New Convolutional Neural Network Topology with Compressed Information to Enhance Accuracy for Image Classification Task

by

Yanbing Jiang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

© Yanbing Jiang 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Source coding and deep learning are two major branches in the field of information processing. Source coding encodes information that can be summarised with patterns into certain representation without semantic consideration. On the other hand, deep learning utilises multi-layers of representations with increasing levels of abstraction to learn the patterns that cannot be summarised easily. What is interesting is that source coding itself makes great contributions to the field of deep learning. The key that makes deep learning successful is the inclusion of cascading non-linear layers that help the network to abstract multi-level features. Source coding, such as image compression, contains fundamental non-linear operations including quantisation and rounding. How the non-linearity from the compression could further help deep learning is the inspiration of this research even though common sense tells us that compression usually results a worse ability to do recognition. This paper proposes the idea of integrating source coding and deep learning to have better accuracy performance in image classification.

Image classification is one of the most popular tasks in the field of deep learning. Based on human visions perception to classify object(s) in images, when the images are compressed, such as by JPEG, the humans recognition ability deteriorates. Nonetheless, it is not usually the case in machine's perspective. Compressed images may be recognised better by machine based on our observation. In order to improve the accuracy of image recognition, this study focuses on improving the pre-processing operation before image input into the neural network. At the meantime, we proposed a new Convolutional Neural Network (CNN) topology, which absorbs original input along with its various compressed versions. JPEG image compression is friendly for human when the images are compressed with higher quality. However, what level of the compressed image is machine friendly is uncertain. This topology facilitates the compressed information across the compression inputs from low to high qualities and lets the machine to learn from all potential compressed information by itself. We trained the topology with proposed Block-by-block training method and were able to increase the accuracy of state-of-art CNN for image classification: 0.374% increase in Top-1 accuracy, 0.346% increase in Top-5 accuracy in terms of Inception V3 model and 0.39% increase in Top-1 accuracy and 0.228% increase in Top-5 accuracy in terms of ResNet-50 V2 model.

What's more, we can state that compression can highlight the contrast of the objects and discard interference information which helps our topology improve the accuracy of image classification based on visual observations. Furthermore, we believe the accuracy performance could be even more outstanding if our topology is applied to the state-of-art EfficientNet (published May 2019).

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Prof. En-hui Yang. His insightful guidance, patience, motivation and all precious support helped me establish the direction of my research and led my moving forward with my studies in depth. It is he who always encourages me to think deep and precisely and direct me to this great direction. I have learned a lot from him and knowledge in the field.

I would like to thank Professors who taught me courses and gave me precious knowledge: Oleg Michailovich, Zhou Wang, Ravi Mazumdar, Mohamed Oussama Damen, Shai Ben-David, Gordon Agnew, and Jochen Koenemann. I gained a lot from their excellent lectures, which built me fundamental knowledge for my research works.

I would like to thank Prof. Oleg Michailovich and Prof. Zhou Wang for being the readers of this thesis and giving me valuable comments and suggestions.

I would also like to thank my colleague in the multimedia communication lab, Hossam Amer, for his continuous support. I have gained a lot of new knowledge, thoughts, and useful suggestions from him through discussions during my research process.

Last but not least, I would like to thank my parents, Mr. Lixin Jiang and Mrs. Ying Xu, and my girlfriend, Yujia Guo, for their unlimited support, understanding, and encouragement.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence 8, 9
CNN	Convolutional Neural Network xi, xii, 2–6, 11–17, 19–22, 24–26, 28–30, 33, 34, 36, 37, 42–46, 57, 58, 60
DCT	Discrete Cosine Transform 6, 7
FN	False Negative 29, 30
FP	False Positive 29, 30
ILSVRC	ImageNet Large Scale Visual Recognition Competition viii, ix, xi, xii, 4, 10, 11, 21, 23, 27, 28, 30, 37, 41, 42, 45, 49, 51–53, 55, 59
JPEG	Joint Photographic Experts Group 1, 3, 4, 6–8, 24, 26–28, 52, 54, 60
QF	Quality Factor xi, 2, 3, 7, 8, 27, 28, 48–52
ReLu	Rectified Linear Units 3, 13, 18
ResNet	Residual Network 15, 21–23, 37, 43, 57
RLE	Run Length Encoder 8
SGD	Stochastic Gradient Descent 32
TCM	Transparent Composite Model xii, 58, 59
TN	True Negative 29, 30
TP	True Positive 29, 30

Chapter 1

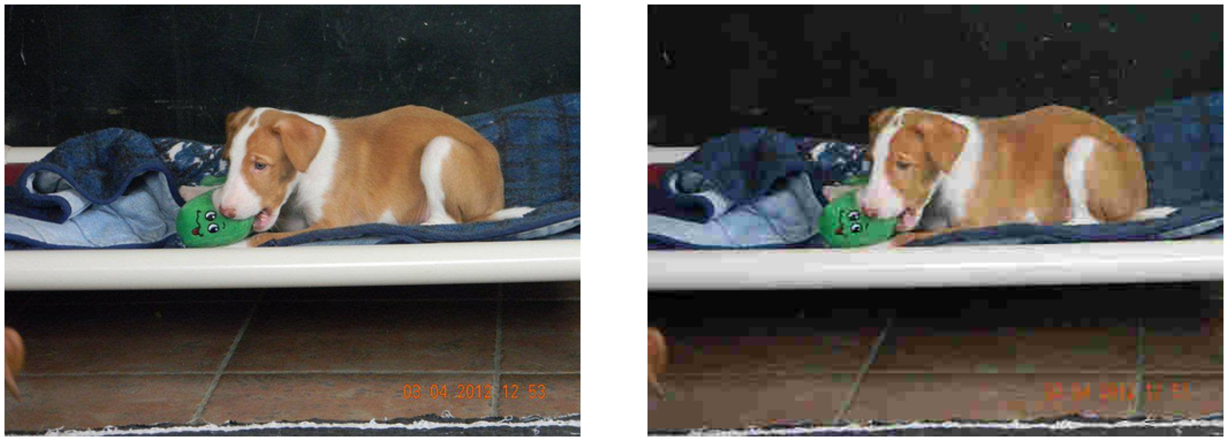
Introduction

1.1 Research Motivations and Question Description

Source coding and deep learning are two major branches in the field of information processing. Source coding encodes information that can be summarised with patterns into certain representation without semantic consideration. On the other hand, deep learning utilises multi-layers of representations with increasing levels of abstraction to learn the patterns that cannot be summarised easily. What is interesting is that source coding itself makes great contributions to the field of deep learning. Therefore, deep learning can be understood as a learning form of source coding. Source coding provides "raw data" input to deep learning. Therefore, whether source coding method, such as compression, can help deep learning to perform better is worth being studied.

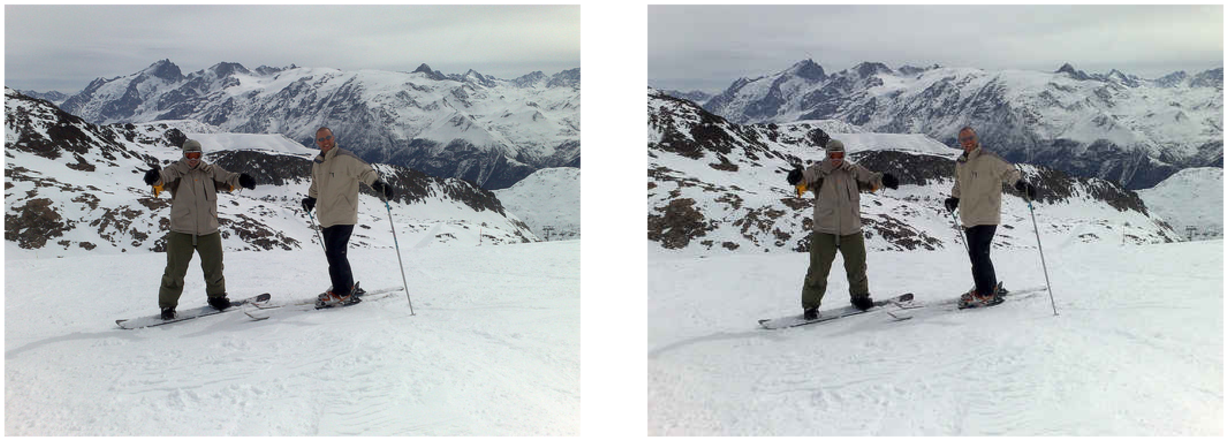
In deep learning, image classification is one of the most challenging tasks. Current research works are developing quickly and have achieved a high level. This task more or less dependent on the input images. One of the most famous image source coding formats or compression methods is Joint Photographic Experts Group (JPEG). For the most famous dataset of image classification, ImageNet, all data inside is in the format of JPEG. However, this well-know lossy image compression format is designed and proposed for human's vision. When an image is compressed more in JPEG format, human's ability to recognisw the object(s) in this image will drop dramatically.

However, what compression level could get the best accuracy performance is uncertain based on the observation by H. Amer et al. [1]. Consider the following pair of images in Figure 1.1, which are an original image with a dog (ground truth label: Ibizan hound, Ibizan



(a) (b)

Figure 1.1: (a) original image #115 with object Ibizan hound from ImageNet ILSVRC-2012 validation set; (b) Compressed version of (a) by JPEG with Quality Factor = 10.



(a) (b)

Figure 1.2: (a) original image #2 with object alp from ImageNet ILSVRC-2012 validation set; (b) Compressed version of (a) by JPEG with Quality Factor = 40.

Podenco) and its JPEG compressed version with QF (concept of QF will be introduced in Section 2.1) equals to 10, which is compressed heavily:

It is obvious from human eyes that the original image is easier to identify. However, the machine (based on Google's Inception V3 image classification CNN) has only 9.9% confidence on the ground truth while 70.6% confidence on the QF = 10 compressed version.

Thus, the machine, in this case, has more confidence in the compressed version. Until here, it can be found out based on the observation that source coding that takes human's vision into account is not actually the best source coding for machines. However, for the image in Figure 1.2, with QF equals to 40, which is compressed moderately, make the machine has the highest confidence level instead. As we could see from these examples, the best JPEG compression quality factor that is suitable for machine is uncertain. This further confirms the observation in [1]. The way how to use compressed information to improve the machine's prediction becomes interesting to discover.

In the field of machine learning, the concept of non-linearity is important for the following reason. Consider a neural network with two layers which has an input x. If only linear operations are applied (e.g. standard fully connected layer with only weights and biases applied), the output of the first layer will be $W_1x + b_1$ and the output of the second layer will be $W_2(W_1x + b_1) + b_2 = W_2W_1x + W_2b_1 + b_2$. If we denote $W_2W_1 = W_3$ and $W_2b_1 + b_2 = b_3$, the output of the second layer becomes $W_3x + b_3$, which is still linear. Even we have hundreds of thousands of layers, the final output is still in the form of $Wx + b$. The performance will be the same as one linear layer, and the features extracted from the thousands of layers will be same as the these extracted from only one layer. By adding non-linearity into the neural network, the interactions between layers can create different feature combinations with the same amount of layers. It makes the machine has better ability to learn more features. At the output layer, different designed features' expressions will be got. There are few popular operations that play a role of non-linearity such as *Rectified Linear Units (ReLu)*. With them, the machine is able to be trained and performed better.

Compression usually contains several non-linear operations such as rounding and quantisation. As stated, non-linearity is crucial for the deep learning. Thus, how compression could help deep learning, especially for the task of image classification, becomes worth to be explored.

1.2 Research Contributions

In this thesis, we proposed a new CNN topology that is able to input the compressed images into the CNN and increase the accuracy performance in both Top-1 and Top-5 perspective. This proposed CNN topology can be applied to state-of-the-art image classification CNN such as *Inception* network and *Residual* network. Due to the fact that which human-friendly compression quality in JPEG is the best for machine is unknown, this topology absorbs a range of qualities for the compressed images from low to high quality. By this, the CNN topology can absorb and learn the important information from the compressed images itself, which, as a result, is able to give a better accuracy performance.

To compare the performance with the state-of-the-art image classification CNN, the new topology is applied to *Inception V3* for *Inception* network and *ResNet-50 V2* for *Residual* network. The famous ImageNet ILSVRC-2012 dataset was used during the experiments. In order to have a valid benchmark, the validation Top-1 and Top-5 accuracy results for both *Inception V3* and *ResNet-50 V2* were reproduced based on the official frozen models. Then, we built and implemented the new topology on *Inception V3* and *ResNet-50 V2* and trained the last few layers by transfer learning-based block-by-block training technique. The results show that this topology does have the ability to increase the accuracy performance even though we only trained the last few layers for each model due to computational resources' limitation.

In addition, some explorations are executed to provide important insights into why input original images along with its compressed versions have better accuracy performance than input original images only. By visually comparing the images between the original image and its compressed version that has the best performance, it could be found out that compressed images more or less perform a function of texture adder and highlighting the main object(s), which obviously help the machine to have better accuracy performance in terms of image classification task.

1.3 Thesis Organisation

The rest of the thesis will be organised as follows:

In Chapter 2, some background knowledge will be given. The brief overview of the JPEG image compression method will be presented. Then, the overview of deep learning, the task of image classification and the famous dataset ImageNet will be introduced. The detail of the CNN including the functionality and properties of the convolution layer, the non-linear layer, the pooling layer, and the fully connected layer will be demonstrated. After that, the state-of-the-art CNN architecture for image classification including Inception network and Residual network will be described.

In Chapter 3, the new CNN topology will be proposed. The details of the architecture will be explained and demonstrated. They include the purpose of such design, how compressed feature would be utilised, how the model is going to be trained and evaluated and so forth. Furthermore, self-reflection over the model will be done as well to discuss the drawbacks and the limitations of such design.

In Chapter 4, experimental results over the two types of image classification CNN architecture type (Inception V3 for Inception network, ResNet-50 V2 for Residual network) will be shown in detail with the comparison with the benchmarks. Also, the experiment performance contributions including an efficient way to process data (either training or validation data) and a better way to utilise computational resources such as GPU and CPU will be explained in detail. In addition, insights into why compressed images could contribute better image classification results with our new topology will be discussed with examples.

In the last Chapter 5, we will make a summary of the thesis and discuss several potential paths for future works related to this interesting topic.

Chapter 2

Background

In this chapter, we will go over some background materials and topics related to this research thesis. We will start with an overview of one of the most common image compression formats called JPEG. Then, we will talk about the deep learning in the field of computer vision, especially the task of image classification and describe the details inside a CNN. After that, we will describe and review two state-of-the-art CNN architectures utilised for the task of image classification that we also utilise during our evaluation experiments.

2.1 Overview of JPEG Image Compression

JPEG is one of the most popular image compression formats utilised in our daily life. It contains several different modes such as *Baseline, Baseline Optimised* and *Progressive*. In this section, we will go over the most common mode, which is *Baseline*. JPEG baseline compression (short denoted as JPEG from now) was published and became popular since 1993 [30]. JPEG is a lossy compression method based on Discrete Cosine Transform (DCT). This transformation plays a similar role as famous Fourier transform does which transforms the information in spatial domain into frequency domain. The purpose to do that is the human visual system is prone to discard high-frequency information such as sharp edges and color hue in images [2]. A process called quantisation, which is a non-linear operation is applied in the frequency domain to discard extra 'useless' high-frequency information. Now, let us view JPEG in detail.

JPEG encoder starts the process by dividing the image into 8×8 blocks in each channel (RGB images have 3 channels with Red, Green, and Blue). Then, each block will operate the following operations:

- DCT transforms each block from 2D spatial domain to frequency domain. The mathematics formula is given by:

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} cos[\frac{(2x+1)u\pi}{16}]cos[\frac{(2y+1)v\pi}{16}] \tag{2.1}$$

  where $u \in [0,7]$ and $v \in [0,7]$ are horizontal and vertical spatial coordinate in frequency domain; $G_{u,v}$ is the DCT coefficient at coordinate (u,v); $g_{x,y}$ is the spatial domain value at coordinate (x,y); $\alpha(\cdot)$ is a normalizing scale factor to make the transformation orthogonal. After DCT, low frequency element will be located at the left top corner of the 8×8 block.

- Quantisation, which is the most important non-linear operations in JPEG that determines the quality of a picture, is applied after DCT. Since human eyes are more sensitive to low frequency information such as brightness difference over a large area, high frequency parts of each block could be more or less discarded without influence the visual feelings on the image. Following shows an example of default quantisation table for luminance:

$$Q = \begin{Bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{Bmatrix} \tag{2.2}$$

  and the quantisated coefficient is calculated as:

$$B_{u,v} = [\frac{G_{u,v}}{Q_{u,v}}] \ for \ u \in [0,7] \ and \ v \in [0,7] \tag{2.3}$$

  Where [x] denotes the rounding. We notice from the default luminance quantisation matrix that the high-frequency portion that located in the right bottom corner are quantised more. Depending on the degree of compression or the quality of the image, the quantisation coefficient could be modified correspondingly. In addition, there is a term called QF that numerically describes the quality of an image and has a certain relationship with the quantisation matrix. QF ranges from 1 to 100 with 100 represents the best image quality while 1 represents the worst. Given a QF, the new quantisation matrix can be calculated based on the default quantisation matrix with Equation 2.2 [5]:

$$QN_{i,j} = round(\frac{50 + S \times Q_{i,j}}{100}) \tag{2.4}$$

where $QN_{i,j}$ denotes the New Quantisation matrix processed from the default quantisation matrix with a certain QF and parameter S equals to $5000/QF$ if $QF < 50$ and $200 - 2 \times QF$ otherwise.

- Then, 2D frequency block is reordered to 1D by the zigzag pattern with the lowest frequency in the front, highest at the end. In this process, Run Length Encoder (RLE) is utilised for efficient encoding by only recording the length of the long sequence of the same value. For instance, after quantisation, there might be a lot of zeroes at the tail part of the 1D sequence, we could only record zero itself and the length of zeroes that appear in the sequence.
- At the end, Huffman coding is applied to finish the process in order to save more space. By doing Huffman coding, the pattern that appears the most frequently will be finally encoded with the shortest code.

The decoder performs the same operations that the encoder does but reversely. The quantisation operation plays an important role to remove unimportant information for human's vision system. So far, no suitable compression method is developed to keep important information for the machines. As discovered in Chapter 1, worse quality in JPEG (i.e. worse in human's vision perception), may be better in machine's perception. This research will utilise JPEG to discover the potential benefits machine is able to get from compression in the task of image classification.

2.2 Image Classification with Deep Learning

2.2.1 Overview of Deep Learning

Deep learning was introduced into the field of machine learning to further achieve the goal of Artificial Intelligence (AI). Deep learning is the process to learn the inherent features and representations of given data. The information obtained during the learning process is helpful to the interpretation of data such as text, image, and sound. Its ultimate goal is to enable machines to have the same analytical learning ability as human beings to recognise information such as text, images and sounds [8]. Deep learning enables machines to imitate human activities such as audiovisual and more or less thinking process. It solves complex pattern recognition and classification problems and makes great progress in the field of AI.

The concept of deep learning was initially proposed and introduced by G. Hinton from the University of Toronto in 2006 who stated two crucial points: a) the multi-layer neural network model has strong feature learning ability and the feature data acquired by the deep learning model could represent the original data well which will greatly help the task of classification and b) for the problem that the training of deep neural network is difficult to reach the optimal level, the layer-by-layer training method (which is similar that the block-by-block transfer training that we proposed in Section 3.3) can be used to solve it [10].

In fact, deep learning is to construct a machine learning architecture model with multiple hidden layers and obtain a large number of more representative feature information through large-scale data training in order to improve the accuracy of classification and prediction. The differences between the deep learning model and the traditional shallow learning model are: a) the structure of the deep learning model contains more layers, and the number of layers containing hidden neurons is usually more than 5 layers, sometimes even more than 10 layers and b) deep learning can extract features in detail from low level to high level by transforming the original input into a new feature space to represent which makes classification or prediction easier to achieve.

One of the keys that make deep learning success is the addition of non-linearity into the model. Thanks to the non-linearity, the deep learning model is able to extract richer features for the model to have better prediction performance. For neural networks, depth refers to the number of combinations of non-linear operations in the functions obtained by network learning. Source coding process, such as compression (image compression in our case), usually contains non-linear operations including rounding, quantisation and so on. Applying the compression to the images and sending them to the deep learning model to further aid the model to make a prediction is possible to perform better and motivates us to discover. Many works have done before to add extra non-linearity inside the model such as activation function or try to make the model deeper and deeper to contain more non-linear operations inside the model while compression could add extra non-linearity at the input stage to assist prediction.

2.2.2 Task of Image Classification

Classification is one of the most important task branches in the field of deep learning. In this type of task, the machine is programmed and revised to specify given inputs to pre-defined n categories [8]. A common branch of the classification is the image classification, which asks the machine to identify what the image represents or contains. Human, after growing from baby, has a certain ability to categorise items he/she sees. We can easily identify that Figure 1.1 contains a dog. Machines need to learn in order to gain the ability to categorise pictures as human does. Usually, an image classification model needs to be built and trained to be that smart. Certain learning algorithms are asked to be developed to let machine output a function $f : \mathbb{R}^n \rightarrow \{1..n\}$ [8]. Modern studies have already done many fantastic works on this task [16][26][28][9]. Usually, image classification models output the probability of the pre-defined categories that the input images belong to. It commonly could not output the position of the object.

There are several famous datasets in the field of image classification that people usually play with. Two of them are going to be introduced here. First of them is the MNIST database, which stands for Modified National Institute of Standards and Technology database. This dataset is developed by Y. LeCun, C. Cortes and C. J.C.Burges for image [17]. It is a dataset that contains only ten categories for handwritten digits from 0 to 9. It has a training set of 60,000 images and a test set of 10,000 images. MNIST is a relatively entry-level dataset in terms of image classification and is widely used in the field of deep learning to test a given model's performance. So far, the best error rate reported can be only 0.23% by using a hierarchical system of convolutional neural network [4] for MNIST.

Another famous image classification dataset is introduced in ILSVRC in 2012 [16]. ImageNet ILSVRC-2012 has a 1000 categories for people to classify in total. Its training set contains around 1.2 million images, the validation set has 50,000 images with 50 images per category and testing set has 150,000 images. Even the ILSVRC challenge has already stopped in 2017, this dataset is still widely used as a standard. In our research, the experiments were done based on the ImageNet ILSVRC-2012. Due to the fact that the ground truth for the testing set is not officially provided, the validation set was utilised for comparing with the benchmark. Usually, there are two different types of metrics to evaluate a model's performance on ImageNet ILSVRC-2012. One is Top-1 accuracy, which means the accuracy that the ground truth appears in the top 1 prediction, and another one is Top-5 accuracy, which represents the percentage of ground truth appears in top 5 predictions. The reason why Top-5 accuracy is introduced is that there are cases that images contain multiple objects, the ground truth might not always at the rank #1 since the occupational ratio of the ground truth object in the image is low. Currently, the best performance for ImageNet ILSVRC-2012 is the Efficientnet done by M. Tan and Quoc V. Le with Top-1 accuracy of 84.4% and Top-5 accuracy of 97.1% [29]. Table 2.1 summarises the accuracy performance of some of the state of art models.

Model Name	Published Year	Top-1 Accuracy	Top-5 Accuracy
VGG 16	2015	71.5%	89.8%
Inception V1	2014	69.8%	89.9%
Inception V3	2015	77.6%	93.8%
Inception ResNet-V2	2016	80.4%	95.3%
ResNet 50-V2	2016	75.6%	92.8%
ResNet 152	2015	76.8%	93.2%
ResNet-200 V2	2016	79.9%	95.2%
SENet	2017	82.7%	96.2%
GPIPE	2018	84.3%	97.0%
EfficentNet	2019	84.4%	97.1%

Table 2.1: Some of the State of art models' performace in ImageNet ILSVRC-2012 image classification task. Results up to one digital place.

2.3 Convolutional Neural Network

CNN was initially proposed by Y. LeCun, who was inspired by the biological brain, in 1988 for computer vision and named it the "Optimal Brain Damage" regularisation methods [18]. It is widely used to deal with the task of image classification for the fact that convolution operation could output a given feature from the sliding window and do the feature extraction, which is important for image classification. Overview of a CNN in the task of image classification is as follows: an input image is processed by convolution operations, which contains several steps and then outputs a pre-defined label for that input.

In the machine's perspective, it "sees" images as a group of numbers or pixels and treats these pixel values independently. It initially does not have the ability to consider and analyse these values together. For typical RGB images, each pixel in each channel is usually assigned by value $\in \{0, 255\}$, which only describe the intensity at that position. With convolution operation, the machine is able to analyse the pixels in a group and extract specific features such as boundaries and curvatures. By multiple convolution operations, a group of abstract features could be filtered out.

In detail, a CNN passes an image from the input layer. Then it lets the image go through a sequence of convolution layers which form the abstract feature maps through the convolution operation. Non-linear activation layers are attached after. They are the keys to add non-linearities into the feature extraction process and also called detect stage. Next, pooling layers are connected. They get the output from the non-linear activation operation and do certain down-sampling spatially. Finally, the data flow through a fully-connected layer to make the label prediction(s) with simple matrix multiplication and bias addition.

2.3.1 Convolution Layer

The convolution operation is a special type of linear operation, which simply replace the general matrix multiplication with convolution operation in the hidden layers [8]. Convolution layer is usually located before any non-linear operation. The matrix of the numbers (either initial image input or the output from the previous layer) is input into the convolution layer with a typical convolution as indicated in Equation 2.5. x is referred to the input of the convolution layer and w is the kernel, or the filter or the sliding window. Usually, the output is called the *feature map*.

$$y(t) = \int x(n)w(t-n)dn = (x * w)(t) \tag{2.5}$$

The kernel moves from the upper left corner of the input matrix. If the output needs to be defined with a typical spatial dimension, the input might be pre-processed by appending extra values such are zeros or average of the neighbor values. Then, the kernel multiplies its value with the input and these multiplications are them summed up. For example, if a 3×3 kernel used, these 9 values will be summarised by only one value after the convolution operation. The kernel is then moved by a certain step size and continue the process. This operation, in the beginning, is able to identify features such as edges and colors. More high-level features could be gained if more such convolution layer is added into the network.

The reason why convolution is helpful for the image classification performance is because of three important perspectives: *sparse interactions, parameter sharing, and equivariant representation* [8]. Starting from *sparse interactions*, this describes that each unit of the output of the convolution layers only interacts with certain input unit(s) [8]. Typically, the kernel size is smaller than the input size. Consider the situation that the input is an image of $2K$ image with resolution 3120×1440 which has 4,492,800 pixels in each channel, the kernel size could meanwhile be only 5×5, which is much smaller than the input size. By this, there can be much fewer parameters to operate, reduce memory and make the computation much more efficient. *Parameter sharing* represents the characteristic of the convolution operation that the kernel parameter among all location is the same rather than separate parameters. During training, only one kernel parameter group needs to be trained. By this, the storage could be saved while not affecting the running time of forwarding propagation [8]. Lastly, *Equivariant representation* means the output is always changed according to input's changes. This feature is caused by parameter sharing. Even based on the same image, if it is rotated or cropped, the same feature should be captured in different location of the new image. By this, convolution is able to extract the abstract features independent of the feature location of the input image.

2.3.2 Non-linear Layer

The non-linear layer, which is also called a detect stage, is usually located after the convolution layer. It brings one activation function to add the non-linearity property into the CNN. Some of the popular non-linear activation functions are listed in Equation 2.6, 2.7 and 2.8 with there plot shown in Figure 2.1. As discussed in the Chapter 1, the non-linearity is important for the neural network to work.The neural network would be not intense enough even it is very deep with non-linearity.

$$ReLu(x) = max(0, x) \tag{2.6}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.8}$$

Currently, the activation function ReLu, shown in Equation 2.6 is the most popular one that introduced by G. Hinton. Its training time in terms of gradient descent is much quicker than the other two types of activation function [16]. With the help of the activation function, the network becomes more powerful and has better ability to learning patterns and features rather than only linear features.

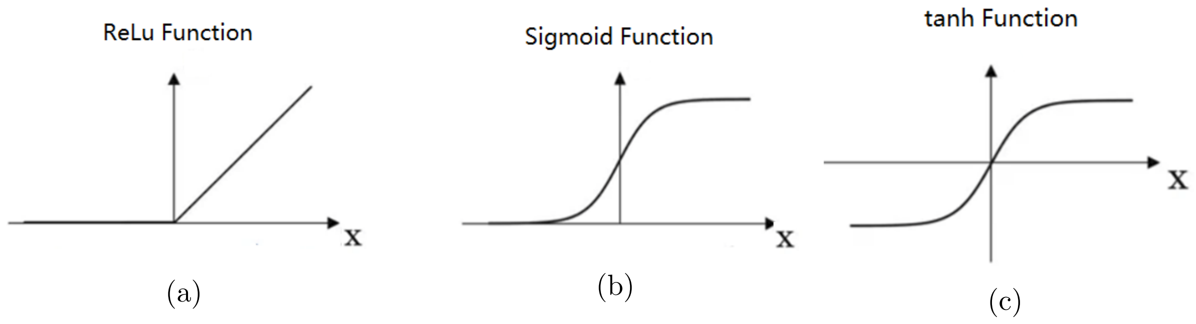


Figure 2.1: (a) ReLu Function Plot; (b) Sigmoid Function Plot; (c) tanh Function Plot

2.3.3 Pooling Layer

The pooling layer is typically appended after the non-linear layer. Pooling operation is actually another non-linear operation in the CNN as well. Some typical pooling functions that applied in the pooling layer include *max pooling and average pooling.* Pooling function further modifies the output from the detect stage by replacing it at a specific position with a statistical representation of the neighbor positions and realise the down-sampling operation. The purpose of the pooling function is to help the output representation become the approximately invariant translation of the input [8]. This invariant property is crucial due to the fact that the answer we would like to get is whether a certain abstract feature is in the input rather than where it is. Since pooling function typically considers a small range of value and output their statistical representation, a small position shift of the image would not greatly change the output of the pooling function. It makes the CNN be able to learn features through each convolution layers instead of learning the position of the object and utilise the position to make the final prediction(s). As a result, when the invariant property works properly with pooling, the statistical efficiency of the network could be significantly improved and less parameter storage is needed [8].

Typically, earlier convolution layers are trying to extract low-level abstract features, which are details, while later convolution layers extract relatively high-level features [32]. Pooling layer plays an important role to realise this phenomenon. If some low-level features are already extracted from the earlier layers, pooling function then plays a role as compression to discard the details, which are no longer need for analysis and make the later layers focus on the high-level features.

In addition, pooling layer is able to be utilised to manage the output size. As introduced, pooling function typically generates the output by considering the neighbor positions. The range of the neighbor positions to be considered could be modified so that the CNN can get a suitable output size regardless of the input size.

2.3.4 Fully Connected Layer

Usually, the convolution layer, non-linear layer, and pooling layer are nested together and are considered as a group. Let us denote them as convolution group. With several convolution groups append one after another, high-level features will be obtained from the output of the last convolution group. The fully connected layer is usually attached at the end with the last high-level features as input and output the prediction array. The prediction array is an array with a size of the number of categories to classify with each cell has a value of the confidence probability of the corresponding label.

The name "fully connected" comes from the fact that each neuron from the feature output (the input of the layer) is connected to the output neurons (the prediction). It typically utilises matrix multiplication operation to realise this, flatten the matrix and yield the 1D prediction array, and finish classifying the input image.

2.4 State-of-the-art CNN Architecture

In this section, two of the most famous image classification CNN architectures are going to be introduced and described in detail, which are *Inception network and Residual Network*. Many studies have been done to get better image classification accuracy performance based on these two CNN architecture styles, as we could found out many candidates of ImageNet challenge through years utilise these two network as basics and made specific modifications [11]. Both styles were experimented and evaluated on the newly designed CNN topology, which will described later in Chapter 3 and 4.

2.4.1 Inception Network

The development of Inception network could be divided into four stages from the introduction of Inception, addition of batch normalisation technique, optimisation of inception and combination with Residual Network (ResNet). Each stage with name *Inception V1, Inception V2, Inception V3, Inception ResNet V2* correspondingly.

Inception V1

Inception network is an important milestone in the development of CNN classifier. Before the existence of Inception, most popular CNN architecture building strategy was just to stack as many convolution layers as possible in order to make the network deeper, hoping to get better accuracy performance. However, simply making the architecture deeper may cause issues as follows:

- There would be too many parameters which can cause overfitting phenomenon [8];
- The backward propagation value would become close to zero when approaching the input if the depth of the CNN becomes deeper. The gradient vanishment could easily happen. This phenomenon is explained with an example below;
- The computational resources required for training deeper CNN would be much larger than shallow CNN.

Gradient Vanishment Example

Suppose the activation function is CNN is Sigmoid $\frac{1}{1+e^{-x}}$ with parameter $e^{-x} = 1$, the derivative of Sigmoid will be

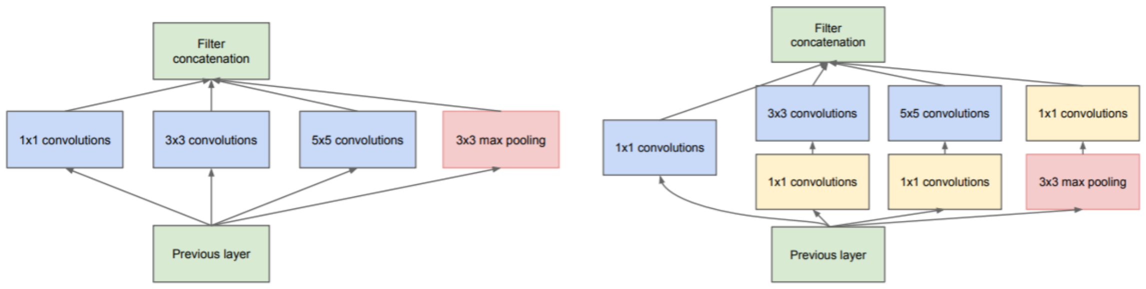
$$\frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{4}$$

when we do the back propagation. If we have many layers (denote the number of layer as n) of such Sigmoid activation function, the parameter update will be vanish easily:

$$\lim_{n\to\infty}(\frac{1}{4}) = 0$$

Inception increases the performance by touching the core architecture rather than the depth of the architecture. The author of the paper believes that the solution to the problem listed above is to change full connection to the sparse connection. Convolution layer is a sparse connection. However, the efficiency of numerical calculations of asymmetric sparse data is low due to the fact that the hardware is optimised for dense matrix, so it is necessary to find the optimal local sparse structure which can be approximated by convolution network. In addition, this architecture can be implemented with existing density matrix computing hardware, and the result is Inception [3].

Figure 2.2 shows how the proposed Inception block looks like. We believed that the reason why Inception block could help is that such block is about to capture multiple different abstract features by having different size of the receptive field at the same level. Utilising certain 5×5 kernel size allows the Inception block to capture features that cover relatively larger areas. It was pointed out that the number of neurons at each step is significantly increased, and the computational complexity is not unlimited by reducing the dimension of larger blocks before convolution with 1×1 kernels [3]. Inception V1 finally gets 89.9% Top-5 accuracy performance in ImageNet challenge.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2.2: Inception Block [3]

Inception V2

The key upgrade for this generation of Inception is adding *Batch Normalisation* into the architecture. The main purpose of batch normalisation is to solve an issue called *Internal Covariate Shift* [13]. This issue happens while updating of the parameters at each layer during the training process. For a CNN, the input of layer n is the output of layer $n-1$. The parameters will be changed at each training step. Since the parameters of the previous layers always change, the distribution of each layers inputs changes during the training process even with the same input for the CNN. This slows down the training by requiring lower learning rates and careful parameter initialisation. This also makes it hard to train models with saturating non-linearity [13]. Intuitively, in order to solve this problem, make sure the output from each layer for each batch of the data have the same distribution is the solution. Batch normalisation realises this by making the normalisation of the output of each layer to Gaussian distribution with N(0, 1) as illustrated in Algorithm 1.

Overall, with batch normalisation, the input distribution at each layer is able to remain the same (Gaussian distribution). As a result, the gradient vanishment can be resolved

Algorithm 1 Batch Normalisation that solves Internal Covariate Shift [13]

```
    Input: A batch of input $\mathbb{B} = \{x_1, ..., x_n\}$; Trainable variables: $\beta, \gamma$
    Output: Normalised output $y_i = BN_{\beta,\gamma}(x_i)$
1: $\mu_{\mathbb{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                                ▷ Batch mean
2: $\sigma_{\mathbb{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathbb{B}})^2$        ▷ Batch variance
3: $\widehat{x_i} \leftarrow \frac{x_i - \mu_{\mathbb{B}}}{\sqrt{\sigma_{\mathbb{B}}^2 + \epsilon}}$                     ▷ Normalisation
4: $y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv BN_{\beta,\gamma}(x_i)$                ▷ Scale and Shift
```

and speed up the training process simultaneously. However, there raises another question that why do we need to scale and shift at the last step of batch normalisation. Consider the popular used activation function ReLu with expression $f(x) = max(0, x)$ and its plot is as shown in Figure 2.1a. As we could see, the activated part is in the region $x \geq 0$. However, half of the information would not be activated when the layer output is normalised to centered at zero. This would finally influence the efficiency of the training process. Therefore, scale variable γ and shift variable β is defined and utilised to adapt the activation function. Their main role is to find a linear and non-linear equilibrium point. By this, we can not only enjoy the non-linear expression ability, but also avoid the problem that the non-linear saturation causes the network convergence to slow down at the end. Overall, Inception V2 gets 92.2% Top-5 accuracy performance in ImageNet.

Inception V3

The third generation of Inception did further optimisation in the Inception block. The improvements can be summarised in three perspectives: further factorise convolution kernel size, utilisation of auxiliary classifier and altering the way to change the size of feature maps [28].

Starting from the factorisation of convolution kernel size, Inception V3 proposed two different ways of factorisation of convolution kernel which are symmetric way and asymmetric way. Symmetric factorisation would replace one convolution kernel with two exactly the same smaller size kernel. For example, a 5×5 kernel could be replaced by two 3×3 kernels as shown in Figure 2.3. With this factorisation, the computation resources needed is only $\frac{3+3\times3}{5\times5} = 18/25 = 72\%$ of the one needed for the initial inception block. The overlap receptive field of the two 3×3 kernels could also play the same role as 5×5 kernel [28]. Another method is asymmetric factorisation, which replaces the initial kernel with two different size kernel. For instance, a 5×5 kernel is going to be replaced by one 1×5 kernel and one 5×1 kernel, as presented in Figure 2.4. By this, the resources could be further reduced to $\frac{5+5}{5\times5} = 10/25 = 40\%$. However, the asymmetric way is not very effective in the first few layers. However, it has an obvious effect on the intermediate layer with the size of 12-20 layers [28]. In conclusion, factorisation of convolution kernel could reduce a huge amount of parameters if there are many Inception blocks. Also, when one convolution kernel is factorised to two, the depth of the CNN is also increased by one. This could add an additional layer of non-linearity that could improve the expressive ability of the model and could handle more spatial features and increase the feature maps' diversity.

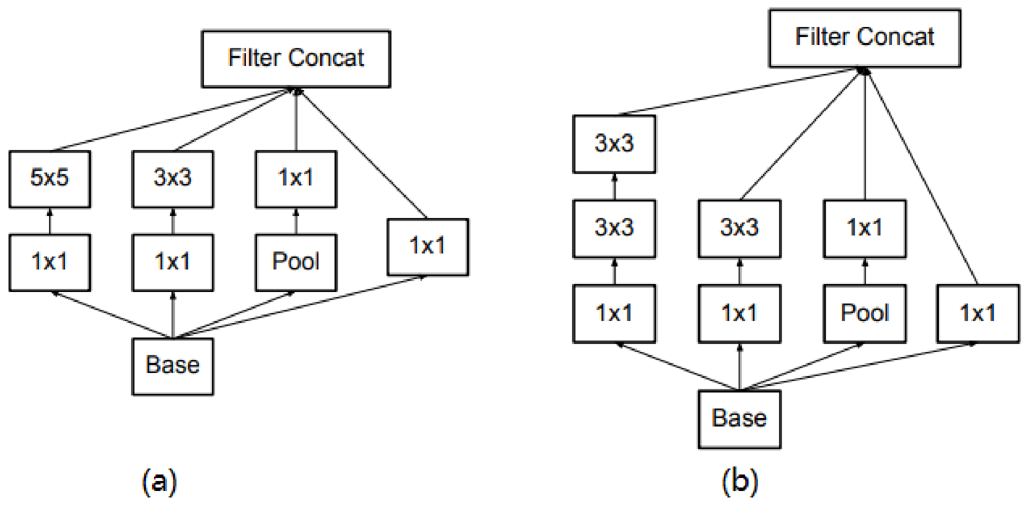


Figure 2.3: (a) the Inception block without symmetric factorisation; (b) the Inception block that replace the 5×5 kernel by two 3×3 kernels [28]

The second improvement is the addition of auxiliary classifier. The auxiliary classifier is an additional classifier that yields the output from the middle of the model. It was demonstrated that the auxiliary classifier could help to have better convergence performance and also stable the training process [19]. We found out why auxiliary classifier could help in another perspective. Usually, when the CNN is too deep, the gradient vanishment becomes a serious problem during training. The auxiliary classifier could help to pass the gradient back effectively to the earlier layers' parameters, which could avoid gradient vanishment and speed up the training process. Sometimes, the features abstract from the middle layers also help to classify objects. By utilising both high-level features and medium level features to identify the object could make the model more confidence to get the correct re-

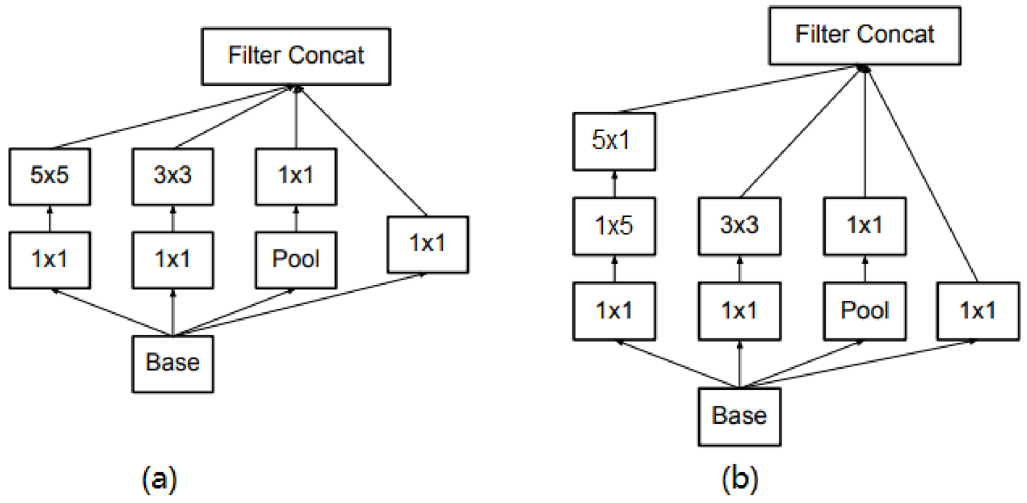


Figure 2.4: (a) the Inception block without symmetric factorisation; (b) the Inception block that replace the 5×5 kernel by one 1×5 kernel and one 5×1 kernel modified from [28]

sult. There is 0.4% Top-1 accuracy and 0.2% Top-5 accuracy improvement if the auxiliary classifier is added into the Inception model [28].

The third upgrade in Inception V3 is an efficient way of grid size reduction. A common CNN usually utilises the non-linear operation pooling, either max pooling or average pooling, to detect the output size at each layer. However, there is always a crucial drawback wherever the pooling is located, either before or after the feature capture with convolution operation (such as Inception block): If the pooling is put before convolution layer, a lot of information is going to be discarded before the convolution layers try to capture the feature due to the fact that pooling plays a role as down-sampling. Fail to capture features will make the model fail to make the right predictions. On the other hand, if the pooling is located after the convolution layers, the convolution layers need to double the filter size in order to remain expressive ability of the features after pooling as shown in Figure 2.6 (b), which will significantly increase the computation time. Inception V3 introduces a way that has two channels, one is the convolution layer, and the other is the pooling layer. The feature maps generated by the two parallel channels are the same size. Finally, the result from the two-channel concatenates together to realise the grid size-reduction [28].

Overall, Inception V3 is able to reach 77.59% Top 1 accuracy and 93.83% Top 5 accuracy as we ran the ImageNet validation set based on TensorFlow tutorial's Inception V3 frozen

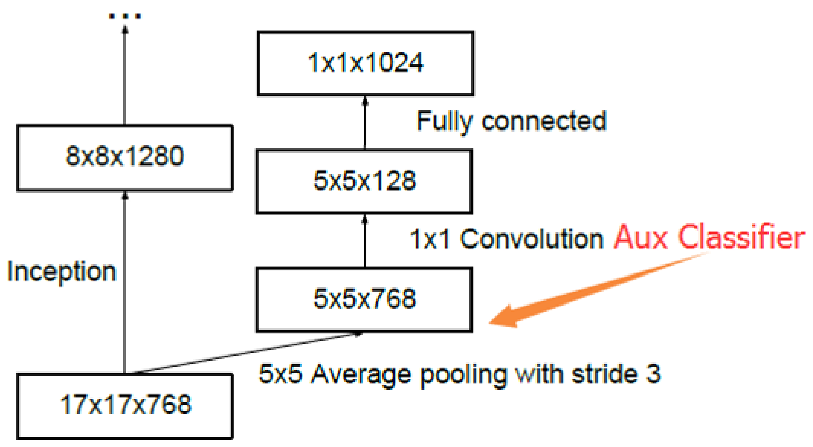


Figure 2.5: Auxiliary Classifier that added in the Inception V3. Modified from [28].

model.

Inception ResNet V2

As ResNet is one of the most popular image classification networks, which will be discussed in detail in the next section, the latest version of Inception tried to merge Inception network with ResNet to get the benefits from both of them. As a result, Residual Inception block was introduced and make the Inception CNN model is able to go over 150 layers and still have the ability to predict more accurately than Inception V3. Inception ResNet V2 is able to reach 80.4% Top-1 Accuracy and 95.3% Top-5 Accuracy in the ImageNet ILSVRC-2012 challenge.

2.4.2 Residual Network

Similar to Inception, ResNet was designed to solve the problem that causes by too deep CNN [9]. If we test a normal CNN with 20 layers and 56 layers, previous common sense will easily state that the CNN with 56 layers is going to yield better results. However, the truth is reversed as shown in Figure 2.8. The deeper 56 layers normal CNN performs worse than the 20 layers one in both training and testing dataset. Hence, not the deeper the network, the better. Through experiments, it can be found out that with the increasing network level, the accuracy of the model is continuously improved. When the number

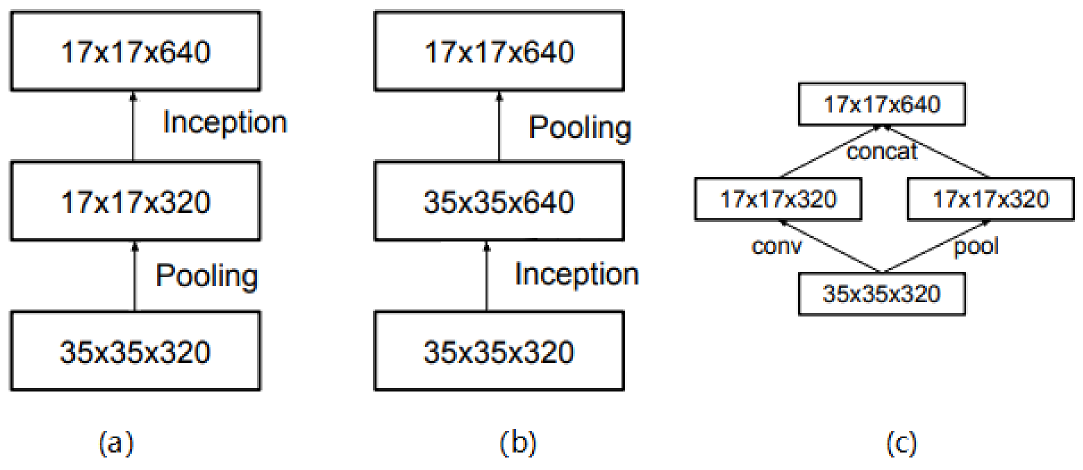


Figure 2.6: (a) Grid size reduction by pooling before convolution, which will loss a lot of information; (b) Grid size reduction by pooling after convolution, which will greatly increase the amount of calculations; (c) The parallel style to reduce the grid size which solves the issue from (a) and (b). Modified from [28].

of layers of the network is increased to a certain number, the training accuracy and test accuracy are rapidly degraded. It indicates that when the network becomes deeper, the network has become more difficult to train [9]. The reason is the same as the one indicated in Inception V1: gradient vanishment which makes the parameter in earlier layers could not be adjusted properly.

ResNet was introduced to solve this problem and help to make the CNN deeper. Consider we have a shallow network (just a few layers) which already reach the saturate situation inaccuracy. Now, attach some identity mapping layers ($y = x$ or input at layer n = output at layer n-1) will increase the number of layers of the model and meanwhile will not increase the error. In other words, the deeper network will not cause worse accuracy performance. The idea of using identity mapping to transfer the output of the former layer directly to the latter is the inspiration of ResNet.

Suppose the input of the network is x and the expected output is $H(x)$. Recall that accuracy performance would be saturate for normal CNN at a certain number of layers and the way we are able to remain the accuracy while increase the layer is attaching identity mappings. Thus, in deeper layers, the learning goal becomes to make the output $H(x)$ become as close as possible to the input x in order not to drop the accuracy performance of the network. The residual block is demonstrated in Figure 2.9. The proposed residual

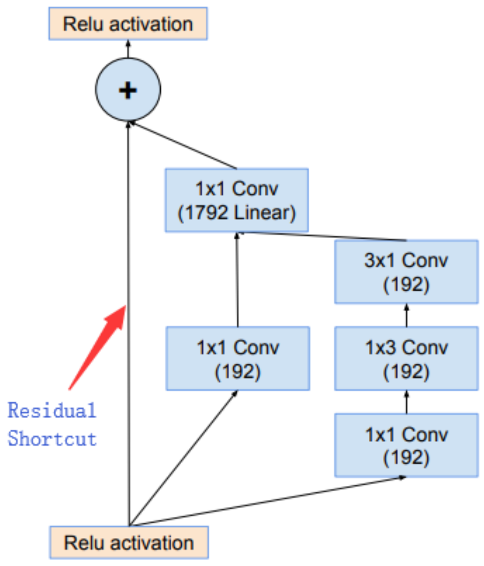


Figure 2.7: An example block of Residual Inception block with a residual short as highlighted [27].

block utilises the shortcut connection, the input x is directly passed to the output without any modification. The output now could be formulated as $H(x) = F(x) + x$. If $F(x) = 0$, then $H(x) = x$, which is the identity mapping we would like to see to make the network deeper. Thus, ResNet is equivalent to changing the learning goal: learning the difference between the target value $H(x)$ and x which is called the residual $F(x) := H(x) - x$ rather than learning a complete output mapping from the input. Therefore, the training goal for the latter layers is to approach the residual result to zero, so that the accuracy does not decrease while the network becomes deeper.

ResNet is actually a combination of several shallow networks. It does not try to solve the problem of gradient vanishment but tries to avoid it. It can accelerate the convergence of the network and have better results for deeper models since it is a combination of several shallow networks while shallow networks will not have obvious gradient vanishment during training. The ResNet performance over ImageNet ILSVRC-2012 could be found in Table 2.1.

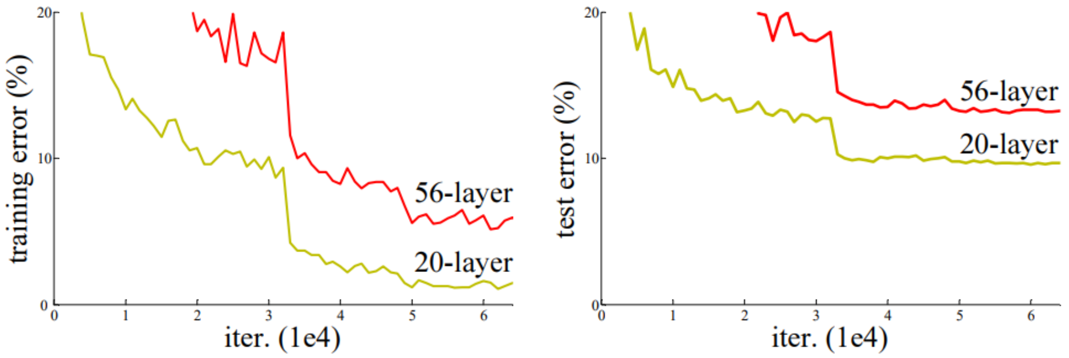


Figure 2.8: Training and testing error trends over time with dataset CIFAR-10 over normal 20 layer and 56 layers CNN. The deeper the CNN, the worse the result. [9].

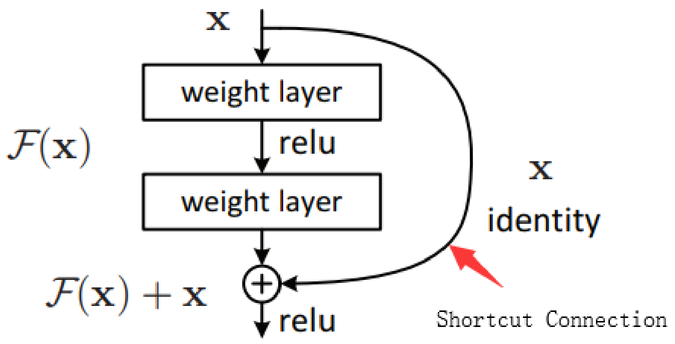


Figure 2.9: Residual block in the ResNet. Modified from [9].

2.5 Summary

In this chapter, we gave an overview of the JPEG image compression in terms of its operation process and pointed out the fact that the compression is fundamentally non-linear operation as it contains non-linear operation such as quantisation, which gave us the inspiration to design the new CNN topology. Then we introduced the image classification task with deep learning. Here, we gave an overview of deep learning and discussed the background of the task of image classification with its popular datasets. Then, we discussed the CNN in detail including its architectures and their purposes respectively. What's more, we reviewed two state-of-the-art CNN architecture that currently people utilised in the task of image classification that we also applied our newly designed topology on, which are Inception network and Residual network. The idea of the new CNN topology for image classification is all inherited from the concepts presented in this chapter and will be described further in the following chapters.

Chapter 3

New CNN Topology for Image Classification

3.1 Overview

As discovered previously, it can be found out that the non-linearity is crucial for the machine to learn richer features. Unlimited linear combination will finally yield a linear combination, which cannot provide any help. Furthermore, compression contains non-linear operations such as quantisation which is discussed in Section 2.1. To combine the non-linearity from the compression to the deep learning's model is one of the crucial motivations to design the new CNN topology for the image classification task.

In this chapter, we will discuss more about this new CNN topology with its designing principles in detail. Furthermore, we will introduce a particular training method that let us get incremental improvements while not over-use the computational resources initially. Also, the parameter control method named parameter sharing will be introduced and applied to this topology's training.

3.2 Architecture Design

From the observations in Chapter 1, we find out that there are cases that images compressed by JPEG have better accuracy performance than the original images. As human's vision system is usually not sensitive to high-frequency content, JPEG wipes out the high-frequency contents to realise compression. Removing them will not influence the visual feeling for human and save the storage simultaneously. However, if we wipe too many contents in the frequency domain such as the contents in medium frequency or even low frequency, which produces medium or low-quality images, will influence human's visual feeling and judgment on the image contents like the one shown in Figure 1.2. Nevertheless, the machine may appreciate these changes made by JPEG. Even in a low-quality image, machine, in some cases, could have better prediction performances. It should be noted that the better performance on low-quality images is not because the training set contains many low-quality images and make the CNN get used to them. It was demonstrated that the images from imageNet ILSVRC are high-resolution [16]. Other than other, images in the training set are always high-quality as well [7]. It could be found out from the following histogram that quality of the images from the training set of imageNet ILSVRC-2012 are high as well. High-quality images dominate the training set while low-quality images rarely happen.

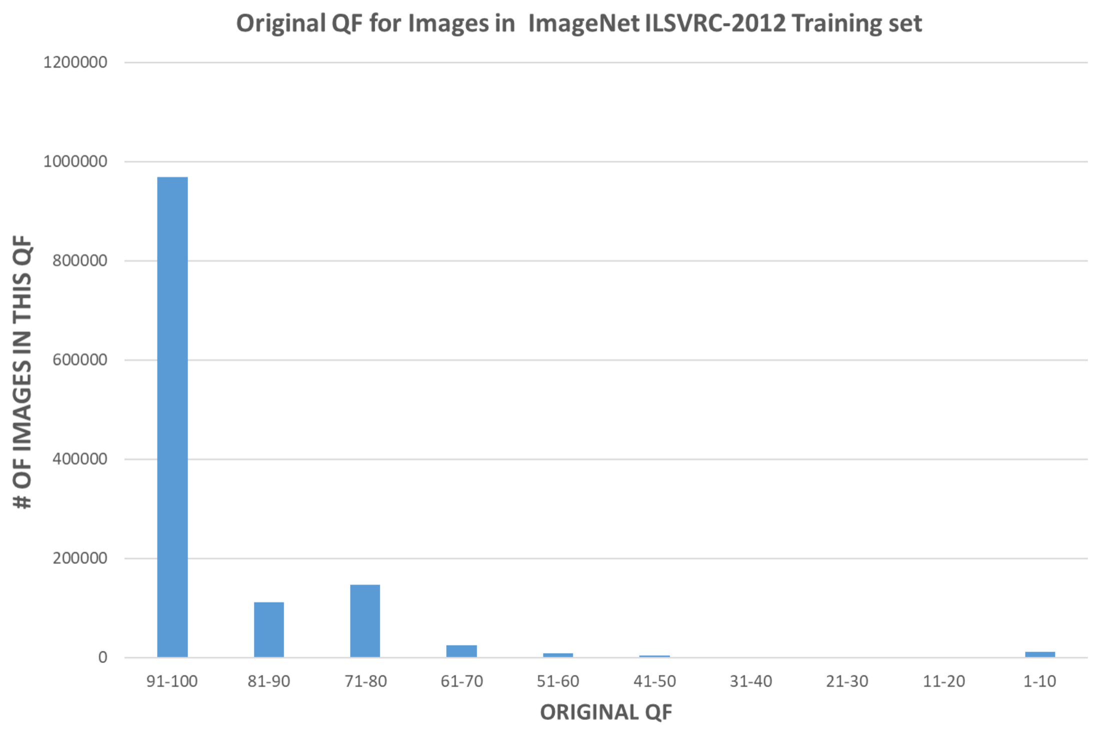


Figure 3.1: The original QF distribution in the training set of imageNet ILSVRC-2012

It is worth noting that the machine does not have the best prediction performance based on the same image quality (QF in JPEG) for all images [1]. Figure 1.1, 1.2 and 3.2 show the three example images that machine (based on Inception V3) yields the best accuracy over the ground truth on the low, medium and high quality of the original image respectively inspired from the observation from H. Amer et al [1]. Figure 1.1 has a best performance at QF equals to 10, which is low quality. Figure 1.2 has a best accuracy at QF equals to 40, which is medium quality while 3.2 has QF 100 to have the best result. Nevertheless, compressed image version may not always improve the accuracy performance. [1]. Therefore, there is no clue for us to conclude which quality factor of the compressed image or original image is the best choice for a given image either statistically or from the information delivered from the image itself.

(a) (b)

Figure 3.2: (a) original image #4 with object soup bowl hound from ImageNet ILSVRC-2012 validation set; (b) Compressed version of (a) by JPEG with Quality Factor = 100. Orignal image has a quanlity factor = 100, (b) and (a) has barely no difference in human's vision system

As a result, the new CNN topology is designed to tackle this situation. With this topology, we send the compressed version of the image from high quality to low quality with a constant quality step. In the case of JPEG compression, we selected the QF = 10:10:100 as inputs for a single image to have a uniform choice on the image quality. Also, since the benchmark is generated from the original image only, the original image is input to the network as a branch in order to get the benefit from the compressed images. As a result, the improvement will be built based on the original benchmark rather than from a random unknown situation.

Unlike data augmentation, our topology combines the variations of the original input images in group and process. The main purpose of data augmentation is to enlarge the dataset. The images processed that after a certain operation, such as flip, rotation, crop and so forth, will become independent images with the same ground truth as the original image and send into the model to train. Convolution is used to extract features from an image, major objection's location difference of the same image will yield different activation. When we have multiple convolution layers that are cascaded together, which extract a different level of features, the final output from the last layer will no longer be the same. By this, the purpose of augmenting the dataset can be realised. In our topology, we would like to append addition information from the compressed images to the original image in order to make the topology to have a better decision by utilising these combined activation generated from the compressed image based on the fact that we observed there are cases that the compressed version of the images are able to have better prediction performance than original images.

As illustrated in the architecture in Figure 3.5, all 11 branches, including original images and 10 compressed images in different qualities, goes through to the SAME CNN independently and the last high-level activation outputs for all of them are merged by concatenation operation before sending into the last decision layer. The last decision layer's weight size will be adjusted correspondingly and finally yield the prediction vector (with a size of 1,000 in ImageNet since it has 1,000 categories) which is in the order of its confidence on all category labels.

3.3 Performance Metrics

The performance metric that we apply to our model is the same as the one that people commonly use in image classification. In this section, we will introduce that metric, give a certain definition, and show how we utilise it during evaluations.

The confusion matrix is one of the most common ways utilised to measure the performance of a model in the task of image classification. To explain, let us denote 1 as "True" and 0 as "False" in a classification task with only these two labels. There will be four situations that might happen: a) True Positive (TP): label is 1 and the prediction is also 1; b) True Negative (TN): label is 1 and the prediction is 0 instead; c) False Positive (FP): label is 0 and the prediction is 1 instead and d) False Negative (FN): label is 0 and the prediction is also 0. The relationship is summarised in Table 3.1, which is as known as the confusion matrix. If we have 1,000 categories to classify, such a table would extend to a size of $1,000 \times 1,000$ correspondingly. The right predictions only appear in the situation a) and d) which are the diagonal of this matrix. As a result, the accuracy performance of a given model could be expressed as Equation 3.1.

Ground Truth Label / Prediction	1 - True	0 - False
1 - True	TP	FP
0 - False	FN	TN

Table 3.1: Confusion Matrix for 2 categories.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

Furthermore, it can be fallen into two categories in terms of *Accuracy*, which are Top-1 accuracy and Top-5 accuracy. Top-1 accuracy means the accuracy when we only consider the rank #1 prediction in the prediction vector. On the other hand, Top-5 accuracy considers the top 5 predictions. When the ground truth label appears in one of the first 5 predictions, it would be treated as a correct prediction. The reason why Top-5 accuracy exists is further discussed in Section 4.4. Also, Top-1 accuracy is crucial due to the fact usually only the rank #1 prediction will be demonstrated as a result to the users in daily life applications such as Google's Vision AI and Huawei's HiVision. The Top-1 accuracy can reflect the performance better when it is in the application stage. As a result, both measurements would be utilised in our experiment to have multiple angle comparisons on the accuracy performance.

3.4 Training Strategies

3.4.1 Training Methodology

Training a raw CNN from scratch (start all parameters from random parameter initialisation settings) is always painful. It is reported that with two NVIDIA® Tesla® P100 GPUs for training on original Inception V3, the rate of training is 284 images per second [21]. Consider the training set from the ImageNet ILSVRC-2012, which has 1,281,167 images. The time we need to finish training only one epoch [1] is:

[1]One Epoch is when a whole dataset is processed (pass forward and backward) the neural network ONCE.

$$1,281,167/284 = 4495.32\ secs = 1.24\ days \tag{3.2}$$

Usually, one epoch is not enough to have the best result. Multiple epochs will cost much longer time for the training. In our situation, we have two NVIDIA® GeForce® RTX 2080 Ti GPUs which are relatively similar to NVIDIA® Tesla® P100 GPU. The main comparison criteria of the GPU for deep learning are a) memory bandwidth which shows the ability of the GPU to handle a large amount of data; b) clock rate which indicates how fast can the GPU process data and c) GPU RAM size which shows the capacity the amount of data GPU can hold one time (i.e. the training batch size). From the comparison we could say with our GPU environment, we would consume similar length of time if we would like to train the model from scratch.

Benchmark	NVIDIA® Tesla® P100	NVIDIA® GeForce® RTX 2080 Ti
Memory Bandwidth	**732.2 GB/s**	616 GB/s
Clock Rate	1329MHz	**1545MHz**
GPU RAM	**16GB**	11GB

Table 3.2: Performance comparison between NVIDIA® Tesla® P100 and NVIDIA® GeForce® RTX 2080 Ti based on the three major benchmarks.

Here, we alter the training method idea based on the concept of transfer learning. The fundamental idea behind transfer learning is utilising the pre-trained model's high-level feature outputs and re-train the tail parts of the model (such as the prediction part and last few convolution groups) to solve customised tasks. With this, the training time to a customised task such as image classification with particular input or categories can be shortened dramatically. The transfer learning process starts with the bottlenecks' [2] generation. Then, we treat these bottlenecks as the input of the tail lays' re-training input and do the general training process lastly. Our model can be considered as doing the same task as the original model does with the same categories but with different inputs. To fully utilise the resource we have, we proposed *Block-by-block training* which is an extension of transfer learning.

[2]Bottleneck is a term we defined as the output of the layers before those need to be re-trained or the input of the layers that need to be retrained or the input for the trainable blocks for the current node and TensorFlow Hub calls this an "image feature vector" [23].

Block-by-block Training

We trained our proposed topology with TensorFlow deep learning framework on two NVIDIA® GeForce® RTX 2080 Ti GPUs with standard Stochastic Gradient Descent (SGD) with a batch size of 100 and last 20 epochs for every block's training. For both Inception V3 and ResNet-50 V2, we utilised pre-trained frozen models which are publicly available as our starting point to training our proposed topology. For each model, we started with last logits block (usually the fully connected layer or the prediction layer), we apply the normal transfer learning and we call this as the first training run. The first training run has a random weights initialisation. After that, we continuously add another extra tail block to process the training run. Starting from the second training run, we load the parameters from the previous training run that has the best Top-1 accuracy performance and let this as initialisation of the training run and gradually decrease the learning rate. The number of training runs n can be decided manually or depends on the machine capacity (e.g. GPUs and CPUs configuration). The detail of the Block-by-block training process is described as a flowchart shown in Figure 3.4. The overall learning rate schedule is applied gradually which will be discussed next.

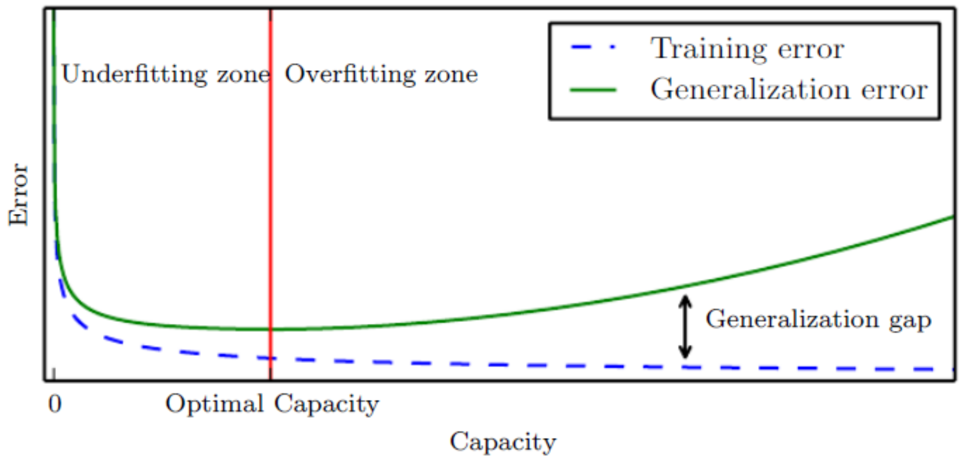


Figure 3.3: Typical reliationship between model capacity and training and testing error. It could be found out that when the capacity of a model becomes larger, the testing or generalisation error becomes bigger. The gap between training error and generalisation error becomes bigger as well as overfitting becomes more serious [8].

Learning Rate Schedule

The last prediction block's training starts with a learning rate of 0.01 by freezing all previous block. When the validation error for current block reaches a plateau (the phenomenon of overfitting observed) with the current learning rate, we stop the training of the current layer and reduce the learning rate by a factor of 10. Each block of training from the second block will start with the new reduced learning rate as a starting point and continue. We stop the training of the current block when no improvement happens in terms of validation error last for one epoch. Overall learning rate schedule was applied as described in Algorithm 2.

Algorithm 2 Learning Rate Schedule

```
1: while training not over do
2:     if Training the last logits block then
3:         $LR \leftarrow 0.01$                ▷ Initial setting of the learning rate for 1$^{st}$ block
4:     else
5:         Unfreeze one more block from the tail of the model
6:         $LR \leftarrow$ LR / 10             ▷ Initial setting of the learning rate for other blocks
7:     Evaluate current model save the checkpoint every 2,000 step.
8:     if Validation error after evaluation reaches a plateau then
9:         Go to Step 1.
```

3.4.2 Parameter Sharing

As introduced, the 11 branches in the new CNN topology takes the original image and 10 images that compressed in different degree as input and they pass through each branch independently. Recall that our observations on the example image Figure 1.1, 1.2 and 3.2 are based on the fact that the original image and the compressed image are evaluated based on the same model rather than the individual model. To be consistent with the settings in the observation, the parameters across the 11 branches maintain the same (shared). In other words, the loss is calculated based on the information from the 11 branches, or the data is passed forward through all 11 branches. However, when updating the parameters, it only backward passes to one branch and other ten branches simply share the same parameters with that branch. The data flow of the forward and backward propagation could be found in the architecture diagram in Figure 3.5.

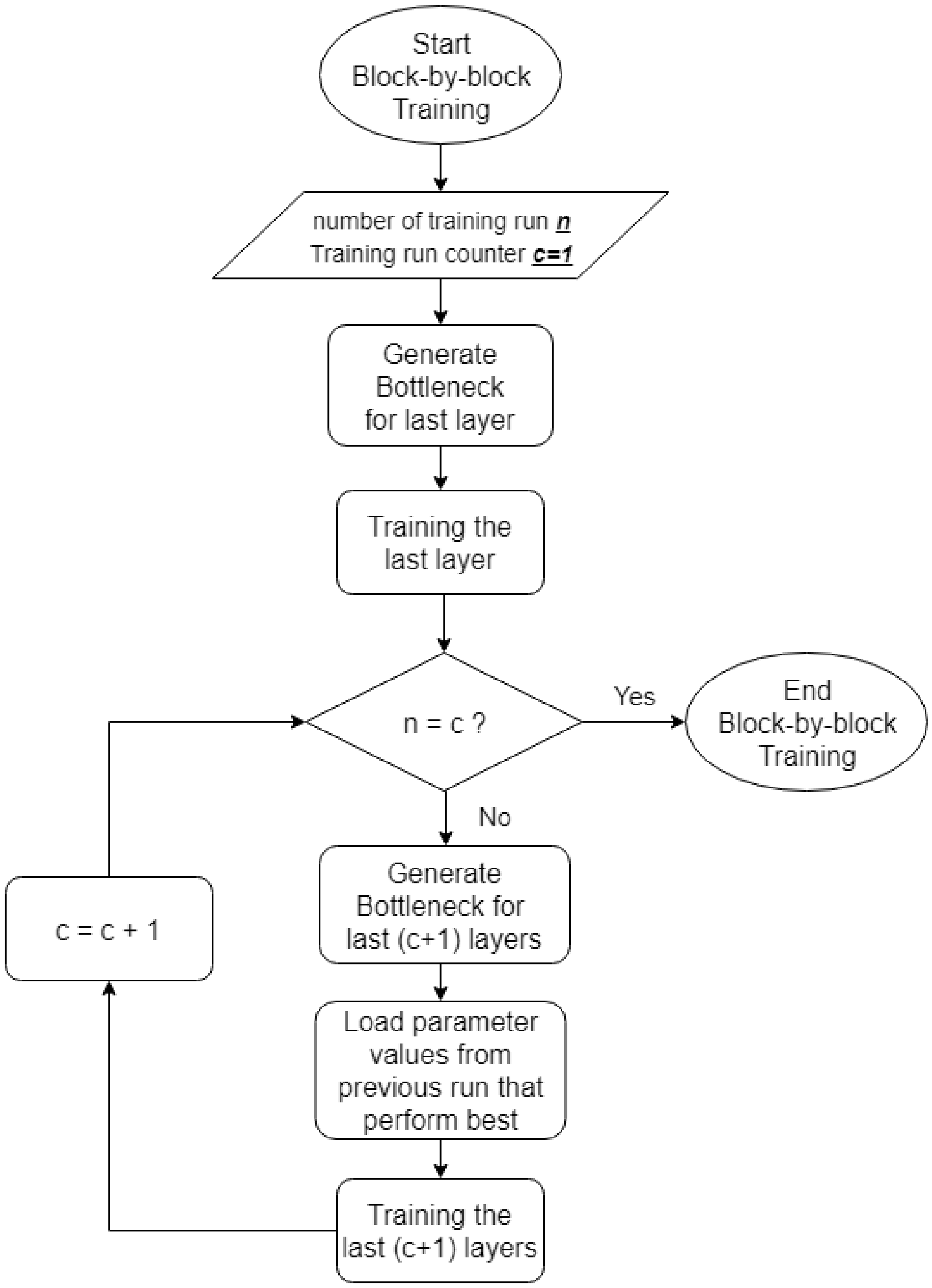
Additionally, if a model has too many variables or the capacity of a model is too large, the model is going to be hardly trained. The phenomenon of overfitting could happen easily as demonstrated in Figure 3.3 [8]. With parameter sharing, we can control the capacity of the model to be at least the same as the original model.

3.5 Limitations of the new CNN topology

The 11 branches of this topology are needed to input the compressed versions of the original images. However, this introduces 11 copies of the original CNN models into this new topology as well. Even though the depth of the topology remains the same as the original model, it becomes much wider. The total number of parameters in the new topology, with both trainable and non-trainable, is 11 times the original images. This could easily make the GPU RAM resource exhausted if we desire to train and evaluate the whole topology. That is why the Block-by-block training method is applied gradually to see whether current training run reaches the upper limit of the GPUs' limitation and not over use the computational resources initially.

3.6 Summary

In this chapter, we proposed the new CNN topology that can be applied in a general CNN model in the task of image classification. We have introduced the detail of this topology design and shown in Figure 3.5 and its principle to input the compressed information to the topology and get the useful information from the compressed input. Also, the general performance measurement, accuracy in Top-1 and Top-5 perspective, are defined to measure the performance of the newly designed topology. A different training strategy called Block-by-block training, which is based on the concept of transfer learning is proposed to train this CNN topology. Parameter sharing is defined and applied to the training of this topology to save computational resources and make the setting of the topology to be consistent with observations'. The limitations of this topology that it makes the number of parameters too large for the machine to hold and make it hard to train the whole topology are also discussed.


Start Block-by-block Training
number of training run n
Training run counter c=1
Generate Bottleneck for last layer
Training the last layer
n = c ?
Yes
End Block-by-block Training
No
Generate Bottleneck for last (c+1) layers
Load parameter values from previous run that perform best
Training the last (c+1) layers
c = c + 1
Figure 3.4: Block-by-block training that based on transfer learning flowchart.

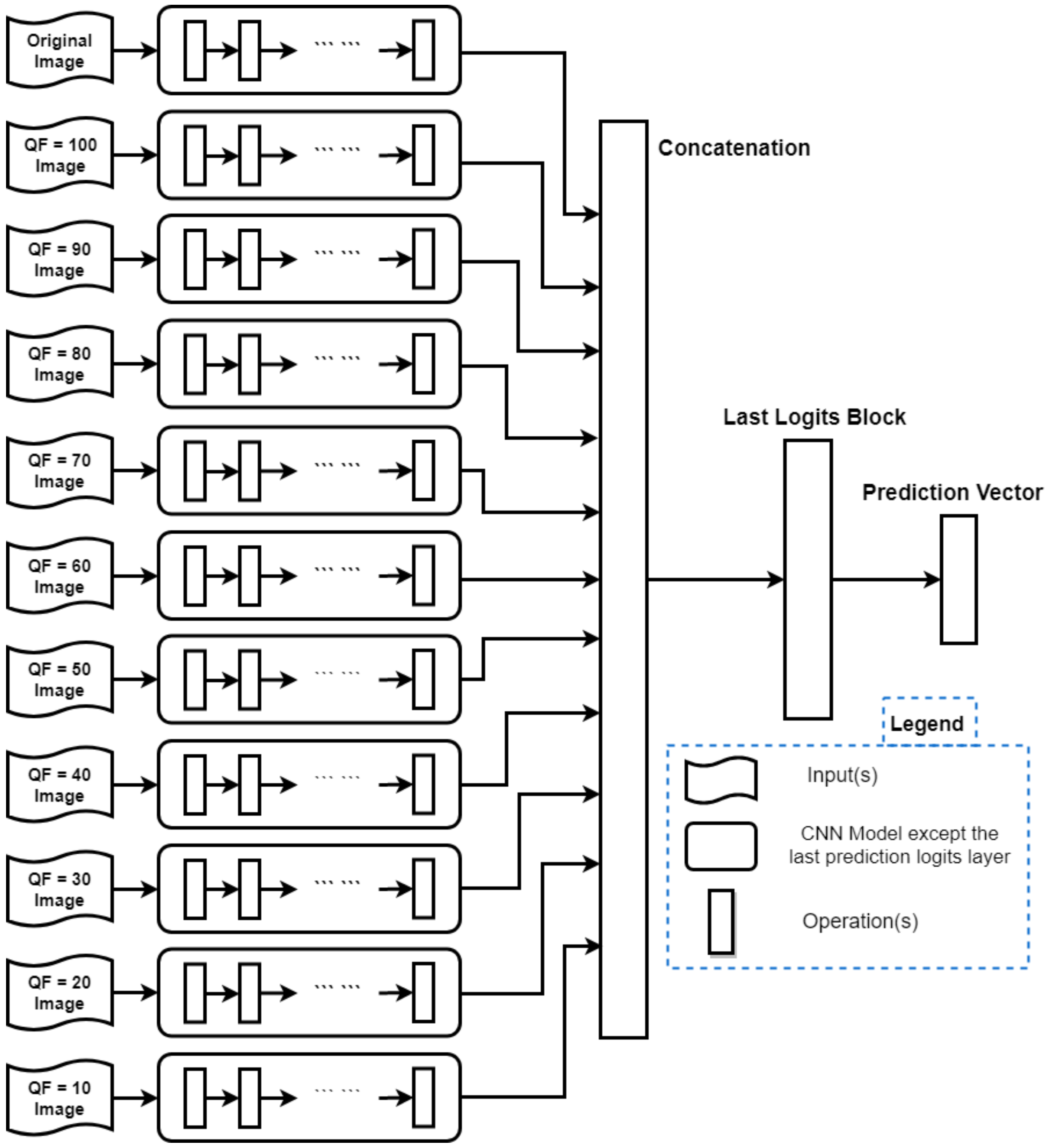


Figure 3.5: New CNN topology.

Chapter 4

Experiment and Analysis

As designed new CNN topology, we will introduce the experiment details in this chapter. We will introduce the data preparation and GPU utilisation for the best training performance as well as the training and evaluation over the popular image classification dataset: ImageNet ILSVRC-2012 based on both Inception-type and ResNet-type network. Then, we will make result comparison with benchmark and analysis the results.

4.1 Data Preparation Efficiency Contribution

4.1.1 Dense Data Representation

As introduced, we chose ImageNet ILSVRC-2012 as our dataset and transfer learning technique is applied to our training process. Since transfer learning process is applied step by step to get incremental improvement in terms of accuracy performance, the bottleneck values are obtained for both training set and validation set at each block's training. It was observed that at all pivotal node[1], the bottlenecks contain a large portion of zeroes. For each node, we calculated the average portion of zeroes based on 15-20 bottlenecks from Inception V3 from both the training set and the validation set as shown in Table 4.1. Each node represents a certain level of the feature that is extracted and learned by the model. For a certain amount of features, we believe that zeroes mean the input image does not contain those specific features at that level. It is undeniable that every picture contains limit information as well as limited features. Majority of the features might not

[1]The input node for current block's training

be presented in the input, as we could observe in large percentage of zeroes at each pivotal nodes from Inception V3.

Pivotal Node Name	Average Percentage of zeroes
mixed_1	56%
mixed_2	64%
mixed_3	53%
mixed_4	73%
mixed_5	72%
mixed_6	76%
mixed_7	90%
mixed_8	71%
mixed_9	89%

Table 4.1: Average percentage of zeroes in the sample training bottleneck from node mixed_1 to mixed_9, the percentage of zeroes is at least 50%

As a result, dense representation of the sparse bottleneck array is proposed as demonstrated in Algorithm 3. At the end of the sparse to dense transformation, two vectors would be obtained, which stored the non-zero values in order and their corresponding positions in the bottleneck array respectively. Consider an short array with 20 values $[0, 0, 0, 0, 0, 0, 5, 7, 0, 0, 0, 2, 0, 0, 12, 0, 0, 0, 0, 1]$, which totally needs 160 bytes in C++. After transformed to dense, what we stored are $value = [5, 7, 2, 12, 1]$ and $position = [6, 7, 11, 14, 19]$, which totally need 80 bytes in C++. By this algorithm, the storage to save the bottleneck values can dramatically be decreased. The read and write process can be more efficient, especially in the situation of 1.2 million training images and 50,000 validation images with 10 quality factor version of each (totally $(1.2M + 50K) \times 11 \simeq 14M$ images).

4.1.2 TFRecord Data Storage Type

TensorFlow organisation initially recommended to save the bottleneck values one by one as *.txt* file for transfer learning [23]. However, the training process usually needs a batch of data to process together. Opening several *.txt* files together is not efficient and makes the training time longer. In 2015, TensorFlow organisation introduced an improvement update in the architecture while a not well-known component is a new file format called TFRecord

Algorithm 3 Bottleneck Values Sparse to Dense Representation

```
1: procedure SPARSE2DENSE(Bottleneck vector b)
2:     $n \leftarrow$ bottleneck size - 1
3:     $value \leftarrow \{\}$                              ▷ Non-zero values
4:     $position \leftarrow \{\}$                           ▷ Non-zero positions
5:     $counter \leftarrow 0$                               ▷ Iteration counter
6:     while $counter \neq n$ do
7:         if $b[counter] \neq 0$ then
8:             $value \leftarrow \{value, b[counter]\}$
9:             $position \leftarrow \{position, counter\}$
10:        $counter \leftarrow counter + 1$
```

[25]. The key of the TFRecord file format is that the data and information are all stored in binary. With a binary file format, the data is serialised and the reading process (which usually happens during training) could become significantly efficient especially when the dataset is extremely large. For a large dataset, we do not want the data fully stored in the memory. TFRecord also gives a feature to let the data to load partially (e.g. batch processing during training) and then process. In other word, data in TFRecord could be called as required, no need to be always in ready condition. In addition, binary data is able to save the storage [12] [25].

What's more, TFRecord file format is designed to work more perfectly with the data pre-processing and pipeline building function inside TensorFlow official framework library. TFRecord stores the data in sequential style such as the bottleneck values in our experiment [25] which make the programming easier to extract the necessary information from the TFRecord file(s).

The recommendation size for each TFRecord file is around 100 - 200 MB each and should not be too large [25]. We decided to transform the training set bottleneck at each node to 10240 TFRecord files and 2560 files for the validation set. Table 4.2 summarises the approximately TFRecord sizes for each Mixed nodes that we trained on.

Pivotal Node Name	**Approx. TFRecord Size for Training set**	**Approx. TFRecord Size for Validation set**
mixed_8	230MB	75MB
mixed_9	230MB	75MB
mixed_10	145MB	42MB

Table 4.2: Approximate TFRecord file size for the Mixed node that we trained on. All files' size is not too large. The size of TFRecord file size is also proportional to the bottleneck size.

4.2 Graphic Processing Unit (GPU) Utilisation

4.2.1 Multi-GPU Utilisation

The most common unit in a computer is the Computation Process Unit(CPU). The properties of the CPU that it has few complex cores and is designed and optimised for a single thread process makes it not the best choice for machine learning, especially the training process. During training, either forward propagation or backward propagation, there is a lot of matrix computation such as matrix multiplication and convolution. For large matrix, the performance of the training is going to be poor if calculation is done in single-thread style.

On the other hand, GPU mainly utilised to process graphic data such as images and videos as its name stated. It has many complex cores and thousands of hundreds of concurrent hardware-based threads. With GPU, a single calculation inside matrix multiplication can be processed in parallel which can improve the performance significantly.

To further improve the performance, multiple GPUs (in our case is 2) are utilised during training and evaluation process. For each training step, the current batch of training data is evenly divided by two. Then, those two groups of data perform forward propagation to calculate the loss and backward propagation to update the parameters independently. By this, the performance during calculation could be even better than single GPU as more calculations could be done in parallel.

4.2.2 Pipelining Configuration

After implementing two GPUs' training settings, it was observed that the GPUs' utilisation is not high during training. We believed that the GPUs were not performing their best. After several testings, it was found out that as GPUs perform fast during training and evaluation, they are always waiting for CPUs to finish preparing the data as the data must be prepared through CPUs. The CPUs' performance, again, becomes the bottleneck of the performance of the training and evaluation. Usually, before the data is sent into the model to perform training with GPUs, it must be extracted and transformed firstly [22]. A normal configuration makes the data preparation and training happen one after another. In another word, when CPUs are preparing the data, the GPUs are idle and doing nothing but waiting for CPUs to finish their work. This is obviously not efficient at all. In order to overcome this problem, the pipelining was developed to make the training step and the pre-processing the data taking place simultaneously. For example, when the GPUs are doing the training step at step k, the CPUs are preparing the training data for the step $k+1$ at the same time. This realises parallel processing in the perspective of task location. The time consumption by with and without pipelining configuration could clearly be found in Fig. 4.1.

4.2.3 Queue Runner

Queues are important TensorFlow objects that allow the machine to compute tensors asynchronously using multiple threads [24]. Consider we have a large dataset to process, such as the training set data from ImageNet ILSVRC-2012, multi-threading and asynchronous operations could speed the data processing up to another level. As discussed, the common performance bottleneck during training is that GPUs are always waiting for CPUs to finish preparing the data. Even with the help from the pipelining, there are still idle time for GPUs. By utilising the queue, the preparation of the data could be more efficient. While one thread is preparing the data and push them into a queue, another thread is doing the same thing and push its data to another queue. The mechanism called *Queue Runner* provided by TensorFlow will manage the queues properly to enable the queues to be able to stop simultaneously and capturing errors.

In order to demonstrate the power of the three techniques could have, we did a small pre-test by applying them on the official TensorFlow transfer learning tutorial posted file and comparing the performance [14]. This experiment was done in Windows 8.1 with CPU Intel(R) Core(TM) i7-5500U and GPU GeForce 940M. It could be found out around 78%

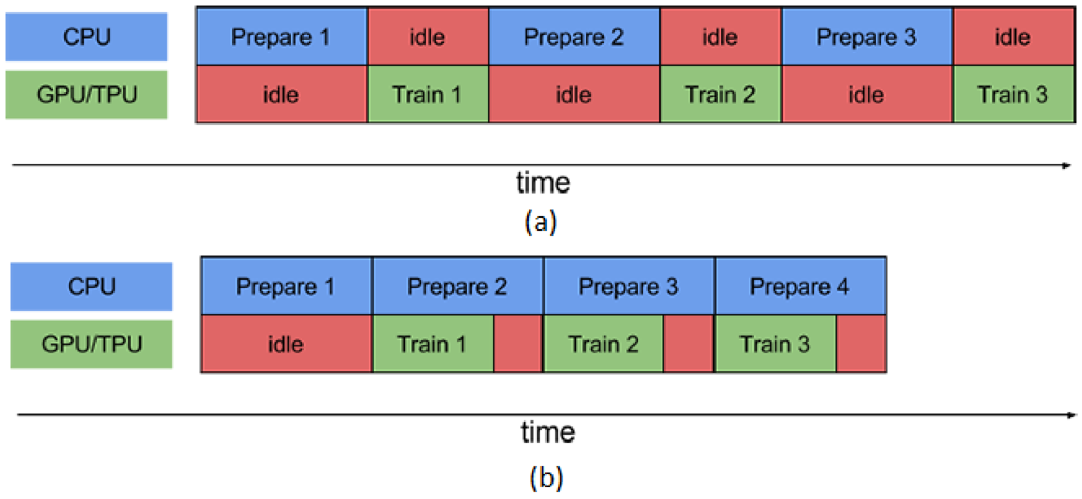


Figure 4.1: (a) shows the timeline without pipelining configuration, the GPUs are always waiting for the CPUs to finish the work; (b) shows the timeline with pipelining configuration, even there is still idle time, the idle time dramatically reduced. The reason that the GPUs always have idel time is due to the fact that the processing time for GPUs are much shorter than CPUs. [22]

of the time could be saved with them which would be even more obvious when the queue runner is applied during training the large ImageNet ILSVRC-2012 dataset.

Item	Time Consumed
Default retrain.py	728 seconds
Applied 3 techniques	**163 seconds**

Table 4.3: Time consumption comparison before and after applying TFRecord, Pipelining and Queue Runner.

4.3 Full Results

4.3.1 Experiment on *Inception V3* Model

Our training experiment on the new designed CNN topology started from applying on Inception V3 model as Inception is one of the most popular image classification CNN blocks. The reason why we chose Inception V3 rather than Inception ResNet v2, which is the newest Inception Network version for Inception network's experiment were that

- Utilising Inception ResNet v2 might not keep pure Inception as our experiment target. If the experiment results show positive sign, whether this CNN topology is suitable for Inception is still unknown. It might be benefit from ResNet.
- Inception ResNet v2 contains over 150 layers [27]. If we utilise Block-by-block training strategy, as discussed in Chapter 3, the process to training all blocks, even one third of the blocks, will be long and the requirements of the computation resources will be large, as per the limitation of the new CNN topology.

Four blocks were trained during the experiment which include the blocks with output nodes *Fully Connected, mixed_10, mixed_9, mixed_8* due to the computation resources' limitation. The architecture details could be found in Figure A.2. While processing training at each block stage, adaptive learning rate was applied as described in Algorithm 2.

The Top-1 accuracy performance is demonstrated in Table 4.4. After training the block with output node mixed_8, a gain of 0.374% in Top-1 accuracy was obtained, which was equivalent to 183 pictures improvements among 50,000 validation images. The overall Top-1 accuracy trend of this training experiment is drawn in Figure 4.2. The dashed line in the figure shows the linear regression of the accuracy trend. We believed that the performance would have a chance to be better if the training could be performed further to earlier blocks. The best Top-5 accuracy that we are able to reach during the experiment is 94.176% at mixed_9 stage which is 0.346% improvement equivalent to 173 images.

Block Trained with Node Name	Top-1 Accuracy	Corresponding Top-5 Accuracy
Default Model	77.59%	93.83%
Fully Connected	77.77%	93.880%
mixed_10	77.896%	94.094%
mixed_9	77.936%	94.11%
mixed_8	**77.964%**	94.148%

Table 4.4: Top-1 Performance for last 4 blocks' training on Inception V3 Model.

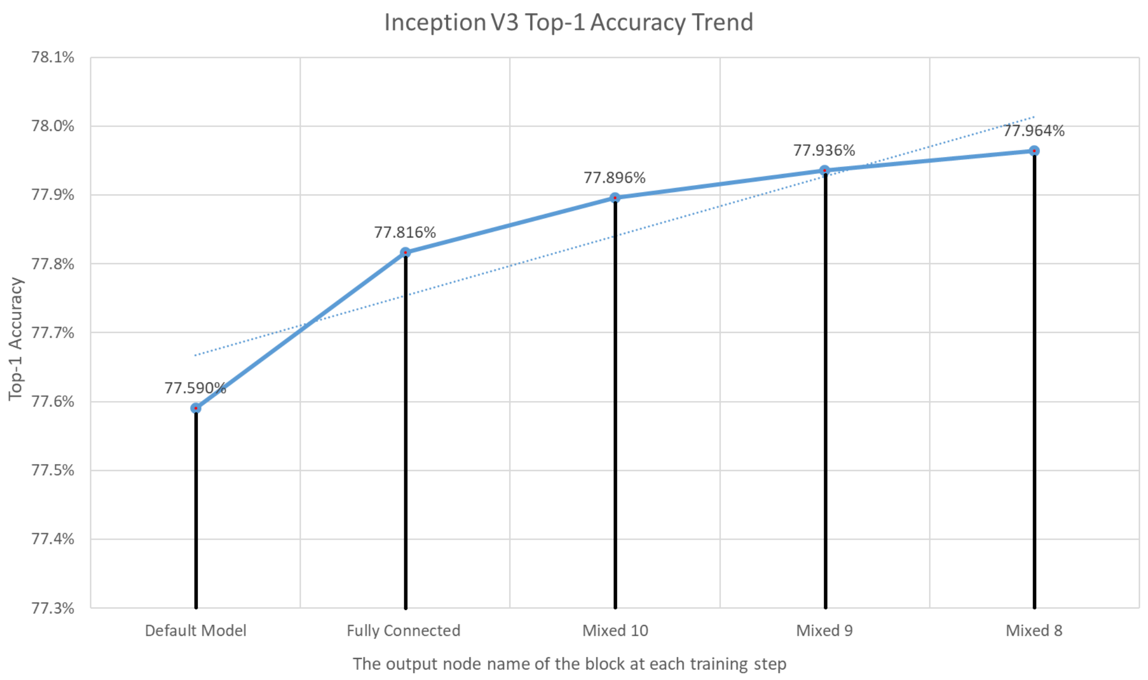


Figure 4.2: Inception V3 Top-1 accuracy trend after training the first 4 blocks.

Block Trained with Node Name	Top-5 Accuracy	Corresponding Top-1 Accuracy
Default Model	93.83%	77.59%
Fully Connected	93.936%	77.816%
Mixed_10	94.13%	77.874%
Mixed_9	**94.176%**	77.856%
Mixed_8	94.158%	77.932%

Table 4.5: Top-5 Performance for last 4 blocks' training on Inception V3 Model.

4.3.2 Experiment on *ResNet-50 V2* Model

As in the experiment of *Inception V3*, the accuracy in both Top-1 and Top-5 perspective could yield improvement in the new CNN topology when applying the Block-by-block transfer learning method. In order to prove the generalisation of this topology of CNN, we also experimented on one of the Residual networks called ResNet-50 V2 (architecture detail

Block Trained	Top-1 Accuracy	Corresponding Top-5 Accuracy
Default Model	75.588%	92.828%
Logits Block	75.852%	93.028%
Block 4	**75.978%**	93.012%

Table 4.6: Top-1 Performance for last 2 blocks' training on ResNet-50 V2 Model.

Block Trained	Top-5 Accuracy	Corresponding Top-1 Accuracy
Default Model	92.828%	75.588%
Logits Block	93.034%	75.788%
Block 4	**93.056%**	75.866%

Table 4.7: Top-5 Performance for last 2 blocks' training on ResNet-50 V2 Model.

could be found in Figure A.3. To be consistent, the same manual learning rate adjustment was applied for this experiment.

In the training experiment on ResNet-50 V2, two blocks were trained inclde the last logits block and the fourth block because of resources' limitation. The Top-1 accuracy results are shown in Table 4.6. The best Top-1 accuracy we were able to obtain was 75.978%, which has 0.39% improvement comparing to the original model benchmark. This improvement means we get 195 more images to be correct in rank #1 in the ImageNet ILSVRC-2012 validation set. The corresponding Top-1 accuracy result trend is shown in Figure 4.3. This trend is similar to the one we drew for the Inception V3 training experiment. It is expected if keep training earlier blocks with enough resources, the result is able to keep climbing up. Moreover, in the Top-5 accuracy perspective, the highest the training process we can reach is 93.056% while training Block 4, which is a 0.228% improvement.

4.3.3 Experiment Results Conclusion

The new CNN topology applies on both Inception network (Inception V3) and Residual network (ResNet-50 V2) can both get over 0.3% Top-1 accuracy improvement and relatively lower improvement in Top-5 accuracy perspective compare to original model's benchmark

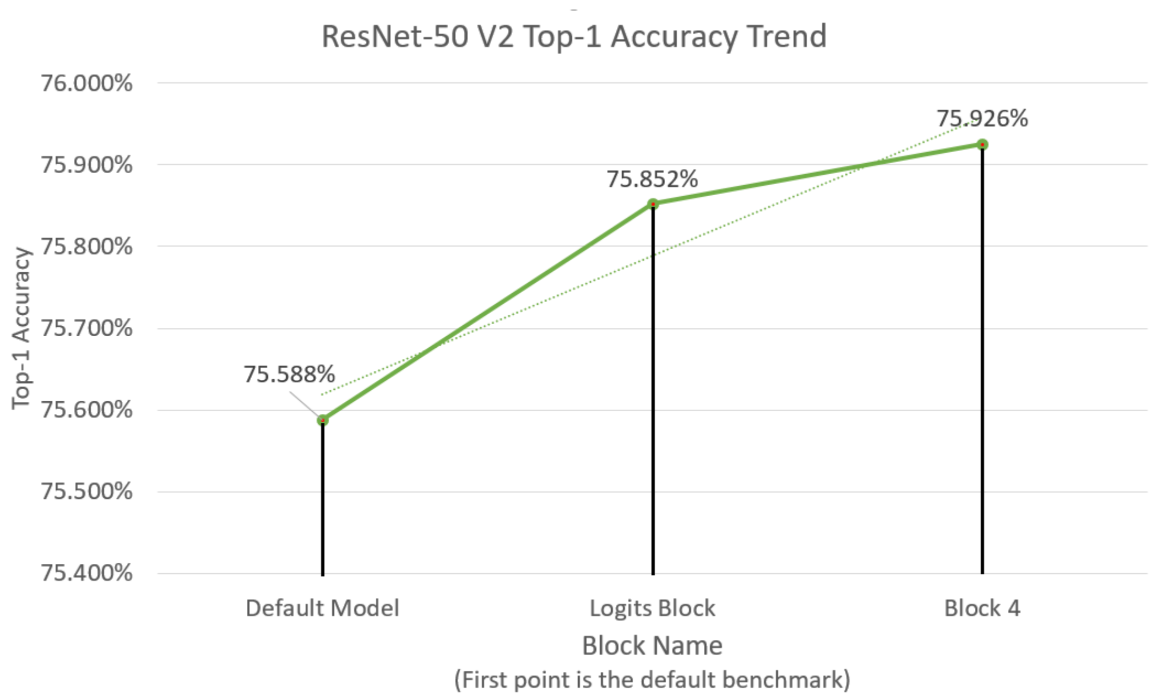


Figure 4.3: ResNet-50 V2 Top-1 accuracy trend after training the first 2 blocks.

after training the last few tail blocks. By this, we are able to conclude that the new CNN topology that digest compressed version of original inputs could further improve the accuracy performance compare to the original model. In addition, the compressed inputs contain more or less useful information for the machine to make better predictions in the task of image classification.

4.4 Analysis and Observations on the Results

We have observed statistically that the compressed images may sometimes help the neural network to make better predictions. In order to better understand why the compressed version images help, we give insights for this phenomenon by analyzing the image examples visually. To begin, we classify all images into three types:

- **Type A**: Images that our trained model topology predicts the ground truth label at rank #1 while the original default model fails to do that;

- **Type B**: Images that the original default model predicts the ground truth label at rank #1 while our trained model topology fails to do that;
- **Type C**: Images other than Type A and Type B images.

Then, we applied one of the models that we obtained: the trained model topology after training the Fully Connected Layer in terms of the Inception V3 model (with Top 1 accuracy 77.77%), to classify the images to those three types. The number of images difference between Type A and Type B, as demonstrated in Table 4.8, is $1,163-1,073 = 90$ images, which is $90/50,000 = 0.18\%$ of the whole validation set, which is the improvement we got during this training: $77.77\% - 77.59\% = 0.18\%$

	Number of images
Type A	1,163
Type B	1,073

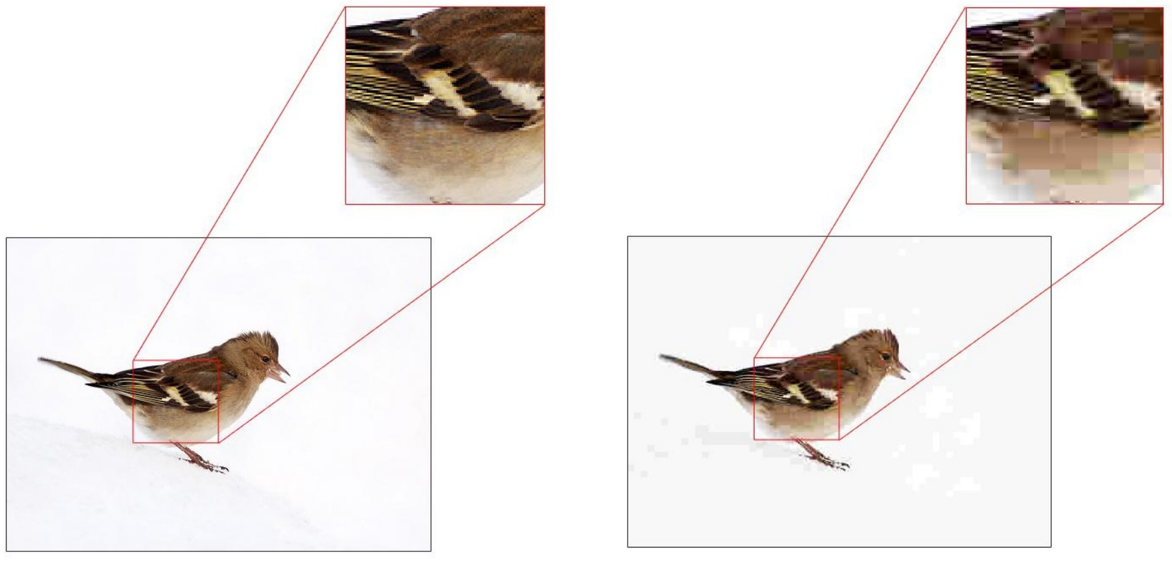
Table 4.8: Number of Type A and Type B images when comparing the CNN topology after training Fully Connected block and the original default model

In order to understand why compressed version images input to the training help, we visualised 80 images from **Type A** to compare the original input and its compressed version with the best performance independently. Then, we understand them in feature extraction's perspective why the compression inputs help the training results. The pictures in Type A can be categorised into two sub-types: a) confusing labels: there are labels from pre-defined ground truth list that is similar to each other, the machine makes the wrong prediction because they select another similar label as prediction; b) multiple objects: there are cases that the images contain more than one object, the machine will make prediction based on all of them, which one becomes rank #1 prediction is usually unknown. As a result, ImageNet image classification challenge introduced Top-5 accuracy to avoid multi-object confusion. Based on the observed images, 63 out of 80 images belong to confusing label categories and 17 of them belong to multiple objects and no other situation happens. We will observe typical images which could represent each of these two sub-types separately and conclude why compression may help in these two categories.

Confusing Label

Starting with confusing label condition, the image with ground truth "brambling, Fringilla montifringilla", as shown in Figure 4.4. Consider those two images independently, the confidence on the ground truth label based on Inception V3 is only 37.03% and the rank of the ground truth label is not #1 for the initial original image. Inception V3 model yields "junco, snowbird" as the #1 prediction with confidence 38.14%. Some juncos do have a similar color to the brambling as shown in Figure 4.5, but the key difference between these two types of birds is the texture difference on the wings. It could be found out that brambling contains more complicated textures while junco's wings are smoother. Table 4.9 shows the detail of the accuracy difference between the original image and its compressed version with QF = 10. The compressed version, even with a relatively low quality presents a higher prediction performance with 71.67% confidence on the ground truth.

By visualizing the difference between the original image and its QF = 10 version, the block effect caused by the compression, as shown in Figure 4.4b, is able to enhance the texture effects on the wings. The extra intense textures are generated from the edge of the blocks caused by compression. With these textures, the machine is more confidence to categorise this bird to brambling rather than junco due to the fact that the neural network relies upon the key edge information.



(a) (b)

Figure 4.4: (a) Original #651 image from validation set with ground truth "brambling, Fringilla montifringilla"; (b) Compressed Version of (a) with Quality Factor = 10

Other than that, another typical example of the confusing label in Type A is #898 image with ground truth label "lynx, catamount" from the validation set as shown in

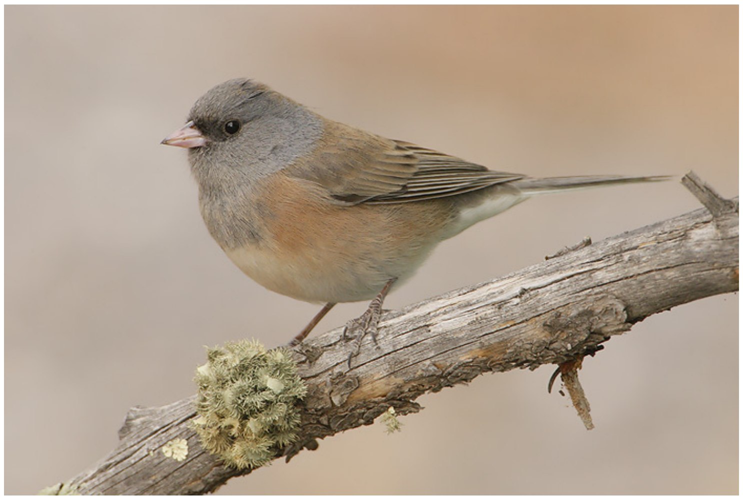


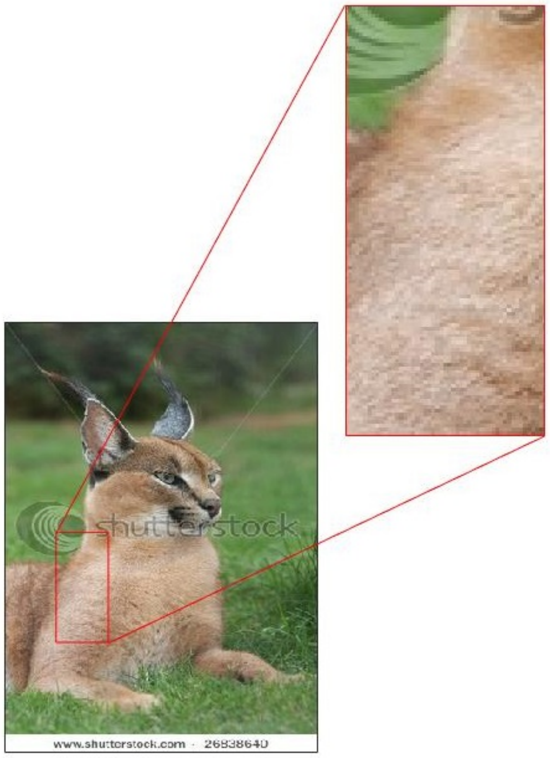
Figure 4.5: Example image with ground truth "junco, snowbird" from the training set of ImageNet ILSVRC-2012.

Label	Original Image		Quality Factor = 10	
	Rank	Accuracy	Rank	Accuracy
brambling, Fringilla montifringilla	2	37.03%	1	71.67%
junco, snowbird	1	38.14%	2	12.03%

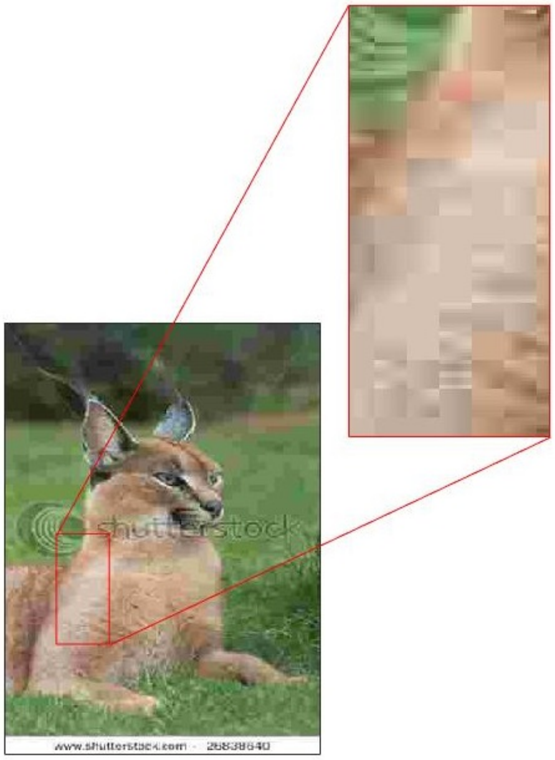
Table 4.9: Accuracy comparsion between ground truth label "brambling, Fringilla montifringilla" and confusion label junco, snowbird on image #651 in the validation set of ImageNet ILSVRC-2012.

Figure 4.7a. Again, we consider the original image and its QF = 10 version individually. With the original image, the Inception V3 only has 13.15% confidence on the ground truth label while has strong confidence on another label "gazelle". Consider one "gazelle" images from the training that demonstrated in Figure 4.8, the main difference between these two types of object is a gazelle MAY has longhorns on the head while lynx must not have them if we do not take body features into considerations. Other than that, the body of lynx is usually not as smooth as gazelle. From Table 4.10, it could be found out that the QF = 10 version performance much better than the original input with 94.23% confidence on the ground truth label and its confusing label "gazelle" only has 0.0898% confidence, which is very low.

Comparing the original input and its low-quality version side by side visually, the contrast of the lynx's body in QF = 10 version is higher as shown in Figure 4.6. By this, the lynx object is better highlighted. In ground truth's perspective, it could be found out that the long ears part from the lynx is integrated into the background which will no longer be misunderstood as longhorns as circled in Figure 4.7b thanks to the block effect caused by the compression. In addition, the block effect introduces extra textures onto the body of the lynx which further differentiate it from the gazelle even though their body color is close to each other.



(a)



(b)

Figure 4.6: (a) Original #898 image from validation set with ground truth "lynx, catamount"; (b) Compressed Version of (a) with Quality Factor = 10. Both with body part zoomed in the compare the contrast.

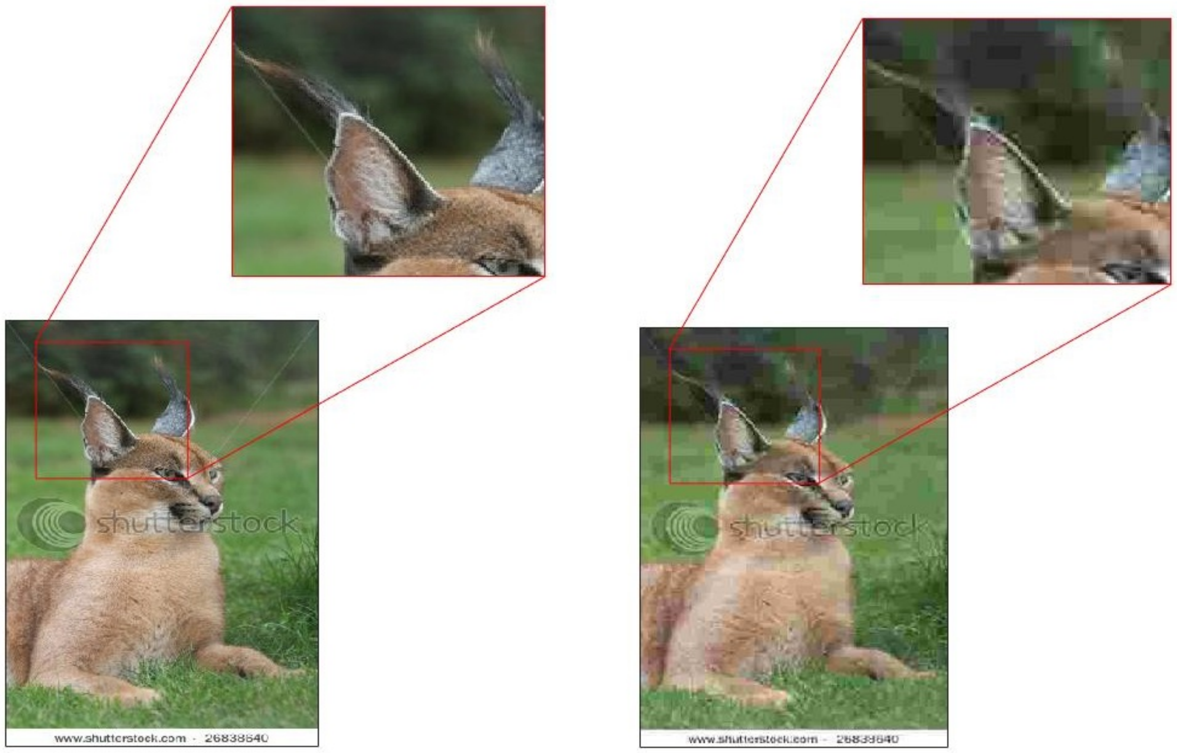

(a) (b)

Figure 4.7: (a) Original #898 image from validation set with ground truth "lynx, catamount"; (b) Compressed Version of (a) with Quality Factor = 10. Both with ears' part zoomed in.

Multiple Objects

In terms of multiple objects condition, let us consider #568 image in the validation set of ImageNet ILSVRC-2012 in Figure 4.9a. From the image, we could see multiple things in the image including the mountain, the skiers, trees and so on. The ground truth given by the dataset is "ski" and the interference object has a label "alp", which is the mountain in the background. The original image has only confidence of 32.99% on the ground truth while has 42.51% confidence on the interference alp by Inception V3. Even though both confidences is not high, it is clear to show that the alp successfully influences the judgment by the machine to select the "ski" as its Top-1 prediction. On the other hand, the compression version of this image with a QF = 10, the ground truth probability increases

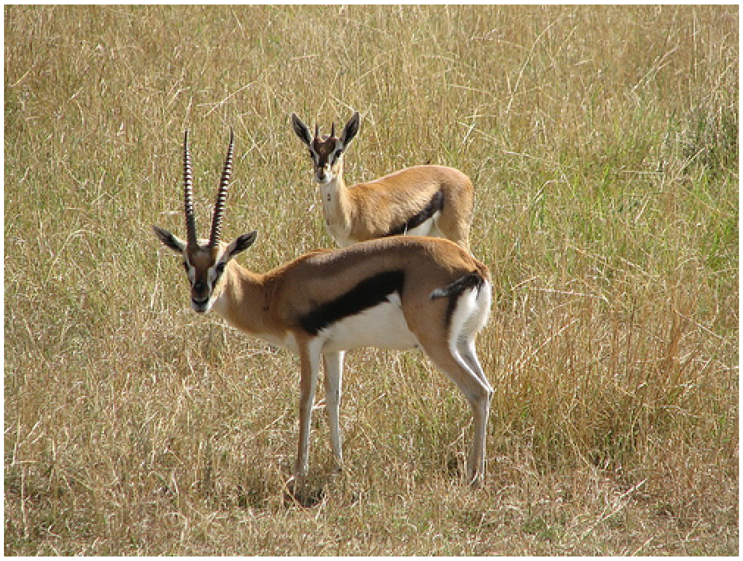


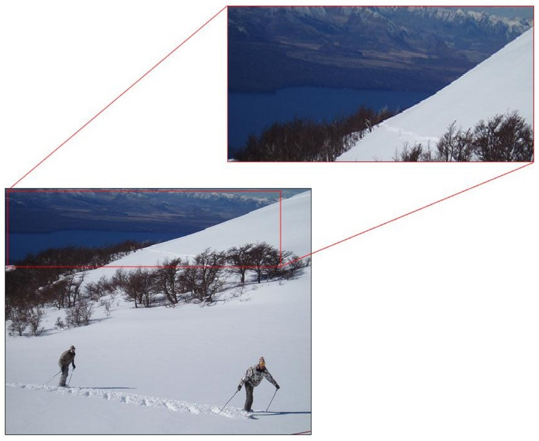
Figure 4.8: Example image with ground truth ”gazelle” from the training set of ImageNet ILSVRC-2012.

Label	Original Image		Quality Factor = 10	
	Rank	Accuracy	Rank	Accuracy
lynx, catamount	2	13.15%	1	94.23%
gazelle	1	38.62%	4	0.0898%

Table 4.10: Accuracy comparsion between ground truth label ”lynx, catamount” and confusion label gazelle on image #898 in the validation set of ImageNet ILSVRC-2012.

to 89.64% while the interference label ”alp” has only 8.70% confidence. The detailed comparison of the accuracy is summarised in Table 4.11.

First of all, the outline of skiers are not heavily distorted which does not seriously influence the judgment. By this, the skiers' part in the image becomes more prominent rather than relatively equivalent importance with the alp in the image, which makes the machine output much higher confidence on the ground truth object. On the other hand, the alp part of the image, which is squared is Figure 4.9, can easily be observed in the original image. However, after compressed with QF = 10, the details of the alp are ruined by the block effect and can hardly be distinguished in the image. In addition, with the compression from JPEG, the contrast of skiers' background is further increased which helps to highlight the skiers and make the skiers more obvious. This further help to have a better


(a)

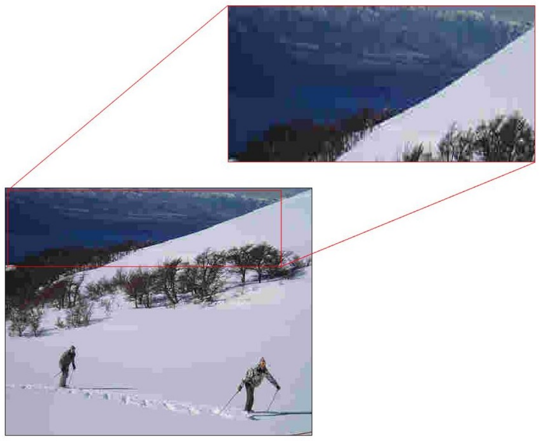

(b)

Figure 4.9: (a) Original #568 image from validation set with ground truth ”ski”; (b) Compressed Version of (a) with Quality Factor = 10.

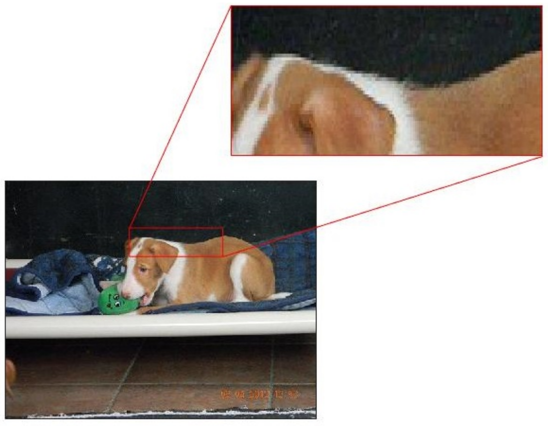
Label	Original Image		Quality Factor = 10	
	Rank	Accuracy	Rank	Accuracy
ski	2	32.99%	1	89.64%
alp	1	42.51%	2	8.70%

Table 4.11: Accuracy comparison between ground truth label ”ski” and interference object label alp on image #568 in the validation set of ImageNet ILSVRC-2012.

prediction.

Another multiple object example is shown earlier in the introduction Chapter in Figure 1.1. It might be unbelievable that this image contains multiple objects as the dog is obvious enough. Based on Inception V3, there are two other labels that influence the right judgment: ”tub, vat” and ”bathtub, bathing tub, bath, tub” while the ground truth label is ”Ibizan hound, Ibizan Podenco”. By observing the training set images with label ”tub, vat” and ”bathtub, bathing tub, bath, tub”, it was found out that the key feature with two labels is white SMOOTH surface while some images show only the edge of the tub, some with the whole tub, some contain people or objects and some are empty. It is worth to note that in ImageNet ILSVRC-2012 dataset, multiple labels may target the same object which makes the machine even human hard to distinguish. Figure 4.12 shows an example image with label ”tub, vat”. The accuracy based on Inception V3 upon those three labels is shown in Table 4.12.

As shown in Figure 4.10, the block effect that in the background decreases the contrast of the background around the outline of the dog which helps to increase the contrast of the dog. The dot-like noise in the background is also removed entirely by the block effect which helps to highlight the dog. Additionally, the smooth white surface is the key that makes the machine believe it is a tub, the block effect generated by JPEG compression make the white surface shown in Figure 4.11b no longer smooth. Meanwhile, the dog itself is still easy to see and the block effect even further supplements extra textures on the dog's body. By this, the machine is able to give the right prediction on the ground truth "Ibizan hound, Ibizan Podenco".


(a)

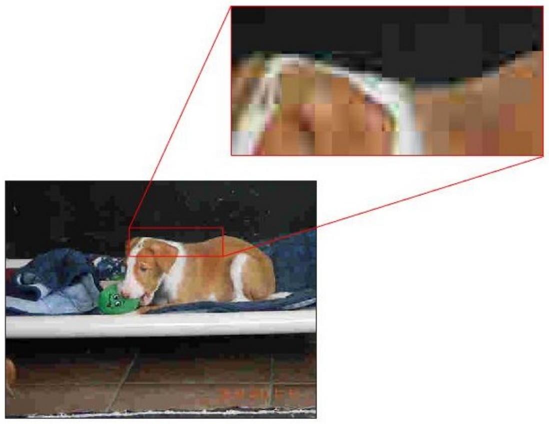
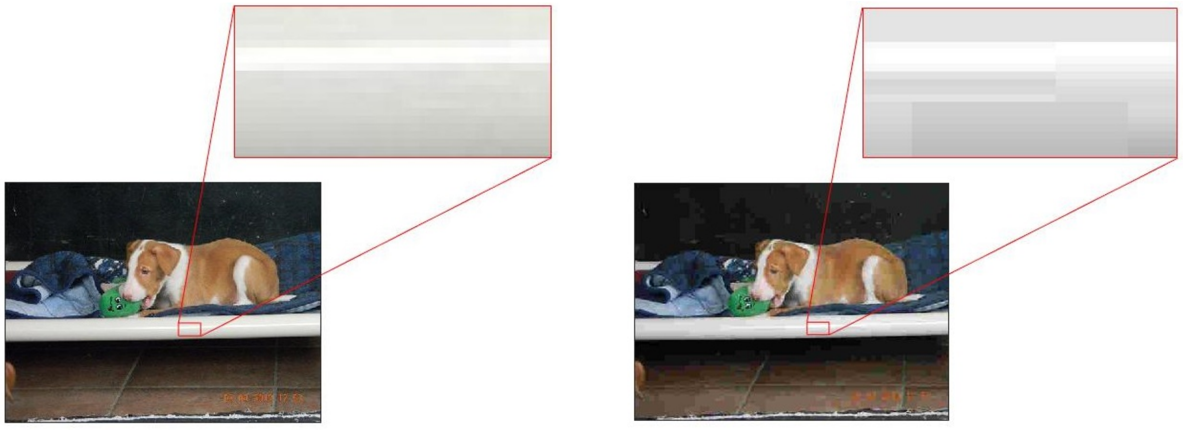

(b)

Figure 4.10: (a) Original #115 image from validation set with ground truth "Ibizan hound, Ibizan Podenco"; (b) Compressed Version of (a) with Quality Factor = 10. Both with white surface part zoomed in. Both with part of the dog's outline zoomed in.

Summary

Overall, from the observations on the examples, we could say compression **MAY** help the machine to improve the accuracy performance by a) utilising the block effect to highlight and intensify the texture and edge features under the condition that the texture features are crucial for making correct decision b) increasing the contrast of the image to highlight the main object and c) utilising the block effect to remove distracting parts in the image in order to highlight the core object. However, the condition to make the compression



(a) (b)

Figure 4.11: (a) Original #115 image from validation set with ground truth ”Ibizan hound, Ibizan Podenco”; (b) Compressed Version of (a) with Quality Factor = 10. Both with white surface part zoomed in.

Figure 4.12: Example image with ground truth ”tub, vat” from the training set of ImageNet ILSVRC-2012.

Label	Original Image		Quality Factor = 10	
	Rank	Accuracy	Rank	Accuracy
Ibizan hound, Ibizan Podenco	3	9.95%	1	70.64%
tub, vat	1	23.40%	No longer in Top-5	
bathtub, bathing tub, bath, tub	2	14.23%	3	2.02%

Table 4.12: Accuracy comparison between ground truth label "Ibizan hound, Ibizan Podenco" and interference object label tub, vat or "bathtub, bathing tub, bath, tub" on image #115 in the validation set of ImageNet ILSVRC-2012.

becomes helpful is strict. For example, if the object in the image occupies a relatively small portion of the image, the block effect caused by compression will ruin the object and make the machine making a wrong prediction at the end. As a result, the condition when the compression could help and how compression can be applied to help the machine to make the right and better prediction are still interesting to be discovered further.

Chapter 5

Conclusion and Future Works

5.1 Conclusion

In this thesis, we have proposed a new CNN topology which is able to input multiple compressed versions of original images along with original image. With this topology, the network is able to digest original and compressed information together to have better accuracy performance in both Top-1 and Top-5 perspective. In order to see the incremental improvement for this topology, a Block-by-block transfer learning method was proposed and applied during our experiments. Instead of training the whole topology, Block-by-block style transfer learning could catch the upper limit of computational resources and does not overwhelm the resources initially by this big topology. Our proposed topology is generalised to be applied to popular image classification CNN networks such *Inception network and Residual network*. In terms of Inception, Inception V3 was selected for our experiment. We are able to get 0.374% improvement in Top-1 accuracy and 0.346% improvement in Top-5 accuracy comparing to the official benchmark from TensorFlow after training over four pivotal blocks including the last fully connected layer. On the other hand, ResNet-50 V2 was chosen for the ResNet experiment. Furthermore, an overall 0.39% improvement in Top-1 accuracy and 0.228% improvement in Top-5 accuracy was achieved compared to the original benchmark. However, this topology does have a limitation that it has 11 times the number of parameters in order to absorb different compressed version of images to input to the model. With this topology, we can use the non-linearity at the input stage from the compression to enhance the accuracy performance in the task of image classification. Also, it was found out by the visualisation of improved images that the compression more or less plays a role as noise remover, smoothing the image content and highlighting the major object's contrast, which help to make better predictions.

5.2 Future Works

5.2.1 Efficient Resource Utilisation Improvement

The limitation of our new CNN topology, as discussed in Section 3.5, is the number of parameters and the width of the model is 11 times of the original model. The design of this topology does not avoid this in order to fully digest compressed information with different compression quality. If it is possible to design an algorithm to do early fusion of the compressed and original info and generate an input with only the original image input size meanwhile maintain the useful information from the compressed images and original image, the size of the topology could be reduced and also gives similar improvements that our proposed topology does. Inputting lossy inputs to the machine, which may help the network's performance, not only proved in our new topology, but also by *Cutout* strategy proposed by T. Devries and G. W. Taylor in 2017. With this strategy, they described a way called *Cutout* that randomly masking out square regions of input during training through data augmentation [6]. With this method, they were able to obtain further accuracy improvements. Similar to masking certain area of the input, Yang and others designed a content-adaptive compression method named *Transparent Composite Model* (TCM). This method can adjust the level of compression at each block in the image depends on the energy (or whether these blocks contains edge(s)) [31]. TCM majorly does compression heavily on smooth blocks and lightly on non-smooth blocks. Such method works fine in pictures with smooth background and helps to highlight the objects as shown in Figure 5.1. After the TCM operation, the outcome in Figure 5.1b even have better prediction confidence than original image for Inception V3, which is 92.44% confidence on ground truth for TCM output versus 90.24% confidence for original input. Similarly, in Figure 5.2, the smooth body is wiped after applying TCM mask and the teeth and the nose are highlighted. The Inception V3 outputs 87% confidence on the ground truth label "tucker" comparing to 81.4% on the original image. It could be concluded that the mask applied by TCM is target-oriented on saving high energy part on the image (the part with edges) and mask the smooth part. We suggest applying TCM type mask strategy and apply on the training process which may be more oriented to increase the accuracy performance.

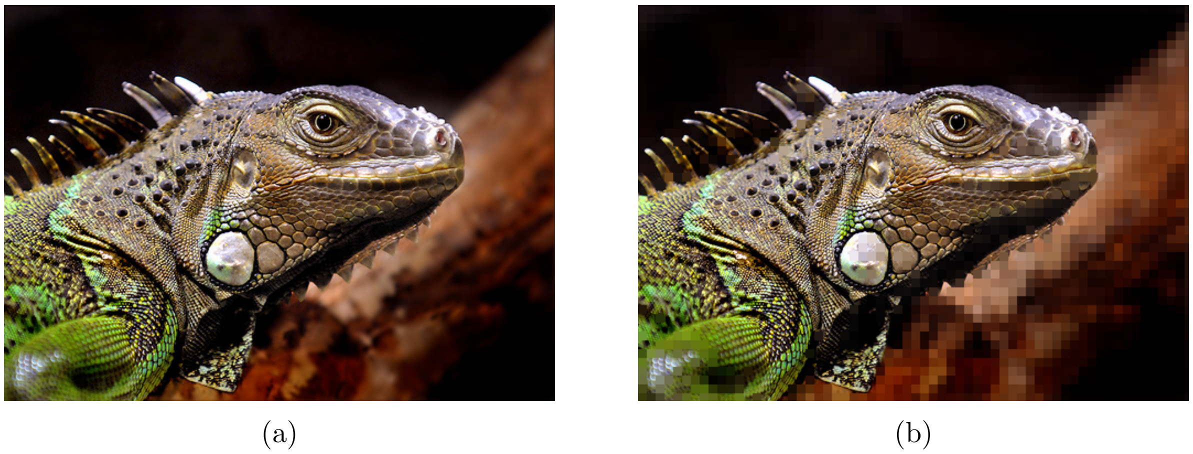


Figure 5.1: (a) Original image #153 with object iguana from ImageNet ILSVRC-2012 validation set; (b) TCM applied on (a), it could be found out that the object with many edges is successfully reserved and the background is compressed heavily and have serious visible block effect.

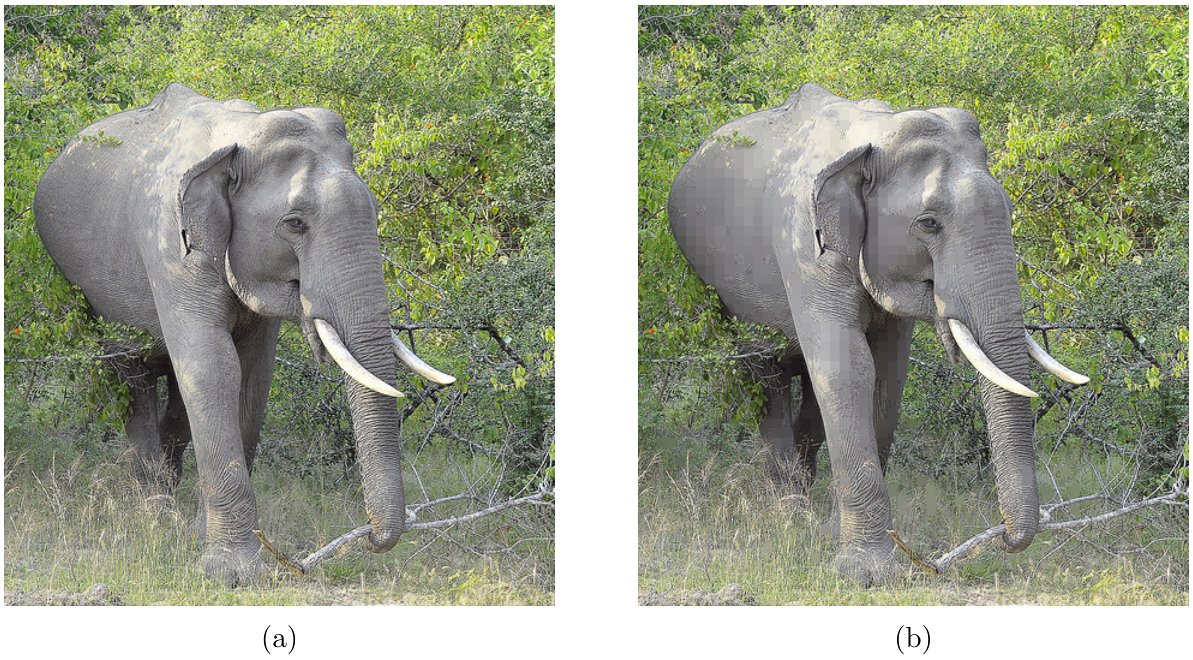


Figure 5.2: (a) Original image #67 with object tusker from ImageNet ILSVRC-2012 validation set; (b) TCM applied on (a), the key for identification, the nose and the teeth is highlighted while masking the smooth body part make the model more confident

5.2.2 Application On Other Models And Machine Learning Tasks

According to Table 2.1, the best performance so far for imageNet is produced by *Efficient-Net*, which can get 84.1% Top-1 accuracy. Compared to the second-best, which is *GPIPE*, only 0.1% improvement was obtained. From the experiment on *Inception V3 and ResNet-50 V2*, the improvement by modifying to model with our new CNN topology can get at least 0.3% improvement in terms of the Top-1 accuracy. We believe that if this topology is applied to the best *EfficientNet* model, it could further improve the accuracy performance.

Besides, our experiment so far is limited to the task of image classification which utilises the dataset from imageNet. In terms of image input, some other tasks such as object detection, object segmentation and so on could be further experimented to guarantee the generalisation of this proposed topology. Also, these applications on video input (which is a sequence of images) could be attempted. A summary of the multiple applications from different perspectives are summarised in Figure 5.3.

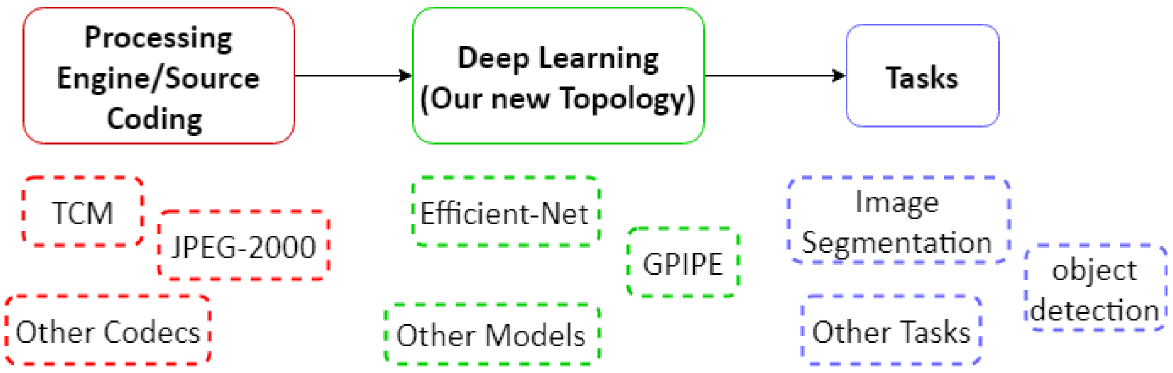


Figure 5.3: Different applications of proposed topology in three perspectives: input source coding method, CNN model and the task scenarios.

5.2.3 Training with Different Compressed Input

The experiments we have done so far, even in the image classification dataset, only use the JPEG compression method. Other types of image compression are worth to be attempted as well such as JPEG-2000, which utilises the wavelet information to do the compression. Overall it could be found out from the experiments that the loss of information from compression does not always lead to low prediction performance, especially in machine's perspective.

References

[1] H. Amer, Y. Jiang, and E. Yang. Image compression helps deep learning. In progress.

[2] Ashutosh Bhown, Soham Mukherjee, Sean Yang, Shubham Chand ak, Irena Fischer-Hwang, Kedar Tatwawadi, Judith Fan, and Tsachy Weissman. Towards improved lossy image compression: Human image reconstruction with public-domain images. *arXiv e-prints*, page arXiv:1810.11137, Oct 2018.

[3] Yangqing Jia Pierre Sermanet Scott E. Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Christian Szegedy, Wei Liu and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[4] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[5] Rémi Cogranne. Determining JPEG Image Standard Quality Factor from the Quantization Tables. *arXiv e-prints*, page arXiv:1802.00992, Feb 2018.

[6] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[7] Samuel F. Dodge and Lina J. Karam. Understanding how image quality affects deep neural networks. *CoRR*, abs/1604.04004, 2016.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[11] imagenet. Imagenet Large Scale Visual Recognition Challenge 2016 (ILSVRC2016). http://image-net.org/challenges/LSVRC/2016/results. [Online; accessed 06-July-2019].

[12] Thomas Gamauf in Mostly AI. Tensorflow Records? What they are and how to use them. https://bit.ly/2n8Wm94, 2018. [Online; accessed 04-July-2019].

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[14] Yanbing Jiang. inception V3 Retrain Fast Input With TFrecord QueueRunner. https://bit.ly/2LgnRM6. [Online; created March-2019].

[15] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[17] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[18] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990.

[19] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. *CoRR*, abs/1409.5185, 2014.

[20] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan. Deepn-jpeg: A deep neural network favorable jpeg-based image compression framework. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2018.

[21] Tensorflow Organization. Benchmark. https://www.tensorflow.org/guide/performance/benchmarks. [Online; accessed 17-July-2019].

[22] Tensorflow Organization. Data Input Pipeline Performance. https://www.tensorflow.org/guide/performance/datasets. [Online; accessed 05-July-2019].

[23] Tensorflow Organization. How to Retrain an Image Classifier for New Categories. https://www.tensorflow.org/hub/tutorials/image_retraining. [Online; accessed 04-July-2019].

[24] Tensorflow Organization. *tf.train.queue$_r$unner.QueueRunner*. https://bit.ly/2Y2pMJW. [Online; accessed 11-July-2019].

[25] Tensorflow Organization. Using TFRecords and tf.Example. https://www.tensorflow.org/tutorials/load_data/tf_records, 2015. [Online; accessed 04-July-2019].

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[27] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[29] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[30] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb 1992.

[31] E. Yang, X. Yu, J. Meng, and C. Sun. Transparent composite model for dct coefficients: Design and analysis. *IEEE Transactions on Image Processing*, 23(3):1303–1316, March 2014.

[32] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

Appendix A

Archtecture of experimented model

A.1 Inception V3

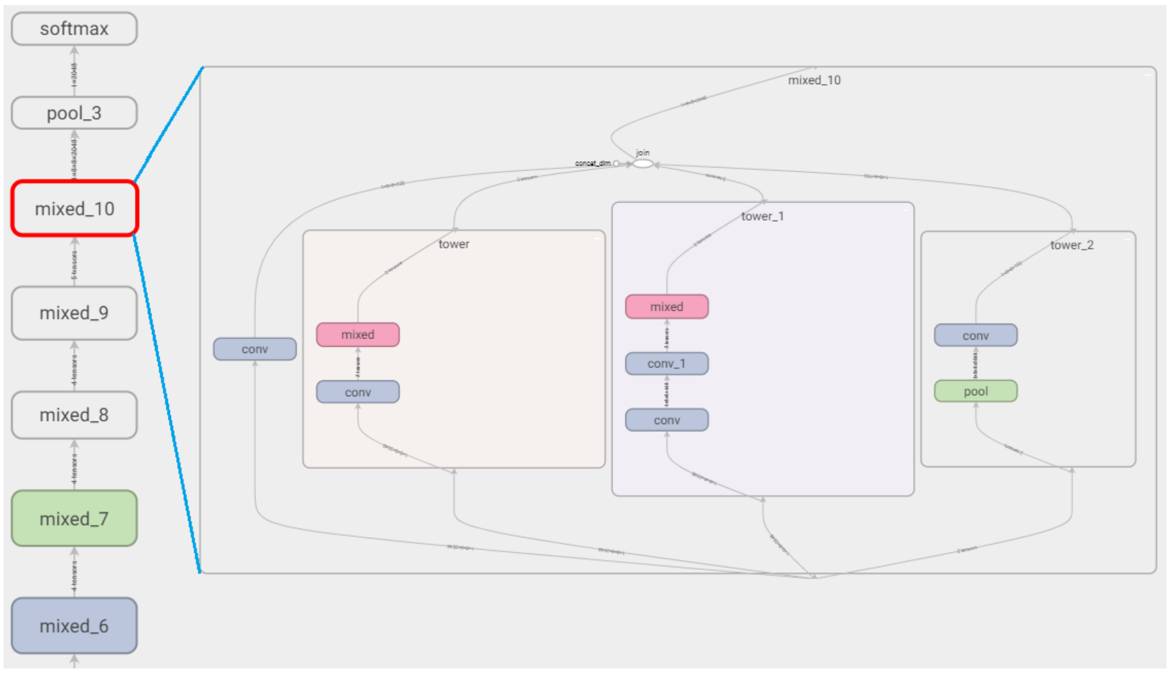


Figure A.1: Later blocks of Inception V3 model with mixed_10 block zoomed in to show the Inception block details.

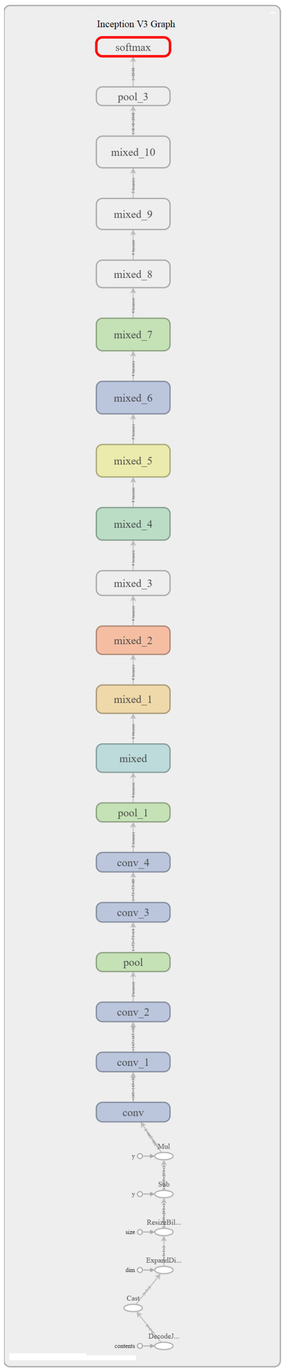


Figure A.2: Overview of Inception V3 model with the name of each block.

A.2 ResNet-50 V2

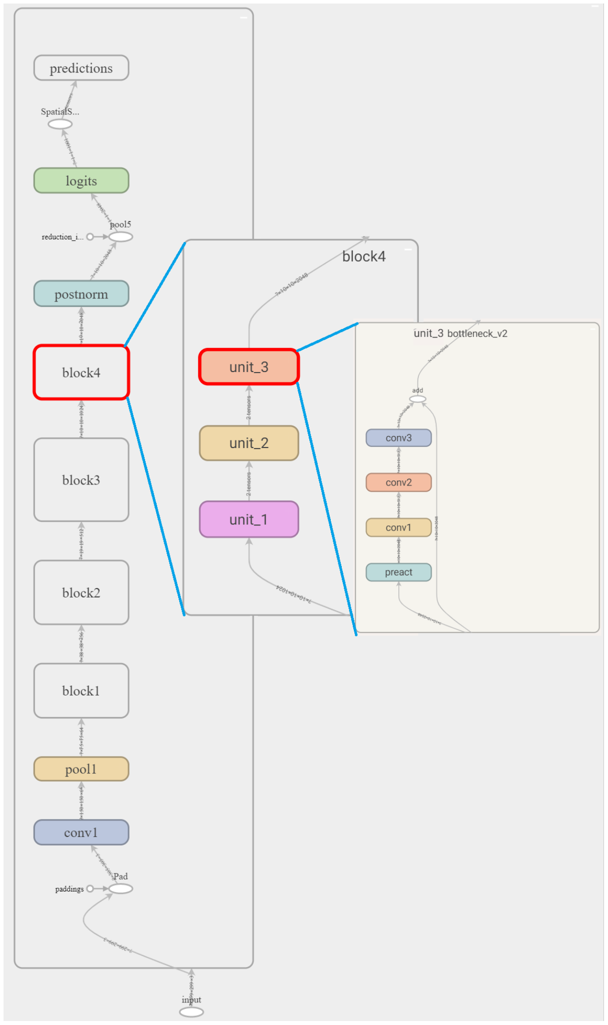


Figure A.3: Overview of Inception V3 model with the name of each block with block 4 zoomed in to show the ResNet block details.