Towards the Design of Neural Network Framework for Object Recognition and Target Region Refining for Smart Transportation Systems

By

Yiheng Zhao

Thesis submitted in partial fulfillment of the requirements
For Master of Computer Science degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Object recognition systems have significant influences on modern life. Face, iris and finger point recognition applications are commonly applied for the security purposes; ASR (Automatic Speech Recognition) is commonly implemented on speech subtitle generation for various videos and audios, such as YouTube; HWR (Handwriting Recognition) systems are essential on the post office for cheque and postcode detection; ADAS (Advanced Driver Assistance System) are well applied to improve drivers', passages' and pedestrians' safety. Object recognition techniques are crucial and valuable for academia, commerce and industry.

Accuracy and efficiency are two important standards to evaluate the performance of recognition techniques. Accuracy includes how many objects can be indicated in real scene and how many of them can be correctly classified. Efficiency means speed for system training and sample testing. Traditional object detecting methods, such as HOG (Histogram of orientated Gradient) feature detector combining with SVM (Support Vector Machine) classifier, cannot compete with frameworks of neural networks in both efficiency and accuracy. Since neural network has better performance and potential for improvement, it is worth to gain insight into this field to design more advanced recognition systems.

In this thesis, we list and analyze sophisticated techniques and frameworks for object recognition. To understand the mathematical theory for network design, state-of-the-art networks in ILSVRC (ImageNET Large Scale Visual Recognition Challenge) are studied. Based on analysis and the concept of edge detectors, a simple CNN (Convolutional Neural Network) structure is designed as a trail to explore the possibility to utilize the network of high width and low depth for region proposal selection, object recognition and target region refining. We adopt Le-Net as the template, taking advantage of multi-kernels of GoogLe-Net.

We made experiments to test the performance of this simple structure of the vehicle and face through ImageNet dataset. The accuracy for the single object detection is 81% in average and for plural object detection is 73.5%. We refined networks through many aspects to reach the final accuracy 95% for single object detection and 89% for plural object detection.

Acknowledgements

Firstly, thanks to Professor Azzedine Boukerche, I gained a chance to pursue master's degree in University of Ottawa and did the research in Paradise Lab. I was also grateful for his kindness to provide me finical supporting. With funding, I can purchase necessary books, documents software and equipment for my thesis experiments. During the study life, he taught us important academic and social network skills. Besides, he led us visiting many famous companies to gain insight into industry field.

Secondly, I give my thank to University of Ottawa, who offers me the chances to learn deep techniques of computer science. It also provides me a great environment for studying and good academic resources in its library. I also enjoy the health welfare it brings to me.

Thirdly, I appreciate my colleagues in Paradise Lab. They always accompany with me and give me supports. Besides, I was inspired by them to write my thesis.

Finally, I appreciative to be born in such happy family. My parents give me gratuitous help, care and finical supports. They did everything for me to ensure that I have a great condition in my life. I'm proud of them and do my best to let them proud of me.

Table of Contents

Abstract……ii

Acknowledgements iii

Table of Contents iv

Nomenclature vi

1 Introduction 1

1.1 Background 1

1.2 Motivation and Contribution 4

1.3 Thesis Outline 6

2 Theories and Literature Review 8

2.1 Related Knowledge 8

2.1.1 Convolution. 8

2.1.2 Artificial Neural Network. 10

2.1.3 Over-fitting and Under-fitting Problem. 14

2.1.4 Techniques for Network Training. 16

2.1.5 Techniques for Plural Object Detection. 21

2.2 State-of-art CNNs and Frameworks 25

2.2.1 Le-Net. 25

2.2.2 Alex-Net. 27

2.2.3 ZF-Net. 30

2.2.4 VGG-Net. 32

2.2.5 GoogLe-Net. 33

2.2.6 Microsoft Res-Net. 34

2.2.7 Crafting GBD-Net. 36

2.2.8 Summary. 37

2.3 Frameworks for Plural Object Detection 38

2.3.1 R-CNN. 38

2.3.2 SPP Network. 39

2.3.3 Fast R-CNN. 41

2.3.4 Faster R-CNN. 42

2.3.5 Summary. 43

3 Research Design and Methodology 44

3.1 Analysis and Design 44

3.1.1 General Structure. 44

3.1.2 Conv Layer. 47

3.1.3 Pooling Layer. 49

3.1.4 Output Layer. 49

3.1.5 Training. 50

3.2 Summary 51

4 Experiments and Results 55

4.1 Functional Test 56

4.1.1 Single Object Recognition. 56

4.1.2 Plural Objects Recognition. 72

4.1.3 Reliability on Scale and Illumination Variance. 76

4.2 Improvements 79

4.2.1 Convolutional Operation. 79

4.2.2 Activation Functions. 81

4.2.3 Pooling Layer. 84

4.2.4 Plural Object Detection. 86

4.3 Summary 86

5 Conclusion and Future Work 88

5.1 Conclusion 88

5.2 Future Work 90

References 92

Nomenclature

Abbreviations

AED	Average Euclidean Distance
AFN	Average Face Label Number
AFT	Accuracy of Face samples Test
ANN	Artificial Neural Network
AVN	Average Vehicle Label Number
AVT	Accuracy of Vehicle Samples Test
BBR	Bounding-box Regression
BP	Back Propagation
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DFT	Discrete Fourier Transform
FPS	Frame Per Second
HOG	Histogram of Oriented Gradient
ILSVRC	ImageNET Large Scale Visual Recognition Challenge
IOU	Intersection Over Union
MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
NoI	Number of Indication
NoL	Number of correct Label
OCR	Optical Character Recognition
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RoF	label Range of Face
RoI	Region of Interest

RoV	label Range of Vehicle
SIFT	Scale-invariant Feature Transform
SPP	Spatial Pyramid Pooling
SURF	Speeded up Robust Features
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent

Mathematical Symbols

$\otimes$	Convolutional Operation
$sigmoid(.)$	Sigmoid Function
Max(A,B)	Max Operation, output the larger value from A and B
A	Arbitrary Matrix or Vector
Loss(.) /L	Arbitrary Loss Function
δ_p	Sensitive Map
σ	Learning Rate
In	Input Data
Out	Output Data
L_{cls}	Binary Log Loss Function
N_{reg}	Number of Anchor Location
b_o	Proposal Region Representation
f(.)	Arbitrary Activation Function

Chapter 1
Introduction

Although human's eyes have a very high recognition rate toward the natural world, it is unstable, exhausted and time consuming for elaborate, tedious or duplicate tasks, such as matching gene strands, criminal faces, detecting tumour, arteriosclerosis or other malign changes in a large database. Moreover, their physical and mental states dramatically affect the recognition rate. According to the Canadian government, 40% fatal driving accidents happen when drivers are not in a state (e.g. drinking alcohol) [94]. Why not let machines help us deal with those tasks? With appropriate techniques as well as equipment, machines work efficiently and stably all the time ignoring emotional influences. In this chapter, we give explanations of what we have learnt for computer vision, why did we gain insight into the field and what are the essential techniques and contributions of our project.

This chapter is separated into three sections: background introduction, thesis motivation and outline. In the first section, the basic concept and history of recognition technique development are introduced chronologically. Then, we give detailed explanation on why did we choose this field and what did we achieve utilizing current techniques and theories. The last section is a brief outline illustration of the rest content.

1.1 Background

Object recognition is not a new research topic. Related to computer vision and image processing, it is a leading-edge discipline aiming at classifying certain object categories for images or videos, which are applied to many visual assistant systems. The concept of OCR (Optical Character Recognition) [2] was first conceived in the 1870s to help blind people. During the period, many inventions sprang up like mushrooms, such as Fournier d'Albe's Optophone [3] and statistical machine [2]. In 1963, Roberts presented a novel algorithm to construct and display a 3D array of solid object from 2D photograph [1]. This led recognition techniques developing from simple

character recognition to a more complex pattern recognition system. During the next few decades, corner [4-7], edge [8, 9] and blob [10, 79] detection algorithms were proposed for low level image processing. Due to the simplicity of feature, those algorithms were only applied to image reinforcement and smoothing for a long period.

After 1990s, as machine learning techniques blooming, new models for recognition have gradually replaced conventional pattern based techniques. Initially, shallow learning approaches, such as logistic regression [11], Bayes classifier [12], decision trees [13], boosting [14], and SVM (Support Vector Machine) [15], replaced template based matching methods. Traditional image matching methods linearly match each input pixel with the target. However, this can only deal with objects with fix size, affine, contour and position. A slight shape deformation will cause matching inconsistency. On the contrary, machine learning classifiers are stable to those changes due to their non-linear demarcation. One advantage is that classifier parameters can be trained automatically using regression algorithms, such as OLSR (Least Squares Regression) [16], Linear Regression [17] or Logistic Regression [11]. Another advantage is the high accuracy rate for classification. Boundaries between different categories are non-linear that can clearly divide the ground truth of different categories. SVM [15] even solves the feature shortage problem by mathematically increasing data dimension with specific kernels. The proposal of machine learning began a new era for compute vision.

Although at that time, ANN (Artificial Neural Network) has already come up [18 - 21], it was much less popular than surface learning based frameworks due to the fact that the latter needs relatively few training samples and has a high processing speed. Traditional ANN, such as MLP (Multi-Layer Perceptron) network, has a very complex structure with full connections to all nodes. At that time, it was almost impossible to compute all parameters of a complex network within acceptable time cost and accuracy, because images as a 2-dimensional dataset, have abundant inputs, which generates millions or even billions of parameters for training. Besides, more samples are also necessary to ensure network coming to stable status. As a consequence, ANN was only introduced to small scale problems, such as data prediction and speech recognition in which field training samples are enough and inputs are few.

A decade later, as enlargement of datasets (e.g. ImageNet, LabelMe), concept of parameter sharing [23], and BP (Back Propagation) algorithm [22], deep learning techniques gradually took

place of surface learning. The first sophisticated network for handwriting recognition [23] was designed by Lecun who imported convolution concept into the neural network. Since then, more and more innovative structures [24-27] and strategies came up to fasten training and testing process as well as decrease error rate. In 2010, ImageNet held Large Scale Visual Recognition Competition aiming at facilitating development of recognition techniques. The host provided large scale image dataset from 1000 categories of different point of view, which enables complex network generation. More advanced structures, such as Alex-Net [24], GoogLe-Net [25] and Res-Net [26] have top-5 recognition rate beyond 90% in the competition.

Limited by the structure, classical networks cannot detect plural objects within one image. Since the increment demands of recognition in the real situation, a series frameworks based on CNN are designed to address the problem. As a pioneer, R-CNN [28] is proposed to segment images into region proposes which is assumed to contain only one object and recognize those region proposes one by one through neural network. Further techniques, such as SPP [29], fast R-CNN [30] and faster R-CNN [31] are all based on this concept.

Today, relying on the development of recognition techniques, many applications have been designed to improve people's lives combined with mobile network routing algorithms [105-110]. For instance, Victor Shaburov and Yurii Monastyrshin implemented facial recognition system [32] into Looksery for facial modification on the photo which was purchased by SnapChat. Research group of Facebook also designed DeepFace, a 9-layer neural network, for face recognition achieving 97% accuracy. Created by Australian Border Force, SmartGate [33] was introduced to Sydney Airport to promote faster and more secure international traveling which also applied facial and fingerprint recognition techniques for passport checking electronically. OCR techniques have widely utilized for document scanning. As for some top-500 global vehicle companies, such as Tesla, road detection is one of the most important foundations for autopilot systems.

In a predictable future, as development of AR (Augmented Reality) technique, image recognition techniques will occupy the most important position. It is my interest to have a deep study in this field and make achievements on AR devices to improve people's lives. This project is a trail to understand the basic theory of computer vision and to implement my ideas towards the recognition network.

1.2 Motivation and Contribution

Traditional vision algorithms (e.g. SIFT [6] and SURF [7]) and surface learning algorithms (e.g. SVM [15]) although are easy to apply in many detection systems, there are many restrictions according to environmental factors. Especially, variation of non-linear illumination, increment of unexpected noises, object rotation, deformation and movement will sharply reduce accuracy rate [35]. ILSVRC (ImageNet Large Scale Visual Recognition Competition) shows that CNNs have a lower error rate than traditional methods for object recognition.

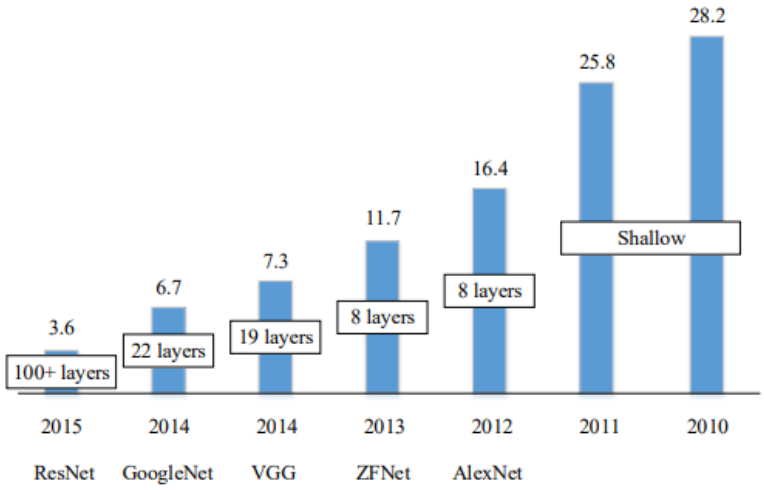


Figure 1.1: The ImageNet ILSVRC challenge results (top-5 error (%)) from 2010 to 2015 [84]

Figure 1.1 illustrates the tendency of best top-5 error rate of published recognition systems during 2010 and 2015. Y-axis is the percentage of error rate while X-axis shows published years and related network names. According to the statistic, the tendency of top-5 error falls down every year from nearly 28.2% to 3.6%. Between 2011 and 2012, error rate is sharply reduced by 9.4%, in which year CNN [24] was introduced into the competition. Bars of 2010 and 2011 indicate the results of shallow methods (e.g. HOG and SVM) based systems, while the rest are designed based on CNN. As the increment of network's depth, accuracy reached to the natural standard (96.4%) in 2015 by Res-Net [26].

From the bar chart (Figure 1.1), we conclude that CNN has better performance than the most classical recognition methods. Studying CNN is a good start for robust system development. Although there are many other neural network structures to achieve image recognition, CNN is the

most efficient one due to the weights sharing. That is the reason why only CNN is practical for high resolution image detection, rather than RNN (Recurrent Neural Network) [37-39] GAN (Generative Adversarial Network) [95] or DBN (Deep Belief Network) [36], and why we choose CNN for studying and researching.

However, current CNN structures mainly focus on accuracy and efficiency improvements of unique object recognition in small scale images (usually 224*224 according to training datasets). When it is directly applied to the real scene with multiple objects and complex background, the system can hardly indicate items, because:

- The sizes of real scene images are not constant. We cannot simply re-scale the inputs in proportion, because the deformation of image likely causes features deficiency. We cannot clip region with required size neither, because without pre-processing techniques, computer doesn't know which area should be recorded and which one should be deleted. Manually refining images is not possible for large scale dataset, such as ImageNet.
- Followed by classifying layer, full connection layers combine all high-level features together regardless of which object do those features belong. The result of each element of output vector is a non-linear weighted sum of all detected features that only represents category distribution, label number or likelihood of entire inputs.

As for image segmentation, although the final goal is to separate different objects on the same image, CNN is only applied to find possible features. Latter, D-CNN [44] structure is created to restore images from CNN outputs. In this structure, a de-convolutional network is inserted into the process which applies an opposite operation. It decodes feature map to restore original images. However, the efficiency is low, and accuracy cannot compete with combined techniques. Besides, the output of this network cannot distinct overlapped objects of the same category.

In terms of Classical frameworks of CNN, they need assist of image segmentation to achieve plural object detection. The idea is firstly cut the original image into various region proposals of the same size with image segmentation algorithms [40-42] in pre-processing step. Then, utilize CNN to recognize the proposals one by one. Commonly, post-processing algorithms [65] are also implemented to refine object box as well as remove redundant indications. Additional processing leads to exponential computational cost [30, 31]: region proposal and feature generation

are the two primary processes costing time (60%-80% of the total time cost). Ren et al. [31] announce that they replace pre-processing step by a simple neural network, while they still need to apply different networks in their framework. Currently, it is still a challenge to precisely and efficiently detect plural objects. To explore whether there exists a network that can directly recognize raw images of multiple objects without region proposal is a motivation for this thesis.

By studying and analyzing state-of-the-art networks and frameworks, a novel simple network structure is proposed in this thesis which achieves multi-object detection in a low error rate and time cost. Experiments are made to test the performance of this structure. By analyzing the result, improvements are designed and applied on this network. This thesis proves a simple structure existed for plural object detection. Although the proposed simple structure may not be the optimal, it gives a new idea to design network.

1.3 Thesis Outline

In Chapter 2, related knowledge, such as what is convolutional computation and what are the current problems existed in CNN, is illustrated in detail. Then the components of CNN are analyzed to give an understanding about what are the functions of different layers in the network. Following this explanation is an introduction about most advanced networks published between 2012 and 2017 to show how developers improve the current network to have a better performance. To solve plural object recognition problem, we also mention many classical frameworks, such as R-CNN. Brief analysis is appended to section 3 to make a comparison between different networks and frameworks. Advantages and disadvantages are also demonstrated in that section.

In Chapter 3, our network structure is proposed and explained in details which we get the inspire from analysis in chapter 2. The main idea is to have a single network but wider hidden layer without full connection layer.

In Chapter 4, firstly we implement various experiments based on the time error rate and cost for both training and test. Further improvements are made according to the experiments. Many techniques toward structure improvements and training efficiency improvements are considered and import into this network. Comparison is given with these two networks to show the performance of improvement.

The last chapter gives a summary about this thesis, evaluating this new CNN network structure by comparing it with most advanced R-CNN frameworks. The shortages and potential resolutions are proposed. Furthermore, we also list plans for future improvement and study orientation.

Chapter 2

Theories and Literature Review

Before explaining our network, it is better to deeply understand theories behind CNNs as well as how did other scholars make contributions to developments. We can be inspired by their work and avoid many mistakes by reading their analysis. This chapter is to give a view for the most useful knowledge, which is divided into three sections. In the first section, related theories and mathematical operations are stated to give general ideas about what is CNN, why it works for image recognition, and how to train networks. In the second part, masterpieces of CNN published on ILSVRC are demonstrated in details followed by a brief summary. In the last section, we introduced frameworks that achieves plural object detection based on CNN.

2.1 Related Knowledge

This section has four parts, focusing on explaining CNN structure, related mathematical methods, how to train networks and how to solve problems happened during training and testing process.

2.1.1 Convolution.

Classical ANNs [20, 21, 45, 46] have complicated connections between neurons to simulate nature brain. All neurons are fully connected between layers, whose links are represented by weights and bias. Those parameters are keys to describe the significance of one neuron affects to its corresponding neighbors. Since image is a 2-dimensional dataset determined by width and heights, normal ANNs for image recognition include abundant parameters. For example, a 100*100 image with 3-layer network structure has more than $3*9.3*10^{157}$ connections. It is impossible for the current computer to train all the connections with acceptable time consuming. Deriving from signal processing, convolutional operation is applied in neural network to reduce parameters bombing, which constructs CNN architecture.

Convolutional operation [47] combines two signals by the integral product of the two functions. In digital signal processing, the real signals can be regarded as convoluting many simple impulses with scaling and shifting. These impulses describe the most important information of real signals while their sizes are much smaller than the signal. So, they can be selected as features to represent the real signals. Reversely, convolutional operation can also be utilized for separating impulses by finding appropriate feature filter functions (also called kernel).

Similarly, images can be regarded as 2-dimensional compound discrete signals. Integral of convolutional operation in continuous signals can be transformed to sum operation for images. Considering efficiency, the best way to recognize object is on its feature maps, because features have very small dataset whereas contains most important properties without much redundant data. Besides, many feature spaces have the characteristic of rotation, scale and illumination invariance. To extract features from intensity map, convolutional operation is the most appropriate method. We assume that kernels are already found in this section. The process to find appropriate kernels will be introduced in network training section.

Convolutional operation is similar to filtering operation of edge detectors [8, 9] while kernels need to be flipped to 180 degrees, because of the characteristic of the signal combination on edge area. To make it specific, each element in the feature map is the accumulation of the weighted input within certain region. The weights are consisted of flipped impulse responses (see Figure 2.1). Starting at upper-left corner, kernel moves on the image from left to right, then one row down, then from left to right again until reaching to bottom-right corner.

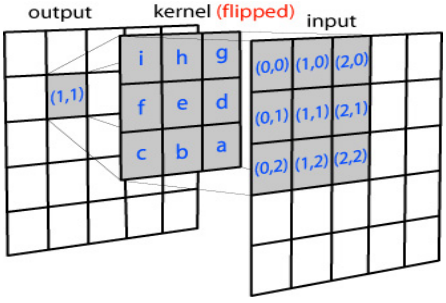


Figure 2.1: 2D Convolitional Operation [85]

$$Out[y,x] = \sum_j \sum_i In[i,j] * h[y-i, x-j] \tag{2.1}$$

When dealing with edges, the missing pixels causes inconsistency in size between input and output maps. Zero-padding is applied to solve this problem. To keep the size consistency,

original image is extended by zero matrixes, whose width depends on kernel size. In most CNN networks, the size of kernel is odd (1*1, 3*3, 5*5, …) to ensure that kernel can be set on the center of pixel.

Although parameters are sharply reduced by sharing weights in the same layer, convolutional computation still has complex of $O(I_w*I_h*W_w*W_h)$. For example, an 800*600 image with 4 types of 5*5 kernels costs $4.7*10^7$ computation for multiplication. To improve computing process, convolution theorem can be utilized. It indicated that the convolutional computation of two signals in spatial domain can be converted to mathematical multiplication in Fourier space. So, we can firstly transform original image and kernels from special domain to frequency domain and make multiplication for one time. Then, transform the result back to the spatial domain. This method has linear growth as kernel size enlarging. This thesis takes advantage of the principle to accelerate the operating process.

2.1.2 Artificial Neural Network.

Inspired by biological neural network, ANN has been proposed for decades. Belonging to deep machine learning (with multiple hidden layers), its goal is to generate a classifier or predictor automatically through mathematical methods. To achieve the goal, most neural networks are consisted of neuromas nodes, which are also named "neurons" or "cells". Each of them has specific functions or structures with untrained parameters. All of them are connected following certain principles. Training process is to adjust parameters iteratively until values of loss function converging. Constituted by input layer, hidden layers and output layer, the network is deemed as a black box that can be applied to complex tasks.

Commonly designed for predicting and recognizing, four of the most famous ANN models are RNN [37, 38, 39], DBN [36], CNN [24, 25, 26, 27], and GAN [95]. In a simple RNN, cells are linearly connected while each of them has a complex structure for data transformation (see Figure 2.2). Similar to FA (Finite Automata) machine, inputs of each cell consist of data vectors and previous output status. This property allows network to predict the tendency of dataset from history. Main problem for this network is to find an appropriate cell structure to prevent "forgetting" problem. Cells only have short-term memory that the influence of past inputs reduced quickly as progress continuing. In a very long input sequence, the original data have no relation to the current

prediction, leading to huge prediction deviation. This phenomenon also causes bottleneck of CNN as depth of structure increasing. Methods to solve this problem are introduced in section 3. On the other size, DBN is constructed layer by layer. Each layer is also a sub network called Restricted Boltzmann Machine (see Figure 2.3). Training this network is to circularly re-input outputs until all outputs become stable. This network is adept at image restoring and parameter initialization for complex network.

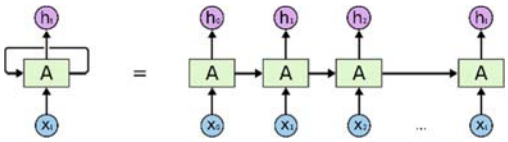


Figure 2.2: A Simple RNN Structure [86]

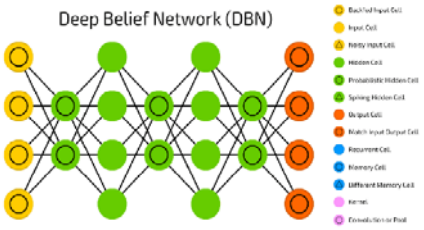


Figure 2.3: Structure of DBN [87]

This thesis mainly focuses on CNN for object recognition, because it has the property of stationarity of statistics and locality of pixel dependencies compared to other neural networks. Compared to standard feedforward neural networks [48, 49], CNNs have fewer connections (sharing parameters) and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse. GAN inherits CNN's concept while inserting a de-convolution structure for image imitating. As it is similar to CNN, we primarily focus on CNN structure.

CNN is a feed-forward multi-layer architecture. Training process is to minimize the value through loss function rather than to stabilize output vectors, like DBN. A complete CNN is combined with many layers: convolution layers (including activation function layer), subsampling layers, fully connected layers, and classification layers.

When dealing with high-dimensional images (e.g. 600*800), it is impractical to fully connect neurons to next layer. For instance, a 600*800 input has $(600*800)^2$ connections to the following layer. As the core of CNN, the main purpose of convolutional layer (i.e. conv layer) is to reduce links by local connection. Instead of connecting neurons to all elements in the next layer, neurons only connect to the ones within local region, which is called receptive field. For example, a neuron of first layer only connects to 5*5 field neurons of the second layer in corresponding

position. Then a 600*800 input have 600*800*5*5 connections which is 1/19200 of full connections.

It is still impossible to adjust all the parameters if each connection has unique weight. Compared to other neural networks, the significant improvement of CNN is weights sharing with convolutional operation. As the concept of CNN, neurons only connect to receptive field of next layer. To share parameters, all receptive fields use the same weight. For instance, previous 600*800*5*5 connects shares only 5*5 parameters (similar to edge operators [8, 9]). This property allows different adjacent input locations connect to the other adjacent locations. Macroscopically, the process of convolutional layer can be considered as feature extraction and abstraction. Features are separated as low level, middle level and high level depended by the depth of the network. Their representation starts from point to corner to edge, and more abstract to describe a global property of objects.

To keep diversified representation for object distinction, in one conv layer, more than one kernels are applied with different parameters, which generates "channels". Values of kernels are weights to detect different type of local features adjusted by machine learning. For example, the first layer of Le-Net [50] only has 6 channels with different 5*5 kernels. In total, there are only 150 weights and 12 biases compared to 32,768 weights and 1024 biases for a regular network [46]. The map of current layer is the linear combination of different channels in specific sequence (see [50]). To ensure the training process generates different kernels, initialization of each kernel should be distributed random.

At the end of conv layer, various activation functions are available to solve non-linear classification and over-fitting problems. Activation function decides whether a neuron should make contributions to the output by reflecting inputs within a certain range. It also enables a more principled for parameter adjustment. Popular activation function includes ReLU (Rectified Linear Unit) [51], Sigmoid and Tanh (Hyperbolic Tangent) functions. The selection of activation function depends on the task that network will solve. Usually, ReLU is the most popular for image recognition due to sparse activation, better gradient propagation, scale-invariance and efficient computation [51]. Furthermore, ReLU has a steeper profile that leads to faster learnability. There are also some other versions of ReLU, such as leaky ReLU and Noisy ReLU to address different

problems. In chapter 4, the performance of standard ReLU, Sigmoid, identity functions are compared in terms of accuracy and training speed.

Feature maps of conv layers are compressed by pooling layers (also called subsampling layers) to reduce the spatial size and data entropy. More importantly, it allows to extract invariances from features for property description. Theoretically, it is possible to construct a network without pooling layer. However, even one feature map of 96*96 image with 7*7 kernel has 9216 feature points with 451,584 multiple operations. In the real network, such as Res-Net or GoogLe-Net, conv layer channels can be as many as 256. Then the total computation for one convolution layer rises to 115,605,504. Besides, sophisticated networks have more than 50 conv layers. Processing costs lots of time for training and practical application. Subsampling layers are designed to address this problem. In some work [93], pooling layers can also be replaced by a strided convolution layer (the stride of kernel is *n* rather than 1).

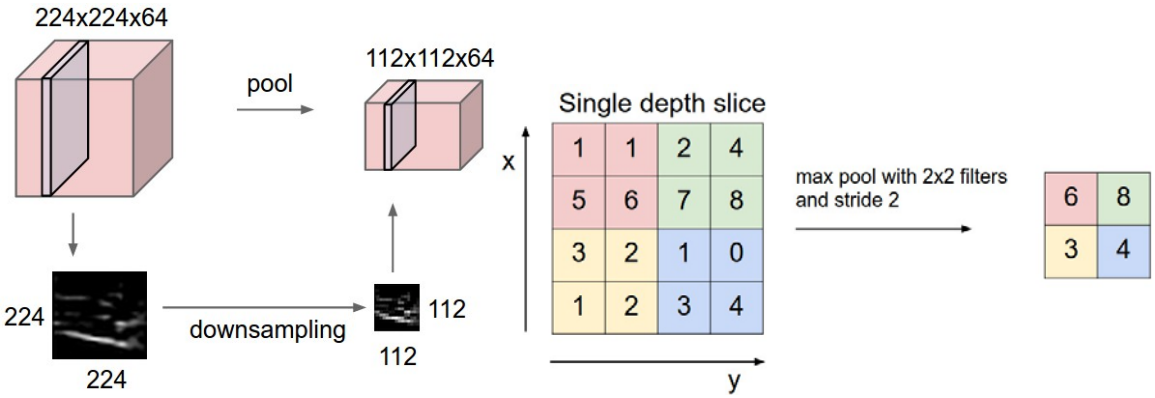


Figure 2.4: Process of Max-pooling Layer [88]

Pooling layer always selects salient features and remove irrelevant data to fasten the speed (see Figure 2.4). The most common settings for a pooling layer is a 2*2 filter with stride of 2. The output of pooling layer is 1/4 the size of inputs which sharply reduce computation cost. Mean pooling and max pooling are two common subsampling approaches. For mean pooling, the output feature map of each point is the average value of each 2*2 filter whereas for max pooling, the point is the max value of correspond region. Notice that there are only two strategies for max pooling: 1) 3*3 filter with stride of 2, which is called overlapping pooling; 2) 2*2 filter with a stride of 2. A large filter is destructive to the result. Conv layers and subsampling layers are interweaved and repeated for many times until feature map can precisely represent the classes. In practical, max pooling has better performance than mean pooling.

Different from Conv Layer, in FC (fully connected) layers, each neuron connects to all neurons in the next layer, which is the same as MLP (Multilayer Perceptron) Network [52]. Local features are combined with weights and bias to form global features which are most appropriate to describe entire image. It is worth to note that conv layers with 1*1 kernels is easily confused with FC layers. Even with 1*1 kernel, conv layer shares this parameter to all neurons while in FC layer each neuron relies on unique parameter. For example, a 32*32 input of conv layer for one channel needs 1 parameter representing weight and 1 for bias while in FC layer, 32*32 weights are necessary as well as 1 bias.

Global features are put into the classification (output) layer for the final recognition. Classification layer can be simple classifiers, such as Softmax regression [43], RBF (Euclidean Radial Basis Function) [53] or logistic function [11]. Much complex classifiers, such as SVM (Support Vector Machine) [15], are also available to increase accuracy. Softmax regression is a simple but most popular classifier that normalizes outputs of network and computes gradient based on the multinomial distribution (see Formula 2.2). Different from logistic whose focus is on two-class problem, Softmax can classify multiple categories by outputting k categories' possibilities. All outputs are reflected between 0 and 1, whose sum is 1.

$$D_c = |y_c - \frac{e^{z_c}}{\sum_{d=1}^{N} e^{z_d}}| \tag{2.2}$$

while y_c is target probability of class c, z_c is the input of classifier, N is the total number of inputs. RBF (radial basis function) is another loss function for network outputs (see formula 2.3). Usually, it is defined as the Euclidean metric between inputs and the center of class, represented by:

$$D_c = e^{-\frac{||x-x'||_2^2}{2\sigma^2}} \tag{2.3}$$

Usually, a loss function is also appended with penalty term, such as L_1, L_2 regularizations, to restrict weights' variation.

2.1.3 Over-fitting and Under-fitting Problem.

In statistics, over-fitting is a modeling error occurring when predictions of samples fit too closely or exactly to training data while fail to fit testing data (see Figure 2.5). It is one of the most common and serious problems when training a neural network, which is reflected in the high variance after

training. The cause of this problem is multifarious. Shortage of training samples is the primary reason, which also stagnated neural network development for decades. Samples don't contain enough features of the category to adjust parameters. It is similar to human's learning process: to a newborn, we teach him from a picture that amaryllis belongs to flower and he understands. Without any other information, he may not classify painted daisy as flower because it has the different color and shape.

The simplest solution is to enlarge sample groups as many as possible to include all class features. However, we cannot ensure all cases are collected, because even the same object has billions of 2-D image reflections according to the illuminations, point of view and distance. Collecting enormous dataset need lots of efforts that cannot be accomplished immediately. Besides, large group training samples will exponentially increase the learning process. Another way is to simplify the network structure (reduce parameter number) by dropout algorithm [54]. The network also should meet with Occam's Razor principle [55].

When loss function (function to judge the suitability of network) has an over-large difference towards target, over-fitting also happens. In this case, a regularization term is added into the formula:

$$\min_{f} \sum_{i}^{n} V(f(x_i), y_i) + \gamma R(f) \tag{2.4}$$

$V(,)$ is an underlying loss function to evaluate similarity between output function $f(x)$ and target y. γ is a weight to control the importance of regularization term. $R(.)$ is the regularization term, typically chosen to impose a penalty on the complexity. Besides, noises included in the training samples also cause the negative influence for training, which can be solved by noise filters. Other methods, such as Bayesian Priors [56], pruning and model comparison, are also applied to reduce the influence of over-fitting.

Under-fitting problem happens when networks cannot capture the trend of training samples (i.e. after training network doesn't have classifying ability). Reflected in a high bias, this problem always happens when the structure of network is too simple. Linear structure of network always meets with this problem, because, dataset always has non-linear distribution. It is impossible to use a single linear classifier to divide complex non-linear dataset (see Figure 2.5). In networks,

activation function is applied to transform linear structure into non-linear one in order to better fit with dataset.

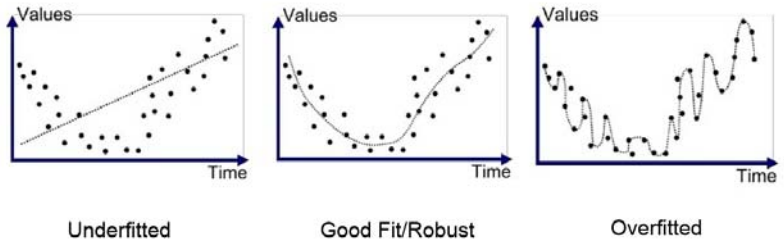


Figure 2.5: Charts of Under-fitting, Regular, and Over-fitting Classification [89]

In other machine learning methods, such as SVM, input vector is reflected to a higher dimension space to enlarge the description ability. Analogously, simple network can be complicated by additional channels, and conv layers. For example, VGG Net [27] has at most 512 channels in one conv layer while Res-Net [26] has 24 layers. During the training process, decreasing learning rate is another method to reduce under-fitting problem. Learning rate controls range ability of parameter gradients. A large learning rate can speed up convergence process while small rate can accurately find global optimal resolution. When a network cannot describe the property of training dataset, it is likely that optimal resolution of parameters has not been reached. For this case, enlarging training iterations is an efficient solution.

2.1.4 Techniques for Network Training.

Depending on the learning process, neural networks can be divided into supervised learning and unsupervised learning networks [57]. The difference is whether training need handcraft guidance. For unsupervised learning, such as DBN [36] and k-means cluster [58], it used to draw inferences from datasets consisting of input data without labeled responses. The method always applied on data that is hard for manually classification. This learning type is most convenient while human cannot control labels. On the contrary, supervised learning networks need a specific guidance for training. It need to compute distance between output and ground truth in order to modifying its parameters. Most CNNs use supervised learning methods due to its structure.

2.1.4.1. Back propagation.

Most popular CNNs, such as Google-Net [25] and Res-Net [26], commonly have millions of parameters. It is an important topic to find an efficient approach for parameters adjusting. Besides some functional restricted learning methods, such as Adagrad, momentum and Newton's method, Back Propagation [59] is one of the most efficient methods adopted in CNNs. The main concept is to compute the derivative contribution for each parameter layers by layers through gradient descent optimization algorithms. Gradient indicates orientation and how much does the error affect parameters. Errors can be computed following different standards, such as Softmax, RBF or L1/L2 norms.

BP algorithm has two phases: forward phase and backward phase. In the first phase, training data goes through the network with initialized parameters. The differences between outputs and targets are computed through certain loss function. In the second phase, values from loss functions are re-input into network from output layer to input layer to calculate the gradient of each parameter following chain rule. In each layer, the gradient is used for adjusting weights and bias. These two phases are repeated until the value of loss function converge.

Assume L(·) and f(·) are arbitrary loss and activation functions; $\mathbf{u}^n$ is convolution function of layer n, equaling to $\mathbf{u}^n = \mathbf{W}^n \boldsymbol{x}^{n-1} + b^n$ while x is the output vector of previous layer, $\mathbf{W}^n$ and b^n are the weight vector and bias for layer n. Then, the gradient of bias for last layer b^N can be represented through chain rule by:

$$\frac{\partial L}{\partial b^N} = \frac{\partial L}{\partial f} * \frac{\partial f}{\partial u^N} * \frac{\partial u^N}{\partial b^N} = \delta^N = f' * L', where\ \frac{\partial u^N}{\partial b^N} = 1 \tag{2.5}$$

δ^N is named as sensitive of layer N. Sensitives are the keys to compute gradient of weights and bias. The sensitive of each layer can be deduced from previous sensitives as below:

$$\delta^n = (w^{n+1})^T \delta^{n+1} f'(u^n) \tag{2.6}$$

Similar as gradient of bias, the gradient of weights can be represented by:

$$\frac{\partial L}{\partial w^n} = \frac{\partial L}{\partial f} * \frac{\partial f}{\partial u^n} * \frac{\partial u^n}{\partial w^n} = \delta^n * \frac{\partial u^n}{\partial w^n} = x^{n-1}(\delta^n)^T \tag{2.7}$$

$\Delta W^n = -\varepsilon \frac{\partial L}{\partial w^n}$, Where ε is learning rate to control increment pace.

When it comes to pooling layers, the computation of sensitive depends on the type of the layer. For average pooling layer, the sensitive of layer n is the up-sampling operation of layer n+1 divided by 4. As for max pooling layers, the sensitive of layer n is the up-sampling operation of layer n+1 with max value in corresponding position and zero in other positions. In this situation, position of max value has to be store in advanced for up-sampling operation.

The initialization of parameters also has significant influence on training convergence. Take sigmoid function as an example: when the absolution of input w increases, the output tends to be smooth while the derivative of this function tends to be 0, which means the step for convergence is gear down. The ideal situation is to enlarge derivative to have quick convergence. That is the reason initialization of parameters is important. A simple way to initialize parameter is to randomly select values in a certain range $(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}})$, while d is the number of neurons in the network. Another initialization is based on [60], which used $(-\sqrt{\frac{6}{H_i+H_{i+1}}}, \sqrt{\frac{6}{H_i+H_{i+1}}})$, while H_i and H_{i+1} are the sizes of the layers before and after the weight matrix. In other networks, Gaussian distribution are applied for randomly generate initializations.

2.1.4.2. Mini-batch gradient descent learning.

Looking back to the five or ten years' CNN history, all the masterpieces are based on the large scale datasets. Then computational explosion comes to the reality. Assume the size of training dataset is 1,000,000 which is very common for a complex CNN training process such as GoogLe-Net. The size of each data is 224*224 (most CNNs take advantage of ImageNet datasets which restricts images to 224*224). Training process utilizes the simplest CNN structure Le-Net. For each data, more than $2.1*10^5$ multiple operations will be made for one iterate convolutional computation which cause $2.1*10^{11}$ computations for the entire dataset. Usually 10^6 iterations are enough for training a network. Then it will cost $2.1*10^{17}$ computations for complete network training. If computation ability for the computer is 5 MIPS, then it will cost $1.3*10^4$ years for training. The time cost is unacceptable. So, different training principles are designed to minimize time cost. Combined with batch and stochastic training [61], which are two extreme strategies for assigning samples, mini-batch [83] strategy is the most convenient and popular strategy for CNN training.

Dataset is separated into certain groups named batches (usually a batch contains 100 image samples). The increment of batch size can fasten the process of training while to achieve the same accuracy as a small number of batch size, more epochs are needed because the gradient of each parameter is largely counteract. Training process is to train these groups one by one. For one batch training, all members in that batch go through the training process until convergence. Then start training with another batch until all batches are trained.

Algorithm 1: Mini-batch [90]

```
loop max epochs times
    loop until all data items used
        for-each batch of items
            compute a gradient for each weight and bias
            accumulate gradient
        end-batch
        use accumulated gradients to update each weight and bias
    end-loop all item
end-loop
```

One improvement is to compute the average error map during forward training for one batch. In the back-word process, only the mean error map is applied to adjust parameters. This will sharply reduce computational cost in back process as well as keeping accuracy. All groups are trained in the same progress. With mini-batch strategy, we can have a reasonably high learning rate, keeping the same number of steps.

2.1.4.3. Dropout.

As has been introduced before, one cause of over-fitting is the lack of training samples for complex network. Sometimes it is impossible to enlarge sample groups because the large scale dataset need lots of efforts to collect while equipment may be limited. Dropout [62] is a common technique dealing with the over-fitting problem by randomly simplifying full connection layers of CNN in each epoch during the training process. [27] has proved that increment of network depth will dramatically reduce the detection error rate. However, with the neural network become more complex, sample increment is inevitable which also costs exponential computation for parameter

adjusting. In order to solve this problem with a small sample group and complex network structure, “dropout” technique is proposed.

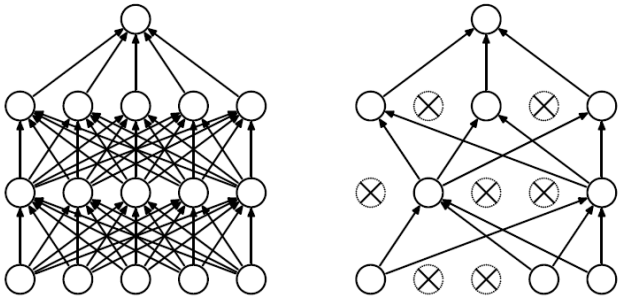


Figure 2.6: FC Network Layer Before and After Dropout [62]

“Dropout” refers to dropping out units in a neural network. The basic idea is to temporarily block random neurons between layers in each training iteration (see Figure 2.6). The probability of neurons restraining can obey some distributions, such as Normal or Bernoulli distributions, or as simple as a constant decimal. It can be understood as randomly generating different simpler sub-network structure in each training iteration by removing stochastic links from original layers. These simple networks all share the parameters, so the total number of parameters for the entire training process is still $O(n^2)$ while usually, for each training process, the number of parameters sharply reduced according to distribution. In many CNNs, “dropout” is always applied to full connected layer with the probability of 0.5.

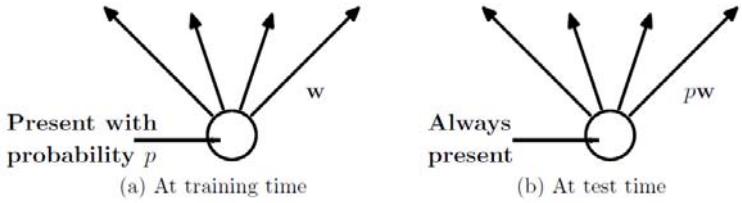


Figure 2.7: Weight Assigning During Training and Test Process [62]

In test process, no connection is blocked. It is not feasible to explicitly average the predictions from exponential thinned models because computing all those sub-networks (for probability equals to 0.5, 2^n different sub-networks can be generated) will cost lots of redundant computation. So all weights should multiple probability that gained in training step to ensure that testing output is close enough for expected training output (see Figure 2.7). Generally, dropout is

similar to create a random forest [63]. In the training process, different "trees" are generated through dropout. In test process, those trees are combined with weight (i.e. probability) to output stable result.

2.1.5. Techniques for Plural Object Detection.

Conventional CNN frameworks for plural object detection meet with efficiency and accuracy problems. In this section, we introduce selective search [64] scheme for efficiency improvement and bounding box regression [65] for accuracy improvement.

2.1.5.1. Selective search.

Before being detected through CNN, objects have to be localized to get the approximate positions because CNN only allows one object recognition at a time. Traditionally, exhaustive search is applicable for region segmentation. A window is chosen to scan the entire image from left-up corner to right-bottom corner. Each scanning step is regarded as a proposed region. To solve scale and resolution inconformity problem, windows of different size are also implied. For a 800*600 image with 224*224 window of 3 different scales, in total (800 - 224) * (600 - 224) * 3 = 649,728 proposals are generated for CNN recognition regarded less of rotation. The number is too large that most regions are redundant and are useless for object distinction.

To filter invalid region proposals, selective search [64] algorithm is designed. The basic principle is to merge nearby pixels step by step which is similar to MSER algorithm.

Algorithm 2: Hierarchical Grouping Algorithm [64]

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r1, \cdots, rn\}$
Initialise similarity set $S = \phi$
foreach neighbouring region pair (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while S $\neq \phi$ **do**

Get highest similarity $s(r_i, r_j) = \max(S)$

Merge corresponding regions $r_t = r_i \cup r_j$

Remove similarities regarding r_i: $S = S \setminus s(r_i, r_*)$

Remove similarities regarding r_j: $S = S \setminus s(r_*, r_j)$

Calculate similarity set S_t between r_t and its neighbours

$S = S \cup S_t$

$R = R \cup r_t$

Extract object location boxes L from all regions in R

The algorithm considers different color space (RGB, Grey, LAB, RGL, HSV, C, H, normalized RGB) to ensure the result is stable to noises and illumination. Furthermore, to have an accurate region proposal, Uijlings et al. define similarity of the pixel region with the combination of four factors: color, texture, size and fit.

Color: intensity values are normalized by 25 bins to generate a color histogram. The color similarity of two regions is represented by the histogram intersection:

$$S_{color} = \sum min(c_i^k, c_j^k) \tag{2.8}$$

while the histogram of new region can be calculated by:

$$c_t = \frac{size(r_i) * c_i + size(r_j) * c_j}{size(r_i) + size(r_j)} \tag{2.9}$$

Texture: SIFT-like [6] measurements is applied. The Gaussian gradient orientation is evenly divided into 8 sections. For each section, value of histogram is generated with 10 bins. Same as color similarity, histogram intersection of texture is also applied in this case to compute texture similarity.

Size: it refers to the number of pixels in each region. The main purpose is to preferentially combine small regions.

$$S_{size} = 1 - \frac{size(r_i) + size(r_j)}{size(img)} \tag{2.10}$$

while $size(img)$ is the pixel number of the entire image.

Fit: measures how well two regions are fitted with each other:

$$Fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(img)} \quad (2.11)$$

while $size(BB_{ij})$ denotes the tight bounding box around r_i, r_j.

The final similarity of the two proposals is a weighted sum of these four standards. Weights are adjusted during the training process. Proposals with high similarity are regarded as redundant regions and removed from proposal list.

2.1.5.2 Bounding-box regression.

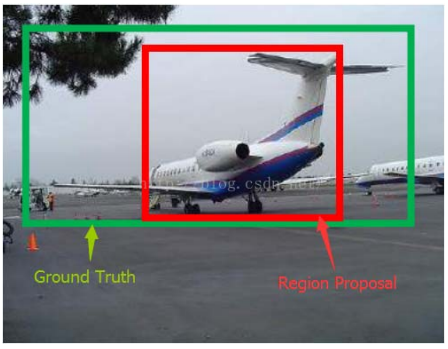


Figure 2.8: Plane Detection with Region Proposal and Ground Truth [65]

Even after selective search algorithm, many proposals are segments of the object that cannot indicate the location of the entire object. For example, (Figure 2.8) the red window is the region proposal generated by selective search and the green one is the ground truth. From CNN, both windows can be correctly labeled as plane while the red window only indicates particle plane (e.g. the wing of plane is missing). In the real situation, it is necessary to indicate the entire object. To solve this problem, BBR (Bounding-box Regression) [65] is designed to adjust region proposals getting close to ground truth.

In this algorithm, a region proposal is represented by a four dimensional vector *R*(*x, y, w, h*), whose elements refers to the center position (*x, y*), width and height separately. The idea is to find a reflection function f that makes the recognized proposals close to ground truth (i.e. $f(R_P) \approx R_{GT}$) through training. In detail, assume *P* is a proposal region. $d_x(P)$, $d_y(P)$ are the scale-invariant

translations of the coordinates (*x, y*). d_w(P) and $d_h(P)$ are the log-space translations of width and height. For *x, y, w* and *h*, translation formulas are shown as below:

$$G_x = P_w * d_x(P) + P_x \tag{2.12}$$

$$G_y = P_h * d_y(P) + P_y \tag{2.13}$$

$$G_w = P_w * e^{dw(P)} \tag{2.14}$$

$$G_h = P_h * e^{dh(P)} \tag{2.15}$$

The next step is to decide transformations: $d_x(P)$, $d_y(P)$, $d_w(P)$ and $d_h(P)$ from training. Because this reflection is linear, the proposals should be close enough to ground truth in the training step. The model cannot well match non-linear transformation. In CNN, when region proposal and ground truth are similar (i.e. IOU>0.6), d_* (* belongs to {*x, y, w, h*}) can be linearly represented by feature map dataset of the last layer (i.e. $d_*(P) = w_*^T * v(P)$, while w^T are coefficients learned by ridge regression and *v(P)* is the feature vector of proposal *P*. The problem can be converted to regression problem: find coefficient vector w that

$$w_* = arg \min_{w_*} \sum (t_*^i - d_*(P))^2 + \lambda ||w_*||^2 \tag{2.16}$$

λ is the regularization constant. Back Propagation [22] or Newton's method are available for w adjusting. Notice that ground truth transformations t_*^i is not the exact value of *G*(*x, y, w, h*), because in loss function, the input is the transformations of region proposal, not the real *P*(*x, y, w, h*). t_*^i are designed according to the relationship between *G* and *P*:

$$t_x = \frac{G_x - P_x}{P_w} \tag{2.16}$$

$$t_y = \frac{G_y - P_y}{P_h} \tag{2.17}$$

$$t_w = log\left(\frac{G_w}{P_w}\right) \tag{2.18}$$

$$t_h = log(\frac{G_h}{P_h}) \tag{2.19}$$

2.2. State-of-art CNNs and Frameworks

Networks introduced in this section are proposed on ImageNet Competition from 2012 to 2017. The understanding of those networks gives much deeper insight into the mechanism of neural network and provides inspiration to create new efficiency networks. Many useful techniques designed and applied are necessary for this thesis and our system.

2.2.1. Le-Net.

This network [23] is the pioneer proposed in 1989 by Lecun. The basic principle is: instead of manually assigned parameters, rely on machine learning to adjust neurons of convolutional operators. The goal is to minimize parameters (weights and bias) without overly decreasing recognition accuracy. The structure is utilized for number handwriting and machine-printed character recognition. The network was further improved in 1998. This section introduces the second version of Le-Net.

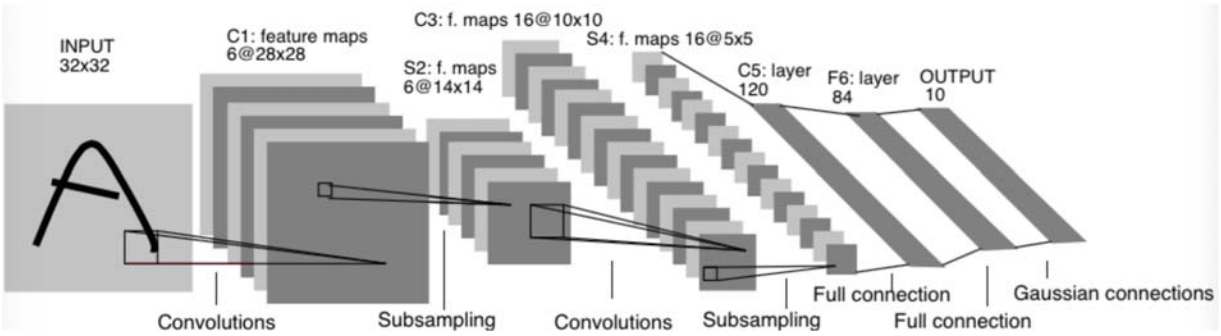


Figure 2.9: Structure of Le-Net (1998 version) [23]

The network has seven layers in total (see Figure 2.9): two convolutional layers, two subsampling layers, two full connection layers and one classifier layer. The input is a 32*32 grey scale image with single 20*20 number in the center. The first layer has six channels of 28*28, each of which is generated by different 5*5 kernels with the formula:

$$\boldsymbol{x}_i^n = f((\sum \boldsymbol{x}_i^{n-1} * \boldsymbol{k}_i^n) + b_i^n) \tag{2.20}$$

while $\boldsymbol{x}_i^n$ refers to i_{th} feature map of layer n; $\boldsymbol{k}_i^n$ refers to i_{th} kernel of layer n; b_i^n is i_{th} bias of layer n. $f(.)$ is the Tanh activation function. C1 contains 156 trainable parameters and 122,304 connections. For S2, mean-pooling strategy with 2*2 region of stride of 2 is applied. Feature maps

through this layer are compressed to 14*14. Connections between S2 layer and C3 layer are not fully connected. For each C3 channel (from channel 0 to channel 15), it is the combination of channels of S2 (from channel 0 to channel 6) with X. The connection is shown in Table 2.1.

Table 2.1: Connection Relationships between S2 and C3 [23]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Column represents the channel number of S2 whereas rows represents the number of C3. "X" means channels of corresponding S2 and C3 positions are connected. For example, the first column of the table indicates that channel 0 of C3 connects to channel 0, 1, 2 of S2. Last column means channel 15 of C3 connects to all channels in S2. Notice that the combination arrangement of S2 has 64 cases, while Le-Net only adopts 1/4 of them. Reasons are as below:

- Numbers of connections need to be bounded through non-complete connection scheme to reduce computation cost. The complete connection scheme contains 720 links (A_6^6) whereas incomplete scheme only contains 60 links which is 1/12 of former scheme. As a consequence, incomplete scheme has much higher training efficiency.

- Symmetry of the network need to be broken through non-symmetry scheme. Symmetry structure is responsible for information bottleneck, zero training error, gradient variance explosion and shattered gradients [66]. Breaking symmetry dramatically reduces redundant weights, leading to a better generalization.

C3 is followed by a mean-pooling layer (S4) with the same settings of S2. To be noticed, full connection of C5 is not the same as F6. Actually, C5 is a convolution layer generated by S4 with 1*1 kernels while F6 is a real full connection layer same as MLP [52]. The value of each position in C5 is the sum of all S4 channels with different weights and bias. The final layer is composed of RBF (Radial Basis Function) [67] to compute the Euclidean distance between the

feature map and parameter vector (one for each class with 84 inputs each). The output is represented as below:

$$y_i = \sum_j (x_j - w_{ij})^2 \tag{2.21}$$

The output can be interpreted as a penalty term measuring the fitness between feature map and a model of a category. The reason to apply RBF is that in probabilistic term, the output of RBF can be regarded as non-normalized negative log-likelihood of a Gaussian distribution which can represent the probability of categories [50].

To ensure the configuration of F6 is close enough to corresponding desired class, an appropriate loss function is necessary, which is the combination of MLE (Maximum Likelihood Estimation) criterion and penalties of incorrect classes:

$$L(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_n(Z^n, \boldsymbol{W}) + \log(e^{-j} + \sum_i e^{-y_i(Z^n, \boldsymbol{w})})) \tag{2.22}$$

while $y_n(Z^n, \boldsymbol{W})$ represents the output of correct class with the input pattern Z^n and parameter vector $\boldsymbol{W}$. N is the total number of categories. j is a constant positive value prevents the penalties of class with large value from being pushed further up. The function $\sum_i e^{-y_i(Z^n, \boldsymbol{w})}$ is the discriminative term to prevent "collapsing effect" [50]. After defining network and relative formula, backpropagation algorithm is applied to adjust parameters for 20 iterations. Lecun adopted NIST dataset for training and testing. Learning rate variation during training follows table 2. Error rate for testing samples is 0.95%.

Table 2.1.2: Variance of Learning Rate According to Iterations

Iterations	1	2-3	4-6	7-9	10-13	14-20
Reduced value	0	0.0005	0.0002	0.0001	0.00005	0.00001

2.2.2. Alex-Net.

Krizhevsky and his group designed Alex-Net [24] system receiving only 15.3% error rate for top-5 test and 26.2% for top-1 test of 1000-category recognition in 2012 LSVRC. This network is more complex than LeNet-5, containing 60 million parameters and 650,000 neurons. Alex-Net is not restricted to gray-space handwriting image of 32*32. More categories and larger images with RGB color space are available for the network.

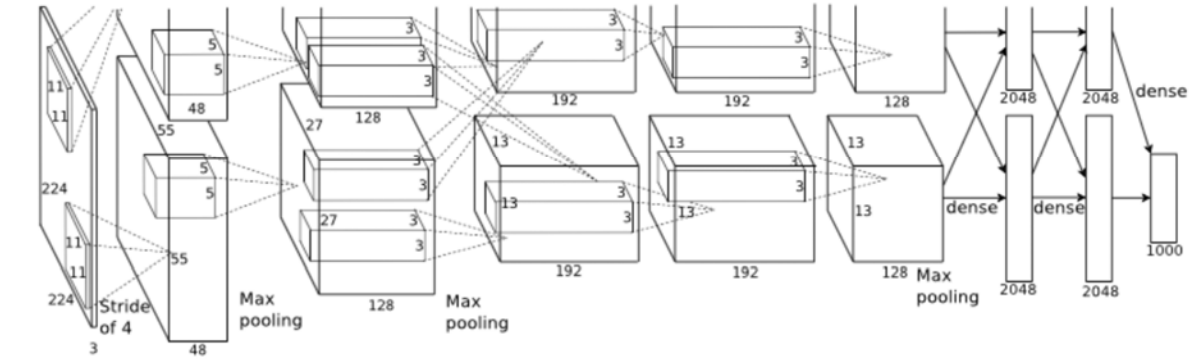


Figure 2.10: Structure of Alex-Net [24]

Compared to Le-Net, Krizhevsky et al. modified structure by inserting other three convolution layers and two full connection layers. In each layer, more channels are extended to select multifarious types of features because the sample size (224*224) is 7 times more than that used for Le-Net (32*32). Max pooling layers only appear after first, second and fifth conv layers. The activation function of each convolution layer is ReLU, because non-saturating nonlinearity function has a fast convergence of learning process compared to other activation functions [51]. No normalization is necessary for input. They utilize Softmax classifier [43] instead of RBF [53] to evaluate the likelihood distribution of categories.

Response-normalization layers are applied after first and second layers. The activity $b_{x,y}^{i}$ is given as below:

$$b_{x,y}^{i} = a_{x,y}^{i} / (k + \alpha \sum_{max(0,i-n/2)}^{min(N-1,i+n/2)} {a_{x,y}^{i}}^{2})^{\beta} \tag{2.23}$$

while $a_{x,y}^{i}$ is the input value of i^{th} kernel on position (x, y); N is the total number of kernels; k, n, α and β are constant values assigned by $k=2$, $n=5$, $\alpha=10^{-4}$, $\beta=0.75$. Similar to local contrast normalization scheme [68], response-normalization [24] layer is applied for brightness normalization, which achieves reduction of error rates for top-1 and top-5 by 1.4% and 1.2%, respectively.

One novel idea for Alex-Net is the utilization of the GPU. GPU has physical advantages (i.e. thousands of cores for parallel workloads) compared to CPU in graphic processing (30% faster than the CPU). To solve the shortage of GPU capacity problem, they utilize two separate GPUs to process convolutional computation independently which makes the computation even faster.

Based on the hardware, it is available to add more kernels in convolution layers to generate more salient features, leading to accuracy rising. As for parallelization scheme, parameters are separated into two groups and put into different GPU separately. To have higher level accurate local features, two group of feature maps are combined together in the third layer. This scheme reduces error rate of top-5 and top-1 by 1.7% and 1.2% separately.

Another difference compared to Le-Net is the utilization of variant kernel resolutions (11*11, 5*5, 3*3). Larger kernel can summarize more general features. At the beginning, since image size is large, the salient regions are sparse that small kernel cannot completely catch the features while kernels of large size can cover wider ranges of pixels containing more features. When the feature map comes with a high level, the density of features increases dramatically. It is necessary to employ small kernels in case of feature overlapping. As for connections between layers, unlike LeNet-5 with non-complete connection scheme, full connection is applied. Dropout technique is applied to adjust parameters efficiently.

In pooling layer, they applied overlapping pooling scheme with 3*3 windows in stride of 2, which reduced the error rate (top-1 and top-5) by 0.4% and 0.3% separately. This scheme also slightly reduces over-fitting problem during training. To further resolve over-fitting problem, two forms of data augmentation are applied for image pre-processing. The first form consists of creating image translations and horizontal reflections by randomly extracting five 224*224 samples from 256*256 images as well as their horizontal reflections. The second form is to alter intensities of RGB channels in training images by PCA (Principle Components Analysis). These two f pre-processing methods reduce top-1 error rate by 1.0%.

In the training process, stochastic gradient descent scheme is applied with a batch size of 128, monument of 0.9 and weight decay of 0.0005. To initialize weights, a zero-mean Gaussian distribution with standard deviation 0.01 is applied. Biases of second, fourth, fifth and FC layers are set to 1 while the rest remain to 0.

2.2.3. ZF-Net.

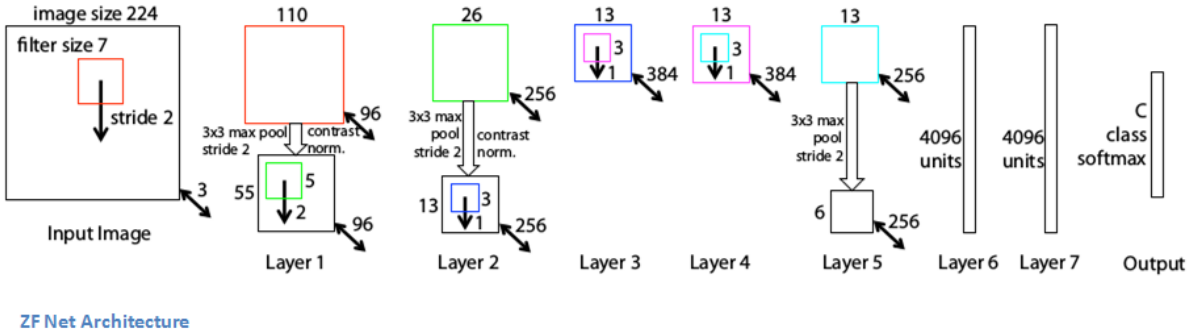


Figure 2.11: Architecture of ZF-Net [69]

Published by Zeiler and Fergus in 2013, ZF [69] Net is an advanced version of Alex-Net. They proposed a novel visualization technique (de-conv net [70]) to monitor computation within intermediate layers in order to refine configurations of Alex-Net. This tool is a convolutional sparse coding model with max pooling mapping features to pixels. The network has the same structure as Alex-Net while designers increase depth of conv layer and shrink the kernel size of the first conv layer.

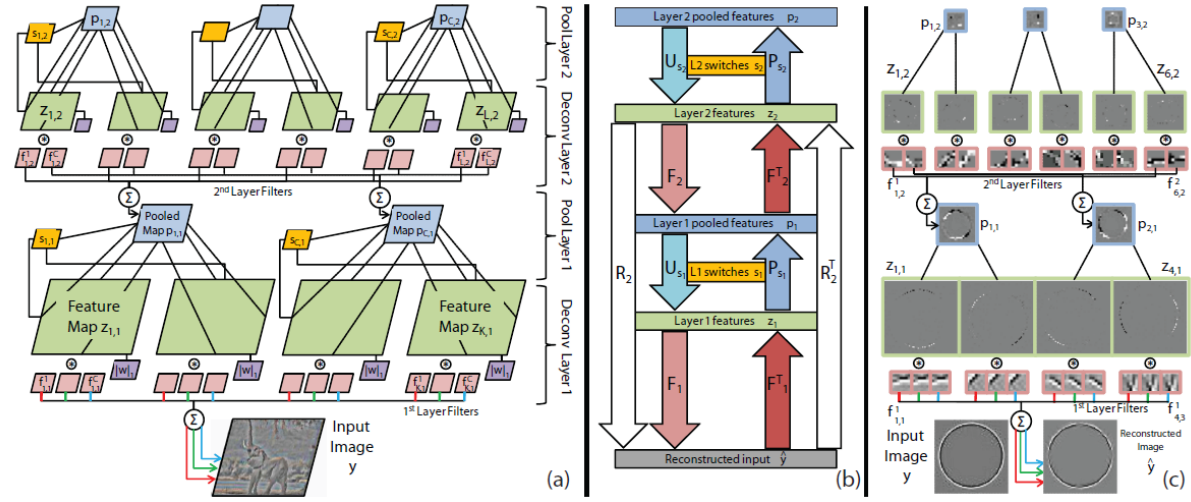


Figure 2.12: Image Processing using De-conv Structure [69]

In a deconv layer (see Figure 2.12), the reconstruction pixel map of layer n for channel c is denoted as a linear sum of all latent feature maps $z_{k,n}$ of the image convoluted with corresponding filters f_c (K_n is the depth of feature map in layer n) by:

$$y_n^c = \sum_{k=1}^{K_n} z_{k,n} * f_{k,c} \tag{2.24}$$

The loss function for training de-conv net of layer n is calculated by:

$$L_n(y) = \frac{\lambda_n}{2}\left||y' - y|\right|_2^2 + \sum_{k=1}^{K_n} |z_{k,n}|_1 \tag{2.25}$$

The first part of loss function is a likelihood term that keeps the input closed to target. The second part is a regularization term to penalize the feature map. The importance of these two parts is controlled by λ_n. The de-conv net still uses BP algorithm for parameter training.

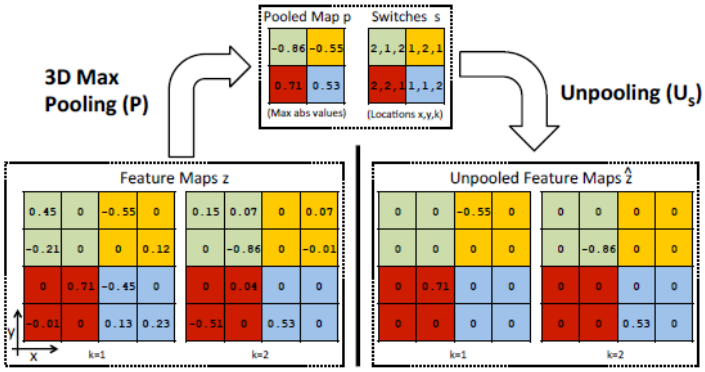


Figure 2.13: Feature Map Unpooling Process [69]

In the pooling process (3D max pooling, see Figure 2.13), the most salient positions with absolute value from all channels are recorded in the switch matrix in the form of (*x, y, k*). Pooled map stores the specific value. After pooling, only the marked positions are assigned with corresponding value, whereas all the others are set to zero.

In the experiments when using this de-conv net to restore each conv layer of Alex-Net, Zeiler discovered that filters of first layer filters contains extremely high and low frequency information. Additionally, the 2nd layer visualization shows promiscuous artifacts caused by the large stride of 4 used in the 1st conv layer. To make a progress, they verified the size of kernels of the first two layers from 11*11 and 5*5 to both 7*7, and reduced the stride from 4 to 2 in order to remain more information in first and second layers. This change also reduces the number of training samples (only 1.3 million images needed compared to 15 million for Alex-Net). Numbers of channels in 3th, 5th, and 7th layers are also adjusted to 512, 1024, and 512 separately to further increase features. They also found the removal of the FC layer only cause slightly error increasing. Overall depth of network is the most important for obtaining good performance. This discovery is further explained by Szegedy et al. [25].

2.2.4. VGG-Net.

VGG Net [27] is another improved net of Alex-Net, published on ImageNet. The network is designed by Karen Simonyan and Andrew Zisserman. Instead of investigating the size of the kernel as well as the width of conv layer for Alex-Net, Karen et al. focus on entire network depth.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architecures of VGG Net. Configuration D produced the best results

Figure 2.14: Different Version of VGG Configurations [27]

Their contribution is mainly for exploring the relationship between network depth and accuracy. In pre-processing phrase, pixels of all RGB channels are subtracted by corresponding mean values via training dataset. Through image pre-processing, the dense matrix is transformed to sparse matrix to fasten convolutional operation. Except for few conv layers that apply 1*1 filter, most employ 3*3 filter with padding of 1 pixel instead of 5*5 filter in Alex-Net and 7*7 in ZF-Net. ReLU is applied as activation function. They designed contrastive network versions by inserting a different number of conv layers with corresponding configurations (Figure 2.14). They also append max-pooling layers after the third and forth conv layer while in Alex-Net or ZF-Net pooling is not considered. Same as former networks, three-FC-layer structure is inherited with the output dimension of 4096, 4096, and 1000 respectively. Softmax is regarded as the output layer for probability evaluation.

During the experiment, they found that the combination of multiple small kernels has the same effort as single large kernels. For instance, two 3*3 kernels have the representation ability as one 5*5 kernel reducing 7 training parameters. Three 3*3 kernel layers have the same effort as one 7*7 kernel layers while reduce 12 training parameters. By implementing small kernels, it is available to extend the depth of the network to at most 19 layers without computational explosion. These changes largely reduce training parameters while increase accuracy. VGG-Net achieves a low error rate of 24.8% for top-1 test and 7.5% for top-5 test separately.

2.2.5. GoogLe-Net.

As one of the most innovative and popular networks, GoogLe-Net [25] was published winning the first prize of ILSVRC-2014 with top-5 error rate of 6.67%. Besides increasing depth (22 layers) and width, the novel idea of this network is the utilization of calculation resources inside the network. Inception module is designed and repeated to generate the entire network following Hebbian principle [71].

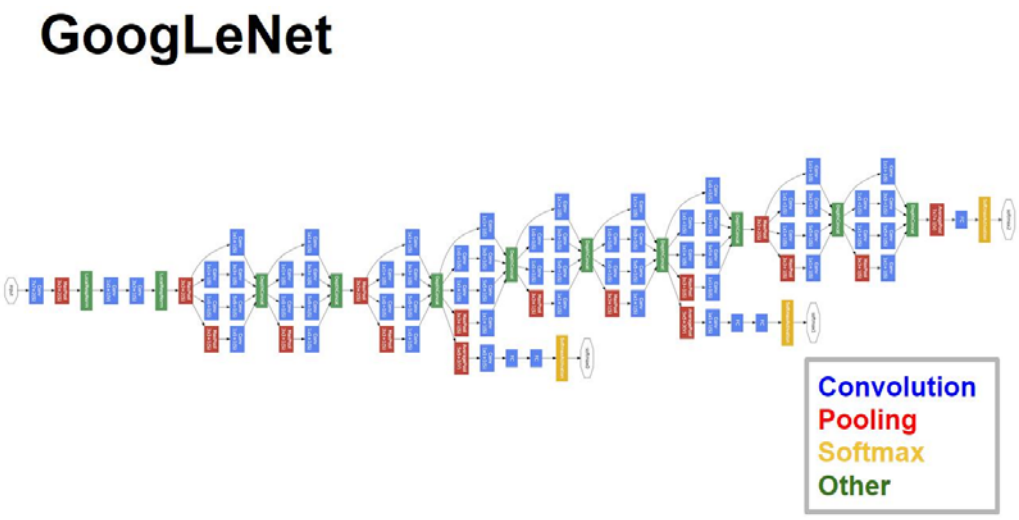


Figure 2.15: General Structure of GoogLe-Net [25]

The most straightforward way to improve network is to extend the width of channels and depth [27, 69] of layers. However, this remains a problem: increment of layers largely raises the number of parameters leading to more computational resources requirement. To deal with the problem, FC layers are replaced by sparsity layers with inception architecture. To deal with multi-scale problem, they applied kernels of different size (1*1, 3*3, 5*5) in the same inception module.

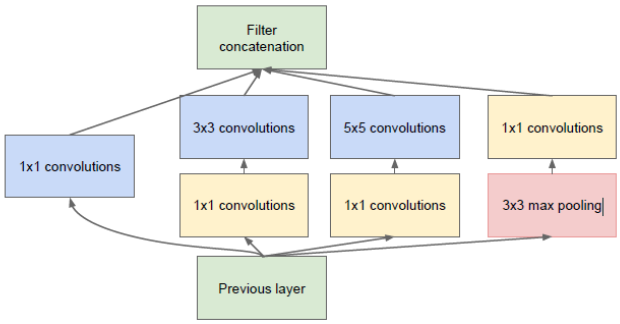


Figure 2.16: Inception Module Structure [25]

In an inception module (Figure 2.16), three types of convolutions are applied as well as one original feature map through the max pooling layer. Various kernels extract features of different scales. The map of previous layer is also regarded as features for the current layer to prevent information missing. One of serious problems met by other network is the bottleneck problem as net depth increasing. Over deep network has inversely higher error rate than relatively shallow network [26]. The phenomenon is same as "forget" problem of RNN. In each conv layer, features are more or less lost because of parameter sharing and pooling operation. After a certain number of conv layer, features of the original image may be completely lost. To break the bottleneck, Szegedy et al. linearly transform input maps with max-pooling and directly send to the next layer. Consequently, the increment of depth and width doesn't have a significant penalty on feature quantity.

To reduce the computational cost for large scale convolutional operation, inputs are merged into three channels with different 1*1 kernels (yellow box in feature 2.16) before convolutional operation. For example, assume that the inputs contain 16 channels. Instead of making convolutional operations for all 16 channels with 3*3 or 5*5 kernels, these channels are merged into only two channels before 3*3 and 5*5 convolution separately. All the maps after processing are concatenated to generate inputs for next layer. ReLU was applied as activation function. Auxiliary classifier is imported among inception modules to solve the vanishing gradient problem by enlarging the discrimination of the features produced in the middle of the network.

2.2.6. Microsoft Res-Net.

Zhang et al. [26] modified VGG [27] network wining LSVRC-2015 first price, which achieves 3.57% of top-5 error. The general structure is shown in Figure 2.17.

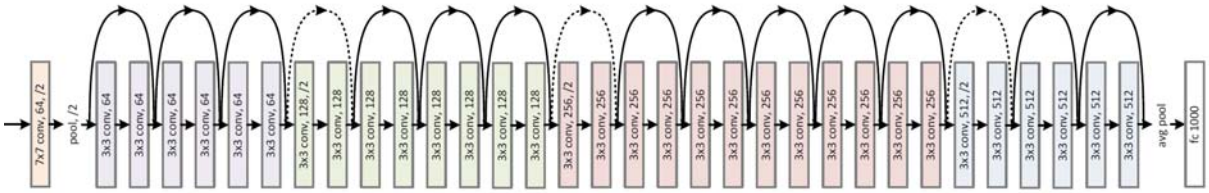


Figure 2.17: Structure of Res-Net [26]

The main goal is to solve the degradation problem [72], which has been addressed in the previous section. Similar to GoogLe-Net that directly sends maps of the current layer to the next layer, Res-Net also remains feature maps through a deep residual learning framework (see Figure 2.18). The framework also eases training of the network.

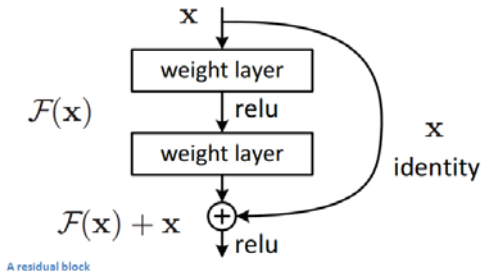


Figure 2.18: Structure of Residual Block [26]

Instead of getting stimulated underlying feature map from stacked nonlinear layers (i.e. $F(x)=H(x)$, while $H(x)$ is the underlying mapping, $F(x)$ is the prediction map from stacked layer), they attempted to generate the difference of the feature map with input (i.e. $F(x)=H(x)-x$). They believe it is easy for a network to reach the residual to 0 than to fit a feature (identity) map. The idea can be achieved by feedforward network with "shortcut connections" [73], referring to connections of identity skipping several layers (Figure 2.18). By doing so, the new network is easy to optimize and has relatively low training error while the bottleneck of network depth is broken. To maintain size consistent of input and outputs, the dotted shortcuts increase dimension by two options: 1) pad identity mapping with zero matrix; 2) apply a linear projection to increase dimension.

The network has only two subsampling layers with max-pooling and average-pooling separately follow first convolution layer and last convolution layer. ReLU is adopted as activation function. Except for the first convolution layer that using 7*7 kernels, all the other layers use 3*3

kernels to reduce parameter. With the help of learning framework, network can be extended to 34 layers without degradation problem.

2.2.7. Crafting GBD-Net.

As winner of LSVRC-2016, CUImage group proposed a novel bi-directional network structure [74]. The aim of GBD-Net is to improve recognition accuracy of plural objects based on the fast RCNN framework [28].

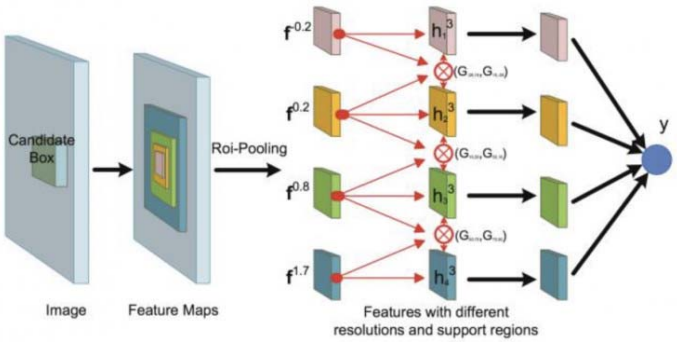


Figure 2.19: Outline of GBD-Net [74]

Adopting fast RCNN as the object detection pipeline, four steps are implemented: 1) create candidate box; 2) create feature map of entire image; 3) apply RoI pooling strategy for features with different resolution; 4) Using BN-Net [75] for proposals recognition.

In region proposal, they believe that single candidate box is not sufficient to localize objects, especially when the candidate contains less or too much relative information. Multiple resolutions of this candidate, including surrounding contextual information are necessary to help describe the property. The same RoI-pooling operations [28] of R-CNN framework is also applied for filtering features of different resolutions with context information generation. Given a proposal region b_o=[x_o,y_o,w_o,h_o], its padded box is denoted by b_p=[x_o,y_o,$(1+p)w_o$,$(1+p)h_o$], while p $\in$ {-0.2, 0.2, 0.8, 1.7}. All these boxes are sent into RoI-pooling layer for feature extraction.

Figure 2.19 shows bi-direction structure. It inputs features f_p of proposals b_p and outputs transformed features h_p^3 for resolution p. To compute h_p^3, two directional connections are designed separately from higher resolution to lower (formula 2.26) and from lower to higher (formula 2.27). The formula of computing h_p^3 is shown as below:

$$h_p^1 = \sigma\left(f_p \otimes w_p^1 + b_p^1\right) + G_p^1 * \sigma\left(h_{p-1}^1 \otimes w_{p-1}^1 + b_p^{1\prime}\right) \quad (2.26)$$

$$h_p^2 = \sigma\left(f_p \otimes w_p^2 + b_p^2\right) + G_p^2 * \sigma\left(h_{p+1}^2 \otimes w_{p+1}^2 + b_p^{2\prime}\right) \quad (2.27)$$

$$h_p^3 = f_p + \beta * \max(h_p^1, h_p^2) \quad (2.28)$$

while *G* is the gate function to control message passing expressed as (sigmoid is the activation function):

$$G_p^1 = sigmoid(f_{p-1} \otimes w_{p-1}^g + b_p^g) \quad (2.29)$$

$$G_p^2 = sigmoid(f_{p+1} \otimes w_{p+1}^g + b_p^g) \quad (2.30)$$

GBD structure refines feature map of different resolution by combining messages of other contexture features to enhance representation ability. For the training process, the loss function is a summation of the cross-entropy loss and the smoothed L1 loss for bounding box regression.

2.2.8. Summary.

Lecun achieved digit handwriting recognition through CNN with 5 layers. Although this network can only recognize low-resolution images in gray space, the concept behind it promotes development of recognition techniques. Based on Le-Net, Krizhevsky proposed a more complicated network successfully achieving 1000 categories recognition task. In this network, 2 GPUs are utilized to fasten the process. In 2014, ZF-Net and VGG-Net are constructed to improve performance of Alex-Net. The former designed a deconv tool to adjust the configuration of Alex-Net whereas later dramatically increase depth of the network. Both networks achieve satisfying results. However, as the increment of network's depth, degradation problem seriously influences the recognition rate. GoogLe-Net proposed inception module to break the bottleneck while Res-Net proposed a deep residual learning framework. GBD-Net aims at improving accuracy of plural object detection by introducing bi-direction structure for proposals of multi-resolutions based on fast R-CNN framework.

2.3. Frameworks for Plural Object Detection

A single CNN structure can only process images of single objects, because in FC layers, all extracted high level features are linearly combined together to represent properties of the entire image. Assume there are two objects O_1 and O_2 with groups of features F_1 and F_2 in the same image. F_1 and F_2 are blended as F_3 through FC, which contains features of both O_1 and O_2. However, classifier of network regards F_3 as one object, meaning O_1 and O_2 are not separable. The mix of multiple category features causes neither of contained objects being classified.

In the real scene, images usually contain more than one objects and may have the complex background (see Figure 2.20).

Figure 2.20: Images of Real Scene (from Google)

This is the reason that CNN cannot directly be applied to the real scene. This section mainly introduces frameworks based on CNN to deal with problems illustrated above. It is important to have a deep understanding for the real scene recognition. So that, we can recognize more complex situations.

2.3.1. R-CNN.

Naïve way for CNN based multiple object detection is to utilize a fixed size window, sliding on the image with step of 1 pixel. Each window region is assumed to contain only one object and is sent to CNN for recognition. However, when it comes to a large image, such as 800*600, the number of windows rises to thousands which will dramatically reduce processing speed. Designed in 2014, R-CNN (Region based Convolutional Neural Network) [28] is a very early version framework to detect multiple objects from images with complex background. The basic idea is very simple: the original image is cut-off to many sub-regions (called region proposals) following

some man-made principles. The system assumes each proposal contains only one object locating in the center area. CNN is executed to recognize proposals one by one.

As the concept, R-CNN framework is combined with a region proposal detector, CNN (for feature extraction and scoring), and regressor (for re-localize object). Other famous object detection systems [29-31] are based on this framework.

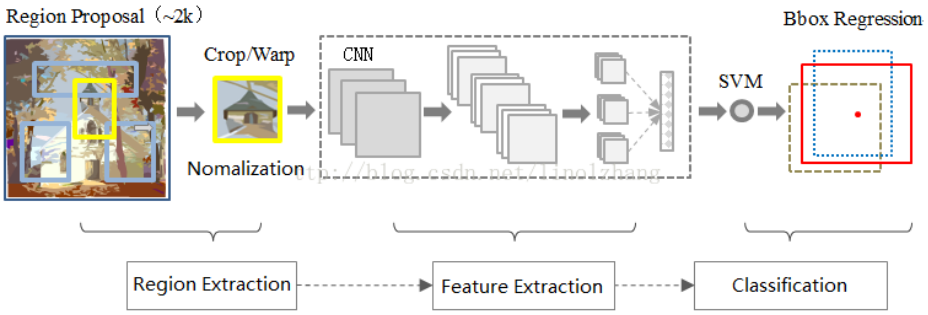


Figure 2.21: Framework of R-CNN [28]

Figure 2.21 is a general object detection process from a complex image. Selective Search Algorithm [64] is applied to generate region proposals from the original image. The number of proposals is restricted to 1000-2000 (plethoric proposals heavily reduce recognition efficiency while an over small number of proposals causes the shortage of information). Because proposals may have different size while CNN only receives data of a certain size, they need to be resized by warping all pixels in a tight bounding box. If proposals are not large enough, instead of warping, the box is dilated to fit the size. 4096-dimensonal features of each proposal are extracted through Alex Network [24] followed by SVM [15]. Because many proposals may indicate the same object, the recognized regions need to be refined, combined and filtered by BBR [65]. In the training process, "supervised pre-training/domain-specific fine-tuning [28]" paradigm is applied. Firstly, network is pre-trained with large single-object dataset. Then several real scene samples with plural objects are imported for network fine-tuning. In total, 20 categories, plus background are prepared for recognition on PASCAL VOC with the accuracy of 53.7%. By applying the framework on the GPU, the detection speed is 13s per image.

2.3.2. SPP Network.

In practical implementing, R-CNN [28] meets with three problems: 1) overloaded proposals (usually 2k regions) need to be stored in advanced before processing leading to lack of memory;

2) artificial proposal cropping and wrapping cause loss of information, especially for large scale proposals; 3) each proposal needs to be processed through CNN while many of them are overlapped together that generates redundant computation cost.

To solve last two problems, SPP (Spatial Pyramid Pooling) network [76] changes image processing order of R-CNN (see Figure 2.22): instead of cropping and warping regions, entire images are directly input into conv network for feature generation. Then, feature map is sent to spatial pyramid pooling for region partition.

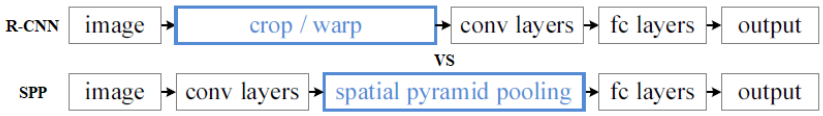


Figure 2.22: Process Comparison Between R-CNN and SPP

The basic idea is to extract features from entire image instead of from proposal regions so that no redundant operation is generated. In this framework, conv layers of ZF-Net are applied for feature extraction. Notice that conv layers accept arbitrary input size while classifiers, such as SVM and Softmax, need fixed data vector as input. SPP [77] is inserted into ZF-Net to transform feature maps to fixed-length vectors (feature proposals) maintaining spatial information.

The idea of SSP layer is similar to histogram generation. Feature maps from conv layers are divided into a fixed number of regions (named “bins”) of different level (see Figure 2.23). In each region, max-pooling operation with corresponding window is applied (i.e. each region only output one value). The output of SSP layer is a 5376-dimensional vector.

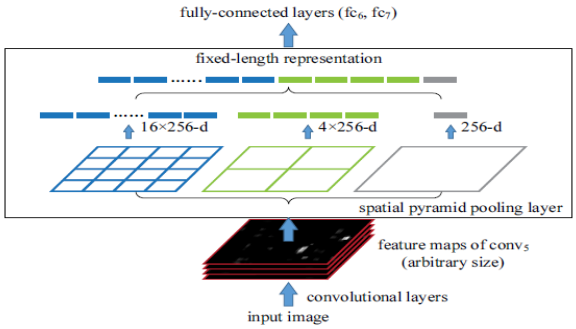


Figure 2.23: Structure of SSP Layer [77]

Output from SSP layer is sent to FC layers of ZF-Net and classified by Softmax. Parameter training for the framework is the same as ZF-Net with BP algorithm. The framework achieves 60.9% accuracy rate for plural object detection (2.6 FPS).

2.3.3. Fast R-CNN.

Compared to traditional R-CNN framework, SPP network dramatically increases efficiency by reducing redundant convolutional operations. However, space cost problem still exists (proposal regions need to be stored in advanced). Besides, Training for SPP network is complex, because parameters before and after SPP layer cannot be fine-tuned in the same training process. To solve these two problems and further increase processing speed, Fast R-CNN [30] was proposed in 2015 by Girshick.

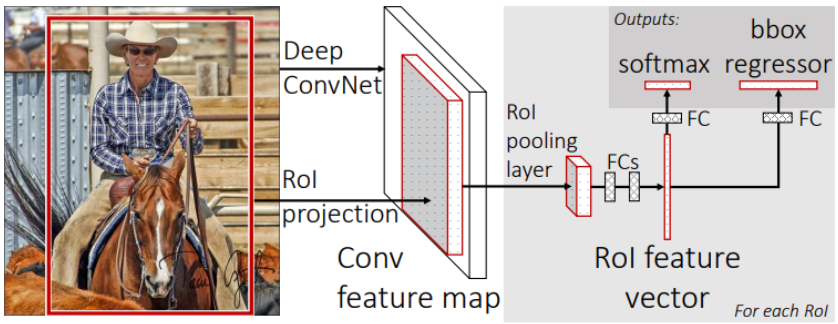


Figure 2.24: Framework of Fast R-CNN [30]

Figure 2.24 is the general structure for object extraction. Feature maps of entire image are generated by VGG conv layers. RoIs are selected by SSP layer [77] of only one pyramid level (called RoI pooling layer). With fewer pyramid level, fast R-CNN saves threefold space than SSP framework. Each feature vector is fed into FC layers of VGG. Outputs are sent to two parallel layers: Softmax layer and bounding-box regression [65]. Former layer estimates the probability of category for proposal regions. Later refines object regions to get close to ground truth. Training process follows traditional BP algorithm [59].

Compared to R-CNN which assigned bounding-box after Softmax layer, they can be processed simultaneously, because they share the same data vectors from FC layer. Training time cost is sharply reduced from 84 hours (R-CNN) to 9.5 hours with the same dataset; test time cost

is reduced from 47 seconds to 0.32 second for one image while keeping accuracy between 66% - 67% on PASCAL VOC 2007 dataset.

2.3.4. Faster R-CNN.

In the SSP and fast RNN frameworks, ROIs are filtered and normalized by spatial pyramid pooling layer. To have a further improvement, Ren et al. [31] proposes RPN (Region Proposal Network) for proposal generation and modification (see Figure 2.25). Because RPN shares full-image convolutional features with discriminate network, it allows nearly cost-free RoI proposals.

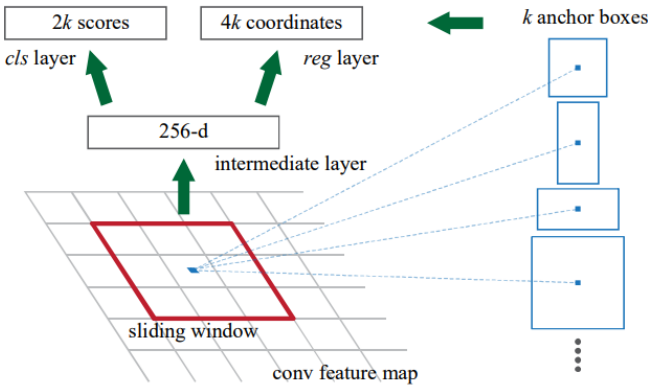


Figure 2.25: Framework of Faster R-CNN [31]

RPN takes conv features (of any size) as input and outputs rectangular object proposals with "objectness" score. It is a single conv layer of 3*3 kernel outputting 256 feature maps. Feature values from all 256 channels of the same position consists of a 256-dimensonal vector to represent information about corresponding position. The vector is sent to box-regression layer (*reg*) and box-classification (*cls*) layer for proposal region generation and refining, same as fast RCNN. The loss function for training RPN is:

$$L(\{p_i, t_i\}) = \frac{1}{N_{cls}} \sum L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum p_i^* L_{reg}(t_i, t_i^*) \qquad (2.26)$$

p_i represents the probability of anchor i in a mini-batch being an object. p_i^* is the ground truth label with 1 if the anchor is positive, otherwise 0. t_i is a 4-dimensonal vector referring to the predicted bounding box while t_i^* is the ground truth of corresponding box. L_{cls} is a binary log loss function. $L_{reg}=R(t_i- t_i^*)$, while R is the robust loss function [30]. $\lambda = 10$. $N_{cls} = 256$. N_{reg} is the number of

anchor location. Faster R-CNN improved accuracy to 78.8% on the PASCAL VOC dataset with 5-17 FPS.

2.3.5. Summary.

Since all CNNs [24,25,26,27,50,74] can only detect single object due to their FC layers, a series of frameworks are designed to assist plural object recognition. The simplest way is to separate original image to abundant proposal regions assuming that each region only contains one object in the center area. R-CNN achieves region proposal segmentation by Selective Search Algorithm [64] and utilize Bounding-box Regression [65] for bounding box refining. However, there exist too many redundant convolutional operations considering each proposal. To decrease the computational cost, SSP network [76] firstly computes conv feature map of the entire image and then uses SSP layer for proposals generation and normalization. Fast R-CNN inherits SPP structure while reducing pyramid level number to fasten process and reduce space cost. The framework also replaces ZF-Net by VGG network to raise accuracy rate. To achieve cost-free for proposal generation, faster R-CNN proposed a simple network (RPN) inserting between feature detecting network and classifier. Faster R-CNN achieves nearly real-time process that are utilized by many recognition systems.

Chapter 3

Research Design and Methodology

In chapter 2, state-of-the-arts CNNs and frameworks are introduced in detail. However, neither of them has a simple but practical structure dealing with multiple object recognition of the real scene. Even for faster R-CNN, two different networks have to be trained before practical usage. In this chapter, a design of simple network is explained, which achieves plural object detection remaining structure simplicity. The chapter consists of two sections: details of how and why to construct network are explained in the first section; the next part is a summary of methods used in the network and the outline of the entire network.

3.1. Analysis and Design

In this section, we introduce our network based on five aspects: general structure, conv layer, pooling layer, loss function, and training methods.

3.1.1. General Structure.

For neural networks of masterpieces in LSVRC, although the accuracy is high enough for real object recognition (exceed 96%), all networks focus on the single object labeling with low resolution (224*224) images. The size of both training and testing samples are fixed which is impractical for real scene. In the real world, even two exactly the same objects may look different. According to environmental factors (e.g. illumination, air density and depth of objects), observer factors (e.g. position, distance and point view), and object factors (movement and deformation), 3-dimensional objects can have diverse 2-dimensional projections. As a consequence, the size of proposal region may vary dramatically. RoI extraction techniques (e.g. selective search, SPP and RPN) must be applied in advance. However, this process is still time consuming with heavily redundant proposals. The questions are: does there exist a single efficient network that can indicate multiple objects from raw image without extra structure? And how to design this network?

A noticeable method is to have another structure named de-convolution network [70] which is designed by Zeiler as a supervising tool. However, in [77], de-convolution is the part of the entire network appended after conv layers. The idea is to reverse feature maps of last conv to intensity map by applying de-conv operation layer by layer until the output has the same size as the original image. The training process is the same as conv layers while no manual label is necessary. Sensitive maps are calculated by comparing output map and original image. However, this method can only restore images that are similar to training samples. In addition, partial occlusion of the object remains outstanding problem. De-conv structure increases depth of the entire network that largely reduce processing efficiency. We cannot use this method for real image recognition, because in the real scene, most objects are overlapped together, especially for the chosen categories, such as vehicles and pedestrians.

It is commonly known that edge detectors [8, 9] have high detecting speed due to the simplicity of operators. The structure of edge detectors and the mathematical theories behind it is similar to CNN. Specifically, edge detectors (operators) can be regarded as a single conv layer with one channel while values of operators are manually deduced by distribution models. RPN [31] takes advantage of simplicity to achieve free-cost of proposal generation. With advanced techniques, such as utilization of the GPU or distributed computing, recognition speed can reach real-time (greater than 24 FPS). The second advantage of simple conv-like structure is unconstrained to inputs' size. Images with arbitrary resolution are available to feed the network. Moreover, it is very convenient to look inside calculation processes of each channel. No sophisticated tool such as de-conv net [70] is required. As a basic module, we can easily extend or modify structures without concerning to many factors.

The first task is to construct such a simple structure with multiple kernels that can efficiently filter different type of features in the real scene. Since traditional detectors only focus on edge or corner features, we need abundant features of different types to describe a complex object. To this end, multiple kernels are necessary. The type of features is decided by kernel matrixes that is deduced by machine learning [22] rather than manual effort. If feasible, we also need to ensure kernels are not similar to each other. Having no contribution to object recognition, redundant feature types are responsible for the more computational cost. The linear combination of weighted features in the homologous position of different channels indicates category labels of

that position. We also considered non-linear functions such as SVM [15]. However, since the entire image may have millions of pixels, non-linear classifiers may sharply reduce recognition efficiency, because each pixel needs a specific classifier.

Generally speaking, our network labels object pixel by pixel. To keep the structure as simple as possible, we follow Occam's razor principle [55], which demonstrates that a simple solution is likely better to be applied for complex general problem. According to the previous analysis, we restrict network depth to only three. To enhance the ability for classification, we verify width of the conv layer. Two most popular activation functions: ReLU and sigmoid are introduced after conv and output layers. With activation functions, our network can deal with noises.

Although there is a very small chance that over-fitting problem happens, because as width increasing, the complexity of the network rises. A small group of training samples may not be enough for parameter refining. We import pooling layer to help solve this problem. To keep the consistency of feature maps, in our pooling layer, we set the non-salient positions of each window with zero. Max-pooling and mean-pooling layers are taken into consideration. As a control group, we also implement a network without pooling layer. We did experiment in chapter 4 to figure out which has the best performance.

We also need to remove FC layers, which has been demonstrated in [29]. Although, FC layers of MLP [52] can achieve size consistency between inputs and outputs, it has two shortages:

- Once the FC layers are introduced, the size of output cannot change. As a consequence, inputs' size is also restricted to meet with the interference of output. The training datasets from ILSVRC are 224*224 images while real scene images don't have particular size. It is not practicable to manually resize the datasets which contain millions of images. Even all the images are refined, the scale of training samples and scene images are not matched. The entire network is not flexible for different size of image training and testing.
- Computational cost is exhausted. Even for a small-scale image, such as 100*100, 10^8 connections are generated, including weights and bias in only one FC layer. Many the real situation images are in at least 800*600 resolution, which demands for billions of parameters. It is not advisable to train such huge number of parameters. Besides, [25] has

proven that the network still works well without FC layer keeping the efficiency dramatically improved.

The network should output a label map instead of probability vector to locate latent objects. Different from image segmentation which uses fixed integer numbers to express different regions, the outputs of the network should be decimals. Usually, image segmentation techniques can only indicate regions belonging to desired class. It has no ability to deal with object (of the same category) partial occlusion problem, because all these regions are assigned to the same value. On the contrary, the decimals indicate distance between outputs and ground truth. The distance can refer to the similarity (or probability) of this pixel to the corresponding class. By doing so, the over-lapped objects can be distinct clearly by setting similarity property. Center positions of object have close label value to ground truth while marginal regions have relatively different values.

3.1.2. Conv Layer.

VGG network [27] proves that the depth of network has a dramatic influence on detecting accuracy. Theoretically, single-layer network is not complex enough to describe the property of large-scale image which contains millions of features. However, its not our goal to scarify efficiency for accuracy by enlarging depth. We believe that as enlargement of network width, same accuracy can still be reached. Unlike conventional networks quartering feature maps by pooling layer, we need to keep the size of feature maps consistency for each layer.

We don't have to consider the saturation problem proposed by [26], because our network only has one convolution layer. We do have to consider the max breadth of network that we can reach to, because over wide networks may generate redundant kernels which make no contribution to recognition. Le-Net uses 6-channle structure for their first conv layer while VGG imports 64 channels. Both of networks have good performance at object recognition. We can assume that channel number between 6 and 64 is a "safe" range. In this case, we initially set number of channels to 12. For R, G, B channels of input, each contains three sub-channels. 12-channel structure is easy to implement and convenient to monitor feature generation process. More importantly, the structure is complex enough for a small number of category distinction. In the future, we will make experiment to figure out the influence of conv width variation to accuracy and efficiency.

Notice that real images likely contain objects of different scales, meaning the same type of feature may be differently expressed according to their scale. Fixed size of kernels will ignore this property that cause incomplete description. For instance, let's make a 11*11 rectangle as an object. The object can be completely described as "rectangle" with 11*11 kernel whereas with a 3*3 kernel, the object is interpreted as the composition of lines and corners in a certain arrangement. With 1*1 kernel, only points are detected which is far away to represent a rectangle. One solution is to use the smallest kernel (1*1) or the second smallest kernel (3*3) for all feature extraction and utilize some principles for feature combination (all large-scale features can be regarded as the combination of small features).

Traditional CNNs utilize multi-layer structure to automatically achieve combination task. Due to the simplicity of the structure, this method doesn't work for our network. We meet with three relative problems: 1) even with enough number of channels, small-scale features (local feature) can hardly describe a complex feature (global feature) within one conv layer, because its detecting region is too small (contains less information); 2) small-scale kernels need abundant channels for feature filtering, meaning explosive number of connections are generated, which causes huge computational cost; 3) The same features of different scales are described differently. The inconsistency of feature scale causes redundant description of the same feature with different kernels.

Inspired by the inception module of GoogLe-Net [25], multiple scale kernels can be applied on the same layer, which will dramatically reduce the negative effort of multi-scale features. Besides, Karen et al. [27] proved that on the condition of large scale training and testing sample, multi-layer small-scale kernels has the same effort as single-layer large-scale kernel, which can reduce parameters for training. Vice versa, we can replace multi-layer structure of a single layer with larger kernels. So, multiple scale kernels (3*3, 5*5, 7*7) are considered in the new structure to sole scale-variant problem and increase potential network depth.

The computational efficiency can be guaranteed by distributing computation. Unlike deep CNN that the computational efficiency of the next layer is restricted by previous layers, all channels of our network can compute independently. We can utilize multiple CPU, GPU or cloud to compute feature maps in parallel.

Activation function is appended to the end of conv layer which is very useful to smooth category boundary in order to solve over-fitting problem. ReLU [51] is proven by Alex-Net [24] to have the best performance on both speed of convergence and solving over-fitting problem. We also applied sigmoid function to figure out which type of function has the best performance for our network in terms of vehicle and face recognition.

3.1.3. Pooling Layer.

The function of pooling layer is to transform feature map into a sparse matrix (squeeze feature map to reduce the map dimension without losing too much information). In the real situation, noises from the background are more than useful features which dramatically influences recognition accuracy. Besides pooling layer, sophisticated CNNs deal with noise also by multi-conv layer structure. Each conv layer has a certain ability to filter noise. However, since our network has only single conv layer whose filtering ability is weak, pooling layer is the main method to remove noise in our case.

Usually, traditional pooling layer changes size of output depending on kernel size and stride. For example, non-overlapped pooling layer with 2*2 windows outputs matrix with 1/4 size of inputs. In our network, we have to keep the output size the same as inputs. We propose a practical scheme to re-construct pooling layer: take the same strategies (max-pooling or mean-pooling) as traditional pooling methods while after selecting the most salient value, the other positions of the window are remained with certain values (mean value for mean-pooling and 0 for max-pooling); So that, there is no variation in size of feature map. The result of the experiment is shown in 4.2.3 with specific configurations.

3.1.4. Output Layer.

The output layer of our network is also simple. Instead of using Softmax [43] or SVM [15], we linearly sum up feature values of all channels together to represent labels for certain positions. The reason is as below:

- Softmax is a probability function that indicates the likelihood of vector belonging to a certain category. The most important fact is the settled size of output with these functions. Conventional CNN frameworks utilize this property to restrict the size of region proposal.

So that all indicated bounding boxes are in the same size. Because no region proposal is generated in our network, the output from conv layers refers to the entire image. We cannot apply these two functions.

- Instead of percent value ($v \in [0,1]$), we need a real number to directly represent labels. The distances of average label values for categories should be large enough.
- The output of Softmax and SVM is a victor without position information. To indicate potential object on the image, position information is a crucial factor.

To increase accuracy, we also append one activation function after output layer.

3.1.5. Training.

We have to carefully consider how to train network since the outputs are the real value map instead of possibilities. Based on back propagation algorithm [22], there should be a loss function to decide convergence of training iteration. However, since the output is the entire image map and requirement of output is the real number, not probability, traditional loss function is not feasible in this case. The representation of the target should be the map with only label numbers on it (i.e. all pixels are assigned to corresponding label number of categories).

There are two strategies for loss function selection: 1) compute difference of each pixel and use Euclidean metric to evaluate similarity between output and target. We have to make experiment to see whether there exists a group of separable kernels. Different from MLP [52] that all neural nodes have specified weights for adjusting, CNN shares weights with neighbors, it is hard to prove the existence of such kernels. 2) compare the mean value of output map with labels. This strategy may reduce the noises from feature map while keeping the general label tendency the same as the target. These two strategies are all applied in experiment to decide which one has better performance.

During experiments, we meet with some critical problems which is analyzed in 4.1. Pre-processing steps are considered to refine images (remove noises, transform color space or data type, adjusting illumination and contrast ratio) only when the network works for the raw image, because it may reduce processing efficiency. To decide the importance of pre-processing to the entire network, it is considerable to: 1) use Gaussian filter for noise removing; 2) utilize different

color space to reduce the influence of illumination; 3) use Histogram of orientated Gradient to compute pattern map. We have to consider the efficiency of pre-processing, especially for computing pattern map, because doing gradient computation for the entire image will cause lots of time. Raw image with RGB color space will be mainly considered in view of most images from camera and videos are represented by that space. However, to prevent gradient error (see 4.1) when applying the sigmoid function for over-large values, original data maps to [0,1].

3.2 Summary

This section is a summary of previous analysis on our network structure. The basic structure is shown as below:

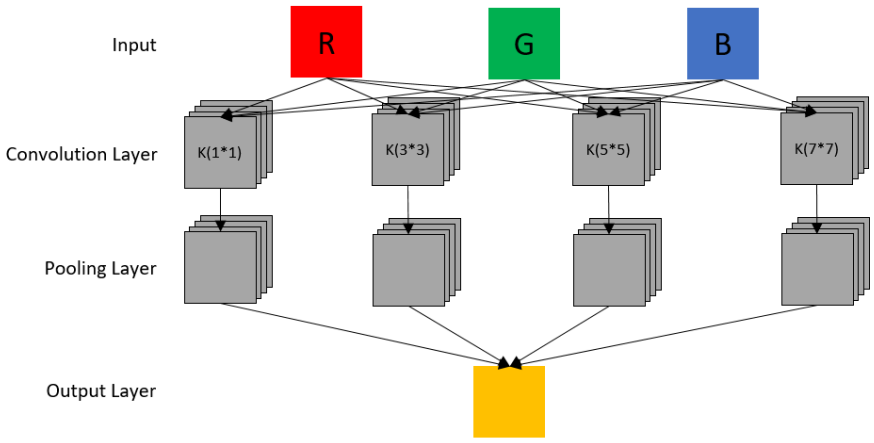


Figure 3.1: Structure of Proposed Network

In total, the network has three layers, besides the input layer: one conv layer, one pooling layer and one output layer. The original image is separated into three channels that only contains values of: red, green and blue. All channels have been pre-processed: color values of pixels are reflected to [0, 1] to remove the training parameter oversize problem.

According to the image, which is separated into three channels (i.e. red channel, blue channel and green channel), each channel connects to 4 sub-channels. These sub-channels are: one channel with 1*1 kernel, one channels with 3*3 kernel, one with 5*5 kernel and one with 7*7 kernel. In the future work, more channels and scale of the kernels will be appended on convolutional layer depending on the test accuracy. Inspired by GoogLe-Net [25], channels with

1*1 kernel is regarded as the linear transformation of original image to prevent loss of information from parameter sharing. The feature map from conv layer is as below:

$$g(x) = f(\boldsymbol{W} * \boldsymbol{X} + b) = f(\sum w * x + b) \tag{3.1}$$

while *f*(.) is activation function with different types. $\boldsymbol{X}$ is input matrix. $\boldsymbol{W}$ and b are weight vectors and bias

The number of channels except for 1*1 kernel will be modified during the experiment to adjust recognition accuracy. The initial value of weights follows Xavier initialization [80], because this principle has a great performance in many networks. The distribution of weight is within the range:

$$[-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}] \tag{3.2}$$

while *m* is the input data dimension, n is the output data dimension. For our network, *m*=*n*=1024, i.e. $init(w) \in [-0.0541, 0.0541]$. Initial weights are generated randomly from this range. As for biases, they are simply set to 0, same as other classical networks. ReLU (i.e. $f(x) = max(0, x)$) is regarded as the main activation function while sigmoid and removal of activation function are also two comparisons during the experience to decide which function has the best performance.

As for pooling layer, we proposed two strategies and did experiments to test which one has the best performance (experiment details is shown in chapter 4.2.3):

1) max-pooling: modified max pooling method with 2*2 scanning window of stride 2, similar to NMS (Non Maximum Suppression) [8]. Except for the max value in the window, all other positions are assigned to 0 rather than remove those positions to reduce map dimension. All remained positions are recorded for the purpose of back propagation processing.

2) mean-pooling: modified non-overlapped mean pooling method. This method follows a classical pooling method which uses 2*2 window with step of 2. Similarly, we also need to keep the consistence of output data size. All the four positions are assigned to the same average value in the window instead of combining four in one.

As an initialization, pooling layer is directly connected to a linear classifier (output layer) whose initialization also follows Xavier rule. The basic Euclidean metric is regarded as loss function. i.e.

$$Loss(x_{ij}) = \frac{1}{2}(y - x_{ij})^2 \tag{3.3}$$

To keep the function concise, we control weights variation by certain learning rate instead of penalty term. The result of loss function represents the error of each pixel in output map. So the sensitive map of output layer is jagged. We also consider another case: the error of output map is represented by only one value. This value is the average of all distances of pixel. The formula is shown as below:

$$Loss(x) = \frac{1}{2N}\sum(y - x)^2 \tag{3.4}$$

N is the total number of output pixels (in this case *N* equals to 1024). For a single object detection, because the output represents only one class while for plural object detection, the output represents label spectrum. As a consequence, each pixel of sensitive map for output layer is represented by the same average error value. We show performance of both cost functions in chapter 4.1.1 and utilize the one with better performance for further experiments.

Datasets: vehicle dataset is from Universidad Politécnica de Madrid (UPM), Vehicle Image Database [81]. This dataset contains 3425 vehicle images with different viewpoint and 3900 non-vehicle images as background. Each image is in size of 360*256. INRIA Person Dataset [82] is used for pedestrian recognition. The dataset contains 2416 persons in size of 96*160 and 1218 background images in size of 437*292. All non-background images contain only one object and they are separated into 90% for training and 10% for testing. Background images of vehicle and pedestrian will be combined together. We collect samples of other categories (face, dog, cat, bicycle, flower, tree, building and bird) directly from google image dataset. All samples are re-sized to 32*32 with only one object on it.

As for training, back propagation algorithm is applied. According to chain rule derivation, for output layer with loss function L and activation function of output layer f_{out}, the variance of weights on output layer is computed by ($\otimes$ refers to convolutional operation):

$$\Delta W_{out} = \frac{\partial L}{\partial W_{out}} = \frac{\partial L}{\partial f_{out}} * \frac{\partial f_{out}}{\partial W_{out}} = L' \otimes f'_{out}(x) \quad (3.5)$$

Variance of bias for output layer is the sum of all elements in L'. Sensitive map of pooling layer is:

$$\delta_p = L' \otimes W_{out} \otimes f'_{conv}(x) \quad (3.6)$$

f_{conv} is activation function in conv layer. After calculating δ_p, weight is updated by (σ is learning rate initially set to 0.1):

$$W_{out} = W_{out} - \sigma * \Delta W_{out} \quad (3.7)$$

Bias of conv layer is the sum of δ_p. The formula to compute derivative of weights in conv layer (W_{conv}) is:

$$\Delta W_{conv} = in \otimes \delta_p \quad (3.8)$$

where *in* is input data. We use the same learning rate of output layer to compute weight in conv layer. Final iteration number is set to 100,000 while at the beginning learning rate is set to 0.1. In each 10,000 iterations, the learning rate is divided by 10. We adopt mini-batch [83] strategy training network.

Chapter 4
Experiments and Results

In this chapter, settings and specific process steps of various experiments are explained. We illustrate the reason to make such experiments and analysis result of each experiment at the beginning of sections. In terms of problems and shortages of the network, we proposed solutions and schemes to consummate our network. The experiments are introduced step by step, which mainly have two sections: proof of various abilities and performance improvement methods. The first section focuses on whether the proposed network works for single and multiple category classification as well as the variance of sample sizes. In the next section, we modify structure and compare different layer types to improve efficiency and accuracy. We also improve convolutional operation according to the data structure in OpenCV by DFT (Discrete Fourier Transform) algorithm.

The computer we rely on is Alienware 13 R2 with Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz. All experiments are applied on one CPU with Windows 10 (64 bits) operating system. The program is compiled on Microsoft Visual Studio 2013 with OpenCV 2.4.0 (Open source Computer Vision) library. From the datasets, we randomly select 50 samples for each class as training group and 100 as testing group. We also resize the image from 224*224 to 32*32 in order to speed up experiments. In total, the processed datasets include 1500 32*32 images from 9 categories (vehicle, face, building, bird, cat, dog, bicycle, flower and tree). We also create 150 pedestrian samples in 32*60 to test whether the network has ability for multi-resolution sample training.

In all experiments, input images are pre-processed: images are divided into three matrixes based on color channels (red, blue and green). All values are normalized (divided by 255) to make sure that any computed value will not beyond the limitation of data type in OpenCV (we use the double data type which has a range in $[-1.7 * 10^{308}, 1.7 * 10^{308}]$). Because in the last step of back-propagation, the padding for input channels is necessary to compute weight derivative matrix, if we extend channels in each training loop, it will cost approximate O(4mn) computations (m is

the total number of extended pixels, n is the number of training epic). We can extract padding process from training process and insert to image pre-process to save computation.

4.1 Functional Test

The first ordeal is the distinction ability test of the proposed structure to different categories. In this case, the distinction ability includes: ability for single class labeling, single object detection, multiple class labeling, and plural object detection on the white-color background. Besides, we also test the stability of the network based on size changing of training samples, size inconsistency of test samples, and adjustment of illumination.

Because the aim of this experiment is to test whether the network can distinct various objects (if small images can be recognized, it will also work for large images) under different conditions, we restrict the input image to 32*32 to fasten the experiment process. With less time cost of experiments, we can make more adjustments to test norms, so that we can fully understand the advantages and shortages of this structure.

We apply simplest structure in this section: 12-multi-scale-convolution layer, no max-pooling layer, no activation layer, one output layer with square error cost function. The maximum number of training iteration is 10,000 to ensure that iteration is large enough to get to saturation condition.

4.1.1 Single Object Recognition.

4.1.1.1 Selection of cost function.

Before functional test, we need to select the cost function (i.e. error function) for network training. In other networks, the outputs are probability vector. Each element in the vector represents the probability that the input belongs to which category. During back-propagation, the error distance (sensitive map of the output layer) between output and the target is computed commonly by log-likelihood function with weights punishment function. However, in our network, the output is the matrix of real numbers which have no specific range. We don't know whether randomly selecting initialization of target label would cause different label distribution of categories. We cannot apply method of the other networks because their category of output is independent to each other while

each pixel of our output represents category spectrum. We have many choices to compute sensitive map. Here, we propose two strategies for cost function:

1) Generate a label matrix with the same size of output. All pixels in the map have exactly the same value as assigned label. The error map of each pixel is computed simply by square error function in the output with corresponding position of pixels in the label map. i.e.:

$$Error(i,j) = (Outut(i,j) - Label(i,j))^2 \tag{4.1}$$

In this strategy, values of pixels in the error map are different. We want to record the influence of each pixel to the entire sensitive map of training so that effort of all errors can be utilized to generate accurate derivative, which is a direct and reasonable way for the most networks. The sensitive map is accurate to evaluate how much should the parameters be refined. To evaluate the performance of this strategy, we need to record and compute the distribution of each pixel value for different category to see whether their label distribution is separable. The most obvious way is to construct label histogram.

2) If the first strategy computes error map in a microcosmic way, the second strategy is in a macroscopical way. We regard the entire output as one class value comparing to target label. First, we compute the average value of output map; then the square error function is applied to compute the distance with target label; finally, we assign this distance value to all error map. i.e.:

$$Error(i,j) = (\frac{1}{n}(\sum Output(x,y)) - Label)^2 \tag{4.2}$$

Considering the feature of CNN, we think this error function is applicable to adjust shared weights, because in each channel we have exactly one kernel. The macroscopical function will catch the general variation tendency and ignore specific but uncorrected changes, which is fatal to parameter refining. However, we need post-process of outputs for object recognition. The output data cannot directly indicate label information, but its average value. In the test section, we need the average value map of images. i.e.:

$$I(x,y) = \frac{1}{mn}\sum_{i=1}^{x+m}\sum_{j=1}^{y+n} Output(i,j) \tag{4.3}$$

while *m* and *n* are width and height of training samples separately. We evaluate second strategy by directly comparing difference distance of two categories.

We make experiment to test the influence and performance of these two strategies with 100 32*32 RGB training samples for vehicle and face. Each class has 50 samples containing only one object in the center. Object covers major area of image (IOU > 0.5). The label assigned to face is 3 and assigned to the vehicle is 1.

We evaluate the performance of the first strategy by listing value distribution of all pixels with label histogram and the distance between two picks. As for the second strategy, we compare the label distance between two classes and the speed of convergence. Notice that we applied different evaluation criterion for two strategies, because the outputs of these two strategies are in different dimensions. Large distance of categories is admired which has high accuracy for the classifier to identify different classes. Because the first strategy outputs label matrix of all pixels, we compute the average label value to compare it with second strategy. To make it convenient for distinction, we label Net-1 as network applied strategy one and Net-2 as the one applied second strategy.

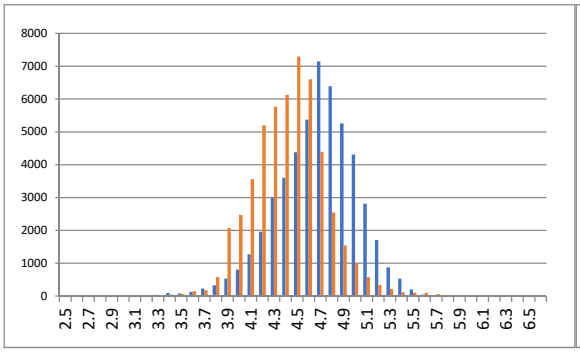


Figure 4.1: Histogram of Net-1

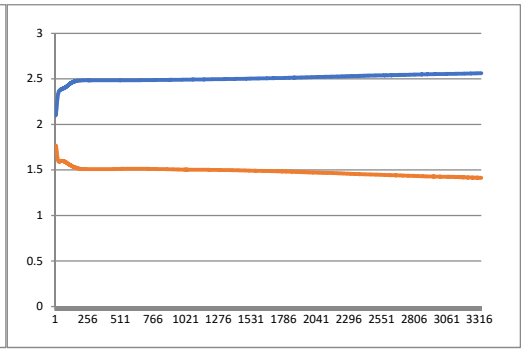


Figure 4.2: Mean Value of Net-2

Data indicated by orange color refers to the vehicle while the one with blue color represents the face. Table on the left side (Figure 4.1) is the label distribution histogram of Net-1 with all 100 samples (in total 102,400 pixels) of the last training iteration. Y-axis represents pixel numbers while x-axis represents bin values of the histogram. The distribution of both categories follows the Gaussion distribution. Most face labels are in the range [4.2, 4.7], whose peak point is at 4.5. Vehicle labels are in [4.5, 5.0] with the peak point at 4.7. 90% pixel values are located between 3.5 and 5.5. More than 60% pixels shares the same value of face and vehicle.

Table on the right side (Figure 4.2) indicates tendency of the average label values for each class of Net-2. Y-axis represents estimated label number while X-axis represents the number of

train iterations. Labels changing starts at 2. After around 249 training loops, tendencies of both classes become stable. At the last loop, trained label to face is 2.6 while to vehicle is 1.4. There is no intersection between two mean labels in the stable period. The largest distance of two categories is 1.0.

It is clearly shown that both strategies more or less have efforts for face and vehicle distinction. However, neighter of them predicts the exact same label as assigned: the peak value of the first strategy is 4.7 for face and 4.5 for the vehicle; second strategy is 2.6 for face and 1.4 for vehicle. Compared to Net-1, labels of Net-2 are much closed to the ground truth. We cannot ensure that training sample vectors of different classes are orthorhombic (i.e. samples from different categories may dependent to each other). Moore-Penrose pseudoinverse is a way to reduce this influence. Sharing parameters of kernels to multiple class will, in some degree, equalizes label values of each category, due to the dependency of output value towards class. In the experiment (4.1.1.2), our network correctly labels one class objects, because label tendency only forward to one orientation. In multiple class training, different sample groups will drag tendency to different orientation. The label of face and vehicle are located around 2 for Net-2 which is the average value of assigned labels.

Before the experiment, we thought Net-1 would have a better performance than Net-2, because parameter adjustment concludes all efforts of pixels in the error map. It should be more accurate to describe variation tendency. However, Net-1 has very close label distribution between two categories which may not be acceptable when recognize real objects. On the contrary, Net-2 has a relatively larger label gap without interaction. We think one reason is that the diversification of error map pixels causing the sensitive map become irrestively complex (contains too many noises). The kernel is shared to all pixels in one channel, this diversification will cause different derivation orientations to parameter adjustment, most of which have a negative influence. In a very high chance, when the label get close to one category, the opposite orientation derivative is met which drag the label to the other category. Output label of the network will never go close to one assigned label. Although Net-2 meets the same problem, this disturbing fact happens slighter than Net-1. The macroscopical strategy reduces the effort of individual derivative, but keep the general tendency.

Net-2 still refine the parameters after 3300 training iterations while Net-1 reach to the stable status before 159 loops. Although sharing parameters can sharply reduce the computation cost for each iteration, it increases iteration numbers. We can explain it as: it is hard to identify an object with just one feature, however as the type of feature increasing, it becomes easier to figure out the object.

We find another fact for Net-2: post processing (compute the average value of output map) is necessary to indicate classes. We generate value histogram of Net-2 outputs same as Net-1, which is shown below:

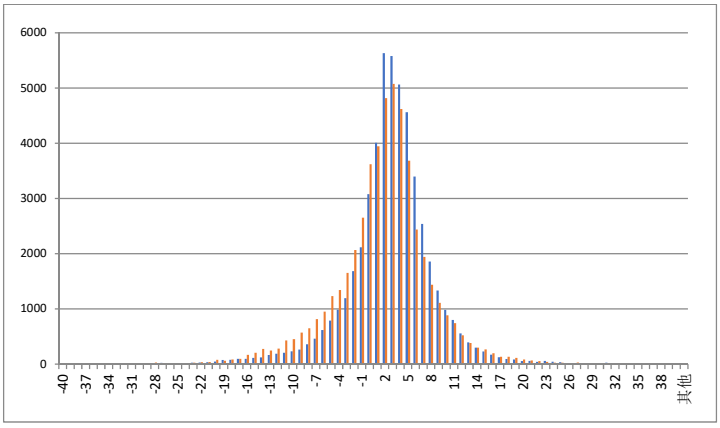


Figure 4.3: Output Histogram of Net-2

The table shows that value distributions of two class in Net-2 completely overlapped together, meaning the output map has no ability to distinct different categories. As a result, if we apply strategy 2 as cost function for multiple object detecting, we also need to apply post processing to compute average value with 32*32 window.

Generally speaking, both networks have the distinct ability in a certain extent. Although their outputs are not the same as assigned, their trained labels can still be utilized for object distinction. Net-2 has much better performance than Net-1 in separating capacity regardless of processing efficiency. As a result, we choose the second cost function for the rest experiments test.

4.1.1.2 Unique category labeling.

The aim of this experiment is to test whether the structure converge after a certain number of training iteration, which can prove the network ability for class labeling with different activation functions. This experiment is also divided into two sub-tests: firstly, training group contains just

one image to test whether the network can label specific image correctly; secondly, extend training samples to 50 images to test whether the network can label a group of images, each of which has different pattern distribution.

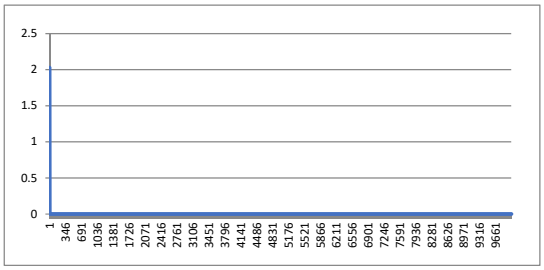


Figure 4.4: Mean Value Tendency of Output

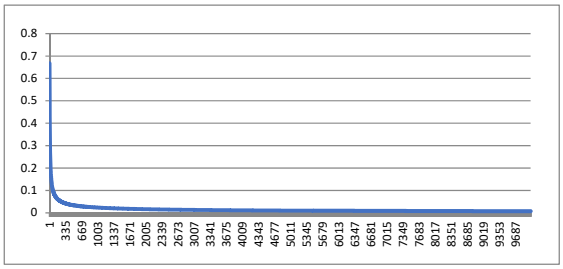


Figure 4.5: Mean Value Trend of Output (modified)

After setting up the network, the first problem met in the experiment when using ReLU as activation function is that, except for the first training loop, all the other loops have the zero AED (Average Euclidean Distance) which makes derivative of parameters become zero (Figure 4.4). We call it “zero-matrix” problem. We don’t find this problem when using sigmoid as activation function (Figure 4.5).

To figure out the reason, we check output matrix values of each step in back propagation processing. After the first loop training, we gained modified parameters which is correct. Then we focus on the second loop of training. After the activation function of the convolution layer, we get a matrix of zero while the computation of convolution layer itself is correct. The problem to make matrix zero caused by convolution operation - the large negative value of bias. The bias adjusted from first iteration is too large (even we initialize all biases as zero) that it cannot be complemented by the multiplication result of weights and inputs that makes all output values of convolution layer under zero. When the negative matrix goes through the ReLU, all values are set to zero. According to this, we add learning rate (share the value with a learning rate of weight) for bias to restrict its amplitude of variation.

We have not found any report of this problem from other masterpieces because they connect networks to classifiers such as Softmax. During the BP algorithm, the sensitive map will multiply first-order derivative of the classifier function which sharply shrinks the value of sensitive map under certain boundary. Because the derivative of bias is the sum of the sensitive map, the growth of bias is also limited. However, the output of our network is the matrix of real numbers (label), we cannot apply for classifier such as Softmax or SVM (the output of these classifiers is

the percentage of chance) to restrict sensitive map values. Another reason is the lack of penalty items in our loss function, which also has a function to restrict growth of parameters.

Besides adding learning rate, the computation of bias's differences also uses the mean value of sensitive map instead of sum. This is because all values of sensitive matrix are the weighted sum of kernel area. The max value of sensitive matrix can be n (n is the number of elements in kernel) times larger than the original value. Sometimes this value can exceed control because the base value is too large. As a result, the normalization of sensitive map will prevent this case happening.

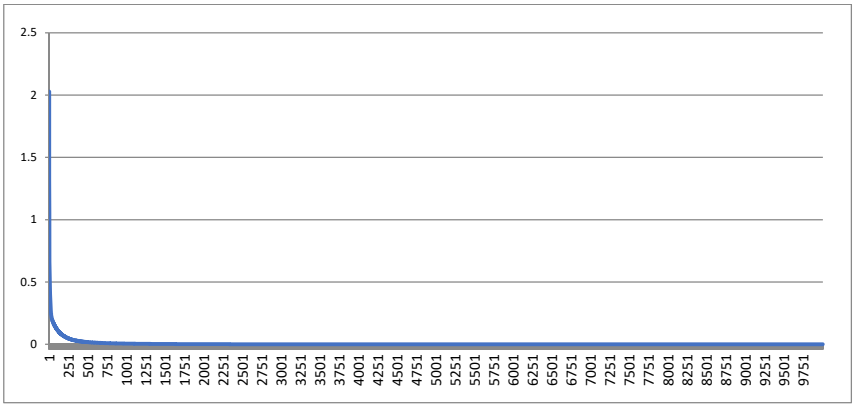


Figure 4.6: AED Trend of Network with ReLU

Figure 4.6 shows AED value of each epic for 10000 training loops after we modify the network. Y-axis represents AED value. X-axis represents the number of loops. The smaller the value of AED is, the closer the output gets to target. AED reduced sharply before 436 loops and then become stable, which indicates that for a single image training, 500 training iterations is enough. The tendency of AED means that the network can correctly label the image with an assigned label number. No zero-matrix problem happens again after network refining.

Notice that the sigmoid function doesn't have zero-matrix phenomenon because the first-order derivative of sigmoid function has already bound the value of sensitive map between 0 and 1 no matter how large the original value is. So, the difference of bias and weights will not increase dramatically. However, the sigmoid function cost more computation than ReLU and has slow restrain tendency. Network without activation function also doesn't have this problem, because there is no "filter" to remove values (under zero) from feature maps. As a consequence, the output matrix cannot be transformed into zero matrix.

It is proven that there exist appropriate parameters of network to describe one specific image correctly. However, if these parameters are only in allusion to specific images, rather than some similar ones, the network still doesn't work for object recognition. So, we also need to see whether the network has ability to extract common properties of a group samples (i.e. objects of the same category but different patterns). In the next experiment, training group is enlarged to 10 face images, each of which only contains one face on the center of the area. In this test, learning rate is 0.01, and training iteration is 500 according to the first section result. The label set for class is 4 to make a difference to the value of AED. We count the average labeling value of these ten images for each loop, and show the result as below:

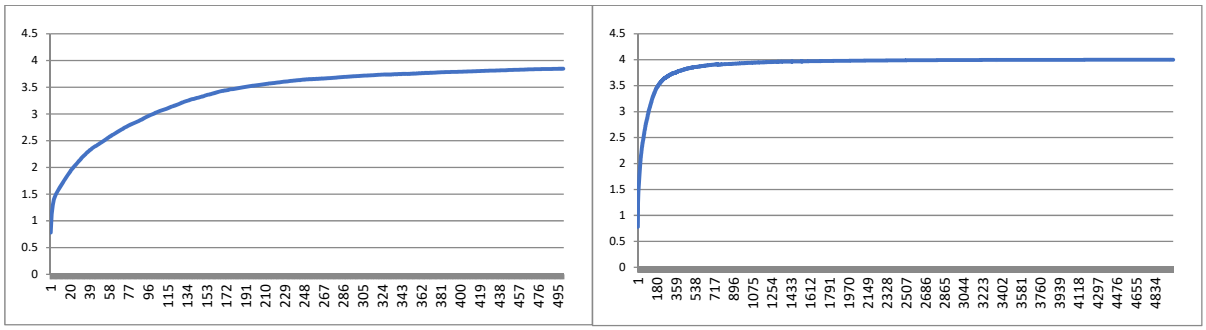


Figure 4.7: AED Trend with Multiple Training Samples

Y-axis represents the label value and X-axis is the number of training loop. It is obvious on the left table that the training process has a correct tendency for face category. However, 500 iterations are not enough to accurately train network. Then we increase training loop up to 5000 and record training labels for each loop which is shown on the right chart. From the table, we notice that for 10-sample training, number of necessary training loop doubles to get saturation status. This is caused by feature noises as well as the difference of each image. As increment of sample quantity, noises also increase. Network need more learning iterations to distinguish common features from noises. Besides, there are more details to describe the face, which cause long learning period to extract common features with limited numbers of parameters. This experiment reminds us that the small number of training iteration cause under-fitting problem. To prevent this problem from happening when a large number of samples are trained, we enlarge training iteration to 5,000 for the rest experiments.

4.1.1.3. Multiple categories labeling.

In this experiment, network will be tested based on whether it can distinct 10 categories objects with 10 samples in each training group. The selected classes relate to traffic scenes: face, vehicle, pedestrian, dog, cat, bicycle, flower, tree, building and bird. All samples are in 32*32 format, RGB color space. Number of training iteration is set to 1000. Because the number of classes is more than previous experiments, we need to increase the training loop to ensure network reaches to stable status. In each iteration, all these ten classes images will be input into network one by one. We set ground truths of these ten classes from 0 to 9 respectively.

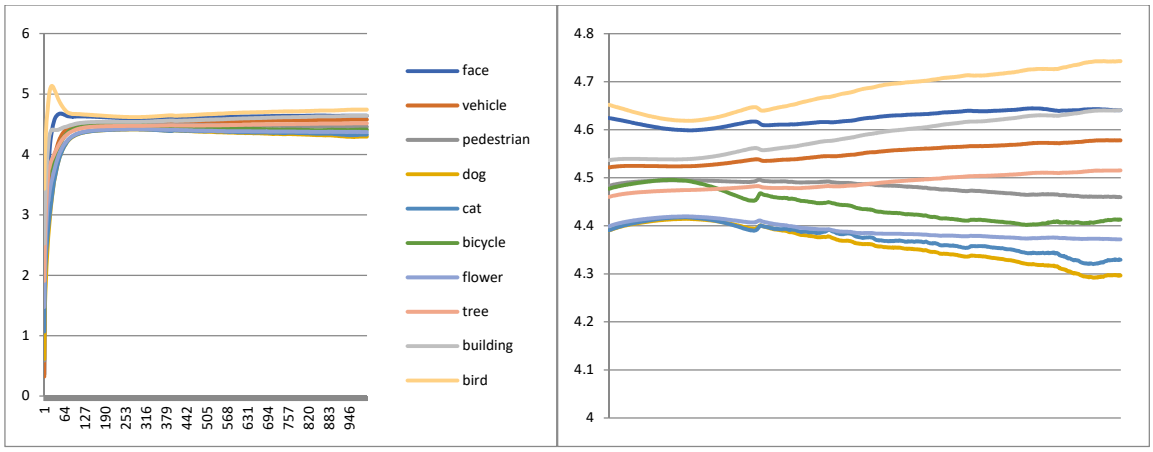


Figure 4.8: Label Trend with 10 Categories

The table on the left side (Figure 4.8) shows general label tendency for 10 categories within 1000 training loops. Labels of all categories converge to the value around 4.5 (the average value of labels) before loop 289 and then divergence in a slow speed. It seems that current network cannot distinct multi-classes. However, when we cut out and amplify tendency map between 289 and 1000 (chart on the right side), we found that the label value of each category still alters and the distance between each label enlarges gradually. There are two hypothesises based on experiment results:

- Training is not complete within 1000 loops. As has been explained as a conclusion of experiment 4.1.1.2, the increment of training samples will dramatically increase the training burden. In this case, because the total training samples increase from 10 to 100, it is reasonable that more training loops are necessary.

- Width of layer as well as depth of network are not enough to distinguish 10 categories (i.e. the network has no ability for at least 10-class discrimination). We only have one convolutional layer with 12 channels in the network. For a sophisticated network such as GoogLe-Net, much deeper structure is applied. Even the simplest CNN structure (Le-Net) has 5 layers for low-resolution handwriting recognition. From the above, network may be too simple to recognize complex objects which may contain many features and noises.

According to emanative label tendency, it is more likely that the network still works for 10 categories recognition, but it requires longer training progress (hypothesis 1). Notice that we don't plan to increase network depth because it will violate our original intention. But we do consider extending conv layer width once the first hypothesis is proven as false. A back-up plan is to insert more 3*3, 5*5, and 7*7 kernels separately for each color channel.

We increase the loop number from 1000 to 5000 aiming at generating a stable network. It is worth to know that all labels float up and down at around 4.5 (mean value of assigned labels) while likelihoods of other networks, such as VGG, are clearly separated. In our network, the output of each element represents spectrum of all categories. It is inevitable that each category is dependent to each other. Besides, we don't know the exact distribution of labels that should be assigned as initialization if there exist one. In our experiment, we just manually assign labels to different classes in the order of magnitude. Arbitrarily assigning labels could extend training process: it is possible that assigned labels is far away from ground truth, so more time is necessary to adjust parameters.

The next experiment is to explore relationship between label distribution and training iteration cost. To remove mean-influence, we start by manually assign values from -5 to 5, whose average value is 0. So the output labels should distribute around 0. We remain configuration of previous experiment to ensure no other parameters varying. Notice that, because activation functions, such as sigmoid and ReLU, filter out negative values from feature map, it is impossible to represent negative labels with the current network. In this experiment, we should remove activation functions or modified them to output negative values.

Figure 4.9 is the label statistics of 5000 iterations' training. The entire training process can be separated into four periods: convergence period (1^{st}-165^{th} loop), temporarily stable period (165^{th}-657^{th} loop), reconstruction period (165^{th}-2133^{th} loop), and final stable period (after 2133^{th}

loop). Generally speaking, all labels are with positive numbers even we remove activation function. The largest gap between different classes is fewer than 0.2, which happens at the first loop. In the second phase, the largest gap shrinks to 0.03. Labels of bird and bicycle classes are overlapped together. Similarly, tree and flower classes also share the same label. In the third phase, all labels shuffle. Many classes intersect together with an average gap increasing. In the last phase, all labels stabilize with labels converging between 0 and 0.1. The maximum gap is around 0.07 between building and dog. According to the tendency gap of this phase, we can separate the classes into three groups: first group contains only bird; second contains face, pedestrian, tree, vehicle and building; the last group contains cat, flower, dog, and bicycle.

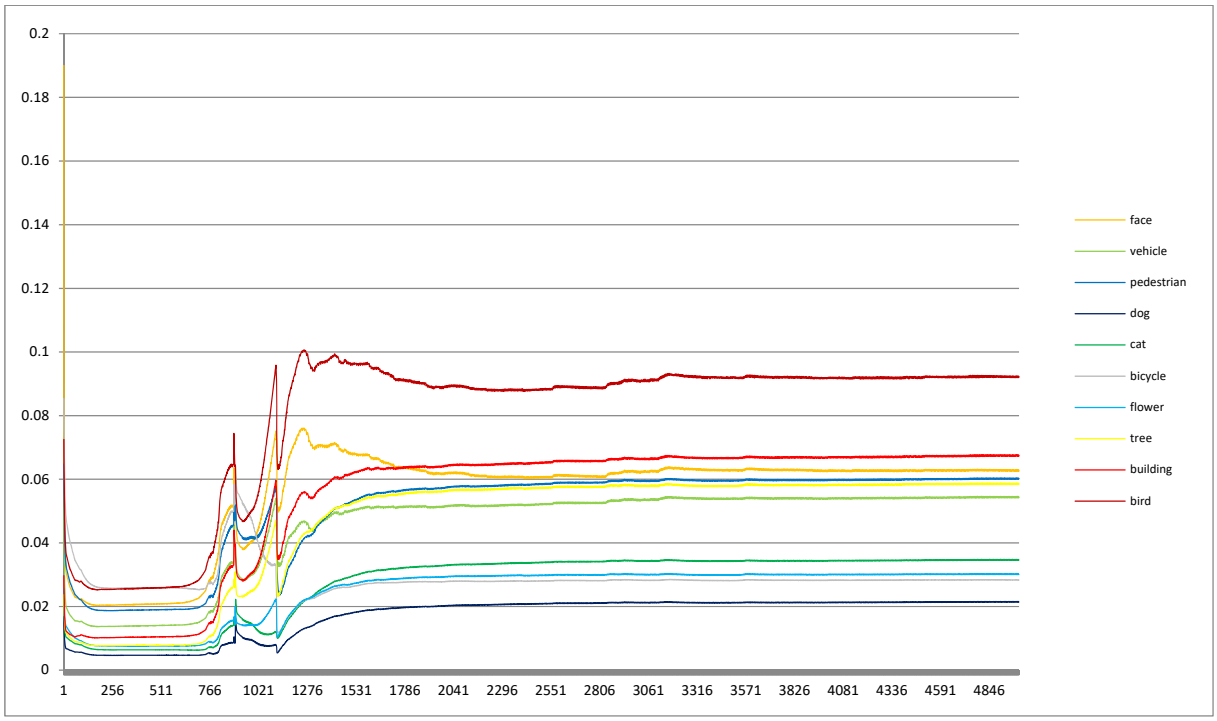


Figure 4.9: Label Trend with assigned Labels [-5, 5]

The stable phrase of Figure 4.9 clearly demonstrates that the label of each category has a clear boundary even though the distance is small. The stable line for this network with 100 samples is after 3281 loops. We notice that when labeling become stable, the order of the value is totally different compared to what we assign:

Table 4.1: Label Comparison Before and After Training

	face	vehicle	pedestrian	dog	cat	bicycle	flower	tree	building	bird
start	-5	-4	-3	-2	-1	1	2	3	4	5
end	0.062686	0.054331	0.060163	0.021446	0.034629	0.028344	0.030119	0.058541	0.067424	0.092215

We also randomly change assigned labels for each category, and make sure the average value is still 0. However, when the ground truth labels are {4, -4, 5, -2, -3, -5, 2, 3, 1, -1}, AEDs are all 0. We found zero-matrix problem which also happened in experiment 4.1.1.1. But this time, all biases are regular while weights are all negative. Large distance between ground truth labels causes the problem. For instance, the maximum gap of labels is 9 between vehicle and pedestrian.

Notice that the distance of output labels between each category is too small. Our next goal is to extend the difference to make it easier for classifying. In our network, because the label also can be negative, we define noises of output to any value that is out of label range. So, we modify the ReLU activation function according to labeling design. In this case, our activation function is:

$$f(x) = \begin{cases} x, & \min(\text{label}) \le x \le \max(\text{label}) \\ 0, & \text{otherwise} \end{cases} \qquad 4.2.2$$

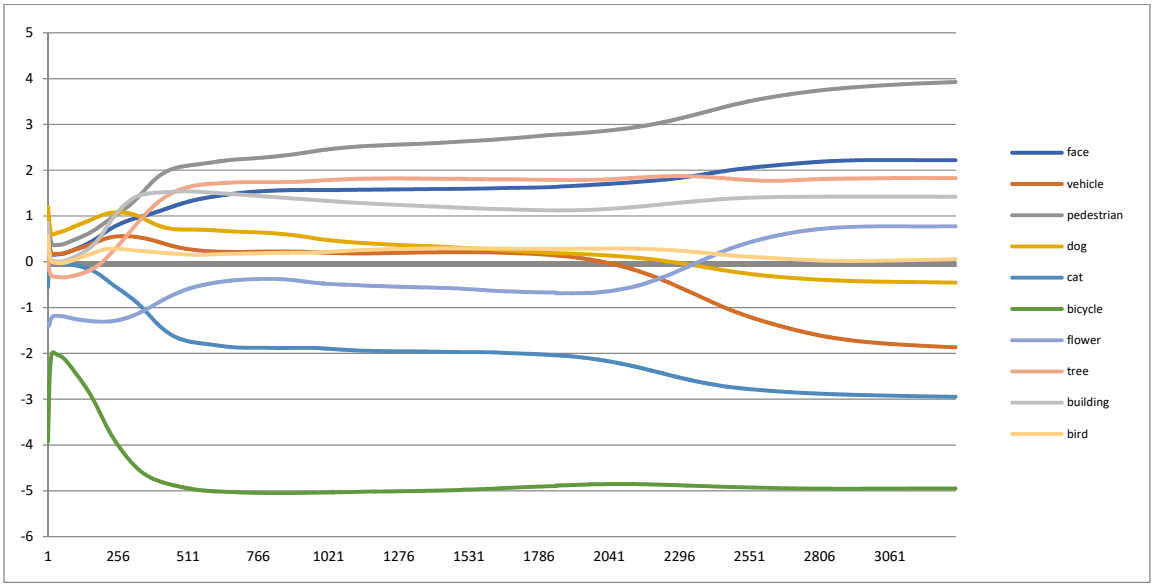


Figure 4.10: Label Trend with modified ReLU

Line chart above shows the result of modified network. Different from Figure 4.9 which contains 4 phases, Figure 4.10 is more complex. The range of output labels is stretched to [-5, 4]. Refining stage ends at loop 628. In this stage, except for bicycle, all the other categories have intersections meaning the initial label is not in the correct orientation. Second stage ends at 1882th loop. Although labels in this stage are relatively stable, vehicle, dog and bird classes restrain together. The third stage (1882 to 2509 loops) is mainly for vehicle, dog, bird, face, tree and flower classes refining. In the last stage, all categories have no cross labels.

However, there still exist categories, such as pedestrian and bird that have very similar labels. We think this caused by selection of target labels. There should be a heavy influence of choosing target labels for classification and training speed. Compared to Figure 4.9, label order of Figure 4.10 is totally different, which proves that no constant label sequence is found during these two experiments. However, initialization of ground truth label heavily influences training time cost (3281 loops for first network and 2965 loops for modified network). According to previous analysis, we import two conditions in the training process: 1) all assigned labels are positive; 2) the distance between nearby labels is fewer than 2. We use label {1,2,3,4,5,6,7,8,9,10} to represent each category which meets with the condition, and get the result as below:

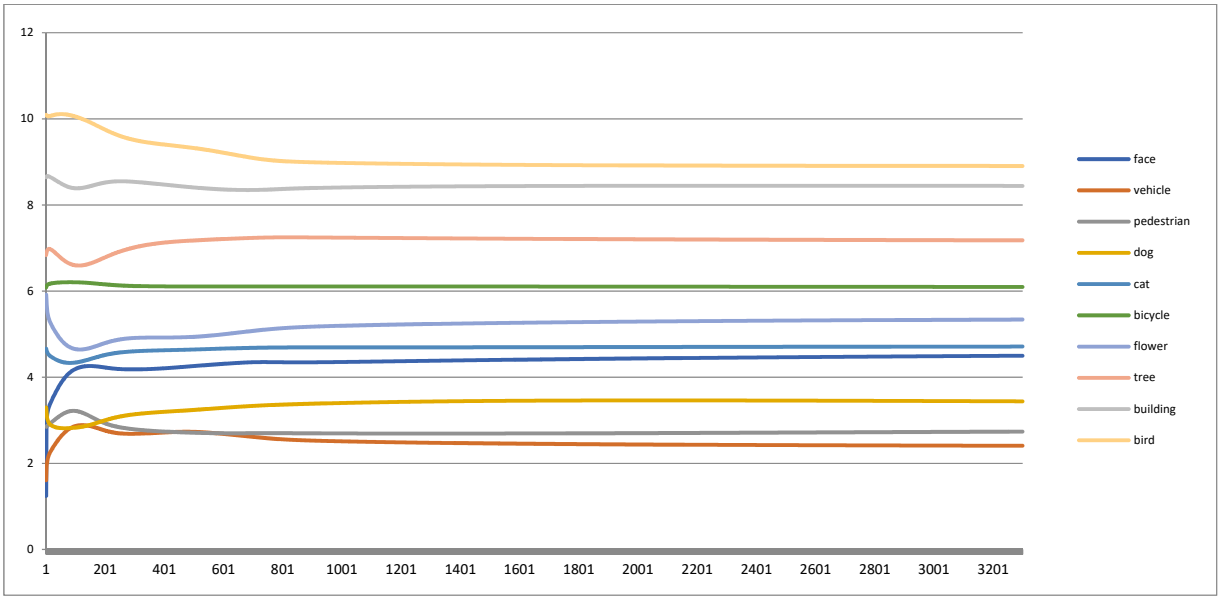


Figure 4.11: Label Trend with Assigned Label from 1 to 10

Figure 4.11 shows result with new label vector with original ReLU version. This time, the process can be separated to only two stages: 1) refining stage (from 1 to 865); 2) stable stage (from 866 to 3300). Compared to Figure 4.10, the addition of conditions makes processing more efficiency and distinguishable with fewer training iterations. All labels are allocated between 2 and 9 with relatively large gap while label distribution is still disparate. The result demonstrates that positive labels work well that all categories have a relatively large distance of AED. We believe that the choose of assigned label largely influence output label.

4.1.1.4. Varience of input size.

One advantage of our network is non-restriction of input size. As long as the size of input is the same as the scanning window for mean value computation (i.e. meet with consistency condition), objects can be labeled successfully. Because of this property, our network can directly recognize real scene images of various resolutions without region proposals.

In this experiment, we use the pedestrian dataset to train the network. Since our network has ability for 32*32 image recognition, we need to test whether it can also classify 32*60 samples. We label pedestrian class as 4 and set training iterations as 500, considering single class recognition with small training samples. The network using same settings as in experiment 4.1.1.2.

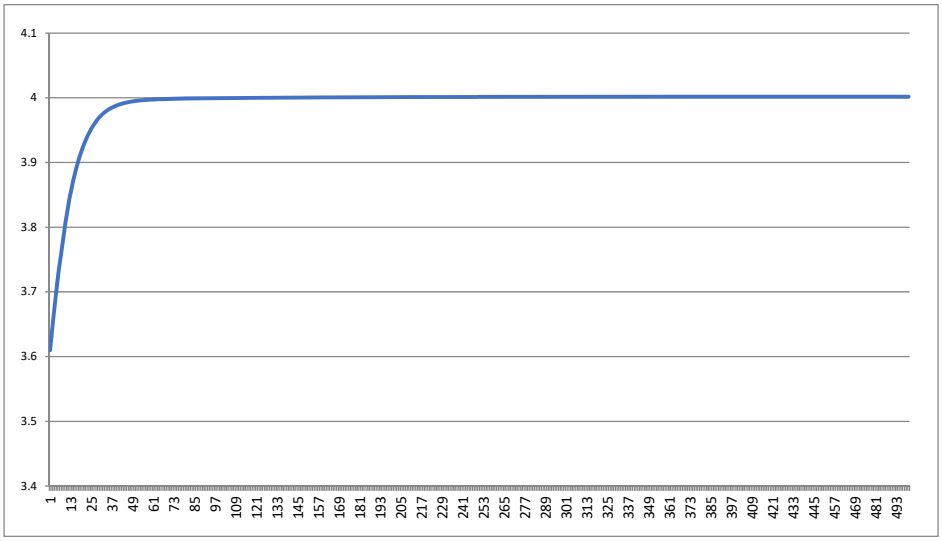


Figure 4.11: Label Trend with 32*60 Training Samples

Figure 4.11 shows the label value tendency for each learning iteration with 32*60 training samples. X-axis refers to iteration index and Y-axis is predicted label value. It starts at around 3.6 and logarithmically increases before 21th iteration. Then the growth of label value decreases gradually until 81th loop, after which label values stabilize at 4.

The tendency is the same as using 32*32 training samples (see Figure 4.7). With different size of input for training, the network still correctly label objects. The simplest network can be regarded as combination of many edge detectors followed by a linear integration. One advantage of our network compared to edge detectors is that we don't have to assign parameter values manually. All parameters are adjusted by BP algorithm that only demands reasonable initialization

and learning rate. Another advantage is that our network can recognize more complex objects other than the edges, owing to multi-channel structure. Feature is not confined to points, lines and corners. Lavish feature types and their combinations allow the net to recognize more complex objects.

4.1.1.5. Accuracy test.

We start with training group size of 100 (50 for vehicle and 50 for face) and testing group size of 200 (100 for vehicle and 100 for face) for 7000 epochs. To enlarge AED between these two classes without causing zero-matrix problem, we assign the label 1 to vehicle and 3 to face. We print labels in each epoch of the stable stage. The boundary value of two classes is calculated by:

$$V_b = \frac{1}{2}(Avg_{train}(face) + Avg_{train}(vehicle)) \tag{4.3}$$

Label tendency chart is shown as below (blue line represents face and orange represents vehicle):

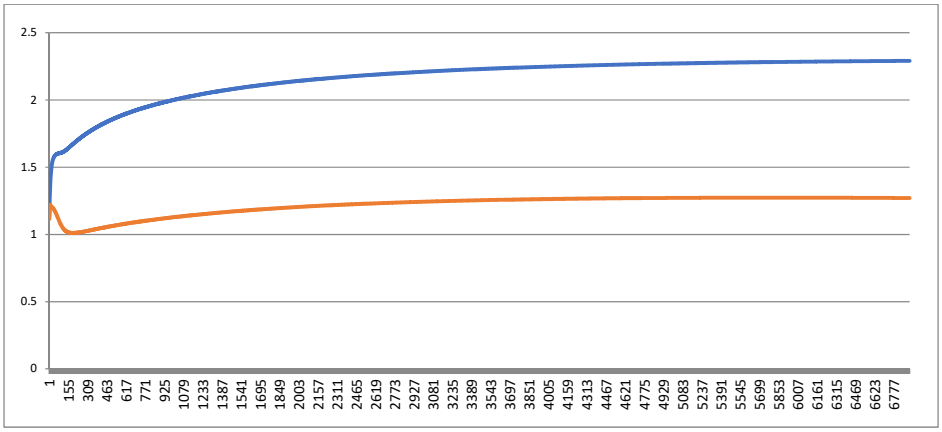


Figure 4.12: Label Trend of Face and Vehicle Class

X-axis shows the index of learning loop and Y-axis represents average label number from output layer. The label of the face starts at around 1.2 and dramatically increases before 346th loop. After that, the increase rate reduced smoothly until 6211th loop. The final average label for face is approximately 2.3. As for tendency of vehicle label, it has four stages. In stage one (before 100th loop), the label number increases dramatically to the peak (1.2). In stage two, number drop down to the valley floor, which is 1 at around 257th loop. In the following stage, number rises slightly and smoothly until 4486th loop. In the final stage, the tendency becomes stable allocating at 1.25.

We select data between 5521th and 7000th iterations to evaluate average accuracy, because labels during this period are stable for both classes. We also select the maximum label value of training sample as the upper boundary of face class and minimum label value as the lower boundary of vehicle class. Meanings of acronym can be found in Abbreviation section.

Table 4.2: Specific Parameters of Network between Iteration 5521 and 7000

Iteration Range	AFN	AVN	RoF	RoV
5521-7000	2.286382	1.273208	1.78-3.12	0.47-1.78
Mean Distance	AFT	AVT	Mean Accuracy	
1.013174	0.82	0.8	0.81	

Prediction convergence of vehicle class is much faster than that of face class. With same image size, most face samples contain large face area (IOU>0.7) while IOU for vehicle is 0.4 on average. Because of this difference, we believe face images contain more complex features than vehicle samples that need more time for modulation. Another conclusion can be obtained: with more features for training, practical object detection has higher accuracy (see AFT and AVT on Table 4.2). We also notice that initial labels only provide general adjustment orientation for machine learning. When the network become stable, outputs are not exactly the same as we assigned. For other networks, their outputs are restricted between [0, 1] as a percentage. They select top-1 or top-5 most likelihood class classifying objects. So, writers don't report the specific percentage value of each class.

Inconformity of labels may be caused by the simplicity of network structure. Not same as other delicate networks [28, 30, 31, 76, 95] who have deep layer structures to transform shallow features into more abstract ones that can reflect the global property of the image, our network has only one conv layer. Features are more specific and localized, making network hard to reach to the target. By using trained average labels instead of assigned labels, this network still achieves the object recognition task. We have received an accuracy rate of 82% for face recognition and 80% for vehicle. One advantage of our network is the small amount of required training iterations. Our network only needs fewer than 6,000 iterations while most sophisticated networks need more than 20,000 epochs.

4.1.2. Plural Objects Recognition.

The final goal of our network is to recognize multiple objects within one image. In the previous experiments, we have proved our network has the ability to distinct single objects of different categories. In the following experiments, we aim at demonstrating that the network also works for plural object detection without region proposals.

4.1.2.1. Categories labeling.

We generate 10 600*800 test images (pure-color background), each of which consists of 10 random objects from test samples. Objects are selected from vehicle and face samples. The positions of these objects are also random (all objects are not beyond the image boundaries) without overlapping region. We utilize configurations and parameters of experiment 4.1.1.5 for object detection. On Figure 4.13, red pixels refer to face class and blue ones refer to vehicle.

We manually evaluate accuracy based on two standards: 1) whether objects are correctly labeled; 2) whether indicated objects are in the correct positions. Because the output of networks is real numbers that don't directly indicate class labels, it also needs to be multiplied by 32*32 mean matrix (explained in 4.1.1.1). The outputs of ten 600*800 images are given in appendix.

Notice that the mean values are not the only condition indicating category. There is a high change that the mean value of background has the same value as vehicle or face label. An example is shown as below:

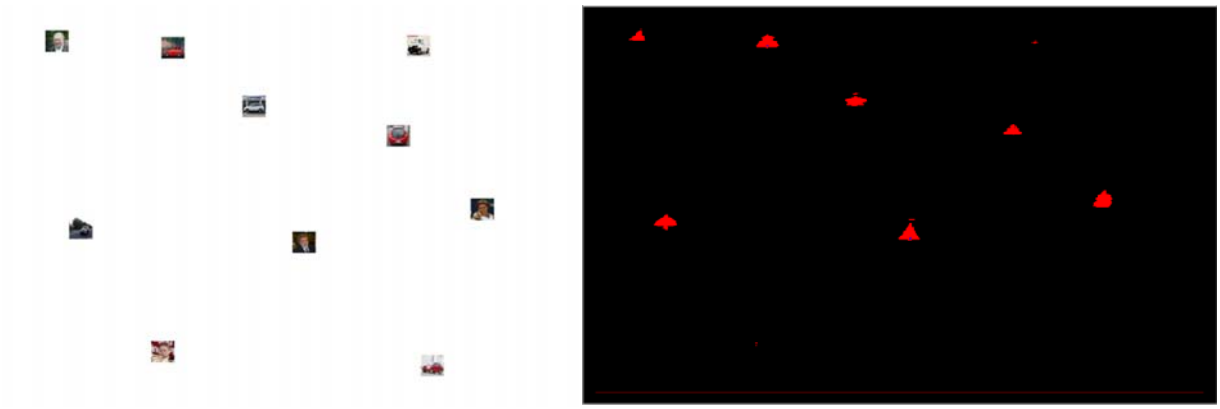


Figure 4.13: Original Image with Plural Objects (left) and Detected Label Map (right)

Map on the left side is the test sample while on the right side is the output. It seems that all objects are indicated with red color. A red line is printed on the down size, which contains nothing in the original image. Actually, there exist objects with correct labels. For example, we amplify the output of the second vehicle region of left-up corner as Figure 4.14:

Figure 4.14: Amplifying Label Map of Vehicle of the Left-up Side

The correct label (blue pixel) is surrounded by the wrong ones (red region). We believe that only with the mean map with a 32*32 kernel is not enough. All the wrong labels are caused by the mix of background and object region. As a vehicle region, most values of output should be lower than 1.78. However, except for center pixel, all the other pixels contain background pixels which have values of 255. Background increases mean value of this region. So, there exist lots of pixels encircling the center point with mean values in the range of [1.78, 3.12] which belongs to face. This problem also happens on the edge of the image (see the red line on the output map).

We notice that although the output map cannot directly indicate class, object pixels obey specific distribution as shown on Figure 4.3. We compute all pixel values (in total 102,400 pixels) of 100 training samples and generate the histogram. Most pixel values regardless of sort distribute between 0 and 4 (44.3% of pixels). According to this property, we consider value distribution of a window region as the second condition. For each proposal region, if the number of pixels with value between 0 and 4 is greater than 454 (44.3% of entire region), we consider it as a candidate.

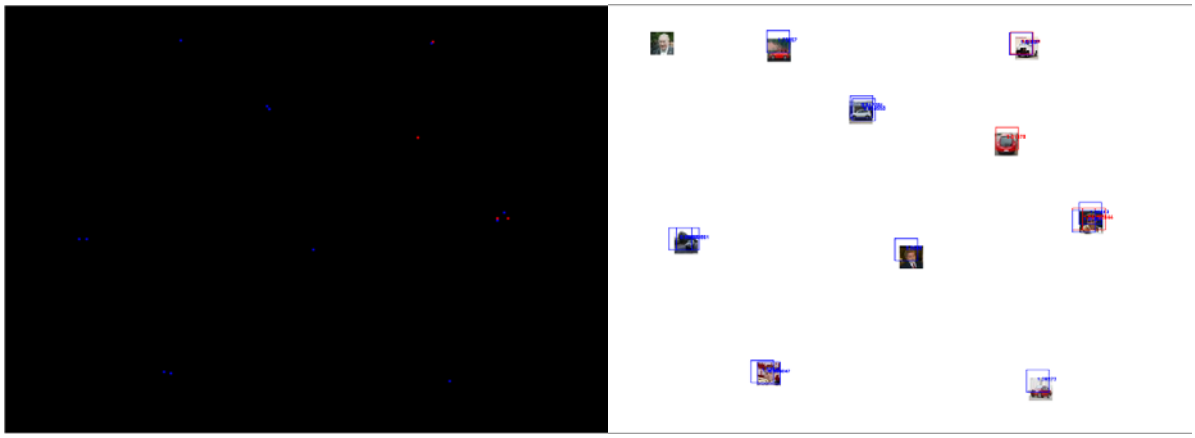


Figure 4.15: Label Map Applying Second Condition (left) and Actual Detection Map (right)

The output has a significant improvement after import second condition. From the left picture, influence of background is significantly reduced. Only pixels in the center area are indicated. General speaking, our proposed network has ability to distinct multiple objects on the same image with improved method. However, boxing fine-tuning techniques such as Non-Maximum Suppression and Bounding-box Regression should be further applied.

4.1.2.2. Accuracy test.

In this experiment, we prepared 10 600*800 test samples combined with total 10 objects of 32*32 test samples. We utilize png. type image instead of jpg. In jpg. images, the original data has to be compressed to save space. This transformation process causes information losing, especially on edge regions, which seriously reduces detection accuracy. The generation of test samples is the same as experiment 4.1.2.1. Each large sample contains at least one face and one vehicle object. The image is pre-processed as mentioned in Chapter 3.2 to narrow down value range. We evaluate the multiple object detection ability by counting how many objects are labeled on the image and how many of them have correct labels.

Table 4.3: Accuracy in Percentage of Plural Object Detection

	Img1	Img2	Img3	Img4	Img5	Img6	Img7	Img8	Img9	Img10
NoI	9	9	8	7	10	9	7	7	9	8
NoL	6	6	6	4	8	7	6	5	6	7
Perc	66.7%	66.7%	75.0%	57.1%	80.0%	77.8%	85.7%	71.4%	66.7%	87.5%

Table 4.3 illustrates testing accuracy for each image. The average NoI is 8.3 (i.e. for each image, 8.3 objects can be indicated with correct location regardless of their category assignment). Within the indicated objects, the average NoL is 6.1. In most images, 6 objects are correctly indicated with both position and category. Only the 4th image has recognized object fewer than a half. We show sample 4 in figure 4.16:

Two face regions and one vehicle region are not detected. Miss-indicated face region with black background has unexpected feature: helmet. All faces of training samples don't have decorations such as glass or hat. Unexpected features serious impede recognition. Similarly, vehicle region shows only the back of a vehicle, while vehicle training samples mainly focus on frontal region. The network lacks relative feature kernels. As for the second face region, there

appears three faces. Two of them are partial occlusion. These fragmentary face features prevent the real face region from being detected. Except for partial occlusion and object fragmentary problem that we cannot deal with temporally, we can increase the detecting rate by enlarging training samples.

Figure 4.16: Actual Detection Map of Sample 4

Notice that there are four indicated regions labeled with both vehicle and face. We believe this problem caused by insufficient filter strategies. The influence of edges still exists. However, we find all these four regions contain more correct label than incorrect one that proves our network has ability for multi-object detection. We can utilize non maximum suppression algorithm [8] to remove incorrect and redundant labels.

To make a short conclusion, our crude network has ability for multi-objects (in average 83% of objects) detection with pure background. Limited by quantity of training samples, the network cannot distinct overlapped or incomplete objects. From those indicated objects, correctly labeled objects occupy 73.5%. As for detection efficiency, the network costs 0.348 second for one 600*800 image. Although this time cost is not enough for real-time simulation, we can allocate channels to different CPUs or GPUs to have parallel computation. Since the most time consuming

is with convolutional operation, in the improvement section, we proposed several solutions to improve operating efficiency.

4.1.3 Reliability on Scale and Illumination Variance.

In the real scene, because of environmental conditions (e.g. illumination strength), object status (e.g. movements), and observer elements (e.g. point of view), same objects have various 2-d mappings. A naive method to improve recognition capacity is to significantly enlarge training samples. However, this needs lots of manual efforts while training such huge dataset is time consuming. We cannot list all cases even with the completest dataset. Further more, since the structure of our network is simple, we cannot avoid under-fitting problem through large-scale samples. Another method is to reinforce the anti-deformation ability to reduce training samples in all cases.

In most CNN related research, scholars didn't mention networks' performance towards object deformation. All their experiments are based on the well-modified samples. In the real application, all the interferential elements need to be considered to increase accuracy. We consider anti-interference ability mainly based on scaling and illumination changing. Both reliability tests are based on single object recognition.

4.1.3.1 Scaling.

In this experiment, we consider three cases: 1) width scaling; 2) height scaling; 3) width and height scaling at the same time. Because our training and test samples are in size of 32*32, the scaling levels are separated into 16. To meet with pooling layer condition (matrix size should be even), width or height of each level will vary by 2 pixels. We utilize test samples from vehicle and face groups and generate their corresponding re-scaled images. In total, we have 3400 samples for this test.

Firstly, we resize the width and height of image in the same rate and record average accuracy rate for each level. The result is shown in figure 4.17. X-axis indicates image size. ES refers to equal scaling; VW refers to varying width; VH refers to varying height. Y-axis is the average accuracy rate computed from 3400 samples. We define 0.9 accuracy rate as acceptable boundary of recognition. It is obvious that the size rescaling below 24*24 (0.75 size of original

image) of ES dramatically reduces accuracy rate. In contrast, the increment of its size only slightly reduces accuracy rate. Even for 48*48 (1.5 size of original image), the accuracy rate still remains beyond 0.9. As for VW, 18*32 (56.28%) is the lower boundary and 38*32 (118.75%) is the higher boundary for our network. The lower limit for HW is 32*28 (87.5%) while the higher limit is 32*44 (137.5%). From data, we can infer that for both ES and HW, the network is solid for image amplification while for VW, network is solid for image shrinking.

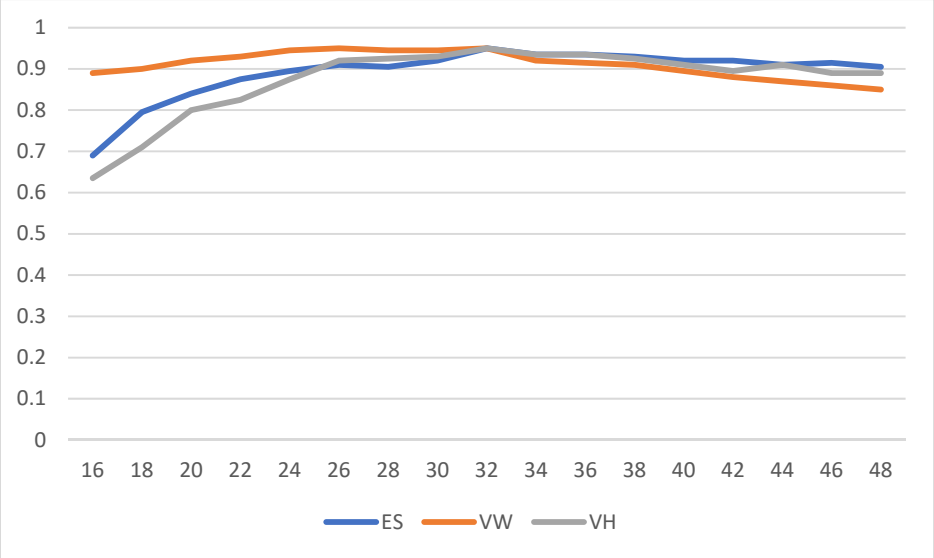


Figure 4.17: Accuracy Trend with Different Scales

The result proves that our network has anti-scaling ability whose range in general is between 0.75 and 1.5 times of training samples in size. We also notice that the network is relatively stable to recognize larger images. To have an efficient training process as well as keeping accuracy rate, we can use small size samples instead of larger ones. Using large size samples will: 1) dramatically increase training times. The time cost increases exponentially. 2) increase the chance to ignore small size object recognition.

4.1.3.2 Illumination.

Illumination is another factor affecting the recognition rate. The ideal network can recognize objects in somber and exposure situations. However, for most recognition systems, local and global illumination changing will significantly reduce detecting rate. Objects in a very low illumination, with different color of lights or partially under shallow are hard to recognize. In this

experiment, we linearly change the brightness of test samples to see whether our network is light invariant. The function that control image brightness and contrast is:

$$g(i,j) = \alpha f(i,j) + \beta \tag{4.4}$$

while α is the parameter to control contrast, β is to control brightness. In our experiment, we assign α as 1 to ignore image contrast change, because in the real situation, the influence of brightness to an image is much more frequent than contrast. We separate brightness into 101 levels. Level 0 represents pixels (0, 0, 0) while Level 100 represents pixels with value (255, 255, 255). Value β is computed by:

$$\beta = 5.1 * Level - 255 \tag{4.5}$$

To ensure transformed values are constrained within [0, 255], we also need to restrict its value:

$$g(i,j) = min(max(f(i,j) + \beta, 0), 255) \tag{4.6}$$

We randomly choose 100 samples from test groups and generate different level images. In total, we generate 10100 samples. We compute accuracy for each level:

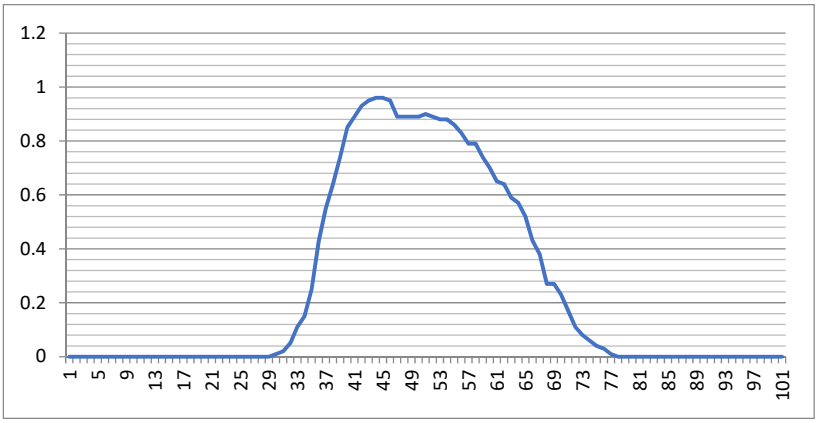


Figure 4.18: Accuracy Trend with Different Brightness Levels

The table (Figure 4.18) shows average accuracy rate for different brightness level. We still define 0.9 as the acceptable accuracy rate. The accuracy keeps beyond 0.9 between level 40 and 55 (level 50 is original training sample brightness), which proves that the network is relatively steady with slight illumination change ($\pm$ 10% variance). To deal with extreme brightness

changing, one way is to extend training groups of different brightness, which has the same problem as 4.1.3.1. Another way is to apply image enforcement techniques to improve image quality.

4.2. Improvements

In previous experiments, our crude network achieves face and vehicle recognition task. We need to improve its performance in both accuracy and processing speed. In this section, we focus on several standards to refine network: selection of activation functions, pooling layers, kernel flipping operation, and application of DFT.

4.2.1. Convolutional Operation.

We found our network has a very low processing efficiency during experiments. In OpenCV, images are stored in a form of "Mat". The most common way to read each element in Mat is to call "Mat.at<type>(x, y)" function. However, this function costs much time because the compiler has to load pointers of each pixel one at a time, especially for a large image (600*800). To avoid redundantly invoking the pointer loading operation, we replace the function by "Mat.ptr<type>(x)". Instead of loading data in specific position one at a time, this function loads the pointer of entire row. During image processing, number of calling pointer loading function is reduced from $w*h$ to h (w represents the width of image while h refers to the height). We can also import DFT algorithm to have a further improvement (similar to phase-only correlation [91]). As analyzed above, we propose four strategies for convolutional operations:

1) Call "filter2D()" function to apply DFT algorithm
2) Call "Mat.at<type>(x, y)" to apply conv operation, which cost O(W*H*w*h) computations for pointer loading. (W, H) represents the width and height of image dividedly where as (w, h) represents the kernel size.
3) Call "Mat.ptr<type>(x)" for image and kernel scanning, costing O(H*h) computations for pointer loading.
4) Transform "Mat" to array by "Mat.ptr<type>(x)", compute convolutional operation in array and transform back to Mat after computation, costing O(2H+2h)

The aim of this experiment is to figure out whether DFT can significantly fasten processing compared to other strategies. The experiment is based on convolutional operations with four different kernels (1*1, 3*3, 5*5, 7*7). To increase the time cost for each operation so that obvious difference can be obtained, each operation is repeated for 10 times. To accurately imitate real image, we use 600*800 random matrix.

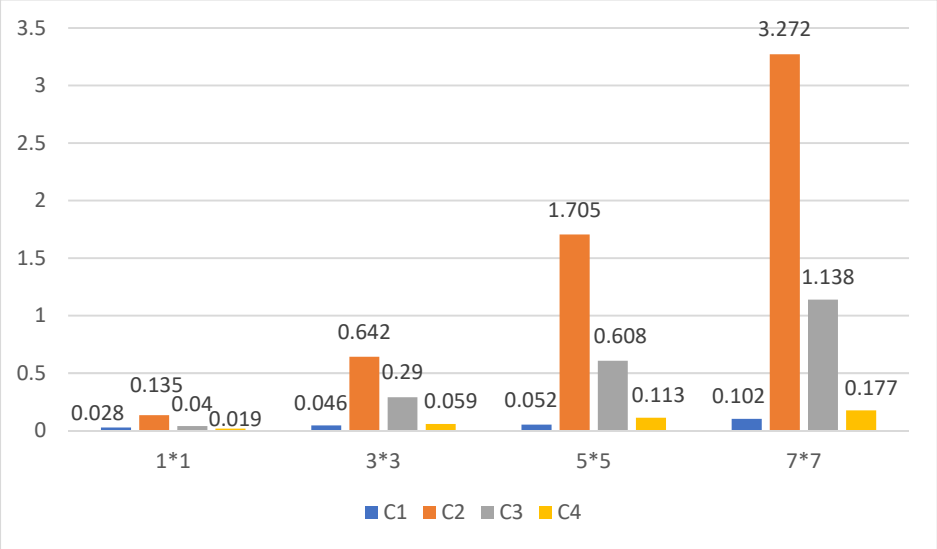


Figure 4.19: Time Cost for Different Conv Strategies

Y-axis represents the time cost whose unit is second. X-axis represents the size of kernels. C1 represents operations with DFT algorithm. C2 represents the one calling “Mat.at<type>(x, y)” function. C3 calls “Mat.ptr<type>(x)” function. C4 contains image transformation from “Mat” to array.

From Figure 4.19, we can conclude that although all versions have time cost growth with size of kernel increasing, C1 increase slightly and linearly while C2 and C3 increases exponentially. C4 costs the least time for 1*1 kernel while C1 has the best performance with all the other kernels. We notice that when deal with 1*1 kernel, the efficiency of transformation between space domain and frequency (Fourier) domain is time-consuming compared to the transformation between Mat and array data type. With high dimensional kernel, DFT is much faster than strategy 4, because kernel scanning in the large image is slow compared to directly multiplying two values in frequency domain.

From the table, we can clearly notice that DFT only cost 0.102 second for one operation with 7*7 kernel. According to the analysis, to have the best improvement in operation efficiency, we apply strategy 4 for the 1*1 kernel operations and strategy 1 for the rest. We also record the time cost for recognition processing before and after improvement:

Table 4.4: Time Cost Comparison for Network Before and After Improvement

	Original	Improved
Time cost (s)	15.807	0.575

During the recognition processing, our network needs 24 convolutional operations (each conv channel needs four operations for both forward and backward computation). The output layer also need 12 operations with 1*1 kernels. Network with new convolutional operation strategy is 27 times faster than original one.

4.2.2. Activation Functions.

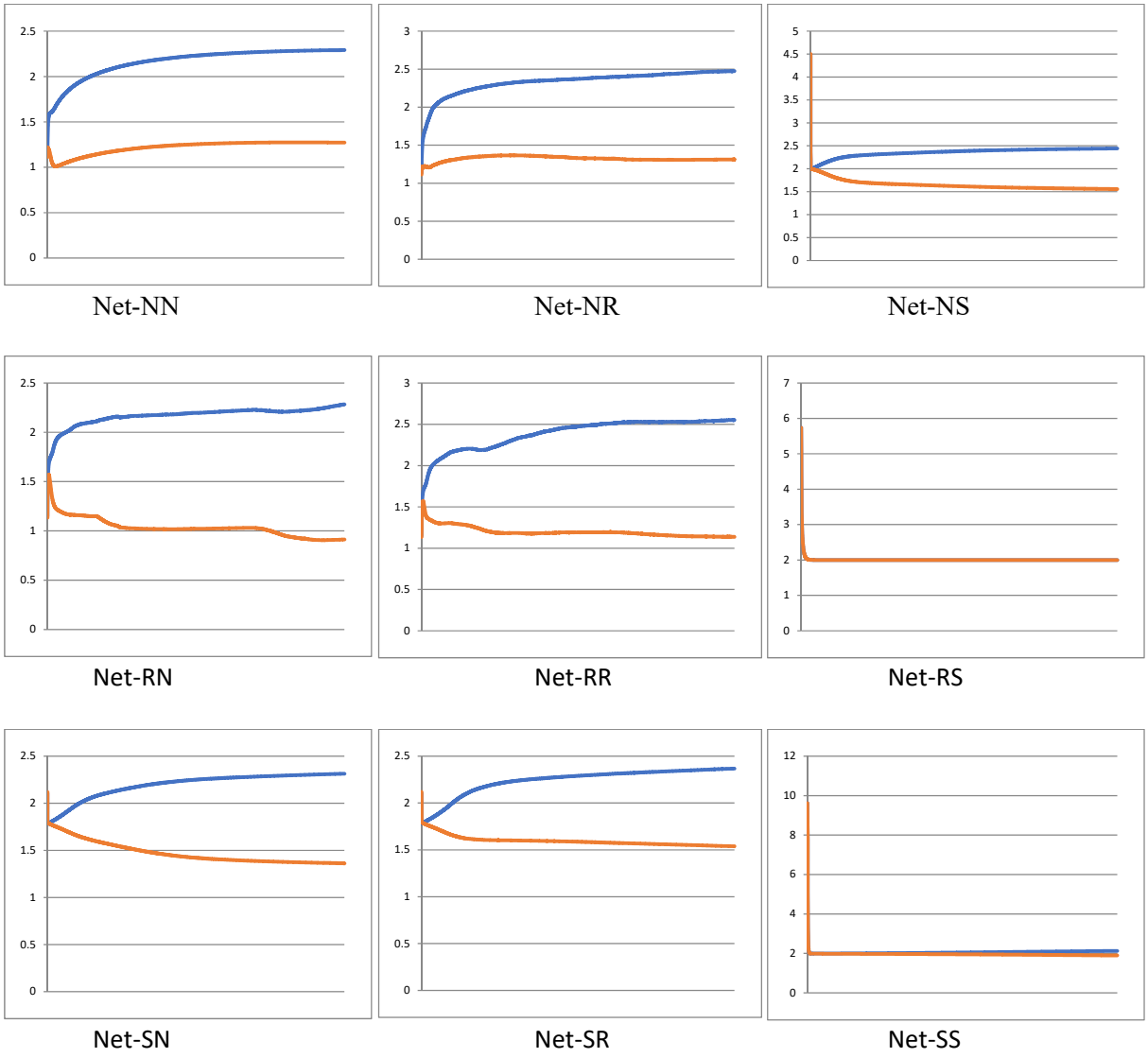
Activation function choosing is one of crucial factors for classification. In this experiment, we focus on two most popular activation functions: Sigmoid and ReLU to improve our network. Because the role of activation function is to help solve non-linear division problem, those activation functions may help increase accuracy rate. Since our network has two activation layers, we derive nine network versions according to the combination arrangement of activation functions:

Table 4.6: Table of Applied Activation Function toward Corresponding Networks

	Net-NN	Net-NR	Net-NS	Net-RN	Net-RR	Net-RS	Net-SN	Net-SR	Net-SS
L1	None	None	None	ReLU	ReLU	ReLU	Sigmoid	Sigmoid	Sigmoid
L2	None	ReLU	Sigmoid	None	ReLU	Sigmoid	None	ReLU	Sigmoid

L1 refers to the activation layer between convolution layer and output layer while L2 refers to the one right after output layer. None means no activation function is applied in correspond layer. We make experiment to test their separation ability based on face and vehicle samples. Same as previous experiments, we set the iteration number to 7000 with 100 training samples. We print average output tendency for each loop. The results are shown as below (blue line refers to face and orange line refers to vehicle):

In the experiment, we initialize face label as 3 and vehicle as 1 to have a relatively large distance between each class. Because sigmoid function will map any value into [0,1] in a non-linear way, we have to modify labels to those networks (Net-NS, Net-RS, Net-SN, Net-SR, Net-SS) as 0.3 for face and 0.1 for vehicle. Otherwise, the differences of oversized values (i.e. $x \in (-\infty, 0] \cap [1, +\infty)$) in error map will be shrunk in percentage, which will cause distortion. If both class labels are out of limitation, network cannot distinct them, because the larger the label is, the smaller the difference is. To have the same scale as other networks, labeling result will be multiplied by 10. X-axis represents iteration number from 0 to 7700.



From the result, except for Net-RS and Net-SS, all the other networks allocate labels at around 2, meaning they have ability to distinct training samples. Most labels are located between 1 and 2.5. We compute the average output labels, distance values of each category, and the boundary values in the stable iterations:

Table 4.7: Label Table of 9 Networks with 5 Classes

	Net-NN	Net-NR	Net-NS	Net-RN	Net-RR	Net-RS	Net-SN	Net-SR	Net-SS
Face	2.286382	2.464501	2.43331	2.467	2.540985	1.998623	2.30718	2.361812	2.118683
Vehicle	1.273208	1.31026	1.566315	0.911851	1.140883	1.998661	1.367863	1.543166	1.908309
SIR	5521	5951	5251	5951	5866	335	5866	6301	6013
AG	1.013174	1.154241	0.866995	1.334848	1.400103	0.00039	0.939317	0.818646	0.210374
BV	1.779795	1.88738	1.999812	1.579275	1.840934	1.998642	1.837521	1.958489	2.013496

SIR:Stable Iteration Range. AG: Average Gap. BV: Boundary Value of Two Categories

It is announced in SVM [15] that large class distance results strong distinctive ability. We separate networks into three groups according to class distance: group 1 (AG>1.0), group 2 (0.5<AG<1.0) and group 3 (AG<0.5).

No sigmoid function is applied in group 1 (Net-NN, Net-NR, Net-RN, Net-RR). In the second group (Net-NS, Net-SN, Net-SR), most networks applied sigmoid function in L1 while group 3 (Net-RS, Net-SS) have sigmoid function in L2. By comparing these three groups, we believe that sigmoid function has a serious restriction for parameter adjusting. The output of sigmoid function is a reflection from $N \in R$ to $N \in (0,1)$ where as derivative in back propagation step significant narrows down value difference of each pixel in sensitive map. Calculated sensitive maps have much smaller value than practical one. In our network, the output is the actual label value rather than possibility percentage, reduction of training sensibility are not available. As a result, parameter refining has slight changes causing it hard to get close to the target. In the third group, the negative influence of sigmoid function starts at the beginning of the backward process (computation starts from output layer), meaning the training tendency of category mixes up at the beginning of each loop. Compared to group 3, group 2 has relatively good performance in class distance, because the negative effect of sigmoid function happens in the second half training process. As a result, this negative effect reduces during training.

Only from the label distance, we get the conclusion that networks with the sigmoid function (either in the first layer or second layer) have the worse distinctable ability than others. Net-RS

even loss distinguish ability as ReLU is not compatible to sigmoid function when it is applied in L1.

We also exam different versions with test samples that are totally different to training samples. The process of experiment and listed parameters are the same as experiment 4.1.1.5:

Table 4.8: Accuracy Table of 9 Networks

	Net-NN	Net-NR	Net-NS	Net-RN	Net-RR	Net-RS	Net-SN	Net-SR	Net-SS
RoF	[1.78, 3.12]	[1.89, 3.67]	[2.00, 4.13]	[1.58, 3.06]	[1.84, 3.30]	[2.00, 2.01]	[1.84, 3.13]	[1.95, 3.60]	[2.01, 2.64]
RoV	[0.47, 1.78]	[0.62, 1.89]	[0.84, 2.00]	[0.00, 1.58]	[0.50, 1.84]	[1.99, 2.00]	[0.48, 1.84]	[0.77, 1.95]	[1.60, 2.01]
AFT	82.0%	86.0%	79.0%	88.5%	91.5%	0.0%	85.5%	84.5%	72.0%
AVT	80.0%	86.0%	85.0%	90.0%	91.0%	0.0%	82.0%	84.0%	67.0%
Avg Acc	81.0%	86.0%	82.0%	87.0%	91.3%	0.0%	83.8%	84.3%	69.5%

From the accuracy result, we found that network with ReLU for both activation layer has the highest accuracy rate (average 91.5%). Networks with sigmoid function all have accuracy under 85%. Net-RS doesn't have distinguish ability.

4.2.3 Pooling Layer.

Pooling layers have the ability to reduce the influence of noises as well as compress data dimension. We believe the appropriate pooling layer helps to improve accuracy rate. There are abundant noises are included in feature maps influencing detecting effort, because we don't have a deep structure to filter them. By applying pooling layer, we hope to keep prominent features and remove distractions.

Based on Net-RR, which has the highest accuracy rate for single object recognition in experiment 4.2.2, we applied two different pooling layers (max-pooling and mean-pooling) as they are currently the most popular and efficient strategies (see Chapter 3.1.3).

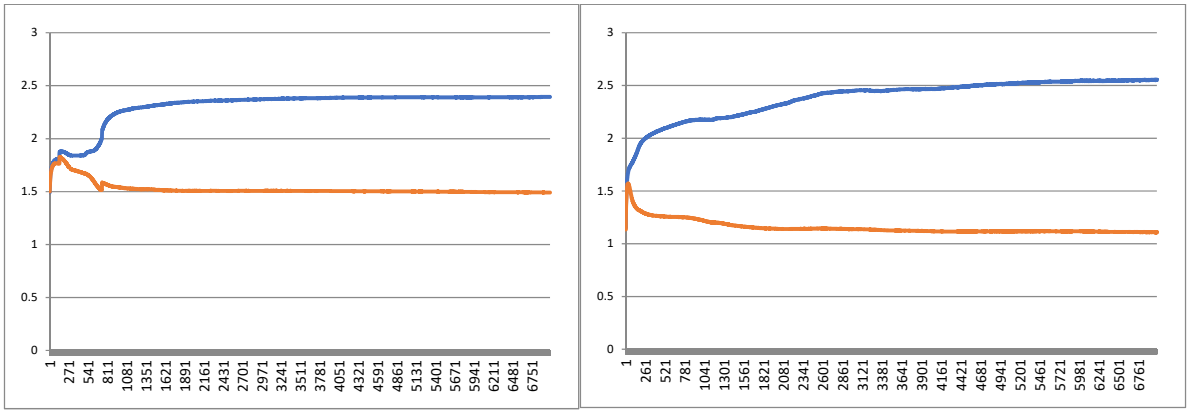


Figure 4.21: Label Trend with Max-Pooling (left) and Mean-Pooling (right)

Figure 4.21 are generated from Net-RR-Max and Net-RR-Mean which refer to network with max-pooling and mean-pooling layers separately. First of all, we can see from the table that the implementation of pooling layer doesn't remove distinctive ability. However, we notice that max pooling layer raise prediction value of vehicle from nearly 1.0 to 1.5 while keeping the label of face unchanged. This change, we believe, will have a negative effect on recognition accuracy, because the category range is narrowed (accuracy rate is proportional to category distance). On the country, mean pooling layer didn't significantly affect training tendency. Slightly, it enlarges the distance, which may have a better performance for recognition. We record accuracy test result for both networks:

Table 4.8: Parameters and Accuracy of Max-pooling Net and Mean-pooling Net

	Face	Vehicle	SIR	AG	RoF	RoV	AFT	AVT	Avg Acc
Max	2.390613	1.498125	4201	0.892488	[1.94, 3.61]	[0.96, 1.94]	74.0%	71.0%	72.5%
Mean	2.545284	1.114212	5601	1.431072	[1.83, 3.49]	[0.55, 1.83]	93.0%	97.0%	95.0%

The accuracy data proves our assumptions. Net-RR-Mean has the best performance with average accuracy of 95.0%. It has the highest accuracy rate in all aspects, compared to the original Net-RR. As what we analyzed, pooling layer removes features with less information. However, we are also curious about accuracy from Net-RR-Max. Max pooling layer even has negative influence on the network, regardless of its filtering function (Accuracy of Net-RR without pooling layer is 91.3% in experiment 4.2.2). One reason causing the problem is the shortage of features. As designed in chapter 3, max pooling layer removes 3/4 raw features from original feature maps. In the masterpieces, the deeper network has ability to abstract remaining features to make them

contains more information. However, since our network don't have such deep structure, the number of remained features don't provide enough information for classification. According to the strategy described in 3.1.3, mean-pooling layer keeps all raw features. It weakens the negative influence by smoothing noise values. So we can still have enough features for network training.

4.2.4 Plural Object Detection.

From functional tests, proposed network structure has the ability of detecting single and multiple objects. The performance of single object detection is satisfied while accuracy is still low for multiple detection. In this experiment, since Net-RR-Mean network has the highest accuracy rate for single object detection, we are curious whether it has the best performance in multiple object detection.

Table 4.9: Accuracy of Plural Object Detection with Mean-pooling Net.

	Img1	Img2	Img3	Img4	Img5	Img6	Img7	Img8	Img9	Img10
NoI	10	10	10	10	10	10	10	10	10	10
NoL	10	9	9	10	10	8	10	8	7	8
Perc	100.0%	90.0%	90.0%	100.0%	100.0%	80.0%	100.0%	80.0%	70.0%	80.0%

We re-exam the test 4.1.2, but this time Net-RR with the mean pooling layer is applied. We calculate the accuracy based on NoI and NoL. It is obvious that the object detecting rate is 100%. The average labeling rate is as high as 89%. The accuracy is high enough for actual usage. These two rates are much higher than Net-NN, which reflects Net-RR with mean pooling layer has the best performance on both single and multiple object detection. Notice that, even for the best network, its multiple object detecting rate is much lower than the single object rate. We attribute this phenomenon to the influence of real scene background.

4.3. Summary

In this chapter, we first prove our simple network has the ability of distinguishing single and multiple objects with mainly two classes (face and vehicle). As well, the extension of category experiments also proves that our network has ability to distinguish at least 10 different categories, which is enough for general traffic object detection. We discover that although training labels are not exactly the same as what we assigned as ground truth, predictions for each category are stable

and uncrossed, meaning with trained labels, different categories can still be recognized. Function of label assignment is to give a general orientation of category label distribution.

Two cost functions are proposed in the experiment. Because of output dimension, we use different evaluation methods to decide their performance. Both of them have separating capacity in some degree. However, function with mean matrix has a strong ability to distinguish single objects of different classes while the class distributions from another function have serious overlapped regions which makes it hard for classifying.

In accuracy test, we applied simplest network structure to test accuracy rate for both single object and multiple objects detection. For single object detection, we can directly utilize average values of the output matrix. For multiple object detection, we also need to filter irrelevant proposals. Accuracy rate for multiple object is lower than the single object detection because a large image contains too many irrelevant regions. It is almost impossible manually discovering principles to completely remove those regions, considering the change of the scene. What we can do is to reduce its negative influence as much as possible. Generally speaking, our proposed principle improves accuracy for multi-object detection. We believe there exists more principles to further improve accuracy.

In section 4.2, through experiment, we select ReLU as activation function and add mean pooling layer to help improve accuracy rate. In the section 4.1.3, we also test the resistance for image rescale and illumination change. The experiment shows that slight change of re-scale and illumination change will not significantly influence detection accuracy. We believe our network has an acceptable performance for single and multiple object detection.

Chapter 5
Conclusion and Future Work

5.1. Conclusion

Since image recognition becomes one of the most popular and useful techniques, influencing many fields, it is worth to study and design a sophisticated system that can simulate human's vision and even performs better than that. Current networks, such as Le-Net [23], GoogLe-Net [25] and Res-Net [26], all focus on single object recognition. For a panoramic picture, additional framework [28, 30, 31, 74, 76] is required to segment original images in areas that only contain one object. Our network overcomes this shortage by combining the concept of edge detector and CNN. Without image segment, the system can have significant improvement in efficiency.

In this thesis, we introduced most popular techniques for object recognition – CNN and its related knowledge. We did research and analyzed advanced CNN structures chronologically proposed on ILSVRC about the concepts, benefits and shortages. We separate three sections to illustrate techniques: concept of CNN, related knowledges and advanced instances. Le-Net, as the earliest convolutional neural network, starts a new era for computer vision. Alex-Net improved Le-Net on both depth and width. The network can be applied to 1000 categories recognition. To refine the structure of Alex-Net, de-conv structure is proposed which generate ZF-Net. This new structure monitors output of each layer and adjusts relative parameters to reach optimal. At the same time, Karen Simonyan and Andrew Zisserman considers the extension of channels and depth of Alex-Net to achieve high accuracy. GoogLe-Net presents multi-scale kernels and more complex structure. Res-Net, on the other size, follows the traditional structure, but introduce recurrent network concepts. By comparing the structure, applicable occasion and performance of each network, we understand the restriction of current networks and the reason for that. Learning novel idea from each network, we gained insight into image processing, neural network and machine learning techniques which are cutting-edge and necessary to improve our lives.

Based on what we learnt, we proposed a network structure that can directly deal with plural object detection. Different from other's work, the concept behind our network is based on edge detector rather than conventional neural network. To achieve real-time processing, our focus is on width of the network instead of depth. Inspired by GoogLe-Net, we combined 12 parameter-unassigned kernels of various sizes (1*1, 3*3, 5*5, 7*7) together and utilized back-propagation algorithm to refine parameters. From the experiment, we know that mean-pooling layer has a very efficient function to reduce feature noises, raise accuracy by 3.7%. To keep the output size constant, we modify traditional pooling layer. ReLU activation layer is also applied in different positions. This layer also has function to increase recognition accuracy by 10.3%. We used the square error function between the average value of output matrix and target as cost function to achieve object recognition task. In terms of efficiency, because the convolutional operation is time consuming, we applied mixed strategy which raises 27 times processing efficiency. The network of final version has relatively satisfactory performance for both single object detection (95%) and plural object detection (89%) with 5 FPS for 600*800 image.

Through experiments, we deeply understand the advantages and shortages of the network. Our network has a high detecting rate for large scale image (600*800) while others' masterpieces only focus on 225*225 regions of a single object. Compared to region based convolution neural networks, there is no additional region proposal needed for our network. So the processing efficiency is high. With large breadth and small depth, the network is easy to utilize distribution computing techniques for further improvement. Compared to Faster R-CNN, the structure of our network is simple with fewer parameters to train. As a result, only a small size of the training group (100 images) and fewer number of learning iterations (7000) are necessary. The training process only costs 10 minutes for 100-sample training of two categories. Most networks, such as Re-Net needs more than 10,000 samples, costing 3-7 days for training. Besides, the size of training samples for our network is not restricted. Theoretically, any size is available as long as it is the same as mean-matrix size. We don't need to resize images to meet with the input condition which saves us time to normalize different datasets. There are also some other advantages we tested during experiments: it has resistance ability for object deformation and illumination changing. So the network can be applied on the most situations.

Once we complete all requirements, the system can be applied for real scene vehicle and face detection with current trained parameters. We can apply the network on AR devices utilizing mobile ad hoc network techniques [96-104]. One common problem of real scene recognition for existing systems is the object overlapping. There is a high change that multiple objects are overlapped together because of viewpoint and depth of focus. However, we can improve the network to solve this problem. The output is of mean map shows the label prediction of each pixel. The difference to the average category label can be regarded as class similarity. Short distance means this pixel more likely belongs to a specific class. As a result, the entire matrix can be transformed to likelihood map which is similar to infrared scene. Two overlapped objects will not be indicated as one object unless their center areas are too close to each other. So our network can recognize more objects on the real scene than current networks.

5.2. Future Work

However, besides the problems we have mentioned, our network also has some disadvantages that need to overcome in the future. First of all, current training process is based on small-size samples (32*32). Although recognition range can be 1.5 times larger keeping high accuracy, in a 600*800 real scene, the objects, such as buildings and vehicles can cross over this range. For example, in a very close distance between an apple and observer, its region can be larger than 200*200 which cannot be recognized by current network.

Secondly, all plural object detection tests are based on pure-color background. We tried several images of a real scene, the accuracy is lower than 40%. We believe that a new category trained by background images is necessary. This category teaches the network what kind of features are noises. However, this method can only temporally reduce noises of a real scene, because considering many factors, such as illumination, combination of different objects, changing viewpoint of the scene, there are billions of complex background, which may not be commonly described.

The third problem is the time cost of recognition processing. Our network can achieve at least 5 FPS for 600*800 images without region proposals. We still need to improve our network to reach to at least 24 FPS for real-time processing. We can utilize GPU instead of CPU to help process images which can fasten 40%. Besides, inspired by Alex-Net, we can also separate

channels into different groups, and use different GPU to deal with groups in parallel. Video stream techniques [111-114] and distributed systems [115-118] can also be applied to fasten processing.

Finally, our network has the consistence problem in terms of mean-matrix size. In label map computation, size of scanning windows should be the same as training samples, otherwise the output of post-process is meaningless. Since the size of training samples of different class can be different, to keep the consistence of mean matrix, we have to compute label map for many times with kernels of different sizes, which costs much time.

For the future work, besides to find solution towards problems mentioned above, we also plan to extend recognition categories, design improvement methods for non-linearly illumination changing. As for vehicle recognition, vehicle localization techniques [119-123] can be applied to reduce recognition time cost. Limited by simplicity of network structure, detection and recognition rate for real scene is low. We may need to complicate network little by little to reinforce distinguishing ability. Once our network is solid, we plan to apply it on video stream with related techniques [124-128] for practical application.

References

[1] Roberts, L. G. (1980). Machine Perception of Three-dimensional Solids. New York: Garland.

[2] Schantz, H. F. (1982). The History of OCR, Optical Character Recognition. *Manchester Center*, VT: Recognition Technologies Users Association.

[3] DAlbe, E. E. (1914). On a Type-reading Optophone. London: Harrison and Sons.

[4] Moravec, H. P. (1980). Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Stanford, CA: Computer Science Dept., Stanford University.

[5] Harris, C., & Stephens, M. (1988). A Combined Corner and Edge Detector. *Proceedings of the Alvey Vision Conference 1988*.

[6] Lowe, D. (1999). Object Recognition from Local Scale-invariant Features. *Proceedings of the Seventh IEEE International Conference on Computer Vision.*

[7] Seo, J., & Yoona, K. (2012). Modified Speeded Up Robust Features(SURF) for Performance Enhancement of Mobile Visual Search System. *Journal of Broadcast Engineering*, 17(2), 388-399.

[8] Canny, J. (1987). A Computational Approach to Edge Detection. *Readings in Computer Vision*, 184-203.

[9] Goel, R., Vashishtha, P. K., Sahni, P., & Kumar, A. (2012). Survey on Image Processing with Edge Detection Techniques. *Paripex - Indian Journal of Research*, 3(5), 177-180.

[10] Bundy, A., & Wallen, L. (1984). Difference of Gaussians. *Catalogue of Artificial Intelligence Tools*, 30-30.

[11] Freedman, D. (2005). *Statistical Models: Theory and Practice.* Cambridge: Cambridge University Press.

[12] Russell, S. N. (2018). *Artificial Intelligence: A Modern Approach*. S.L.: Pearson.

[13] Wang, T., Li, Z., Yan, Y., & Chen, H. (2007). A Survey of Fuzzy Decision Tree Classifier Methodology. *Advances in Soft Computing Fuzzy Information and Engineering*, 959-968.

[14] Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 119-139.

[15] Jandel, M. (2010). A Neural Support Vector Machine. *Neural Networks*, 23(5), 607-613.

[16] Bretscher, O. (2001). *Linear Algebra with Applications*. Upper Saddle River, NJ: Prentice Hall.

[17] Seal, H. L. (1968). The Historical Development of the Gauss Linear Model. New Haven: Yale University.

[18] M. W., & W. P. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5(4), 115-133.

[19] Hebb, D. O. (1949). *The Organization of Behavior.* Mahwah, N.J: L. Erlbaum Associates.

[20] Farley, B.G., & W.A. Clark (1954). Simulation of Self-Organizing Systems by Digital Computer. *IRE Transactions on Information Theory*. 4 (4): 76–84.

[21] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6), 386-408.

[22] Agarwal, M. (2017). Back Propagation in Convolutional Neural Networks - Intuition and Code. Retrieved from https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

[23] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

[24] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6), 84-90.

[25] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., & Rabinovich, A. (2015). Going Deeper with Convolutions. 2*015 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR).

[26] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR).

[27] Symonyan. K., & Zisserman. A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Vision and Pattern Recognition*.

[28] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

[29] Qu, T., Zhang, Q., & Sun, S. (2016). Vehicle Detection from High-Resolution Aerial Images Using Spatial Pyramid Pooling-based Deep Convolutional Neural Networks. *Multimedia Tools and Applications*, 76(20), 21651-21663.

[30] Ross. G. (2016). Fast R-CNN. *Computer ViBKsion Foundation*, 1440-1448.

[31] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149.

[32] Shontell, A. (2015). Snapchat Buys Looksery, A 2-Year-Old Startup That Lets You Photoshop Your Face While You Video Chat. Retrieved from

http://www.businessinsider.com/snapchat-buys-looksery-2015-9

[33] Ewing, I. (2015). New SmartGate Aims to Decrease Airport Queues. Retrieved from http://www.newshub.co.nz/home/new-zealand/2015/07/new-smartgate-aims-to-decrease-airport-queues.html

[34] Freeman, W. T., & Roth, M. (1994). Orientation Histograms for Hand Gesture Recognition. *Automatic Face and Gesture Recognition*.

[35] Zheng, L., Yang, Y., & Tian, Q. (2018). SIFT Meets CNN: A Decade Survey of Instance Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5), 1224-1244.

[35] Guillaumin, M., & Ferrari, V. (2012). Large-scale Knowledge Transfer for Object Localization in ImageNet. *2012 IEEE Conference on Computer Vision and Pattern Recognition*.

[36] Hinton, G. (2009). *Deep Belief Networks*. Scholarpedia, 4(5), 5947.

[37] Shelke, S., & Apte, S. (2010). A Novel Multi-feature Multi-classifier Scheme for Unconstrained Handwritten Devanagari Character Recognition. *2010 12th International Conference on Frontiers in Handwriting Recognition*.

[38] Chao, L., Tao, J., Yang, M., Li, Y., & Wen, Z. (2016). Long Short Term Memory Recurrent Neural Network based Encoding Method for Emotion Recognition in Video. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP).

[39] Li, X., & Wu, X. (2015). Constructing Long Short-term Memory Based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP).

[40] Barghout, L., & Sheynin, J. (2013). Real-World Scene Perception and Perceptual Organization: Lessons from Computer Vision. *Journal of Vision*, 13(9), 709-709.

[41] Batenburg, K., & Sijbers, J. (2009). Adaptive Thresholding of Tomograms by Projection Distance Minimization. *Pattern Recognition*, 42(10), 2297-2305.

[42] Mobahi, H., Rao, S. R., Yang, A. Y., Sastry, S. S., & Ma, Y. (2011). Segmentation of Natural Images by Texture and Boundary Compression. *International Journal of Computer Vision*, 95(1), 86-98.

[43] Anzai, Y. (1992). *Pattern Recognition and Machine Learning*. Boston: Academic Press.

[44] Hu, Y. H., & Hwang, J. (2001). *Handbook of Neural Network Signal Processing*. Boca Raton, FL: CRC Press.

[45] Mohammed, M., Khan, M. B., & Bashier, E. B. (2017). *Machine Learning: Algorithms and Applications*. Boca Raton: CRC Press.

[46] Minsky, M., & Papert, S. (2017). *Perceptrons an Introduction to Computational Geometry*. Cambridge, MA: The MIT Press.

[47] Ghosh, S. K. (2013). *Digital Image Processing*. Oxford: Alpha Science International.

[48] Auer, P., Burgsteiner, H., & Maass, W. (2008). A Learning Rule for Very Simple Universal Approximators Consisting of a Single Layer of Perceptrons. *Neural Networks*, 21(5), 786-795.

[49] Tahmasebi, P., & Hezarkhani, A. (2011). Application of a Modular Feedforward Neural Network for Grade Estimation. *Natural Resources Research*, 20(1), 25-32.

[50] LeCun, Y., Botton, L., Bengio, Y., & Haffner, P. (2009). Gradient Based Learning Applied to Document Recognition. *Intelligent Signal Processing*.

[51] Glorot, X., Bordes, A., & Bengio, Y. (2011). *Deep Sparse Rectifier Neural Networks*. AISTATS, 315-323.

[52] Collobert, R., & Bengio, S. (2004). Links between Perceptrons, MLPs and SVMs. *Twenty-first International Conference on Machine Learning - ICML 04*.

[53] Broomhead, D., & Lowe, D. (1988). Multivariate Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321-355.

[54] Liang, J., & Liu, R. (2015). Stacked Denoising Autoencoder and Dropout Together to Prevent Over-fitting in Deep Neural Network. *2015 8th International Congress on Image and Signal Processing (CISP)*.

[55] Sober, E. (2015). *Ockam's Razor: A User's Manual.* Cambridge University Press. p. 4.

[56] El-Shaarawi, A. H., & Piegorsch, W. W. (2002). *Encyclopedia of Environmetrics* (Vol. 3). New York: J. Wiley.

[57] Russell, S. N. (2018). *Artificial Intelligence: A Modern Approach*. S.L.: Pearson.

[58] MacKay, D. (2003). Chapter 20: An Example Inference Task: Clustering. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 284–292.

[59] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323(6088), 533-536.

[60] Glorot, X., & Yoshua, B. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. *International Conference on Artificial Intelligence and Statistics*.

[61] McCaffrey, J. (2015). Variation on Back-Propagation: Mini-Batch Neural Network Training. Retrieved from https://visualstudiomagazine.com/articles/2015/07/01/variation-on-back-propagation.aspx

[62] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Over-fitting. *Journal of Machine Learning Research*, 1929-1958.

[63] Ho, T. (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278–282.

[64] Uijlings, J, et al. Selective Search for Object Recognition. *International Journal of Computer Vision*, vol. 104, no. 2, 2013, 154–171.

[65] Dickerson, Naomi. (2017). Refining Bounding-Box Regression for Object Localization.

[66] Fok, R., An, A., & Wang, X. (2017). Spontaneous Symmetry Breaking in Deep Neural Networks.

[67] Broomhead, D., & Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*. 2: 321–355.

[68] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. (2009). What Is the Best Multi-Stage Architecture for Object Recognition? *In International Conference on Computer Vision*, 2146–2153.

[69] Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, 818-833.

[70] Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. *2011 International Conference on Computer Vision*.

[71] Hebb, D. O. (2002). *The Organization of Behavior*. Mahwah, N.J: L. Erlbaum Associates.

[72] He, K., & Sun, J. (2015). Convolutional Neural Networks at Constrained Time Cost. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[73] Bishop, C. M. (1999). *Neural Networks for Pattern Recognition*. Oxford: Oxford Univ. Press.

[74] Zeng, X., Ouyang, W., Yan, J., Li, H., Xiao, T., Wang, K., & Wang, X. (2017). Crafting GBD-Net for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1-1.

[75] Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[76] He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, 346-361.

[77] Grauman, K., & Darrell, T. (2005). The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *Tenth IEEE International Conference on Computer Vision (ICCV05)* Volume 1.

[78] Aich, S., & Stavness, I. (2017). Leaf Counting with Deep Convolutional and Deconvolutional Networks. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*.

[79] Matas, J., O. Chum, M. Urban, and T. Pajdla. (2002). Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. *Procedings of the British Machine Vision Conference 2002*.

[80] Glorot, Xavier, and Yoshua Bengio. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. *International Conference on Artificial Intelligence and Statistics*, 249-56.

[81] Vehicle Image Database. Retrieved from https://www.gti.ssr.upm.es/data/Vehicle_database.html.

[82] INRIA Person Dataset. Retrieved from http://pascal.inrialpes.fr/data/human/.

[83] Sebastian, R. (2017) An Overview of Gradient Descent Optimization Algorithms. Sebastian Ruder. Retrieved from http://ruder.io/optimizing-gradient-descent/.

[84] Zhang, Z., Xing, F., Su, H., & Yang, L. (2017). Recent Advances in the Applications of Convolutional Neural Networks to Medical Image Contour Detection. ResearchGate.

[85] Ahn, H. (2008). Convolution. Retrieved from http://www.songho.ca/dsp/convolution/convolution.html

[86] Abolafia, D. (2016). A Recurrent Neural Network Music Generation Tutorial. Retrieved from https://magenta.tensorflow.org/2016/06/10/recurrent-neural-network-generation-tutorial

[87] Chandrayan, P. (2017). Deep Learning: Deep Belief Network Fundamentals – Codeburst. Retrieved from https://codeburst.io/deep-learning-deep-belief-network-fundamentals-d0dcfd80d7d

[88] Stanford University. (2018). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/convolutional-networks/#pool

[89] Bhande, A. (2018). What Is Underfitting and Over-fitting in Machine Learning and How to Deal With It. Retrieved from https://medium.com/greyatom/what-is-underfitting-and-over-fitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

[90] McCaffrey, J. (2015). Variation on Back-Propagation: Mini-Batch Neural Network Training. Retrieved from https://visualstudiomagazine.com/articles/2015/07/01/variation-on-back-propagation.aspx

[91] Miyazawa, K., Ito, K., Aoki, T., Kobayashi, K., & Nakajima, H. (2005). An Efficient Iris Recognition Algorithm Using Phase-Based Image Matching. *IEEE International Conference on Image Processing 2005*.

[92] Viola, P. and Jones, M. (2001). Robust Real-time Object Detection, *2nd International Workshop on Statistical and Computational Theories of Vision – Modelling, Learning, Computing and Sampling*, Vancouver, 1–20.

[93] Radford, Alec, Metz, Luke, and Chintala, Soumith. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

[94] Transport Canada. (2017). Road Safety in Canada. Retrieved from https://www.tc.gc.ca/eng/motorvehiclesafety/tp-tp15145-1201.htm

[95] Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi et al. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, 2672-2680.

[96] HABFernandes De Oliveira, Boukerche, A., Nakamura, E. F., & Loureiro, A. A. (2009). An Efficient Directed Localization Recursion Protocol for Wireless Sensor Networks. *IEEE Transactions on Computers*, 58(5), 677-691.

[97] Cunha, F., Villas, L., Boukerche, A., Maia, G., Viana, A., Mini, R. A., & Loureiro, A. A. (2016). Data Communication in VANETs: Protocols, Applications and Challenges. *Ad Hoc Networks*, 44, 90-103.

[98] Elhadef, M., Boukerche, A., & Elkadiki, H. (2008). A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 68(3), 321-335.

[99] Boukerche, A., Martirosyan, A., & Pazzi, R. (2008). An Inter-cluster Communication based Energy Aware and Fault Tolerant Protocol for Wireless Sensor Networks. *Mobile Networks and Applications*, 13(6), 614-626.

[100] Pazzi, R. W., & Boukerche, A. (2014). Propane: A Progressive Panorama Streaming Protocol to Support Interactive 3d Virtual Environment Exploration on Graphics-Constrained Devices. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1), 1-22.

[101] Antoniou, T., Chatzigiannakis, I., Mylonas, G., Nikoletseas, S., & Boukerche, A. (2004). A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *37th Annual Simulation Symposium, 2004. Proceedings*, 29(4), 477-489.

[102] Boukerche, A., & Turgut, D. (2007). Secure Time Synchronization Protocols for Wireless Sensor Networks. *IEEE Wireless Communications*, 14(5), 64-69.

[103] Boukerche, A., & Abrougui, K. (2006). An efficient leader election protocol for mobile networks. *Proceeding of the 2006 International Conference on Communications and Mobile Computing*.

[104] Boukerche, A., Pazzi, R., & Araujo, R. (2005). HPEQ A Hierarchical Periodic, Event-driven and Query-based Wireless Sensor Network Protocol. *The IEEE Conference on Local Computer Networks 30th Anniversary*, 560-567.

[105] Villas, L. A., Boukerche, A., Ramos, H. S., H. A. B. F. De Oliveira, Araujo, R. B., & Loureiro, A. A. (2013). DRINA: A Lightweight and Reliable Routing Approach for In-Network Aggregation in Wireless Sensor Networks. *IEEE Transactions on Computers*, 62(4), 676-689.

[106] Multirate Geographic Opportunistic Routing Protocol Design. (2011). *Multihop Wireless Networks*, 193-211.

[107] Boukerche, A., & Darehshoorzadeh, A. (2014). Opportunistic Routing in Wireless Networks: Models, Algorithms, and Classifications. *ACM Computing Surveys*, 47(2), 1-36.

[108] Zhang, Z., Pazzi, R. W., & Boukerche, A. (2010). A Mobility Management Scheme for Wireless Mesh Networks Based on A Hybrid Routing Protocol. *Computer Networks*, 54(4), 558-572.

[109] Villas, L., Boukerche, A., Araujo, R. B., & Loureiro, A. A. (2010). Highly Dynamic Routing Protocol for data aggregation in sensor networks. *The IEEE Symposium on Computers and Communications,* 496-502.

[110] Boukerche, A., Araujo, R., & Villas, L. (2006). A Wireless Actor and Sensor Networks QoS-Aware Routing Protocol for the Emergency Preparedness Class of Applications. *Proceedings. 2006 31st IEEE Conference on Local Computer Networks,* 832-839.

[111] Rezende, C., Boukerche, A., Ramos, H. S., & Loureiro, A. A. (2015). A Reactive and Scalable Unicast Solution for Video Streaming over VANETs. *IEEE Transactions on Computers*, 64(3), 614-626.

[112] Boukerche, A., Hong, S., & Jacob, T. (2002). An Efficient Synchronization Scheme of Multimedia Streams in Wireless and Mobile Systems. *IEEE Transactions on Parallel and Distributed Systems*, 13(9), 911-923.

[113] Boukerche, A., Pazzi, R. W., & Feng, J. (2008). An End-To-End Virtual Environment Streaming Technique for Thin Mobile Devices Over Heterogeneous Networks. *Computer Communications*, 31(11), 2716-2725.

[114] Rezende, C., Boukerche, A., Ramos, H. S., & Loureiro, A. A. (2015). A Reactive and Scalable Unicast Solution for Video Streaming over VANETs. *IEEE Transactions on Computers*, 64(3), 614-626.

[115] Boukerche, A., & Dzermajko, C. (2004). Performance Evaluation of Data Distribution Management Strategies. *Concurrency and Computation: Practice and Experience*, 16(15), 1545-1573.

[116] Boukerche, A., Mcgraw, N., Dzermajko, C., & Lu, K. (n.d.). Grid-Filtered Region-Based Data Distribution Management in Large-Scale Distributed Simulation Systems. *38th Annual Simulation Symposium*, 259-266.

[117] Boukerche, A., & Tropper, C. (1998). A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*, 9(8), 748-757.

[118] Boukerche, A., Roy, A., & Thomas, N. (2000). Dynamic grid-based multicast group assignment in data distribution management. *Proceedings Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*.

[119] Abumansoor, O., & Boukerche, A. (2012). A Secure Cooperative Approach for Nonline-of-Sight Location Verification in VANET. *IEEE Transactions on Vehicular Technology*, 61(1), 275-285.

[120] Boukerche, A., Rezende, C., & Pazzi, R. W. (2009). Improving Neighbor Localization in Vehicular Ad Hoc Networks to Avoid Overhead from Periodic Messages. *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, 1-6.

[121] Oliveira, H. A., Nakamura, E. F., Loureiro, A. A., & Boukerche, A. (2005). Error Analysis of Localization Systems for Sensor Networks. *Proceedings of the 2005 International Workshop on Geographic Information Systems*.

[122] Abrougui, K., Pazzi, R. W., & Boukerche, A. (2010). Performance Evaluation of Location-Based Service Discovery Protocols for Vehicular Networks. *2010 IEEE International Conference on Communications*, 12(3), 717-735.

[123] Boukerche, A., & Rogers, S. (n.d.). GPS query optimization in mobile and wireless networks. Proceedings. *Sixth IEEE Symposium on Computers and Communications*.

[124] Boukerche, A., & Samarah, S. (2008). A Novel Algorithm for Mining Association Rules in Wireless Ad-Hoc Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(7), 865-877.

[125] Boukerche, A., Das, S., Fabbri, A., & Yildiz, O. (n.d.). Exploiting Model Independence for Parallel PCS Network Simulation. *Proceedings Thirteenth Workshop on Parallel and Distributed Simulation*.

[126] Boukerche, A., & Ren, Y. (2008). A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. *Proceedings of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*.

[127] Villas, L. A., Boukerche, A., Guidoni, D. L., Oliveira, H. A., Araujo, R. B., & Loureiro, A. A. (2013). An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks. *Computer Communications*, 36(9), 1054-1066.

[128] Boukerche, A., Zarrad, A., & Araujo, R. (2007). A Novel Gnutella Application Layer Multicast Protocol for Collaborative Virtual Environments over Mobile Ad-Hoc Networks. *2007 IEEE Wireless Communications and Networking Conference*, 911-924.