Automatic CNN Channel Selection and Effective Detection on Face and Rotated Aerial Objects.

Zhenyu Fang

Centre for Signal & Image Processing

Electronic & electrical engineering

University of Strathclyde, Glasgow

June 18, 2020

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Winston Churchill

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Abstract

Balancing accuracy and computational cost is a challenging task in computer vision. This is especially true for convolutional neural networks (CNNs), which required far larger scale of processing power than traditional learning algorithms. This thesis is aimed at the development of new CNN structures and loss functions to tackle the unbalanced accuracy-efficiency issue in image classification and object detection, which are two fundamental yet challenging tasks of computer vision. For a CNN based object detector, the main computational cost is caused by the feature extractor (backbone), which has been originally applied to image classification. Optimising the structure of CNN applied to image classification will bring benefits when it is applied to object detection. Although the outputs of detectors may vary across detection tasks, the challenges and the design principles among detectors are similar. Therefore, this thesis will start with face detection (i.e. a single object detection task), which is a significant branch of objection detection and has been widely used in real life. After that, object detection on aerial image will be investigated, which is a more challenging detection task. Specifically, the objectives of this thesis are: 1. Optimising the CNN structures for image classification; 2. Developing a face detector which enables a trade-off between computational cost and accuracy; and 3. Proposing an object detector for aerial images, which suppresses the background noise without damaging the inference efficiency.

For the first target, this thesis aims to automatically optimise the topology of CNNs to generate the structure of fixed-length models, in which unnecessary convolutional kernels are removed. Experimental results have demonstrated that the optimised model can achieve comparable accuracy to the state-of-the-art models, across a broad range of datasets, whilst significantly reducing the number of parameters.

To tackle the unbalanced accuracy-efficiency challenge in face detection, a novel context enhanced approach is proposed which improves the performance of the face detector in terms of both loss function and structure. For loss function optimisation, a hierarchical loss, referred to as "triple loss" in this thesis, is introduced to optimise the feature pyramid network (FPN) based face detector. For structural optimisation, this thesis proposes a context-sensitive structure to increase the capacity of the network prediction. Experimental results indicate that the proposed method achieves a good balance between the accuracy and computational cost of face detection.

To suppress the background noise in aerial image object detection, this thesis presents a two-stage detector, named as "SAFDet". To be more specific, a rotation-anchor-free-branch (RAFB) is proposed to regress the precise rectangle boundary. As the RAFB is anchor free, the computational cost is negligible during training. Meanwhile, a centre prediction module (CPM) is introduced to enhance the capabilities of target localisation and noise suppression from the background. As the CPM is only deployed during training, it does not increase the computational cost of inference. Experimental results indicate that the proposed method achieves a good balance between the accuracy and computational cost, and it effectively suppresses the background noise at the same time.

Contents

List of Figures

List of Tables

Acknowledgements

At the beginning, I would like to thank the financial support from the University of Strathclyde and my first supervisor Prof. Stephen Marshall.

I am a lucky child on my way of research, because of my supervisors: Prof. Stephen Marshall and Dr. Jinchang Ren. Steve helped me to build my research skills. He always shares ideas with me, which are really inspiring especially in my first year. He is the one who has opened the gate of "academia" for me. As the research going deeper, he provided me with precious opportunities of getting involved in different projects, which greatly broadened my horizons and increased my experiences. Meanwhile, I also learned how to cooperate with others, which will be beneficial throughout my whole life. I also want to show my gratitude to my second supervisor, Dr. Jinchang Ren. He is a thoughtful and patient supervisor. Due to the weakness in my English writing skills, I have long been struggling to present myself well. It was him who helped me express and promote my research result appropriately and precisely. Sometimes I worked until midnight, but I could still get feedback from him for the questions I asked. Whenever I need help, he is always here.

Further more, I would like to thank the staffs and colleagues at CeSIP, including Christine Bryson for her support, Dr. Paul Murray for his advises on my annual reviews and my final VIVA, Dr. He Sun (we had the VIVA at the same day, what a coincidence!) for the academic discussions we had and always sharing the recent updates of deep learning with me, Dr. Yijun Yan for the advices on both research and daily life, Ha Viet Khanh (also as my good neighbour) for his knowledge on image super resolution and TensorFlow, Dr. Julius Tschannerl for his humour and culture exchange, and Fraser Macfarlane for the talks about the academic and the common

hobbies. All of them are unforgettable.

I would like to thank my parents for their selfless support and understanding. "Just follow your heart. We will support your decision." With the encouragement from my parents, I started my PhD research life. As parents, they can not help being worried about their child who is fighting for his academic dreams in a foreign country. However, every time when my research was not going well, they were always the ones who encouraged and supported me to go further. These years are tough, not just for me, but also for them.

At the end, I want to thank my love, Shiyun Zhao, to stand by my side, now and forever.

Acronyms / Abbreviations

2D Two-Dimensional

3D Three-Dimensional

ANN Artificial Neural Network

AP Average Precision

bbox Bounding Box

BN Batch Normalisation

BO Bayesian Optimisation

CGP Cartesian Genetic Programming

CNN Convolutional Neural Network

DARTS Differentable Architecture Search

DE Differential Evolution

DenseNet Densely Connected Network

DMP Deformable Part Models

DSFD Dual Shot Face Detector

DSOD Deeply Supervised Object Detectors

DSSD Deconvolutional Single Shot Detector

EA Evolutionary Algorithm

ES Evolutionary Strategy

FC Fully Connected (layer)

FLOP Floating Point Operation

FPN Feature Pyramid Network

GA Genetic Algorithm

GE Genetic Evolution

GEP Gene Expression Programming

GI Genetic Improvement

GN Group Normalisation

GP Genetic Programming

GPU Graphics Processing Unit

HBB Horizontal Bounding Box

HOG Histograms of Oriented Gradients

ICS Internal Covariate Shift

IoU Intersaction over Union

LBP Local Binary Patterns

LGP Linear Genetic Programming

LReLU Leaky Rectified Linear Units

mAP Mean Average Precision

MLP Multilayer Perceptron

NAS Neural Architecture Search

NEAT Neuro Evolution of Augmenting Topology

NIN Network in Network

NMS Non Maximum Suppression

OBB Oriented Bounding Box

PReLU Parametric Rectified Linear Units

SVM Support Vector Machine

R-CNN Regions with CNN features

R-FCN Region-based Fully Convolutional Networks

ReLU Rectified Linear Units

ResNet Residual Network

RFBNet Receptive Field Block Network

RGB Red, Green and Blue

RPN Region Proposal Network

SFD Single Shot Scale-invariant Face Detector

SIFT Scale-invariant Feature Transform

SPPnet Spatial Pyramid Network

SRN Selective Refinement Network

SSD Single Shot MultiBox Detector

Chapter 1

Introduction

This chapter will give a brief introduction of the motivation as well as the contributions of this thesis. At the end of this chapter, the organisation of the whole thesis will be introduced.

1.1 Aims and motivations

Convolutional Neural Network (CNN) is originally inspired by the structure of cat's primary visual cortex [1], where cells are sensitive to a small region (known as receptive field). Having been developed and refined for more than two decades, CNN is now widely used in many computer vision tasks as a deep learning architecture, e.g. classification [2], detection [3], super resolution [4], restoration [5], and segmentation [6]. This thesis focus mainly on the applications of CNNs in classification, face detection and aerial object detection.

Image classification is a fundamental task for a wide range of fields. Traditional algorithms conduct classification tasks in two steps: Feature extraction which is handcrafted [7–9], and Classification using a statistical classifier [10–13]. While some user-selected features of traditional algorithms can be beneficial in a specific context, it may not be the case once the context changes. In recent years, CNNs have become a dominant computer vision approach in image classification. Compared with the two-step traditional classification pipelines, CNN processes the task using an end-to-

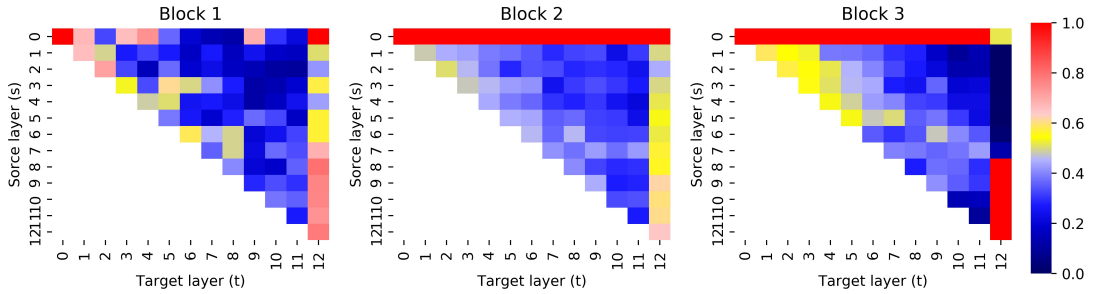


Figure 1.1: The average absolute filter weights of convolutional layers in DenseNet40 ($G = 12$), trained on the CIFAR-10 dataset. The colour patch (s, t) denotes the average L1 normalised (by number of input depth) weights in the layer "t" which takes input from the layer "s" in the same block. The last column of each block denotes the transition layer for the first two blocks and the fully connected layer for the last block respectively.

end method, which consists of both feature extractor and classifier. The self-feature generation approach improves the robustness of CNN, producing the state-of-the-art results in many fields [2, 14, 15].

To generate higher level feature maps, the number of convolutional layers of the state-of-the-art CNN has been increased generation by generation [2, 16]. More recently, Residual Network (Resnet) [17] and Densely Connected Convolutional Networks (Densenet) [18] have expanded this number to more than 100 layers. However, the channels in the inputs of a layer may not be all necessary. Veit et al. [19] has found that the deep residual network performs as a combination of several shallow networks. The heat map of Densenet (shown in Fig. 1.1) demonstrates the same effect: i.e. not all channels in the concatenated input feature maps are essential for the classification tasks. Only a few channels are weighted with a relatively high value, which will be used in the following layers. The remains, however, are of low values, which is not that necessary and can be removed. As discovered by Goodfellow et al. [20], a CNN may be misled when adding the input feature map by a small value (0.007 for example). The redundant information from these low significance inputs will not only impede the prediction accuracy, but also cause a waste of computational resource.

Image classification predicts the category of an image, while object detection locates and identifies objects in an image. However, the features required for decision making

Figure 1.2: Examples of easy, medium and hard faces on WIDER FACE dataset, which are highlighted using blue, green and red boxes, respectively. According to the detection difficulty, which is measured by the average recall of EdgeBox detector [21] with 8,000 proposals per image, faces are split into three subsets: easy, medium and hard. For WIDER FACE dataset, the rates for these three subsets are 92%, 76%, and 34%, respectively.

in those two application are similar, which makes it possible to apply CNNs, although originally trained for classification, to object detection as feature extractors.

Face detection is a basic task in various computer vision and face related applications [22]. According to the level of difficulty in detection, which is measured by the average recall of EdgeBox detector [21] with 8,000 proposals per image, faces are split into three subsets: easy, medium and hard. For WIDER FACE dataset, the rates of these three subsets are 92%, 76%, and 34%, respectively. Examples of easy, medium and hard faces are shown in the Figure 1.2. At first, handcrafted feature extractor played an important role, such as Haar-like features proposed by Viola-Jones [23], in which the face image is segmented to several patches via multi-scale sliding windows. For each patch, the classification work is conducted by a two-class cascade classifier. Based on this pipeline, the following subsequent works [24–27] improve the accuracy by optimising the cascade detectors. Limited by the complexity of Haar-like features, cascade

classifier is only sensitive to the frontal face. To improve the robustness, deformable part models (DPM) [28,29] build features by considering the relationship of deformable facial parts. However, handcrafted features are effective only on specific poses and angles, which are unable to handle multi-scale and multi-angle faces [30, 31] in the wild. Thanks to the breakthrough on the convolutional neural networks (CNN), which extract features automatically without manual work, a series of CNN based models are proposed on object detection. Studies on Faster-RCNN [3] find that an end-to-end CNN is more robust and accurate than handcrafted object detector. To increase the inference speed, Single Shot MultiBox Detector (SSD) [32] proposed a multi-stage prediction structure, which could predict objects from low-level to high-level feature maps. However, as the low-level feature maps are in lack of semantic information [33], SSD has difficulty in detecting small objects. To tackle this drawback, feature pyramids are proposed, with a "backward path", which can link the high-level feature map to the low-level feature map for more effective feature fusion. Due to different requirements of fields, Faster-RCNN, SSD and Feature Pyramid Network (FPN) are widely used in face detection [34–43]. However, detection of hard faces is still a challenging task. Compared with easily detected faces, the resolution of hard faces is always low, which are interfered by, such as, blurry, occlusion, illumination and makeup. Those interferences cause the lack of visual consistency [41].

Object detection is one of the basic tasks in computer vision. Most of existing CNN detectors are anchor based, which is a predefined rectangle. During box prediction, detector predicts the difference between anchor and the actual box size. Hence, the selection of anchor is important for accurate prediction. There are two types of anchors: horizontal anchor and rotate anchor, which are shown in Figure 1.3. For horizontal anchor, the edges of anchors are horizontal with the image edges. For rotate anchor, however, anchor edges are rotated with a rotation angle. In aerial image, object detection tends to locate and identify objects from bird-views. Different from general object detection [44, 45], the bounding boxes in aerial images are arbitrarily oriented, rather than horizontal as in other images. Furthermore, the complex background in aerial images increases the difficulty of feature extraction, which makes it hard for detector

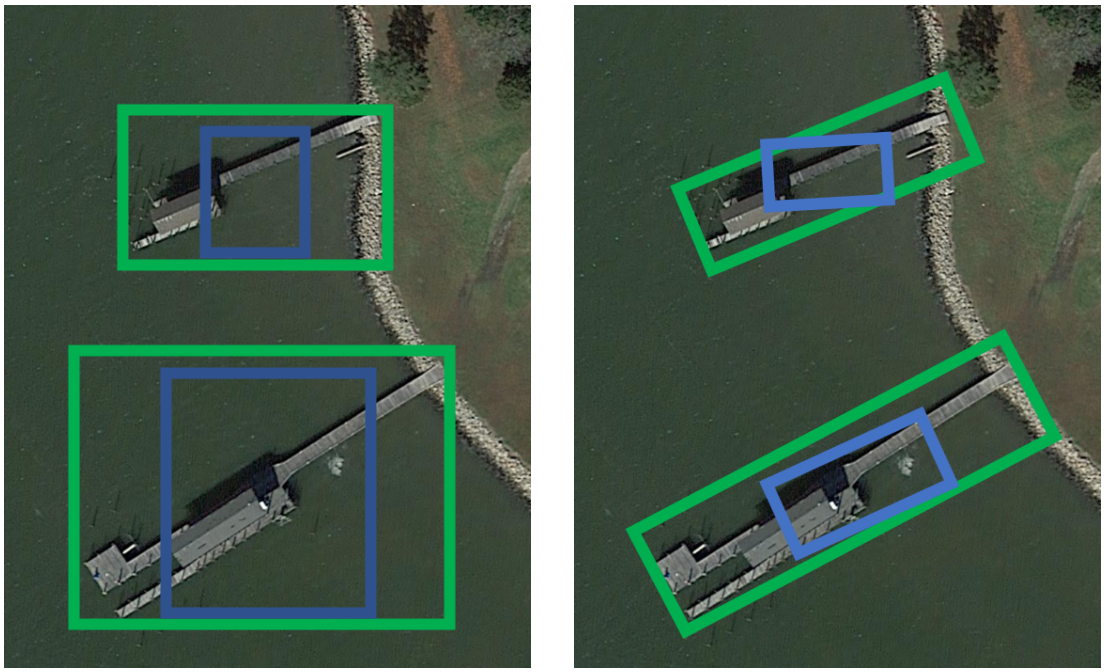


Figure 1.3: Examples of horizontal anchor (a) and rotate anchor (b). In each example, the anchors are highlighted by blue and the prediction results are coloured using green

to distinguish the (small) foreground objects. Existing methods [46–50] have reached promising results by utilizing two-stage detectors [3, 51, 52], i.e. to select the regions of interest (ROIs) via a region proposal network (RPN) followed by a two-branch subnet for subsequent category classification and box regression.

Most existing methods predict ROIs based on horizontal anchors [3]. However, as horizontal proposals are along with the image edge, the extracted feature of an object may contain features of the background and the surrounding objects, especially for objects with a high aspect ratio or those densely positioned. The misalignment [48, 53] between extracted feature and object feature will interfere the following detection procedure. Examples of misalignment are shown in Figure 1.4, where the majority area of the bounding box is the background (a) and the surrounding objects (b). There are two main solutions to the misalignment problem, i.e. either replacing the horizontal anchors by the rotate anchors [54–59] or enhancing the capability of noise suppression [48, 54, 60]. However, on the one hand, rotate anchors need to take the predefined angle (or orientation of object) into account, hence the number of anchors can be

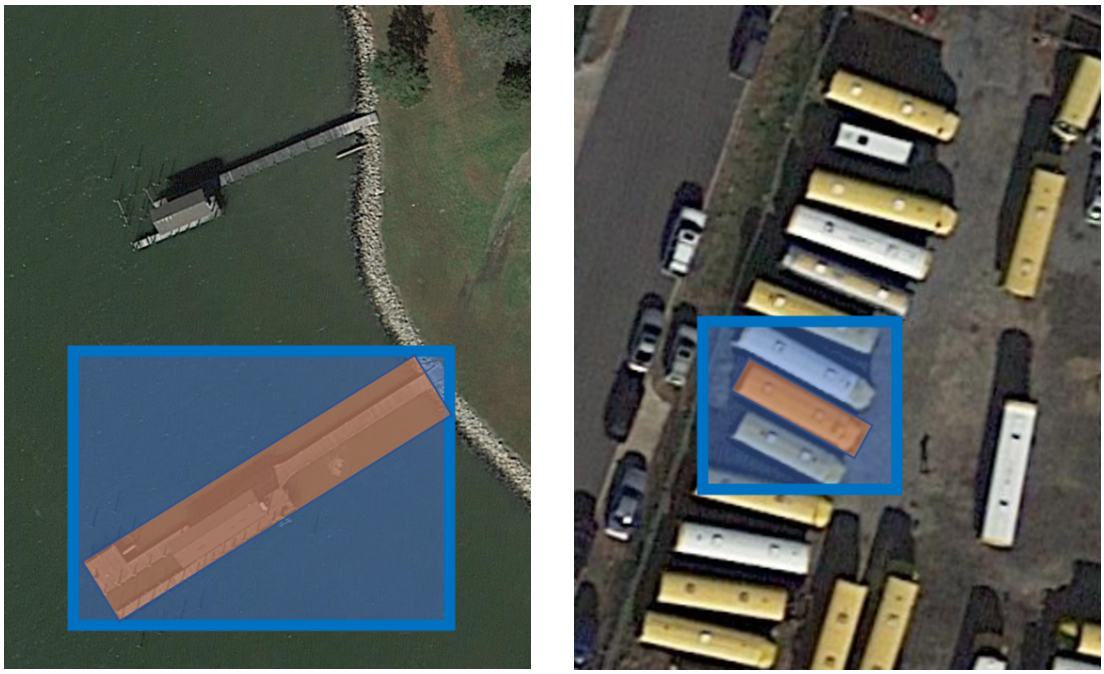


Figure 1.4: Examples of misalignment caused by (a) background feature and (b) feature from surrounding objects. In each example, the extracted feature is highlighted by blue and the object feature is coloured using orange

dramatically increased. On the other hand, the capability of feature extraction relies on certain assistant modules [39,60,61], such as multiple convolutional layers for enhanced feature extraction. As a result, both solutions mentioned above increase the total computational cost and are not practically ideal due to the poor efficiency.

Therefore, the aim of research presented in this thesis is to develop novel CNN structures to tackle the imbalance of accuracy and computational cost, which can be utilised in classification, face detection and aerial object detection. In summary, the objectives of this thesis are defined as follow:

1. Optimise the topology of CNNs for classification, in which unnecessary convolutional kernels are reduced.

2. Propose an effective face detector of small, blurred and partially occluded faces in the wild. Meanwhile, reach a trade-off between computational cost and accuracy.

3. For aerial image detection, address the misalignment caused by horizontal proposals and the interference caused by highly complex backgrounds.

1.2 Main contributions of this thesis

By achieving the objectives highlighted in Section 1.1. The key contributions of this thesis can be summarised as follow:

1. **A mechanism is proposed that automatically optimises the topology of CNNs to generate the structure of fixed length models.**

    i. An effective GA training pipeline is proposed to select the key input channels of CNN layers automatically;

    ii. With limited computational resources, mechanisms are proposed to optimise the GA process by applying 'weight inheritance' to reduce the total computational cost without degrading the training or testing accuracy of the models;

2. **Improve the detection performance on hard face through the optimisation of both the structure and the loss function.**

    i. Based on FPN, a training strategy is proposed which calculates losses through different pathways, while during inference, the backward pathway is applied for prediction, which increases the accuracy without causing additional computation cost.

    ii. A feature fusion module is proposed, which consists of a mixed network structure to enhance the capability of feature extraction from the fused feature maps.

    iii. When compared with other VGG-16 based face detector, the proposed method achieves superior performance over a number of state-of-the-art methods on the hard subset of WIDER FACE dataset and reaches a balance between the accuracy and speed. With an appropriate anchor setting, the proposed method can reach the state-of-the-art on the easy and medium subsets, while keeping the considerable performance on the hard subset at the same time.

3. **A two-stage semi-anchor-free detector is proposed for object detection of aerial images.**

i. A semi-anchor-free detector (SAFDet) is proposed for effective detection of oriented objects in aerial images;

ii. A rotate-anchor-free branch (RAFB) is proposed to tackle the misalignment problem caused by horizontal anchors, which can predict the OBB directly without any predefined anchor setting;

iii. A center prediction module (CPM) is implemented to enhance the capability of feature extraction during the training of RPN, hence it saves the computational cost of inference;

1.3 Thesis organisation

The remaining parts of this thesis is organised as follows:

Chapter 2 introduces related works with respect to the research topics, including CNN structure optimisation, face detection and aerial object detection. Since both face detection and aerial object detection are sub-classes of object detection. The background of general object detection is also reviewed to build a solid understanding.

Chapter 3 presents the main theoretical backgrounds which form the mathematical foundation of this thesis. First, the principle of evolutionary algorithms is introduced, which is applied to optimise the structure of CNN in this thesis. After that, theoretical knowledge of CNN are discussed in detail, including widely used layers, the structure of CNN, and the commonly used loss functions.

Chapter 4, 5, and 6 present the main contribution of this thesis by tackling the challenges on CNN structure optimisation, hard face detection, and background noise suppression for aerial object detection, respectively.

Chapter 7 concludes the work of this thesis, and it points out the possibilities for future work that can be expected to further improve the performance of current methods.

Chapter 2

Background and Related Work

2.1 Introduction

As previously described, the main topics of this thesis include the optimisation of the Convolutional Neural Network (CNN) structure on image classification, face detection and oriented aerial object detection. Since most CNN frameworks of object detection are adopted from CNN models on image classification, the development of CNN architecture design on image classification will be fully explored first, including the manually designed methods and the self-structure-generation methods. This chapter presents a comprehensive review of the generic object detection, followed by discussion over related works of both face detection and oriented aerial object detection separately.

2.2 CNN Architecture Design for Image Classification

Balancing the computational efficiency and the accuracy is a critical task in designing a convolutional neural network, as it can help to reduce the computational cost and facilitate more portable implementations i.e. on mobile or low power devices [62–64]. Two possible optimisation approaches include: i) Optimising the connection method between layers [17,18,65]; and ii) Optimising the convolutional paradigm, e.g. the kernel size [16, 66], and the activation function [67]. These two methods can be conducted both by manual design and self-generation. Manual-design-based methods are more robust but required several attempts to achieve the best solution. Self-learning-based

methods, on the other hand, can optimise the model architecture but may suffer from overspecialised.

2.2.1 Manually Designed CNN Architectures

Manually designed CNN achieved great early success firstly on the MNIST dataset [2]. After that, Alexnet [14] was proposed for large-scale practical image classification, where the state-of-the-art results were reported on the ILSVRC 2012 classification dataset [15]. However, a large CNN kernel size is inefficient, causing huge computational cost. To tackle this drawback, in NIN (Network in Network) [66], 1×1 kernel convolution is introduced to improve the efficiency of channel-wise information transmission. In VGG [16], only 3×3 and 1×1 convolutional kernels were applied, producing a deeper network with higher accuracy. Inception structure was proposed in GoogleNet [65], where the kernel size of filters can vary within a layer, increasing the information transmission pathways between layers.

Batch normalisation [67] reshapes the distribution of the input, which significantly improves the training efficiency. With increasing numbers of layers, the issue of gradient vanishing or gradient exploding [68] emerges, where the weights of the network struggle to converge (gradient vanishing or gradient exploding: As weights in neural networks upgrade according to the backpropagation, which gets gradients via the chain rule. Under the certain method, extreme gradients from the high-level layers are amplified layer by layer, and eventually cause the gradients of low-level layers vanishing or exploding. As a result, the training failed). To ease this training difficulty, the Residual Network (ResNet) [17, 69] was proposed to allow groups of layers of the network to learn the difference between input and output rather than the input-to-output transformation for each layer. This has significantly reduced the total number of the parameters and allowed the number of layers to be extended to over 100. However, it is found that there are many redundant layers in the ResNet [19, 70]. To further improve the efficiency of the information transformation, in Huang et al. [18], a dense connection strategy, DenseNet, is proposed to connect the outputs of all formal layers as the input of the following layer rather than sum all outputs in the ResNet. A comparable

accuracy to ImageNet was achieved while the number of parameters was reduced to 1/3 of the original ResNet. However, there still exists many unnecessary layers in the DenseNet, leaving potential for further optimisation.

2.2.2 Neural Evolution Method

To simulate the genotype-based evolution process of nature, evolutionary algorithms (EAs) have been presented in the literature to optimise a model by encoding it to a binary string or even a string of integers or floating-point numbers. At the early stage, EAs were used as adaptive mechanisms to update the weights in neural networks while fixing the network architecture [71]. As presented by Stanley et al. [72], the network architecture evolved using the Neuro Evolution of Augmenting Topologies (NEAT) algorithm, in which the connection or disconnection between nodes was evolved through mutation. When combined with the compositional pattern producing networks [73], NEAT can be used to evolve a larger scale neural network [74].

A straightforward way for coding the model is direct coding [72], where all the nodes and their connection strategies are stored in its genotype. Direct coding performs well when training the network completely using EAs. However, the information of a direct-coding-based genotype string may be redundant when using EAs to optimise only one hyperparameter. Thus, an alternative mechanism, indirect coding, has been presented in the literature to optimise a specific hyperparameter of the neural networks [75–79]. These include i) kernel reformation for a fixed architecture [79], ii) optimisation of the connections [75, 76, 78, 80], and iii) adjustment of the number of filters at each layer [75, 77].

When introducing the backpropagation algorithm [81] for optimising the weights of the neural networks, it becomes unnecessary to train such weights using an EA. This is because backpropagation algorithm helps to converge the weights to the global minimum more easily and quickly. In order to gain the benefit of both back-propagation and EAs, one strategy is to combine them [82–84], where they can be used to simultaneously train the weights of the neural network and optimise the architecture. Although these methods achieve comparable results to manually designed CNN models, they can

only optimise the coding method and the training efficiency rather than the scale of the whole model. This has led to their models being too small to cope with large-scale image classification tasks. For instances, to fit the scale of ImageNet where the model is originally searched on CIFAR10, GeNet [76] stacks manually designed layers and LEMONADE [85] enlarges the number of layers per block. More recently, the scale of the model has been considered in the network architecture [75, 77, 80] that evolves the process to design the optimised architecture with a large searching space achieving comparable results on the CIFAR-10 dataset [86].

2.2.3 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) is a process of automatic architecture engineering. In image classification, NAS methods have outperformed most of manually designed models. In this chapter, two widely used NAS approach, **gradient** based NAS and the **reinforcement learning** based NAS, will be summarised. Other methods such as Bayesian Optimisation (BO) [87], pruning based [88], and meta-learning [89] will not be covered here, because of inefficiency [90] or low performance [91].

Reinforcement Learning NAS. As presented by Zoph et al. [92], reinforcement learning was used to generate a fixed length structure of a CNN layer by layer. Furthermore, the addition or removal of identical connections is also considered, which helps the model achieve state-of-the-art results. A similar training framework was utilised by Baker et al. [93], where Q-learning was applied to explore the structure of the model in each layer. Instead of training with a fixed length of "Gene string", the number of layers was determined within the reinforcement learning itself. Reinforced Evolutionary Neural Architecture Search (RENAS) [94] integrated reinforced mutation into an evolution algorithm to increase the evolve efficiency. Mobile NAS (MNAS) [95] proposed a factorised hierarchical search space to balance the flexibility and the search space size, which was applied for light-weight network generation. PROXYLESSNAS [91] removed the proxy task learning via path-level binarisation. To reduce the scale of network, Facebook-Berkeley-Nets (FBNets) [96] utilised a layer-wise search space to specify the type of blocks for each layer.

Gradient NAS. Instead of searching the structure via an "agent", gradient NAS optimises the structure using gradient descent. A differentiable NAS method (DARTS) was proposed in DARTS [90], applied on both convolutional and recurrent networks. To increase the computational efficiency, stochastic NAS (SNAS) [97] replaced the feedback mechanism with more efficient gradient feedback from generic loss.

With the rapid development of deep learning, there are plenty of CNN architectures proposed in recent years. Here the effective CNN architectures based on manual designed and NAS are listed in the Table 2.1 and Table 2.2, respectively. More architectures are introduced in [98–100]

2.2.4 Training Strategy of Evolution Based CNN

To optimise the architecture of the CNN, each network model is considered as an "individual" and its fitness value is measured based on the classification accuracy on the validation datasets [75–77, 80, 110]. After the mutation and crossover (excluding [75], which replaces the crossover by computing a huge number of combinations), the model is retrained to determine the fitness value individually. Each new individual can be either fully trained i.e. trained 100 epochs on the CIFAR-10 dataset [75, 76, 80, 110] or partially trained [77] i.e. 5 epochs on the same dataset. In comparison to the fully trained approach, partial training improves the training efficiency but suffers from overestimation or under estimation caused by random initialisation from a Gaussian distribution [111]. In Real et al. [75], it is found that an alternative training strategy inherits benefits from both fully and partial training, where each "individual" is partially trained after structure evolution from full training. If a variable is reused, its value will be taken from the last generation instead of being reinitialised from scratch.

Similar works have been conducted by Real et al. [75] and Xie et al. [76]. The search space of the proposed method is the input channels of each layer, while in Real et al. [75] and Xie et al. [76] the search space is the connection method between different layers. Even the elements of a layer are also considered in Xie et al. [76]. A large search space makes both methods hard to be optimized under limited computational resources. To improve the training speed and reduce the computational cost, the proposed method

Table 2.1: Summary of manual designed CNN

Model	Key Features
Architectures with Stacked layers	
AlexNet [14]	Use large size kernels (maximum 11)
ZFNet [101]	Reduce the maximum kernel size to 7
NIN [66]	Use 1×1 convolution as channel-wise MLP
VGG [16]	Reduce the maximum kernel size to 3 Enlarge layer number to 19
Architectures with Skip connections	
Inception [65]	Output is concatenated by multiple convolutional layer with different kernel size
HighwayNet [102]	Output of low layer can be added to the high layer after passing a sigmoid function
ResNet [17]	Output of low layer is added to the high layer
DenseNet [18]	Output of low layer is concatenated to the next layer
Variants of ResNet and DenseNet	
SENet [103]	Add one more branch into the residual block, where the input is the average along height and width of the original input feature map
FractalNet [104]	Replace the convolution layer in ResNet by residual block
ResNext [105]	Replace the convolution layer in ResNet by inception block
HRNet [106]	Fuse feature maps from different resolutions
Res2Net [107]	Replace the convolution layer in ResNet by dense block
Light-weight methods	
SqueezeNet [62]	Use residual block with fewer channels
MobileNet [63]	Replace convolution by separable convolution
ShuffleNet [64]	For each layer, take partial channels from the former outputs

in Chapter 4 does not discard the crossover step as proposed in Real et al. [75]. Experimental results in Section 4.3 show that, by using crossover, the total computational cost can be dramatically reduced whilst maintaining the test accuracy. The original weight inheritance method [75] is optimized for the residual structure which is not fit for the densely connected structure. As a result, in Chapter 4, a new pre-trained weight inheritance method is proposed to initialize the weights of the model, which are more

Table 2.2: Summary of NAS

Model	Searching Method
NASNet [108]	Reinforcement Learning
GeNet [76]	Genetic algorithm
LEMONADE [85]	Evolutionary algorithm
SNAS [97]	Gradient from backpropagation
Proxyless [91]	Gradient from backpropagation and reinforcement learning
DTARS [90]	Gradient from backpropagation
FBNet [96]	Gradient from backpropagation
RENASNet [94]	Gradient from backpropagation
MnasNet [95]	Reinforcement Learning
AmoebaNet [109]	Evolutionary algorithm

suitable for densely connected structures.

2.3 CNN for Object Detection

2.3.1 Generic Detection

Image classification specifies the categories of an image, but is unable to localise the exact positions of objects. To further discover the semantic information in an image, it is essential to precisely estimate the sizes and the coordinates of objects in an image. This task to solve the challenge is referred as object detection. According to the application scene, there are plenty of the sub-classes in object detection, such as ship detection [112], face detection [35–37] and scene text detection [47, 113].

Before the existence of CNN-based object detection methods, the pipeline of traditional object detection methods intend to tackle this task as regional image classification. As the positions and sizes of objects are varied in images, it is a natural choice to "scan" an image via a pre-defined rectangle, which refers to as "sliding window". Each time the sliding window moves a specified distance (known as "stride") along x-axis or y-axis, a classifier will conduct a classification to the sliding window region. An object is detected if the result is true and its coordinate and the size will be described by the existing position and the size of the sliding window. The classification process on each

sliding window region can be mainly divided into two steps: feature extraction and classification.

Before classification, a feature extractor is applied on grey-scale image to extract visual features, such as Haar-like [23], Histograms of Oriented Gradients (HOG) [7], Local Binary Patterns (LBP) [8] and Scale-invariant Feature Transform (SIFT) [9]. After that, a classifier,e.g. adaboost [10], support vector machine (SVM) [11], random forests [12] and artificial neural network [13], will assign the region an category ({foreground, background} for single class detection and {class 1,class 2, ... class n, background} for multi-class detection) based on the extracted features.

Based on these manual-designed local feature extractors and shallow learning architectures, state of the art results have been achieved on the PASCAL VOC object detection competition [44] before 2012. However, the limitations of the detection pipeline are obvious: For the sliding window design, different objects may exist in any positions of the image with arbitrary aspect ratio and scale, single scale sliding window may be unable to precisely fit the object. As a result, image should be scaled multiple times (image pyramid), which is computationally expensive and produces lots of background windows. As for the feature extraction, those hand-crafted features only produce benefits in specific applications and may fail once transferred to a new application.

In recent years, CNN based object approaches have become the dominant on object detection. During the process of Two-Dimensional (2D) convolution, a kernel convolves images from top-left to the bottom-right, making the convolution kernel a naturally sliding window. Compared with the two-step traditional detection pipelines, CNN processes the task using an end-to-end method, i.e. the network generates the desired feature maps itself from low level to high level. The self-feature-generate approach improves the robustness of CNN, producing state-of-the-art experimental results in many applications: from single-class object detection tasks [35–37, 47, 112, 113], to more complex large-scale object detection tasks [3, 32, 33, 114]. On the other hand, CNN detector locate an object by regressing the difference between the predefined rectangles (the rectangles are known as "anchors") [3, 32, 33, 114] or the coordinates of the pixel [115–122] (these methods are known as "anchor-free"), which makes the

localisation more accurate than the traditional methods.

Following the work in [123], the frameworks of the CNN based object detection approaches can mainly be divided into two categories: multi-stage detectors and single-stage detectors. For multi-stage detectors, the category classification and the box regression are cascaded, which is similar to [23], while single-stage detectors conduct classification and the regression only once. Multi-stage detectors mainly include Regions with CNN features (R-CNN) series [3,51–53], Spatial Pyramid Network (SPPnet) [124], Region-based Fully Convolutional Networks (R-FCN) [125], Feature Pyramid Network (FPN) [33], cascade RCNN [126], RefineDet [127], Libra R-CNN [128], and Light-Head R-CNN [129], while the single-stage methods mainly includes MultiBox [130], AttentionNet [131], Gird-CNN (G-CNN) [132], YOLO [116], Single Shot MultiBox Detector (SSD) [32], Deconvolutional Single Shot Detector (DSSD) [133], Deeply Supervised Object Detectors (DSOD) [134], Receptive Field Block Network (RFBNet) [135], RetinaNet [114], and TridentNet [136], as well as the anchor-free methods [115–122]. Multi-stage detectors take the advantage of accuracy while single-stage detectors are better at computational speed.

2.3.2 Face Detection

Back to 1990s, face detection became increasingly important in computer vision, which has been widely used in multiple applications such as face recognition, facial expression recognition, and face tracking [22]. At early stage, face detection mainly extracted feature using a hand-crafted feature extractor, such as Haar-like features [23], control point set [137] and the Deformable Part Model (DPM) [28,29]. These detectors reached promising detection accuracy and high efficiency at the same time.

Recently, results in [3,36] indicate that CNN can extract more powerful features than hand-crafted face detectors. As a result, CNN based face detectors become dominating in face detection in the last decade [35, 39–42, 138, 139].

2.3.2.1 Structures of CNN Face Detector

According to the structure of CNN, most of existing CNN face detectors can be divided into two categories, i.e. multi-step detectors (SSD-like [32], one stage only.) and single-step detectors (faster-RCNN-like [3], containing one stage [37] or two [36]). A single-step detector [32, 34, 35] produces a promising accuracy using the feature map, which is extracted from the deepest layer of its backbone. However, the stride of the deep layer is often quite large (usually 16 [36] or 32 [140]). As a result, the information of tiny faces may vanish. To tackle this issue, multi-step detectors detect faces on feature maps extracted from different depths of CNN, where shallow layers are for detecting small faces and deeper layer for large faces. However, due to the limitation caused by the insufficient capability of feature extraction, shallow layers are not rich enough for extracting semantic information as deep layers [33]. In order to enrich the semantic information on shallow layers, [33] proposed a top-down pathway, where the feature maps of deep layers and shallow layers are fused together, using addition [35], element-wise multiplication [141] or concatenation [140].

2.3.2.2 Feature Extraction Module

Faster-RCNN (RCNN, Regions with CNN features) [3] firstly presented a convolutional subnet for face detection, in which the subnet contains a single 3×3 convolutional layer, followed by two sibling 1×1 convolutional layers (also called "detection head") for classification and box regression, respectively. To reduce the computational cost, SSD [32] replaced the two subnets in the Faster-RCNN with a subnet with two 3×3 convolutional layers. To further increase the capabilities of classification and regression, based on the SSD detection head, RetinaNet [114] inserts four additional 3×3 convolutional layers before the last two layers. However, a 4-layer subnet has significantly increased the computational cost: for a typical FPN-based face detector, there are in total 6 feature maps which means 12 additional feature extraction subnets will be added. Even though the weights of subnets are shared in between, the computational costs for each subnet are independent to each other. To balance the accuracy and the computational cost, in recently proposed methods, a series of inception based subnets [67] are introduced

to replace the four-layer subnet. For example, FANet [40] and Pyramidbox [41] have found that a simple two-branch inception module can keep the accuracy as retina head (consists of 5 convolutional layer for feature extraction and 1 convolutional layer for prediction) when using the SSD head (consists of 1 convolutional layer for prediction). Selective Refinement Network (SRN) [42] introduced a four-branch residual-inception subnet [142], as a replacement of the first two layers of Retina head. Dual Shot Face Detector (DSFD) [141] applied the dilation convolution into the subnet, which expands the receptive field without increasing the computational cost significantly.

2.3.2.3 Loss Function Design

Imbalanced ratio of positive examples and negative samples during training impedes the performance significantly, especially for SSD-like detectors [114, 143, 144]. To address this issue, online hard example mining (OHEM) [143] automatically selects hard-negative examples as three times of positive examples. In order to further make use of easy-negative examples, Lin et al. [114] proposed a focal loss which weights the loss of examples according to the difficulty of learning. Another two applicable strategies are multi-task prediction and hierarchical learning. For multi-task prediction, detection head will be assigned additional face-related prediction tasks, such as key points detection [139], head-body detection [41], face attention [39], and face segmentation [138]. Different from multi-task prediction, tasks of the hierarchical learning are the same as the ordinary object detection training yet predicting objects from different "pathways". FANet [40] applied the FPN [33] structure in evaluation but predicted faces from one forward and two backward pathways during training. DSFD [141] narrowed the range of prediction layers to two pathways and assigning different anchor sizes for each pathway. SRN [42] cascaded prediction results from both pathways, which reduced easy-false-positive examples significantly.

However, existing methods cannot well balance the accuracy and the computational cost. Large scale models, with multiple pathways and deep backbones [39, 42, 138, 141], improve the accuracy with a sacrifice of computational cost. On the other hand, the structures of high-efficiency face detectors are always shallow [140]. As a result, the

accuracy is not high enough in some dense detection scenes [31]. To address this issue, a novel context enhanced approach, as well as the triple loss training strategy, are detailed in the Chapter 5.

2.3.3 Oriented Aerial Object Detection

2.3.3.1 Anchor based oriented aerial object detection

For general object detection, existing methods, such as SSD [32], faster-RCNN [3], feature pyramid network [33], and RetinaNet [114], predict the bounding box of the object by regressing the difference between a number of candidate boxes at each pixel location (known as "anchor") and the actual one (the ground truth) in the image. For object detection in aerial images, however, there is no angle element in the horizontal anchor setting and the rotate angle is fixed to 90 degrees. This increases the difficulty of rotated angle regression. Moreover, horizontal anchors usually contain more background area than the rotate anchor, which has resulted in the problem of misalignment between the detected ROI and the ground truth. To address this challenge, rotated anchor setting is introduced [54,58,59,61,112], in which anchors at different angles are predefined. However, the increased number of anchors ("$numscales \times numaspectratios \times numangles$" compared with "$numscales \times numaspectratios$") makes it more time-consuming than the horizontal anchor setting in object detection.

2.3.3.2 Anchor-free Object Detection

The anchor-free strategy is originally designed for general object detection [115–122, 145]. Different from anchor-based methods [3, 32, 33, 114], anchor-free methods predict one bounding box per pixel. Instead of regressing the difference between the actual bounding box and the anchor, anchor-free methods predict the bounding box based on the coordinates of the feature map. However, the regression formats are different between anchor-free methods. DenseBox [115], CornerNet [117] and FoveaBox [120] predict the upper-left and bottom-right coordinates of the bounding box, known as the "XYXY" mode. YOLO [116], DuBox [145] and CSP [121] predict the center coordinate, and the associated width and height of the bounding box, known as the "XYWH"

mode, which is the same mode as used in the anchor-based methods. Different from these two prediction modes, FCOS [119] predicts the distances between the center point and the four boundary lines of the bounding box, respectively. Anchor-free methods can actually reduce the computational cost on the box regression branch. However, due to the lack of anchors in anchor-free methods, the original intersection over union (IoU) matching scheme cannot be applied. Thus, the positive assignment becomes a challenging task [119–121, 145].

2.3.3.3 Multi-task Loss of Oriented Aerial Object Detection

Previous works apply multi-task prediction [53, 60, 139], i.e. jointly conduct multiple prediction tasks on each object during training, such as bound box prediction, key point, and segmentation, to enhance the capacity of box prediction. Apart from bounding box regression and category classification, multi-task prediction can also be applied in other important tasks such as determining the keypoints [139], image segmentation [53] and attention detection [40, 60]. As mentioned in [146], existing challenges contain low resolution, noise, and crowd. Previous works tackle these challenges by using attention structures [39, 60, 103, 147, 148], where supervised attention structures [39, 40, 60] consider all the pixels within the boxes as the foreground. However, the anchors centered at the boundary of the bounding boxes, especially for those with large aspect ratios (height/width), may have low IoU matching values than the predefined threshold. These boundary pixels are supposed to be categorised as the background rather that the foreground.

For aerial object detection, there are challenging scenarios including small objects and dense arrangement. Thus, in Chapter 6, a two-stage aerial object detector is proposed. To gain the benefits from both anchor-based and anchor-free regression schemes, an anchor-free branch is introduced in the region proposal network (RPN) during training and apply anchor-based regression branch during inference. In order to locate the position of objects more precisely, a centreness loss is presented in the RPN, which predict the centre of objects only. Details will be described in the Section 6.2.

2.4 Summary

In this chapter, the background and related works of CNN based image classification and the CNN based object detection have been comprehensively reviewed. The CNN based image classification part introduced the development of CNN structure, including manually designed methods and self-structure-generation methods. Then the review of objection detection presents the development of generic object detection, which acts as the stem of the following works. After that, studies on the two branches of general object detection, namely face detection and oriented aerial object detection, are further reviewed with an exploration of the detection framework and the multi-task training strategy. According to the review, related mathematical backgrounds will be detailed in Chapter 3, while the contributions within these fields will be developed in Chapter 4, 5 and 6.

Chapter 3

Theoretical Background

3.1 Introduction

The content of this chapter forms the theoretical and mathematical foundations of the whole research, which lies in mainly two folds: 1. evolutionary algorithms (EAs) and 2. convolutional neural networks (CNNs).

First of all, the mechanism of Evolutionary Algorithms (EAs) are detailed. Since EAs are widely used in many fields and it is impractical to cover all, the introduction of EAs in this chapter focuses on the role of EAS in the network optimisation, to which the work presented in Chapter 4 makes a significant contribution. After that, the mathematical foundations of CNNs are introduced in Section 3.3, which starts from a review of different types of layers, to the loss functions of CNNs.

3.2 Evolutionary Algorithms in NAS

EAs are a heuristi c-based framework for solving problems that would take expensive computational cost to optimise. There are plenty sub-classes of EAs, which are frequently used by industry and research, including Evolutionary Strategy (ES), Genetic Algorithms (GA), Genetic Programming (GP) , Genetic Improvement (GI), Grammatical Evolution (GE), Linear Genetic Programming (LGP), Cartesian Genetic Programming (CGP), Differential Evolution (DE) and Gene Expression Programming (GEP) [149]. The general processes of EAs are similar [150], as concluded in Figure

3.1. Within the steps in the process, there exists many variations. In recent years, EAs are widely used in NAS [85, 109, 151–158] to optimise the structures of CNN models. As the aim of adopting EAs in this thesis is to optimise the structure of CNN, the introduction of EAs in this section will be focused on the application of EAs in NAS.

Before the process starts, a string is defined to present the structure of CNN, which refers to *individual*. At the first step, a number of individuals (known as *population*) are initialised. The quality of each individual is determined by a predefined objective function, where the procedure referred to as fit. For the NAS, this could be the loss on the test set of the target dataset, such as CIFAR [86], ImageNet [15] and COCO [45]. To obtain this, a model, where a string presents, will be trained on the train set of the target dataset. However, the value of the string will not be modified during training. Based on this fitness value, three evolutionary operations are conducted to evolve the value of the string(also the architecture of the model): *selection*, *crossover* and *mutation*.

Selection. In order to keep the well-performing structure and remove the bad-performing ones, a selection step is firstly applied to the population. Take tournament selection [159], which is applied in the Section 4.1, as an example. In each generation, the population is randomly sampled to $S(S<I)$ individuals I times, where each "individual" might be resampled, forming I candidate groups. Only the best individual is selected from each of the candidate groups, which is a "parent" of the next generation (the "parent group" is denoted as "P") and all the other individuals in each candidate group are discarded.

Crossover. After generating the "parent" individuals, the "child" individuals can be generated by combining the "parent" individuals in a pairwise manner and conducting crossover on each pair of the "parent" individuals. In crossover, bits from different individuals but at the same position in the string are randomly "swapped". The probability of crossover for each pair is P_C and the probability of crossover of a bit is Pb_C. Crossover modifies the structures of a pair simultaneously. When the structures of a paired individuals are modified, the fitness values of the pair will be re-evaluated.

Mutation. After crossover, each "child" individual flips some of its bits randomly,

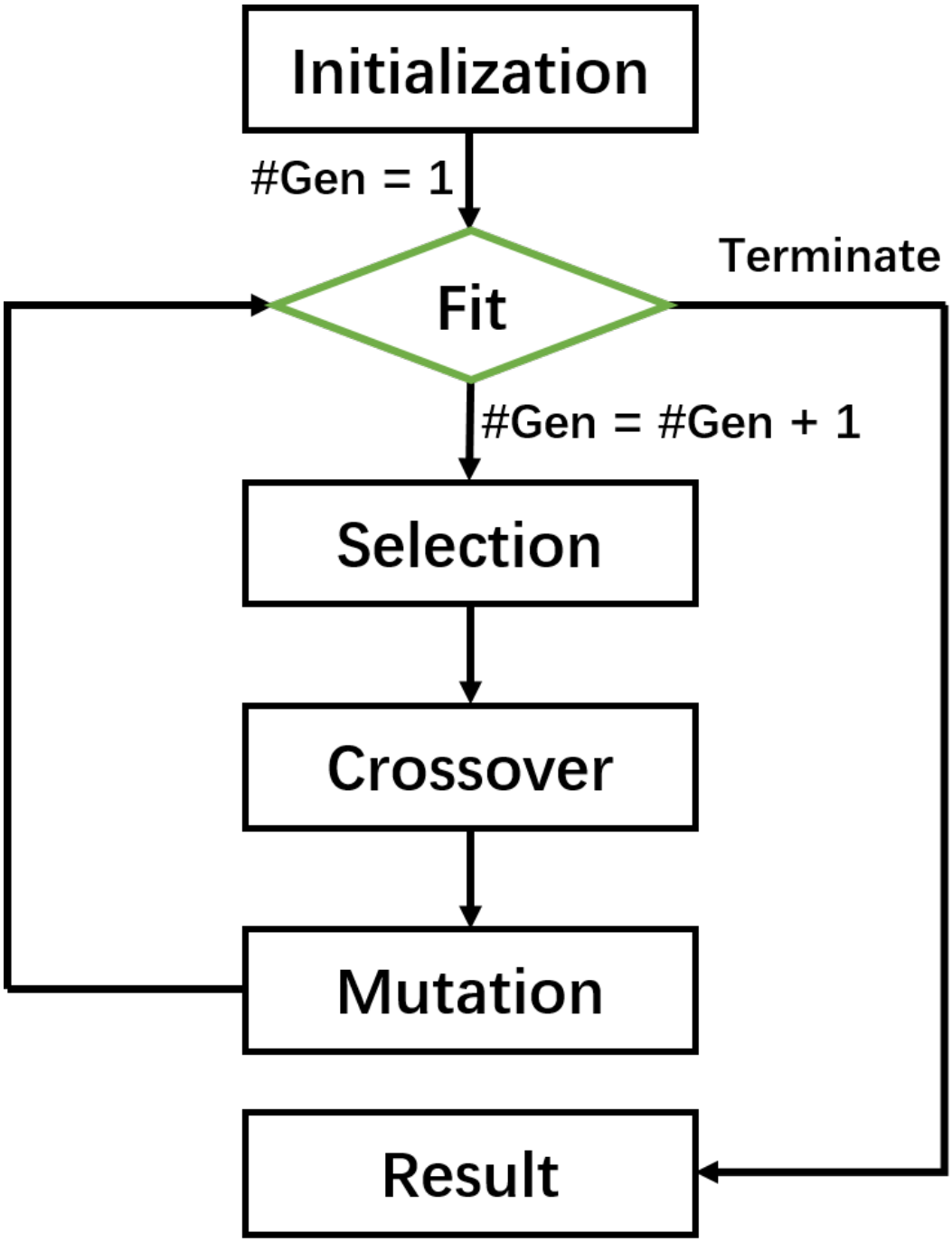


Figure 3.1: Basic process of evolutionary algorithms. The order of crossover and mutation can be altered according to the specified cases.

which is known as mutation. Selection and crossover remove the bad structures, while mutation brings more variance to the structure. The probability of mutation for each

pair is P_M and the probability of mutation of a bit is Pb_M. If the structure of a "child" individual is mutated, its fitness value will be re-evaluated. There exists a special situation that binary bits of a layer are all zero. This means no feature map inputted to this layer. To avoid overspecialised, the first and the last bits are automatically flipped to "1" rather than discarding the layer as did in Real et al. [75], Xie et al. [76], and Sun et al. [77]. Experimental results in Real et al. [75] indicates that, without this constraint, the model generated on CIFAR-10 has fewer layers. However, when applying on CIFAR-100 dataset, the generated structure becomes suboptimal. As a result, the number of layers should not be stabilised during the evolution process.

Eventually, the algorithm will terminate, where one of two situations emerges (sometimes both): 1. the number of generations has achieved the maximum; 2. the fitness value has reached the predefined threshold. When the algorithm is terminated, individuals of the current generation are all returned.

3.3 Convolutional Neural Networks

A convolutional neural network (CNN) is a class of artificial neural network (ANN), that consists of one or more layers and are used mainly for image processing, such as classification, segmentation, and image super resolution. There are various types of layers for different functionality, including feature extraction, dimension reduction, data regularisation, data biasing, decision making, and so on. In this section, at first, the related types of layers are introduced, followed by the loss functions and the methods of gradient descent, which is applied for network training. After that, the commonly used structures of CNN are presented.

3.3.1 Layer Types in CNN

3.3.1.1 Fully Connect Layer

Starting from multilayer perceptron (MLP) [160], fully connected (FC) layers are widely used in the machine learning models [160–164]. In fully connected layers, a basic computational unit is *node*, also known as *neuron*. A *connection* between nodes refers to

a linear transfer function across two layers. Nodes at the same layer are not connected. In FC, a node connects all the nodes from its former layer and add them together as the input. Then, the input is subject to an $activation function$, which generates the output of the node.

Assuming that there are two FC layers in CNN, which are denoted as l and $l+1$ respectively. The output of node j in the layer $l+1$ can be mathematically expressed by:

$$Z_j^{l+1} = f(\sum_{i=1}^{N}(w_{ij}^l Z_i^l + b_{ij}^l)) \tag{3.1}$$

where Z_i^l is the output of node i in the layer l. Z_j^{l+1} is the output of node j in the layer $l+1$. w_{ij}^l denotes the weight and b_{ij}^l is the bias element, respectively. $f(*)$ is the activation function. Simply use linear transformation makes it difficult to fit nonlinear problems. As a result, non-linearity is achieved via activation function. By using nonlinear functions, a network can approximate an arbitrarily function, given that it is sufficiently large [165–167] Figure 3.2 shows an example of two FC layers' subnet.

For image classification and detection, the commonly used activation functions are sigmoid, softmax and rectified linear unit (ReLU). The Sigmoid function is also known as logistic function [168]. Before the existence of ReLU, the Sigmoid function is a non-linear function (as visualised in Figure 3.3) which used mostly in Artificial Neural Network (ANN) [169]. The formula of sigmoid function is given below:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

As seen in Eq. 3.2, sigmoid function tends to predicting probability based on output. Thus, it always appears in the output layers, achieving a success in binary classification problems. Recently, [114] proves that sigmoid functuion can also been applied in multi-class prediction problems (i.e. the number of classes is larger than 2), where the probability of each class is predicted individually. At early stage, the sigmoid function

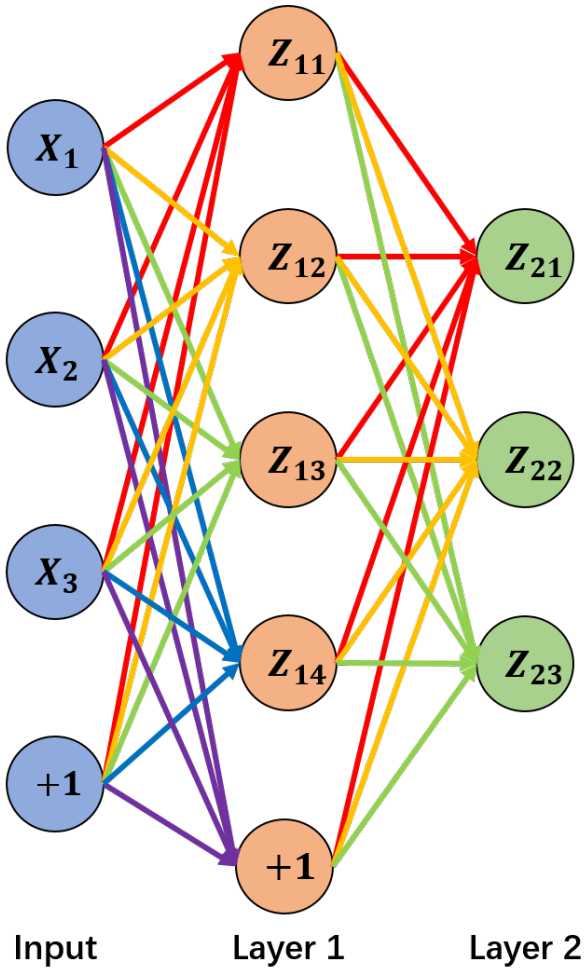


Figure 3.2: An example of two FC layers' subnet. weights are expressed by arrow. +1 indicates the bias elements

are also used in the hidden layers (layers between the input layer and the output layer) of shallow ANNs as it is easy to understand [170]. However, when apply sigmoid as the activation function of hidden layers, there exists two major drawbacks. Firstly, the exponential function in the sigmoid increases the computational cost, reducing the total speed of the network. Secondly, sigmoid suffers sharp damp gradients, slow convergence and gradient saturation [167] in the backpropagation algorithm [81], increasing the training difficulty of ANNs.

Similar to the sigmoid function, the Softmax function [171] is another class of activation function to compute probability based on output. However, instead of predict

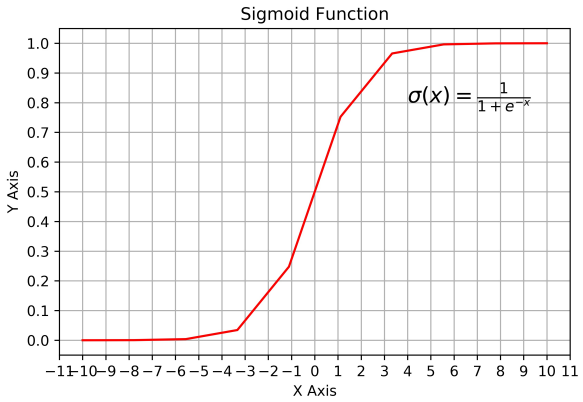


Figure 3.3: The sigmoid activation function

the probability of each node individually, the softmax function predicts the probability distribution from all nodes in the output layer. The probability of a node i in the output layer can be mathematically expressed by:

$$f(Z_i^{out}) = \frac{e^{Z_i^{out}}}{\sum_{j=1} e^{Z_j^{out}}} \tag{3.3}$$

where Z_i^{out} denotes the output of node i. From the Eq. 3.3, the range of a node is $[0, 1]$, and the summation of the probabilities equals to 1. When the number of classes equals to 2, i.e. binary classification problems, the softmax function becomes identity with the sigmoid function.

The ReLU was proposed in [172], and has been widely applied in deep learning applications [171, 173]. The ReLU thresholds each input node, where negative values are set to zero while keeping non-negative values unchanged. The formula of ReLU is given by:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{3.4}$$

As the ReLU is a combination of linear functions, it maintains the properties of linear

functions [169]. Thus, it is easier to optimise, when compared with the sigmoid function [171]. It also performs the better capability on generalisation compared to the sigmoid function [174, 175]. However, the ReLU has a significant drawback that it discard the nodes with negative values. As a result, the gradients are also zero on negative-value nodes, which makes the training easy to fail, i.e. gradients are all zeros on a layer, causing the weight update interrupted. To tackle this issues, the leaky ReLU and the parametric Rectified Linear Units (PReLU) are proposed.

The leaky ReLU activation functiuon (LReLU) assigns a negative slope (e.g. 1e-2 in Pytorch [176]) to the ReLU to the weight updates through all nodes during the backpropagation process [177]. The LReLU function is defined as:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{3.5}$$

where the parameter α is the negative slope introduced as above. However, as the magnitude of the negative slope is small, the performance improvement of network using the LReLU function is not obvious when compared with that using the ReLU function [169, 177].

The parametric ReLU [111], denoted as PReLU, is a variant of the LReLU function. The format of PReLU is the same as the LReLU, except that the negative slope α is a node-wise trainable variable which can be optimised using the backpropagation algortithm. As reported in He et al. [111], by adopting the PReLU, the performance is marginally improved in large scale image recognition tasks [15], when compared with the network using ReLU function.

3.3.1.2 Convolution Layers

One limitation of FC is the huge number of parameters. If apply a MLP, where the number of the hidden layer is 1024, to CIFAR dataset [86]. As the size of CIFAR images is $32 \times 32 \times 3$, the number of parameters on the hidden layer is $32 \times 32 \times 3 \times 1024 = 3,145,728$. As reported in Pascanu et al. [178], for a shadow ANN, such amount of parameters increases the training difficulty, causing gradient exploding or vanishing

gradients. To tackle this, convolution layer is proposed to reduce the number of parameters. In a convolution layer, each output node only takes local-wise input nodes, rather than global-wise. The operation of multiple dimensional discrete convolution can be mathematically expressed by:

$$Y = \sum_{k_1} \sum_{k_2} \cdots \sum_{k_M} X(k_1, k_2, \cdots, k_M) h(k_1, k_2, \cdots, k_M) \tag{3.6}$$

where X is the input feature map. h is the convolution *kernel*. M denotes the number of dimensions. In CNN, $M = 2$ [16,17,105] and $M = 3$ [179,180] are widely used, which refers to Two-Dimensional (2D) and Three-Dimensional (3D) convolution, respectively. The process of convolution can be visualised in Figure 3.4. As shown, the kernel moves on the feature map from the top-left to the bottom-right. The number of pixels that each time the kernel moves is the *stride* of the kernel. The shape of feature map is commonly expressed by $B \times H \times W \times C$, where B, H, W, C are the batch size (i.e. the number of images), height, width and the number of channels, respectively (e.g. for a single RGB image, the size can be expressed by $1 \times H \times W \times 3$). During convolution operation, the weights of kernels are assigned separately for each channel.

Different from FC, which takes all nodes as input, nodes in convolution layers are affected by a region of nodes in the feature map. Thus, the *receptive field* of a node is proposed to measure the number of nodes (or "pixels" in CNN) with respect to the original image, that make contributes to the result. The method of calculating the receptive field is given by:

$$RF_{out} = RF_{in} + (k - 1) * s_{tot} \tag{3.7}$$

where RF_{out} and RF_{in} are the receptive field of the output and the input feature maps, respectively. k is the size of kernel. s_{tot} is the total stride of the existing layer, which is obtained by the size of input image divided by the size of the input feature map. Knowing the receptive field of each layer is important for network design and diagnose [181]. It is critical for each pixel of the output layer to have an adequate

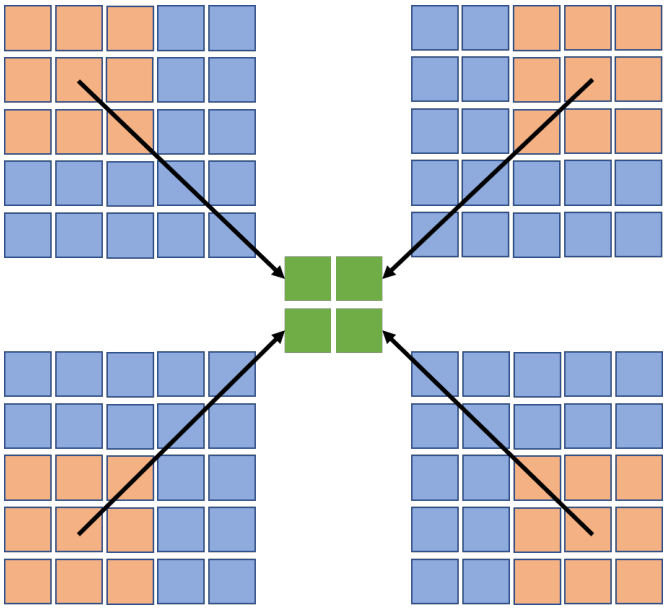


Figure 3.4: An example of the convolution layer. The blue square presents the input feature map, where the orange square is the convolutional kernel, and the green square is the output feature map. The four subplot of feature maps on the figure indicates the moving path of the convolutional kernel: from the top-left to the bottom-right

large receptive field [182], especially for the pixel-wise prediction tasks, such as object detection [35], stereo matching [183] and semantic image segmentation [184].

As seen in Eq. 3.7, there are three methods to increase the receptive field of the output layers: stacking more layers, increasing the kernel size and adopting sub-sampling layers (pooling layers). Increasing the kernel size remarkably enlarges the receptive field [185]. However, the number of parameters also increases, causing training difficulty and reduction of computational cost. An alternative method is dilated convolution [186] (also known as "atrous convolution" [187]).

Dilated convolutions are applied to increase the receptive field of output nodes without increasing the number of kernel parameters, which is a low cost method compared with enlarging the kernel size. To be concrete, dilated convolutions expand the kernel size by inserting spaces between the kernel elements [188]. The distance between each kernel elements are *dilated rate* (denoted as r), which is a hyperparameter conntrolling the "actual" size of kernel. When the $r = 1$, dilated convolutions becomes the basic convolution. An example of dilated convolution operation is shown in Figure 3.5, where

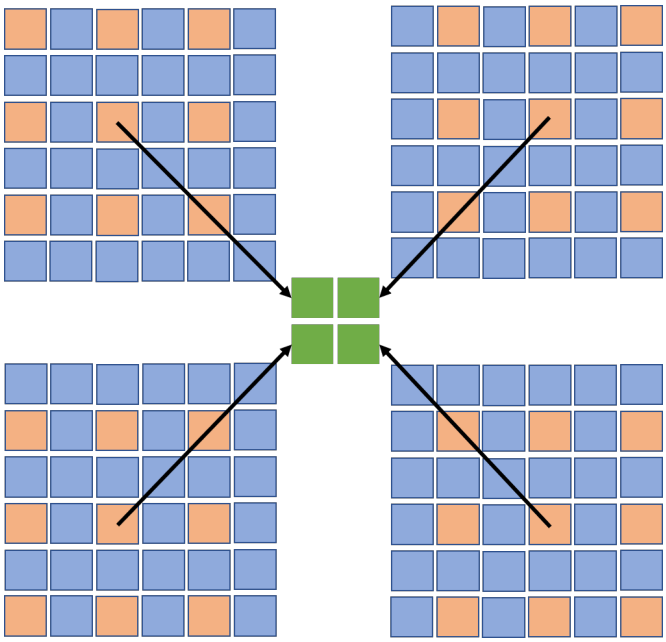


Figure 3.5: An example of the dilated convolution layer ($r = 2$).

the dilated rate is 2.

The input of dilated convolution is "sampling" in a region, rather than depends on all nodes. As a result, the dilated convolution may lost information during forward propagation, when compared to large-kernel convolution. It is recommended that deploy dilated convolution layers at the end of CNNs as high-level feature extractors [135, 141].

Specifically, when the kernel size is reduced to 1, as shown in Figure 3.6, such convolutional layer becomes "channel-wise MLP", which is capable of enhancing model discriminability and regularising feature map. As the network going deeper, 1×1 convolution layer also becomes a key role in adjusting the depth of feature map.

3.3.1.3 Pooling Layers

Pooling layer is another important module in CNNs. On the one hand, as introduced in the Eq. 3.7, a pooling layer reduce the size of feature map, which rapidly increases the receptive fields of the following convolution layers. On the other hand, it reduces the computational cost of CNNs. Under limited computational resources, this operation

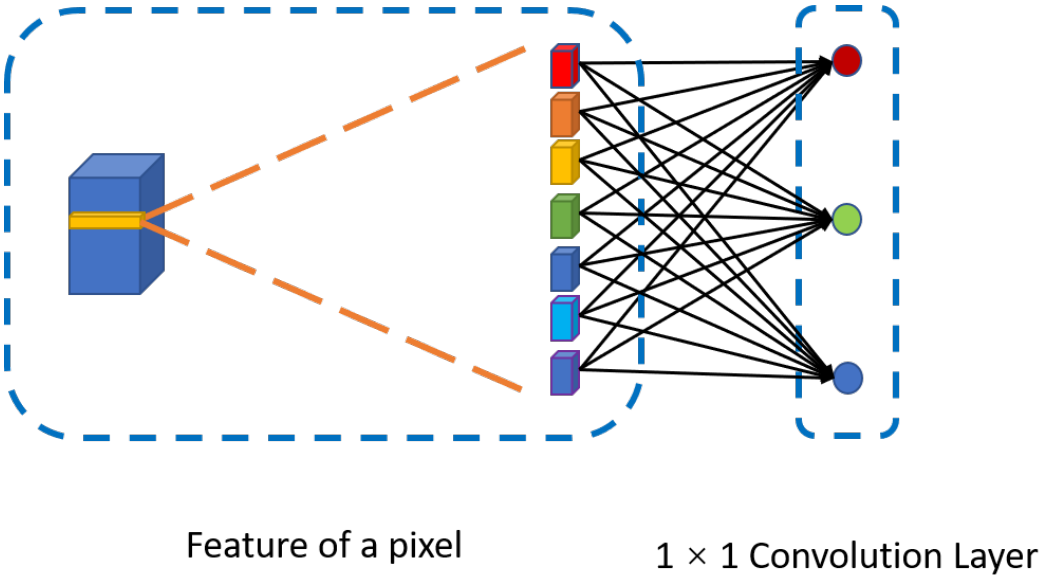


Figure 3.6: Example of the 1×1 convolution layer. The kernel size shown in the figure is $1 \times 1 \times 7 \times 3$

helps CNNs implement more layers, which increases the prediction performance [16].

There are two widely used pooling layers: average pooling and max pooling. The mechanism of pooling layers are similar to the regular convolution: a kernel slides from the top-left to the bottom-right of the input feature map, where the size reduction is achieved via large stride (typically $s = 2$ [16, 17]). For pixels in the kernel, average pooling takes the average value over all pixels as output, while max pooling outputs the max value. To interpret another way, max pooling captures the most important pixel in a region, while average layer only smooth feature map. As a result, for image-like data, max pooling is outperforms average pooling [189]. A variant of average pooling is global average pooling (GAP) [17], which takes average value through all the pixels in a channel, where the output is fed to FCs for final prediction.

3.3.1.4 Normalisation Layers

During backpropagation, weights (and bias) of each layer update synchronously. The input distributions of high-level layers is affected after each iteration, which is amplified as the network goes deeper. Thus, lower learning rate (a hyperparameter to control

the magnitude of weights update at each iteration) is required to stabilise the training. However, reducing the learning rate increases training iterations while reduces prediction accuracy [17, 67]. This phenomenon is referred to as *internal covariate shift* (ICS) [67]. To tackle this, batch normalisation (BN) layer [67] is proposed. The operation of BN is given by [67]:

$$\begin{aligned} \mu_{\mathcal{B}} &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_{\mathcal{B}}^2 &= \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \\ \widehat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ y_i &= \gamma \widehat{x}_i + \beta \end{aligned} \tag{3.8}$$

where $\mathcal{B}$ is the set of mini-batch ($\mathcal{B} = \{x_1, x_2 \dots x_m\}$) and m is the mini-batch size. $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are the mean and variance over a mini-batch, respectively. γ and β are two trainable variables for reshaping the distribution of input. ϵ is a small constant for avoiding the propagation crush when zero variance. For convolution layer, $m = B \times H \times W$, where B, H, W are the batch size, height and width of feature map, respectively. As seen in Eq. 3.8, the input data x_i is whitened, i.e. linear transform data to zero means and unit variances (using $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$). However, simply whiten data will constrain the representation ability [67, 190]. Thus, the whitened data is then reconstructed via a linear transformation (via γ and β). As the functionality of BN layers is reshape the distribution of the activation input. The BN Layer is implemented after a convolution layer or a FC Layer, but before the activation function. As illustrated by Ioffe et al. [67], the BN layer also acts as a regulariser for reducing overspecialised.

As mentioned above, the BN layer normalise data along the dimensions of $B \times H \times W$. A small batch size causes inaccurate estimation of the means and variance of mini-batch [190], causing model errors increase. As a result, a large batch size is essential for BN to work (typically 32 per GPU [67, 142]). However, for applications requires large

image size for training (e.g. detection [3, 191] and segmentation [184]), the batch size is vary small (ranged between 1 [3] to 8 [119] per GPU). The demand of batch sizes is impractical. Other variants of BN, such as batch renormalisation (BR) [192] and synchronized BN [193], alleviates the demand on large batch size, but still unable to tackle it effectively. Thus, group normalisation (GN) [190] is proposed for small batch training.

The mechanism of GN is the same as BN, except the definition of mini-batch. In GN, the feature map is firstly divided into G groups, i.e. the feature size of each group is $(B \times H \times W \times \frac{C}{G})$. Here G is a pre-defined hyper-parameter which is assigned as 32 by default [190]. For each group, GN conducts normalisation along $(H \times W \times \frac{C}{G})$ axes, without on B axis. Thus, the mini-batch size of GN is $(H \times W \times \frac{C}{G})$ which gets rid of the restriction on batch-size. Experimental results in GN [190] indicates that, for image recognition, when the batch size is below 16 per GPU, GN outperforms BN dramatically. However, when the batch size is above 16 per GPU, BN slightly outperforms GN. This indicates that BN is superior than GN, when the computational resource is sufficient.

3.3.2 Network Structures

The sections above introduce the different types of layers, where the structures of CNNs are built based. In this section, the architectures of CNN, related to this thesis, are reviewed.

3.3.2.1 AlexNet Style

The structure of AlexNet style (as shown in Figure 3.7) is straight forward, which simply stacks layers one after another. Convolution layers and max pooling layers are the main components. At the end of the network, three FC layers are used for the final prediction output. The size of feature map is reduced jointly by convolution layer and max pooling layer. For AlexNet style CNNs, due to the limited number of layers, the size of feature map is reduced rapidly. The represents CNNs of this style are AlexNet [14] and ZFNet [101], as concluded in Figure 3.8.

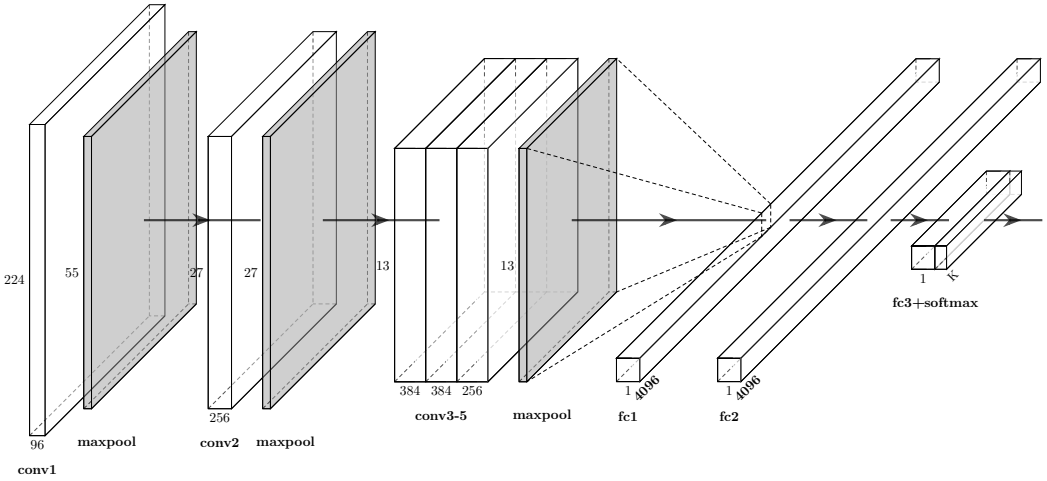
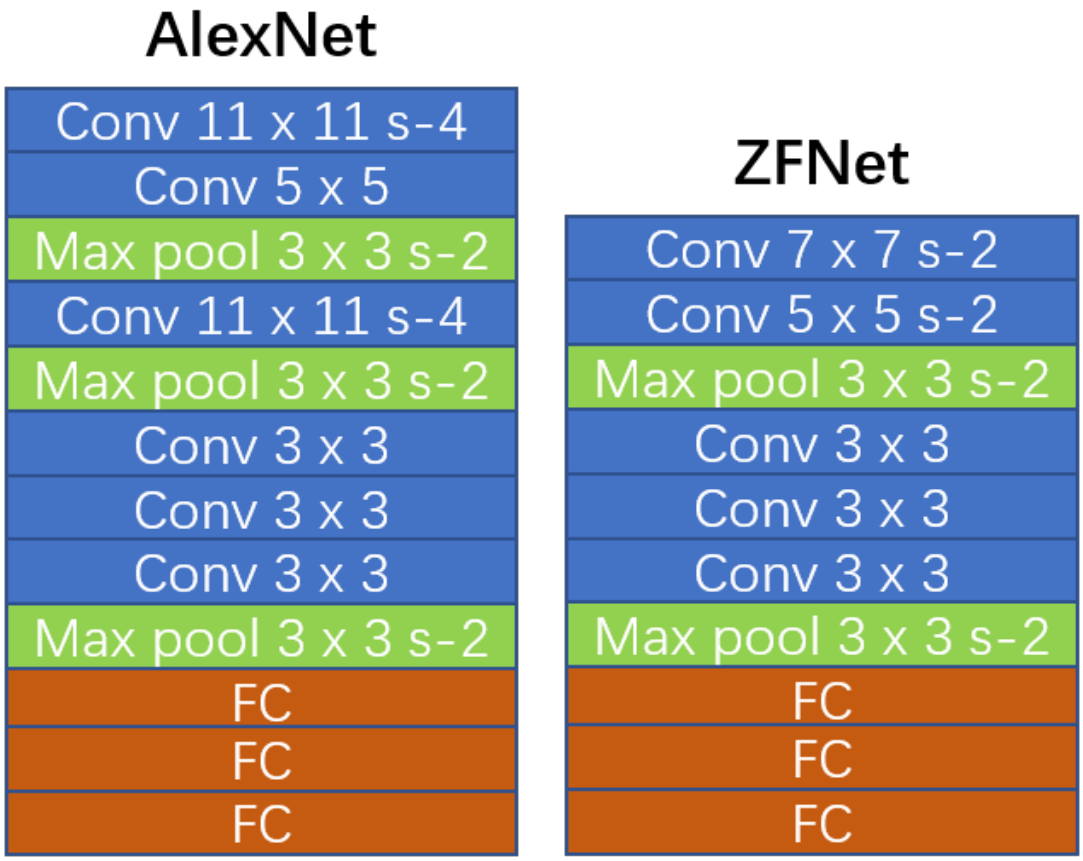


Figure 3.7: Architecture of AlexNet. layers colored by grey indicate pooling layers. For each layer, the input size is shown on the y-axis and the number of channels is taged on the x-axis. The architecture of ZFNet is omitted as it is the same as AlexNet, except the size of the first convolution layer. The details are compared on the Figure 3.8.

3.3.2.2 VGG Style

VGG style (as detailed in Figure 3.9) is a milestone in the development of CNN structure, where the core ideas are still used in the lateral CNN structures. Starting from VGG network [16], the structures of CNN gets deeper. For VGG, the maximum depth reaches 19 [16]. The structure of VGG style is similar to AlexNet style, which also cascaded connected. In VGG, the feature map is reduced by four times, which is realised by max pooling layers with out using convolution layers. Convolution layers between two max pooling layers are formatted as a "module", i.e. the sizes of output feature map are identical in a module. The decision making layer is also the same as AlexNet. The kernel sizes of convolution layers in VGG are small, which are 3×3 and 1×1. By stack small size kernel layer, model discriminability are enhanced [16, 65, 66] without hurting the receptive field [16]. A generalised structure of VGG is expressed in Figure 3.10, which is also applicable for Inception nets [65, 67, 142, 194], ResNet [17, 69] and Densely Connected Network (DenseNet) [18].



(a) AlexNet (b) ZFNet

Figure 3.8: Structures of AlexnNet and ZFNet. The expression format of convolution layer and max pooling layer is: "layer type", "kernel size", "stride", e.g. "Conv 11 $\times$ 11 s-4" indicates a convolution layer where the kernel size is 11 with stride as 4. The strides omitted are 1 by default.

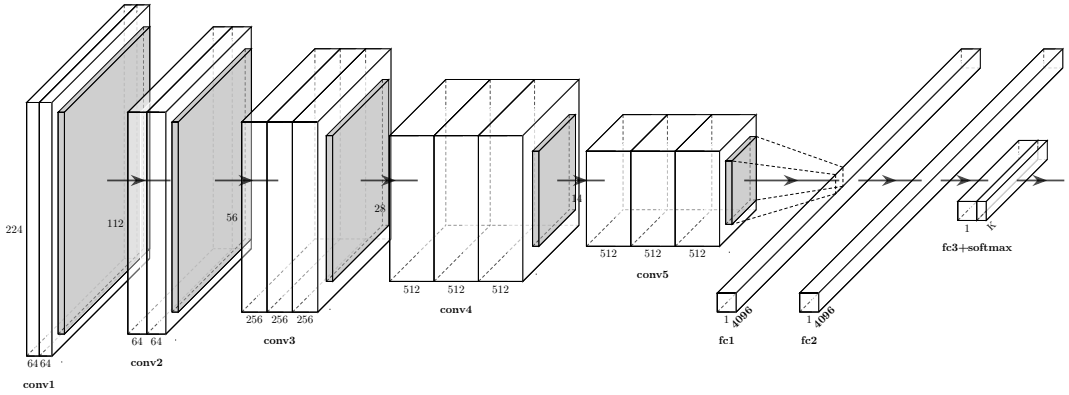


Figure 3.9: Architecture of VGG16. layers colored by grey indicate pooling layers. For each layer, the input size is shown on the y-axis and the number of channels is taged on the x-axis.

A CNN architecture can be described via the implementation method of each layer and the type of each layer. The number of channels, however, is varied with the tasks. Thus, to simplify the description, the remaining part of this chapter will focus on introducing the layer connection method and the layer type as in Figure 3.10

3.3.2.3 Inception Style

As suggested in VGG, a straightforward way to improve the accuracy is that increase the number of layers. However, on the one hand, this brings the computational cost significantly. On the other hand, the efficiency of the network is also decreased [18,65], i.e. some of kernel weights are close to zero. Thus, to increase the network scale while maintaining the utilisation efficiency of computational cost, inception block [65,67,142, 194] is proposed.

Technically speaking, the inception block, describes the connection method between layers, rather than the structure of whole network as VGG. The inception is firstly proposed in GoogleNet [65], which is also known as "inception-v1". The inception block utilises the multiple kernel convolution layers in parallel. The outputs of different layers are then concatenated along the channel axis ("C" axis for $B \times H \times W \times C$) as the input of the next inception module. With the utilisation of inception block, the depth

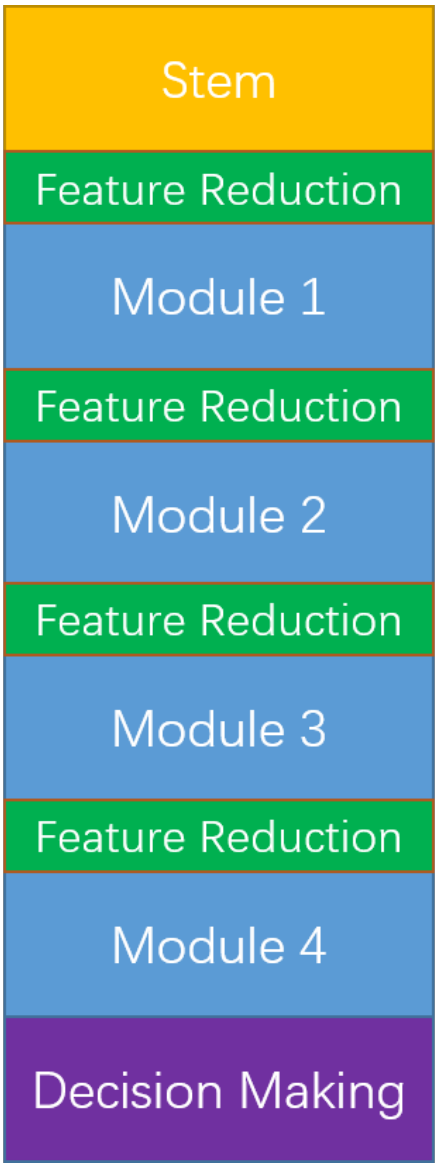


Figure 3.10: A generalised structure of VGG. This structure is also applied in Inception nets, ResNet and DenseNet. The "Stem" refers to the first module. The "Feature Reduction" is the layer where the size of feature map is reduced. "Decision Making" is the module where the prediction is made. For VGG, this module includes three FC layers.

of network achieves 22 layers. Compared to VGG, the error rate is reduced by 0.83% on ImageNet dataset [15] (a large scale image classification dataset), while the number of parameters is about 21 times fewer (6.7977 million for GoogleNet vs 144 million for VGG).

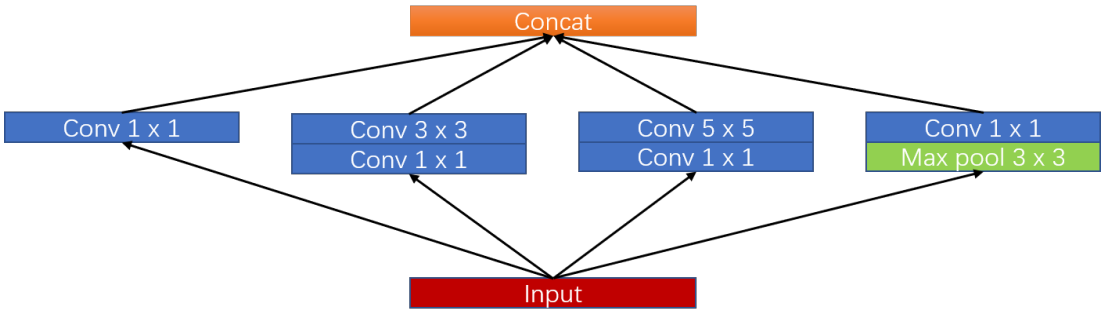
There are another three variances of inception block. The structure of inception-v2 [67] adopts BN into GoogleNet. In inception-v3 [194], a combination of $1 \times n$ and $n \times 1$ kernels are introduced as a replacement of $n \times n$, which further reduce the number of parameters. In inception-v4, the residual connection [17] is combined, which is introduced in the following section. Examples of each inception variance are shown in Figure 3.11 Inception networks outperform other manually designed CNNs. However, it also brings additional manual work on network design, e.g. the number of blocks in each module, the number of kernel types in each block and the kernel size of $1 \times n$ and $n \times 1$ layers.

3.3.2.4 ResNet and DenseNet Style

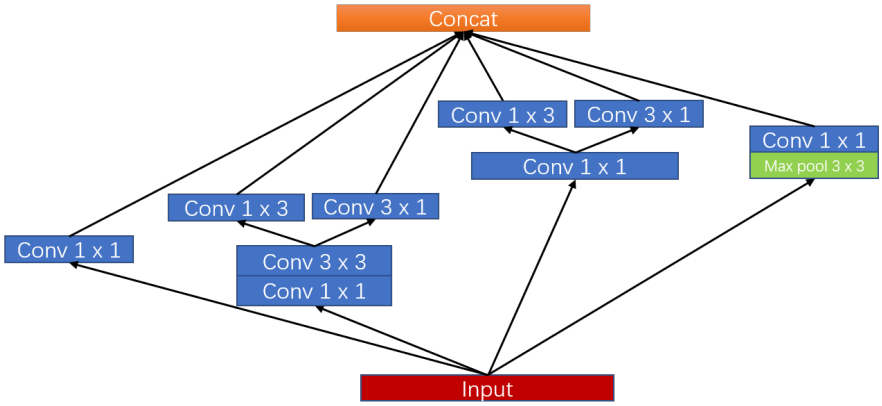
For VGG style CNNs, there exists *degradation* problem [17,18,102,195]: as the network goes deeper, the accuracy increased. However, when the number of layers exceeds a certain value, depends on the scale dataset, the accuracy drops rapidly. This problem is caused by network optimisation (e.g. vanishing or exploding gradient, inappropriate weight initialisation methods and high learning rate) rather than overfitting [17, 18, 102, 195]. To tackle the degradation problem, deep residual network (ResNet) [17, 69] has been proposed, which extent the number of layers over 1,000 without having degradation.

In ResNet, the general structure is the same as VGG, while introduced residual block in each module. The structure of residual block can be shown in Figure 3.12a. There are two 3×3 convolution layers in each block. The input "skip connects" to the output directly in a residual block, which can bypass the necessary layers. As a result, the desire layers are generated by the network during training. As the number of layer increases, the computational burden increase. To reduce the computational cost, a bottleneck residual block is also presented in ResNet [17]. The structure is visualised in Figure 3.12b. In the bottleneck residual block, the number of channels is quartered by a 1×1 kernel. After conduct a 3×3 convolution, the channel number is recovered by another 1×1 convolution layer.

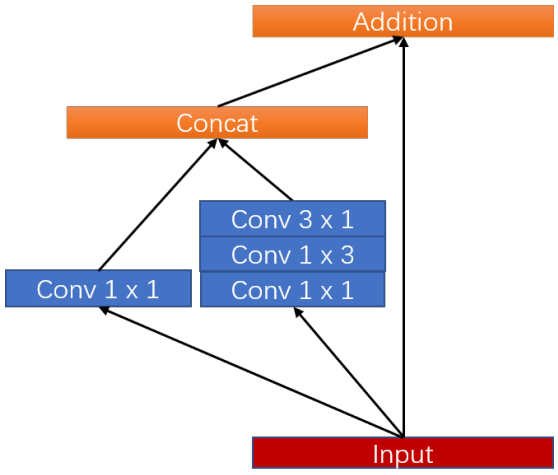
The skip connection in ResNet bypasses unnecessary layers, but also increases the



(a) Inception-v1

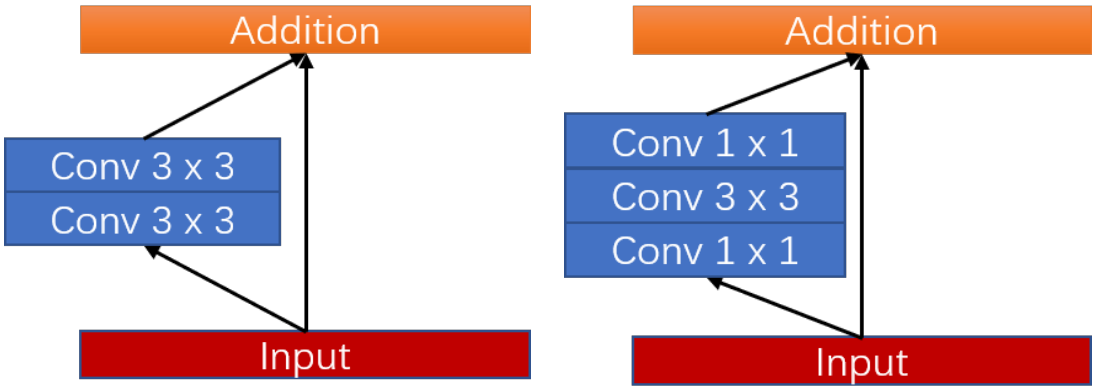


(b) Inception-v3



(c) Inception-v4

Figure 3.11: Example blocks of inception-v1 (GoogleNet), inception-v3 and inception-v4. The Inception-v2 is omitted as it adopts the same structure as inception-v1 except adding BN layers. The "Concat" indicates the concatenation layer, where feature maps from different layers are concatenated along C axis. The "Addition" indicates the addition layer, where feature maps from different layers are added. The expression formats are the same as in Figure 3.10.



(a) Original residual block

(b) Bottleneck residual block

Figure 3.12: Structures of residual block.

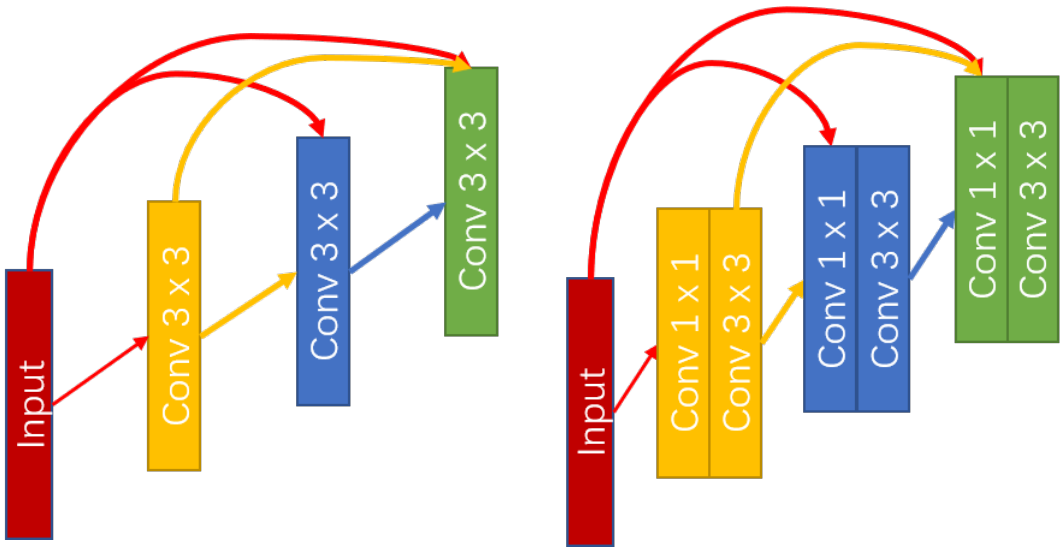
unnecessary computational cost. To increase the utilisation efficiency of network, densely connected network (DenseNet) is proposed. The densely block in DenseNet is visualised in Figure 3.13. In dense block, feature map are directly concatenated as GoogleNet, without using summation as in residual block. Experimental results in DenseNet [18], DenseNet only use $\frac{1}{3}$ parameters of ResNet, while reaching a similar performance.

3.3.3 Loss Functions

A loss function is used to measure the difference between the prediction values of CNNs and the corresponding labels. During training, weights are updated via backpropagation, in order to reduce this difference. The format of loss function is varied with the task of CNN. In this section, the loss functions applied in the following chapters are introduced.

3.3.3.1 Cross-entropy

Cross-entropy loss (also called log loss) is widely used for optimising discrete probability prediction tasks, e.g. image classification [14, 16, 17], bounding box classification in object detection [3,52] and segmentation [53,184]. Thus, the range of network prediction



(a) Original dense block

(b) Bottleneck dense block

Figure 3.13: Structures of dense block, the concatenation layers are omitted for clarity. Each arrow points to a layer where the output of the existing layer will be used as input. Feature maps from different layers are concatenated firstly. Figure (a) is the original implementation. As the network going deeper, the number of input channels increases rapidly, which will cause a huge computational burden. To tackle this, in the bottleneck structure (b), the number of input channels will be reduced by a 1×1 convolution layer at first, then it will be passed to a 3×3 convolution layer as the original setting.

is $[0, 1]$. Cross-entropy can be mathematically expressed by:

$$L_{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) \tag{3.9}$$

where N is the number of samples. y_i is the label and p_i is the prediction result from sigmoid or softmax function. For binary classification problems, the cross-entropy becomes:

$$L_{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{3.10}$$

3.3.3.2 Focal Loss

Focal loss [114] is a variant of cross-entropy for tackling class imbalance in object detection. As shown in Eq. 3.9, the loss of each instance equally contributes to the total loss. However, in object detection, only few pixels represents foreground. As a result, negative instances take majority during training. On the one hand, this phenomenon will reduce the training efficiency as most of them are easy instances, which is easy to be classified, providing no contributions for optimisation. On the other hand, the training is misled by the dominated negatives, causing model degeneration [3, 52, 114]. The principle of focal loss is that suppress the contribution of easy instances (both positive and negative) and enhance the contribution of hard instances. The formula of focal loss is given by:

$$L_{focal} = -\frac{1}{N}\sum_{i=1}^{N} \alpha(1 - p_i^t)^\gamma \log(p_i) \tag{3.11}$$

where α and γ are two balanced variants. By default, $\alpha = 0.25$ and $\gamma = 2$ [42,114,196]. p_i^t is the scaling factor defined by [114]:

$$p_i^t = \begin{cases} p_i & \text{if } y = 1 \\ 1 - p_i & \text{otherwise} \end{cases} \tag{3.12}$$

3.3.3.3 L1 and L2 Loss

Mean absolute error (also know as L1 loss) and mean squared error (also know as L2 loss) are widely used in continuous regression tasks, e.g. bounding box regression [3,32], image super resolution [197, 198]. The L1 loss is given by:

$$L1 = \frac{1}{N}\sum_{i=1}^{N}|y_i - p_i| \tag{3.13}$$

and the L2 loss is given by:

$$L2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2 \tag{3.14}$$

L1 is often applied as an auxiliary loss in CNNs [197, 198] to guarantee sparseness of representations [199]. L2 loss constrains the magnitude of weights in CNN, which prevents the network from overfitting. Thus, L2 loss is often used as regularisation loss [16, 17, 105, 200].

3.3.3.4 Smooth L1 Loss

Smooth L1 loss [52] is a variant of L1 loss, which is applied to bounding box regression in object detection [3, 32, 52]. Assume that the output of bounding box regress is v, the smooth L1 loss is defined by:

$$\mathrm{L}_{smooth-L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3.15}$$

where $x = y - v$. Previous object detection methods use L2 loss for regression [51, 124], which is unstable caused by the unbounded ground-truth. To tackle this, learning rates should be tuned carefully. The comparison between L2 loss and smooth L1 loss can be visualised in Figure 3.14. Compared to L2 loss, smooth L1 loss is insensitive to outlier targets [52], which enhances the robustness of training.

When applied CNNs into object detection

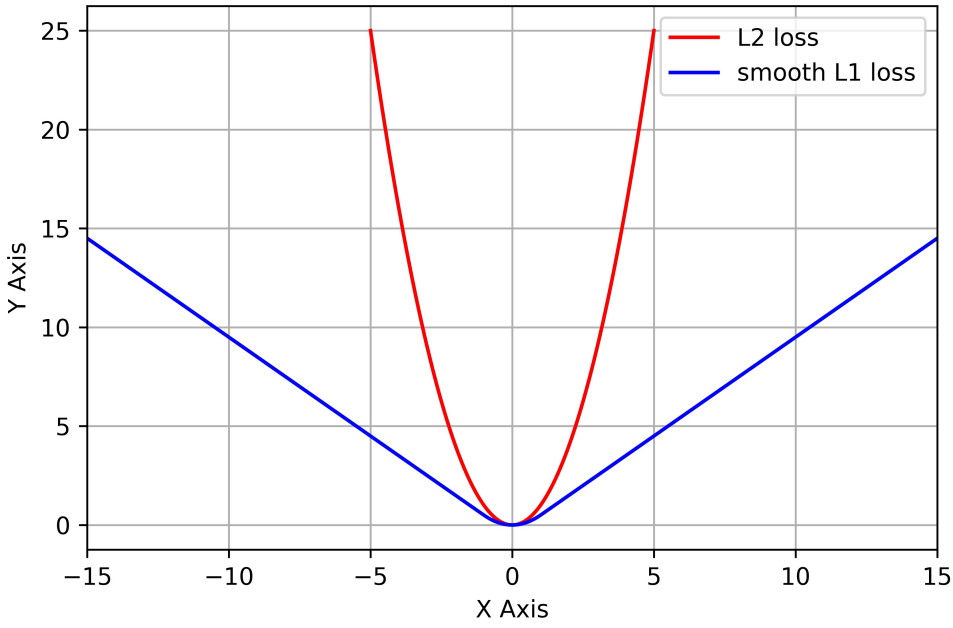


Figure 3.14: Comparison between smooth L1 loss and L2 loss.

3.4 Summary

This chapter provides the review of the mathematical background used in this thesis. In Section 3.2, the mathematical foundation of EAs are reviewed. EAs optimises the performance of population by conducting evolutionary operations in multiple iterations. There are three optional operations: selection, crossover and mutation. When applying EAs in NAS, individual presents the structure of CNN, and the fit value of individual is evaluated by the accuracy (or error rate) of the CNN, which the individual presents for. After several iterations, the population with the best performance will be selected as the final output of EAs.

Then, the technical background of CNNs is reviewed. There are four commonly used layers: FC, convolution, pooling and normalisation. The structure of CNN is built by stacking those layers. According to the development of CNN, four styles of

CNN structures are summarised. At the end of this section, loss functions, which is utilised for the optimisation of layer weights, are introduced.

The review of mathematical background is essential for the work within the following chapters. Chapter 4 investigates the architecture searching method using genetic algorithm, which is introduced in the Section 3.2. The effective manual designed CNN architectures introduced in the Section 3.3 act as the feature extractor for CNN based object detection discussed in the Chapter 5 and Chapter 6. Considering that one of the contributions in Chapter 5 and Chapter 6 is loss optimisation, it is essential to bring in the knowledge of loss functions as introduced in the Section 3.3.3 as well.

Chapter 4

Optimising the Convolutional Neural Network with Pre-Trained Weight Inheritance and Genetic Selection of Input Channels

4.1 Introduction

In this thesis, a robust approach is proposed to achieve a better trade-off between efficiency and accuracy by selecting key channels from inputs. The whole structure is self-generated without manual interference. Generating the model with a large searching space [75, 92, 93], e.g. a searching space including layer types, connection methods between layers (cascade, addition or concatenation), and the number of channels per layer, would involve a huge amount of computational cost, and the generated model could easily suffer from overspecialised. Thus, the framework of the model is constrained, which is learnt using state-of-the-art Convolutional Neural Network (CNN) models [17,18], as the "prior knowledge" of the model. Finding the best one by enumer-

ating all the combinations is impractical, because the number of permutations increases exponentially with the number of layers. Rather than a full search of all the combinations, a more effective solution to optimise the structure is to use neural architecture search (NAS), which may be based on an evolutionary algorithm (EA) [75], [76], [77], reinforcement learning (RL) [92], [94], [95], gradient descent [90], [91], or other methods [89] [87]. For the existing approaches, there exist the following drawbacks:

i. **High computational complexity and cost**: For neural evolution methods [75], [76], [77], [80], [110], denote the number of populations as I and the number of the generation as N_{G}, the whole training procedure will be required to train $I \times N_{\mathrm{G}}$ individuals. In order to achieve a higher performance, the values of both numbers should be large, e.g. 1,000 in GeNet [76]. In Real et al. [75], experiments were conducted on about 250 high-end computers. A similar limitation also exists in the reinforcement learning and gradient descent based method. The "discover" process also consumes a huge amount of computational resource. In Baker et al. [93], which adopts reinforcement learning, about 10 Graphic Processing Units (GPUs) were deployed.

ii. **Additional hyperparameters** are required to control the searching process: hyperparameters are the parameters defined before starting the learning process. Compared with manual designed methods, NAS requires more hyperparameters to define the training process, such as searching space, agent setting (reinforcement learning based), and the range of each bit on the evolutionary string (EA based).

iii. **The transferability is weak**: models trained by evolutionary algorithm (EA) perform well on the dataset, where the architectures learned, but poorly on new datasets.

To tackle these challenges, in this chapter an improved pipeline is proposed for training, in which a Genetic algorithm (GA) is employed to optimise the self-generated model. As the method focuses mainly on reducing the computational cost, a binary encoding method is applied to represent the structure of the model in a binary string where "1" and "0" indicates whether the feature is allowed to pass into a layer or

not. Three GA operations, selection, crossover and mutation, are employed to evolve the structure. After conducting the selection on each generation, poorly-performing structures are discarded. The performance of the model is measured by evaluating the accuracy on a validation/test dataset.

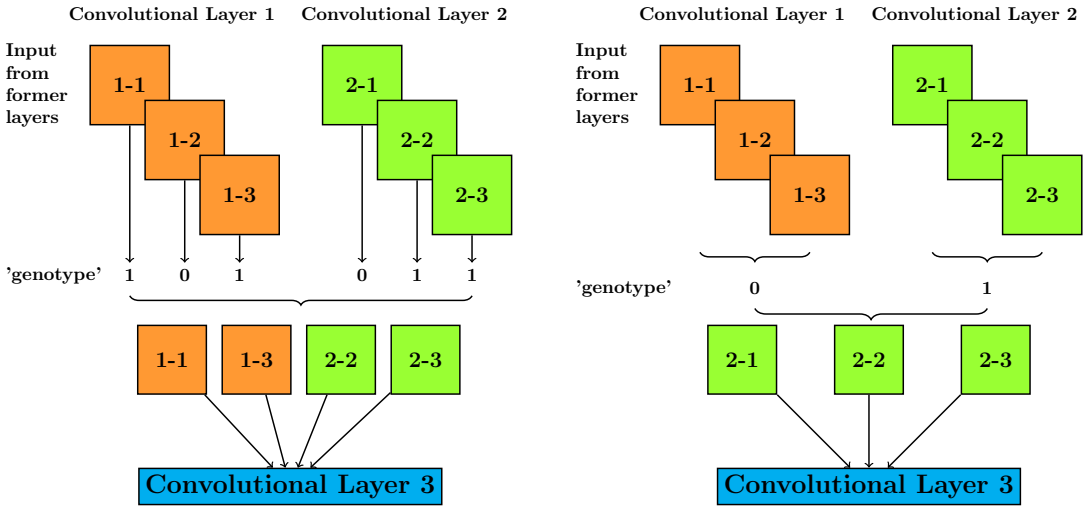
To improve the training efficiency, a variable-inheritance-fine-tune training method is proposed. Following the experimental settings in Real et al. [75], instead of training each "individual" from scratch, the proposed method applies "variable inheritance" to reduce computational cost on each "individual". This means that a reused kernel will be reinitialised using the values obtained from training on the previous generation rather than randomly generated from a Gaussian distribution. The proposed method utilises the structure of the Densely Connected Network (Densenet) as the baseline framework. Compared with the baseline, the optimised model reduces the number of parameters by up to 30% whilst maintaining the same accuracy. Although the GA procedure is mainly conducted on CIFAR-10 [86], the model produced also performs well on other datasets and is easily transferred onto large-scale datasets. The experiments can be conducted on a single GTX1080Ti GPU hence the model is very portable and affordable.

The remaining parts of this chapter are organised as follows. The design of the GA training pipeline and the experimental results are detailed in Section 4.2 and Section 4.3, respectively. Finally, some concluding remarks of this chapter are given in Section 4.4.

4.2 The Proposed Algorithm

In this section, a detailed description of the proposed approach is presented which discusses how a GA-based method can be used to remove the redundant convolutional kernels. Training from scratch without any constraints is not feasible. Even with very few constraints, the training process can be significantly speeded up whilst reducing the risk of overspecialised to the referenced dataset ("referenced dataset" refers to the dataset where each "individual" is trained). To this end, the proposed approach borrows from the frameworks of manually designed models to constrain the model architecture.

Specifically, the model is based on two state-of-the-art architectures, the ResNet



(a) Channel coding (b) Layer coding

Figure 4.1: Example of how the channel coding method (a) and the layer coding method (b) filter the input of a convolutional layer. The layer has two preceding layers with $G = 3$. A channel coding string to represent the input (from the layer "0" and layer "1") of the last convolutional layer (layer "2"), calculated using Eq. 4.1, will be "**101-011**" Thus, for the channel coding method, the length of the binary string is 6, where each bit presents a single channel of the input. While for the layer coding method, if represented as "**0-1** using Eq. 4.2". As a result, the length will become only 1/3 of the channel coding.

(Residual Network, as seen in Figure 3.12) and the DenseNet (as seen in Figure 3.13) styles with skip-connect inputs between different layers using dense connection. More details of the constraints are discussed in Section 4.2.1. As the GA is applied to evolve the architecture of the model, in Section 4.2.2, two binary coding methods are proposed based on previous work [75–77]: channel coding and layer coding. In Section 4.2.3, a simple but efficient genetic process is detailed, which consists of the selection, crossover and mutation operations.

To improve the training speed without degrading the model performance, in Section 4.2.4 two strategies are proposed: 1. a partial-training and fine-tuning strategy to obtain the fitness value of the evolved structure and 2. a pre-trained weight inheritance method to initialise the weights of the model, which is more suitable for densely connected structures and easy to implement.

4.2.1 Pre-defined Elements of the Model

The literature suggests that the architecture of a model generated by a large searching space [75,77,92,110] is more efficient than manually designed model, because the benefit of the high probability of mutation. However, the self-generated structure paradigm may be less robust than manually designed methods [75, 86]. On the other hand, generating a model without constraints consumes a huge amount of computational resources [75, 92]. As a result, in this chapter, to increase the transferability while reducing the computational cost, the following constraints are predefined by taking some manually-designed structures [17, 18, 70] as the "prior" of the proposed model:

i. Similar to ResNet [17] and DenseNet [18], the network will be split into N blocks, in which the size of the feature remains the same in each layer. Its width and height will be halved by a "transmission layer" [18] before passing to the next block.

ii. A "transmission layer" consists of 1) a 1-by-1 convolutional layer for fusing all channels and 2) a 2-by-2 mean pooling layer for down-sampling the feature maps. The depth of the output of a transmission layer will either remain the same or be halved depending on the structure of a "convolutional layer". This allows consistent comparisons with other approaches. Therefore, if a convolutional layer contains only one 3-by-3 kernel followed by a batch normalisation (BN) layer [67] and a ReLU layer [69], the depth remains the same. If a convolutional layer contains two convolutional sublayers, one with a 1-by-1 kernel and the other with a 3-by-3 kernel (both with BN-ReLU before each convolution as well), known as the "bottleneck structure", the depth will be halved. this chapter labels the "compressed-bottleneck structure" as BC, and only the BC structure will be used after the architecture is generated.

iii. The depth of the output of the convolution layer is unchanged in each block, denoted as "growth rate" G [18].

iv. The number of layers in each block is fixed during training.

v. The skip-connection is allowed. Features from different layers will be processed by

concatenation as in DenseNet [18], instead of addition [17].

4.2.2 Coding Method

To improve the training speed and attain a comparable performance while reducing the computational cost, binary coding [76] and a genetic algorithm are used to evolve the architecture of the model, instead of integer/float number coding [75, 77, 110]. For each convolutional layer, a binary string is assigned to "gate" the input, which is a concatenation of the outputs of all the previous convolutional layers. Hence, the search space corresponds to a binary string representing the structure of the model. For each bit, "1" and "0" means the associated feature can "pass" into the layer or "bypass" the layer respectively. Each bit of the string can represent either a single channel or multiple channels of the input feature map. Here this chapter proposes two binary coding methods, i.e. Channel coding and Layer coding.

4.2.2.1 Channel Coding

In channel coding, each bit represents only one channel of the concatenated feature map. Therefore, for a three-convolutional-layer block with the number of initial input channels of N_{init} and a growth rate of G, the length of the strings for all three layers are N_{init}, $N_{init} + G$ and $N_{init} + 2G$, respectively. Thus, the length of the string of the block is $3N_{init} + 3G$. As such, the length of binary string L for a single block is

$$\begin{aligned} L_{channel} &= (N_{init} + G) + (N_{init} + 2G) + \cdots + \\ &\quad (N_{init} + (C-1)G) \\ &= C \times N_{init} + \frac{C(C-1)}{2}G \end{aligned} \tag{4.1}$$

where C is the number of convolutional layers in the block.

In channel coding, the model can be more flexible than the plain DenseNet for training as it can fit a densely connected structure with an arbitrary number of layers and an arbitrary number of filters in each layer if G is small enough and N_{init} is large enough. An example is illustrated in Fig. 4.1. Assume a single CNN block has 3 layers

(labelled as "0", "1", and "2") with a growth rate of 3, a channel coding string to represent the input (from the layer "0" and layer "1") of the last convolutional layer (layer "2") will be "**101-011**".

4.2.2.2 Layer Coding

Channel coding works well when the scale of the model is small. When the number of layers and the growth rate are large, the length of the string for the whole model becomes very long and difficult to train. For a model with $N_{init} = 24$, $C = 12$ and $G = 12$, when the number of blocks is 3 ($N = 3$), the length of the binary string for the first block, calculated from Eq. (4.1), will be 1080. An effective way to reduce the length of the binary string is "bit-sharing", which is referred to as "Layer coding" in this chapter. In layer coding, each bit represents the status of the channels from the same convolutional layer. The length of the layer coding can be determined by

$$\begin{aligned} L_{layer} &= 1 + (1+1) + (1+2) + \cdots + \\ &\quad (1 + (C-1)) \\ &= \frac{C(C+1)}{2}, \end{aligned} \tag{4.2}$$

where C is the number of convolutional layers in the block.

The layer coding above can significantly reduce the training difficulty, however, it only fits to a model whose layer depth is the magnitude of the growth rate (ignoring the initial filter size). Taking the same model in Fig. 4.1 for example, if apply layer coding to the third layer, the length will become only 1/3 of the channel coding if represented as "**0-1**".

Although channel coding is more flexible, the length of its string is limited by the fixed growth rate and will increase the training difficulty under limited training steps. On the other hand, the model generated by layer coding can support arbitrary growth rates according to the scale of the dataset. Experimental results in Section 4.3 indicate that channel coding leads to slightly higher accuracy than the layer coding on the MNIST dataset but performs worse when transferred to the CIFAR-10 dataset. Thus,

Genetic process of structure evolution

I: initial population
T: The number of generations to conduct evolving process
P: parent population is denoted

Weight Initialization
Fully train the baseline (i.e. DenseNet) on the reference dataset **D**
Individual Initialization
(1) Generate individuals in **I** via **B(0.5)**
(2) Partially train and evaluate the individuals on **D**
for **i** in range(**T**):
    P = [ ]
    for **ii** in range(length(**I**)):
        Random select **S** individuals from **I**
        Select the best individual, and save it into **P**
    Crossover
    for **iii** in range(length(**I**)/2):
        Conduct crossover for **P**[iii] and **P**[$iii + 1$] with $\mathbf{P_c}$ and $\mathbf{Pb_c}$
    Mutation
    for **iiii** in range(length(**I**)):
        Conduct mutation for **P**[$iiii$] with $\mathbf{P_m}$ and $\mathbf{Pb_m}$
    Evaluation
    Evaluate **P** on **D**
    Best Selection
    Save and store the best individual in the generation **i**

Algorithm 1: A Python style pseudocode of the genetic process in the proposed method. For the selection operation, tournament selection is utilised. After that, in the crossover part, bits from different individuals but at the same position in the string are randomly "swapped". The probability of crossover for each pair is P_C and the probability of crossover of a bit is denoted as Pb_C. During crossover, the structures of a pair are modified simultaneously, and mutation are conducted to bring more variance to the structure. The probability of mutation for each pair is P_M and the probability of mutation of a bit is Pb_M.

to balance the computational cost and the accuracy, the proposed method will apply the layer coding method to code the model.

4.2.3 GA-based structure evolution

The genetic process for evolving the architecture of the model is given in Algorithm. 1. In the Weight initialisation step, after pre-training the baseline model, weights will be initialised using the values from the baseline model which will be detailed in Section 4.2.4. As the possible values of each bit in the binary string are '0' and '1',

each bit is randomly initialised using the Bernoulli distribution with a probability of 0.5, i.e. B(0.5). The fitness value for each "individual" in the first generation is the classification accuracy of the model on the validation dataset D after fine-tuning of training. The number of individuals, i.e. "population size" I, in the first generation will remain the same during the following iterations. The evolving process will conduct T generations. Following the suggestions in [76, 77], three evolutionary operations are assigned to evolve the architecture of the model: selection, crossover and mutation.

Specifically, the initialization method is kept as in [76, 77] for making a fair comparison. However, such method discards half of features at the first generalisation, which might increase the number of searching generations. This leaves future works to optimise the initialisation method.

For the selection operation, tournament selection [159] is utilised as commonly used in [75, 76]. After that, in the crossover part, bits from different individuals but at the same position in the string are randomly "swapped". The probability of crossover for each pair is P_C and the probability of crossover of a bit is denoted as Pb_C. During crossover, the structures of a pair are modified simultaneously, and mutation are conducted to bring more variance to the structure. The probability of mutation for each pair is P_M and the probability of mutation of a bit is Pb_M. To avoid overspecialised, the first and the last bits are automatically flipped to "1" rather than discarding the layer as did in [75–77]. Experimental results in Real et al. [75] indicates that, without this constraint, the model generated on the CIFAR-10 has fewer layers. However, when applying on the CIFAR-100 dataset, the generated structure becomes suboptimal. As a result, the number of layers should not be stabilised during the evolution process.

4.2.4 Pre-trained Weight-inheritance based Individual Training Strategy

After the structure evolution, some "individuals" need to be retrained to determine the fitness value. Using the approach, which is partially trained from scratch [77, 110], is hard to converge the weights to the global optimal, causing underestimation or overestimation of the model. Meanwhile, the fully trained method [76] has an extremely

high computational cost and takes too long to train. In the original weight inheritance partially training method, weights inherit their values from the last update and are reused during the training instead of initializing from scratch, generating a well-performed model structure. The original weight inheritance method [75] guarantees that reused weights are actually fully trained, without being constrained by the limited training steps of each generation. However, this strategy assumes that the model takes the residual structure as baseline without implementing a concatenation operation. In residual structure, the output can be interpreted by

$$X_{n+1} = F(X_n)_n + X_n, \tag{4.3}$$

where X_n is the input of the nth layer, $F(*)_n$ is the convolution operation of layer n which may contain two convolution sublayers with a batch normalisation and ReLU connected after each of those [69], or three convolutional sublayers known as "bottleneck" [17]. In a densely connected structure, the output can be formatted as

$$X_{n+1} = F([X_n, X_{n-1}, \ldots, X_1])_n, \tag{4.4}$$

where $[X_n, X_{n-1}, \ldots, X_1]$ is the concatenated inputs from all former layers of the layer n.

As seen in ResNet, the convolutional operation minimises the residual error between the input and the output, hence its magnitude is usually very small. As suggested in Veit et al. [19] that the performance of ResNet will not be significantly affected when removing several layers. Thus, the modification of the weights in a layer only slightly affects the outputs of the following layers and can be easily fine-tuned by a few training steps. However, in a densely connected structure, the output of a layer will be connected as part of the input to all the following layers, which has a direct affect. The modification of a layer in a densely connected structure will strongly affect the structure of the model, leading to possible oscillation of the optimisation process. Experimental results in Section 4.3 shows that the performance of the densely-connected model trained using the original weight inheritance strategy is almost equivalent to that

trained from scratch. The fitness values of individuals oscillate with a small margin between generations. On the other hand, the original weight inheritance method must record all the latest trained variables regardless of whether they will be reused or not, which is flexible for models without a predefined framework. However, in the proposed training pipeline, the total number of variables is fixed, and the framework is predefined. Recalling the weights and reformatting the structure each time the training starts is inefficient and redundant.

To tackle this issue, a pre-trained weight inheritance strategy is proposed for training each "individual". Before the evolutionary procedure starts, this strategy first sets all bits on the genotype string to one as the baseline and fully train the baseline. The length of the model is fixed, and its growth rate remains the same during the procedure, the structure of which is the same as the plain DenseNet. When the evolutionary procedure starts, the weights from each "individual" will be initialised using the values from the well-trained baseline instead of from scratch. Each "individual" will be fine-tuned for several epochs to optimise the fitness value, i.e. each "individual" is partially trained. All fine-tuned weights will not be inherited by the next generation, and the weights of the next generation will be initialised using the values from the baseline as well. To achieve this, during the fine-tuning procedure, the binary string will be partitioned back onto each layer (when conducting evolutionary steps, binary strings from different layers are concatenated together as discussed in Section 4.2.2.2) and act as a binary mask for each channel of the input. For each filter, the forward-propagation process and the back-propagation process are interpreted as follow

Forward :

$$\begin{aligned} X_{n+1}^{m} &= \sum_{i=1}^{D} bin_i \times F(X_i, K_i^m)_n \\ &= \begin{cases} \sum_{i=1}^{D} F(X_i, K^m{}_i)_n, & if\ bin_i = 1 \\ 0, & if\ bin_i = 0 \end{cases}; \end{aligned} \tag{4.5}$$

Backward :

$$\begin{aligned}\frac{\partial Loss}{\partial K_i^m} &= \frac{\partial Loss}{\partial X_{n+1}} \times \frac{\partial X_{n+1}}{\partial K_i^m} \\ &= \frac{\partial Loss}{\partial X_{n+1}} \times bin_i \times \partial F(X_i, K_i^m)_n \\ &= \begin{cases} \frac{\partial Loss}{\partial X_{n+1}} \times \partial F(X_i, K_i^m)_n, & if\ bin = 1 \\ 0, & if\ bin = 0 \end{cases}\end{aligned} \tag{4.6}$$

where $\mathbf{X^m_{n+1}}$ denotes the $\mathbf{m_{th}}$ channel of the input for the layer $\mathbf{n+1}$; $\mathbf{D}$ is the depth of the input $\mathbf{X}$; $\mathbf{bin_i}$ denotes the binary bit of the $\mathbf{i_{th}}$ channel of $\mathbf{X}$ (channels generated by the same layer will have the same bit value, as described in Section 3.2.2); $\mathbf{K^m_i}$ denotes the $\mathbf{m_{th}}$ channel of the kernel and $\mathbf{F(*)_n}$ denotes the convolutional operation of the layer n.

Let the size of the input X of a layer be $32 \times 32 \times 3$, it can be generated by three convolutional layers beforehand, and the size of baseline kernel (denoted as K) of the layer is $3 \times 3 \times 3 \times 24$. The length of the corresponding binary string will be 3. If the string is "$1 - 1 - 0$", it is obvious that the first two channels of each filter can be trained using the back-propagation algorithm. However, the last channel of each filter is excluded from both forward propagation and backpropagation as its input is an all-zero feature map.

4.3 Experimental Results and Discussions

4.3.1 Datasets and pre-processing

4.3.1.1 MNIST

The MNIST [2] is a handwritten digit dataset for recognition tasks of digits from 0 to 9, which contains 60,000 images for training and 10,000 images for testing. As the number of the epochs for fully training is small, e.g. 20 epochs in this section, the experiment

does not split a validation dataset during training. No data augmentation or pre-processing is applied during training for this dataset in order to reproduce consistent conditions to other approaches.

4.3.1.2 CIFAR-10 and CIFAR-100

The CIFAR-10 and CIFAR-100 datasets [86] contain colored natural images in 10 classes and 100 classes, respectively. Both datasets have 50,000 images for training and 10,000 images for testing and each image has 32×32 pixels. A split of 5,000 images from the 50,000 training images are collected as the validation dataset and the remaining 45,000 images are kept for training. The data augmentation method for the two CIFAR datasets is the same as used in [17,18,66,69,104,195,201–203], where 4 pixels are padded on each side of the original image or its horizontal flip. A 32×32 image is cropped randomly from the padded image. When testing, the input images remain the same as the original without padding and randomly cropping. For pre-processing, pixel-based normalisation is used to normalise the image using the channel means and the channel standard deviations. To make a fair comparison with other methods [18, 69, 76, 77, 80], the model uses the CIFAR-10 training dataset as the reference dataset (in the genetic process) to generate the structure of model and the benchmark dataset (in the model evaluation process). After that, the structure is fixed and the CIFAR-10 and CIFAR-100 datasets are chosen as benchmark datasets to evaluate the performance of the model as in [18, 69, 70, 104, 111, 201]. For the CIFAR-10 dataset, the model is not trained on the validation dataset when it acts as a reference dataset. When it acts as a benchmark dataset, the model is trained on the validation dataset only at the final epoch as in DenseNet [18].

4.3.1.3 SVHN

Similar to the MNIST dataset, the Street View House Numbers (SVHN) dataset [204] is also an image dataset of $0-9$ digits with a size of 32×32 pixels each. In addition to 73,257 training images and 26,032 testing images, there are 531,131 images for extra training. the SVHN dataset is only utilised as the benchmark dataset. For a fair

comparison with other models, no data augmentation step is applied and the model is trained using all images from the training dataset and the extra training dataset except for 6,000 images used as a validation dataset as in [18, 66, 70, 202, 205, 206]. The range of the images is normalised from $[0, 255]$ to $[0, 1]$ as data pre-processing [18, 207]. The model loads the weights with the lowest validation error during training and evaluate it on the test dataset.

4.3.1.4 ImageNet

The ILSVRC 2012 classification dataset [15] is a large-scale image dataset. which consists of 1.2 million training images and 50,000 validation images, uniformly distributed in 1,000 classes. The augmentation method is used as in [17, 18, 69], where the per-pixel normalised image is resized without modifying the $width/height$ ratio as scale augmentation. A 224×224 sub-image is randomly cropped from the scaled image or its horizontal flip with color augmentation. As most of methods evaluate their results on the validation set [16–18, 69, 85, 90, 109] the proposed model is also tested on the validation set in the same way, and the single-crop classification errors are reported.

4.3.2 Channel Coding vs Layer Coding

To fairly compare the performances between the channel coding and the layer coding methods, the structure of the model is generated using the MNIST dataset and evaluate the results on both the MNIST and CIFAR10 datasets. the model are assigned by 3 blocks, i.e. $N = 3$, and 8 convolutional layers without a bottleneck structure in each block. The number of outputs of each convolutional layer is 8, i.e. $G = 8$. The depth of the input image is expanded to 16 using a 3-by-3 convolutional layer before entering the first block. The model is completed with a global average pooling, a 10-output fully-connected layer, and the softmax output. As the MNIST is easy to train, weight inheritance strategy is not applied, and every individual is fully trained from scratch.

Training implementation. Weight decay is assigned by 0.0001 and the model is optimised using Stochastic Gradient Descent (SGD) with a momentum of 0.9. Weights are initialised by using the method in [18, 111]. These models are trained with a batch

size of 64 on a single GTX1080Ti GPU, where each "individual" is trained by 20 epochs on the MNIST dataset. learning rate is initialised by 0.1, then is divided by 10 at 5 epochs, 10 epochs and 15 epochs. When the generated architecture of the model is transferred to the CIFAR-10 dataset, the structure is fixed. Then, the structure-fixed model is fully train from scratch. Following the settings on [18], the model is trained by 300 epochs on the CIFAR10 dataset and the learning rate is initialised as 0.1, divided by 10 at 150 and 225 epochs with the same weight decay and batch size as used on the MNIST dataset.

Genetic hyperparameters. Following the approaches used in [75, 76], the population size is $I = 20$. The number of generations is $T = 30$ with the sample size of tournament selection $S = 3$. The probability of the pair-wise crossover and the bit-wise crossover for each "individual" are $P_C = 0.2$ and $Pb_C = 0.2$ respectively. The probability of the individual mutation and the bit mutation are $P_M = 0.8$ and $Pb_M = 0.05$, respectively. It takes about 10 GPU days to conduct each genetic process, and the experimental results are given in Fig. 4.2.

A similar setting for the number of generations and the population size can be found in [75, 76], the number of population is 20 and the number of generations is 50. The generated model has no specific gains after 30 generations. In [75], the method with a population size of 2 has the best result at an early stage. For the proposed model, when the number of generations is set to 50, the best model is found from the 24th generation as shown in Fig. 4.3.

This is because: 1). MNIST and CIFAR are small datasets, for which it is easy to find the optimal models; 2). Compared with manually designed models such as DenseNet and ResNet with over 100 layers, the model sizes of self-structure generation from [75, 76] and the proposed approach are relatively small, i.e. around 20-30.

the model is fully trained with the best structure on the CIFAR-10 dataset, where an accuracy of **88.3%** and **92.3%** are achieved from the channel-coding based model and layer-coding based model, respectively. When tested on the reference dataset (MNIST), channel coding slightly outperforms layer coding but underperforms the layer coding by a large margin on transfer learning. a similar experiment is also conducted on the

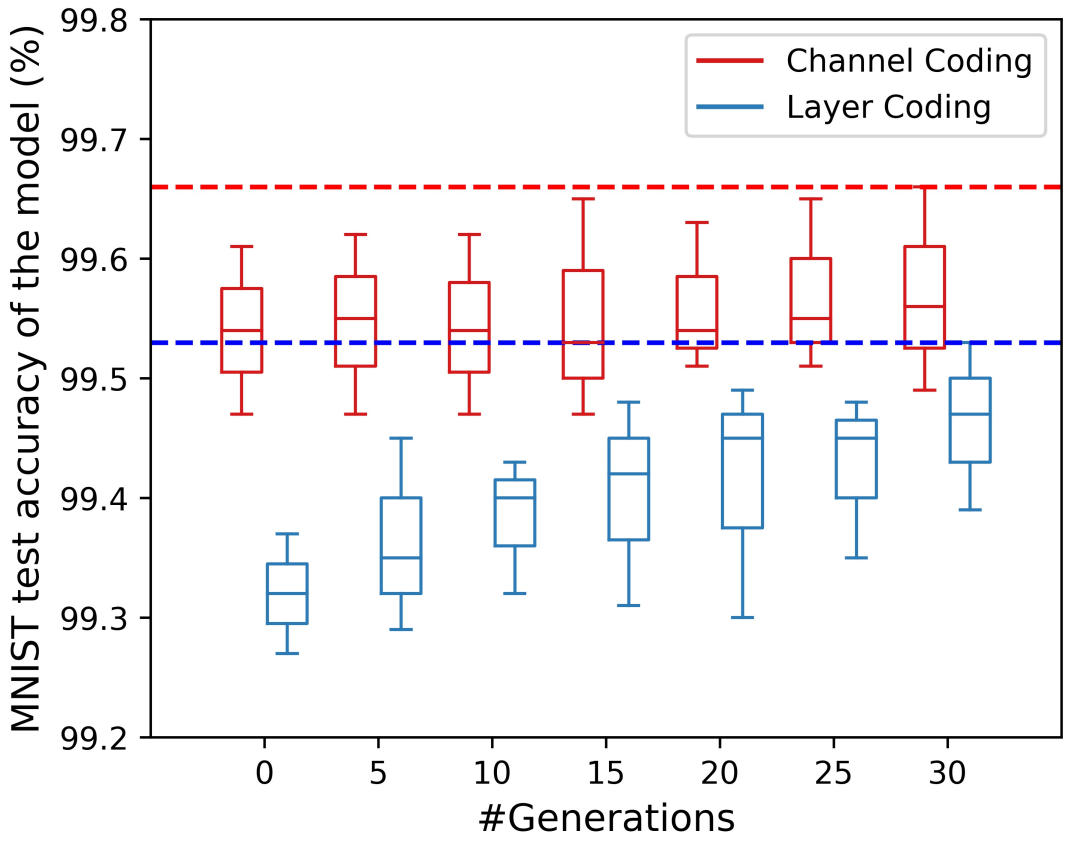


Figure 4.2: The distributions of Test accuracy (%) in each generation on the MNIST dataset, using the two coding methods. The genetic process conducts 30 generations for both methods separately. The best performance the through the whole generations is highlighted via a dashed line. When comparing with the best architecture between two methods, it is obvious to see that the result trained using channel coding method (denoted by dash red) outperforms the layer coding method-based training (denoted by dash blue) by 0.13%. However, when compared with the accuracy growth between generations, layer coding outperforms channel coding.

CIFAR-10 dataset for both methods, and the results are shown in Table 4.1. It turns out that the channel coding method requires **66%** more generations than the layer coding to reach the same test accuracy (**92.6%**).

In summary, the channel coding method is capable of generating a more competitive model than the layer coding as long as the number of generations is sufficiently large, while layer coding performs better at transfer learning. To balance the training cost and the performance of transfer learning, layer coding is applied in all the following

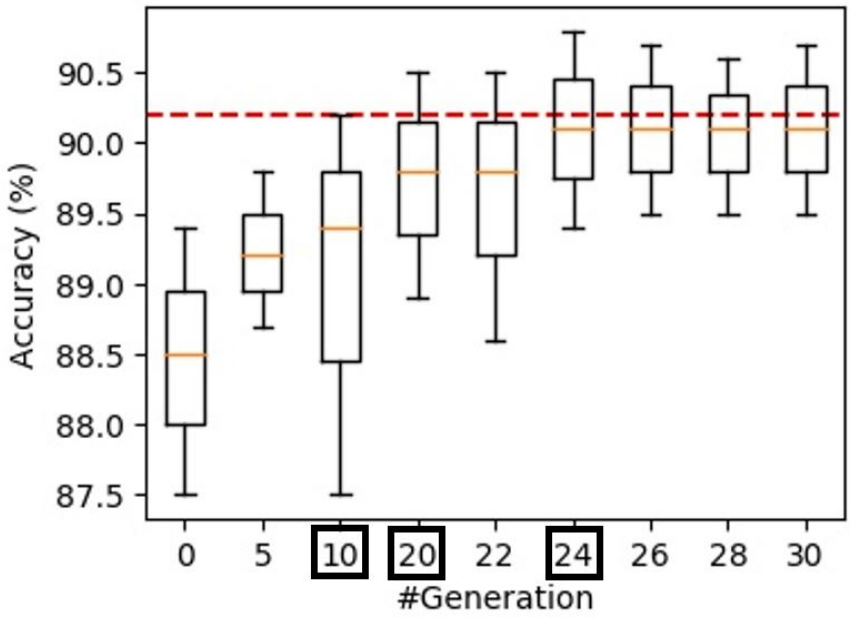


Figure 4.3: Test accuracy (%) of the model on the CIFAR-10 dataset using the proposed training pipeline given in Section 3 with the genetic process conducted over 30 generations. The baseline (**Generation 0: 90.2%**) is denoted using the red dash line and highlight the key generation IDs using a black box. With the performance comparable to the baseline at **Generation 10** (**90.2%**), the model outperforms the baseline starting from **Generation 20** (**90.5%**) and achieves the best among all individuals at **Generation 24** (**90.8%**).

Table 4.1: Relationship between the Number of Generation and the Test Accuracy on CIFAR-10

Test accuracy of the fully trained best structure(%)		
#Generations	Channel coding	Layer coding
10	90.40	90.30
20	90.87	91.10
30	91.23	**92.63**
50	**92.69**	92.60

experiments.

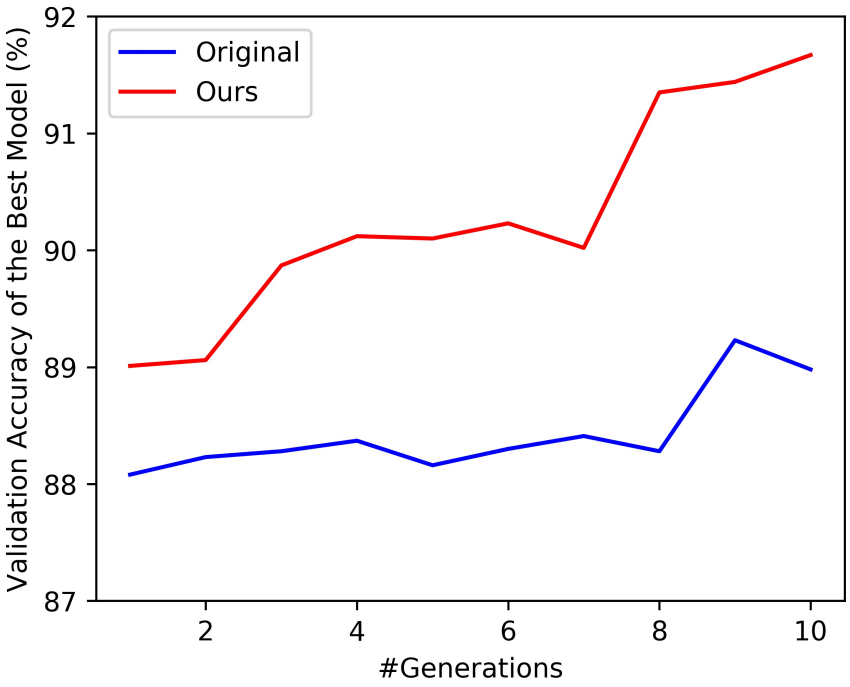


Figure 4.4: Comparison of test accuracy on the CIFAR-10 dataset using the original (blue line) and the proposed (red line) weight inheritance methods.

4.3.3 Weight Inheritance Methods

This section compares the difference between the original weight inheritance and the proposed pre-trained weight inheritance strategy, where the layer coding method is utilised to derive the architecture of the model during the genetic process. The baseline structure of the model is the same as discussed in Section 4.3.2 with the same weight decay. the CIFAR-10 dataset is used as the reference dataset D and implement the Adam optimiser [208] in a weighted training pipeline. For the original method, each "individual" is trained around 30 epochs with a fixed learning rate of 0.01 and evaluated on the validation dataset. After the evolutionary steps, each model inherits the weights trained from the last generation. For the proposed pre-trained weight inheritance method, it first fully trains the model with all bits set to 1 in the binary string under the same learning rate fixed at 0.01, which makes it similar to the DenseNet. All weights are stored in a "checkpoint model". During the fine-tuning procedure, each "individual" is trained for 30 epochs, which is the same as the original method. The

Table 4.2: Architecture of the Model

Block	**Layer ID**	**Output points to**
1	0	1, 2, 3, 6, 8, 9
	1	2, 3, 4, 5, 7, 9, 10
	2	6, 8, 9
	3	5, 6, 8, 9
	4	6, 8, 10
	5	8, 9, 10
	6	10
	7	10
	8	-
	9	-
	Transition 1	
2	0	1, 2, 6, 7, 8, 10
	1	2, 5, 6, 9
	2	-
	3	7
	4	7, 10
	5	6, 9, 10
	6	7, 9
	7	-
	8	-
	9	-
	Transition 2	
3	0	1, 3, 4, 5, 6, 7, 8, 9, 10
	1	4, 7
	2	3, 4, 5, 7, 8, 9, 10
	3	5, 7, 9, 10
	4	5, 6, 7
	5	6, 10
	6	7
	7	-
	8	9
	9	10
Global Average Pooling		
FC-Softmax		

learning rate decays by 10, i.e. 0.001, to avoid the weights oscillating between the local minima and the global minima. The genetic hyperparameters are the same as in Section 4.3.2 except that the number of generations is reduced to 10, i.e. $T = 10$. Each

of the genetic training procedure takes about 10 days on a single GTX1080Ti.

Experimental results are summarised in Figure 4.4. As seen, the increment of the test accuracy from the validation dataset is insignificant during the training process when using the original weights inheritance method. The weights of the model can inherit very limited information of the last generation. The reason behind this is the structural difference of the models, where the original weight inheritance method is designed for the residual structure while the proposed model is for the densely connected structures. In addition, the proposed pre-trained weight inheritance method is more efficient. The weights can therefore inherit more information from the last generation using the proposed method, and the model can even reach the same test accuracy as the baseline. The experiments in Section 4.3.4 indicate that if the model is evolved by more generations, the self-structure-generating model can even outperform the baseline but using fewer parameters.

4.3.4 Generating the Model Architecture on the CIFAR-10 Dataset

In the previous two sections, the coding methods and the individual weight optimisation strategies are introduced. In this section the final training pipeline of the proposed GA based self-generating structure method. Following the work in [75,76,92], the proposed method assigns the CIFAR-10 dataset as the reference dataset D. For the baseline design, the number of layers of the model is slightly upscaled for transfer learning on different scales of the datasets while the number of the growth rate is downscaled to improve the training speed. As a result, the model consists of 10 convolutional layers in each block without the bottleneck structure and the growth rate is assigned as $G = 4$. Thus, the total length of the binary string is 162. The depth of the initial feature map is 32. Other settings are the same as that in the per-training weight inheritance strategy as discussed in Section 4.3.3.

Evolutionary results of each generation are presented in Fig. 4.3, where the test accuracy of the baseline is 90.23%. After 10 generations, the derived model reaches a similar accuracy as the baseline. After 20 generations, it outperforms the baseline, achieving an accuracy of 90.8% on the validation dataset at the 24th generation. Indi-

viduals in the later generations perform worse than the former generations due to the high individual mutation rate P_M. A high mutation rate brings benefits of generating a high-performance model but fails to guarantee the mutation variance. the details of the best individual is displayed in Table 4.2, which summarises the structure of the model and layer connections in each block. The model generated using the GA consists of three blocks with 10 convolutional layers in each block. At the end of the first two blocks, a transition layer is connected to fuse the feature map. The third block consists of a global average pooling layer and a fully connected layer to make the final prediction. Layer id "0" indicates the input of the block and the "Output points to" column denotes the layers to which the output of the current layer will be fed. "-" means that the output of the current layer will only be fed to the transition layer. As the output of the last convolutional layer can only be fed to the transition layer, the last layer is omitted in the table.

The first block, i.e. the most densely connected one is visualised in Fig. 4.5. As seen, the connection path of the model is sparser than the baseline (Densenet). The feature maps from the first three layers are frequently reused in each block, and the reuse frequency of each layer output is reduced with the increasing layer id. This validates the importance of the shallow feature map in the final classification. This phenomenon is also found in many manually designed CNN models [6, 17, 18].

4.3.5 Classification Test on Multiple Datasets

With the selected best structure, it is also tested on multiple benchmark datasets. Apart from the single convolutional layer structure, the "bottleneck" structure is also considered in the model, where the number of layers is either 34 (without bottleneck) or 64 (with bottleneck). The genetic generated densely connected model is denoted as "GADNet" for the non-bottleneck structure and "GADNet-BC" for the bottleneck structure. The proposed model will be compared with other manually designed models, especially the variance of the ResNet and the DenseNet, as well as other self-structure-generating approaches. The proposed method is trained and tested by 5 times. The average result is displayed when compared with other methods.

Table 4.3: Test Error Rates on Multiple Datasets

Model	#Params (M)	#Layers	C10	C100	SVHN
Manually designed methods					
HighwayNet [102]	-	-	7.72	32.39	-
FractalNet [104]	38.6	21	4.60	23.73	1.87
ResNet by [70]	1.7	110	6.41	27.22	2.01
with Stochastic Depth	1.7	110	5.23	24.58	1.75
Wide ResNet [206]	36.5	28	4.17	20.50	-
With Drouput	2.7	16	-	-	1.64
DenseNet [18]	1.0	40	5.24	24.42	1.79
DenseNet (K=12)	4.0	100	4.10	20.20	1.67
DenseNet (K=24)	27.2	100	3.74	19.25	**1.59**
DenseNet-BC (K=40)	25.6	190	**3.46**	**17.18**	-
Evolutionary algorithm methods					
Evolution-C10 [75]	5.4	-	5.40	-	-
Evolution-C100	40.4	-	-	23.00	-
CGP-CNN [84]	3.9	-	23.48	-	-
CGP-CNN (ResSet)	0.8	-	23.47	-	-
GeNet#1 [76]	-	12	7.19	29.03	1.99
GeNet#2	-	12	7.10	29.05	1.97
The proposed methods					
GADNet (G=12)*	0.7	34	6.03	26.00	1.81
GADNet (G=12)	0.6	34	5.71	25.50	1.74
GADNet (G=32)*	4.8	34	4.39	21.83	1.65
GADNet (G=32)	3.9	34	4.35	21.10	**1.61**
GADNet-BC (G=32)*	2.4	64	4.06	21.00	1.71
GADNet-BC (G=32)	2.0	64	**4.02**	**20.50**	1.70

(* indicates the baseline of the model)

4.3.5.1 Test on the CIFAR-10, CIFAR-100 and SVHN datasets

Experimental results, shown in Table 4.3, indicate that the proposed model has advantages over manually designed and other self-structure-generating models on accuracy,

transfer capability, parameter saving and efficiency of feature reuse as explained below.

Accuracy. The best-performance model with 64 layers only lags the state-of-the-art method (DenseNet-BC (K=40) with $L = 190$) by no more than 0.6% on the CIFAR-10 and SVHN datasets and no more than 2% on the CIFAR-100 dataset. This is due to the large margin of the length between models (169 layers for the best DenseNet, which is 2.6 times longer than the proposed model), as the proposed model surpasses the baseline by a slight margin. It is conceivable that the proposed model can reach a more com-parable result by extending the length of the model. Apart from the DenseNet, the performance of the proposed model on the CIFAR-10 dataset surpass FractalNet with drop-path regularisation [104] and wide ResNet by 15% lower and 5% lower respectively. On the CIFAR-100 and SVHN datasets, the proposed model produces similar results to the wide ResNet.

Saving of parameters and computational cost.The relationship between the number of the parameters and the test accuracy on the CIFAR-10 dataset is shown in Fig. 4.6. As seen, the proposed method outperforms state-of-the-art manually designed models, while using significantly fewer parameters and less computation time to achieve the comparable results. When comparing the parameter requirement with the densely connection and the genetic connection, they show a similar trend. For instance, the best model lags the best DenseNet by 0.5% but with 92.2% fewer parameters. As the number of FLOPs (floating point operations) required is in proportion to the number of parameters, models generated using the GAs can, as a result, reduce the computational cost significantly while achieving comparable classification accuracy. This section also measures the evaluation times (forward propagation only) on the CIFAR-10 dataset for both the DenseNet and the proposed method, as shown in Fig. 4.7. As seen, under a similar test accuracy, this method only requires 1/3 of the computational time than the DenseNet. Compared with the best model of DenseNet, i.e. DenseNet-BC190 (K=40), the proposed model GADNet-BC64(G=32) lags 0.5% on accuracy but improves the evaluation speed by about 6 times, directly benefiting from the fewer parameters of the model.

High efficiency of feature reuse. Following the measurement method in DenseNet

[18], this section measures the efficiency of feature reuse in each convolutional layer using the absolute average weights of the input channels. A higher value in the layer l whose input is from the layer s denotes that the feature map from the layer s is strongly used in layer l. In Figure 4.8, this section plots the heat maps of a 34-layer model using both densely connected (baseline, shown in (Figure 4.8a)) and the genetic connected (the proposed method, shown in (Figure 4.8b)) methods. Compared with the densely connected method, the proposed method maintains almost all the strong features whilst discarding the weakly used features from the input of each convolutional layer. This has indicated that the GA improves the efficiency of feature reuse significantly. There are also few "cold zones" in the heat map of the proposed method, possibly due to the hyperparameter settings of the GA, which is left for future investigation.

4.3.5.2 Test on large-scale dataset of the ImageNet

This section also evaluates the proposed model on the ImageNet by upscaling its depth and width. As the size of the images in ImageNet is far larger than the CIFAR and SVHN dataset, the feature maps of the model is down sampled as in [14, 16–18, 76] by connecting a densely connected block with 6 bottleneck layers [18] at the input end of the model. As a result, the first block is densely connected and the following three are genetically connected. different growth rates are assigned in each block, which are 32, 32, 64 and 128 for each of the four blocks, respectively. Experimental results are given in Table 4.4. As seen, the proposed genetic connection model (denoted as "GADNet-expand") can yield comparable results to manually designed methods as well as other self-structure-generating approaches. This has validated that this model is robust even with the scale variance, while most of the self-generating structure methods may fail on large scale classification tasks such as ImageNet.

Comparison with other manually designed light-weight models. This section also compares the proposed approach with other manually designed light-weight models, such as SqueezeNet [62], MobileNet [63] and ShuffleNet [64]. As shown in Table 4.4, The proposed method leads SqueezeNet by 9.1% on top-1 but with a **54%** parameter reduction. Although the number of parameters is the same as ShuffleNet,

Table 4.4: Validation Error Rate on ImageNet

Model	#Params (M)	Top-1 Err.	Top-5 Err.
State-of-the-art methods			
AlexNet [14]	60	43.45	20.91
VGG-19 [16]	144	27.62	9.12
ResNet-18 [17]	11.7	30.24	10.92
ResNet-34 [17]	21.8	26.70	8.58
ResNet-152 [17]	60.2	21.69	5.94
DenseNet-121 [18]	7.2	25.35	7.83
DenseNet-169 [18]	13.0	24.00	7.00
Manually designed light-weight methods			
SqueezeNet [62]	7.7	41.2	18.0
MobileNet-224 [63]	4.2	29.4	10.5
ShuffleNet [64]	**5.0**	**29.1**	**10.2**
Self-structure-generating methods			
NASNet-A (4@1056) [108]	5.3	26.0	8.4
GeNet [76]	30.6	27.87	9.74
LEMONADE [85]	-	28.3	9.6
SNAS [97]	4.3	27.3	9.2
DTARS [90]	4.7	26.7	8.7
FBNet-C [96]	5.5	25.1	-
MnasNet-A3 [95]	5.2	24.3	6.7
AmoebaNet-A3 [109]	469	**17.1**	**3.4**
GADNet-expand (Proposed)	5.0	**27.35**	**8.91**

GADNet outperforms it by 1.3%. When compared with the proposed method, the MobileNet-224 has 16% less parameters whilst its classification accuracy is 1.6% lower than GADNet. The existing manually designed methods optimise the structure by either optimising the convolutional operation (MobileNet, ShuffleNet) or connecting additional short-cut (SqueezeNet), which are all local optimisation methods. On the contrary, GADNet optimises the connection routine across the whole network, which is a form of global optimisation. The results in Table 4.4 have clearly indicated that the global optimisation is more efficient than local optimisation. Nevertheless, the proposed

Table 4.5: GPU Requirements of NAS and the proposed method

Model	#Params (M)	#GPUs	C10 Err. (%)
NAS methods			
MetaQNN [93]	-	10	7.02
NAS v3 [92]	7.1	39	4.47
Progressive NAS [209]	3.2	100	3.63
NASNet-A [108]	3.3	450	3.41
AmoebaNet [109]	3.2	450	3.34
DARTS [90]	3.9	1	2.95
LEMONADE [85]	13.1	16	**2.58**
The Proposed Method			
GADNet-EXT	2.0	1	3.34
GADNet-EXT+	5.3	1	**2.91**

(The model is searched using DARTS by 5 runs and the best performance is released)

method did not optimise the convolutional structure. It is deduced that a combination of both global and local optimisation methods can further improve the optimisation performance.

4.3.5.3 NAS vs. GA

Specifically, this section compares the proposed GA based optimisation method with Neural Architecture Search (NAS). Most existing NAS methods [90, 91, 94, 97, 108, 109] report their results based on additional enhancements such as cutout [210], path dropout [108] and auxiliary towers [90]. For a fair comparison, these enhancements are also adopted on the "GADNet-BC (G=32)"and the enhanced model is labeled as "GADNet-EXT" in Table 4.5. Results in Table 4.5 indicate that the model derived by the proposed method reaches comparable results as those from NAS, but with fewer parameters. For example, the proposed method achieves a comparable error rate as AmoebaNet, but with a reduction of 37.5% on parameters. A similar conclusion can

be seen when compared on ImageNet, as shown in Table 4.4. On the other hand, the proposed method aims to improve the computational efficiency of CNN while NAS is designed to improve the accuracy, causing the difference of improved performance. This section also compares, shown in Table 4.5, the computational resources required by NAS and the proposed GA method. As can be seen, some NAS methods need a RNN to guide the structure of CNN, which naturally requires more GPUs than GA. Some of the NAS methods even require 100+ GPUs [108, 209], raising the impractical difficulty of implementation, particularly in research labs. Although LEMONADE and DARTS outperform the proposed method by 0.39%, at least 75% increase on the parameters is significant. As the performance of deep learning models is closely related to its scale [16–18], to validate the transferability under a larger scale, the model is upscaled by adding 10 densely connected layers at the end of the third block. At the same time, this section sets the growth rate as 40, which is denoted as "GADNet-EXT+" in Table 4.5. The results indicates that by using more layers, the proposed method achieves a similar performance as DARTS. When comparing with LEMONADE, the error rate lags by 0.33% but saves about 66% parameters, reaching a good balance between the computational cost and the accuracy. On the other hand, NAS based approaches need adaptation and additional training for testing on a large dataset such as ImageNet. In contrast, the proposed method does not need such adaptation and additional training when migrating from a small dataset to larger ones, where the scalability can help to significantly save the computational cost.

4.3.5.4 More implementation details

The batch size is 64 for all datasets (CIFAR, SVHN and ImageNet) and each model is trained using the stochastic gradient descent (SGD) optimiser with a Nesterov momentum [211] of 0.9 without dampening. The weight decay is fixed to 0.0001 on each dataset. The learning rate is initialised as 0.1. When training on the CIFAR and SVHN datasets, the learning rate is divided by 10 on 150 and 225 epochs and the training terminates at 300 epochs. When training on the ImageNet, the learning rate is divided by 10 on 30 and 60 epochs and the training stops at 90 epochs. When train-

ing on the SVHN dataset, a dropout layer is added [212] as in [18, 104, 207], with the dropout rate to 0.2. All experimental tests on the evaluation section are conducted using Pytorch [213].

4.4 Summary

This chapter applied the GA to optimise the structure of the CNNs. The GA assigns a binary bit to each layer of the input to remove the redundant features from the feature map. This chapter first proposes two encoding methods which enable the GA to evolve the structure of the model. Then this chapter applies a simple but effective genetic process containing selection, crossover and mutation. Conventional evaluation methods, in which each "individual" is fully trained, consumes extremely high computational resources and takes a very long time for training. Previous work that optimises the individual training by using weight inheritance between generations, is restricted by the residual structure. Thus, the proposed approach, a pre-trained weight inheritance method can not only reduce the individual training time, but it also releases the constraints of the residual structure. Experimental results indicate that the proposed model can reach competitive classification accuracy, while requires significantly fewer parameters.

Rather than generating structures with a large searching space, the proposed method is built based on predefined constraints, which has significantly reduced the computational cost and is capable of generating a well performing structure under limited training generations. When tested on the CIFAR-10 dataset, the proposed model saves more than 90% of the parameters in comparison to the state-of-the-art models. The comparable results from the ImageNet has validated that the proposed model is robust in both the variance and the scale of the classification tasks. At the current stage, the proposed approach is only validated on CNN. In the future, it will be evaluated using other popular deep learning networks, such as RNN and LSTM.

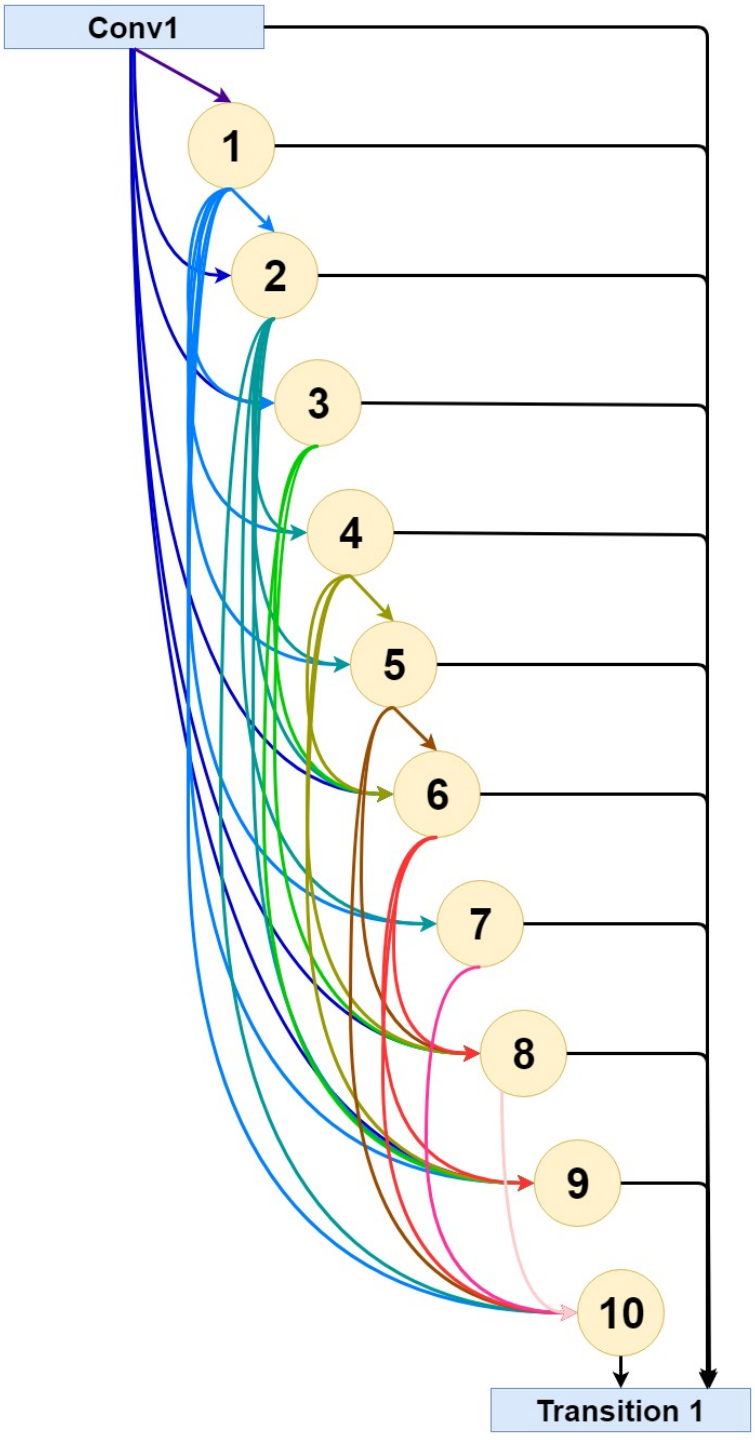


Figure 4.5: Visualisation of the first block. Each numbered circle denotes a convolutional layer. Connections between layers are expressed by colour arrows. "Conv1" denotes the first convolutional layer before the first block.

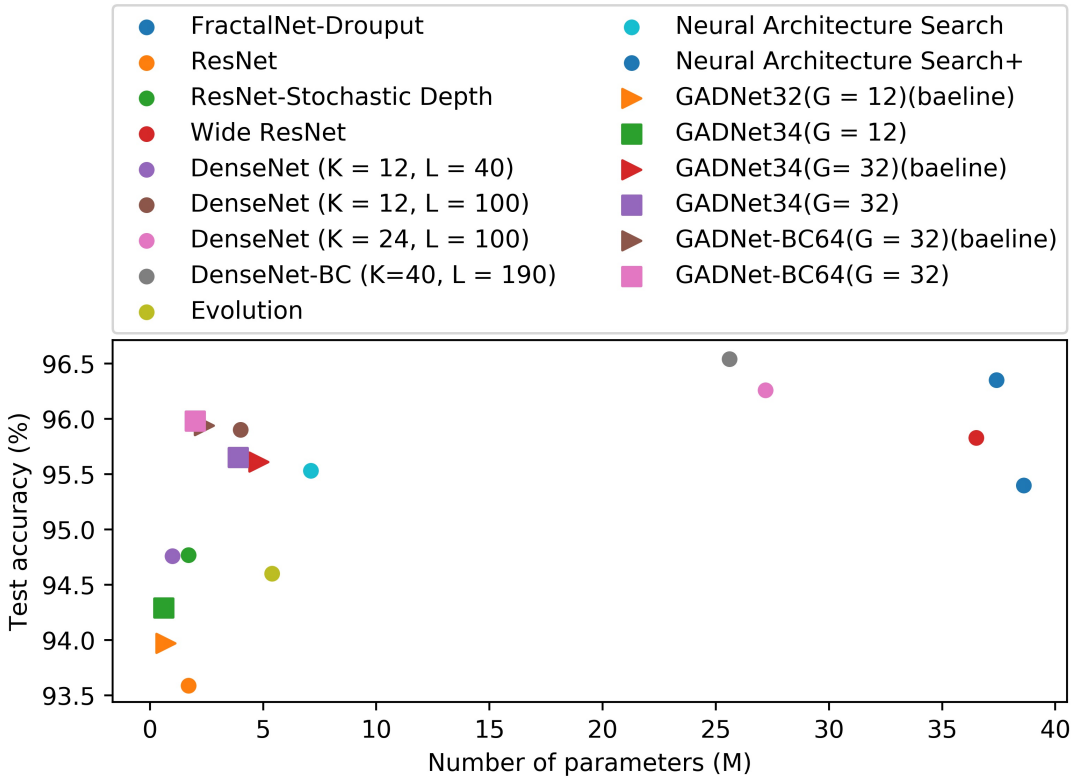


Figure 4.6: The number of parameters (M) and the corresponding test accuracy (%) of each model on the CIFAR-10 dataset. The proposed models are labelled using large squares, while the baseline models are marked as large triangular. Other models are denoted by dots.

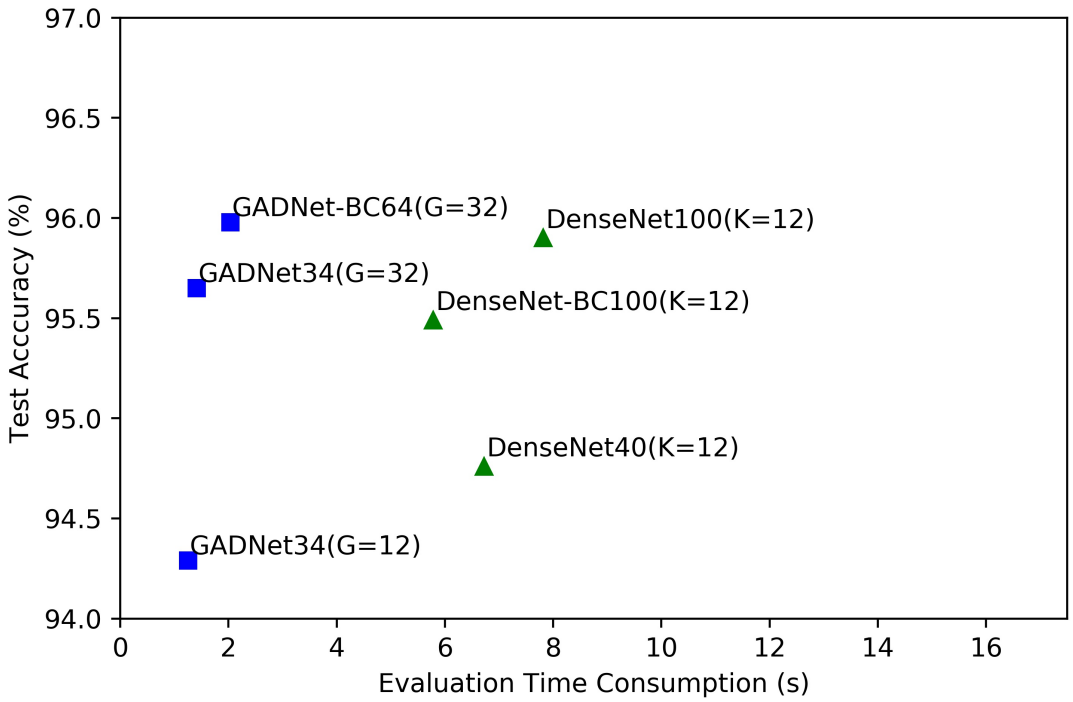
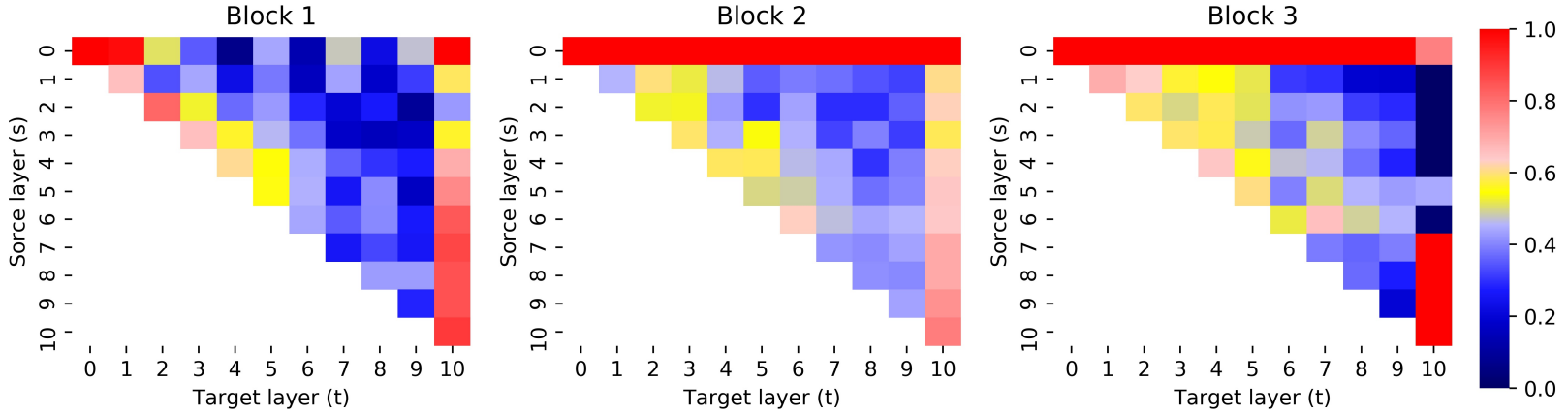
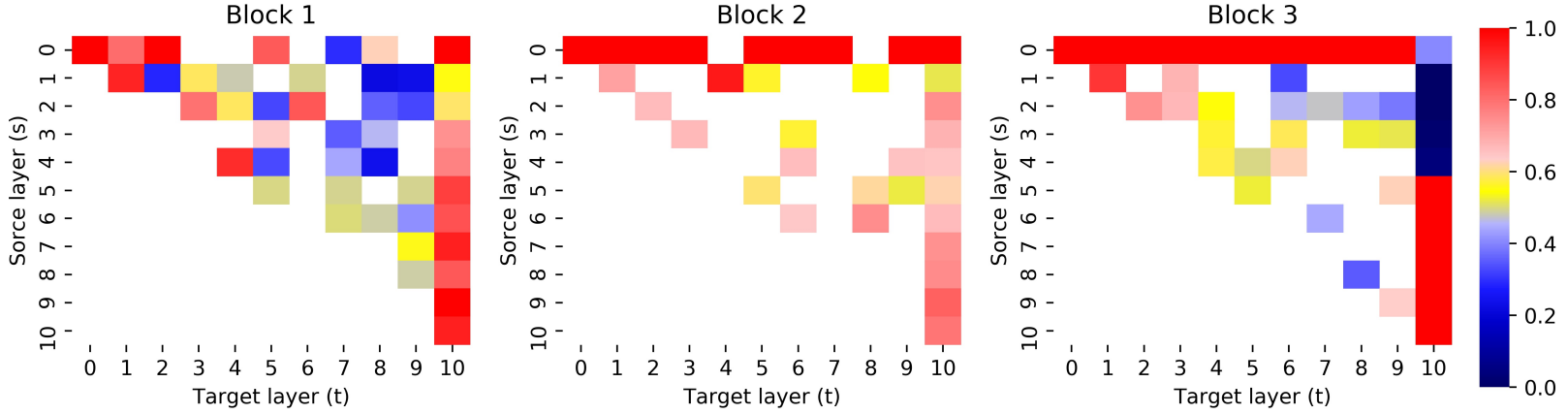


Figure 4.7: Comparison of the evaluation times (S) of the proposed method (marked by squares) and the DenseNet (labelled by triangles). Evaluation times are measured on the CIFAR-10 test dataset, where only forward propagation is conducted. All models are run on a single GTX1080ti during the time measurement.



(a) DenseNet34 (G = 12)



(b) GADNet34 (G = 12)

Figure 4.8: The average absolute filter weights of convolutional layers in (a) DenseNet34 (G = 12) and (b) GADNet34 (G = 12), trained on the CIFAR-10 dataset. The colour patch (s, t) denotes the average L1 normalised (by number of input depth) weights in the layer "t" which takes input from the layer "s" in the same block. The last column of each block denotes the transition layer for the first two blocks and the fully connected layer for the last block respectively.

Chapter 5

Triple Loss for Hard Face Detection

5.1 Introduction

Though previous Convolutional Neural Network (CNN) based face detectors have made a remarkable progress, detection of hard faces, i.e. faces with low resolution (<20 pixels) and high interference, is still a challenging task. Compared with easily detected faces, the resolution of hard faces is always low, which are interfered by, such as, blurry, occlusion, illumination and makeup. Those interferences cause the lack of visual consistency [41]. Existing methods tackle this challenge from both structure and loss function optimisations in the deep learning framework. Structure optimisation aims to improve the performance by enhancing the capability of feature extraction, which can be conducted in two ways. The first is to apply a deeper CNN feature extractor (backbone) [37,141,214], e.g. ResNet-101 [17], ResNet-152 [17], ResNeXt-101 [105] and DenseNet [18]. The second is to assign a subnet in the backward pathway [40–42, 141] to enhance the merged feature extraction. As the scale of the network grows larger [37,138,141], the accuracy on the hard face detection improves, but the computational cost increases at the same time. Instead of increasing the scale of the model, the loss optimisation strategy [39–41, 138, 139, 141] optimises the weights of each layer by assigning multiple tasks during the training, such as key point [139], attention [39],

segmentation [138], and head-body detection [41].

This chapter presents a novel mechanism to optimise the performance on hard face detection through optimisation of both the structure and the loss function. For structure optimisation, a new feature fusion module (FFM) embedded in the backward pathway is proposed to make full use of the high-level and low-lever features. To avoid increasing the computational cost significantly, both dilated convolution and small-size-kernel convolution (1-by-N and N-by-1 kernels) are utilised in the FFM. For loss optimisation, a "triple loss" training strategy is proposed, which covers three resources in the training process: i.e. forward path (the first level), backward path (the second level) and the extended path (the third level). The first two paths are the same as Feature Pyramid Network (FPN) and the feature maps of the extended path are simply extracted from the features of the backward path, by an additionally proposed FFM. However, during inference, only results predicted from the second level will be considered, i.e. all the irrelevant layers will be discarded. Through this training and inference strategy, the proposed network suppressed the increase of computational cost when compared with other FPN based methods [39–42,138,141]. By taking VGG-16 [16] as the backbone, the proposed model achieves comparable results with other models which utilise much deeper backbones. When evaluated on the WIDER FACE database, compared with other VGG-16 based face detector, the proposed method reaches the state-of-the-art on the hard subset. Although accuracy and computational cost seem to conflict to each other in face detection, by using the proposed FFM and the triple loss training, the proposed method reaches a good balance between these two metrics. Experimental results show that, without considering the non-maximum-suppression (NMS), the proposed method can detect faces by taking $29.7ms$ for a VGA-resolution image with 640 pixels in width and 480 pixels in height.

The rest of the chapter is organised as follows: Section 5.2 details the proposed FFM and triple loss training strategy. Section 5.3 presents the experiment results, including the ablation learning, comparison analysis and further discussions. Finally, some concluding remarks are given in Section 5.4.

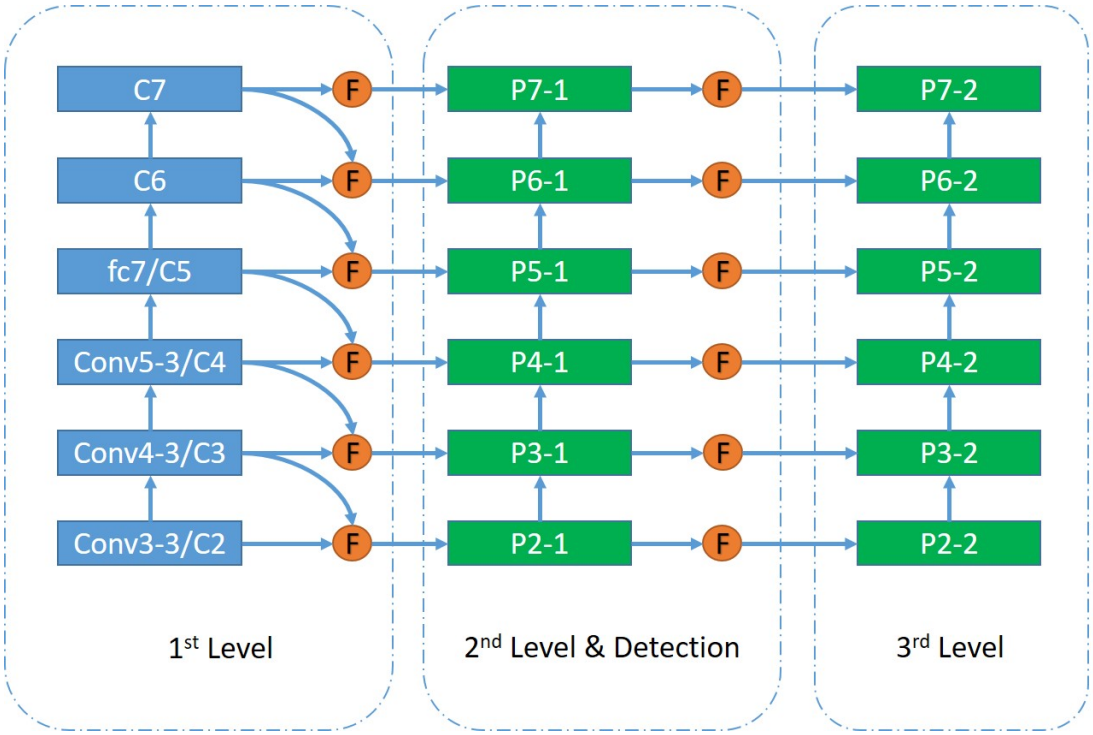


Figure 5.1: The proposed network structure trained with the "triple loss" training strategy. Feature maps of the first layer are generated through "Conv3-3", "Conv4-3", "Conv5-3", "Conv-fc7", "Conv6-2", and "Conv7-2", where the first four are from VGG-16, and the last two are from newly added layers. Two newly added layers are for detecting large scale faces, which are the same as used in Single Shot MultiBox Detector (SSD) and Single Shot Scale-invariant Face Detector (SFD). The reduced sizes of feature maps related to the original image are 4, 8, 16, 32, 64, and 128. The feature maps of the second level are denoted as {P2-1, P3-1, P4-1, P5-1, P6-1, P7-1}. For the third level, a single FFM is assigned for each layer, where the input of each FFM is the feature map derived from the second level, to obtain the feature maps of the third level. Feature maps of the third level are labelled as {P2-2, P3-2, P4-2, P5-2, P6-2, P7-2}. The structure of FFM on the third level is the same as the second level without weight sharing. Details of FFM (denoted by "F") are shown in Figure 5.2

5.2 The Proposed Approach

This section presents the proposed triple loss training strategy, as well as the feature fusion module for face detection. First, the whole network structure will be illustrated, followed by the structure of the proposed feature fusion module. After that, the triple loss training strategy will be detailed.

5.2.1 Overall Network Structure

Figure 5.1 illustrates the structure of the proposed network, which is composed of three levels according to the predicted outputs of triple loss. In the first level, feature maps are generated through a pre-trained backbone. As the triple loss is designed for generalised face detection, in this thesis, VGG-16 [16] is mainly considered as used in [35,40,41]. As a result, following the structure in SFD [35], feature maps of the first layer are generated through "Conv3-3", "Conv4-3", "Conv5-3", "Conv-fc7", "Conv6-2", and "Conv7-2", where the first four are from VGG-16, and the last two are from newly added layers. Two newly added layers are for detecting large scale faces, which are the same as used in SSD and SFD. The reduced sizes of feature maps related to the original image are 4, 8, 16, 32, 64, and 128.

Feature maps from the deeper layer have more semantic information extracted [33]. In order to obtain more semantic information for low-level feature maps, feature maps are normalised using 1×1 convolutional kernels in the top-down routine as suggested in [141]. From "Conv3-3" to "Conv6-2", normalised feature map is up-sampled from up-layer and conduct elementwise product with the current one. After that, a feature fusion module is deployed to enhance the capability of feature extraction and increase the receptive field. Hence, feature maps extracted by FFM are used to form the second level. Results in [40–42, 141] show that such a module is helpful for improving the performance of the detector, and the experiments as detailed in Section 5.3 have also verified this point. The third level is simply extracted from the second level by a proposed additional FFM without any top-down connection, whilst there are two top-down connections in FANet. However, experimental result shows that the proposed method outperforms FANet by 0.3% in detecting small faces without assigning the top-down connection at the third level.

Letting the feature map of the layer j ($j \in [1, 6]$) from the level i ($i \in [1, 3]$) be $\Phi_{(i,j)}$, the feature map of the next level $\Phi_{(i+1,j)}$, in FANet, can be mathematically defined as:

$$\begin{aligned} \Phi_{fusion} &= f_{eleprod}\left(f_{1\times1}\left(\Phi_{(i,j)}\right), f_{1\times1}\left(\Phi_{(i,j+1)}\right)\right) \\ \Phi_{(i+1,j)} &= f_{1\times1}\left(f_{\text{concat}}\ \left(f_{\text{inception}}\left(\Phi_{fusion}\right)\right)\right) \end{aligned} \tag{5.1}$$

where $f_{1\times1}$, $f_{eleprod}$ and f_{concat} indicate the operations of 1×1 convolution, elementwise production, and feature concatenation respectively; Φ_{fusion} is the fused feature map after elementwise production; and $f_{\text{inception}}$ indicates a inception subnet structure. On the other hand, in the proposed method, the feature map in level 2 and level 3 can be expressed by:

$$\begin{aligned}
\Phi_{fusion} &= f_{eleprod}\left(f_{1\times1}\left(\Phi_{(1,j)}\right), f_{1\times1}\left(\Phi_{(1,j+1)}\right)\right) \\
\Phi_{(2,j)} &= f_{1\times1}\left(f_{\text{concat}}\ \left(f_{\text{inception}}\left(\Phi_{fusion}\right)\right)\right) \\
\Phi_{(3,j)} &= f_{1\times1}\left(f_{\text{concat}}\ \left(f_{\text{inception}}\left(\Phi_{(2,j)}\right)\right)\right)
\end{aligned} \tag{5.2}$$

Similar to [32,35,40–42] feature maps of the first level are extracted from the forward path. For the second level, as given in Eq. 5.2, the initial feature map and the feature map derived from its upper layer are convolved by a $1\times1\times256$ kernel, respectively. The two normalised feature maps are fussed via elementwise product, which is taken as the inputs to the FFM of the second level. Afterwards, it will pass a three-branch inception subnet, which is the dominant part for FFM and will be detailed in the next section. The outputs from different branches are concatenated and the number of channels is normalised to 256. The feature maps of the second level are denoted as {P2-1, P3-1, P4-1, P5-1, P6-1, P7-1}. For the third level, a single FFM is assigned for each layer, where the input of each FFM is the feature map derived from the second level, to obtain the feature maps of the third level. Feature maps of the third level are labelled as {P2-2, P3-2, P4-2, P5-2, P6-2, P7-2}, see in Figure 5.1. The structure of FFM on the third level is the same as the second level without weight sharing.

5.2.2 Feature Fusion Module

In this section, a feature fusion module is proposed to enhance the capability of feature extraction, where the fused feature map is extracted through the backward pathways, as well as to increase the receptive field. At present, the mostly used backbone for face detection are VGG-16 [34, 35, 40, 41] and Residual Network (ResNet) [39, 42, 138, 141], where the kernel shapes are 3×3 and 1×1. As a result, the effective receptive field of each layer is in a square shape. However, experimental results in RFB [135] indicate

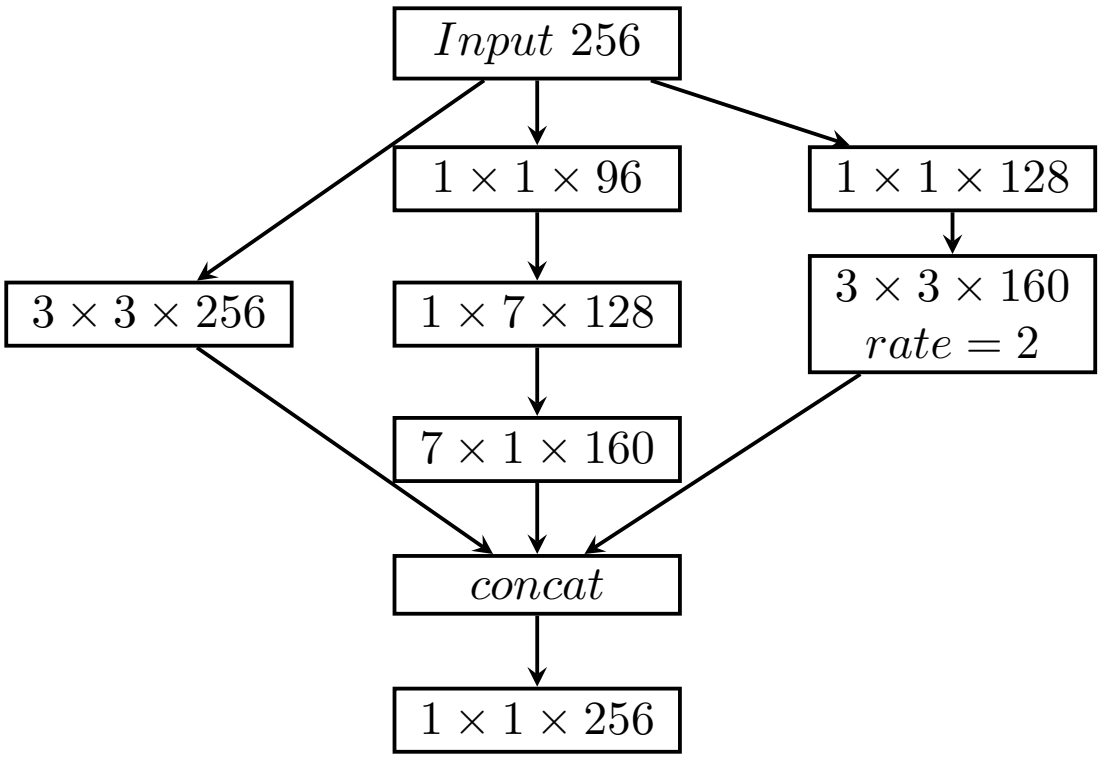


Figure 5.2: The proposed feature fusion module (F-block in Figure 5.1).

that for some non-square objects (i.e. aspect ratio is not 1), the shapes of effective receptive field may not be the typical shape of squares. As illustrated in SRN [42], this issue seems not crucial for frontal face detection, because the aspect ratio is about 1. However, this is important for multi-pose face detection as the aspect ratios can vary between 0.5 and 2. To tackle this challenging problem, a feature fusion module with multiple shapes of kernels is presented. The structure of the proposed FFM is shown in Figure 5.2. Following the design in [40–42,141], the inception structure [65] is utilised in the proposed feature fusion module, which consists of three branches. The first branch has a single 3×3 convolutional layer to smooth feature as in FPN [33]. Inspired by RFB [135] and Dual Shot Face Detector (DSFD) [141], the second branch consists of two dilation convolutional kernels, in order to further increase the receptive field sparsely. As illustrated in SRN [42] a densely feature extractor is also important for refining the effectiveness of the receptive field. However, using $N \times N$ kernels will increase the computational cost significantly. Hence, to balance the computational cost

and detection accuracy, in the third branch, a $1 \times N$ and $N \times 1$ structure is employed to extract dense features. At the end of the module, feature maps from the three sub-networks are concatenated together and then smoothed by a 1-by-1 convolutional layer. Experimental results show that the proposed FFM, using the combination of dilation convolution and $1 \times N$ (with $N \times 1$), performs better than the existing ones [40–42,141].

5.2.3 Triple Loss Training Strategy

This section will introduce the proposed triple loss training strategy in details. As described in Section 5.2.1, feature maps are splitted into three levels. During training, a classification layer and a regression layer are assigned on each feature map in all three layers. To improve the inference efficiency, the model only uses two 3×3 convolutional layers for classification and regression (detection head) separately [39, 42, 141], without a retina head [114]. The triple loss function (TL) is defined as follows:

$$\begin{aligned} &TL(\Phi_{(1,1)}, \Phi_{(1,2)}, ..., \Phi_{(2,j)}, ...\Phi_{(i,j)}, \mathbf{A}) \\ &= \sum_{i=1}^{3} \omega_i L(\Phi_{(i,1)}, \Phi_{(i,2)}, ..., \Phi_{(i,j)}, A_i) \end{aligned} \tag{5.3}$$

where A_i and ω_i denote respectively the anchor setting and the adjusting parameter with respect to level i ($i \in [1, 3]$) as there exist three levels in triple loss. Experimental result shows that the magnitudes of the loss from all three levels are the same, i.e. the contribution of each level to the loss is the same, which is similar to those reported in DSFD [141] and FANet [40]. The proposed method uses the cross-entropy loss (as in [39, 41, 141]) and the smooth L1 loss [3] to determine the classification loss and the regression loss, respectively. To be more specific, the total loss function of each level can be expressed as below:

$$\begin{aligned} TL(p, p^*, t, t^*) = &\sum_{i=1}^{3} \frac{1}{N_{conf}} L^i_{conf}(p_i, p^*_i) \\ &+ \frac{1}{N_{loc}} [p^*_i = 1] L^i_{loc}(t_i, t^*_i) \end{aligned} \tag{5.4}$$

where, for level i, L^i_{conf} and L^i_{loc} are the confidence prediction loss and the localisation loss terms, respectively; p_i, p^*_i, t_i and t^*_i refer respectively to the predicted probability, ground truth probability, predicted regression target and ground truth box regression target. The Iverson bracket indicates a function $[p^*_i = 1]$ outputs 1 when the condition holds true, i.e. only the regression loss of positive instances will be minimised during the training.

The anchor setting is another key factor that affects the performance of face detector. As suggested in DSFD [141], assigning small anchor size on the forward pathway improves the prediction performance, even it is not used during inference. Hence, the anchor sizes of the first level are halved compared with the following levels, as shown in Table 5.4. The aspect ratio is set as 1.25 for all three levels as suggested in SRN [42].

During inference, the second layer is selected to conduct face detection. The detection heads of the first and the third levels, as well as the additional feature fusion modules in the third level, are discarded. Hence, the proposed face detector will not add additional parameters and computational cost compared to other FPN based methods [39–41, 141].

5.3 Experiment Results and Discussions

First, the proposed method is analysed in detail to clarify the effectiveness of the contributions. The final model is evaluated on two commonly used face detection benchmark datasets, FDDB [30], and WIDER FACE [31].

5.3.1 Training Datasets and Hyperparameters

The ablation learning is conducted on the WIDER FACE dataset [31], which consists of 393,703 annotated face bounding boxes in 32,203 images. Images in WIDER FACE are splitted into three sub-datasets: training, validation and test dataset. Performance is evaluated in terms of average precision (AP) with the Interception-of-Union (IoU) set to 0.5, and the AP is measured by sampling the precision values where the corresponding recall values are $0, 0.1, 0.2, \ldots, 1$. Instead of describing the result by a single mean

average precision (mAP) over the whole validation dataset or test dataset, there are three subsets according to the detection difficulty levels: Easy, Medium, and Hard, based on the detection rate of EdgeBox [21]. The training dataset, which has 12,880 images, is applied as the only training dataset in this thesis. Results of ablation learning are compared on the validation dataset with 3,226 images. In the end, the proposed model is evaluated on the test datasets (with 16,097 images) and the results are collected from the official test server.

The training procedure adopts the same data augmentation method as [41, 141]: At first, it conducts random flipping, colour distortion, etc., which is the same as in SSD [32]. For image resizing, with a probability of 0.6, it conducts the original image resize method as introduced in SSD. Otherwise, it resizes the image using data-anchor-sampling as in Pyramidbox [41]. To balance the ratio of positive and negative training instances, the online hard example mining (OHEM) is used in a similar way as [32,35,41,141] and assign the ratio of positive: negative is set as 1:3. In the end, a 640 $\times$ 640 patch will be resized from each cropped image patch. Image expansion [35, 42] is also included in augmentation but the results seemed quite poor. It is deduced that expansion may not fit for low batch size training. As a result, expansion is not include in the augmentations.

The backbone network is initialised by the pretrained VGG on ImageNet. All newly added convolution layers' parameters are initialised by the 'xavier' method [215]. SGD with momentum is adopted for weight optimisation and weight decay is set as 0.9 and 0.0005 to train the models. The batch size is set to 12. The learning rate is initialised to 1e-3 and is decayed by 10 when at 80K and 100K steps, respectively. During inference, the settings of hyper parameters are the same as in [32, 35, 41, 141]. The second level predicts the top 5K high confident detections, followed by non-maximum suppression, with the Jaccard overlap of 0.3, to produce the top 750 high confident bounding boxes per image.

5.3.2 Model Analysis

In this section, a series of ablation experiments will be conducted on the WIDER FACE dataset to analyse how each contribution module improves the performance in detail. For a fair comparison, the parameter settings are kept the same, including anchor setting, training hyper parameters, data augmentations, etc., for all the experiments.

As the structures of recent proposed face detectors [39–42,141] contain both bottom-up and top-down pathway as FPN, the ablation study uses FPN as a baseline to make a fair comparison. The anchor setting of the baseline is the same as SFD and PyramidBox, which is [16, 32, 64, 128, 256, 512], and the aspect ratio is 1.25 as in SRN [42]. All models in this section are trained on the training set and evaluated on the validation set.

5.3.2.1 Feature Fusion Module

First, this section will show how the proposed feature fusion module improves the performance of the baseline. In Table 5.1, the performance of different feature fusion modules are compared on the WIDER FACE validation dataset. As observed, with the same backbone (VGG-16) and the network structure (FPN), the proposed feature fusion module surpasses the baseline by 0.7%, 1.1% and 4% on the easy, medium and hard subsets, respectively. When compared with other feature fusion modules, the proposed module reaches the best on the medium and the hard subsets, which leads the state-of-the-art method by 0.1% on the medium subset and 0.4% on the hard subset, respectively. It is deduced that such improvement of increased accuracy is contributed by the combination of dilated convolution and the ordinary convolution. Under a similar computational cost, dilated convolution increases the receptive field of the feature map significantly [135]. However, a large receptive field may also harm the performance of small object detection [41].

To balance the performance on various scales, a concatenation of feature maps from both convolutional layers is needed. It is noticed that when compared with CPM [41], the proposed feature fusion module lags by 0.1% on easy subset. It is deduced that this is caused by the larger output channel number of CPM: the number of the output

Table 5.1: Effectiveness of various feature fusion approaches in terms of AP.

Component	Easy	Medium	Hard
Baseline [40]	94.3	92.9	83.8
+CEM [40]	94.8	93.6	84.4
+CPM [41]	**95.1**	93.9	87.4
+RFE [42]	94.9	93.8	87.2
+FEM [141]	94.9	93.9	87.5
+FFM (Proposed)	**95.0**	**94.0**	**87.8**
+FFM-512 (Proposed)	**95.2**	**94.0**	**87.9**

channels from CPM is 512, which is the double of the proposed FFM. This large scale of the subnet will consume a huge amount of computational cost, which will be shown in Section 5.3.2.5 later. On the contrary, the proposed FFM is more light-weighted, reaching the balance between accuracy and the inference efficiency. To validate the performance of the proposed FFM on a larger number of output channels, the structure is kept unchanged but expand the output channels to 512 and add batch normalisation [67] before each convolutional layer as used in CPM, labelled as "FFM-512" in the table. As seen, when the number of output channels is doubled, the proposed FFM outperforms the CPM on all three subsets.

5.3.2.2 Triple Loss Training

This section evaluates the performance of the triple loss training strategy in detail. An ablation learning is conducted to show how each level affects the model. SSD is used as the baseline, which calculates the loss only using the first level. The loss from the second and the third level are calculated separately, where the proposed feature fusion module will be applied. Finally, level by level, it combines the losses from different levels into training. During evaluation, for single level loss training, the result is obtained from that training level, while for multi-level training, the results are collected from the deepest level. Experimental results are given in Table 5.2.

From the first three rows in Table 5.2, it is clear to see that the accuracy increases as the scale of the network increases, when using the single level loss training. However,

Table 5.2: Results of the triple loss on the WIDER FACE validation subset.

Component	Easy	Medium	Hard
1st level	94.0	93.0	83.5
2nd level	95.0	94.0	87.8
3rd level	95.2	94.3	88.0
1+2 levels	95.6	94.7	89.1
1+2+3 levels	**95.8**	**94.8**	**89.7**

the increasement between levels decreases at the same time. To balance the computational cost of training and the evaluation accuracy, the fourth level is not added in the experiment. It is deduce that the contribution from the fourth layer might be minor for face detection.

When using multi-level training, which is shown in the last two rows, it is worthy to find out that the performance of the model is increased significantly. When compared with the models trained on a single level (on the third level), the performance of the model measured using detection accuracy, trained via triple loss, is increased by 0.6%, 0.5%, 1.7% on the three subsets, respectively. Experimental results indicate that when the scale of the model is fixed, the multi-level training strategy helps to increase the performance of the model, especially on the medium and the hard subsets.

5.3.2.3 Prediction Level

Predicting using the third-level feature map increases the performance. However, it also increases the computational cost. As the anchor in the second and the third levels are identical, it is possible to predict via the second level. When evaluating through the second level, feature maps from the third level will be omitted hence the total computational cost is reduced. The result comparison of different prediction levels is presented in Table 5.3. In this test, after trained using triple loss, the model predicts result through the second and the third level separately. Compared the result predicted from the third level, the AP predicted through the second level is the same on the easy subset, and 0.1% better on the medium and hard subset. This indicates that the third level is essential during training but seems unnecessary during evaluation. In

Table 5.3: Comparison of results on different prediction levels.

Prediction Level	Easy	Medium	Hard
3rd	95.8	94.8	89.7
2nd	**95.8**	**94.9**	**89.8**
2nd + 3rd	95.7	94.7	89.5

summary, the proposed triple loss training strategy improves the AP without increasing the computational cost during inference.

Furthermore, to validate the performance of multi-level prediction, the prediction results predicted from both the second and the third levels are also collected, which is shown in the last row of Table 5.3. Apparently, prediction from two levels does not bring an increase but a decrease on the prediction result. On the other hand, as prediction heads from both levels are applied, this will increase the computational cost. As a result, multi-pathways inference is not utilised in the model.

5.3.2.4 Effect of Anchor Design

As anchor design is a key factor of the box size regression [40, 42, 141], this section discusses how the anchor size affects the performance. In DSFD, experimental results show that a smaller anchor size on the forward pathway (first level), which is halved compared to the backward pathway, can further improve the performance. Motivated by this observation, the anchor size is fix by $[8, 16, 32, 64, 128, 256]$, as suggested in DSFD, on the first level and vary the anchor sizes on the second level and the third level. Based on the findings in Section 5.3.2.3, the second level is used as the prediction level during inference.

Experimental results are shown in Table 5.4. When the anchor size increases progressively, the third level impedes the final prediction during inference. Then the anchor size is swapped between the second level and the third level. As a result, the model further improves the AP by 0.4% and 0.3% on the easy and medium subsets respectively (see row 2 of Table 5.4), when compared with the identity setting (row 1). It is not surprising to see the poor performance on the hard subset because the large anchor

Table 5.4: Comparison of results on different anchor assignments.

Predefined anchor sizes: **A1**: [16, 32, 64, 128, 256, 512] **A2**: [32, 64, 128, 256, 512, 1024] **A3**: [(16,32), (32,64), ..., (512,1024)]			
Anchor applied	Easy	Medium	Hard
A1 (2^{nd}, 3^{rd})	95.8	94.9	**89.8**
A1 (2^{nd}), A2 (3^{rd})	**96.2**	**95.2**	82.5
A3 (2^{nd}), A2 (3^{rd})	96.1	95.0	88.6

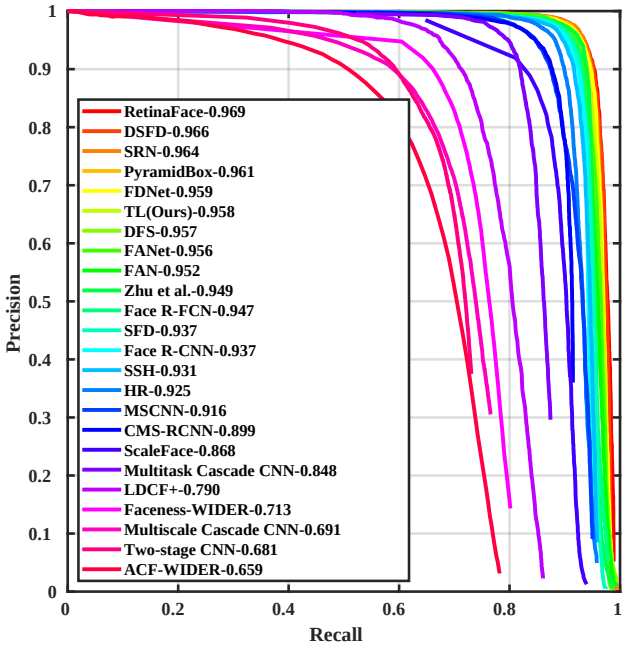
size is unsuitable for detecting small faces, which mainly belong to the hard subset. Consequently, the number of anchors in the second level is doubled, as shown in row 3, to gain benefits of both designs. In summary, the identity setting is important for hard face detection, while progressive setting brings increase on the other two subsets.

5.3.2.5 Comparison with Other Face Detectors

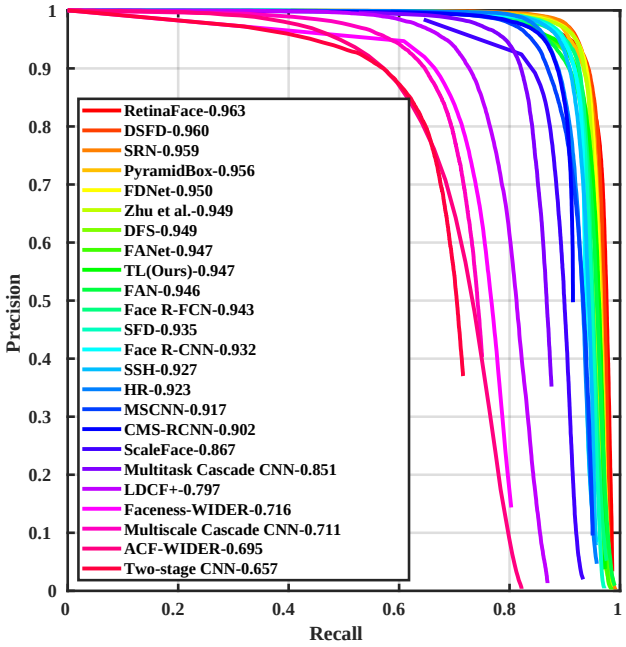
This section presents the comparison between the proposed method with other algorithms. The APs of three subsets on the WIDER Face are respectively given in Figure 5.3, Figure 5.4, Figure 5.5, of which the model uses the identity anchor setting on the second and the third levels. As the accuracy relates to the scale to backbone, the backbones of the state-of-the-art methods are summarised in Table 5.5. As observed, the proposed model reaches the best performance on the hard subset, when compared with other VGG-16 based models; it also attains the best AP on the easy and medium subsets, when using the progressively anchor setting (labelled by "TL-LA"). Even when compared with the state-of-the-art methods on the hard subset, as shown in Figure 5.6, the proposed method only sacrifices the accuracy by about 0.6% but with much more computational saving.

5.3.2.6 Effects of Backbone

To evaluate the robustness of the proposed detector, the performance on the Resnet-50 is also validated, which is shown in the last two rows of Table 5.5. As most of the

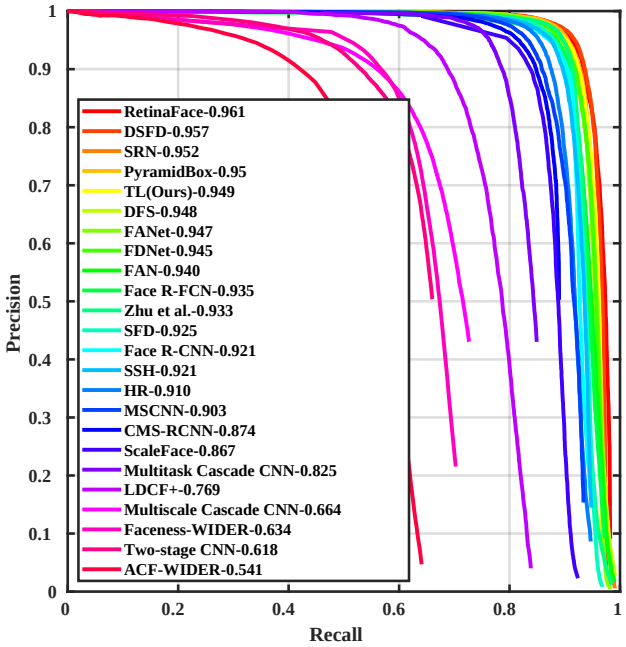


(a) Val: Easy

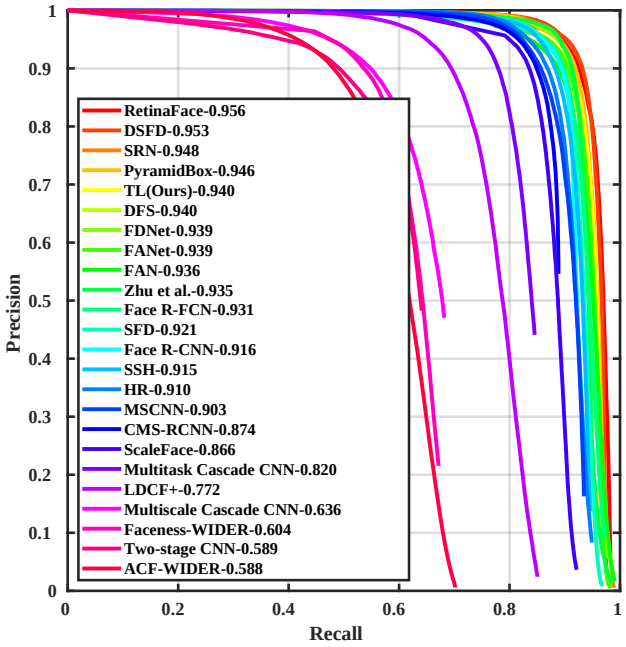


(b) Test: Easy

Figure 5.3: Precision-recall curves on WIDER FACE validation and test sets (Easy).

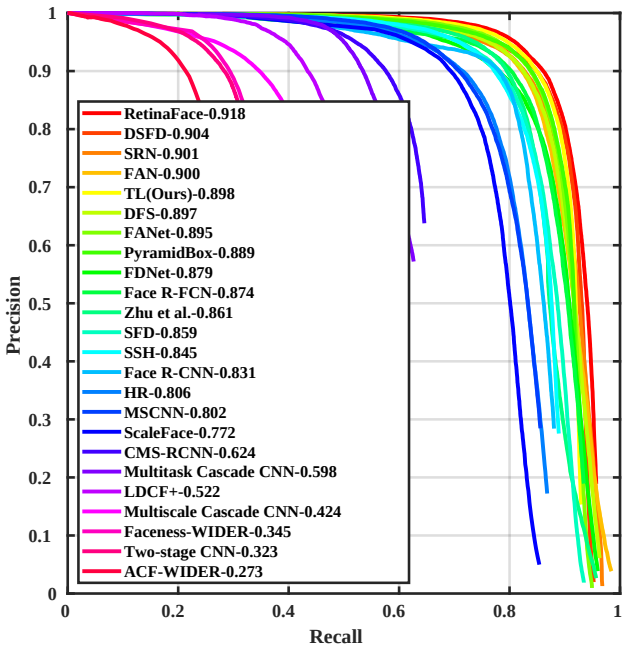


(a) Val: Medium

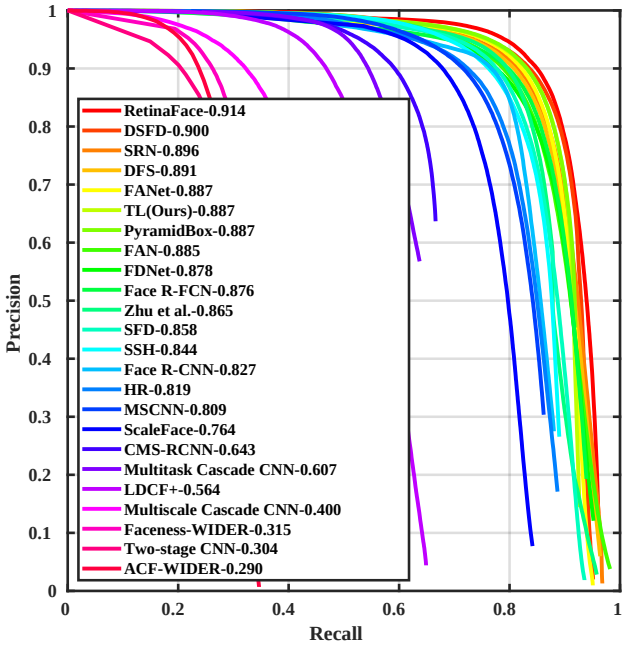


(b) Test: Medium

Figure 5.4: Precision-recall curves on WIDER FACE validation and test sets (Medium).



(a) Val: Hard



(b) Test: Hard

Figure 5.5: Precision-recall curves on WIDER FACE validation and test sets (Hard).

Table 5.5: Result comparison on the WIDER FACE validation set.

Methods	Backbone	Easy	Medium	Hard
ScaleFace [216]	ResNet-50	86.8	86.7	77.2
HR [37]	ResNet-101	92.5	91.0	80.6
Face R-FCN [217]	ResNet-101	94.7	93.5	87.4
Zhu [214]	ResNet-101	94.9	93.3	86.1
RetinaNet [42]	ResNet-50	95.1	93.9	88.0
SRN [42]	ResNet-50	96.4	95.2	90.1
DSFD [141]	ResNet-152	96.6	95.7	90.4
RetinaFace [218]	ResNet-50	**96.5**	**95.6**	**90.4**
RetinaFace [218]	ResNet-152	**96.9**	**96.1**	**91.8**
CMS-RCNN [219]	VGG16	89.9	87.4	62.4
MSCNN [220]	VGG16	91.6	90.3	80.2
Face R-CNN [36]	VGG19	93.7	92.1	83.1
SSH [34]	VGG16	93.1	92.1	84.5
S3FD [35]	VGG16	93.7	92.5	85.9
PyramidBox [41]	VGG16	**96.1**	**95.0**	88.9
FANet [40]	VGG16	95.6	94.7	**89.5**
TL (Proposed)	VGG16	95.8	94.9	**89.8**
TL-LA (Proposed)	VGG16	**96.2**	**95.2**	82.5
TL-res50 (Proposed)	ResNet-50	95.7	94.7	88.6
TL-res50-RH	ResNet-50	96.0	94.9	88.4

ResNet-based face detectors [39, 42, 138, 141] apply retinanet prediction head as in Lin et al. [114], for a fair comparison, both Single Shot MultiBox Detector (SSD) prediction head and retinanet prediction head are deployed in the model. During the training, the batch size is increased to 16 as DSFD did. Limited by the computational resource, the training procedure can only use a batch size of 12 when training on the retinanet head. Compared with the retinanet, the proposed method increases the performance by more than 0.6% on all three subsets. By using the retinanet head, the performance has been further improved by 0.3% and 0.2% on the easy and medium subset, respectively. As for the decrease on the hard subset, it is might caused by the decrease of batch size because both batch size and training image size are crucial for the final performance [196]. This

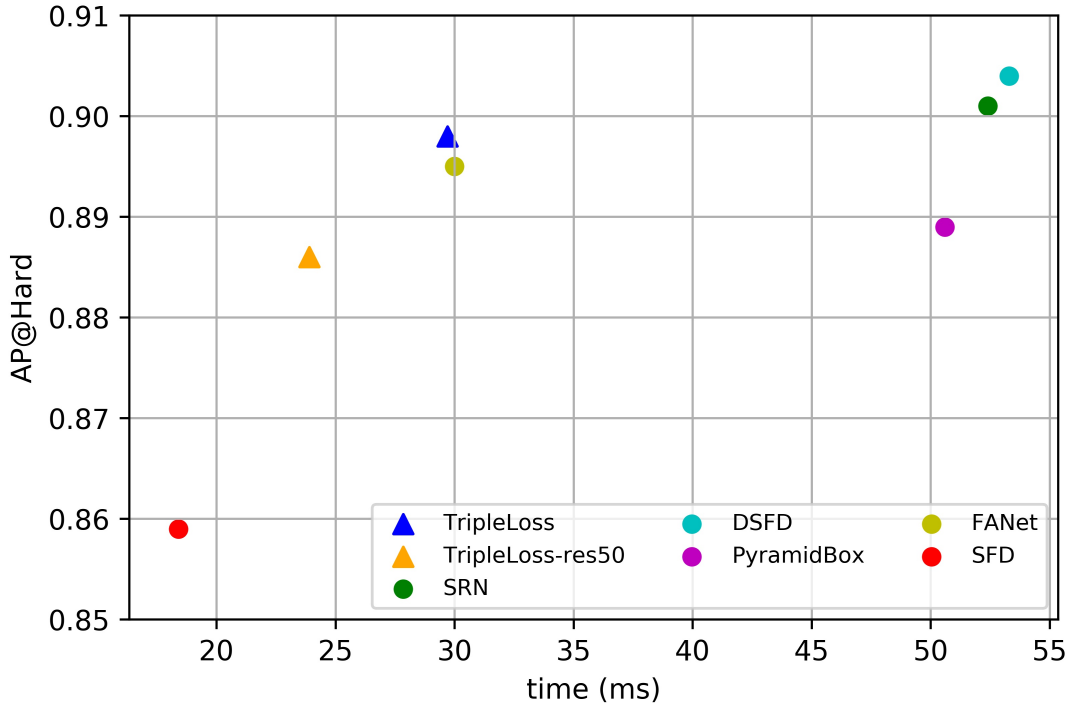


Figure 5.6: Time consumption among the state-of-the-art methods.

will be solved such issues when more computational resource is available in the future.

When compared with VGG-16, ResNet-50 outperforms on the easy subset without using the large anchor setting. However, the hard set AP is lower by 1.2%. It is deduced that this is caused by the nature of ResNet: the scales of the hard set are always small, which are detected by the low-level feature maps. As suggested in ISRN [196], ResNet reduces the feature map size earlier than VGG, consequently losing the information of small objects. On the other hand, however, it will be more efficient because of the early reduction of size. In summary, ResNet based methods are more suitable for speed-prioritised applications, while VGG backbone can be applied for detection of small faces.

When compared with other ResNet-50 based detectors, the proposed method outperforms most of them on easy and medium subsets, except for Selective Refinement Network (SRN) and RetinaFace [218]. In SRN, the prediction results from the first and the second levels are cascaded to reduce the number of false positives and refine

the positions of boxes, which improves the AP but also sacrifices the computational efficiency. However, with a slight decrease of the AP, the proposed method can achieve a good balance between the AP and the computation cost, as discussed in the next section. RetinaFace [218] achieves the state-of-the-art performance on the hard subset with the ResNet-152 used as backbone. For a fair comparison, RetinaFace with ResNet-50 as backbone is benchmarked with the proposed approach. As seen in Table 5.5, the proposed method lags by 1.8% on hard subset than RetinaFace when using ResNet-50 as backbone. The small difference is caused mainly by extra information such as facial landmarks and Three-Dimensional (3D) positions used in RetinaFace, in addition to Two-Dimensional (2D) face bounding box, which is the only information required in the proposed triple loss training model. On one hand additional features have led to significantly increased dimension of the prediction layer (from 6 to 160) and the associated computational cost. On the other hand, this inspires the future work to combine 3D information to further improve the proposed model. Furthermore, the face landmark prediction in RetinaFace relies on supervised training, which requires additional work for manual labelling the samples. In contrast, such extra labelling is avoided in the proposed approach. The training using ResNet-152 as backbone is also conducted, limited by computational resources, where both the batch size and the learning rate are reduced to 8 and 5e-4, respectively, whilst doubling the training epochs. As shown in Table 5.5, even the model is affected by the low batch size in training, the proposed method slightly lags RetinaFace (ResNet-152) by 0.7% and 0.6%, on easy and medium subset without using any additional labeled samples. This has further validated the efficacy of the proposed approach.

5.3.2.7 Inference Speed

As described in the last section, the proposed triple training strategy and FFM can balance between the detection accuracy and the computational cost. Figure 5.6 illustrates the inference speed, accompany with the accuracy, among the state-of-the-art methods. For a fair comparison, all the methods are deployed on Pytorch [176] without conducting the non-maximum suppression. All the tests are conducted on a single GTX1080Ti.

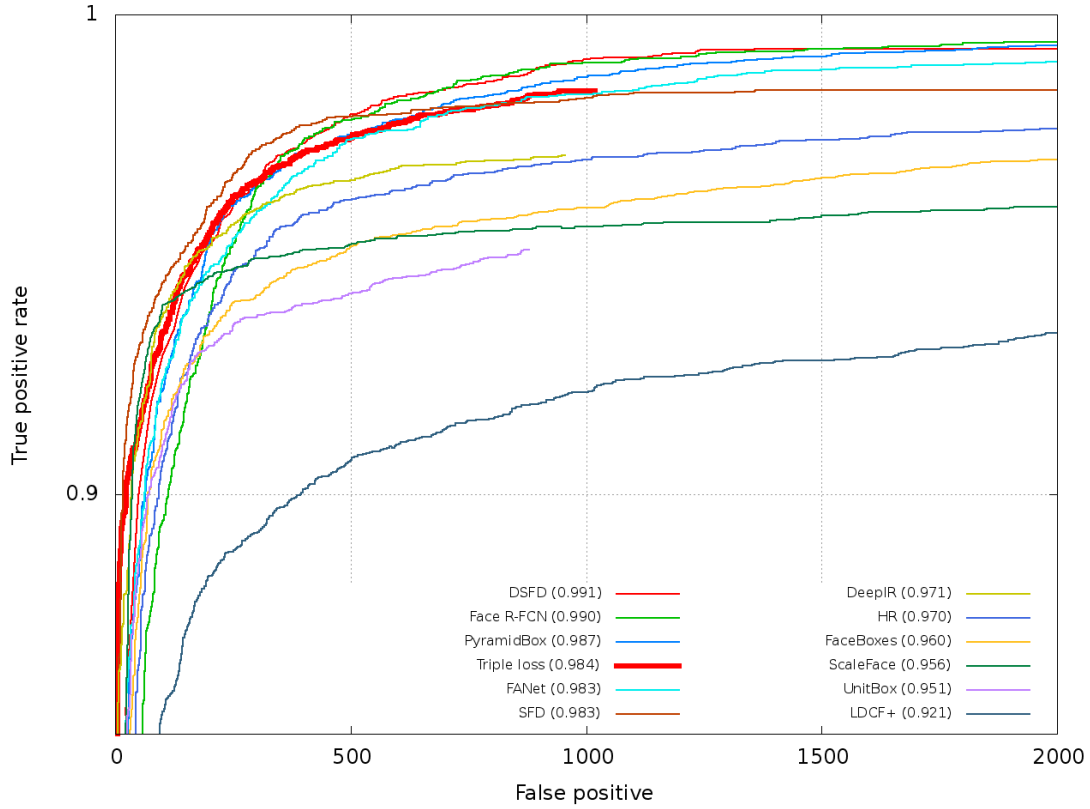


Figure 5.7: FDDB Discrete ROC Curves.

As seen, a deep backbone [141] or a heavy subnet for feature extraction [40–42] improves the detection accuracy by sacrificing the speed. However, a light-weight network structure [35] seems insufficient. As a result, the proposed FFM can improve the accuracy by slightly increasing the computational cost during inference. Different from the existing training methods [40, 42], the proposed triple loss training strategy only increases the computational cost during training without affecting the inference efficiency.

5.3.2.8 Evaluation on FDDB

To validate the performance on multiple datasets, the model is also evaluated on the FDDB dataset [30], where the proposed model is trained on the WIDER FACE dataset. In FDDB, there are 5,171 faces in 2,845 images taken from the faces in the wild dataset. Different from the WIDER FACE, faces in FDDB are labelled by ellipses. To show the robustness of the proposed method, the ellipses regressor is not trained offline.

Instead, the ellipses regressor in SFD [35] is adopted to transform the final prediction results from rectangle to ellipse. There exist unlabeled faces in the original dataset. For a fair comparison, additional annotations are added as in [35, 40, 41, 141] and the results are reported on discontinuous ROC curves [30], as shown in Figure 5.7. As seen, the proposed method achieves 98.4% when the number of false positives equals to 1,000. When compared with other state-of-the-art methods [35,40,41,141], the proposed method lags by no more than 0.8% by applying a relatively light-weight backbone. By applying a deeper backbone, FPN-based face detectors [35, 40, 42, 141, 196, 218] all show a certain degree of improved performance. To verify the performance of the proposed approach under a deeper backbone, additional experiments are conducted on the FDDB dataset using the ResNet-152 as backbone, which is trained on the WIDER FACE dataset. Due to limited available computational resources, the training procedure has selected a low batch size of 8 for comparison (learning rate = 5e-4, training steps = 240k), where the accuracy achieved from ResNet-152 and VGG-16 became 98.0% and 97.6%, respectively. This on one hand has shown that a low batch size indeed leads to degraded accuracy, as a larger batch size with VGG-16 can produce an accuracy of 98.4%. On the other hand, it validates that a deeper backbone can further improve the classification accuracy. As the proposed method is also FPN-based, it is deduced, by using the same batch size training, its performance can also be further improved when using a deeper backbone.

5.4 Summary

In this chapter, a novel training strategy is proposed, as well as an accuracy-computational cost balanced feature fusion strategy for single shot face detector, which is applied on the problem of unconstrained face detection.

A feature fusion module is introduced to balance between the computational cost and the accuracy of the face detector. The module combines both dilated convolution and the small-kernel-size convolution in the module, which marginally improves the accuracy, especially on small objects. Furthermore, a training strategy is proposed,

which refers to triple loss training, for FPN based face detector. During training, it takes the advantage of hierarchical loss from both forward and backward paths. During the evaluation module, however, only feature maps from the second level will be utilised, which improves the accuracy without affecting the inference efficiency.

Experimental results indicate that the proposed FFM and the triple loss training strategy are effective for identifying hard faces. Taking VGG-16 as the backbone, the proposed model achieves the state-of-the-art on the hard subset of the WIDER FACE validation dataset, when compared with other VGG-16 based face detectors. By assigning a larger anchor size, the performance can be further improved on the easy and medium subset. Without bells and whistles, the proposed method achieves comparable results on multiple common face detection benchmarks, when compared with other large-scale face detectors.

As the performance of the proposed network relies heavily on the scales of anchor setting, the future work will focus on the removal of anchor prior, i.e. anchor free [119, 121], to the model.

Chapter 6

SAFDet: A Semi-anchor-free Detector for Effective Detection of Oriented Objects in Aerial Images

6.1 Introduction

Object detection aims to detect the existence of objects, rather than focusing on the outline of object as the segmentation task. As a result, the shape of an object is represented by a rectangle, which is referred to as "bounding box" (bbox). By using bbox, the size of an object can be easily described by the height and the width. More specifically, the target of detection becomes finding out the centre coordinate, height and width of the object bbox. For Convolutional Neural Network (CNN) based object detection, detector extracts feature map using a CNN extractor, and then it uses two sibling convolution layers (detection head) to conduct classification (classifier) and bbox regression (regressor) at each pixel. For foreground pixels, the regressor regresses the sizes and the centre coordinates of bboxes based on either a predefined bbox or the stride and the pixel coordinates of the feature map. Thus, the predefined bbox, defined with human priority, is known as "anchor", and the detector based on

anchor is called "anchor-based" detector. As opposed to anchor-based detectors, detectors regress bboxes from the stride and the pixel coordinates of the feature map are "anchor-free".

Bboxes can be split into two categories: horizontal bounding box (HBB) and oriented bounding box (OBB), as shown in Figure 6.3. Edges in HBB are parallel with the image edges, so a HBB can be mathematically expressed by $[x, y, h, w]$, which are the centre coordinate, height and width of bbox, respectively. Different from HBB, OBB is rotated based on its horizontal position. As a result, an OBB can be expressed by $[x, y, h, w, \theta]$, where θ is the rotation angle with respect to the x-axis.

Oriented bounding box (OBB) is more preferable than horizontal bounding box (HBB) in accurate object detection. Most of existing work utilise a two-stage detector for locating the HBB and OBB, respectively, which have suffered from the misaligned horizontal proposals and the interference from the complex background. To tackle these issues, region of interest transformer and attention models were proposed, yet they are extremely computationally intensive.

In this chapter, a semi-anchor-free detector (SAFDet) is proposed for effective detection of oriented objects, in which two novel modules are introduced to suppress the noise caused by horizontal anchors from Region Proposal Network (RPN) without sacrificing the computational efficiency. First, a rotate-anchor-free branch (RAFB) is utilised to compensate the limitation caused by horizontal anchors. As illustrated in Figure 6.1, the RAFB network directly predicts the OBB without any predefined rotate anchors. Thus, for each object, RPN can jointly predict the HBB and OBB. Experimental results indicate that, by using the horizontal ROIs for the ROI pooling [3], RAFB improves the detection performance by 1.6% without increasing the inference computational cost. Second, a center prediction module (CPM) is proposed on RPN for suppressing the noise caused by background. In CPM, only the pixels around the object's center are classified as the foreground, rather than to consider all the pixels in the bounding box [39, 60]. Moreover, CPM is implemented during training only, hence it will not increase the computational cost of inference yet further improves 1% of accuracy in the experiments.

(a) (b) (c)

Figure 6.1: Examples of ROI prediction using horizontal anchor (a), rotate anchor (b) and the proposed method (c), respectively. In each example, only a single anchor is shown for clarity. Anchors are denoted by red, while the predicted bounding boxes are labeled by yellow. As the rotate ROIs in the proposed method are predicted via anchor free, they are specially highlighted by dashed blue.

The remaining parts of this chapter are organised as follows: 1. Section 6.2 introduces the design of the proposed approach, especially the RAFB and the CPM modules; 2. Section 6.3 presents the experimental results, including ablation study and discussion; 3. Some concluding remarks are drawn in Section 6.4.

6.2 The Proposed Method

6.2.1 Overall Architecture

The flowchart of the proposed SAFDet detector is sketched in Figure 6.2. Following the work in [48, 60], the ResNet [17] is taken as the backbone and denote the outputs for "conv2", "conv3", "conv4", and "conv5" of the Residual Network (ResNet) as {C2, C3, C4, C5}, respectively. Existing two-stage networks [47, 48] adopt the C4 as the feature map, where the total stride(s) is large (s=16). As a result, the resolution of the feature map is too coarse for detecting small objects, such as vehicles, ships and bridges. When applying the feature pyramid network (FPN) [33] on the backbone, the total stride will be reduced to 4, causing an increase on the computational cost of the prediction. To balance the accuracy and the computational cost, the {C3, C4} are used in the FPN. Hence, the total stride becomes 8. In the FPN, only a single convolutional layer is applied on C3 to adjust the number of channels without using any other enhancement

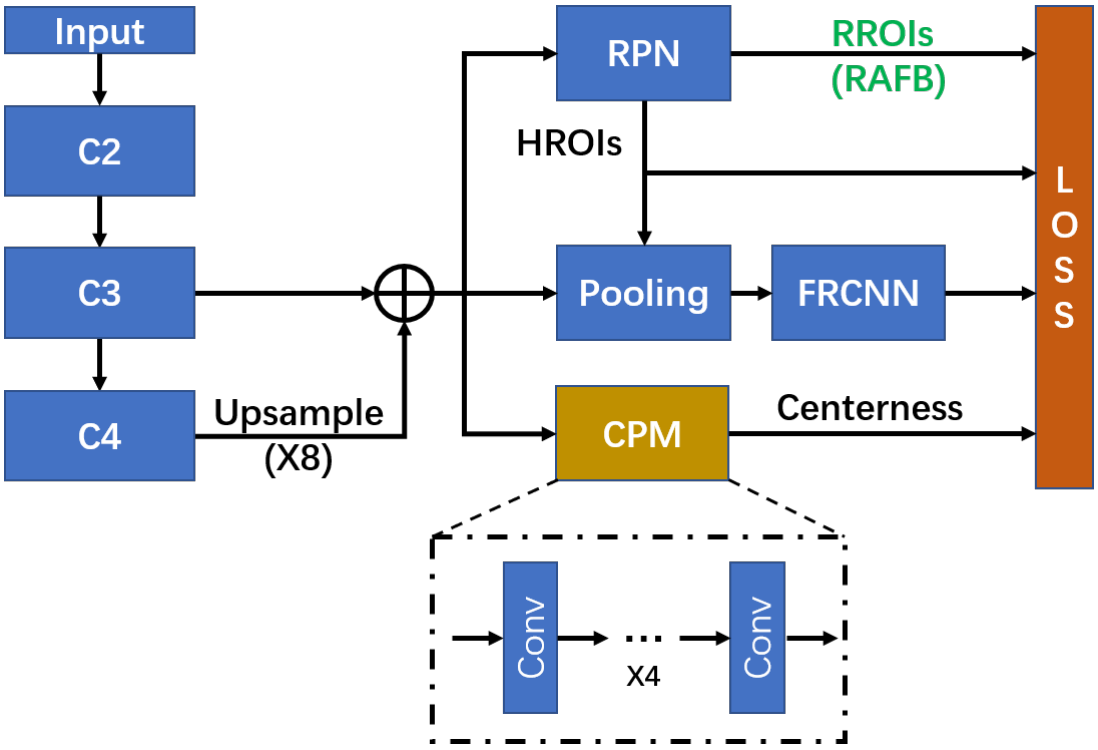


Figure 6.2: The proposed network structure.

modules [54, 60]. For regressing the bounding box of RPN, the horizontal anchor is utilised to save the computational cost and predict the corresponding rotate bounding box (RBB) using an anchor-free method. The horizontal ROIs will be used in the following ROI pooling layer [3]. For the fast RCNN, the network predicts both OBB and the corresponding HBB as in [47, 60, 61, 221]. Relevant details are presented in the following sections.

6.2.2 Rotation Anchor Free Branch

As described in R3Det [61], the rotate anchor setting in the RPN achieves a better performance when compared with horizontal box prediction. However, rotate anchor setting generates more boxes at each point of the feature map, which significantly increases the computational cost at the same time. To improve the performance without increasing the computational cost, for each horizontal box prediction, the proposed method additionally predicts its corresponding OBB using an anchor-free method, which acts

as an auxiliary loss during the training. For the original horizontal branch, the typical four-element presentation method [3,32,47] is adopted. The four elements can be mathematically expressed as $(x_c, y_c, w_{hbb}, h_{hbb})$, denoting respectively the center coordinates, the width and the height of a box. Similarly, the OBB is encoded in the same way as the HBB with one additional element for the rotation angle. As a result, the five elements are $(x_c, y_c, w_{rot}, h_{rot}, \theta)$, where w_{rot} and h_{rot} denote the width and the height of a rotate box, respectively. Consistent with OpenCV [222], θ is the acute angle to the x-axis, ranging in $[-90, 0)$. As the center coordinates of HBB and OBB are the same [146], during training, the prediction of the center coordinates is shared between HBB and RBB. To summarise, the output of the RPN for each box prediction, after the decoding method, has in total seven elements, i.e. $(x_c, y_c, w_{hbb}, h_{hbb}, w_{rot}, h_{rot}, \theta)$.

6.2.2.1 Regression Scheme of the Rotation Branch

As the rotation branch is anchor free, only horizontal anchors are defined for the RPN. An OBB is assigned as the foreground if its corresponding HBB is assigned as the foreground, i.e. the OBB matching is based on the intersection over union (IoU) value between the horizontal ground truth box and the horizontal anchor, as in [32, 52]. For rotate box regression in RPN, the regression method of the centre coordinates are the same as in HBB due to coordinate sharing, and directly predict the size (w_{rot}, h_{rot}) and the rotation angle (θ) using:

$$\begin{aligned} w_{rot} &= \exp(w_{out}) * s; \\ h_{rot} &= \exp(h_{out}) * s; \\ \theta &= -90^\circ + \frac{\theta_{out}}{\pi} * 180^\circ; \end{aligned} \tag{6.1}$$

where w_{rot}, h_{rot} and θ are the predicted width, height and the rotation angle by detector, respectively; w_{out}, h_{out} and θ_{out} are the corresponding encoded outputs from the prediction layer, respectively; s indicates the total stride of the feature map. Similar to [3, 119, 121], the exponential function, denoted as "$exp()$" above, is applied to keep the output in positive. As the prediction is conducted on the feature map, which is

down-sampled by the total stride (s), the s is multiplied to scale the predicted box size to its actual size. The range of θ is in $[-90, 0)$, the magnitude of which is quite larger than other elements. To balance the magnitude between different elements, the regressor predicts the radian using Eq. 6.1. As a result, the range of θ_{out} is in $[0, \frac{\pi}{2})$.

6.2.3 Center Prediction Module

As mentioned in [60, 146], existing challenges contain low resolution, noise, crowd, and freely oriented objects. Previous works tackle these challenges by using supervised attention structures [39, 60], which consider all the pixels within the boxes as the foreground. However, on the one hand, the anchors where the centres are located at the boundary of the bounding boxes, may have low IoU matching values than the predefined threshold, especially for those boxes with large aspect ratios (height/width). On the other hand, the receptive field of boundary pixels are the same as the centre pixel. It means that the boundary pixels contain more background information, which may interfere with the feature extraction. As a result, these boundary pixels are supposed to be categorised as the background rather than the foreground, which indicates that previous labelling methods of attention structures may introduce more noise. To remove the ambiguous instance and enhance the foreground features against the complex background, a supervised centre prediction module (CPM) is proposed, as illustrated in Figure 6.2. To be more specific, the structure consists of four convolutional layers for feature extraction and one binary prediction layer, which is the same as the RetinaNet [114]. Different from the existing labelling methods [39, 60], only the centre pixel is taken as the foreground, and the remaining pixels in the bounding box are all labelled as the background. Let the prediction score and the ground truth at the position (i, j) be p_{ij} and y_{ij}, respectively. the variant focal loss is deployed as in [118, 121]:

$$L_{CPM} = -\frac{1}{N} \sum_{i=1}^{H/s} \sum_{j=1}^{W/s} \alpha_{ij} FL\left(p_{ij}, y_{ij}\right); \tag{6.2}$$

where H and W denote the height and the width of the original image, respectively; s is the total stride, which is the same as in Eq. 6.1; N is the normalisation factor.

Specifically, $FL(*)$ is the original focal loss [114] and α_{ij} is the adjusting factor:

$$FL\left(p_{ij}, y_{ij}\right) = \begin{cases} \left(1-p_{ij}\right)^2 \log\left(p_{ij}\right), & \text{if } y_{ij}=1 \\ \left(p_{ij}\right)^2 \log\left(1-p_{ij}\right), & \text{otherwise} \end{cases}; \tag{6.3}$$

$$\alpha_{ij} = \begin{cases} 1, & \text{if } y_{ij}=1 \\ \left(1-\max_{k=1,2,\ldots,K} G_k\left(i,j,x_c^k,y_c^k,w_k,h_k\right)\right)^4, & \text{otherwise} \end{cases}; \tag{6.4}$$

where K is the total number of objects. x_c^k, y_c^k, w_k, and h_k are the center coordinates, width and height of the horizontal bounding box k, respectively. $G_k(*)$ is a Two-Dimensional (2D) Gaussian mask centred at the bounding box k, which is formulated as:

$$G_k\left(i,j,x_c^k,y_c^k\right) = \exp\left(-\left(\frac{\left(i-x_c^k\right)^2}{2\sigma_{w_k}^2} + \frac{\left(j-y_c^k\right)^2}{2\sigma_{h_k}^2}\right)\right); \tag{6.5}$$

where the variances $(\sigma_{w_k}^2, \sigma_{h_K}^2)$ are proportional to the width (w_k) and height (h_k) of individual objects.

Experimentally, assigning N as the number of all instances on the feature map gives a better performance than the original setting, where the loss is normalised by the number of objects in the image.

6.2.4 Loss Function

In summary, the total loss of the network is defined as:

$$L_{tot} = \lambda_1 L_{CPM} + \lambda_2 L_{rotaf} + L_{RPN} + L_{FRCNN}; \tag{6.6}$$

where λ_1 and λ_2 are the weight factors; L_{CPM} and L_{rotaf} are the losses of center prediction module and the anchor-free rotate box prediction branch, respectively. L_{RPN} is the total loss of the region proposal network as in Faster-RCNN [3]. L_{FRCNN} is the total loss of the fast RCNN (Regions with CNN features) subnet [3], including HBB regression loss, OBB regression loss and the category prediction loss as in [47, 60, 61]. To balance the magnitudes between loss elements, $\lambda_1 = 1$ and $\lambda_2 = 0.1$ are assigned

during the following experiments.

6.3 Experiments and Analysis

Two benchmark datasets, DOTA [146] and HRSC2016 [57], are chosen for ablation study and performance evaluation, which are well known for oriented object detection. Relevant details are presented as follows.

6.3.1 Datasets and Implementation Details

6.3.1.1 DOTA

The benchmark DOTA (**D**ataset for **O**bject de**T**ection in **A**erial images) [146] is an aerial image dataset for object detection. Images of DOTA are sampled from multiple sensors and platforms, where the sizes range from around 800×800 to $4,000 \times 4,000$ pixels. There exist 2,806 aerial images and 188,282 instances in total. The annotations of objects are various on scales, oriented angles, and aspect ratios, which are classified in 15 commonly used categories. The names of these categories and their abbreviations used in the chapter are PL-Plane, BD-Baseball diamond, BR-Bridge, GTF-Ground field track, SV-Small vehicle, LV-Large vehicle, SH-Ship, TC-Tennis court, BC-Basketball court, ST-Storage tank, SBF-Soccer-ball field, RA-Roundabout, HA-Harbor, SP-Swimming pool, and HC-Helicopter. The ratios of the training set, validation set, and test set are 1/2, 1/6 and 1/3 of the total number of samples, respectively. There are two detection tasks in the DOTA dataset, where task 1 is for OBB regression, while task 2 is for HBB regression. As the aim of this chapter is for improving the performance on OBB detection, the task 1 is focused in the experiments.

6.3.1.2 HRSC2016

The HRSC2016 dataset (high resolution ship collections 2016) [57] collects 1061 images with 2976 annotated instances from six famous harbors, which are widely used for ship detection. The scenarios contain ships both on sea and close inshore. The image sizes

range from 300×300 to $1,500 \times 900$ pixels. The numbers of the training set, validation set, and test set are 436, 181 and 444 images, respectively.

6.3.1.3 Implementation Details and Evaluation Metrics

When training on DOTA, the original images are uniformly cropped to patches of 800×800 pixels with a stride of 200 pixels. In ablation study, only the training set is utilised, and the result is evaluated on the validation set. When compared with other detectors, both the training and validation sets are applied for training and the results are tested on the test set, which are then submitted to the official evaluation server for evaluation. For image pre-processing, each images patch is first subtracted by the mean values of the ImageNet on the RGB channels. Afterwards, randomly flipping is adopted with a probability of 0.5. The learning rate is initialised as 3e-4 and is divided by 10 at 100k and 200k iterations, respectively. The training terminates at 300k iterations.

For the HRSC2016 dataset, both the training and validation sets are utilised for training. The training images are resized to $(800, 1024)$, where 800 and 1024 indicate respectively the short and the maximum size of the image. The pre-processing steps are the same as used for training on DOTA. Thus, no additional augmentation methods, e.g. image pyramid and image rotation, are applied. The total iterations are 12k. The initial learning rate is 5e-4, which is divided by 10 on 9k iterations. Results are tested on the original image size, and then evaluated using the standard VOC-style Average Precision (AP) metrics [48, 61, 221].

6.3.2 Ablation Study

In this section, a serious of ablation studies are conducted to validate the effect of each component in the proposed approach. During the ablation study, the ResNet-50 is taken as the backbone, and relevant models are implemented by the Tensorflow [223].

6.3.2.1 Baseline Setup.

As the proposed method is an improved two-stage object detector, the baseline starting built on the Faster-RCNN. Previous works [47, 48, 60, 112] indicate that jointly predict-

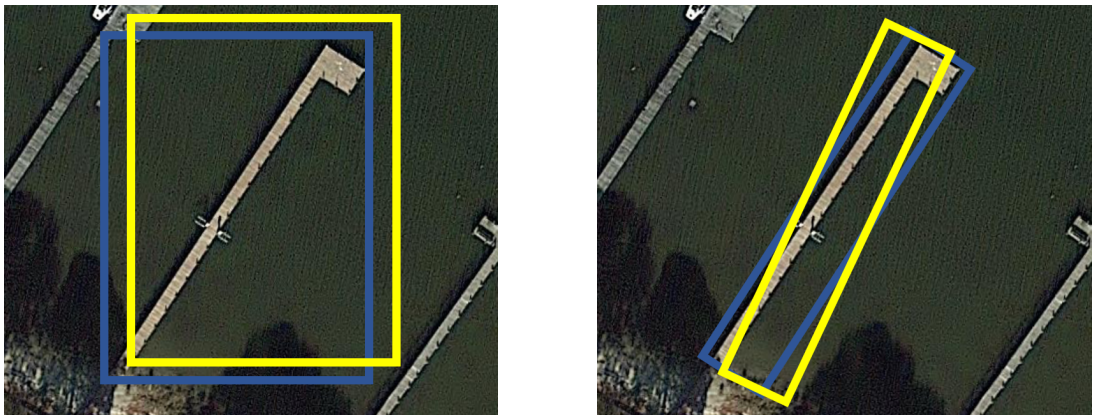
ing the OBB and its boundary rectangle (HBB) can improve the performance of OBB prediction. As a result, in the baseline, HBB is also predicted during training, which was actually an example of R2CNN [47] with a single ROI pooling kernel rather than multiple ROI pooling kernels in the original R2CNN.

The total stride is another key factor that affects the performance of the object detectors. Faster-RCNN sets the total stride as 16, which is too large for detecting small objects (e.g. ship, small vehicles, and bridge et al.) for DOTA. Thus, a top-down pathway is essential for increasing the resolution of the feature map [33, 60]. With the decreased total stride, the accuracy increases, especially for small object detection, as highlighted in Table 6.1. As can be seen, the R2CNN has the lowest computational speed when the total stride is 16, though it achieves the highest mean average precision (mAP) when the total stride is 4. As a result, to balance the computational cost and the accuracy, the total stride is set to 8 in the baseline R2CNN.

6.3.2.2 Effect of Rotation Anchor Free Branch.

As discussed in Section 6.2, Multi-task bounding box regression brings benefits for performance improvement. The effect of rotation-anchor-free-branch (RAFB) is summarised in Table 6.2. As seen, the performance is improved by 1.59% without increasing the computational cost of the detector. As for each category, RAFB further improves the detection performance on instances with large aspect ratios and/or low resolutions, e.g. 5.7% on bridges, 5.8% on helicopters, and 7.5% on ground field tracks.

Anchor vs Anchor Free. To make a contrast, the network is also trained using RAFB for proposal prediction on RPN, which directly predicts the OBB rather than the HBB. As the original IoU matching scheme is unsuitable for anchor-free methods, the matching scheme as in CSP [121] is directly used. However, the mAP is only 38%. This is different from the observations on HBB detection [118,119,121], that anchor-free outperforms anchor-based methods. It is deduced that is caused by two folds. First, for two-stage HBB detection, a slight shift on the x-y coordinate does not significantly affect the feature extraction as most of object region is still within the bounding box. However, for OBB detection, a slight error on the rotate angle will cause missing on



(a) HBB (b) OBB

Figure 6.3: Examples of bounding box prediction on HBB (a) and OBB (b). Ground truth bounding boxes are denoted by blue, while predicted bounding boxes are colored by yellow. Only the centre target are labelled for clarity

most of the regions, leading to information loss on feature extraction, especially for instances with large aspect ratio. The information loss on feature extraction will then affect the performance of the second stage prediction. Similar observations, on the effect of angle error, are reported in [47, 48]. An example of HBB detection and OBB detection on the same object is shown in Figure 6.3. Second, the CSP matching scheme is originally designed for HBB box prediction. As proposed in [119, 224], the matching scheme is vitally important for anchor-free detectors, which motivates the future work for proposing an anchor-free matching scheme for OBB detection.

6.3.2.3 Effect of Center Prediction Module.

As previously discussed, the center prediction module (CPM) is beneficial to locate the center of objects. The CPM is only implemented during training, hence it will not increase the computational cost of inference, as shown in Table 6.2. With the CPM, the mAP of detector is further improved by 1%. Compared with RAFB, which improves the performance on low-resolution objects, CPM can further bring benefits on detection of large-scale objects, e.g. baseball diamond, basketball court and soccer-ball field.

An additional experiment is conducted, in which CPM is replaced by the attention module and the results are shown in Table 6.2 for comparison. When adopting the

attention module into the detector, the mAP increases by 0.2% only. On the one hand, this improvement is not as significant as CPM is applied. On the other hand, it also increases the computational cost of the detector. When assigning the attention module as an auxiliary loss, the mAP drops by about 0.6%. Therefore, the results indicate that CPM is more effective than the attention module for OBB detection in aerial images.

6.3.3 Comparison with the State-of-the-Art Detectors

6.3.3.1 DOTA

When compared with the state-of-the-are methods, image pyramid is popularly utilised [60,61,221]. Deeper backbone, such as ResNet-101, is adopted for a fair comparison. As image pyramid requires resize and redetect an image multiple times, it is meaningless to measure the inference time. Same as the previous works, when compared with other methods, the inference time is not measured on DOTA. The proposed semi-anchor-free detector is denoted as "SAFDet". As shown in Table 6.3, the good results from RoI-Transformer, SCRDet and R3Det are gained from feature extraction and feature fusion, in which the improved accuracy costs on the significantly decreased computational speed. SAFDet lags SCRDet by about 1.3%, but it has no additional feature extraction or fusion module, reaching a good balance between the accuracy and computational cost. Experimental results in R3Det [61] indicate that SCRDet improves the accuracy at a cost of much-degraded efficiency. As a result, the speed of SCRDet is lower than R3CNN. On the contrary, a comparison of inference speed, as detailed in the next section, shows that the proposed method is much faster than R3Det.

6.3.3.2 HRSC2016

Table 6.4 illustrates the results in comparison to the state-of-the-art methods on the HRSC2016 dataset. As seen, the proposed method reaches 89.38% mAP without adopting any image pyramid, which outperforms all other methods. The inference speed with respect to each detector are also listed in Table 6.4. The relationship between the accuracy and inference is illustrated in Figure 6.4. As seen, the proposed method achieves 11 fps on the HRSC2016 dataset, which further validates that the SAFDet method has

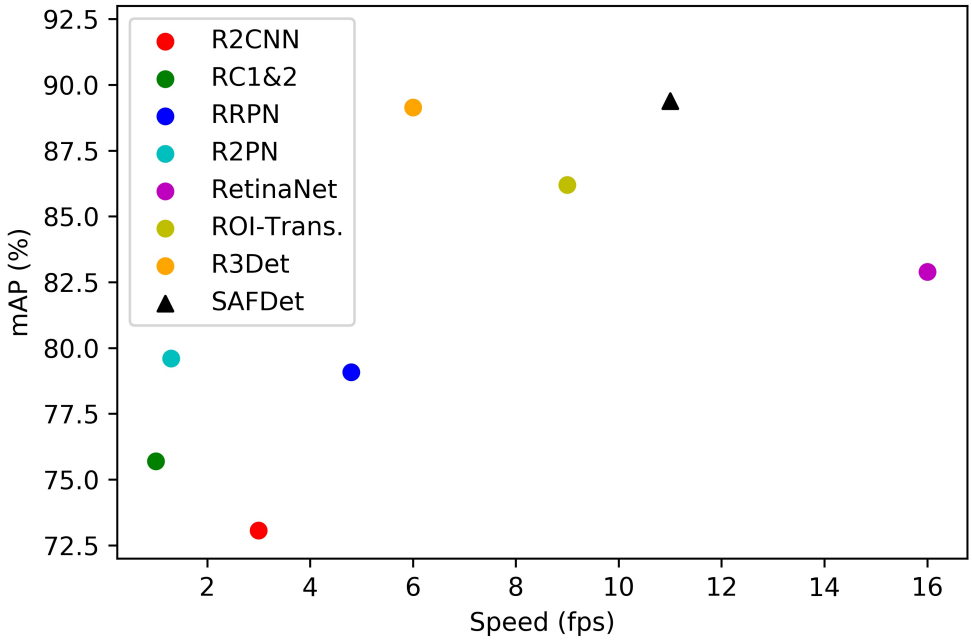


Figure 6.4: Accuracy versus speed on HRSC2016 dataset.

apparent advantages in well balancing between the accuracy and the computational cost.

6.4 Summary

In this chapter, a semi-anchor-free detector (SAFDet) is proposed for effective detection of oriented objects in aerial images. Two new modules are introduced to tackle the detection difficulty of low-resolution, noisy, large aspect ratio and freely oriented objects. Rotation anchor free branch is introduced to assist the HBB prediction on RPN, with negligible extra computation cost during the training. Similar to the attention module, CPM is proposed to suppress the background information and enhance the foreground information. However, CPM is only implemented for training, so that it does not increase inference cost. By adopting those two modules, the proposed model improves the mAP by about 3% without bringing any additional computational cost on inference. When compared with the state-of-the-art detectors, the proposed method

achieves effective results on the challenging DOTA and HRSC2016 datasets without lowering its high efficiency.

Table 6.1: The performances of R2CNN with different total strides. Models are trained and validated on the DOTA dataset (%). Computational time are measured on a single V100 GPU with the input size 800×800. The names of these categories and their abbreviations used in the chapter are PL-Plane, BD-Baseball diamond, BR-Bridge, GTF-Ground field track, SV-Small vehicle, LV-Large vehicle, SH-Ship, TC-Tennis court, BC-Basketball court, ST-Storage tank, SBF-Soccer-ball field, RA-Roundabout, HA-Harbor, SP-Swimming pool, and HC-Helicopter.

Total Strides	**mAP**	**Time (ms)**	**PL**	**BD**	**BR**	**GTF**	**SV**	**LV**	**SH**	**TC**	**BC**	**ST**	**SBF**	**RA**	**HA**	**SP**	**HC**
16	56.22	**65**	86.09	65.02	22.73	49.46	38.26	49.84	52.26	90.44	53.60	74.03	61.95	61.70	55.14	47.92	34.88
8	61.53	72	88.17	68.54	30.64	53.10	49.41	59.43	63.76	90.73	52.34	83.29	63.80	62.09	63.24	52.71	41.83
4	**65.08**	238	88.99	71.13	**40.54**	61.83	**51.60**	**66.99**	**68.33**	90.73	**59.81**	83.65	63.65	62.72	64.60	**52.92**	**48.72**

Table 6.2: Effect of each component in the proposed method on the DOTA dataset (%), where 'RAFB' indicates the rotation anchor free branch; 'Att' is the attention module and 'CPM' denotes the center prediction module. 'loss only' means that the attention module is only applied during training, as an auxiliary loss. The names of these categories and their abbreviations used in the chapter are PL-Plane, BD-Baseball diamond, BR-Bridge, GTF-Ground field track, SV-Small vehicle, LV-Large vehicle, SH-Ship, TC-Tennis court, BC-Basketball court, ST-Storage tank, SBF-Soccer-ball field, RA-Roundabout, HA-Harbor, SP-Swimming pool, and HC-Helicopter.

Methods	mAP	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC
R2CNN-8 (Baseline)	61.53	88.17	68.54	30.64	53.10	49.41	59.43	63.76	90.73	52.34	83.29	63.80	62.09	63.24	52.71	41.83
+RAFB	63.12	88.59	68.57	36.92	61.68	48.65	61.57	57.86	90.53	60.44	79.30	71.95	63.75	58.47	50.88	47.67
+RAFB+ATT	63.33	88.81	72.28	38.04	64.10	47.33	60.37	58.67	90.65	57.89	79.01	72.01	64.29	57.48	51.54	47.53
+RAFB+ATT (loss only)	62.46	88.86	69.86	35.01	62.87	48.59	60.49	58.91	90.32	55.10	79.27	69.34	60.56	57.96	51.80	47.35
+RAFB+CPM	64.17	88.63	71.34	37.43	63.51	47.11	62.04	58.62	90.71	64.20	79.23	74.91	63.46	58.06	52.86	50.44

Table 6.3: Quantitative comparison with the state-of-the-art methods on the Task 1 of DOTA (%). The names of these categories and their abbreviations used in the chapter are PL-Plane, BD-Baseball diamond, BR-Bridge, GTF-Ground field track, SV-Small vehicle, LV-Large vehicle, SH-Ship, TC-Tennis court, BC-Basketball court, ST-Storage tank, SBF-Soccer-ball field, RA-Roundabout, HA-Harbor, SP-Swimming pool, and HC-Helicopter.

Methods	**mAP**	**PL**	**BD**	**BR**	**GTF**	**SV**	**LV**	**SH**	**TC**	**BC**	**ST**	**SBF**	**RA**	**HA**	**SP**	**HC**
FR-O [146]	52.93	79.09	69.12	17.17	63.49	34.20	37.16	36.20	89.19	69.60	58.96	49.4	52.52	46.69	44.80	46.30
R-DFPN [112]	57.94	80.92	65.82	33.77	58.94	55.77	50.94	54.78	90.33	66.34	68.66	48.73	51.76	55.10	51.32	35.88
R2CNN [47]	60.67	80.94	65.67	35.34	67.44	59.92	50.91	55.81	90.67	66.92	72.39	55.06	52.23	55.14	53.35	48.22
RRPN [58]	61.01	88.52	71.20	31.66	59.30	51.85	56.19	57.25	90.81	72.84	67.38	56.69	52.84	53.08	51.94	53.58
ICN [54]	68.20	81.40	74.30	47.70	70.30	64.90	67.80	70.00	90.80	79.10	78.20	53.60	62.90	67.00	64.20	50.20
RoI-Trans. [48]	69.56	88.64	78.52	43.44	75.92	68.81	73.68	83.59	90.74	77.27	81.46	58.39	53.54	62.83	58.93	47.67
SCRDet [60]	72.61	**89.98**	80.65	**52.09**	68.36	68.36	60.32	72.41	**90.85**	**87.94**	**86.86**	**65.02**	**66.68**	**66.25**	68.24	**65.21**
R3Det [61]	71.69	89.54	**81.99**	48.46	62.52	**70.48**	**74.29**	**77.54**	90.80	81.39	83.54	61.97	59.82	65.44	67.46	60.05
SAFDet (proposed)	71.33	89.78	80.77	46.61	**75.31**	66.47	60.41	66.43	90.27	82.94	85.88	63.19	62.33	65.06	**70.27**	64.29

Table 6.4: Results comparison with the state-of-the-art methods on HRSC2016 (%). As the HRSC2016 is a ship detection dataset, so mAP is also the AP of ship detection.

Methods	**mAP**	**Inference Speed**
R2CNN [47]	73.07	3fps
RC1 & RC2 [225]	75.70	1fps
RRPN [58]	79.08	4.8fps
R2PN [59]	79.60	1.3fps
RetinaNet [61]	82.89	16fps
RRD [226]	84.30	slow
RoI-Trans. [48]	86.20	9fps
R3Det [61]	89.14	6fps
SAFDet (proposed)	**89.38**	11fps

Chapter 7

Conclusions and Future Works

7.1 Conclusions

The general objective of this thesis is to develop accuracy-efficiency balanced techniques for Convolutional Neural Network (CNN) applications of RGB image. These include various methodologies for classification, face detction and aerial object detection, as presented in Chapters 4, 5 and 6 respectively. The main contributions of the thesis are summarised in detail as follows.

i. In Chapter 4, the genetic algorithm (GA) is adopted to optimise the structure of the CNNs, where the input channels are encoded by a binary string for removing the redundant features from the feature map. Firstly, two encoding methods are proposed to enable the GA to evolve the structure of the model. Then, three genetic operations are adopted, including selection, crossover and mutation. To further reduce the computational cost during training, a pre-trained weight inheritance method is proposed, where the structure is fine-tuned on a pretrained baseline. Experimental results indicate that the proposed model can reach competitive classification accuracy, while requires significantly fewer parameters. Rather than generating structures with a large searching space, the proposed method is built based on predefined constraints, which has significantly reduced the computational cost and is capable of generating a well performing structure under limited training generations. When tested on the CIFAR-10 dataset, the proposed model saves

more than 90% of the parameters in comparison to the state-of-the-art models. The comparable results from the ImageNet has validated that the proposed model is robust in both the variance and the scale of the classification tasks.

ii. Chapter 5 presents a novel single shot face detector to tackle the problem of unconstrained face detection, which improves the performance by optimising both structure and loss function. To be concrete, an accuracy-computational cost balanced feature fusion module as well as a novel training strategy are proposed. The proposed feature fusion module is introduced to balance the computational cost and the accuracy of the face detector. Dilated convolution has been combined with the small-size-kernel convolution in the module, which marginally improves the accuracy, especially on small objects. Furthermore, a training strategy named as triple loss training has been proposed for Feature Pyramid Network (FPN) based face detector. It takes the advantage of hierarchical loss from both forward and backward paths during training. while within the evaluation, only feature maps from the second level will be utilised, which improves the accuracy without affecting the inference efficiency. Experimental results indicate that the proposed FFM and the triple loss training strategy are effective for identifying hard faces. Taking VGG-16 as the backbone, the proposed model achieves the state-of-the-art accuracy on the hard subset of the WIDER FACE validation dataset, when compared with other VGG-16 based face detectors. By assigning a larger anchor size, the performance can be further improved on the easy and medium subset. Without bells and whistles, i.e. no augmentation methods applied to the input image, the proposed method achieves comparable results on multiple common face detection benchmarks when compared with other large-scale face detectors.

iii. Chapter 6 focuses on tackling the detection difficulty of low-resolution, noisy and large aspect ratio on oriented aerial image detection. Two modules have been proposed to optimise the performance of the commonly used two-stage detector. First, RAFB is introduced to assist the HBB prediction on RPN, which predicts the OBBs of objects with HBB simultaneously. However, as no anchors are re-

quired on RAFB, the increase in computation cost is negligible. After that, CPM is proposed to suppress the background information and enhance the foreground information. The structure of CPM is similar to the attention module, but it is only implemented as an auxiliary module and will be discarded during inference. By adopting those two modules, the proposed model improves the mAP by about 3% without bringing any additional computational cost on inference. When compared with the state-of-the-art detectors, the proposed method achieves comparable results on the challenging DOTA and HRSC2016 datasets, without impeding its high efficiency.

7.2 Future Works

Followed by the results and conclusions highlighted in this thesis, the potential directions for future research are summarised as follows.

1. For the CNN channel selection method proposed in Chapter 4:

   (1). At the current stage, the proposed approach is only validated on CNN. In the future, it can be evaluated using other popular deep learning networks, such as RNN and LSTM.

   (2). The proposed approach improves the performance by optimising the CNN structure, which is a global optimisation strategy, while local optimisation methods, which improve the performance of CNN by optimising the layers in CNN, have also achieved remarkable results. This inspires the future work to explore the possibility of combining the global optimisation methods with local optimisation methods.

   (3). Existing methods initialise genotype string by a B(0.5), but it is not optimal as half of the channels are discarded in the initial generation. The future work may also investigate into the initialisation method to reduce the total evolving iterations.

   (4). Recent NAS methods search the architectures via the gradient during training, which can further reduce the training cost, but the hyperparameter setting is

still complex. Therefore, the future work could find an approach to reduce the hyperparameters with machine learning algorithm.

(5). Existing works search architectures based on supervised learning. However, for some of the classification tasks, the sample size is small. Searching architectures using few-shot learning or unsupervised learning is still a challenging task.

2. For the single shot face detector proposed in Chapter 5:

(1). Since the performance of the proposed network relies heavily on the scales of anchor setting, the future work could study on the removal of anchor prior, i.e. anchor free.

(2). At the current stage, the proposed approach is constrained by the batch size. It is essential to propose a low-batch-size-trained ($batchsize < 16$) detector whose performance is still comparable with models trained by large batch size.

(3). Experimental results indicates that combining Three-Dimensional (3D) information with Two-Dimensional (2D) information significantly improves performance on accuracy. In the future, face detection may be conducted via both 2D and 3D for improving the detection accuracy.

3. For the SAFDet proposed in Chapter 6:

(1). Existing anchor-free matching schemes is originally designed for HBB regression, which can not well address the OBB regression challenges. This encourages the development of a new anchor-free matching scheme for OBB matching.

(2). Existing anchor-based matching schemes match ground truths to anchors via IoU threshold. However, this value is always low for OBB matching, causing a higher possibility of mismatching on positive proposals. A rotate IoU matching method is required especially for the OBB in the future work.

4. Future works for general object detection

(1). Current CNN detectors still consist of lots of post-processing steps, such as decoding and non maximum suppression (NMS), which requires additional manual

work to adjust those hyperparameters. These post-processing steps should be simplified in the future work.

(2). An object is presented by a bounding box, but it is inappropriate if part of the object is blocked. An alternative method could be inferring an object based on "smaller" segments, such as eyes, nose, ears for face detection.

(3). The cost of bbox labelling is tremendous, and more studies on applying few-shot learning or unsupervised learning to object detection are needed.

(4). There are plenty of publicly available datasets about 3D object detection, which enables further investigation into 3D object detection.

(5). Most foreground objects take the minority area in images, while the entire images are convolved by multiple convolution layers during feature extraction. The computational cost can be expected to be reduced by removing the background patches at a early stage of CNN.

Bibliography

[1] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[4] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[5] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in neural information processing systems*, pp. 2802–2810, 2016.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *international Conference on computer vision & Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE Computer Society, 2005.

[8] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 12, pp. 2037–2041, 2006.

[9] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[10] D. P. Solomatine and D. L. Shrestha, "Adaboost. rt: a boosting algorithm for regression problems," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2, pp. 1163–1168, IEEE, 2004.

[11] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[12] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[13] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[19] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in neural information processing systems*, pp. 550–558, 2016.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[21] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European conference on computer vision*, pp. 391–405, Springer, 2014.

[22] S. Roy and S. Podder, "Face detection and its applications," *International Journal of Research in Engineering & Advanced Technology*, vol. 1, no. 2, pp. 1–10, 2013.

[23] P. Viola, M. Jones, *et al.*, "Rapid object detection using a boosted cascade of simple features," *CVPR (1)*, vol. 1, no. 511-518, p. 3, 2001.

[24] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 65–86, 2008.

[25] M.-T. Pham and T.-J. Cham, "Fast training and selection of haar features using statistics in boosting-based face detection," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–7, IEEE, 2007.

[26] S. Liao, A. K. Jain, and S. Z. Li, "A fast and accurate unconstrained face detector," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 211–223, 2015.

[27] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Aggregate channel features for multi-view face detection," in *IEEE international joint conference on biometrics*, pp. 1–8, IEEE, 2014.

[28] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, "Face detection without bells and whistles," in *European conference on computer vision*, pp. 720–735, Springer, 2014.

[29] J. Yan, Z. Lei, L. Wen, and S. Z. Li, "The fastest deformable part model for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2497–2504, 2014.

[30] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[31] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[33] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[34] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, "Ssh: Single stage headless face detector," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4875–4884, 2017.

[35] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3fd: Single shot scale-invariant face detector," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 192–201, 2017.

[36] H. Jiang and E. Learned-Miller, "Face detection with the faster r-cnn," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pp. 650–657, IEEE, 2017.

[37] P. Hu and D. Ramanan, "Finding tiny faces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 951–959, 2017.

[38] S. Zhang, X. Wang, Z. Lei, and S. Z. Li, "Faceboxes: A cpu real-time and accurate unconstrained face detector," *Neurocomputing*, 2019.

[39] J. Wang, Y. Yuan, and G. Yu, "Face attention network: An effective face detector for the occluded faces," *arXiv preprint arXiv:1711.07246*, 2017.

[40] J. Zhang, X. Wu, J. Zhu, and S. C. Hoi, "Feature agglomeration networks for single stage face detection," *arXiv preprint arXiv:1712.00721*, 2017.

[41] X. Tang, D. K. Du, Z. He, and J. Liu, "Pyramidbox: A context-assisted single shot face detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 797–813, 2018.

[42] C. Chi, S. Zhang, J. Xing, Z. Lei, S. Z. Li, and X. Zou, "Selective refinement network for high performance face detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8231–8238, 2019.

[43] S. Qu, K. Huang, A. Hussain, and Y. Goulermas, "Mpssd: Multi-path fusion single shot detector," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, 2019.

[44] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[46] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, 2016.

[47] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2cnn: Rotational region cnn for orientation robust scene text detection," *arXiv preprint arXiv:1706.09579*, 2017.

[48] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, "Learning roi transformer for detecting oriented objects in aerial images," *arXiv preprint arXiv:1812.00155*, 2018.

[49] X. Li and S. Wang, "Object detection using convolutional neural networks in a coarse-to-fine manner," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 11, pp. 2037–2041, 2017.

[50] M. Liao, B. Shi, and X. Bai, "Textboxes++: A single-shot oriented scene text detector," *IEEE transactions on image processing*, vol. 27, no. 8, pp. 3676–3690, 2018.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[52] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[53] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[54] S. M. Azimi, E. Vig, R. Bahmanyar, M. Körner, and P. Reinartz, "Towards multi-class object detection in unconstrained remote sensing imagery," in *Asian Conference on Computer Vision*, pp. 150–165, Springer, 2018.

[55] L. Liu, Z. Pan, and B. Lei, "Learning a rotation invariant detector with rotatable bounding box," *arXiv preprint arXiv:1711.09405*, 2017.

[56] Z. Liu, J. Hu, L. Weng, and Y. Yang, "Rotated region based cnn for ship detection," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 900–904, IEEE, 2017.

[57] Z. Liu, H. Wang, L. Weng, and Y. Yang, "Ship rotated bounding box space for ship extraction from high-resolution optical satellite images with complex backgrounds," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 8, pp. 1074–1078, 2016.

[58] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3111–3122, 2018.

[59] Z. Zhang, W. Guo, S. Zhu, and W. Yu, "Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 11, pp. 1745–1749, 2018.

[60] X. Yang, J. Yang, J. Yan, Y. Zhang, T. Zhang, Z. Guo, X. Sun, and K. Fu, "Scrdet: Towards more robust detection for small, cluttered and rotated objects," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8232–8241, 2019.

[61] X. Yang, Q. Liu, J. Yan, and A. Li, "R3det: Refined single-stage detector with feature refinement for rotating object," *arXiv preprint arXiv:1908.05612*, 2019.

[62] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[63] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[64] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[66] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[67] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[68] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[70] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*, pp. 646–661, Springer, 2016.

[71] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms.," in *ICGA*, vol. 89, pp. 379–384, 1989.

[72] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[73] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.

[74] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[75] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the*

34th International Conference on Machine Learning-Volume 70, pp. 2902–2911, JMLR. org, 2017.

[76] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1379–1388, 2017.

[77] Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks for image classification," *arXiv preprint arXiv:1710.10741*, 2017.

[78] F. Gruau, "Genetic synthesis of modular neural networks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 318–325, Morgan Kaufmann Publishers Inc., 1993.

[79] M. Kim and L. Rigazio, "Deep clustered convolutional kernels," in *Feature Extraction: Modern Questions and Challenges*, pp. 160–172, 2015.

[80] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pp. 293–312, Elsevier, 2019.

[81] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[82] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[83] T. Breuel and F. Shafait, "Automlp: Simple, effective, fully automated learning rate and size adjustment," in *The Learning Workshop*, vol. 4, p. 51, Utah, 2010.

[84] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra, "Convolution by evolution: Differentiable pattern producing networks," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 109–116, ACM, 2016.

[85] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[86] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[87] S. Cao, X. Wang, and K. M. Kitani, "Learnable embedding space for efficient neural architecture compression," *arXiv preprint arXiv:1902.00383*, 2019.

[88] X. Dong and Y. Yang, "Network pruning via transformable architecture search," *arXiv preprint arXiv:1905.09717*, 2019.

[89] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *Advances in Neural Information Processing Systems*, pp. 8699–8710, 2018.

[90] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[91] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.

[92] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[93] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[94] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, and X. Wang, "Renas: Reinforced evolutionary neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4787–4796, 2019.

[95] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

[96] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

[97] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[98] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *arXiv preprint arXiv:1901.06032*, 2019.

[99] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[100] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

[101] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[102] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[103] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[104] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[105] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[106] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[107] S. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[108] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[109] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019.

[110] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 497–504, ACM, 2017.

[111] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[112] X. Yang, H. Sun, K. Fu, J. Yang, X. Sun, M. Yan, and Z. Guo, "Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks," *Remote Sensing*, vol. 10, no. 1, p. 132, 2018.

[113] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: an efficient and accurate scene text detector," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 5551–5560, 2017.

[114] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[115] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," *arXiv preprint arXiv:1509.04874*, 2015.

[116] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[117] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750, 2018.

[118] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6569–6578, 2019.

[119] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," *arXiv preprint arXiv:1904.01355*, 2019.

[120] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi, "Foveabox: Beyond anchor-based object detector," *arXiv preprint arXiv:1904.03797*, 2019.

[121] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5187–5196, 2019.

[122] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," *arXiv preprint arXiv:1904.11490*, 2019.

[123] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, 2019.

[124] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[125] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, pp. 379–387, 2016.

[126] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018.

[127] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4203–4212, 2018.

[128] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 821–830, 2019.

[129] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," *arXiv preprint arXiv:1711.07264*, 2017.

[130] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2147–2154, 2014.

[131] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. So Kweon, "Attentionnet: Aggregating weak directions for accurate object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2659–2667, 2015.

[132] M. Najibi, M. Rastegari, and L. S. Davis, "G-cnn: an iterative grid based object detector," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2369–2377, 2016.

[133] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[134] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "Dsod: Learning deeply supervised object detectors from scratch," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1919–1927, 2017.

[135] S. Liu, D. Huang, *et al.*, "Receptive field block net for accurate and fast object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 385–400, 2018.

[136] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," *arXiv preprint arXiv:1901.01892*, 2019.

[137] Y. Abramson, B. Steux, and H. Ghorayeb, "Yef real-time object detection," in *International Workshop on Automatic Learning and Real-Time*, vol. 5, p. 7, 2005.

[138] W. Tian, Z. Wang, H. Shen, W. Deng, B. Chen, and X. Zhang, "Learning better features for face detection with feature fusion and segmentation supervision," *arXiv preprint arXiv:1811.08557*, 2018.

[139] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[140] S. Zhang, X. Zhu, Z. Lei, X. Wang, H. Shi, and S. Z. Li, "Detecting face with densely connected face proposal network," *Neurocomputing*, vol. 284, pp. 119–127, 2018.

[141] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang, "Dsfd: dual shot face detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5060–5069, 2019.

[142] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[143] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

[144] K. Huang, H. Yang, I. King, and M. R. Lyu, "Learning classifiers from imbalanced data based on biased minimax probability machine," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, IEEE, 2004.

[145] S. Chen, J. Li, C. Yao, W. Hou, S. Qin, W. Jin, and X. Tang, "Dubox: No-prior box objection detection via residual dual scale detectors," *arXiv preprint arXiv:1904.06883*, 2019.

[146] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3974–3983, 2018.

[147] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3588–3597, 2018.

[148] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

[149] A. N. Sloss and S. Gustafson, "2019 evolutionary algorithms review," *arXiv preprint arXiv:1906.08870*, 2019.

[150] D. Tauritz, "Introduction to evolutionary computing comp 5970-002/6970-003/6976-v04–auburn university fall 2019–assignment series 2 gpac: A genetic programming & coevolution approach to the game of pac-man," 2019.

[151] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in

Proceedings of the Genetic and Evolutionary Computation Conference, pp. 419–427, 2019.

[152] M. S. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova, "Assemblenet: Searching for multi-stream neural connectivity in video architectures," *arXiv preprint arXiv:1905.13209*, 2019.

[153] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[154] D. R. So, C. Liang, and Q. V. Le, "The evolved transformer," *arXiv preprint arXiv:1901.11117*, 2019.

[155] J.-M. Pérez-Rúa, V. Vielzeuf, S. Pateux, M. Baccouche, and F. Jurie, "Mfas: Multimodal fusion architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6966–6975, 2019.

[156] X. Li, Y. Zhou, Z. Pan, and J. Feng, "Partial order pruning: for best speed/accuracy trade-off in neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9145–9153, 2019.

[157] Y. Xiong, R. Mehta, and V. Singh, "Resource constrained neural network architecture search: Will a submodularity assumption help?," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1901–1910, 2019.

[158] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo, "Evolving space-time neural architectures for videos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1793–1802, 2019.

[159] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*, vol. 1, pp. 69–93, Elsevier, 1991.

[160] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[161] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[162] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, 2009.

[163] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[164] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[165] K. Hornik, M. Stinchcombe, H. White, *et al.*, "Multilayer feedforward networks are universal approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[166] Y. Cho and L. K. Saul, "Large-margin classification in infinite neural networks," *Neural computation*, vol. 22, no. 10, pp. 2678–2697, 2010.

[167] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.

[168] J. Turian, J. Bergstra, and Y. Bengio, "Quadratic features and deep architectures for chunking," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pp. 245–248, 2009.

[169] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[170] R. M. Neal, "Connectionist learning of belief networks," *Artificial intelligence*, vol. 56, no. 1, pp. 71–113, 1992.

[171] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[172] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[173] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[174] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, *et al.*, "On rectified linear units for speech processing," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3517–3521, IEEE, 2013.

[175] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613, IEEE, 2013.

[176] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[177] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, 2013.

[178] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013.

[179] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[180] B. B. Ahn, "The compact 3d convolutional neural network for medical images," *Standford University*, 2017.

[181] H. Le and A. Borji, "What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks?," *arXiv preprint arXiv:1705.07049*, 2017.

[182] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 4898–4906, 2016.

[183] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5695–5703, 2016.

[184] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[185] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4353–4361, 2017.

[186] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[187] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[188] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[189] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, pp. 92–101, Springer, 2010.

[190] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[191] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1476–1481, 2016.

[192] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Advances in neural information processing systems*, pp. 1945–1953, 2017.

[193] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6181–6189, 2018.

[194] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[195] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, pp. 2377–2385, 2015.

[196] S. Zhang, R. Zhu, X. Wang, H. Shi, T. Fu, S. Wang, and T. Mei, "Improved selective refinement network for face detection," *arXiv preprint arXiv:1901.06651*, 2019.

[197] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144, 2017.

[198] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, pp. 694–711, Springer, 2016.

[199] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[200] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, p. 78, 2004.

[201] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[202] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

[203] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[204] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[205] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[206] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," *arXiv preprint arXiv:1204.3968*, 2012.

[207] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[208] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[209] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.

[210] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[211] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[212] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[213] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[214] C. Zhu, R. Tao, K. Luu, and M. Savvides, "Seeing small faces from robust anchor's perspective," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5127–5136, 2018.

[215] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[216]

[217] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li, "Detecting faces using region-based fully convolutional networks," *arXiv preprint arXiv:1709.05256*, 2017.

[218] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-stage dense face localisation in the wild," *arXiv preprint arXiv:1905.00641*, 2019.

[219] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "Cms-rcnn: contextual multi-scale region-based cnn for unconstrained face detection," in *Deep Learning for Biometrics*, pp. 57–79, Springer, 2017.

[220] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *european conference on computer vision*, pp. 354–370, Springer, 2016.

[221] J. Wang, J. Ding, H. Guo, W. Cheng, T. Pan, and W. Yang, "Mask obb: A semantic attention-based mask oriented bounding box representation for multi-category object detection in aerial images," *Remote Sensing*, vol. 11, no. 24, p. 2930, 2019.

[222] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, 2000.

[223] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[224] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," *arXiv preprint arXiv:1912.02424*, 2019.

[225] W. LB *et al.*, "A high resolution optical satellite image dataset for ship recognition and some new baselines," 2017.

[226] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai, "Rotation-sensitive regression for oriented scene text detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5909–5918, 2018.

Appendix A

List of Author's Publications

A.1 Journal publications

i. **Z. Fang**, J. Ren, S. Marshall, H. Zhao, Z. Wang, K. Huang, and B. Xiao, "Triple loss for hard face detection," *Neurocomputing, 2020.*

A.2 Journal / Conference publications under consideration

i. **Z. Fang**, J. Ren, S. Marshall, H. Zhao, S. Wang, and X. Li, "Optimizing the Convolutional Neural Network with Pre-Trained Weight Inheritance and Genetic Selection of Input Channels," submitted to *Pattern Recognition.*

ii. **Z. Fang**, J. Ren, S. Marshall, H. Sun, J. Han, "SAFDet: A Semi-anchor-free Detector for Effective Detection of Oriented Objects in Aerial Images," submitted to *European conference on computer vision.*