{H\times W\times C}$, the DCT attention weights $\bar{A}$ are multiplied with the input feature map X. Specifically, each of the single scalar of $\bar{A}_w$ were multiplied with the corresponding DCT channel group of input X_w, and later concatenation was applied to compute the final attention output feature.

$$Compute\ Attention\ Output: Y \in \mathbb{R}^{H\times W\times C}$$

$$\begin{aligned} Y = X \otimes \bar{A} &= concat\{X_w \cdot \bar{A}_w\}_{w=0}^{w=\frac{C}{f}-1} \\ &= \left\{X_0 \cdot \bar{A}_0, X_1 \cdot \bar{A}_1, \dots, X_{\frac{C}{f}-1} \cdot \bar{A}_{\frac{C}{f}-1}\right\} \end{aligned} \quad \text{Eq. 3.41}$$

Figure 3.21 explains the abstract form of the attention mechanism where $\bar{A}$ was applied to the input feature map. The input X is partitioned into several groups of DCT channel sets (X_w) following the $\bar{X}$ fashion, as shown in different colours. Each of the X_w is applied with the corresponding scalar attention of $\bar{A}_w$ to produce a subsequent DCT channel group of feature output Y_w at the same location.

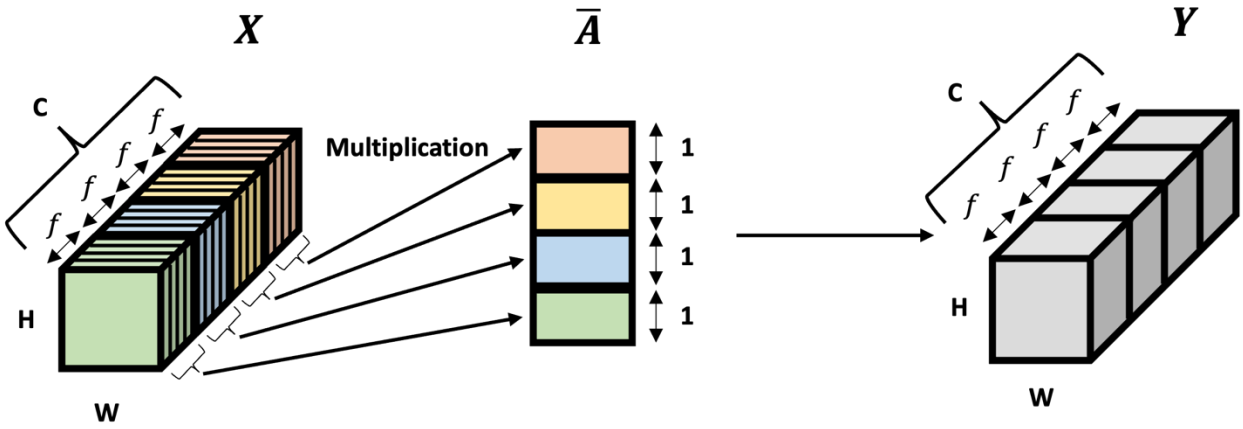


Figure 3.21: Abstract representation of the application of attention weights ($\bar{A}$) from HyMod-ECA to the input feature map (X) through a DCT channel group point of view.

The motivation of applying a single attention weight of $\bar{A}_w$ towards the designated input DCT channel group of X_w lies beyond providing attention to capturing the interactions within discriminative features at specific frequency ranges. The fast 1D convolution output ($\bar{A}$, also the attention weights) carries a shape of $(N, 1, \frac{C}{f})$, where N refers to the batch size. $\bar{A}$ carries the relative importance between each DCT feature group across the channel direction. By multiplying a DCT channel group (X_w) with its corresponding attention weight (X_w), less discriminative information will be suppressed. This concept addresses the research gap in considering DCT channel groups in the compressed domain FGVC. Henceforth, the final hybrid modified ECA can be represented with the below formula, with $\Phi(*)$ indicating the overall computation:

$$Y = X \otimes \sigma\left\{[GAP(X)] * \left[K_{(f,1,f,0)}\right]\right\}_{conv_{1D}} \quad \text{Eq. 3.42}$$

$$Y = \Phi(X)$$

The key differences between the original ECA and the HyMod-ECA are summarised in Table 3.6.

Table 3.6: Key differences between original ECA and HyMod-ECA.

Variations\Specifications	**Original ECA**	**Hybrid Modified ECA**
Kernel shape	$(N, 1, 3)$	$(N, 1, f)$
Stride size	1	f
Padding	1	0
Attention weights depth ($\bar{A}$)	*Same as input*	$\frac{C}{f}$
Computation complexity	$O[(C - f + 3) \cdot (f - 1) \cdot f]$	$O[C \cdot (f - 1)]$
Trainable parameters of K	$Normally$ 3	f

From the empirical formulation of hybrid modified ECA, the number of channels per DCT channel group (f) is an important variable that can have a significant impact on the overall attention mechanism in terms of computation and the learning of frequency feature group. The 'f' determines the 1D convolution kernel size. It hinders an indirect relationship with the 1D convolution computational complexity. With a larger size of 'f', the computational complexity reduces and vice versa. A larger 'f' with a larger number of trainable parameters also indicates the model's increased ability to govern more global relationships within DCT channel groups instead of localised DCT group features in FGVC.

Based on a preliminary set of exploratory experiments, several primary conditions are set up to restrict the initialisation of 'f' to ensure the proper working of the attention module. The conditions are as below:

(iii) The number of channels per DCT channel group (f) should always be a positive even integer. '%' indicates the modulo operation.

$$f \,\%\, 2 \equiv 0, 2 \leq f < C$$

(iv) The number of channels per DCT channel group (f) should always be fully divisible by the channel depth (C).

$$C \,\%\, f \equiv 0$$

These limits are to ensure that the output produced from the 1D convolution can be a whole number, and that the convolution process can operate without any centre shift. Besides, by doing so, the attention weights can be properly assigned to each input DCT channel group to achieve direct correspondence. In all the experiments, the number of channels per DCT channel group (f) is obtained by dividing the channel number by 4, following the partitioning technique earlier applied towards the input DCT on viewing the tensor as a sequential stack of LMH-DCTC structure.

In short, the preliminary effort on developing a hybrid modified ECA on the DCT domain for FGVC showcases the importance of fundamental multi-frequency intra-group information exchange. The multiplication of the scalar attention weight with the respective DCT channel group retains the correspondence of the DCT channel group and the output, avoiding information mixing between different DCT channel groups. The attention process will later produce a series of amplified DCT channel groups that potentially contain essential fine-grained features that distinguish itself from other classes to improve FGVC in the DCT domain.

3.4.1 Hybrid Modified ECA on Intra-Group DCT Channel Interaction

To demonstrate the workings of the hybrid modified ECA (HyMod-ECA) on the adaptive DCT (Adapt-DCT) CNN, a brief recap is initially provided on the M-Skipped-3 architecture from section 3.2.2. Then, the integration of HyMod-ECA with Adapt-DCT CNN is formulated. VGG-16 is used as the backbone for all the experiments for consistency. Further down, several experimental setups with a few variations are designed to prove the working concept of HyMod-ECA. The output of an intermediate feature map from the l^{th} convolution block is:

$$Y^l = X^{l+1} = G(X^l)$$

Where $G(*)$ denotes the function of a convolution block. Y^l is the output from l^{th} convolution block with the input of X^l. Y^l is also treated as the input towards $l+1$ convolution block, indicated by X^{l+1}. X_{M-DCTC} is the medium DCTC partition obtained from the original DCT input image. The final feature map (Y^{final}) is acquired by concatenating (denoted with $\oplus$) a shallowly convolved M-DCTC (denoted as $\mathcal{g}(*)$) and the output from the final convolution block, denoted by Y^L, where the total number of available convolution blocks is $l = L$.

$$Y^{final} = Y^L \oplus Y^{M-DCTC} = G(X^L) \oplus \mathcal{g}(X_{M-DCTC})$$

Specifically, in VGG-16 where a typical 3 convolution block is employed, the HyMod-ECA module is applied towards the feature map output from convolution blocks 1 and 2, and the final feature map of Y^{final}. A simple block diagram is shown in Figure 3.22:

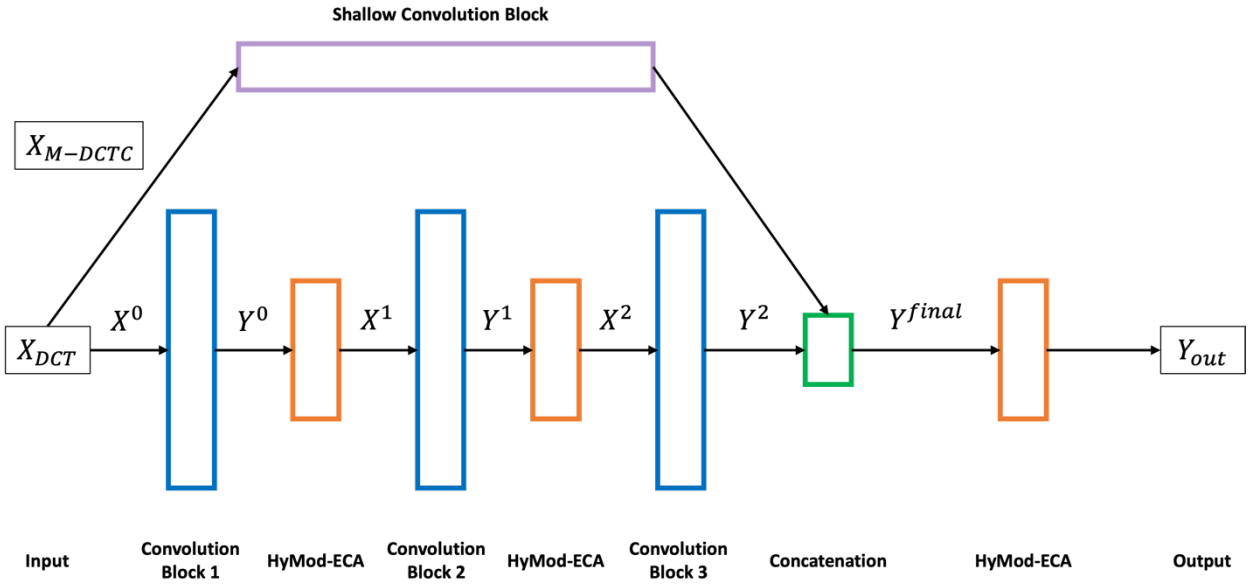


Figure 3.22: Implementation of HyMod-ECA integrated with Adapt-DCT CNN with a baseline model of VGG-16.

For convolution block 1 ($l = 0$) and 2 ($l = 1$), the following formulation applies:

$$Y_{att}^{l} = \Phi(G(X^{l}))$$

Y_{att}^{l} is the intermediate output feature with an associated attention mechanism from the l^{th} convolution block, Φ is the HyMod-ECA module. To obtain the final feature map (Y_{out}) before forwarding to fully connected layers and classifier:

$$Y_{out} = \Phi[Y^2 \oplus \mathcal{g}(X_{M-DCTC})]$$

Y^2 is the output feature map from the third convolution block. For input towards convolution block 2 (X^1) and 3 (X^2), where attention is applied towards the output from the previous convolution block:

$$Y_{att}^{0} = X^1, Y_{att}^{1} = X^2$$

The amplified intermediate output Y_{att}^{l} will be forwarded to the next block ($l + 1$) for convolving with the Adapt-DCT kernel instead of the original output (Y^l), which involves the Adapt-DCT kernel $\mathcal{K}_{TS}^{l+1}$ and the relative pointwise convolution kernel $\mathcal{K}_{fr}^{l+1}$, where $\mathcal{K}_{fr}^{l+1} = \Psi\{\mathcal{K}_{TS}^{l+1} \cdot \mathbb{B}\}$ where the function 'Ψ' represents the application of element-wise multiplication and spatial summation:

$$Y^{l+1} = \mathcal{K}_{fr}^{l+1} * X^{l+1} = \mathcal{K}_{fr}^{l+1} * Y_{att}^{l}$$

Table 3.7 shows the integration of HyMod-ECA with the Adapt-DCT CNN on VGG-16. The kernel properties (f, k, s, p) indicate the (number of filters, spatial size of the kernel, stride size, and padding size) respectively. The HyMod-ECA modules are added into three slots of the model, denoted by K_{att}^{l} in Table 3.7. Specifically, the attention is added after the max pooling layer from the convolution blocks 1 and 2, and after the concatenation of convolution block 3 output with the M-Skipped feature embedded before the classifier.

Table 3.7: Architecture of the Adapt-DCT CNN of VGG-16 with the integrated HyMod-ECA module.

Block layer	**Kernel** $[f, k, s, p]$	**Convolution type**	**Details**
Convolution block 1	[256, 1, 1, 0] x 3	Pointwise convolution	Activation function: PReLU
2D max pooling	[-, 2, 2, 0]	-	-
HyMod-ECA (K_{att}^{0})			
Convolution block 2	[512, 1, 1, 0] x 3	Pointwise convolution	Activation function: PReLU
2D max pooling	[-, 2, 2, 0]	-	-
HyMod-ECA (K_{att}^{1})			
Convolution block 3	[512, 1, 1, 0] x 3	Pointwise convolution	Activation function: PReLU
2D max pooling	[-, 2, 2, 0]	-	-
Concatenate & reshape	-	-	Concatenate with M-skipped output and reshape into a 1D tensor
HyMod-ECA (K_{att}^{2})			
Classifier	-	-	Class number as output

With HyMod-ECA applied towards the feature output (Y_{att}^{l}) from convolution blocks 1 and 2, the low and medium-level DCT feature groups are regulated in such a way that critical interactions and relationships within fine-grained features of DCT nature can be captured. With the attention output feeding towards the next convolution block as input, plus the combination of Adapt-DCT convolution, adaptive DCT learning on top of the DCT channel group can be achieved. The final attention output of Y_{out} contains DCT channel groups of early low-level M-DCTC features and high-level general fine-grained features. It helps to regulate the learning process such that attention is fairly applied to the essential M-DCTC feature groups and the global context based on a DCT representation. In other words, intra-group DCT channel interactions between general fine-grained context and high-level DCT feature sets can be learned to enrich complex high-level learning of DCT domain CNNs.

3.5 Experimental Design

This section outlines the essential components of the experimental setup and its prerequisites. The initial segment explains the datasets employed, their processing methodology, the chosen performance metrics, and the requisite system setup specifications. The subsequent segment elaborates on the specific experimental setups for each method discussed in the preceding section.

3.5.1 Characteristics and Processing of Datasets

The primary aim of this research is to establish a foundational understanding of compressed domain FGVC. FGVC is designed to excel in distinguishing between fine-grained subcategories within a broader category, where the subcategories are characterized by more specific datasets featuring subtle distinctions. Certain datasets, such as the one focused on butterflies, encompass both discriminative and global features, while others solely emphasise fine-grained features. Consequently, it contributes to a collection of a well-balanced mixture of robust datasets. In other words, the chosen datasets are therefore deemed resilient and well-aligned with the interest of this research. This research initially concentrates on small-scale FGVC datasets, specifically those with fewer than 50 classes as a foundational phase. Subsequently, a larger FGVC dataset (Oxford Flowers) and some general datasets (CIFAR and Mini ImageNet) are employed to assess the algorithm's robustness. In comparison to other benchmark datasets, the selected datasets are notable for their robustness, small scale, and limited number of images per class. This characteristic renders them highly suitable for facilitating the achievement of the research objectives thus formulating conclusions. To be more specific, eight medium-sized FGVC datasets below 50 classes, one large-sized FGVC dataset comprising 103 classes, and three general datasets, all fully annotated, were meticulously chosen. Notably, while the Covid-19 dataset consists of grayscale images, the other datasets contain colour images with RGB channels. The FGVC datasets collectively represent numerous classes that span fine-grained classifications of various subcategory species, as detailed in Table 3.8. Furthermore, Figure 3.23 illustrates examples of classes from these datasets, providing a visual representation of the fine-grained classification pursued in this research.

Table 3.8: Number of classes, dataset genre, and the source(s) of various datasets.

Datasets	# of Classes	Dataset Genre	Source
Covid-19	3	FGVC	[160]
Sheep Breed	4	FGVC	[161][162]
Flowers	5	FGVC	[163]
Leeds Butterfly	8	FGVC	[164]
Monkey	10	FGVC	[165]
Spider Breed	15	FGVC	[166]
Snake Breed	35	FGVC	[167]
Butterfly	50	FGVC	[168]
Oxford Flowers	102	FGVC	[169]
CIFAR10	10	General	[170]
CIFAR100	100	General	[170]
Mini ImageNet	100	General	[171]

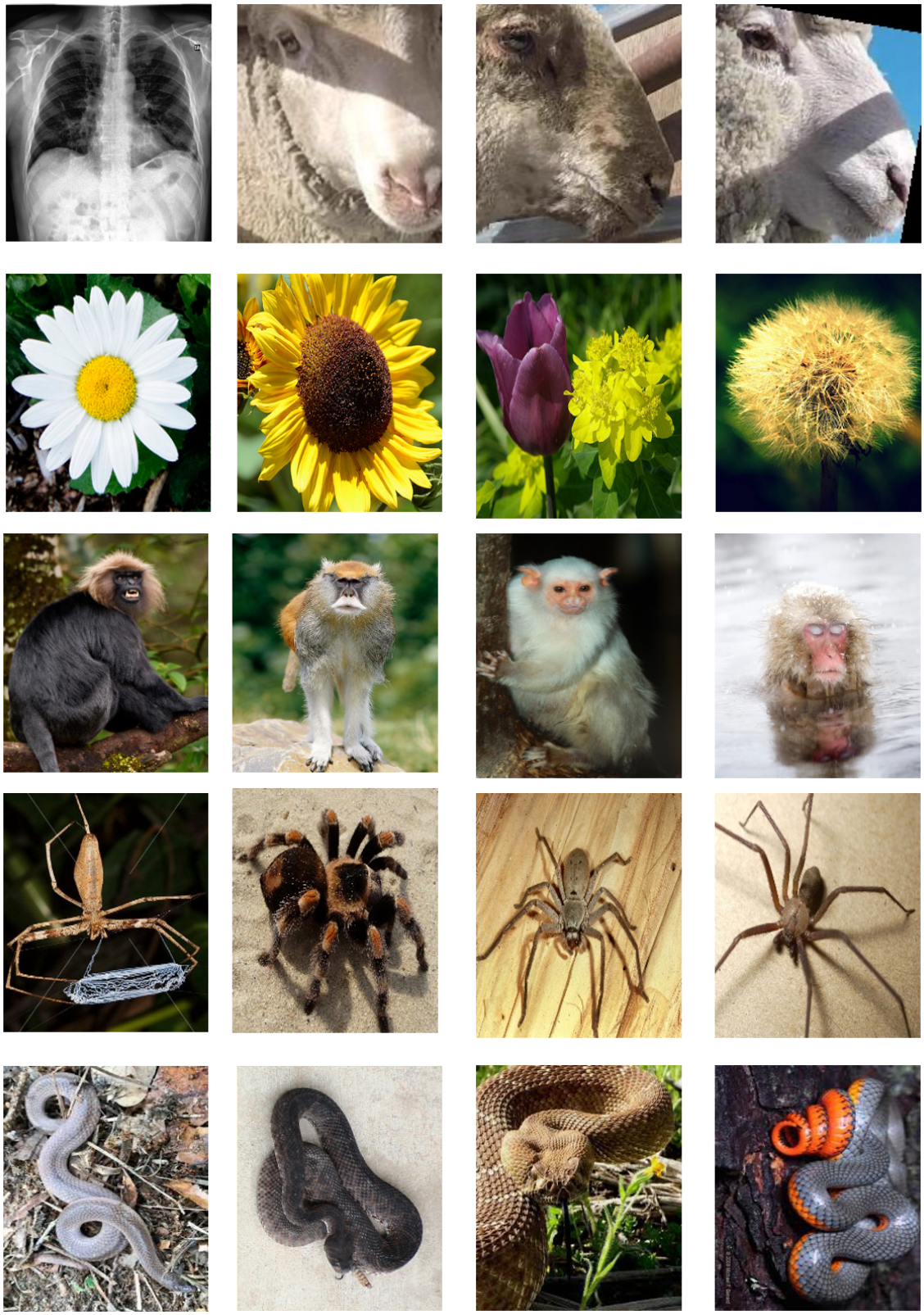


Figure 3.23: A small subset of images from the datasets [160][161][162][165].

The rationale behind selecting different datasets for evaluating the M-Skipped DCT-CNN, the Adapt-DCT CNN, and the HyMod-ECA DCT-CNN lies in tailoring the dataset choices to specific objectives and methodologies, thereby enriching the depth and breadth of this study. For the foundational phase of understanding compressed domain FGVC, the focus was on five small-scale datasets with fewer than 50 classes. It provides a controlled environment for initial insights into the M-Skipped DCT-CNN algorithm. This deliberate choice eases the process of reaching meaningful conclusions, given the smaller number of classes. While transitioning to the exploration of the Adapt-DCT CNN, a diverse set of datasets was employed. Six medium-sized FGVC datasets (Covid19-3C grayscale, Flowers-5C, Monkey-10C, Spider-15C, Snake Breed 35C, Butterfly-50C), a large-sized FGVC dataset (Flowers-104C), and general datasets (CIFAR10, CIFAR100, Mini Image Net 100C) were strategically chosen. The strategy involved comprehensive data selection which spanned across different class sizes, and allowed for a thorough analysis across various scales on the Adapt-DCT approach which was integrated into the M-Skipped architecture. This strategy ensures a detailed evaluation of the algorithm's performance under different conditions. In the final section, the attention mechanism which was implemented on top of previous algorithms was examined using a set of five FGVC datasets. The emphasis here was on capturing fundamental aspects of how attention mechanisms influence the performance, particularly in the context of smaller-scale compressed domain FGVC. In fact, this selection aligns with the research objective of forming a foundational understanding of the relationship between HyMod-ECA and the FGVC within small-scale datasets. The datasets chosen for each section of the research are systematically presented in Table 3.9, providing clear analytics of the dataset selection strategy employed throughout this thesis.

Table 3.9: Datasets used in each algorithm.

Datasets	M-Skipped DCT-CNN	Adapt-DCT CNN	HyMod-ECA DCT-CNN
Covid-19		✓	
Sheep Breed	✓		
Flowers	✓	✓	✓
Leeds Butterfly	✓		
Monkey	✓	✓	✓
Spider Breed	✓	✓	✓
Snake Breed		✓	✓
Butterfly		✓	✓
Oxford Flowers		✓	
CIFAR10		✓	
CIFAR100		✓	
Mini ImageNet		✓	

To produce compressed domain input, a conversion from RGB to DCTC is needed. Following the standard JPEG compression CODEC, the images (except CIFAR10, CIFAR100, and Covid19-3C) were initially resized into 224×224 pixels and converted from the RGB domain to the YCbCr domain. The CIFAR10 and CIFAR100 consisted of a 3-channel RGB matrix with a spatial dimension of 32×32, whereas each of the Covid19-3C images contained a single grayscale channel carrying a spatial dimension of 224×224. Chroma subsampling and quantization were not applied to reduce information loss in the images. 2×2 DCT partition was applied on the CIFAR10 and CIFAR100, while the 8×8 partition was applied on other datasets followed by forward 2D-DCT-II. The tensor went through zigzag encoding (conversion of $p \times p$ partitions into a $1 \times p^2$ depth-wise DCTC representation, where $p = \{2, 8\}$ in this case) and feature-wise standardisation. Table 3.10 provides details on the resolution of the original RGB images, the applied DCT partition, and the corresponding shape of the DCTC for each dataset.

Table 3.10: Input shape and DCT partition of various datasets.

Datasets	**RGB Shape**	**DCT Partition ($p \times p$)**	**DCTC Shape**
Covid-19	$224 \times 224 \times 1$	8×8	$28 \times 28 \times 64 \times 1$
Sheep Breed	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Flowers	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Leeds Butterfly	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Monkey	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Spider Breed	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Snake Breed	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Butterfly	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
Oxford Flowers	$224 \times 224 \times 3$	8×8	$28 \times 28 \times 64 \times 3$
CIFAR10	$32 \times 32 \times 3$	2×2	$16 \times 16 \times 4 \times 3$
CIFAR100	$32 \times 32 \times 3$	2×2	$16 \times 16 \times 4 \times 3$
Mini ImageNet	$224 \times 224 \times 3$	2×2	$28 \times 28 \times 64 \times 3$

The result from the forward 2D-DCT operation will generate a tensor carrying integers ranging between -1024 to +1023. Feature scaling was performed by applying normalization to scale the integers into the range of approximately -1 to $+1$. The remaining process of quantization and Huffman Encoding from the full JPEG compression algorithm were excluded as only lossless DCT coefficients were required for this research. This process is considered as partial JPEG compression. Conversely, the complementary process of applying partial decompression by decoding the JPEG image to acquire the DCTC during inference is straightforward. Finally, the dataset was saved as a NumPy array with a standard PyTorch directory format. The block diagram for the partial compression and decompression is shown in Figure 3.24.

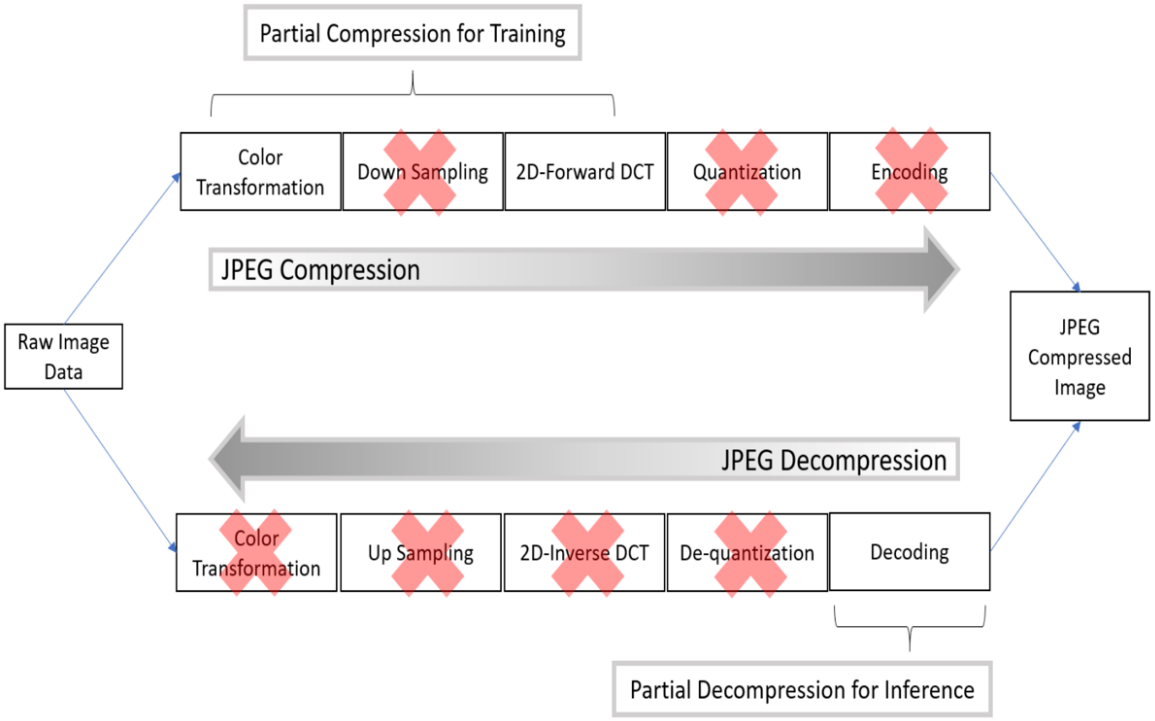


Figure 3.24: Block diagram of the partial compression and partial decompression from JPEG CODEC. The crossed-out sign indicates the processes that are discarded from the standard JPEG CODEC from the partial compression and decompression.

Within the comprehensive collection of FGVC datasets, the total number of images per class ranges between approximately 10 to 1000. In each experimental setup, the dataset is split carefully into training, validation, and test sets. It is aimed to ensure a robust evaluation of the developed algorithm in compressed domain FGVC. For datasets with predefined splits, the established partitions were followed accordingly. In cases where no predefined split exists, a common practice is applied: 10% of the total images per class were designated for testing, another 10% for validation, and the remaining images are allocated to the training set. This partitioning strategy satisfies a balance between retaining sufficient training data for model learning and a comprehensive testing set for diligent evaluation. The specific details of dataset splitting are tabulated in Table 3.11.

Table 3.11: Number of images per class on training, validation, and testing set for various datasets.

Datasets	Training images per class	Validation images per class	Testing images per class
Covid-19	50~85	20~26	20~26
Sheep Breed	336	42	42
Flowers	500~900	26	26
Leeds Butterfly	40	20	20
Monkey	80~00	26~30	26~30
Spider Breed	100~200	10	10
Snake Breed	224	48	48
Butterfly	50~120	10	10
Oxford Flowers	18~800	5~230	5~230
CIFAR10	5000	1000	1000
CIFAR100	500	100	100
Mini ImageNet	570	30	30

Given the challenge of a lower number of training images per class in some of the datasets, the entire training process incorporates various techniques such as early stopping, the utilization of a cosine anneal schedule for learning rate adjustment, dropout, and batch normalization. These methods serve to regularize the model, avoid overfitting, and shorten the training process. Given the specific characteristics of the fine-grained images, it is crucial to avoid relying on data augmentation techniques and to explore alternative methods to enhance dataset diversity. Throughout the training phase, the validation metrics are computed to guide the tuning of hyperparameters. The testing metrics are only employed after the completion of the training process. The testing images are not used to tune the hyperparameter. This separation satisfies the objectivity of model evaluation.

3.5.2 Performance Metrics and Evaluation Criteria

The core motivation of this research is to address the issue of information redundancy in FGVC through compressed domain analytics. To address the fundamental issue of this work, a set of well-designed quantitative metrics is employed to evaluate the performance of the developed algorithm to achieve the objectives. These metrics encompass classification error, the number of trainable parameters, and convergence speed. The classification error serves as a fundamental metric, offering an initial assessment of the algorithm's performance in compressed domain FGVC [98] [100][101][26][104][105]. It is essential to provide a thorough analysis of how well the developed algorithm addresses the identified issue earlier. Given the emphasis on the compressed domain, the number of trainable parameters plays a critical role in evaluating the comprehensiveness between content and model optimisation. A higher compression gain reflects the algorithm's ability to leverage minimal fine-grained features within a smaller model for effective FGVC. To evaluate the effective learning of the proposed algorithm on compressed domain FGVC, the convergence speed is included as an examination metric. By adopting appropriate features in the compressed domain algorithm, the convergence speed can be enhanced, leading to a shorter learning duration. It also poses the effective development of compressed domain FGVC.

While generally, the classification error is an important metric for evaluating model performance, this research places secondary interest on error reduction. In fact, the classification error is utilised as a baseline reference to assess the model's performance against the issue addressed. Furthermore, the scope of this work revolves around the compressed domain. Henceforth, the achievement is evaluated based on the delicate balance between comparable classification error and compression gain.

The goal of conducting compressed domain analytics is to enhance compression gain, which directly relates to the number of trainable parameters. While training speed is influenced by these parameters, it can exhibit vast differences due to factors such as the number of training images, hardware configurations, and operating devices. This variability in behavior extends to inference time as well. Remarkably, the training speed can be comparable across models with different numbers of trainable parameters. The number of trainable parameters is directly correlated with computational complexity,

serving as an indicator of the compression gain in the computation of compressed domain CNNs. Additionally, the training speed is linked to convergence speed, with the specific evaluation metric covered in the result and discussion. Given the central focus in this research is on compressed domain analytics, compression gain stands out as the key metric for addressing this concern. Through the measurement of the number of trainable parameters in experiments, the reduction of these parameters serves as an interpretable metric for interpreting compression gain to address the problem statement and objectives. Hence, the measurement of training speed and inference timing metrics serves as a secondary consideration in evaluating the performance.

In the subsequent section, each quantitative metric for the experiment is defined, accompanied by their respective supplementary metrics.

1) Classification error (ε)

The classification error is computed by dividing the sum of false positives and false negatives over the total number of testing images. In the context of this research, testing error is considered when there is a misclassification within the selected test sets from each class. A lower classification error indicates a better performance. The formula is:

$$\varepsilon = \frac{\boldsymbol{False\ Positives + False\ Negatives}}{\boldsymbol{False\ Positives + False\ Negatives + True\ Positives + True\ Negatives}}$$

Supplementary metrics:

- Best Testing Accuracy: The highest testing accuracy achieved during the inference process, measured in percentage (%). Higher values indicate better performance, computed as:

$$(\mathbf{1} - \boldsymbol{classification\ error}) \times \mathbf{100}$$

- Error Trend: It is determined by tracking the average changes of the testing error across different model variants by using a gradient. This metric yields a trendline that portrays sequential changes in classification error throughout the progressive experimental design. The gradient is computed as a linear approximation function, expressed in percentage (%). The outcomes of this analysis are commonly presented in the concluding rows of each table wherever

applicable. A negative gradient in the error trend resembles a decline in testing error, reflecting an improvement in model performance. Conversely, a positive gradient indicates the opposite scenario. A gradient value smaller than 0.10% is deemed as a constant gradient.

- Error Change: Performance differences in terms of classification error between the developed framework against the reference counterpart. The difference is expressed as a percentage and typically showcased in the final rows of the tables. A negative sign accompanied by a downward arrow (↓) signifies a reduction in classification error where a performance improvement is obtained, while a positive sign with an upward arrow (↑) denotes the opposite scenario. The error change can be obtained as follows:

$$(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon_{Counterpart}}) \times \mathbf{100}$$

- Precision: Measurement of accuracy of positive predictions made by the model.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives} + \boldsymbol{False\ Positives}}$$

- Sensitivity: It is so known as *recall*, measuring model's capability to correctly identify positive instances of each category.

$$\boldsymbol{Sensitivity} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{False\ Negatives} + \boldsymbol{True\ Positives}}$$

- Specificity: It gauges the model's ability to predict false instances of each class.

$$\boldsymbol{Specificity} = \frac{\boldsymbol{True\ Negatives}}{\boldsymbol{True\ Negatives} + \boldsymbol{False\ Positives}}$$

- F1 Score: Harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives. It is important to capture the overall performance of a model when there is an imbalance between precision and recall. The F1 score for each class is calculated and averaged.

$$\boldsymbol{F1\ Score} = \mathbf{2} \times \frac{\boldsymbol{Precision} \cdot \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}}$$

Both sensitivity and specificity serve as essential metrics for assessing the FGVC performance in terms of intraclass scores. A higher model's sensitivity indicates a better capacity to capture positive instances, whereas a higher specificity denotes enhanced proficiency in rejecting negative instances. To ensure a comprehensive evaluation, the F1 score is computed, providing a fair measurement that compares the performance of compressed domain FGVC against the standard algorithm.

2) Number of trainable parameters

The number of trainable parameters denotes the overall count of parameters that were tuned and optimized throughout the model training process. These parameters include the weights and biases inherent in both convolutional layers and fully connected layers. It is measured in millions (mil.). A lower number of parameters leads to higher compression gain, resulting in lightweight architecture that is easier to optimize during training. This simplicity also contributes to higher robustness, reducing the risk of memorizing irrelevant details which could potentially lead to overfitting.

Supplementary metrics:

- Compression Ratio: This ratio defines the relationship between the total trainable parameters of the developed model variants in comparison to the counterpart. A ratio above 1 indicates a reduction in parameters and vice versa.

$$\frac{Reference\ Model\ TP}{Compressed\ Domain\ Model\ TP}$$

- Parameter change: Parameters difference in terms of total number of trainable parameters between the developed framework against the reference counterpart. The difference of change is calculated based on the following equation:

$$\frac{Original\ TP - Optimized\ TP}{Original\ TP} \times 100$$

3) Convergence speed

The convergence speed serves as an indicator of the epoch during the training process at which the model attains over 75% validation accuracy. It implies the training speed and convergence rate of the model, with lower values being preferable.

Supplementary metrics:

- Convergence speed ratio: The ratio between the number of epochs required for the developed system to reach 75% validation accuracy and its counterpart, offering insights into the relative efficiency of the convergence speed.

The first research objective aims to incorporate different levels of frequency bands within DCTC features. The achievement of this objective is evaluated based on the number of trainable parameters and classification errors. In essence, this is intended to measure the efficacy of various frequency ranges in terms of feature representation for FGVC and identify the architecture that yields the optimal performance for these features. The goal is to establish a foundational framework between model and frequency domain feature representations, emphasising the broader exploration of the general emergence and impact of compressed domain features on specific architectures. By extending the frequency features beyond L-DCTC, a reduction in the number of trainable parameters and classification errors as compared with the standard algorithm demonstrate the contributions proposed to achieve this objective. The intraclass score primarily targets the consistency of classification results and the representation capability between subclasses. While the intraclass score carries a secondary role compared to the primary focus of this research on the compressed domain, it is evaluated based on the overall F1 score. The F1 score is computed and compared between the key architecture from the M-Skipped network variants and compared against the standard VGG-16 algorithm. It aims to achieve a complementary understanding of the FGVC analysis between different systems.

The second research objective centres on convolutional kernel analytics. The analytics involve the composition of kernels using DCT-BF hence promoting robustness. As such, the number of trainable parameters and convergence speed are used to determine the achievement of this section. In this context, the intraclass variation carries limited relevance for evaluating the compressed domain technique concerning the delicate balance between optimisation and performance. Metrics such as compression gain, and convergence speed prove to be more prominent for such evaluation. Compressed domain analytics play a role in reducing feature redundancy. Ideally, the proposed algorithm can offer a fair balance of compression gain and convergence speed.

The third research objective emphasises the interactions among frequency features to enhance effective learning. This is evaluated based on the convergence speed and classification error. Ultimately, it is expected that a better algorithm can improve convergence speed with comparable classification performance. The integration of the adaptive kernel technique into the M-Skipped DCT-CNN, combined with HyMod-ECA, constitutes the fully developed framework for compressed domain FGVC. A comprehensive comparison between the developed algorithm and the standard algorithm is conducted to evaluate their performance on practical FGVC. This assessment involves the computation of intraclass metrics such as precision, sensitivity, specificity, and f1-score. Ideally, the framework exhibiting superior scores is expected to demonstrate better performance on FGVC. While these performance metrics provide valuable insights, they hold secondary significance and serve as a subset of metrics emphasised in this research on compressed domain analytics. Therefore, it is imperative to prioritize core metrics such as the number of trainable parameters and convergence speed for a thorough evaluation of the system in this research.

3.5.3 System Setups

All the experiments were implemented on a system running on an Intel Xeon CPU, with 6 cores and a frequency of 2.2GHz to 4.1GHz. The system was equipped with a RAM of 16GB and a Solid-State Drive of 500 GB. The system was running on top of Ubuntu 16.04LTS., and the computing language used for this research was based on Python 3.6. PyTorch (version 1.8) was used to implement the CNN models and evaluate their performance. All the CNN models in each experiment were set to train for a maximum of 100 epochs accompanied by stochastic gradient descent (SGD) with an initial learning rate of 0.01. A cosine anneal scheduler was used during the training with the minimum learning rate set to 0.0001. Early stopping was imposed if the validation performance did not increase for more than 10 epochs to avoid overfitting.

Due to the stochastic nature and variability during training a CNN which arises from the initialisation of random weights, can lead to slight variations in results across different runs. Through employing multiple trials serves to address this inherent variability, thus fostering a robust comprehension and ensuring reliable results. Through the experimentation presented in this thesis, it has been demonstrated that conducting the same experiment three times produces consistent outcomes, with the majority of standard deviations in evaluation metrics falling within an acceptable range. This approach yields reliable results while maintaining experimental efficiency. Therefore, this observation justifies the choice of three trials for all experiments to pursue an early understanding of the performance trend pertaining to small-scale scale-compressed domain FGVC. Additionally, due to the factors such as computational cost and computational constraints, three trials satisfy the research objectives and attains a balance between obtaining dependable results and experimental feasibility. The best-performing models in terms of the lowest testing error were logged. The mean and standard deviation of the performance across three trials were computed for all the following experiments.

3.5.4 M-Skipped DCT-CNN

This section explains the experimental design adopted on M-Skipped DCT-CNN. It consisted of two core experiments followed by an ablation study. The first experiment evaluates the network's performance by feeding different frequency ranges of DCTC as input to the baseline VGG-16 pointwise CNN. Specifically, L-DCTCs, M-DCTCs, H-DCTCs, and complete DCTCs were tested individually. They were denoted as L-DCTC, M-DCTC, H-DCTC, and All-DCTC respectively. This experiment reveals the relative importance of each DCTC frequency range for compressed domain FGVC. One of the objectives was to determine if M-DCTC or H-DCTC alone could provide enough mid-high frequency domain information for feature extraction.

The second experiment implemented a single skipping connection from the input toward the end of the convolutional blocks individually. The connection involves a shallow convolutional layer receiving the M-DCTC in addition to the baseline VGG-16 network, denoted as the 'M-Skipped' branch. The motivation for this is to enable the integration of higher-level feature maps in L-DCTC in conjunction with shallow-level M-DCTC representations. By concatenating the convolved M-DCTC with the feature maps produced from L-DCTC, it was hypothesized that a more robust integrated feature representation of fine-grained compressed information could be generated.

In this experiment, three variants of single M-Skipped branches were employed. The M-Skipped branch was individually concatenated towards the output from the first, second, and last convolutional block from VGG-16. They were denoted as M-Skipped-1, M-Skipped-2, and M-Skipped-3 consecutively. The overview of each variant of the M-Skipped branch is shown in Figure 3.7 in section 3.2.3. Max-Adaptive Pooling was used towards the end of the M-Skipped branch instead of Average-Adaptive Pooling as the former is found to have better performance in the context of FGVC.

The convolutional layer for all the M-Skipped variants carries a single pointwise convolutional layer containing 128 filters. A different number of filters were tested, and it was found that 128 resulted in optimal performance. In each of the M-Skipped variants from Table 3.12, only one of the three 'M-Skipped-x' and 'Concate-x output' was used, while the other convolutional blocks remained the same without any concatenation applied.

Table 3.12: M-Skipped variations model implementation.

Layer	**[f, k, s, p]**	**M-Skipped**	**[f, k, s, p]**
Conv1_1, 2, 3	[256, 1, 1, 0]	M-Skipped-1	[128, 1, 1, 0]
2D Max-pooling	[-, 2, 2, 0]	2D AdapMaxPool	
Concate-1 output	$14 \times 14 \times (256 + 128)^{d}$		
Conv2_1, 2, 3	[512, 1, 1, 0]	M-Skipped-2	[128, 1, 1, 0]
2D Max-pooling	[-, 2, 2, 0]	2D AdapMaxPool	
Concat-2 output	$7 \times 7 \times (512 + 128)^{d}$		
Conv3_1, 2, 3	[512, 1, 1, 0]	M-Skipped-3	[128, 1, 1, 0]
2D Max-pooling	[-, 2, 2, 0]	2D AdapMaxPool	
Concat-3 output	$3 \times 3 \times (512 + 128)^{d}$		
Classifier	Softmax output (Number of classes)		

d. In this case, the M-skipped filter consists of 128 channels.

These experiments were followed up by connecting multiple M-Skipped branches at all convolutional blocks from the main network. The illustration is shown in Figure 3.25. It is intended to compare the performance between single and multiple deeply convolved M-Skipped variants. The single M-Skipped connection was extended by implementing two additional variants as described below:

- Concatenating the output of the M-Skipped branch which carries a single convolutional layer of 128 filters with all three convolutional blocks of the main branch, denoted as 'M-Skipped-123-extended'.

- Increasing the number of convolutional layers in the M-Skipped convolutional branch from one to three, while retaining the same number of filters in each convolutional layer, denoted as 'M-Skipped-123-extended-deep'.

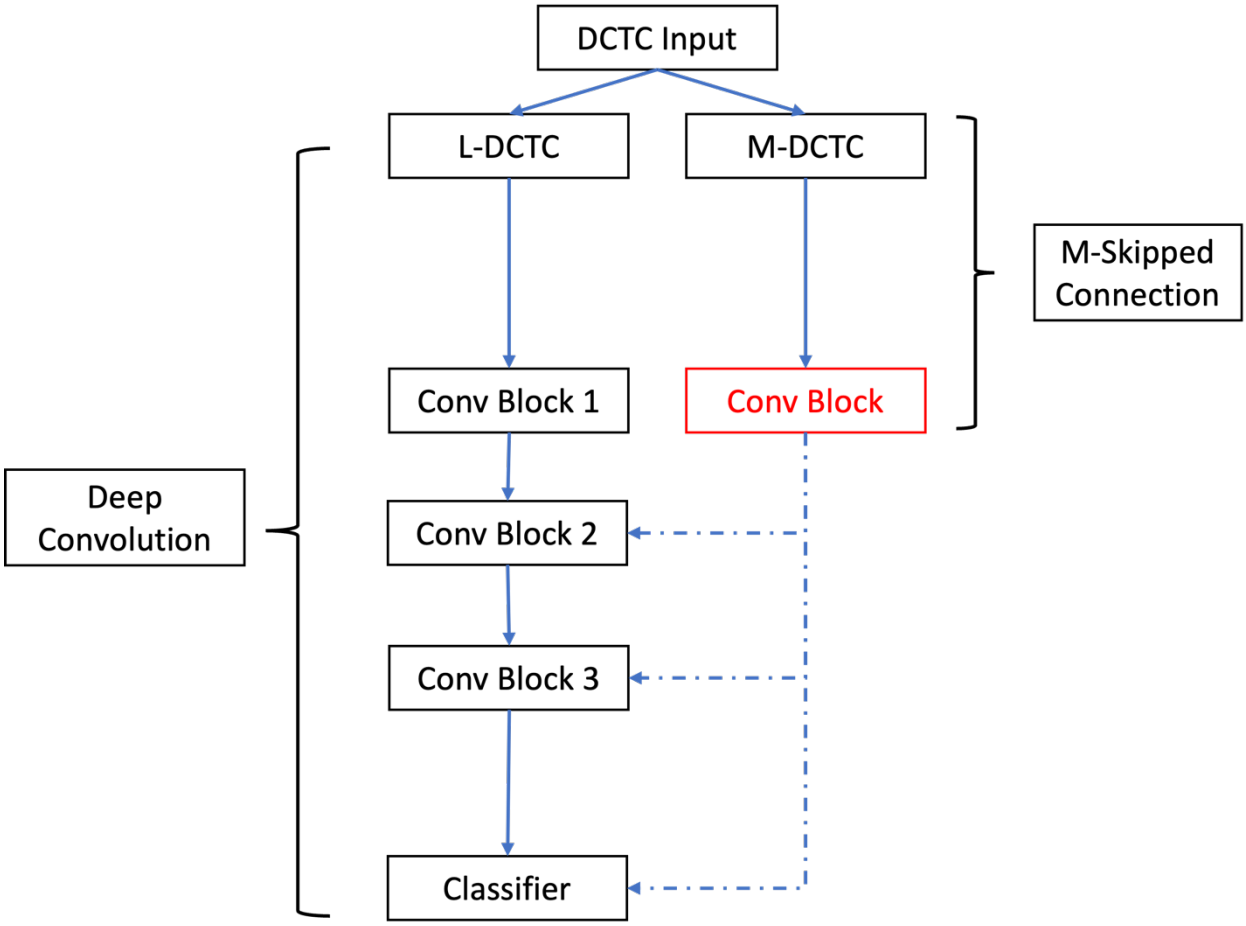


Figure 3.25: M-Skipped DCTC with extended convolutional branch.

In the ablation study, a series of experiments were designed to investigate the significance of integrating diverse frequency components, aiming to establish a comprehensive performance comparison with the former M-Skipped-3 DCT-CNN. The initial experiment introduced fully connected layers into the M-Skipped-3 variant, denoted as 'M-Skipped-3-FC'. The intention is to evaluate the impact of the existence of fully connected layers in compressed domain FGVC. The study also included a comparison involving ReLU as the activation function in the M-Skipped-3 without fully connected layers, denoted as 'M-Skipped-3-FC-ReLU'. Both were then compared against the former M-Skipped-3 variant in terms of classification error. Subsequently, the second experiment replaced the input of the skipping connection in M-Skipped-3 from experiment 1 with H-DCTCs, while maintaining the other setups unchanged, denoted as 'H-Skipped-3'. This adjustment seeks to explore the distinctions arising between combining H-DCTCs as opposed to M-DCTCs with the high-level features derived from L-DCTCs in the primary network. The final experiment incorporated the

original RGB image of all datasets as input to a standard VGG-16 algorithm. This configuration was established to create a baseline comparison between the standard algorithm and compressed domain FGVC. The F1-Score is computed and compared between the H-Skipped-3, the standard VGG-16, and the former M-Skipped-3 variant for general FGVC intra-class analysis. These carefully designed experiments collectively contribute to a deeper understanding of the impact of varying frequency components on the performance of compressed domain FGVC models.

3.5.5 Adaptive-DCT Pointwise Convolutional Neural Network

The performance of Adapt-DCT CNN was assessed in several image classification tasks including FGVC and general datasets. To the best of the author's knowledge, there are currently no available benchmark results on the fine-grained datasets considered in this experiment, which consisted of DCT input. The entire convolution process that is considered in this thesis consists of DCT input, DCT feature maps, and DCT kernel. Therefore, it is not possible to directly compare these results with other benchmarking literature of similar design.

The baseline experiments and models were designed carefully to serve as vital reference points for comparative analysis between Adapt-DCT CNN variants. In all experiments, the Adapt-DCT kernel was implemented on top of the M-Skipped DCT-CNN, with the input image comprising the DCTC tensor. The replacement of a convolutional layer in the CNN with an Adapt-DCT convolution kernel is straightforward. It can be treated as a plug-and-play process. It enables all the experiments to be carried out without going through modifying complex convolution blocks and layers.

Several experiments were designed to test out the Adapt-DCT convolution kernel. Two main studies were presented via three experiments to establish the empirical support for the Adapt-DCT concept, including spatial upscaling and frequency optimisation. Three of the main experiments were conducted on six medium-sized FGVC datasets (Covid19-3C grayscale, Flowers-5C, Monkey-10C, Spider-15C, Snake Breed 35C, Butterfly-50C), whereas three additional datasets were included in the ablation study (CUB-144C, CIFAR10 and CIFAR100).

3.5.5.1 Spatial upscaling of Adaptive DCT Kernel

In the first experiment, the pointwise convolutional kernel in the M-Skipped-3 DCT-CNN was replaced with the Adapt-DCT kernel. Three different spatial variants of the Adapt-DCT kernel were explored, including 2×2, 4×4, and 8×8. The corresponding DCT basis functions of frequency bases remained with the spatial dimension, i.e., no pruning or optimisation was applied.

Let the initial Adapt-DCT kernel ($\mathcal{K}_{TS}$) carries the shape of $\Re^{N \times N \times C}$, where C is the number of channels and $N \times N$ represents the spatial dimension. Different spatial dimensions will produce different numbers of frequency bases (N^2) and numbers of channel weights set per frequency base ($\frac{C}{N^2}$). The relationships between the spatial dimensions of Adapt-DCT kernel (N), numbers of DCT frequency bases (N^2) and the numbers of channel weights ($\frac{C}{N^2}$) are depicted in Table 3.13:

Table 3.13: Functionality and specifications of spatial dimensions of Adapt-DCT kernel.

Spatial dimension of Adapt-DCT kernel	2×2	4×4	8×8
Number of DCT frequency base	4	16	64
Channel weights set per DCT frequency base	$\frac{C}{4}$	$\frac{C}{16}$	$\frac{C}{64}$

The objective of implementing different spatial dimensions of Adapt-DCT kernel is motivated in several ways. The experiment is expected to lead to an early understanding on the potential of capturing certain frequency domain features along the CNN. It is also helpful to realize the optimal spatial dimension of the Adapt-DCT kernel for different contexts and datasets. Three variations of the spatial dimension of Adapt-DCT kernel and the corresponding abbreviations are listed in Table 3.14. (K, L, F) denotes the (spatial dimension of the kernel, number of DCT basis function of frequency bases, number of channels C). The performance comparison of these variants on different datasets will be measured using classification error.

Table 3.14: Variations and specifications of Adapt-DCT kernel with spatial upscaling.

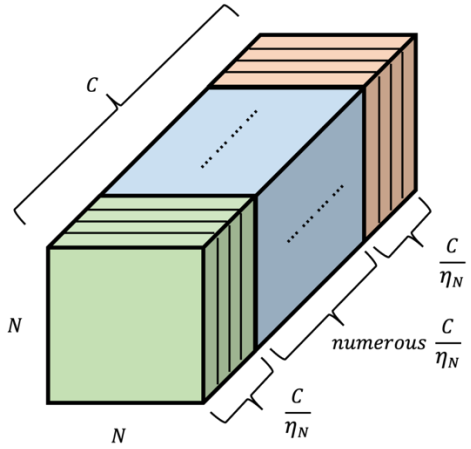
Abbreviation	Convolution block 1 (K, L, F)	Convolution block 2 (K, L, F)	Convolution block 3 (K, L, F)
MS3-AD-0204	$(2 \times 2, 4, 256)$	$(2 \times 2, 4, 512)$	$(2 \times 2, 4, 512)$
MS3-AD-0416	$(4 \times 4, 16, 256)$	$(4 \times 4, 16, 512)$	$(4 \times 4, 16, 512)$
MS3-AD-0864	$(8 \times 8, 64, 256)$	$(8 \times 8, 64, 512)$	$(8 \times 8, 64, 512)$

3.5.5.2 Optimisation of DCT Basis Functions of Frequency Bases in Adaptive DCT Kernel

The second experiment studied the effects of reducing the frequency bases of DCT basis functions forming the Adapt-DCT kernel. With a specific number of channels at each convolutional layer (C), pruning away the less important frequency bases will lead to a growing number in channel weight associated to each of the remaining frequency bases (increase in $\frac{C}{N^2}$ whereby N^2 is replaced with η). It is suggested that not all of the frequency bases are contributing towards effective learning and feature representations. Therefore, the optimisation was applied to the frequency bases whereas the spatial bases were retained.

The depth-wise dimension of the original N^2 layer was replaced with η during optimisation. The formulation and criteria of η is established in section 3.3.6. Two spatial dimensions of Adapt-DCT kernel were explored in this experiment, particularly 8×8 and 4×4. With an Adapt-DCT kernel carrying a spatial dimension of 4×4, by optimising the frequency bases, a channel of $\eta = 1$ (leftover DC base) and $\eta = 4$ can be formed. A comparison of the performance between depth-wise levels of 1, 4 and 16 (original) will be presented in Chapter 4. An Adapt-DCT kernel with a spatial dimension of 8×8 can accommodate several depth-wise level optimisations including $\eta = 1$, $\eta = 4$ and $\eta = 16$. A similar comparison will be made between the depth-wise levels of 1, 4, 16 and 64 (original).

A simple illustration is shown in Figure 3.26 on the depth-wise level optimisation based on the Adapt-DCT kernels of 8×8 and 4×4. C represents the number of channels across the convolution layer, N denotes the spatial dimension of the Adapt-DCT kernel and η represents the number of optimised frequency bases.



$$N = \{4, 8\}$$

$$\eta_8 = \{1, 4, 16, 64\}$$

$$\eta_4 = \{1, 4, 16\}$$

Figure 3.26: Depth-wise optimisation on DCT basis functions of frequency bases in Adapt-DCT kernel.

Five variations of depth-wise channel optimisation of Adapt-DCT kernel and their abbreviations are listed in Table 3.15. The evaluation metric is done based on the convergence speed and classification error. It is important to compare these metrics concerning the frequency pruning of the Adapt-DCT kernel.

Table 3.15: Variations and specifications of frequency adaptive DCT-BF kernel optimisation

Abbreviation	Convolution block 1 (K, L, F)	Convolution block 2 (K, L, F)	Convolution block 3 (K, L, F)
MS3-AD-0401	$(4 \times 4, 1, 256)$	$(4 \times 4, 1, 512)$	$(4 \times 4, 1, 512)$
MS3-AD-0404	$(4 \times 4, 4, 256)$	$(4 \times 4, 4, 512)$	$(4 \times 4, 4, 512)$
MS3-AD-0801	$(8 \times 8, 1, 256)$	$(8 \times 8, 1, 512)$	$(8 \times 8, 1, 512)$
MS3-AD-0804	$(8 \times 8, 4, 256)$	$(8 \times 8, 4, 512)$	$(8 \times 8, 4, 512)$
MS3-AD-0816	$(8 \times 8, 16, 256)$	$(8 \times 8, 16, 512)$	$(8 \times 8, 16, 512)$

3.5.5.3 Pruning Effects of Frequency Bases with Fewer Trainable Parameters

In experiment 2, the optimisation of DCT basis functions of frequency bases was explored where the total number of channels (C) was kept constant. It is also vital to study the effects of pruned frequency bases with a reduced number of channels to achieve a higher compression gain. The objective is to study the effects of pruned frequency based on the speed of convergence and performance in a compressed domain CNN. This experiment served as an extension of the prior experiment. Besides pruning away the less essential frequency bases, the channel weights set per frequency base ($\frac{C}{N^2}$) were kept constant. An example for a 4×4 kernel would result in 16 channel weights per frequency base $\left(\frac{256}{64}\right)$ for convolutional block 1 and 32 channel weights per frequency base $\left(\frac{512}{64}\right)$ for convolutional blocks 2 and 3. The formulation on an 8×8 Adapt-DCT kernel is similar following the same logic. By conducting direct pruning on the frequency bases will reduce the total number of trainable parameters.

Five variations in experiment 3 are depicted in Table 3.16, with the corresponding abbreviation and the specifications for each convolutional block. (K, L, F) denotes the '(spatial dimension of Adapt-DCT kernel, number of DCT basis function of frequency bases, number of channels C'). As the focus here is to prune away the frequency bases to achieve higher compression gain, the evaluation is built based on the percentage change in the number of trainable parameters and the gradient trend in classification error.

Table 3.16: Variations and specifications of pruning of frequency adaptive DCT-BF kernel.

Abbreviation	Convolution block 1 (K, L, F)	Convolution block 2 (K, L, F)	Convolution block 3 (K, L, F)	M-Skipped
MS3-AD-0401-opt	$(4 \times 4, 1, 16)$	$(4 \times 4, 1, 32)$	$(4 \times 4, 1, 32)$	(-, 1, 8)
MS3-AD-0404-opt	$(4 \times 4, 4, 64)$	$(4 \times 4, 4, 128)$	$(4 \times 4, 4, 128)$	(-, 1, 32)
MS3-AD-0801-opt	$(8 \times 8, 1, 4)$	$(8 \times 8, 1, 8)$	$(8 \times 8, 1, 8)$	(-, 1, 2)
MS3-AD-0804-opt	$(8 \times 8, 4, 16)$	$(8 \times 8, 4, 32)$	$(8 \times 8, 4, 32)$	(-, 1, 8)
MS3-AD-0816-opt	$(8 \times 8, 16, 64)$	$(8 \times 8, 16, 128)$	$(8 \times 8, 16, 128)$	(-, 1, 32)

3.5.5.4 Ablation Study

In the ablation study, different spatial dimensions of the Adapt-DCT kernel were applied to different convolution blocks. Following the foundational concept where the feature maps of the earlier network contain lower-level patterns, a larger Adapt-DCT kernel was used in the earlier block of the network. In the latter network, more detailed and complex higher-level features were found. In essence, an 8×8 kernel was used in the first convolutional block, 4×4 was used in the second convolutional block and 2×2 was used in the last convolutional block. This variant is denoted as 'MS3-AD-842'. Another two basic CNNs with VGG-16 as the core architecture were presented for baseline comparison. The first model is denoted as 'VGG16-PC', where three convolutional layers were used to form a convolutional block. A total of three convolutional blocks were stacked together to form the baseline VGG-16. All the convolutional layers in this model featured Adapt-DCT pointwise convolutional kernel in line with the frequency domain CNN. The VGG16-PC was implemented without any skipped connection. ReLU was used as the activation function in this model and the fully connected layers at the end of the network were removed. The second baseline CNN was the M-Skipped-3 DCT-CNN. It is denoted as 'MS3-base'. All three of the models that were tested in the ablation study are listed in Table 3.17.

Table 3.17: Variations and specifications of varying spatial size of Adapt-DCT kernel on frequency adaptive DCT-BF kernel optimisation.

Abbreviation	Convolution block 1 (K, L, F)	Convolution block 2 (K, L, F)	Convolution block 3 (K, L, F)
MS3-AD-842	$(8 \times 8, 64, 256)$	$(4 \times 4, 16, 512)$	$(2 \times 2, 4, 512)$
VGG16-PC*	$(1 \times 1, -, 256)$	$(1 \times 1, -, 512)$	$(1 \times 1, -, 512)$
MS3-base*	$(1 \times 1, -, 256)$	$(1 \times 1, -, 512)$	$(1 \times 1, -, 512)$

**Models with asterisk signs are not implementing the Adapt-DCT kernel.*

3.5.6 Experimental Setup of Hybrid Modified ECA

Within this section, the implementation of the HyMod-ECA module takes place on top of the M-Skipped-3 DCT-CNN which carries the Adapt-DCT convolutional kernel. The variant which was abbreviated as 'MS3-AD-0404' was selected as the baseline model in this experiment for its balance between performance and compression gain, as detailed in Table II and III in the Appendix. Four distinct experiments were designed to address specific research questions outlined in Chapter 1, particularly concerning the concept of intra-group DCT channel interaction and the optimisation of HyMod-ECA on Adapt-DCT CNN. It facilitates a thorough evaluation and fair comparison of the impact of the HyMod-ECA module.

In the first experiment (denoted as 'MS3-0404 ECA-ORG'), the ECA module adapted from the original literature without any modification was integrated directly into the M-Skipped-3 Adapt-DCT CNN as a baseline comparison. To obtain cross-DCT channel interactions, 1D fast convolution was used with a kernel $K_{(f,k,s,p)}$, which carries a set of convolutional properties of (f, k, s, p). Following the original ECA implementation, the optimal number of kernel filters f is set to a default of 3, with a padding size p and stride size s of 1, yielding $K_{(3,1,1,1)}$.

The second experiment (denoted as MS3-0404 ECA-AD) demonstrates the proposed HyMod-ECA on top of the M-Skipped-3 Adapt-DCT CNN, with a modified 1D convolutional kernel carrying properties of $(f, 1, f, 0)$, 'f' indicates the channel depth of the kernel. With a minor increment in the number of trainable parameters manifested in the kernel attention weights, it was expected that the implementation of HyMod-ECA would improve the performance of the Adapt-DCT CNN over the original ECA. This is due to its emphasis on the intra-group DCT channel interaction on top of the adaptive learning of fine-grained DCT features (Adapt-DCT kernel).

The third experiment (denoted as MS3-0404 ECA-AD-1C1A) intentionally includes only one convolution layer instead of 3 in every convolutional block in the primary network followed by HyMod-ECA. The motivations come twofold. Firstly, the experiment intends to achieve compression gain with reduced trainable parameters and secondly, to encourage the attention module to focus on DCT channel information in addition to the previous section in Adapt-DCT CNN where spatial context is considered.

A comparable performance was expected to be attained with higher compression gain as compared with its counterpart baseline variations (MS3-0404 ECA-AD).

In the last experiment, the 'MS3-0404 ECA-AD' variant was attached with two supplementary fully connected layers positioned before the classifier near the end. Each of these fully connected layers incorporates a ReLU activation function and comprises 1024 nodes. This augmentation was introduced with the specific aim of capturing non-linear relationships within the DCT channel. The main experiments with their respective abbreviations and details are outlined in Table 3.18

Table 3.18: Experimental specifications and respective model abbreviations of HyMod-ECA.

Abbreviations	**Specifications**
MS3-0404 ECA-ORG	Original ECA with $K_{(3,1,1,1)}$ integrated into the Adapt-DCT CNN.
MS3-0404 ECA-AD	HyMod-ECA with $K_{(f,1,f,0)}$ integrated into the Adapt-DCT CNN.
MS3-0404 ECA-AD-1C1A	HyMod-ECA with $K_{(f,1,f,0)}$ integrated into the Adapt-DCT CNN, with only *one convolution layer per convolution block* instead of three.
MS3-0404 ECA-ADFC	HyMod-ECA was added to the Adapt-DCT CNN, with two fully connected layers towards the end before the classifier.

To ensure the robustness of this research, an empirical investigation was conducted in an ablation study to compare the technique demonstrated by the Hybrid Cosine Basis Convolution (CBC) technique [105] with the developed DCT method presented in this research. Given the primary focus of this research is on compressed domain FGVC, this comparative analysis with the hybrid CBC aligns with the common objective of exploring compressed domain approaches, setting it apart from the more general SOTA fine-grained approaches conducted in the spatial domain. The alignment of objectives is facilitated by hybrid CBC's computation of compressed domain features on a common dataset, i.e. Monkey-10C, as depicted in this research. Furthermore, the VGG-16 baseline model which was employed in hybrid CBC offers a relevant benchmark for comparison with the model variants presented in this study. Consequently, a fair and meaningful comparison can be conducted between the two approaches.

This comparative study is dedicated to presenting the trade-off between compression gain and performance. In contrast to conventional SOTA fine-grained methodologies that predominantly function within the spatial domain, the distinctive emphasis of this study lies in the context of frequency domain analytics. Here, the core focus is on fine-grained feature extraction and representation within the compressed domain, with a deliberate prioritization of achieving compression gain while maintaining comparable performance. Notably, while the hybrid CBC model processes RGB input images in the spatial domain, the various variants of the developed model in this research operate with DCTCs as input images. The comparison between these models primarily depends on key metrics such as classification error rate, parameter compression ratio, and convergence speed ratio.

To address practical FGVC problems, a concluding experiment is formulated to classify Flower (5C) and Leeds Butterfly (8C) utilizing a unified model. Specifically, both datasets were combined and employed, comprising a total of 13 classes. A comparison is established by comparing the performance of a standard VGG-16 algorithm utilizing RGB data from the datasets, against the comprehensive model developed in this research (MS3-0404 ECA-AD). The analysis focuses on main performance metrics, including precision, sensitivity, specificity, and f1-score. These metrics collectively contribute to a thorough understanding of FGVC analysis, particularly concerning intra-class scores between spatial domain and compressed domain. Recognizing the significance of intra-class score in FGVC analysis, the metrics of compression ratio and convergence speed ratio are also presented for comparison, aligning with the central focus of this research.

Chapter 4 Results and Discussion

4.1 M–Skipped DCT–CNN

4.1.1 Low, Medium, and High DCT Coefficients

Following the training setup in section 3.5.4, Table 4.1 summarises the results. The table shows the comparison of classification error comparing test images for L-DCTC, M-DCTC, H-DCTC, and All-DCTC. The input channel for the first convolutional layer is changed accordingly to fit the L-DCTC ($28 \times 28 \times 16 \times 3$), M-DCTC ($28 \times 28 \times 32 \times 3$) and H-DCTC ($28 \times 28 \times 16 \times 3$) representations respectively. The M-skipped connection is not applicable in this experiment.

Table 4.1 depicts the difference in classification error between varying DCTC conditions and the performance changes across FGVC datasets. Generally, L-DCTC and sometimes All-DCTC performed better than standalone M-DCTC and H-DCTC. In brief, all datasets exhibit better performance on All-DCTC or L-DCTC as the sole input compared to M-DCTC or H-DCTC. Generally, with M- or H-DCTC as the only input, the classification error is above 0.4. With All-DCTC as input, most of the datasets have a lower error rate of around 1% to 4% when compared with L-DCTC. In contrast, Flowers and Sheep Breed have no performance difference. The experiment conducted in this section once again confirmed that the conventional approach of utilising All-DCTC or L-DCTC as the only input is advantageous, as All-DCTC can fully represent the partition features in the compressed domain, while L-DCTC contains the majority of the information required for the model to learn well. In other words, All-DCTC and L-DCTC can be used as a general feature representation in FGVC.

As previously mentioned, the inclusion of individual M-DCTC and H-DCTC representations aimed to assess whether the deep CNN could effectively learn discriminative features in FGVC within the mid-high frequency spectrum. However, the observed lower performance, as indicated in Table 4.1, suggests that mid-high DCTCs may not be as important as L-DCTCs. H-DCTC input generally has the lowest accuracy as it tends to consist of irrelevant features, often noise. According to the preliminary results in Table 4.1 and Table 4.6 in the ablation study, it was revealed that H-DCTC could not provide sufficient features for the network, hence leading to its exclusion from

the subsequent experiments. Using M-DCTCs as standalone input only introduces medium-varying features. In essence, M-DCTC on its own will not furnish adequate information, highlighting the necessity of L-DCTCs to enhance overall performance. This finding motivated the subsequent experiments where M-DCTCs were introduced and built around the foundational L-DCTC feature.

Table 4.1: Comparison of classification error on individual DCTCs input.

Abbreviation	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed
All-DCTC	**0.0972**	**0.2615**	**0.1979**	**0.2721**	**0.2244**
L-DCTC	**0.0972**	**0.2615**	0.2021	0.3113	0.2400
M-DCTC	0.4028	0.4513	0.5833	0.6434	0.6867
H-DCTC	0.4643	0.5615	0.6812	0.6752	0.7111

4.1.2 Single and Multiple M-Skipped Connection

Tables 4.2, 4.3, and 4.4 show the test error for each M-Skipped variant. With regards to Table 4.2, Monkey and Spider Breed attained better performance with M-Skipped-1, Sheep Breed and Flowers achieved better performance with M-Skipped-2 while Leeds Butterfly had better performance with M-Skipped-3. By computing the average performance metrics across each M-Skipped variant in Table 4.2, it is clear that M-Skipped-3 obtained the overall lowest classification error, which is 0.1895. It was found that by concatenating the output of a single M-DCTC skipped convolutional branch at the last convolutional block, i.e. M-Skipped-3, it generated the best results. Hence, it is suitable to adopt M-Skipped-3 as the baseline architecture over the other variants.

By referring to Table 4.3, by comparing the 'M-Skipped-123-extended' to the 'M-Skipped-3' variant, a maximum of 6% error rate increase was observed in the extended model on Spider Breed. Its deeper counterpart obtained the lowest error rate on Monkey and Spider Breed over the single M-Skipped variant with a difference error rate of up to 3%. However, the model carries an additional 0.2 million trainable parameters compared to the single M-Skipped variant. From Table 4.4, it can be seen that by adopting an M-Skipped connection, a reduction in error rate of up to 7.5% can be

obtained in medium-sized FGVC datasets, when compared to a more conventional network, i.e. All-DCTC variant. This suggests that M-DCTC contains additional higher frequency details that when integrated with deeply convolved L-DCTC representations, a feature-rich frequency domain representation can be obtained.

From these results, it is clear that the deeper M-Skipped convolutional branch significantly enhances the original M-DCTC feature representation and improves performance on some of the datasets. Nevertheless, the single M-Skipped architecture still offered a clear balance between parameters and performance. This experiment once again shows that M-DCTC is important for FGVC and that the specific approach adopted for integrating M-DCTC is not necessarily trivial. In other words, the specific way in which M-DCTC is integrated can significantly affect performance in the FGVC domain.

Table 4.2: Comparison of classification error between three M-Skipped variants and the corresponding average performance.

Abbreviation	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed	Average
M-Skipped-1	0.0873	0.2589	0.1667	**0.2574**	**0.2156**	0.1972
M-Skipped-2	**0.0695**	**0.2436**	0.1625	0.2709	0.2666	0.2026
M-Skipped-3	0.0853	0.2538	**0.1229**	0.2610	0.2245	**0.1895**
Error Change (Best variant vs M-Skipped-3)	+1.58% (↑)	+1.02% (↑)	0.0000	0.36% (↑)	-0.89% (↓)	

Table 4.3: Comparison of classification error between multiple M-Skipped variants against the M-Skipped-3.

Abbreviation	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed	Average
M-Skipped-3	**0.0853**	**0.2538**	**0.1229**	0.2610	0.2245	0.1895
M-Skipped-123-extended	0.0873	0.3051	0.1792	0.2402	0.2889	0.2201
M-Skipped-123-extended-deep	0.0893	0.2692	0.1479	**0.2230**	**0.1845**	**0.1816**

Table 4.4: Comparison of classification error between the best-performing M-Skipped variation and the baseline variation without the M-Skipped branch (All-DCTC).

Abbreviation	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed
All DCTC	0.0972	0.2615	0.1979	0.2721	0.2244
Best performing M-Skipped	0.0695	0.2436	0.1229	0.2574	0.1845
Error Change	-1.19% (↓)	-1.79% (↓)	**-7.5% (↓)**	-1.47% (↓)	-3.99% (↓)

4.1.3 Ablation Study

In classical CNNs, fully connected layers were typically employed at the end of the model to improve non-linear mapping within feature maps. Surprisingly, it was found that compressed domain CNNs often exhibit similar or sometimes inferior performance when compared to models without fully connected layers, as evidenced in Table 4.5. M-Skipped-3 without fully connected layers still offers the overall best average performance with the lowest error as compared to other variants. Furthermore, the inclusion of fully connected layers leads to an increase in the overall parameters by approximately 7 million, as shown in Table 4.7. Thus, the fully connected layers were not employed in this research to achieve higher compression gain and avoid unnecessary algorithm complexity.

The initial choice of the ReLU activation function caused a clustering of features around the x and y-axis when plotted. It poses a challenge for the classifier to draw clear classification boundaries between classes. Moreover, when datasets were normalized between 0 to +1 and used on ReLU, the data scaling range was considerably narrow when compared to the feature standardization achieved with a range of -1 to +1 on PReLU. Feature map comparisons in Figure 4.1 illustrate the PReLU-based feature maps exhibit more pronounced clustering, allowing a more visible decision boundary. The concept is further supported by the results in Table 4.5. It shows that the use of PReLU in the baseline variant of M-Skipped-3 can achieve the lowest average error of 0.1895 over M-Skipped-3-ReLU.

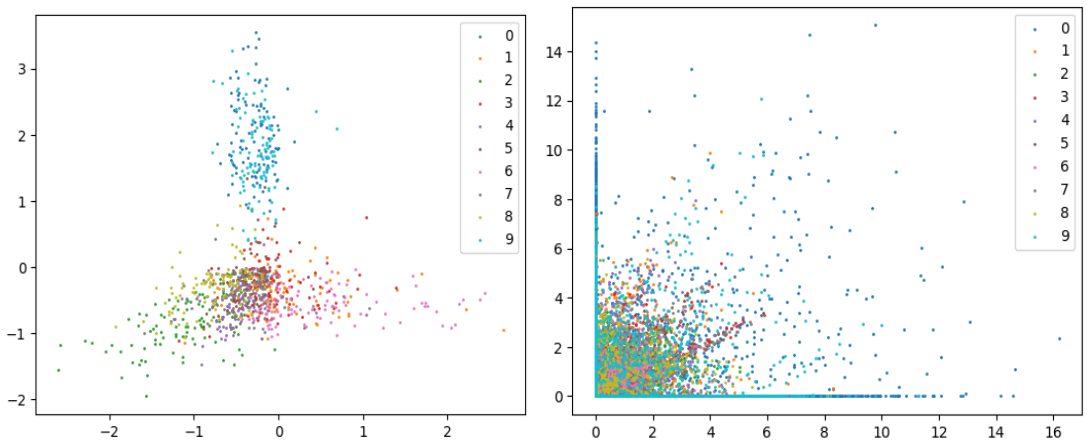


Figure 4.1: Feature map representations of activation function outputs from ReLU (Right) and PReLU (Left).

Table 4.5: Comparison of classification error between baseline M-Skipped-3 variant with PReLU ('M-Skipped-3'), baseline M-Skipped-3 variant with fully connected layers ('M-Skipped-3-FC), and baseline M-Skipped-3 variant with ReLU ('M-Skipped-3-ReLU').

Abbreviation	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed	Average
M-Skipped-3	0.0853	0.2538	0.1229	0.2610	0.2245	**0.1895**
M-Skipped-3-FC	0.0714	0.2769	0.1416	0.2512	0.2978	0.2078
M-Skipped-3-ReLU	0.1071	0.2666	0.1771	0.2904	0.2844	0.2251

The final investigation exploited the intraclass scores, particularly focusing on the comparison between the developed algorithm (M-Skipped-3) and the H-Skipped-3 variant, alongside a standard SOTA VGG-16 algorithm. The F1-Score across these variants was computed and presented in Table 4.6. An in-depth examination of the intraclass scores shows that M-Skipped-3 outperforms H-Skipped-3. It attains a better average F1-Score across all datasets. While H-Skipped-3 delivers higher F1-Score in specific datasets such as Sheep Breed and Spider Breed, with improvements of 0.5% and 3.99% respectively, M-Skipped-3 consistently exhibits better performance across other datasets. Despite both variants carrying the same number of parameters, as outlined in Table 4.7, the average F1-Score demonstrates that M-Skipped-3 excels in providing better FGVC analysis compared to H-Skipped-3.

From the average F1-Score formulated in Table 4.6, the M-Skipped-3 variant that is developed in this section surpasses the performance of the standard VGG-16 algorithm by 2.2%. On an individual dataset basis, except Spider Breed showing better performance with the standard algorithm, all other datasets have better FGVC analysis on M-Skipped-3. Furthermore, the number of parameters exhibited by the standard algorithm is significantly greater than that of M-Skipped-3. According to Table 4.7, M-Skipped-3 contains 1.7 million parameters whereas the standard algorithm contains 134.4 million. The compression ratio of standard VGG-16 against the M-Skipped-3 reaches up to 79 times. By considering the key emphasis of this research on compression gain and compressed domain, coupled with the better overall F1-Score of the developed system, it is promising that small-scale compressed domain FGVC excels over the classical VGG-16 approach in terms of FGVC analysis.

Table 4.6: Comparison of F1-Score between M-Skipped-3, H-Skipped-3, and the standard VGG-16 algorithm.

Abbreviation (Domain)	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed	Average
M-Skipped-3 (DCT)	94.03	**75.58**	**88.60**	**79.57**	79.26	**83.41**
H-Skipped-3 (DCT)	**94.63**	75.16	84.22	75.84	82.65	82.50
Standard VGG-16 (RGB)	82.30	74.47	88.19	77.91	**83.16**	81.21

Table 4.7: Comparison of number of trainable parameters and compression ratio between different variants.

Abbreviation	Number of Parameters (mil)	Compression Ratio
Standard VGG-16	134.4	1
M-Skipped-3	1.7	79
M-Skipped-123-extended	1.9	70
M-Skipped-123-extended-deep	1.9	70
M-Skipped-3-FC	8.9	15
M-Skipped-3-ReLU	1.7	79
H-Skipped-3	1.7	79

4.2 Adaptive DCT CNN

4.2.1 Spatial Properties of Adaptive DCT Kernel

The weighting and upscaling of spatial properties on the Adapt-DCT kernel are discussed in this section. The setup of this experiment is explained in section 3.5.5. The underlying motivation is to examine various spatial resolutions of the DCT-BF to form the convolutional kernel. The term 'upscaling' is used to describe the ability to initialize the kernel with spatial dimensions of 2×2, 4×4 and 8×8 to weigh the DCT-BF and subsequently to form the final 1×1 convolutional kernel.

From Table 4.8, with an increasing spatial dimension on Adapt-DCT kernel, Flowers and Spider Breed showed increasing error trends of 0.22% and 2.22% respectively. On the other hand, COVID-19, Monkey, and Butterfly showed decreasing errors of 7.49% and 0.55% respectively. The overall best result was obtained with a spatial dimension of 8×8 on COVID-19, Monkey, and Butterfly, while the spatial dimension of 2×2 excelled on Flowers and Spider. This suggests that smaller spatial dimension provides fewer spatial harmonics (only key ones) to form the pointwise convolutional kernel from the adaptive weighting of DCT basis functions and vice versa. The smaller dimension focuses on composing fewer key representations. It is suggested that Flowers and Spider only require a few representational capacities of the pointwise convolution filter for DCT feature learning. On the contrary, a larger dimension serves a higher representational capacity within the pointwise convolution filter that can ease the feature learning of COVID-19, Monkey, and Butterfly.

Table 4.8: Comparison of classification error with increasing spatial dimension of Adapt-DCT kernel.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0204	0.2121	**0.2564**	0.2267	**0.1778**	0.3788	0.1707
MS3-AD-0416	0.1364	0.2590	0.2194	0.2155	**0.3675**	0.1727
MS3-AD-0864	**0.0624**	0.2607	**0.2157**	0.2222	0.3772	**0.1587**
Error Trend	-7.49% (↓)	+0.22% (↑)	-0.55% (↓)	+2.22% (↑)	-0.08 % (↓)	-0.60% (↓)

With an increasing spatial dimension on the Adapt-DCT kernel, a prevalent performance improvement can be observed with up to a 7.49% error drop on Covid19. The Snake Breed exhibited a nearly constant error rate across distinct spatial dimensions. It was observed that an increase in spatial dimension did not contribute to an improvement in its recognition performance. The resulting trend for different spatial dimensions of the Adapt-DCT kernel is dataset-dependent. Figures 4.2 and 4.3 display the visualization of part of the Adapt-DCT kernel on COVID-19 and Spider Breed on different Adapt-DCT variants.

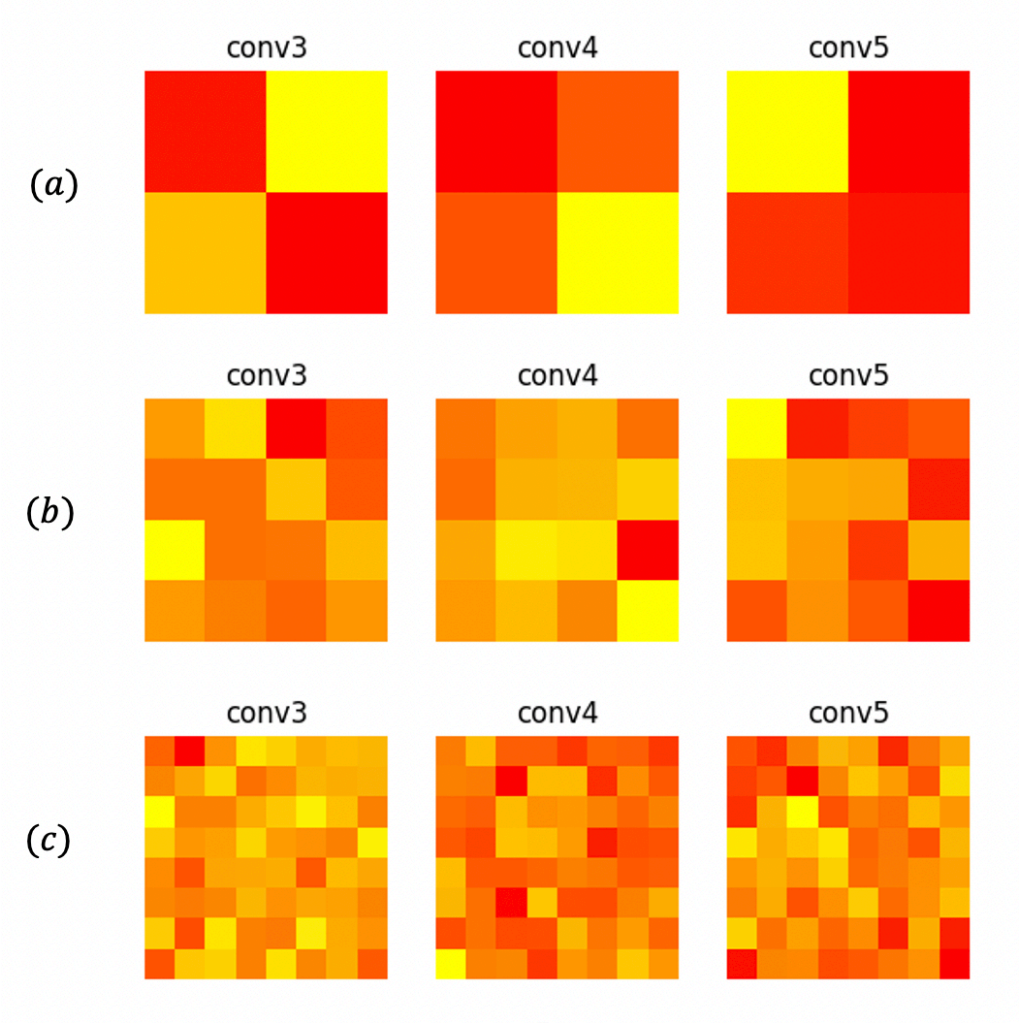


Figure 4.2: The first layer of the convolution kernel is shown for each convolution block, with conv3 referring to convolution block 1, conv4 referring to convolution block 2, and conv5 referring to convolution block 3. Covid19-3C on three different spatial dimensions of Adapt-DCT CNN, (a) MS3-AD-0204; (b) MS3-AD-0416; (c) MS3-AD-0864.

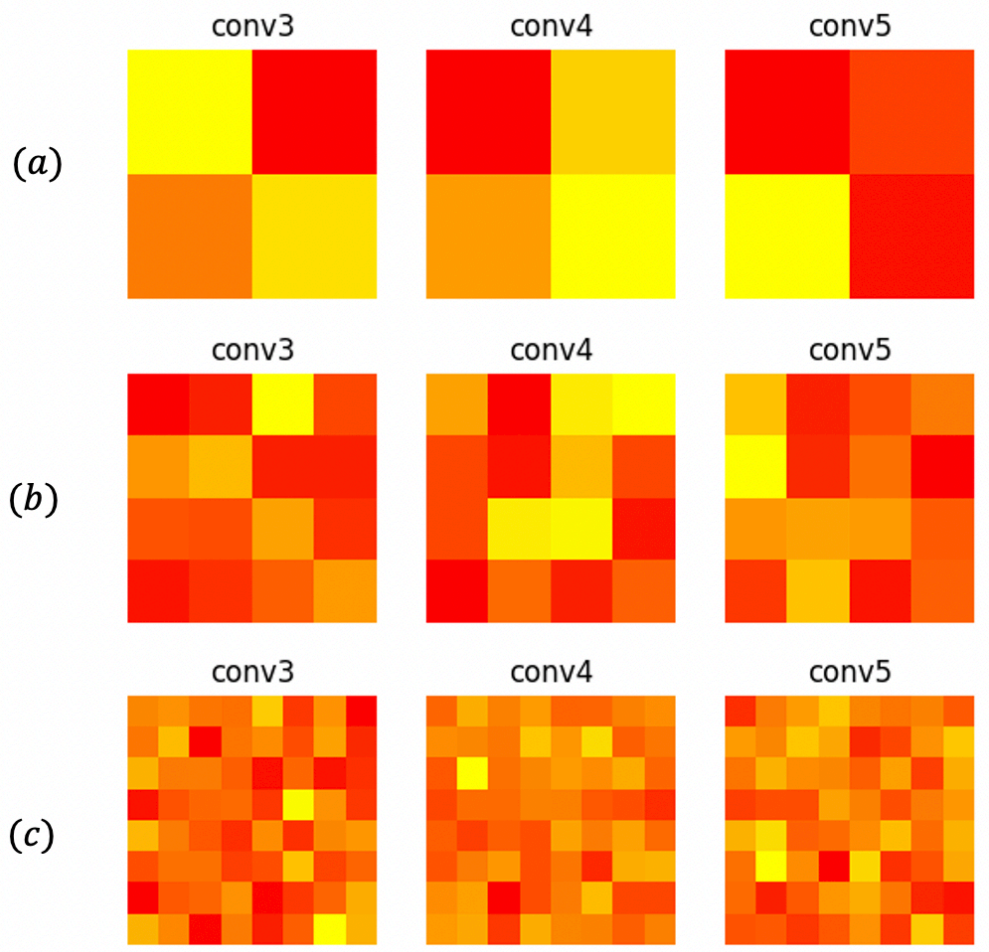


Figure 4.3: The first layer of the convolution kernel is shown for each convolution block, with conv3 referring to convolution block 1, conv4 referring to convolution block 2, and conv5 referring to convolution block 3. Spider-15C on three different spatial dimensions of Adapt-DCT CNN, (a) MS3-AD-0204; (b) MS3-AD-0416; (c) MS3-AD-0864.

The spatial upscaling of the Adapt-DCT kernel contributes to an early understanding of how different spatial dimensions of weighting the DCT basis functions can benefit compressed domain CNNs. A fundamental technique on the capability of filter representation is established from the perspective of the DCT domain. The spatial dimension for optimal performance differs for each dataset. Thus, the spatial properties can be reinstated on top of the DCT pointwise convolution kernel in Adapt-DCT convolution.

4.2.2 Optimisation of DCT Basis Functions of Frequency Bases with Increasing Channel Weights Set

The spatial and frequency bases of DCT basis functions are equally important for the Adapt-DCT kernel in compressed domain image classification. The optimisation of frequency bases came in two forms as discussed in section 3.5.5. The first technique pruned away higher frequency bases while maintaining the total number of channels, which increased the number of channel weights per frequency base. Whereas the second method restrained the number of channel weights per frequency base in addition to the former pruning. Hence, it will reduce the total number of trainable parameters. The prior method intended to achieve frequency-based optimisation while retaining the parameters while the latter method solely focused on pruning and reducing the parameters. The results of the first technique will be discussed in this part. Two variants of the Adapt-DCT kernel with spatial dimensions of 4×4 and 8×8 were presented. The performance was compared between the frequency bases carrying an original number of bases (N^2 where N is the spatial dimension of Adapt-DCT kernel) and the pruned bases (η). More setup information can be found in section 3.5.5.

It is crucial to analyse the convergence speed and the classification error with the frequency pruning of the Adapt-DCT kernel. The classification error and error trend of each frequency base with 4×4 and 8×8 spatial dimension was tabulated in Tables 4.9 and 4.10. For the spatial dimension of 8×8, with a reduction of frequency bases along the channel direction and an increase of 'channel weights set' per frequency base, Spider Breed exhibits a reducing error while Covid19, Monkey, and Butterfly show increasing errors. Flowers and Snake Breed have a near-constant error regardless of the number of frequency bases. An error trend of -0.87% can be found in the Spider with a reduction in the number of frequency bases. Monkey exhibits an increment in error trend of +3.76% under the same pruning condition. This suggests that Spider benefits from the pruning effect in the spatial dimension of 8×8 while performance on Monkey degrades. For the spatial dimension of 4×4, with a reduction of frequency bases and an increase of channel weights set, COVID-19, Monkey, and Butterfly exhibit an error reduction while Monkey, Spider, and Snake Breed show increasing error. COVID-19 benefits from the frequency base pruning with an error trend of -2.27%, while Spider

experienced the highest increment in error trend of +3.78%.

When most of the frequency bases were removed while remaining only one frequency base (4 × 4 is denoted as MS3-AD-0401, 8 × 8 is denoted as MS3-AD-0801), this is referred to as the 'DC frequency base'. Spider shows the best performance on a spatial dimension of 8 × 8 with a DC frequency base while Butterfly achieves the best performance on a spatial dimension of 4 × 4. COVID-19 and Snake Breed did not benefit from the frequency base optimisation in both spatial variants of 4 × 4 and 8 × 8 respectively. This can be recognised when both of the datasets presented the lowest error on the original number of frequency bases. Generally, the optimal performance is obtained when the first 4 frequency bases are used, i.e., when the depth-wise channel contains 4 frequency bases (denoted as MS3-AD-xx04). This concept is proven when Flowers, Snake Breed, and Butterfly present the lowest error on MS3-AD-0804. While COVID-19, Flowers, Monkey, and Spider Breed attain the lowest error on MS3-AD-0404.

Table 4.9: Comparison of classification error trend with spatial Adapt-DCT kernel size of 8.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0864	**0.0624**	0.2607	0.2157	0.2222	0.3772	0.1587
MS3-AD-0816	0.0657	0.2590	**0.2022**	0.2267	0.3736	0.1593
MS3-AD-0804	0.0808	**0.2513**	0.2145	0.2133	**0.3728**	**0.1560**
MS3-AD-0801	0.0791	0.2641	0.3370	**0.1978**	0.3788	0.1760
Error Trend	+0.65% (↑)	+0.02% (↑)	+3.76% (↑)	-0.87% (↓)	+0.04% (↑)	+0.49% (↑)

Table 4.10: Comparison of classification error trend with spatial Adapt-DCT kernel size of 4.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0416	0.1364	0.2590	0.2194	0.2155	**0.3675**	0.1727
MS3-AD-0404	**0.0758**	**0.2408**	**0.2108**	**0.2134**	0.3724	0.1653
MS3-AD-0401	0.0909	0.2564	0.2819	0.2911	0.3750	**0.1607**
Error Trend	-2.27% (↓)	-0.13% (↓)	+3.13% (↑)	+3.78% (↑)	+0.37% (↑)	-0.60% (↓)

The convergence speed for the optimisation of frequency bases is presented in Tables 4.11 and 4.12. Each table indicates the results for different spatial dimensions of 8×8 (Table 4.11) and 4×4 (Table 4.12). The convergence metric of the Snake Breed was excluded from both tables as it did not achieve a classification error below 0.25. The variant with the best convergence speed is shown in bold numeric in the tables. The convergence speed ratio was obtained by comparing the original frequency base with the earliest converge variant. For the spatial dimension of 8×8, all of the datasets attained early convergence with the frequency base optimisation as compared with the original one. Monkey achieved up to 1.28 times faster convergence on the MS3-AD-0804 as compared with the original counterpart. With the frequency base optimisation conducted on the spatial dimension of 4×4, Covid19 exhibits up to 1.33 times faster convergence while Spider Breed shows no improvement. The highest convergence speed can be found in the context of the spatial dimension of 4×4 on the MS3-AD-0401 on Covid-19, Monkey, and Butterfly. For the spatial dimension of 8×8, insufficient trends can be drawn to provide a conclusive statement on which variant will lead to the fastest convergence.

Table 4.11: Epoch reaching 75% test accuracy for spatial dimension of 8.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed
MS3-AD-0864	8.33	21.00	15.00	18.00	6.33
MS3-AD-0816	**7.00**	**20.00**	21.67	17.00	6.00
MS3-AD-0804	8.00	23.00	**11.67**	17.00	**5.33**
MS3-AD-0801	7.67	25.00	14.00	**16.00**	5.67
Convergence Speed Ratio (vs MS3-AD-0864)	1.19	1.05	1.28	1.13	1.19

Table 4.12: Epoch reaching 75% test accuracy for spatial dimension of 4.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed
MS3-AD-0416	8.00	18.67	13.33	**11.00**	6.67
MS3-AD-0404	7.00	**16.00**	13.00	16.33	6.67
MS3-AD-0401	**6.00**	20.33	**12.00**	15.33	**6.33**
Convergence Speed Ratio (vs MS3-AD-0416)	1.33	1.17	1.11	1.00	1.05

The technique of pruning away frequency bases with increasing channel weight sets allows the optimisation of frequency bases and early convergence on most of the datasets. Early convergence is achieved as it is suggested that fewer frequency bases were required to be optimised, hence promoting robustness. The representation of a pointwise convolutional filter with frequency base optimisation allows more filters of the same frequency base to be generated. This suggests that the model can improve feature learning if the DCT features contain a large portion of similar context that matches the filter's pattern of the remaining frequency bases.

4.2.3 Optimisation of DCT Basis Functions for Trainable Parameters Compression

As the objective here is to prune the frequency bases to achieve fewer trainable parameters, the evaluation is built based on the compression ratio (Tables 4.15 and 4.16) and the error change (Tables 4.13 and 4.14). From Tables 4.13 and 4.14, all of the datasets show increasing errors with a reduced number of frequency bases and trainable parameters. With a spatial dimension of 8 × 8 on the Adapt-DCT kernel, the steepest error trend of +16.76% can be found on the Snake Breed while the lowest trend of +2.87% was found on Flower. With a spatial dimension of 4 × 4, the steepest trend of +17.27% was exhibited on Snake Breed while the lowest trend of +1.01% was found on Covid19

By referring to Tables 4.15 and 4.16, the optimisation of frequency based on spatial dimensions of 8 × 8 and 4 × 4 offers a 93% to 99% reduction in the number of trainable parameters concerning their original variant. The lowest classification error across the board can be found on MS3-AD-0816-opt (for spatial dimension of 8 × 8) and MS3-AD-0404-opt (for spatial dimension of 4 × 4), except Covid19, which attained the lowest error on MS3-AD-0401-opt. In other words, in this experimental setup, MS3-AD-0816-opt and MS3-AD-0401-opt offer the best performance for most of the datasets.

Table 4.13: Best performing classification error trend comparison with spatial Adapt-DCT kernel size of 8, with increasing DCT frequency basis functions and trainable parameters.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0816-opt	**0.1010**	**0.2410**	**0.2623**	**0.2289**	**0.4147**	**0.2073**
MS3-AD-0804-opt	0.1768	0.2641	0.3689	0.2955	0.7198	0.2807
MS3-AD-0801-opt	0.2727	0.3487	0.4669	0.3800	0.8343	0.4187
Error Trend	+7.07% (↑)	**+2.87%** (↑)	+8.60%	+5.40% (↑)	**+16.76%** (↑)	+8.53% (↑)

Table 4.14: Best performing classification error trend comparison with spatial Adapt-DCT kernel size of 4, with increasing DCT frequency basis functions and trainable parameters.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0404-opt	0.1717	**0.2615**	**0.2487**	**0.2400**	**0.4426**	**0.2107**
MS3-AD-0401-opt	**0.1566**	0.2872	0.3456	0.3356	0.7129	0.2860
Error Trend	**+1.01%** (↑)	+1.41% (↑)	+6.31% (↑)	+6.00% (↑)	**+17.27%** (↑)	+5.66% (↑)

Table 4.15: Specifications and reduced percentage of trainable parameters of the optimised variants with respect to the original variant of spatial Adapt-DCT kernel of 8 × 8. TP reduced is measured in percentage (%) with reference to MS3-AD-0864-org.

Abbreviation	**Number of Parameters (mil)**	**Parameter Change (%)**
MS3-AD-0864-org	102.53	-
MS3-AD-0816-opt	6.76	93.41
MS3-AD-0804-opt	0.51	99.50
MS3-AD-0801-opt	0.05	99.98

Table 4.16: Specifications and reduced percentage of trainable parameters of the optimised variants with respect to the original spatial Adapt-DCT kernel of 4 × 4. TP reduction is measured in percentage (%) with reference to MS3-AD-0416-org.

Abbreviation	**Number of Parameters (mil)**	**Parameter Change (%)**
MS3-AD-0416-org	25.85	-
MS3-AD-0404-opt	1.74	93.26
MS3-AD-0401-opt	0.14	99.46

Generally, the frequency pruning with very few trainable parameters provides higher compression gain. This approach can offer a hybrid understanding of the performance of each variant before conducting detailed experiments on the model that contains the original number of frequency bases. The tradeoff of this approach is the mediocre performance. It is suggested that the representation capability of pointwise convolutional filter drops concerning the reduced frequency bases with consistent channel weights set per frequency base. This scenario constrains the learning capability of a filter hence leading to a performance drop. In a nutshell, the optimisation of DCT-BF with pruning the frequency bases opens up the window to customise the DCT-BF to form a convolution filter.

In Figure 4.4, the noticeable gap between training and validation metrics entails the presence of a potential overfitting issue. This gap may arise due to factors such as inherent noise, a limited dataset, or the complexity of the model. Several precautions have been implemented carefully to address overfitting through regularisation techniques. These involve applying dropout by randomly omitting neurons during training to prevent the model from relying heavily on specific neurons. Additionally, a cosine annealing scheduler and early stopping have been adopted to control the training progress and foster model robustness. Introducing randomness compels the model to learn diverse features, ultimately promoting more robust and generalised representations. Beyond these measures, the predominant way to further reduce overfitting is by obtaining more data. This is currently infeasible given the context of the datasets and the research objectives.

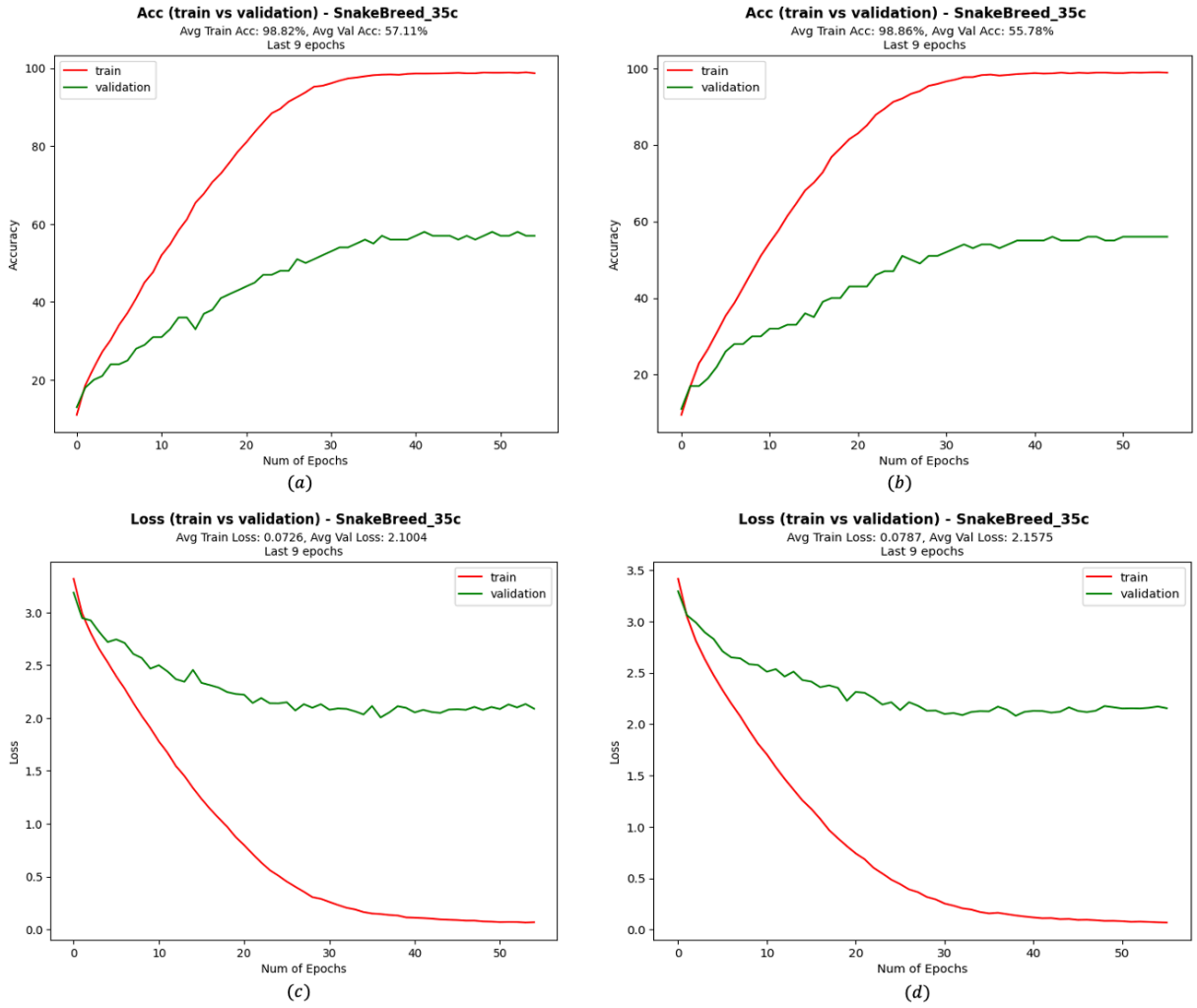


Figure 4.4: Training curves for Spider-35C. (a) Validation accuracy training curve on MS3-AD-0816-opt; (b) Validation accuracy training curve on MS3-AD-0404-opt; (c) Validation loss training curve on MS3-AD-0404-opt; (d) Validation loss training curve on MS3-AD-0816-opt.

4.2.4 Ablation Study

In this section, a simple performance comparison between 2 baseline models and the best variant of Adapt-DCT CNN was presented. Tables 4.17 and 4.18 establish the classification error obtained from the baselines and Adapt-DCT CNN. The results are shown based on 7 FGVC datasets (Covid-19, Flowers, Monkey, Spider Breed, Snake Breed, Butterfly, Flowers) and 3 general datasets (Cifar-10, Cifar-100, Mini Image Net).

From a general glance, all the datasets exhibit improved performance in the Adapt-DCT CNN condition over the baseline models. An improvement with up to 13.97% error drop on the Monkey-10C dataset can be found. The results in Table 4.17 show that FGVC datasets with below 100 classes can achieve up to 14% error drop over the baseline model and the M-Skipped-3. In Table 4.18, with the adoption of the Adapt-DCT kernel, Flowers achieves up to 4% error drop over the baseline models, while the general datasets (CIFAR10 and CIFAR100) achieve between 2% to 8% error drop.

The mediocre performance of both CIFAR10 and CIFAR100 is due to the dimension constraints of the original images. With an original RGB image size of $32 \times 32 \times 3$, applying 2D-DCT according to a partition of 2×2 will result in a DCT input tensor of $(16 \times 16) \times (2 \times 2) \times 3$. The mapping of the 2×2 partition into the layer dimension will produce a final tensor of $16 \times 16 \times 12$. There is more information loss when compared with the standard RGB input. By applying 2D-DCT with a partition of 8×8 on the standard RGB input of $224 \times 224 \times 3$, it will produce a corresponding DCT input of $28 \times 28 \times 192$. Only the Adapt-DCT variant with a spatial dimension of 2×2 is applied on CIFAR10 and CIFAR100. This is because CIFAR datasets are pre-processed with a DCT partition of 2×2. Applying larger spatial dimension results in negligible performance improvement from the preliminary experiment conducted, hence it is excluded from the analysis. Another variant from the VGG-16 baseline model that replaced all the convolutions with Adapt-DCT kernels was implemented without the M-Skipped architecture. This variant produced the worst result over most of the datasets. Hence, this baseline variant was excluded from the analysis.

Table 4.17: Best performing test error comparison between best performing Adapt-DCT CNN, M-Skipped-3, and normal VGG-16 pointwise CNN.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
VGG16-PC	0.1717	0.2720	0.3419	0.2489	0.4123	0.2253
MS3-base	0.2071	0.2448	0.2194	0.2489	0.3780	0.1807
Best Adapt-DCT variant	**0.0624**	**0.2408**	**0.2022**	**0.1778**	**0.3675**	**0.1560**
Error Change (Best variant vs baseline)	-10.93%	-3.12%	-13.97%	-7.11%	-4.48%	-6.93%
Error Change (Best variant vs M-Skipped-3)	-14.47%	-0.40%	-1.72%	-7.11%	-1.05%	-2.47%

Table 4.18: Best performing test error comparison between best performing Adapt-DCT CNN, M-Skipped-3, and normal VGG-16 pointwise CNN.

		Top 5	Top 5
Datasets / **Model**	Cifar-10C	Cifar-100C	Flowers-104C
VGG16-PC	0.3938	0.3765	0.2099
MS3-base	0.4437	0.3818	0.1743
Best Adapt-DCT variant	**0.3683**	**0.3232**	**0.1699**
Error Change (Best variant vs baseline)	-2.55%	-5.33%	-4.00%
Error Change (Best variant vs M-Skipped-3)	-7.54%	-5.86%	-0.44%

In this final part of the ablation study, an investigation into varying the spatial dimensions of the Adapt-DCT kernel implemented onto each convolutional block is attempted. This background concept is built based on the idea that early convolutional features contain lower-level fundamental patterns while deeper features consist of complex features. The full architecture is denoted as 'MS3-AD-842-org' and presented in section 3.5. This model is implemented on 6 FGVC datasets and 1 general benchmark dataset (mini-image net of 100 classes). It is intended to study the error change between varying spatial dimensions of Adapt-DCT with respect to the best-performing variant. According to Table 4.19, all of the FGVC datasets fall short on error by up to +1.78% when compared to the best-performing Adapt-DCT variant. The general benchmark dataset (mini-image net) obtained a top-5 classification accuracy of 73.93% with Adapt-DCT CNN as compared with similar work which attained a top-1 accuracy of 84.81% on few-shot learning [33].

Table 4.19: Performance comparison with varying spatial dimensions for Adapt-DCT kernel with the best Adapt-DCT variant.

Abbreviation	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
Best Adapt-DCT variant	**0.0624**	**0.2408**	**0.2022**	**0.1778**	**0.3675**	**0.1560**
MS3-AD-842-org	0.0657	0.2564	0.2083	0.1956	0.3710	0.1627
Error Change	+0.33% (↑)	+1.56% (↑)	+0.61% (↑)	+1.78% (↑)	+0.35% (↑)	+0.67% (↑)

4.2.5 Summary of top performing Adapt-DCT variants for FGVC datasets

The classification error and convergence speed comparison between all Adapt-DCT variants are presented in Tables II and III in the Appendix. The errors are ranked based on 3 of the lowest classification errors. These metrics for each dataset are bolded. Their corresponding variants are listed in the last 3 rows in the table. The models that were tested in the ablation study are indicated with parenthesis. From Table II, MS3 AD-842 from the ablation study attained the top-3 lowest error on 4 (Covid-19, Monkey, Spider, Snake Breed) out of 6 of the datasets. By purely considering the main experiments besides the ablation study, MS3-AD-0816 and MS3-AD-0404 achieved the top-3 lowest error on 3 out of 6 of the datasets respectively. Given that MS3-AD-0816 and MS3-AD-0404 carry 103mil and 25mil number of trainable parameters respectively, an optimal variant that will suffice the trade-off between performance and compression gain would be MS3-AD-0404.

4.3 Hybrid Modified Efficient Channel Attention

4.3.1 Experimental results and discussion of HyMod-ECA

CBAM and ECA were initially explored in preliminary experiments by attaching both attention modules respectively into the MS3-AD-0404 variant, adopted from the baseline Adapt-DCT CNN. It was found that ECA attained better performance (i.e., a lower classification error rate of 2-5%) on the same baseline experimental setup as compared to CBAM. Therefore, ECA was selected over CBAM as the baseline channel attention module for Adapt-DCT CNN. To determine where to add the HyMod-ECA module after the last convolution block, both scenarios where the HyMod-ECA module was added after the last convolution block before and after concatenating the M-Skipped feature were tested. The results show that the latter performs better.

The general performance metric used in this section is the classification error (ε) on test sets. Bold text showcases the top-performing variants out of all the attention mechanism counterparts on the same baseline model. In Table 4.20, four types of attention variants added on MS3-AD-0404 are compared against each other. The error change between the original ECA and the HyMod-ECA is explicitly compared to draw a clear indication of the improvement made with the proposed HyMod-ECA. In Table 4.20, the HyMod-ECA is compared against the MS3-AD-0404 and the best-performing Adapt-DCT CNN without attention.

Table 4.20: Performance comparison in terms of classification error for all attention variants on Adapt-DCT CNN.

Abbreviation	Flowers	Monkey	Snake Breed	Snake Breed	Butterfly
MS3-0404 ECA-ORG	0.2231	0.2169	0.2133	**0.3673**	0.1620
MS3-0404 ECA-AD	**0.2205**	**0.2059**	**0.1778**	0.3734	**0.1580**
MS3-0404 ECA-AD-1C1A	0.2436	**0.2059**	0.2155	0.3802	0.1833
MS3-0404 ECA-ADFC	0.2410	0.2096	0.3133	0.3855	0.1827
Error Change (Original ECA vs ECA-AD)	-0.26% (↓)	-1.10% (↓)	-3.55% (↓)	+0.61% (↑)	-0.40% (↓)

According to Table 4.20, HyMod-ECA achieves the best performance on most of the FGVC datasets across the board except for Snake Breed, where applying the original ECA module attains better performance over the other variants. By purely comparing the proposed HyMod-ECA over the original ECA, the FGVC datasets exhibit performance improvements of up to 3.55%. Only a slight error increase of 0.61% can be observed in the Snake Breed. The results are strongly suggestive of the fact that the utilization of DCT channel group interaction by implementing HyMod-ECA outperforms the original ECA on medium FGVC tasks. The original ECA exhibited better performance than the HyMod-ECA on the snake dataset. It is suggested that the individual DCT channel interaction is being prioritised over intra-group interaction. The result indicates that medium FGVC tasks generally prioritise intra-group DCT channel interaction over individual channel relationships except for Snake Breed.

Table 4.21: Performance comparison in terms of classification error of HyMod-ECA Adapt-DCT CNN with its baseline counterpart without attention and the best performing Adapt-DCT CNN for the dataset without attention.

Abbreviation	**Flowers**	**Monkey**	**Snake Breed**	**Snake Breed**	**Butterfly**
Best variation from the previous chapter	0.2359 (MS3-AD-842-04)	0.2022 (MS3-AD-0816)	0.1778 (MS3-AD-0204)	0.3675 (MS3-AD-0416)	0.1560 (MS3-AD-0804)
MS3-AD-0404 (Baseline)	0.2408	0.2108	0.2134	0.3724	0.1653
MS3-0404 ECA-AD	**0.2205**	**0.2059**	**0.1778**	**0.3734**	**0.1580**
Error Change (Baseline vs ECA-AD)	**-2.03% (↓)**	**-0.49% (↓)**	**-3.56% (↓)**	**+0.10% (↑)**	**-0.73% (↓)**
Error Change (Best vs ECA-AD)	**-1.54% (↓)**	**+0.37% (↑)**	**-0.00%**	**+0.59% (↑)**	**+0.20% (↑)**

As shown in Table 4.21, when comparing the HyMod-ECA CNN against its baseline variant without attention, the most improved performance is found on the Spider. It achieves up to a 3.56% error drop with the adoption of HyMod-ECA as compared with its baseline model without attention. On the flip side, a negligible increase in error rate of 0.10% was observed on the Snake. By comparing the HyMod-ECA with the best-performing variants of Adapt-DCT CNN without attention mechanism, an improved performance was found on the Flowers of 1.54%, while the Spider achieves the same performance. Minor error rate increment was found on the other FGVC datasets, but these were maintained below 1%, which is minimal. The comparison against the best-performing model without attention is to provide an advanced and extensive benchmark against the original baseline counterpart.

The Spider and Flowers obtained up to 2% and 3.56% performance improvements by using HyMod-ECA over its baseline counterpart without attention. This suggests that these two fine-grained datasets contain more DCT channel groups of fine-grained object parts that fully utilise the intra-group DCT channel interaction mechanism of HyMod-ECA. On the other hand, the result also suggests Monkey and Butterfly consist of lesser DCT channel group features that can take advantage of the DCT channel group interaction of the HyMod-ECA module. Figures 4.5(a) and 4.5(b) show a few samples from different classes of Spider and Monkey. The Snake Breed

overall has a higher error rate than all other FGVC datasets. A minor drop of 0.10% in performance does not conclusively imply that implementing DCT channel interaction is worse.

The intention of comparing the HyMod-ECA against its baseline counterpart without attention is to prove that the inclusion of weight sharing across channel groups on top of spatial pointwise convolution of the Adapt-DCT kernel can benefit FGVC in the DCT domain. Intra-group DCT channel interactions are not sufficiently captured by solely adopting the Adapt-DCT kernel, thus, with the proposed attention module, the performance of the baseline model can be improved.

(a) Spider [166] visualisation from three different classes of dataset.

(b) Monkey [165] from three different classes of dataset.

Figure 4.5: Several samples from a subset of classes of the fine-grained monkey [165] and spider datasets [166].

The number of trainable parameters for all four variants of the attention mechanism is listed in Table 4.22. Table 4.23 compares the performance difference with the inclusion of fully connected (FC) layers in HyMod-ECA while Table 4.24 measures the performance difference as a result of optimizing HyMod-ECA. From Table 4.23, the inclusion of FC layers will cause a significant performance degradation across all the FGVC datasets as compared with the original HyMod-ECA model, ranging from an increase in error rate of 0.37% to 13.55%. The DCT features from different groups are combined and flattened when going through the FC layers. It is suggested that the FC layers introduce additional transformation and non-linearity which can cause the frequency information to be less distinctive. Combining the ablation study of integrating FC layers into the M-Skipped-3 CNN, one can suggest that FC layers will not ease FGVC in the DCT domain.

Table 4.22: Number of trainable parameters for each attention mechanism variant.

Abbreviation	Trainable Parameters (mil)
MS3-0404 ECA-ORG	25
MS3-0404 ECA-AD	25
MS3-0404 ECA-AD-1C1A	6.72
MS3-0404 ECA-ADFC	32

Table 4.23: Performance comparison in terms of classification error of HyMod-ECA with and without fully connected layers.

Abbreviation	Flowers	Monkey	Snake Breed	Snake Breed	Butterfly
MS3-0404 ECA-AD	0.2205	0.2059	0.1778	0.3734	0.1580
MS3-0404 ECA-ADFC	0.2410	0.2096	0.3133	0.3855	0.1827
Error Change	**+2.05% (↑)**	**+0.37% (↑)**	**+13.55% (↑)**	**+1.21% (↑)**	**+2.47% (↑)**

Table 4.24: Performance comparison in terms of classification error with the optimisation of HyMod-ECA.

Abbreviation	Flowers	Monkey	Snake Breed	Snake Breed	Butterfly
MS3-0404 ECA-AD	0.2205	0.2059	0.1778	0.3734	0.1580
MS3-0404 ECA-AD-1C1A	0.2436	0.2059	0.2159	0.3802	0.1833
Error Change	**+2.31% (↑)**	**-0.00%**	**+3.81% (↑)**	**+0.68% (↑)**	**+2.35% (↑)**

4.3.2 Ablation Study

The comparison metrics among the Hybrid CBC algorithm, M-Skipped-3 DCT-CNN, Adapt-DCT CNN, and HyMod-ECA DCT-CNN are presented in Table 4.25. The M-Skipped-3 DCT CNN demonstrates a promising compression ratio, achieving up to 5.23 times in comparison to its Hybrid CBC counterpart. Meanwhile, the baseline model of the HyMod-ECA (Adapt-DCT-0404) attains the fastest learning rate and convergence speed ratio compared to the hybrid CBC. Impressively, this mode accomplishes a classification error rate below 25% within 12 training epochs. The best-performing variant in this study achieves a minimal error rate of 20.56%, showcased by the optimized HyMod-ECA model. Although a slight increase in error of approximately 7% was exhibited when compared to the Hybrid CBC, the HyMod-ECA CNN in the DCT domain exhibited a model compression ratio up to 1.29 times and a learning speed 5.6 times faster. In essence, while the developed method incurs a slight increase in error, leading to a less accurate model, it successfully achieved the primary objective of this research by delivering higher compression gain and faster learning speed in the compressed domain. Despite a little compromise in model accuracy, the compressed domain CNN presents three significant advantages: a reduced model size with fewer parameters, accelerated learning speed with diminished computational complexity, and the capability to fully operate in the DCT domain when handling input images of DCT nature.

Table 4.25: Comparison of several metrics (rows) between the developed M-Skipped-3, Adapt-DCT CNN, and HyMod-ECA versus the comparable model in CBC on a 10-classes Monkey FGVC dataset. Bold text shows the best results for each row.

	VGG-CBC-SpFw [105]	**M-Skipped-3**	**Adapt-DCT-0404**	**HyModECA-AD-1C1A**
Best Error Rate (%)	**13.13**	21.94	21.08	20.59
Convergence Speed	96	15	**12**	17
Number of trainable parameters	8,726,000	**1,666,990**	25,624,622	6,745,462
Compression Ratio	1.00	**5.23**	0.34	1.29
Convergence Speed Ratio	1.00	6.4	**8.0**	5.6

Table 4.26 presents the intra-class scores between the concluding model developed in this research and the standard VGG-16 algorithm. This final experiment which was conducted on the heterogeneous combined datasets of Flowers and Leeds Butterfly, offers a comprehensive FGVC usage comparison based on several metrics. The results indicate that the standard VGG-16 implemented in the RGB domain attained better FGVC analysis with higher intra-class scores. The overall F1-score obtained from the standard VGG-16 algorithm is 8.13% higher than that of the developed system in this research. While the sensitivity and specificity of the standard VGG-16 algorithm shared a better performance of 8.51%, it also exhibited higher precision by 5.84% compared to the developed system. This suggests that the spatial context in RGB images remains relatively relevant for composing discriminative features for FGVC analysis in classical CNNs. However, as shown in Table 4.27, the standard VGG-16 has remarkably more parameters and slower convergence speed compared to the model in this work. Specifically, a compression ratio of 20 times and a convergence speed ratio of 1.2 times that of the standard VGG-16 were observed. Regardless of the classical approaches, the implementation of compressed domain CNNs that focus on the basis functions and frequency analytics still provides a

comparable performance yet achieves better compression gain as addressed in the objective of this research.

Table 4.26: Overall intraclass metrics comparison of the concluding model in this research about the standard VGG-16 algorithm.

Abbreviation	**Precision**	**Sensitivity**	**Specificity**	**F1-Score**
MS3-0404 ECA-AD (DCT)	76.58	70.46	70.46	70.64
Standard VGG-16 (RGB)	82.42	78.97	78.97	78.77
Accuracy Change	-5.84% (↓)	-8.51% (↓)	-8.51% (↓)	-8.13% (↓)

Table 4.27: Performance metrics comparison of the concluding model in this research about standard VGG-16 algorithm.

Evaluation Metrics / Model	Convergence Speed (epochs)	Trainable Parameters (mil.)	Compression Ratio	Convergence Speed Ratio
MS3-0404 ECA-AD (DCT)	17	6.7	20	1.2
Standard VGG-16 (RGB)	20	134.3		

4.4 Summary

In M-Skipped DCT-CNN, the integration of M-DCTCs with low DCTCs was implemented in the FGVC domain. The work addresses a research gap concerning the hindrance of feature learning concerning higher DCTC bands in a compressed domain FGVC context. Feature correlation issues in FGVC datasets exist interchangeably between spatial and compressed domains. Discriminative localized fine-grained feature representations lie naturally within the spectrum of DCT coefficients and hence can be extracted with ease in the DCT domain. The learning process of fine-grained information in the frequency domain is arguably simpler when compared to the spatial domain. This chapter answers the research question of whether and how appropriate learning of M-DCTCs, in addition to L-DCTC learning, can be implemented to improve representation learning in FGVC. A systematic approach for testing different frequency ranges verified that L-DCTC is the main essential range for feature extraction in FGVC. Overall, a reduction of about 7% classification error rate was achieved by utilizing an M-skipped connection when compared to the model without the skip-connection in FGVC, whereas minimal performance difference was obtained in a non-FGVC comparison dataset. The M-Skipped DCT-CNN has an overall better F1-Score than the standard VGG-16 algorithm by 2.2%. More importantly, it dominates the standard algorithm with a compression ratio of up to 15 times. It is crucial for compressed domain FGVC to emphasise higher frequency bands without significantly increasing computational complexity. The parameter reduction and improved F1-Score achieved by the M-Skipped network address the research problem of pruning across different frequency levels, resulting in a smaller, yet high-performing model. Typically, FGVC models are challenging to deploy due to the large number of parameters required to extract discriminative features, as fine-grained details are often less distinguishable across different classes. The success of the M-Skipped DCT-CNN enables its deployment in real-world applications with limited computational resources, such as mobile devices, embedded systems, or environments with constrained processing power. This network offers a scalable and efficient solution for deployment across various platforms.

The work is further extended by incorporating adaptive DCT basis functions on kernel analytics on top of M-Skipped DCT CNN. Subsequently, a novel Adaptive DCT (Adapt-DCT) convolution kernel is developed. The Adapt-DCT kernel is intended to form the pointwise convolution kernel which is generally found in compressed domain

DCT related CNNs. The Adapt-DCT convolution kernel carries trainable parameters to weight each of the DCT basis functions of spatial and frequency base respectively. A final pointwise convolution kernel is produced from the result of applying element-wise multiplication and spatial summation between the Adapt-DCT kernel and the DCT basis functions. It involved weighting the DCT basis functions to form the convolutional kernel. This technique provides an early approach to improve the robustness of pointwise convolutional filters in the compressed domain as compared with the classical 3×3 filter in the spatial domain. With the replacement of the Adapt-DCT kernel on the original pointwise convolution kernel, FGVC datasets obtained between 1% to 8% classification error reduction. For general classification tasks, CIFAR datasets obtained a 2% to 8% error rate drop. The general benchmark classification task on Tiny ImageNet of 100 classes using Adapt-DCT CNN achieved a top-5 accuracy of 73.93%. The performance improvements from the prior experimental setups demonstrate that the compressed domain approach can effectively address more challenging FGVC problems. It introduces a DCT-based methodology tailored for FGVC, where highly detailed and localised feature extraction is essential. This approach enhances the efficiency of convolutional kernels while minimising the need for larger, more complex algorithms. By leveraging a DCT-based kernel formulation, the methodology fosters a more robust and dynamic convolutional algorithm, advancing its importance in both general and fine-grained classification tasks.

In the final section, a novel concept of capitalizing on intra-group DCT channel interactions by modifying the channel attention mechanism was proposed. The module is inherited and improved based on a popular attention mechanism called 'Efficient Channel Attention' (ECA) to fit the usage of this research for FGVC in the DCT domain. A channel attention mechanism prioritising channel interactions within DCT groups was developed, namely 'Hybrid Modified ECA' (HyMod-ECA). It serves the objective of exploring intra-group DCT channel interaction and relationship on top of the adaptive learning of DCT basis function in FGVC in the frequency domain. The development of HyMod-ECA pushes the boundaries of convergence speed and parameter reduction through the implementation of a DCT-based attention mechanism. The results demonstrated a notable reduction in parameters and improved convergence speed, significantly shortening the overall training and deployment timeline compared to conventional methods, as the model converges more quickly in the compressed domain. This attention mechanism addresses the previously overlooked intra-group DCT

channel relationships, which are crucial for optimising performance in FGVC tasks. Although there is a slight decline in F1-Score compared to the contemporary Hybrid CBC mechanism, HyMod-ECA still achieves the overall research objective by delivering competitive performance alongside resource optimisation. The HyMod-ECA is demonstrated on top of M-Skipped DCT-CNN which involves the Adapt-DCT kernel. It is found to outperform the original ECA on several medium FGVC datasets. The successful implementation of HyMod-ECA achieves up to 3.5% classification error reduction over the original ECA and the prior Adapt-DCT baseline model without an attention module. Besides, the optimisation of the baseline Adapt-DCT model with HyMod-ECA attains a parameter reduction of up to 73% with no performance degradation on Monkey-10C. It is proven that the intra-group DCT channel interactions carry more importance over cross-channel interactions in the DCT domain FGVC.

Chapter 5 Conclusion and Future Works

This research implements DCT-based methodologies within CNNs for compressed domain small-scale FGVC. This involves the integration of the HyMod-ECA module into the M-Skipped DCT-CNN framework, which consists of the Adapt-DCT convolutional kernel to accomplish the aim of this work.

In a preliminary exploration of adopting low, medium, and high-frequency coefficients (L-DCTCs, M-DCTCs, H-DCTCs) independently through a fully pointwise VGG-16 network, it became evident that L-DCTCs and M-DCTCs share essential features for effective classification. As a result, a novel architectural branch called the M-Skipped DCT connection was developed to ease the passage of low-level features from the shallowly convolved M-DCTCs, allowing them to skip through certain network segments and combine with the deeply convolved L-DCTCs. Integrating the M-Skipped branch with the output from the final convolutional block (M-Skipped-3) yielded significant improvements. It achieved a notable 7.5% reduction in classification error compared to its baseline counterpart without the M-Skipped branch on Leeds Butterfly. The M-Skipped-3 variant demonstrated an average classification error of 18.95% over five FGVC datasets in contrast with the baseline model's error rate of 22.44%, resulting in an average error reduction of 3.49%. Furthermore, the M-Skipped-3 algorithm showed an average F1-Score of 83.41% across the same datasets, surpassing the standard VGG-16 algorithm's score of 81.21% by 2.2%. Remarkably, the M-Skipped-3 network comprised only 1.7 million parameters, while the standard algorithm contained a substantial 134.4 million parameters. This shows that the developed algorithm offers a parameter reduction of 98%. This section addressed the research gap concerning the optimal learning and integration of frequency bands beyond L-DCTCs.

Based on the proven benefits of the M-Skipped DCT CNN, the research progressed towards integrating the DCT methodology into a pointwise convolutional kernel within the same CNN framework. This novel approach involved developing an adaptive learning mechanism to weigh the DCT basis functions, resulting in the construction of the adaptive DCT (Adapt-DCT) convolutional kernel. The spatial and

frequency properties of the Adapt-DCT kernel were thoroughly evaluated. The Adapt-DCT variant which carries an adaptive weight with a spatial size of 4×4 and an optimized frequency channel of 4 is denoted as MS3-AD-0404. This variant achieved promising results with an average classification error of 21.31% and a convergence speed of 11.80 epochs across 6 FGVC datasets. When comparing these results against the former M-Skipped DCT CNN, which exhibited an average classification error of 24.65% and convergence speed of 13.00 epochs over the same datasets, the MS3-AD-0404 variant attained a notable 1.1 times faster convergence speed and a 3.34% reduction in classification error. Furthermore, with the MS3-AD-0404 variant containing 25.85 million parameters compared to the standard VGG-16 algorithm's 134.4 million, a compression gain of 5.2 times was achieved. The MS3-AD-0404 strikes an optimal balance between parameter reduction and convergence speed. Additionally, the robustness of the Adapt-DCT kernel was also tested on a general dataset. It attained a top-5 testing accuracy of 73.93% on tiny images from 100 classes. Several contributions were made including the modulation of spatial size and frequency coefficient pruning of DCT-BFs to produce a kernel. These contributions addressed a research gap in compressed domain CNNs by adaptively expediting the DCT-BFs to construct a more robust pointwise convolutional kernel through spatial and frequency analytics.

In the concluding phase of this research, the focus was placed on the interaction among DCT channel groups through an attention mechanism referred to as 'Hybrid Modified Efficient Channel Attention' (HyMod-ECA). This innovative approach was evaluated on five distinct FGVC datasets. It reveals a notable performance enhancement compared to both the original ECA and the baseline model without an attention module in most datasets. HyMod-ECA exhibited a drop in classification error of up to 3.5%. Moreover, an ablation study was conducted, where certain convolutional layers were removed from each baseline convolutional block while keeping the HyMod-ECA to increase compression gain. Particularly, the Monkey-10C achieved a parameter saving of 73% compared to the baseline HyMod-ECA model without sacrificing performance. This dataset also achieved a convergence speed ratio of 1.2 times faster and a compression ratio of 20 times greater than the standard VGG-16 algorithm, with a slight decline in F1-Score by 5.84%. This novelty significantly contributes to the understanding of the utility of intra-group DCT channel interactions alongside the individual cross-channel relationships in DCT domain FGVC. The integration of the

HyMod-ECA module on top of the Adapt-DCT M-Skipped CNN concurrently addressed the research gap involving the interpretability and correspondence between DCT channel groups.

This research exhibits the successful integration of CNNs in compressed domain FGVC using DCT-based techniques. The three major innovations are the M-Skipped-3 convolutional skipping branch, the adaptive DCT kernel, and the hybrid modified efficient channel attention mechanism. The outcomes and insights derived by these innovations involve comparable performance, enhanced convergence speed ratios, and improved compression gain in relation to the spatial domain. These achievements collectively demonstrate how the objectives of this research were achieved. They are facilitated through the following contributions:

1. The study of individual LMH-DCTCs and integration of M-DCTCs was facilitated through the development of the M-Skipped DCT CNN. This approach yielded an average classification error reduction of 3.49%, a 2.2% increase in the F1-score, and a 98% reduction in parameters compared to the standard VGG-16 algorithm. This outcome offers an alternative approach to the pruning problem frequently encountered within the context of compressed domain FGVC. By extending the analysis beyond conventional L-DCTC frequency bands, the research objective of integrating diverse frequency levels to enhance FGVC performance in compressed domain applications is addressed.

2. The study of spatial and frequency properties of DCT basis functions to formulate a convolutional kernel was facilitated through the development of the Adaptive DCT pointwise convolutional kernel. This technique led to a 1.1 times improvement in convergence speed and a 3.34% reduction in classification error compared to the M-Skipped DCT CNN without the Adapt-DCT kernel, along with a compression gain of 5.2 times relative to the standard VGG-16 algorithm. This outcome addresses the limitations of conventional convolutional kernel composition, which often leads to excessive parameters and reduced robustness. Through employing DCT basis functions, the research objective to enhance the kernel formulation is met, thereby reducing computational complexity and achieving significant resource optimisation.

3. The study of interactions between DCT channel groups was facilitated through the development of HyMod-ECA. This algorithm achieved an average classification error drop of 1% compared to the original ECA. It also delivered a 1.2 times increase in

convergence speed and a 20 times compression ratio relative to the standard VGG-16 algorithm, with a minor drop in F1-Score by 5.84% on the Monkey-10C. This outcome solves the problem of attention mechanisms that solely focus on individual feature channels. By integrating the relationships and interactions among feature channels, the research objective is met, thus enabling improved resource efficacy in compressed domain FGVC.

While feature modification and model optimisation can enhance the efficacy of compressed domain FGVC, the unavoidable challenges following information loss during the compression of fine-grained images impose limitations. This hinders the developed algorithm from achieving desirable accuracy performance in FGVC. Nonetheless, the primary scope of this research is to explore compressed domains to tackle information redundancy in FGVC. Although the developed algorithm achieved the research objectives, it is crucial to address future work toward enhancing the model robustness in terms of classification accuracy while concurrently maintaining a comparable level of compression gain.

In the majority of experiments conducted, VGG-16 served as the foundational baseline model. An avenue for extending this research involves the application of the developed techniques to other SOTA CNN architectures such as ResNet and MobileNet. This extension could offer insights into the adaptability of the proposed methodologies across diverse network architectures. Besides, a promising direction for future research lies in exploring dynamic mechanisms integrated within the network to further enhance compression gains. Specifically, about the Adapt-DCT kernel, the network architecture could be designed to dynamically assign varying sets of DCT basis functions based on the interaction between feature maps and the convolving kernel. This dynamic adaptation could enable the network to learn assorted combinations of DCT basis functions tailored to represent the weights optimally. Regarding the HyMod-ECA module, improvements could involve the continuous formulation of 1D convolutional kernel sizes throughout model training. For instance, the integration of dilated convolutions could be explored to investigate various combinations of DCT channel groups to form higher-level and more complex representations. Finally, further exploration involves the implementation of a random mixture of individual and group DCT channel interactions. This approach could uncover novel insights into the

harmonic effects of combining different types of DCT channel interactions thus improving the model's representational capacity.

In conclusion, compressed domain image analytics and concomitant neural architectural innovations can reduce information redundancy and hence improve the robustness of a model. This thesis has proposed several innovations in this direction and, it is hoped, has opened several useful directions for further improvements.

Bibliography

[1] D. Ruby, “ChatGPT Statistics for 2023: Comprehensive Facts and Data,” Demand Sage. Accessed: Mar. 07, 2023. Available: https://www.demandsage.com/chatgpt-statistics/.

[2] E. B. Picaro, “What is Apple’s Photonic Engine and how does it boost low-light photos?”, Pocket-Lint. Accessed: Mar. 07, 2023. Available: https://www.pocket-lint.com/phones/news/apple/162585-apple-photonic-engine-low-light-photos-how-does-it-work-iphone-camera/.

[3] D. Ruby, “26+ iPhone User & Sales Statistics (Fresh Data 2023),” Demand Sage. Accessed: Mar. 07, 2023. Available: https://www.demandsage.com/iphone-user-statistics/.

[4] W. Rawat and Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017, doi: https://doi.org/10.1162/neco_a_00990.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Jun. 2014, doi: https://doi.org/10.1109/cvpr.2014.81.

[6] “Convolutional Neural Networks,” Bayern Collab. Accessed: Mar. 07, 2023. Available: https://wiki.tum.de/display/lfdv/Convolutional+Neural+Networks.

[7] J. P. Bharadiya, “Convolutional Neural Networks for Image Classification,” *International Journal of Innovative Science and Research Technology*, vol. 8, no. 5, pp. 673–677., May 2023, doi: https://doi.org/10.5281/zenodo.7952031.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Jun. 2016, doi: https://doi.org/10.1109/cvpr.2016.90.

[9] Z. Wu, C. Shen, and A. van den Hengel, “Wider or Deeper: Revisiting the ResNet Model for Visual Recognition,” *Pattern Recognition*, vol. 90, pp. 119–133, Jun. 2019, doi: https://doi.org/10.1016/j.patcog.2019.01.006.

[10] Z. Pan, A. G. Rust, and H. Bolouri, “Image redundancy reduction for neural network classification using discrete cosine transforms,” *Proceedings of the IEEE-*

INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Jul. 2000, doi: https://doi.org/10.1109/IJCNN.2000.861296.

[11] Y. Wang, S. Ye, S. Yu, and X. You, "R2-Trans:Fine-Grained Visual Categorization with Redundancy Reduction," *arXiv.org*, Apr. 21, 2022. Accessed: May 03, 2022. Available: http://arxiv.org/abs/2204.10095.

[12] C. Deng, Q. Chen, X. Zou, E. Xu, B. Tang, and W. Xia, "imDedup: A Lossless Deduplication Scheme to Eliminate Fine-grained Redundancy among Images," *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, May 2022, doi: https://doi.org/10.1109/icde53745.2022.00085.

[13] Z. Wang, C. Li, and X. Wang, "Convolutional Neural Network Pruning with Structural Redundancy Reduction," *arXiv.org*, Apr. 07, 2021. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2104.03438.

[14] Y. Wang, K. Lv, R. Huang, S. Song, L. Yang, and G. Huang, "Glance and Focus: a Dynamic Approach to Reducing Spatial Redundancy in Image Classification," *arXiv.org*, Oct. 11. 2020, [Online]. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2010.05300.

[15] J. Liang, T. Zhang, and G. Feng, "Channel Compression: Rethinking Information Redundancy Among Channels in CNN Architecture," *IEEE Access*, vol. 8, pp. 147265–147274, 2020, doi: https://doi.org/10.1109/access.2020.3015714.

[16] R. Zheng, Z. Yu, Y. Zhang, C. Ding, H. V. Cheng, and L. Liu, "Learning Class Unique Features in Fine-Grained Visual Classification," *arXiv.org*, Mar. 16, 2021. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2011.10951.

[17] D. Zhai, X. Zhang, X. Li, X. Xing, Y. Zhou, and C. Ma, "Object detection methods on compressed domain videos: An overview, comparative analysis, and new directions," *Measurement*, vol. 207, p. 112371, Feb. 2023, doi: https://doi.org/10.1016/j.measurement.2022.112371.

[18] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learning Image and Video Compression through Spatial-Temporal Energy Compaction," *arXiv.org*, Jun. 27, 2019. Accessed: May 03, 2022. Available: http://arxiv.org/abs/1906.09683.

[19] W. Xiao, N. Wan, A. Hong, and X. Chen, "A Fast JPEG Image Compression Algorithm Based on DCT," *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, Nov. 2020, doi: https://doi.org/10.1109/SmartCloud49737.2020.00028.

[20] B. Rajesh, M. Javed, Ratnesh, and S. Srivastava, "DCT-CompCNN: A Novel Image Classification Network Using JPEG Compressed DCT Coefficients," *2019 IEEE*

Conference on Information and Communication Technology, Dec. 2019, doi: https://doi.org/10.1109/cict48419.2019.9066242.z.

[21] R. V. Babu, M. Tom, and P. Wadekar, "A survey on compressed domain video analysis techniques," Multimedia Tools and Applications, vol. 75, no. 2, pp. 1043–1078, Nov. 2014, doi: https://doi.org/10.1007/s11042-014-2345-z.

[22] M. Ulicny and R. Dahyot, "On using CNN with DCT based Image Data," *Proceedings of the 19th Irish Machine Vision and Image Processing conference*, Aug. 2017.

[23] K. Cheinski and P. Wawrzynski, "DCT-Conv: Coding filters in convolutional networks with Discrete Cosine Transform," *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, doi: https://doi.org/10.1109/ijcnn48605.2020.9207103.

[24] K. Xu, M. Qin, F. Sun, N. C. Beaulieu, Y.-K. Chen, and F. Ren, "Learning in the Frequency Domain," *Computer Vision and Pattern Recognition*, Jun. 2020, doi: https://doi.org/10.1109/cvpr42600.2020.00181.

[25] T. Hossain, Shyh Wei Teng, D. Zhang, S. Lim, and G. Lu, "Distortion Robust Image Classification Using Deep Convolutional Neural Network with Discrete Cosine Transform," *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, doi: https://doi.org/10.1109/icip.2019.8803787.

[26] M. Ulicny, V. A. Krylov, and R. Dahyot, "Harmonic Networks: Integrating Spectral Information into CNNs," *arXiv.org*, Dec. 07, 2018. Accessed: Mar. 07, 2023. Available: https://arxiv.org/abs/1812.

[27] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv.org*, Feb. 2016. Accessed: Mar. 07, 2023. Available: https://arxiv.org/abs/1510.00149.

[28] S. Herbreteau and C. Kervrann, "DCT2net: An Interpretable Shallow CNN for Image Denoising," *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 31, pp. 4292–4305, 2022, doi: https://doi.org/10.1109/TIP.2022.3181488.

[29] Z. Wang, Y. Yang, A. Shrivastava, V. Rawal, and Z. Ding, "Towards Frequency-Based Explanation for Robust CNN," *arXiv.org*, May 2020. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2005.03141.

[30] K. Gao, H. Aliakbarpour, G. Seetharaman, and K. Palaniappan, "DCT-Based Local Descriptor for Robust Matching and Feature Tracking in Wide Area Motion

Imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 8, pp. 1441–1445, Aug. 2021, doi: https://doi.org/10.1109/lgrs.2020.3000762.

[31] X. Shen, J. Yang, C. Wei, B. Deng, J. Huang, X. Hua, X. Cheng, and K. Liang, "DCT-Mask: Discrete Cosine Transform Mask Representation for Instance Segmentation," *arXiv.org*, Apr. 27, 2021. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2011.09876.

[32] X. Chen and G. Wang, "Few-Shot Learning by Integrating Spatial and Frequency Representation," *arXiv.org*, Jul. 06, 2021. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2105.05348.

[33] A. Yang, M. Li, Z. Wu, Y. He, X. Qiu, Y. Song, W. Du, and Y. Gou, "CDF-net: A convolutional neural network fusing frequency domain and spatial domain features," *IET computer vision*, vol. 17, no. 3, pp. 319–329, Feb. 2023, doi: https://doi.org/10.1049/cvi2.12167.

[34] R. Du, D. Chang, A. K. Bhunia, J. Xie, Z. Ma, Y. Z. Song, and J. Guo, "Fine-Grained Visual Classification via Progressive Multi-granularity Training of Jigsaw Patches," *Lecture notes in computer science*, pp. 153–168, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-58565-5_10.

[35] H. Zheng, J. Fu, Zheng-Jun Zha, J. Luo, and T. Mei, "Learning Rich Part Hierarchies With Progressive Attention Networks for Fine-Grained Image Recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 476–488, Jan. 2020, doi: https://doi.org/10.1109/tip.2019.2921876.

[36] F. Zhang, L. Liu, G. Zhai, and Y. Liu, "Multi-branch and Multi-scale Attention Learning for Fine-Grained Visual Categorization," *Springer eBooks*, pp. 136–147, Jun. 2021, doi: https://doi.org/10.1007/978-3-030-67832-6_12.

[37] X. Dai, S. Gong, S. Zhong, and Z. Bao, "Bilinear CNN Model for Fine-Grained Classification Based on Subcategory-Similarity Measurement," *Applied Sciences*, vol. 9, no. 2, p. 301, Jan. 2019, doi: https://doi.org/10.3390/app9020301.

[38] M. Buckler, N. Adit, Y. Hu, Z. Zhang, and A. Sampson, "Dense Pruning of Pointwise Convolutions in the Frequency Domain," *arXiv.org*, Sep. 16, 2021. Accessed: Mar. 07, 2023. Available: http://arxiv.org/abs/2109.07707.

[39] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-Domain Dynamic Pruning for Convolutional Neural Networks," *2018 Neural Information Processing Systems*, Dec. 2018. Available:

https://proceedings.neurips.cc/paper_files/paper/2018/hash/a9a6653e48976138166de32772b1bf40-Abstract.html.

[40] S. Singh, "Reduction of Blocking Artifacts In JPEG Compressed Image," *arXiv.org*, Feb. 11, 2014. Accessed: Mar. 07, 2023. Available: https://arxiv.org/abs/1210.1192.

[41] P. Najgebauer, R. Scherer, and L. Rutkowski, "Fully Convolutional Network for Removing DCT Artefacts From Images," *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, doi: https://doi.org/10.1109/ijcnn48605.2020.9207249.

[42] S. Khan, M. A. Irfan, A. Arif, S. T. H. Rizvi, A. Gul, M. Naeem, and N. Ahmad, "On Hiding Secret Information in Medium Frequency DCT Components Using Least Significant Bits Steganography," *Computer Modelling in Engineering & Sciences*, vol. 118, no. 3, pp. 529–546, Mar. 2019, doi: https://doi.org/10.31614/cmes.2019.06179.

[43] Q. L. Xu, "A Research on Information Hiding Algorithm Based on Frequency Blocks of DCT Coefficients," *2019 Annual Meeting on Management Engineering*, Dec. 2019, doi: https://doi.org/10.1145/3377672.3378034.

[44] W. W. Yee, P. S. Kah, and L. M. Ang, "M-band wavelet transform in face recognition system," *International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, May 2008, doi: https://doi.org/10.1109/ecticon.2008.4600468.

[45] Y. A. Shah, N. Ahmad, and M. Naeem, "Identification of critical bands in DCT domain representation for fingerprint recognition," *18th International Conference on Automation and Computing (ICAC),* Sep. 2012.

[46] Tan et al., "Classification of Compressed Domain Images Utilizing Open VINO Inference Engine," *International Journal of Engineering and Advanced Technology*, vol. 9, no. 1, pp. 1669–1678, Oct. 2019, doi: https://doi.org/10.35940/ijeat.a2709.109119.

[47] V. Sharma, S. Rai, and A. Dev, "A Comprehensive Study of Artificial Neural Networks," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 10, pp. 278–284, 2012.

[48] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, doi: https://doi.org/10.1037/h0042519.

[49] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of trends in Practice and Research for Deep Learning,” *arXiv.org*, 2018. Accessed: Mar. 07, 2023. Available: https://arxiv.org/abs/1811.03378.

[50] J.-C. B. Loiseau, “Rosenblatt’s perceptron, the very first neural network,” *Towards Data Science*, Mar. 11, 2019. Accessed: Mar. 07, 2023. Available: https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a.

[51] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, no. 61, pp. 85–117, Jan. 2015, doi: https://doi.org/10.1016/j.neunet.2014.09.003.

[52] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: https://doi.org/10.1038/nature14539.

[53] S. Lawrence and C. L. Giles, “Overfitting and neural networks: conjugate gradient and backpropagation,” *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000, doi: https://doi.org/10.1109/ijcnn.2000.857823.

[54] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: https://doi.org/10.1007/bf00344251.

[55] G. Gidion, L. F. Capretz, M. Grosch, and K. N. Meadows, “Trends in Students Media Usage,” *Computational Science and Its Applications – ICCSA 2016*, pp. 491–502, 2016, doi: https://doi.org/10.1007/978-3-319-42085-1_38.

[56] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten Digit Recognition with a Back-Propagation Network,” *Adv Neural Inf Process Syst*, pp. 396–404, 1990. Available: https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

[57] R. H. Nielsen, “Theory of the Backpropagation Neural Network**Based on ‘nonindent’ by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE.,” *Neural Networks for Perception*, pp. 65–93, 1992, doi: https://doi.org/10.1016/b978-0-12-741252-8.50010-8.

[58] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150203, Apr. 2016, doi: https://doi.org/10.1098/rsta.2015.0203.

[59] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural networks see the world --- A survey of convolutional neural network visualization methods," *Mathematical Foundations of Computing*, vol. 1, no. 2, pp. 149–180, 2018, doi: https://doi.org/10.3934/mfc.2018008.

[60] Sneha H.L., "2D Convolution in Image Processing.", *All About Circuits*. Accessed: Dec. 01, 2019. Available: https://www.allaboutcircuits.com/technical-articles/two-dimensional-convolution-in-image-processing/.

[61] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," Nov. 2018. *arXiv.org*, 2018. Accessed: May 26, 2023. Available: http://arxiv.org/abs/1811.03378

[62] K. Dhanasree, "Data Analytics: Role of Activation function In Neural Net," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 5, pp. 299–302, 2019.

[63] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Fully convolutional neural networks for remote sensing image classification," *International Geoscience and Remote Sensing Symposium (IGARSS)*, Jul. 2016, doi: https://doi.org/10.1109/igarss.2016.7730322.

[64] Analytics Vidhya, "Fundamentals of Deep Learning - Activation Functions and their use," *Analytics Vidhya*, Feb. 22, 2019. Accessed: Aug. 21, 2019. Available: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[65] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010. Available: https://icml.cc/Conferences/2010/papers/432.pdf.

[66] R. Prabhu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," *Medium*, Mar. 04, 2018. Accessed: Dec. 01, 2019. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[67] B. Gao and L. Pavel, “On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,” *arXiv.org*, Aug. 20, 2018. Accessed: Dec. 01, 2019. Available: https://arxiv.org/abs/1704.00805.

[68] “Neocognitron - Basic principle of the neocognitron,” Accessed: Dec. 01, 2019. Available: https://www.kiv.zcu.cz/studies/predmety/uir/NS/Neocognitron/en/hierarch-det.html.

[69] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial Intelligence Review*, vol. 53, Apr. 2020, doi: https://doi.org/10.1007/s10462-020-09825-6.

[70] M. Sornam, K. Muthusubash, and V. Vanitha, “A Survey on Image Classification and Activity Recognition using Deep Convolutional Neural Network Architecture,” Dec. 2017, doi: https://doi.org/10.1109/icoac.2017.8441512.

[71] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, Mar. 2007, doi: https://doi.org/10.1080/01431160600746456.

[72] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: https://doi.org/10.1109/5.726791.

[73] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F. F. Li, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015, doi: https://doi.org/10.1007/s11263-015-0816-y.

[74] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: https://doi.org/10.1109/tpami.2016.2577031.

[75] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv.org*, Apr. 10, 2015. Accessed: Dec. 01, 2019. Available: https://arxiv.org/abs/1409.1556.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[77] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *2017 IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), pp. 2261–2269, Jul. 2017, doi: https://doi.org/10.1109/cvpr.2017.243.

[78] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv.org*, Apr. 2017. Accessed: Dec. 01, 2019. Available: http://arxiv.org/abs/1704.04861.

[79] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv.org*, Nov. 2016. Accessed: Dec. 01, 2019. Available: https://arxiv.org/abs/1602.07360.

[80] B. Monien, R. Preis, and S. Schamberger, "ImageNet Classification with Deep Convolutional Neural Networks," *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007. doi: https://doi.org/10.1201/9781420010749.

[81] T. Mantoro and F. Alfiah, "Comparison methods of DCT, DWT and FFT techniques approach on lossy image compression," Nov. 2017, doi: https://doi.org/10.1109/ced.2017.8308126.

[82] U. Bhade, S. Kumar, P. Dwivedy, S. Soofi, and A. Ray, "Comparative study of DWT, DCT, BTC and SVD techniques for image compression," *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, Oct. 2017, doi: https://doi.org/10.1109/rise.2017.8378167.

[83] A. H. M. J. I. Barbhuiya, T. A. Laskar, and K. Hemachandran, "An Approach for Color Image Compression of JPEG and PNG Images Using DCT and DWT," *IEEE Xplore*, Nov. 01, 2014. doi: https://doi.org/10.1109/CICN.2014.40.

[84] D. Mehta and K. Chauhan, "Image Compression using DCT and DWT-Technique", *International Journal of Engineering Sciences Research Technology*, vol. 2, no. 8, pp. 2133-2139, 2013.

[85] P. Rani and A. Arora, "A Survey on Comparison between DCT and DWT Techniques of Image Compression," *International Journal of Science and Research (IJSR)*, 2015.

[86] V. Elamaran and A. Praveen, "Comparison of DCT and wavelets in image coding," *2012 International Conference on Computer Communication and Informatics (ICCCI 2012)*, Jan. 2012, doi: https://doi.org/10.1109/iccci.2012.6158923.

[87] C. E. Harisharan and H. Kaur, “Review of Increasing Image Compression Rate Using (DWT+DCT) and Steganography,” *International Journal of Recent Trends in Engineering and Research*, vol. 3, no. 6, pp. 67–72, Jun. 2017.

[88] O. Ghorbel, I. Jabri, W. Ayedi, and M. Abid, “Experimental study of compressed images transmission through WSN,” *Proceedings of the International Conference on Microelectronics (ICM)*, pp. 1–6, 2011, Dec. 2011, doi: https://doi.org/10.1109/icm.2011.6177378.

[89] S. J. Pinto and J. P. Gawande, “Performance analysis of medical image compression techniques,” *Asian Himalayas International Conference on Internet*, pp. 1–4, 2012, doi: https://doi.org/10.1109/ahici.2012.6408455.

[90] A. K. Sharma, U. Kumar, S. K. Gupta, U. Sharma, and S. Lakshmiagrwal, “A survey on feature extraction technique for facial expression recognition system,” *2018 4th International Conference on Computing Communication and Automation (ICCCA 2018)*, pp. 1–6, 2018, doi: https://doi.org/10.1109/ccaa.2018.8777550.

[91] N. Janu, P. Mathur, S. K. Gupta, and S. L. Agrwal, “Performance analysis of frequency domain based feature extraction techniques for facial expression recognition,” *7th International Conference on Cloud Computing, Data Science and Engineering*, Jan. 2017, doi: https://doi.org/10.1109/confluence.2017.7943220.

[92] K. Delac, M. Grgic, and S. Grgic, “Face recognition in JPEG and JPEG2000 compressed domain,” *Image and Vision Computing*, vol. 27, no. 8, pp. 1108–1120, Jul. 2009, doi: https://doi.org/10.1016/j.imavis.2008.10.007.

[93] A. Deshpande, S. Dubey, H. Shaligram, A. Potnis, and S. Chavan, “Iris recognition system using block based approach with DWT and DCT,” *Annual IEEE India Conference (INDICON)*, Dec. 2014, doi: https://doi.org/10.1109/indicon.2014.7030396.

[94] K. Tewari and R. L. Kalakoti, “Fingerprint recognition and feature extraction using transform domain techniques,” *2014 International Conference on Advances in Communication and Computing Technologies (ICACACT 2014),* Aug. 2014, doi: https://doi.org/10.1109/eic.2015.7230719.

[95] M. J. Er, W. Chen, and S. Wu, “High-speed face recognition based on discrete cosine transform and RBF neural networks,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 679–691, May 2005, doi: https://doi.org/10.1109/TNN.2005.844909.

[96] L. Leng, J. Zhang, M. K. Khan, J. Xu, and K. Alghathbar, “Dynamic weighted discrimination power analysis in DCT domain for face and palmprint recognition,”

International Conference on Information and Communication Technology Convergence (ICTC), Nov. 2010, doi: https://doi.org/10.1109/ictc.2010.5674791.

[97] S. Dabbaghchian, A. Aghagolzadeh, and M. S. Moin, "Feature extraction using discrete cosine transform for face recognition," *2007 9th International Symposium on Signal Processing and its Applications*, Feb. 2007, doi: https://doi.org/10.1109/isspa.2007.4555358.

[98] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTS," *arXiv.org*, Mar. 06, 2014. Accessed: Dec. 01, 2019. Available: https://arxiv.org/abs/1312.5851.

[99] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, "FCNN: Fourier Convolutional Neural Networks," *Lecture notes in computer science*, pp. 786–798, Jan. 2017, doi: https://doi.org/10.1007/978-3-319-71249-9_47.

[100] L. Gueguen, A. Sergeev, R. Liu, and J. Yosinski, "Faster Neural Networks Straight from JPEG," *Neural Information Processing Systems (NIPS)*, Dec. 2018. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/7af6266cc52234b5aa339b16695f7fc4-Abstract.html.

[101] A. Ghosh and R. Chellappa, "Deep feature extraction in the DCT domain," *International Conference on Pattern Recognition*, Dec. 2016, doi: https://doi.org/10.1109/icpr.2016.7900182.

[102] Y. Xu and H. Nakayama, "DCT Based Information-Preserving Pooling for Deep Neural Networks," *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, doi: https://doi.org/10.1109/icip.2019.8802962.

[103] M. Ulicny, V. A. Krylov, and Rozenn Dahyot, "Harmonic Networks with Limited Training Samples," *MURAL - Maynooth University Research Archive Library (National University of Ireland, Maynooth)*, Sep. 2019, doi: https://doi.org/10.23919/eusipco.2019.8902831.

[104] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, Y. Shuicheng, and J. Feng, "Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks With Octave Convolution," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, doi: https://doi.org/10.1109/iccv.2019.00353.

[105] A. Ciurana, A. Mosella-Montoro, and J. Ruiz-Hidalgo, "Hybrid Cosine Based Convolutional Neural Networks," *arXiv.org*, Apr. 03, 2019. Accessed: Jan. 13, 2024. Available: https://arxiv.org/abs/1904.01987.

[106] Y. Xu and H. Nakayama, “Shifted spatial-spectral convolution for deep neural networks,” *1st ACM International Conference on Multimedia in Asia*, Dec. 2019, doi: https://doi.org/10.1145/3338533.3366575.

[107] H. Pan, X. Zhu, S. Atici, and A. E. Cetin, “DCT Perceptron Layer: A Transform Domain Approach for Convolution Layer,” *arXiv.org*, Nov. 15, 2022. Accessed: Jan. 13, 2024. Available: http://arxiv.org/abs/2211.08577.

[108] X. Li, J. Wu, Z. Sun, Z. Ma, J. Cao, and J.-H. Xue, “BSNet: Bi-Similarity Network for Few-shot Fine-grained Image Classification,” *IEEE transactions on image processing*, vol. 30, pp. 1318–1331, Jan. 2021, doi: https://doi.org/10.1109/tip.2020.3043128.

[109] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local Neural Networks,” *arXiv (Cornell University)*, Nov. 2017, doi: https://doi.org/10.48550/arxiv.1711.07971.

[110] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent Models of Visual Attention,” *Neural Information Processing Systems*, 2014. Available: https://proceedings.neurips.cc/paper/2014/hash/09c6c3783b4a70054da74f2538ed47c6-Abstract.html.

[111] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019, doi: https://doi.org/10.1109/tpami.2019.2913372.

[112] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “CBAM: Convolutional Block Attention Module,” *Computer Vision – ECCV 2018*, pp. 3–19, 2018, doi: https://doi.org/10.1007/978-3-030-01234-2_1.

[113] B. Wang, S. Zhang, J. Zhang, and Z. Cai, “Architectural style classification based on CNN and channel–spatial attention,” *Signal, Image and Video Processing*, Apr. 2022, doi: https://doi.org/10.1007/s11760-022-02208-0.

[114] Y. Cheng, X. Wang, X. Xie, W. Li, and S. Peng, “Channel pruning guided by global channel relation,” *Applied intelligence*, vol. 52, no. 14, pp. 16202–16213, Mar. 2022, doi: https://doi.org/10.1007/s10489-022-03198-9.

[115] J. Kong, H. Wang, C. Yang, X. Jin, M. Zuo, and X. Zhang, “A Spatial Feature-Enhanced Attention Neural Network with High-Order Pooling Representation for Application in Pest and Disease Recognition,” *Agricultural*, vol. 12, no. 4, pp. 500–500, Mar. 2022, doi: https://doi.org/10.3390/agriculture12040500.

[116] S. Wang, Y. Zhu, S. Lee, D. C. Elton, T. C. Shen, Y. Tang, Y. Peng, Z. Lu, and R. M. Summers, “Global-Local attention network with multi-task uncertainty loss for

abnormal lymph node detection in MR images," *Medical image analysis*, vol. 77, pp. 102345–102345, Apr. 2022, doi: https://doi.org/10.1016/j.media.2021.102345.

[117] C. Scribano, G. Franchini, M. Prato, and M. Bertogna, "DCT-Former: Efficient Self-Attention with Discrete Cosine Transform," *Journal of Scientific Computing*, vol. 94, no. 3, Feb. 2023, doi: https://doi.org/10.1007/s10915-023-02125-5.

[118] Y. Wang, Y. Qi, C. Xu, M. Lou, and Y. Ma, "Learning multi-frequency features in convolutional network for mammography classification," *Medical & biological engineering & computing*, vol. 60, no. 7, pp. 2051–2062, May 2022, doi: https://doi.org/10.1007/s11517-022-02582-4.

[119] Y. Huang, C. Zhou, L. Chen, J. Chen, and S. Lan, "Medical Frequency Domain Learning: Consider Inter-class and Intra-class Frequency for Medical Image Segmentation and Classification," *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2021, doi: https://doi.org/10.1109/bibm52615.2021.9669443.

[120] M. H. Guo, T. X. Xu, J. J. Liu, Z. N. Liu, P. T. Jiang, T. J. Mu, S. H. Zhang, R. R. Martin, M. M. Cheng, and S. M. Hu, "Attention mechanisms in computer vision: A survey," *Computational Visual Media*, Mar. 2022, doi: https://doi.org/10.1007/s41095-022-0271-y.

[121] S. Pande and B. Banerjee, "Adaptive hybrid attention network for hyperspectral image classification," *Pattern Recognition Letters*, vol. 144, pp. 6–12, Apr. 2021, doi: https://doi.org/10.1016/j.patrec.2021.01.015.

[122] N. Guo, K. Gu, J. Qiao, and J. Bi, "Improved deep CNNs based on Nonlinear Hybrid Attention Module for image classification," *Neural Networks*, vol. 140, pp. 158–166, Aug. 2021, doi: https://doi.org/10.1016/j.neunet.2021.01.005.

[123] Y. Fan, J. Liu, R. Yao, and X. Yuan, "COVID-19 Detection from X-ray Images using Multi-Kernel-Size Spatial-Channel Attention Network," *Pattern Recognition*, vol. 119, p. 108055, Nov. 2021, doi: https://doi.org/10.1016/j.patcog.2021.108055.

[124] Z. Qin, P. Zhang, F. Wu, and X. Li, "FcaNet: Frequency Channel Attention Networks," *arXiv.org*, Dec. 2020. Accessed: Jan. 13, 2024. Available: https://doi.org/10.48550/arxiv.2012.11879.

[125] J. Gao, Q. Wang, and Y. Yuan, "SCAR: Spatial-/channel-wise attention regression networks for crowd counting," *Neurocomputing*, vol. 363, pp. 1–8, Oct. 2019, doi: https://doi.org/10.1016/j.neucom.2019.08.018.

[126] X. Wang, J. Shi, H. Fujita, and Y. Zhao, “Aggregate attention module for fine-grained image classification,” *Journal of ambient intelligence & humanized computing/Journal of ambient intelligence and humanized computing*, vol. 14, no. 7, pp. 8335–8345, Nov. 2021, doi: https://doi.org/10.1007/s12652-021-03599-7.

[127] X. Ke and Y. Zhang, “Fine-grained Vehicle Type Detection and Recognition Based on Dense Attention Network,” *Neurocomputing*, vol. 399, pp. 247–257, Mar. 2020, doi: https://doi.org/10.1016/j.neucom.2020.02.101.

[128] Z. Xiangyu, “Wavelet-Attention CNN for Image Classification,” *arXiv.org*, Jan. 23, 2022. Accessed: Jan. 13, 2024. Available: http://arxiv.org/abs/2201.09271.

[129] H. Chen, L. Cheng, G. Huang, G. Zhang, J. Lan, Z. Wu, C. M. Pun, and W. K. Ling, “Fine-grained visual classification with multi-scale features based on self-supervised attention filtering mechanism,” *Applied Intelligence*, vol. 52, no. 13, pp. 15673–15689, Mar. 2022, doi: https://doi.org/10.1007/s10489-022-03232-w.

[130] X. Liu, L. Zhang, T. Li, D. Wang, and Z. Wang, “Dual attention guided multi-scale CNN for fine-grained image classification,” *Information Sciences*, vol. 573, pp. 37–45, Sep. 2021, doi: https://doi.org/10.1016/j.ins.2021.05.040.

[131] X. Ke, Y. Huang, and W. Guo, “Weakly supervised fine-grained image classification via two-level attention activation model,” *Computer Vision and Image Understanding*, Mar. 2022, doi: https://doi.org/10.1016/j.cviu.2022.103408.

[132] F. Liu, H. Xu, M. Qi, D. Liu, J. Wang, and J. Kong, “Depth-Wise Separable Convolution Attention Module for Garbage Image Classification,” *Sustainability*, vol. 14, no. 5, pp. 3099–3099, Mar. 2022, doi: https://doi.org/10.3390/su14053099.

[133] P. Singh, P. Mazumder, and V. P. Namboodiri, “Context extraction module for deep convolutional neural networks,” *Pattern Recognition*, vol. 122, p. 108284, Feb. 2022, doi: https://doi.org/10.1016/j.patcog.2021.108284.

[134] N. Wang, Y. Bai, K. Yu, Y. Jiang, S. Xia, and Y. Wang, “Adaptive Frequency Learning in Two-branch Face Forgery Detection,” *arXiv.org*, Mar. 27, 2022. Accessed: Jan. 13, 2024. Available: http://arxiv.org/abs/2203.14315.

[135] V. K. Sharma, K. K. Mahapatra, and B. Acharya, “Visual object tracking based on discriminant DCT features,” *Digital Signal Processing*, vol. 95, p. 102572, Dec. 2019, doi: https://doi.org/10.1016/j.dsp.2019.08.002.

[136] Y. Yan, S. Duffner, P. Phutane, A. Berthelier, X. Naturel, C. Blanc, C. Garcia, and T. Chateau, “Fine-grained facial landmark detection exploiting intermediate feature

representations," *Computer Vision and Image Understanding*, vol. 200, p. 103036, Nov. 2020, doi: https://doi.org/10.1016/j.cviu.2020.103036.

[137] L. Wang, L. Xu, W. Tian, Y. Zhang, H. Feng, and Z. Chen, "Underwater image super-resolution and enhancement via progressive frequency-interleaved network," *Journal of Visual Communication and Image Representation*, vol. 86, p. 103545, Jul. 2022, doi: https://doi.org/10.1016/j.jvcir.2022.103545.

[138] C. Pu, H. Huang, and L. Yang, "An attention-driven convolutional neural network-based multi-level spectral–spatial feature learning for hyperspectral image classification," *Expert Systems with Applications*, vol. 185, p. 115663, Dec. 2021, doi: https://doi.org/10.1016/j.eswa.2021.115663.

[139] M. Peker, "Classification of hyperspectral imagery using a fully complex-valued wavelet neural network with deep convolutional features," *Expert Systems with Applications*, vol. 173, p. 114708, Jul. 2021, doi: https://doi.org/10.1016/j.eswa.2021.114708.

[140] E. Pan, X. Mei, Q. Wang, Y. Ma, and J. Ma, "Spectral-spatial classification for hyperspectral image based on a single GRU," *Neurocomputing*, vol. 387, pp. 150–160, Apr. 2020, doi: https://doi.org/10.1016/j.neucom.2020.01.029.

[141] S. O. Ayat, M. Khalil-Hani, A. A.-H. Ab Rahman, and H. Abdellatef, "Spectral-based convolutional neural network without multiple spatial-frequency domain switchings," *Neurocomputing*, vol. 364, pp. 152–167, Oct. 2019, doi: https://doi.org/10.1016/j.neucom.2019.06.094.

[142] Z. Zhou, X. Wang, C. Li, M. Zeng, and Z. Li, "Adaptive deep feature aggregation using Fourier transform and low-pass filtering for robust object retrieval," *Journal of Visual Communication and Image Representation*, vol. 72, p. 102860, Oct. 2020, doi: https://doi.org/10.1016/j.jvcir.2020.102860.

[143] Q. Li, L. Shen, S. Guo, and Z. Lai, "WaveCNet: Wavelet Integrated CNNs to Suppress Aliasing Effect for Noise-Robust Image Classification," *IEEE Transactions on Image Processing*, vol. 30, pp. 7074–7089, 2021, doi: https://doi.org/10.1109/tip.2021.3101395.

[144] Y. Wang and Z. Wang, "A survey of recent work on fine-grained image classification techniques," *Journal of Visual Communication and Image Representation*, vol. 59, pp. 210–214, Feb. 2019, doi: https://doi.org/10.1016/j.jvcir.2018.12.049.

[145] J. Chen, J. Hu, and S. Li, “Learning to locate for fine-grained image recognition,” *Computer Vision and Image Understanding*, vol. 206, p. 103184, May 2021, doi: https://doi.org/10.1016/j.cviu.2021.103184.

[146] A. E. Eshratifar, D. Eigen, M. Gormish, and M. Pedram, “Coarse2Fine: a two-stage training method for fine-grained visual classification,” *Machine Vision and Applications*, vol. 32, no. 2, Feb. 2021, doi: https://doi.org/10.1007/s00138-021-01180-y.

[147] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, and J. Luo, “Revisiting local descriptor based image-to-class measure for few-shot learning,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, doi: https://doi.org/10.1109/cvpr.2019.00743.

[148] W. Y. Chen, Y. C. F. Wang, Y. C. Liu, Z. Kira, and J. bin Huang, “A closer look at few-shot classification,” *arXiv.org*, Jan. 2022. Accessed: Jan. 13, 2024. Available: https://doi.org/10.48550/arxiv.1904.04232.

[149] L. Qiao, Y. Shi, J. Li, Y. Tian, T. Huang, and Y. Wang, “Transductive episodic-wise adaptive metric for few-shot learning,” *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2019, doi: https://doi.org/10.1109/iccv.2019.00370.

[150] F. Hao, F. He, J. Cheng, L. Wang, J. Cao, and D. Tao, “Collect and select: Semantic alignment metric learning for few-shot learning,” *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2019, doi: https://doi.org/10.1109/iccv.2019.00855.

[151] M. Ulicny, V. A. Krylov, and R. Dahyot, “Harmonic Networks for Image Classification,” *British Machine Vision Conference (BMVC)*, 2019. Accessed: Jan. 13, 2024. Available: https://mural.maynoothuniversity.ie/15155.

[152] G. Yang, Y. He, Y. Yang, and B. Xu, “Fine-Grained Image Classification for Crop Disease Based on Attention Mechanism,” *Frontiers in Plant Science*, vol. 11, Dec. 2020, doi: https://doi.org/10.3389/fpls.2020.600854.

[153] Y. Zhu, R. Ding, W. Huang, P. Wei, G. Yang, and Y. Wang, “HMFCA-Net: Hierarchical multi-frequency based Channel attention net for mobile phone surface defect detection,” *Pattern Recognition Letters*, vol. 153, pp. 118–125, Jan. 2022, doi: https://doi.org/10.1016/j.patrec.2021.11.029.

[154] S. Li, C. Ge, X. Sui, Y. Zheng, and W. Jia, “Channel and Spatial Attention Regression Network for Cup-to-Disc Ratio Estimation,” *Electronics*, vol. 9, no. 6, p. 909, May 2020, doi: https://doi.org/10.3390/electronics9060909.

[155] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, “ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, doi: https://doi.org/10.1109/cvpr42600.2020.01155.

[156] H. Shiratori, H. Goto, and H. Kobayashi, “An efficient text capture method for moving robots using DCT feature and text tracking,” *International Conference on Pattern Recognition*, Jan. 2006, doi: https://doi.org/10.1109/icpr.2006.243.

[157] V. Maheshkar, S. Kamble, S. Agarwal, and V. Kumar, “Feature Image Generation Using Low, Mid and High Frequency Regions for Face Recognition,” *International journal of multimedia and its applications*, vol. 4, no. 1, pp. 75–82, Feb. 2012, doi: https://doi.org/10.5121/ijma.2012.4107.

[158] S. Dabbaghchian, A. Aghagolzadeh, and M. S. Moin, “Reducing the effects of small sample size in DCT domain for face recognition,” *2008 International Symposium on Telecommunications*, Aug. 2008, doi: https://doi.org/10.1109/istel.2008.4651378.

[159] The University of Montreal, “Covid-19 Image Dataset,” Kaggle. Accessed: Aug. 21, 2019. Available: https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset.

[160] D. Agrawal, S. Minocha, S. Namasudra, and S. Kumar, “Ensemble Algorithm using Transfer Learning for Sheep Breed Classification,” *15th International Symposium on Applied Computational Intelligence and Informatics*, May 01, 2021, pp. 199–204. doi: https://doi.org/10.1109/SACI51354.2021.9465609.

[161] S. Abu Jwade, A. Guzzomi, and A. Mian, “On farm automatic sheep breed classification using deep learning,” *Computers and Electronics in Agriculture*, vol. 167, p. 105055, Dec. 2019, doi: https://doi.org/10.1016/j.compag.2019.105055.

[162] K. C. Tung, “Flower Images jpg,” *Mendeley Data*, vol. 1, Oct. 2020. Accessed: Dec. 01, 2019. doi: https://doi.org/10.17632/738sdjm6h9.1.

[163] J. Wang, K. Markert, and M. Everingham, “Learning Models for Object Recognition from Natural Language Descriptions,” Jan. 2009, doi: https://doi.org/10.5244/c.23.2.

[164] G. Montoya, J. Zhang, and S. Loaiciga, “10 Monkey Species,” Kaggle. Accessed: Jan. 01, 2020. Available: https://www.kaggle.com/datasets/slothkong/10-monkey-species.
[165] Gerry, “YIKES! Spiders -15 Species Classification,” Kaggle. Accessed: Jan. 01, 2020. Available: https://www.kaggle.com/datasets/gpiosenka/yikes-spiders-15-species.
[166] D. Dutta, “Identifying different Breeds of Snakes”, Kaggle. Accessed: Jan. 01, 2020. Available: https://www.kaggle.com/datasets/duttadebadri/identifying-different-breeds-of-snakes.
[167] E. Duck, “Butterfly Classification Dataset”, Kaggle. Accessed: Jan. 01, 2020. Available: https://www.kaggle.com/datasets/pnkjgpt/butterfly-classification-dataset.
[168] “Flower Classification with TPUs,” Kaggle. Accessed: Jan. 01, 2020. Available: https://kaggle.com/competitions/flower-classification-with-tpus.
[169] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” 2009. Accessed: Jan. 01, 2020. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[170] “ImageNet 1000 (mini),” Kaggle. Accessed: Jan. 01, 2020. Available: https://www.kaggle.com/datasets/ifigotin/imagenetmini-1000.

Appendix

This section demonstrates the derivation of the computational complexity of the original ECA and the Hybrid Mod-ECA. The notations and variables used in this appendix are independent of the rest of this chapter. Let the depth of the 1D convolution filter kernel be k, the 1D input feature channel depth dimension be d, the convolution stride size be s, and the padding size be p. The instantaneous computational complexity (O_{inst}) of the 1D convolution process is obtained by considering the multiplication and summation of a single convolution process.

$$O_{inst} = number\ of\ multiplication + number\ of\ summation$$

$$O_{inst} = (output\ feature\ map\ size) \times (kernel\ size) + (output\ feature\ map\ size) \times (kernel\ size - 1)$$

$$\therefore O_{inst} = \left[\frac{(d + 2p) - k}{s}\right] \times k + \left[\frac{(d + 2p) - k}{s}\right] \times (k - 1)$$

In the original ECA, $k = j, d = C, p = s = 1$. The computational complexity yields:

$$O_{ECA} = \left[\frac{(C + 2(1)) - j}{1} + 1\right] \times [j \cdot (j - 1)] = (j) \cdot (j - 1) \cdot (C - j + 3)$$

In the HyMod-ECA, $k = j, d = C, p = 0, s = j$. The computational complexity yields:

$$O_{HyMod-ECA} = \left[\frac{(C + 2(0)) - j}{j} + 1\right] \times [j \cdot (j - 1)] = \left[\frac{C - j + j}{j}\right] \cdot j \cdot (j - 1) = C \cdot (j - 1)$$

Table I: Performance comparison in terms of classification error between various M-Skipped architectures and Ablation Study.

Abbreviations	Parameters (mil.)	Sheep Breed	Flowers	Leeds Butterfly	Monkey	Spider Breed
All-DCTC	1.7	0.0972±0.0124	0.2615±0.0077	0.1979±0.0072	0.2721±0.0132	0.2244±0.0301
L-DCTC	1.7	0.0972±0.0091	0.2615±0.0077	0.2021±0.0158	0.3113±0.0118	0.2400±0.0333
M-DCTC	1.7	0.4028±0.0281	0.4513±0.0270	0.5833±0.0036	0.6434±0.0241	0.6867±0.0982
H-DCTC	1.7	0.4643±0.0412	0.5615±0.0133	0.6812±0.0165	0.6752±0.0042	0.7111±0.0329
M-Skipped-1	1.7	0.0873±0.0171	0.2589±0.0044	0.1667±0.0072	0.2574±0.0224	0.2156±0.0077
M-Skipped-2	1.7	**0.0695±0.0182**	**0.2436±0.0160**	0.1625±0.0165	0.2709±0.0149	0.2666±0.0577
M-Skipped-3	1.7	0.0853±0.0124	0.2538±0.0154	**0.1229±0.0095**	0.2610±0.0373	0.2245±0.0168
M-Skipped-123-extended	1.9	0.0873±0.0150	0.3051±0.0347	0.1792±0.0144	0.2402±0.0118	0.2889±0.0102
M-Skipped-123-extended-deep	1.9	0.0893±0.0060	0.2692±0.0000	0.1479±0.0095	**0.2230±0.0225**	**0.1845±0.0204**
M-Skipped-3-FC	8.9	0.0714±0.0103	0.2769±0.0539	0.1416±0.0130	0.2512±0.0118	0.2978±0.0907
M-Skipped-3-ReLU	1.7	0.1071±0.0206	0.2666±0.0160	0.1771±0.0308	0.2904±0.0434	0.2844±0.0844

Table II: Best performing classification error between all adaptive DCT-BF kernel variations.

Abbreviations	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0864	**0.0624±0.0382**	0.2607±0.0199	0.2157±0.0174	0.2222±0.0253	0.3772±0.0021	**0.1587±0.0095**
MS3-AD-0816	**0.0657±0.0231**	0.2590±0.0117	**0.2022±0.0337**	0.2267±0.0240	0.3736±0.0050	**0.1593±0.0103**
MS3-AD-0804	0.0808±0.0315	0.2513±0.0193	0.2145±0.0106	0.2133±0.0231	0.3728±0.0009	**0.1560±0.0106**
MS3-AD-0801	0.0791±0.0239	0.2641±0.0247	0.3370±0.1377	**0.1978±0.0139**	0.3788±0.0015	0.1760±0.0399
MS3-AD-0416	0.1364±0.0660	0.2590±0.0270	0.2194±0.0202	0.2155±0.0329	**0.3675±0.0057**	0.1727±0.0117
MS3-AD-0404	0.0758±0.0152	**0.2408±0.0234**	**0.2108±0.0202**	0.2134±0.0115	**0.3724±0.0019**	0.1653±0.0179
MS3-AD-0401	0.0909±0.0455	0.2564±0.0088	0.2819±0.0721	0.2911±0.0391	0.3750±0.0032	0.1607±0.0050
MS3-AD-0204	0.2121±0.0263	0.2564±0.0117	0.2267±0.0165	**0.1778±0.0234**	0.3788±0.0071	0.1707±0.0121
MS3-AD-0801-opt	0.2727±0.0152	0.3487±0.0270	0.4669±0.0369	0.3800±0.0067	0.8343±0.0055	0.4187±0.0064
MS3-AD-0804-opt	0.1768±0.0532	0.2641±0.0270	0.3689±0.0454	0.2955±0.0234	0.71980.0146	0.2807±0.0101
MS3-AD-0816-opt	0.1010±0.0315	**0.2410±0.0193**	0.2623±0.0140	0.2289±0.0269	0.4147±0.0081	0.2073±0.0181
MS3-AD-0401-opt	0.1566±0.0437	0.2872±0.0347	0.3456±0.0459	0.3356±0.0509	0.7129±0.0054	0.2860±0.0140
MS3-AD-0404-opt	0.1717±0.0438	0.2615±0.0308	0.2487±0.0106	0.2400±0.0176	0.4426±0.0024	0.2107±0.0031
(MS3-AD-842-org)	**0.0657±0.0231**	0.2564±0.0044	**0.2083±0.0056**	**0.1956±0.0214**	**0.3710±0.0048**	0.1627±0.0050
(VGG16-PC*)	0.1717±0.0232	0.2718±0.0235	0.3419±0.0702	0.2489±0.0154	0.4123±0.0027	0.2253±0.0110
(MS3-base*)	0.2071±0.0631	**0.2448±0.0155**	0.2194±0.0215	0.2489±0.0539	0.37800.0066	0.1807±0.0046
Top-1 lowest	MS3-AD-0864	MS3-AD-0404	MS3-AD-0816	MS3-AD-0204	MS3-AD-0416	MS3-AD-0804
Top-2 lowest	MS3-AD-0816 (MS3-AD-842-org)	MS3-AD-0816-opt	(MS3-AD-842-org)	(MS3-AD-842-org)	(MS3-AD-842-org)	MS3-AD-0864
Top-3 lowest	-	(MS3-base*)	MS3-AD-0404	MS3-AD-0801	MS3-AD-0404	MS3-AD-0816

Table III: Best performing speed of convergence (reaching over 75% classification accuracy, measured in epochs) between all adaptive DCT-BF kernel variations.

Abbreviations	Covid19	Flowers	Monkey	Spider Breed	Snake Breed	Butterfly
MS3-AD-0864	8.33±1.15	21.00±7.21	15.00±1.00	18.00±7.21	0.00±0.00	6.33±1.15
MS3-AD-0816	7.00±1.73	20.00±3.46	21.67±9.24	17.00±0.00	0.00±0.00	6.00±0.00
MS3-AD-0804	8.00±1.00	23.00±6.08	**11.67±2.08**	17.00±3.00	0.00±0.00	**5.33±0.58**
MS3-AD-0801	7.67±1.53	25.00±8.00	14.00±1.73	16.00±4.58	0.00±0.00	5.67±0.58
MS3-AD-0416	8.00±1.00	18.67±7.09	13.33±1.12.00	**11.00±3.00**	0.00±0.00	6.67±0.58
MS3-AD-0404	7.00±1.00	16.00±4.00	13.00±2.00	16.33±2.89	0.00±0.00	6.67±1.15
MS3-AD-0401	6.00±2.00	20.33±8.39	12.00±2.65	15.33±2.08	0.00±0.00	6.33±1.15
MS3-AD-0204	10.00±1.73	12.33±14.98	15.33±2.31	16.33±1.53	13±0.00	7.33±0.58
MS3-AD-0801-opt	15.33±11.85	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
MS3-AD-0804-opt	**4.33±1.53**	18.33±15.89	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
MS3-AD-0816-opt	7.33±1.53	26.00±3.46	20.33±0.58	20.33±2.31	0.00±0.00	9.00±0.00
MS3-AD-0401-opt	9.67±3.79	19.33±503	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
MS3-AD-0404-opt	8.67±2.89	**3.00±5.20**	18.67±4.16	19.67±1.53	0.00±0.00	12.00±1.73
MS3-AD-842-org	7.67±0.58	20.67±4.16	13.67±2.31	17.00±1.73	0.00±0.00	**5.33±0.58**
VGG16-PC*	8.67±2.08	25.00±1.73	1.33±2.31	22.00±1.73	0.00±0.00	11.33±1.15
MS3-base*	9.00±3.61	16.33±3.06	15.00±2.00	16.67±2.08	0.00±0.00	8.00±1.00

Table IV: Average classification error, average convergence speed, and number of parameters between several adaptive DCT-BF kernel variants and the former M-Skipped network across FGVC datasets.

Abbreviations	Classification Error	Convergence Speed (Epochs)	Number of Parameters (mil)
MS3-AD-0864	0.2162	13.73	102.53
MS3-AD-0816	0.2144	14.33	102.53
MS3-AD-0804	0.2148	13.00	102.53
MS3-AD-0801	0.2388	13.67	102.53
MS3-AD-0416	0.2284	11.53	25.85
MS3-AD-0404	0.2131	11.80	25.85
MS3-AD-0401	0.2427	12.00	25.85
MS3-base*	0.2465	13.00	1.70

Table V: Classification error on the testing dataset for all attention variations on Adapt-DCT CNN.

Abbreviations	Flowers	Monkey	Snake Breed	Snake Breed	Butterfly
MS3-0404 ECA-ORG	0.2231±0.0335	0.2169±0.0184	0.2133±0.0346	**0.3673±0.0075**	0.1620±0.0035
MS3-0404 ECA-AD	**0.2205±0.0222**	**0.2059±0.0097**	**0.1778±0.0168**	0.3734±0.0079	**0.1580±0.0125**
MS3-0404 ECA-AD-1C1A	0.2436±0.0494	**0.2059±0.0097**	0.2155±0.0329	0.3802±0.0015	0.1833±0.0129
MS3-0404 ECA-ADFC	0.2410±0.0311	0.2096±0.0074	0.3133±0.0933	0.3855±0.0084	0.1827±0.0300