Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DIGITAL IMAGE FORENSICS VIA META-LEARNING AND FEW-SHOT LEARNING

by
Yuxi Shi

Digital images are a substantial portion of the information conveyed by social media, the Internet, and television in our daily life. In recent years, digital images have become not only one of the public information carriers, but also a crucial piece of evidence. The widespread availability of low-cost, user-friendly, and potent image editing software and mobile phone applications facilitates altering images without professional expertise. Consequently, safeguarding the originality and integrity of digital images has become a difficulty. Forgers commonly use digital image manipulation to transmit misleading information. Digital image forensics investigates the irregular patterns that might result from image alteration. It is crucial to information security.

Over the past several years, machine learning techniques have been effectively used to identify image forgeries. Convolutional Neural Networks(CNN) are a frequent machine learning approach. A standard CNN model could distinguish between original and manipulated images. In this dissertation, two CNN models are introduced to recognize seam carving and Gaussian filtering.

Training a conventional CNN model for a new similar image forgery detection task, one must start from scratch. Additionally, many types of tampered image data are challenging to acquire or simulate.

Meta-learning is an alternative learning paradigm in which a machine learning model gets experience across numerous related tasks and uses this expertise to improve its

future learning performance. Few-shot learning is a method for acquiring knowledge from few data. It can classify images with as few as one or two examples per class. Inspired by meta-learning and few-shot learning, this dissertation proposed a prototypical networks model capable of resolving a collection of related image forgery detection problems. Unlike traditional CNN models, the proposed prototypical networks model does not need to be trained from scratch for a new task. Additionally, it drastically decreases the quantity of training images.

DIGITAL IMAGE FORENSICS VIA META-LEARNING AND FEW-SHOT LEARNING

by
Yuxi Shi

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann
Department of Electrical and Computer Engineering

August 2022

APPROVAL PAGE

DIGITAL IMAGE FORENSICS VIA META-LEARNING AND FEW-SHOT LEARNING

Yuxi Shi

Dr. Mengchu Zhou, Dissertation Advisor — Date
Distinguished Professor of Electrical and Computer Engineering
New Jersey Institute of Technology

Dr. Yun-Qing Shi, Dissertation Co-Advisor — Date
Professor Emeritus of Electrical and Computer Engineering
New Jersey Institute of Technology

Dr. John Carpinelli, Committee Member — Date
Professor of Electrical and Computer Engineering
New Jersey Institute of Technology

Dr. Edwin Hou, Committee Member — Date
Professor of Electrical and Computer Engineering
and Associate Dean of Academic Affairs
New Jersey Institute of Technology

Dr. Xuan Liu, Committee Member — Date
Associate Professor of Electrical and Computer Engineering
New Jersey Institute of Technology

Dr. Frank Shih, Committee Member — Date
Professor of Computer Science
New Jersey Institute of Technology

BIOGRAPHICAL SKETCH

Author: Yuxi Shi

Degree: Doctor of Philosophy

Date: August 2022

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2022

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ,2017

- Bachelor of Science in Electrical Engineering,
  Guangdong University of Technology, Guangzhou, P. R. China, 2013

Major: Electrical Engineering

Presentations and Publications:

Ding, F., Shi, Y., Zhu, G., & Shi, Y. Q. (2020). Real-time estimation for the parameters of Gaussian filtering via deep learning. *Journal of real-time image processing*, *17*(1), 17-27.

Ding, F., Shi, Y., Zhu, G., & Shi, Y. Q. (2019). Smoothing identification for digital image forensics. *Multimedia tools and applications*, *78*(7), 8225-8245.

Ye, J., Shi, Y., Xu, G., & Shi, Y. Q. (2018, October). A convolutional neural network based seam carving detection scheme for uncompressed digital images. In *International workshop on digital watermarking* (pp. 3-13). Springer, Cham.

Song, W., Wang, H., Zhang, X., Xia, J., Liu, T., & Shi, Y. (2022). Deep-sea Nodule Mineral Image Segmentation Algorithm Based on Pix2PixHD. *Computers, materials & continua*, *73*(1), 1449–1462.

Song, W., Dong, L., Zhao, X., Xia, J., Liu, T., & Shi, Y. (2022). Review of Nodule Mineral Image Segmentation Algorithms for Deep-Sea Mineral Resource Assessment. *Computers, materials & continua*, *73*(1), 1649–1669.

To my mother Lin Bai and my father, Hongyan Shi

ACKNOWLEDGMENT

Looking back the last five years, it is a meaningful journey in my life. First of all, I want to appreciate Prof. Yun Q. Shi. Prof. Shi took a lot of time to guide me through my Ph.D. program. I am grateful for all his assistance at this time. Also, I would like to express my gratitude to Prof. Mengchu Zhou. I am really pleased to have his help to overcome difficulties in my study. Without their guidance and encouragement, I could not have completed my Ph.D. dissertation.

In addition, I am so very thankful to my committee members: Prof. Xuan Liu, Prof. Prof. John Carpinelli, Prof. Edwin Hou and Prof. Frank Shih. Their insightful and instructive suggestions made my dissertation better.

Thank you also to the Department of Electrical and Computer Engineering for their teaching assistantship support.

Then I would like to thank all faculty members in Department of Electrical and Computer Engineering. They are always willing to provide help for students with patience. Chatting with Prof. Cong Wang is a delightful experience.

Thanks should also go to my best friend Dr. Zhangyi Shen. It was an unforgettable and enjoyable time we spent together in the past years.

I would be remiss in not mentioning my parents. I could not have undertaken this journey without their understanding and support. Their belief in me has kept my spirits and motivation high during this process.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

.

LIST OF FIGURES
(Continued)

CHAPTER 1

GIVING COMPUTERS THE ABILITY TO LEARN FROM DATA

1.1 A Brief History of Neural Networks

The use of artificial intelligence is nearly ubiquitous in our daily lives. Due to the advancement of modern technology, artificial intelligence is playing an ever-increasing role in business and industry. Artificial intelligence powers Google's sophisticated search engine, Amazon's recommendation system, and Tesla's self-driving technology. There is no doubt that artificial intelligence has made significant technological progress in recent years.

It takes scientists a considerable amount of time to build artificial intelligence. In his book The Organization of Behavior, published in 1949, Donald Hebb developed a model of brain cell interaction. Based on their excitation, Hebb's model summarizes his beliefs on how neurons interact with one another. This is the most fundamental theory for artificial neural networks in existence today.

Arthur Lee Samuel of IBM created a checkers-playing computer software in the 1950s. Even on an IBM commercial computer, memory is quite restricted at the time. Samuel utilized the algorithm presently known as alpha-beta pruning. He created a scoring function. It recorded the locations of every piece on the board and computed the probability of victory for both sides. The computer's next move is determined using a minimax strategy, which subsequently developed into the minimax algorithm. The computer software optimizes the value of its scoring function, presuming that its opponent would attempt to do the same for its next step.

In his time, Samuel popularized the terminology "machine learning." His checkers-playing program was the first self-learning program to be effective. His efforts indicate the potential for artificial intelligence to attain human levels.

Frank Rosenblatt at the Cornell Aeronautical Laboratory developed the perceptron, a binary classification method, in 1957. It was initially implemented as software for the IBM 704. A five-ton IBM 704 computer could detect images based on its original idea. There was just one layer in Rosenblatt's perceptron, but current neural networks include millions. The perceptron principle laid the groundwork for contemporary deep learning and neural networks.

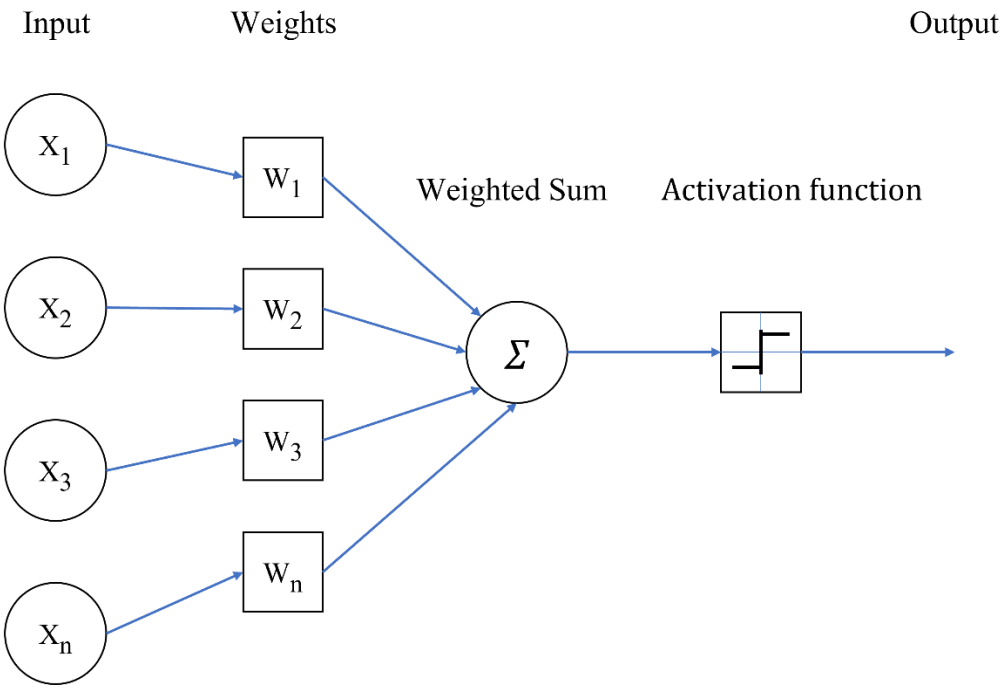


Figure 1.1 Basic structure of a perceptron.

Following are the major components of a perceptron:

1. Input: Here, x is the feature value and n is the total number of features. The input vector for a perceptron is $[x_1,x_2,x_3...x_n]$.

2. Weights: Weight must be multiplied by the input feature's value. We assign starting values to the weights, which will be updated throughout the training process. The

weight vector is denoted as [w_1,w_2,w_3…w_n].

3. Weighted Sum: Multiply each feature's value x_n by its corresponding weight w_n . We denote the weighted sum as $\sum x_i w_i$ for every i in [1,2,3.....n].

4. Activation Function: Typically, activation function is a nonlinear function used for nonlinear regression and nonlinearly separable classification problems.

5. Output: The output of Perceptron is the predicted output depending on the input feature. It aids the perceptron in updating the values of the weights.

Multilayer perceptron (MLP) was discovered in the 1960s. With three or more layers, major breakthrough is possible. It has a far greater capacity for learning than a single perceptron. Multiple perceptrons are arranged in a layer. A minimum of three layers comprise an MLP: an input layer, a hidden layer, and an output layer. MLP has complete connectivity. This indicates that one node in one layer is connected to all nodes in the following layer.

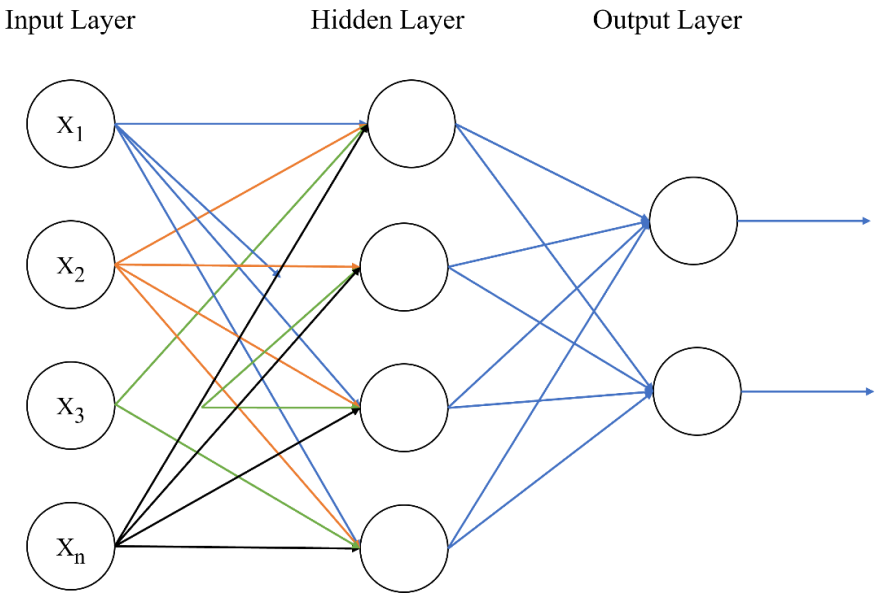


Figure 1.2 A basic structure of multilayer perceptron contains three layers.

A perceptron could only tackle problems of binary classification. Depending on its activation function, an MLP is capable of doing both classification and regression. MLP was a prevalent machine learning approach for speech recognition, picture identification, and text translation in the 1980s.

In the past two decades, neural networks have captured the attention of more scientists and researchers than ever before. Large-scale neural networks have evolved. Applications in the real world concentrate mostly on image recognition, picture segmentation, and speech recognition.

In 2011, the team led by Dan Ciresan developed multi-column deep neural networks (MCDNN) [1] to win the final phase of the German traffic sign recognition benchmark at the International Joint Conference on Neural Networks (IJCNN). It is the only technique in this competition that exceeds human recognition by 99.46%.

AlexNet [2], which reached a top-five error rate of 15.3% in 2012, was a significant development in the machine learning community. Network architecture depth is a prerequisite for success. AlexNet consists of convolutional neural networks (CNN). The first five convolutional layers of the architecture create feature maps. Some of them link to layers with maximum pooling. The final three levels are fully-connected layers. The complexity of CNN makes human-competitive performance achievable.

More and more scientists and researchers develop their neural networks with deep architecture, inspired by AlexNet. AlexNet is regarded as one of the most significant achievements in the field of deep learning research. According to Google Scholar, the original publication has been referenced over 10,000 times as of 2022.

Google researchers developed GooLeNet, a 22-layer deep convolutional neural

network. It was presented in the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14). It could accomplish computer vision tasks like image classification and facial recognition. Microsoft Research Asia developed a CNN model with over 100 layers in 2015. Its performance was superior to AlexNet and it won the 2015 ImageNet competition. In the present day, CNN model such as ResNet [3] and DenseNet [4] may be created with hundreds of layers.

1.2 Hardware Development amid Machine Learning

In the past, the computing capability of computers restricted the growth of machine learning. When perceptron was successfully implemented for the first time on the IBM 704, the computer system consisted of vacuum tubes. In its day, the IBM 704 was regarded as a highly dependable machine. However, every eight hours on average [5, 6] the IBM 704 failed. This was an essential program size restriction. Since the IBM 704 most likely failed before program translation or compilation.

Today's computer machines are far more powerful than the IBM 704. Distributed computing, parallel computing, and graphics processing units (GPU) are crucial considerations. These cutting-edge technologies push the boundaries for machine learning.

Distributed computing is a subfield of computer science that investigates distributed systems. The distributed system consists of groups of interconnected computers. All computers have a basic objective for their work, yet can do distinct jobs. Each computer has its own dedicated memory. They converse through the exchange of messages. Thus, even if one computer experiences a problem with its assigned task, other computers can continue to operate. A distributed computing system might tolerate

individual computer failure. This is a significant factor for the widespread use of distributed computing in the commercial and industrial sectors.

Parallel computing refers to the processing of several calculations concurrently. Frequently, a big computing work may be subdivided into smaller tasks, which can then be solved concurrently. Conceptually, distributed computing and parallel computing have certain similarities. There is an overall concept for classifying them. Using shared memory, all processors in parallel computing might transfer information amongst themselves. Each CPU in distributed computing has its own memory. A message exchange system allows processors to send and receive messages. The structure of their network is typically represented as a graph with one finite-state machine per node. Figures 1.3 and 1.4 illustrate a typical distributed computing system and parallel computing system.

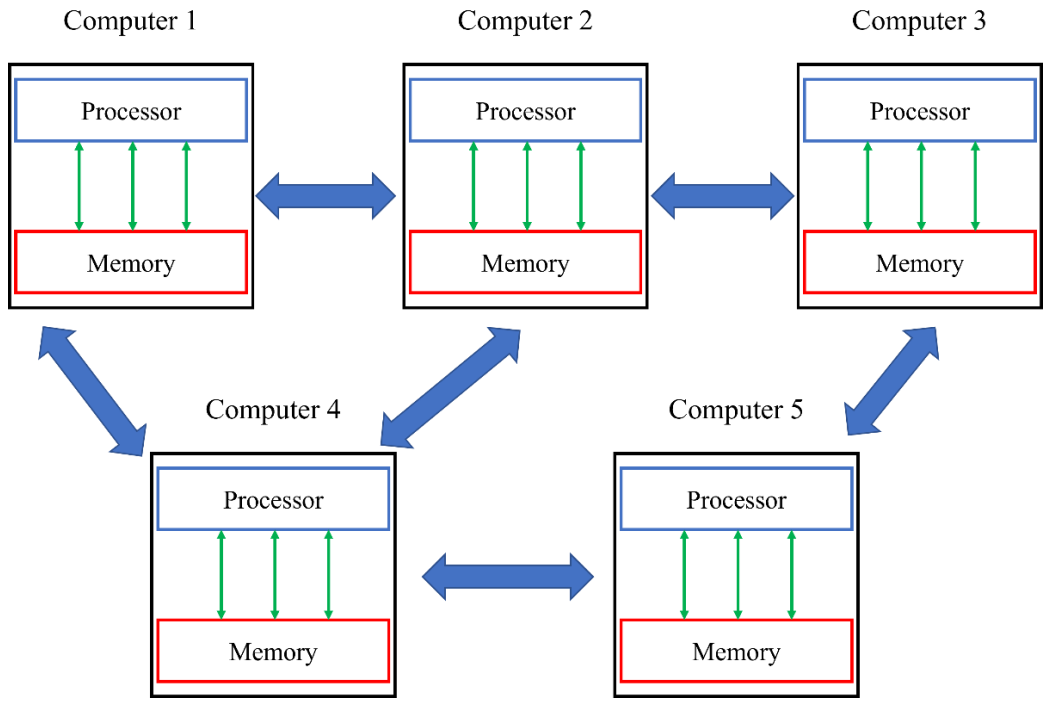


Figure 1.3 A typical distributed computing system.

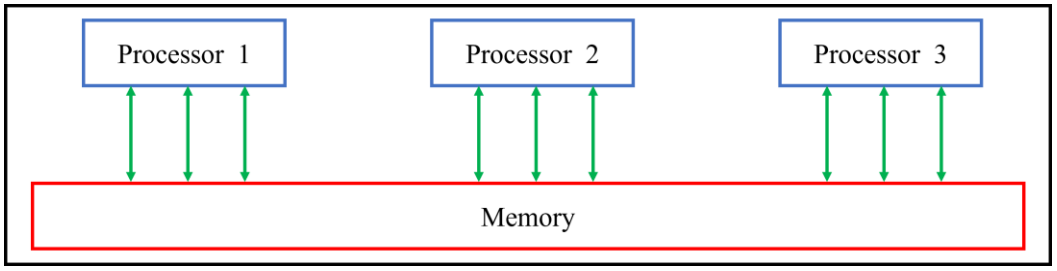


Figure 1.4 A basic parallel computing system.

The most recent graphics processing units(GPU) are another crucial technology that has grown to provide specific advantages for machine learning. Amid the rise of machine learning, the GPU have become one of the most essential computer technologies. GPU are extensively utilized in distributed and parallel computing for both personal and business applications.

GPU were initially meant to speed 3D computer graphics processing. Over time, GPU have gotten more programmable and are now able to produce more realistic environments in games and media using improved lighting and shadowing algorithms.

Typically, machine learning demands a great deal of computer power. As the complexity of deep learning algorithms increases, it takes longer to execute programs. A model of neural networks may have more than 100 billion parameters. Training this type of model is a time-consuming task.

Models of machine learning can be processed more quickly if all operations are executed simultaneously rather than sequentially. So why are GPU gaining popularity? Because GPU provide tremendous acceleration support for parallel processing of massive, continuous input data. A GPU contains a huge number of cores, which leads to improved parallel computation. In addition, GPU have a greater memory bandwidth, making them appropriate for processing massive volumes of data. The most advanced GPU have far

larger buses and faster memory clock rates than any CPU (central processing unit) available today. GPU are a specific type of hardware. The majority of image processing related machine learning tasks, including as face recognition, image classification, and image segmentation, are executed on GPU.

The quantity of data has been a significant factor in the rapid development of machine learning in recent years. We live in an age where data is expanding at an astounding rate. In several industries, including social media, healthcare systems, businesses, etc., data collecting is commonplace. Regardless of the machine learning technology, data plays an essential role. Instead of manually identifying patterns from vast amounts of data, machine learning algorithms might discover insights and make predictions. They are facilitating the effective transformation of data into knowledge.

1.3 Essential Development Tools for Machine Learning

Over the years, machine learning has developed from a theoretical concept to business applications used in our daily life. Machine learning was considered by many as a complex idea only for computer scientists. However, in recent years, machine learning has attracted more attention from individuals outside the scientific research fields.

How to build a machine learning development environment on my personal computer? What are the most popular development tools for machine learning? What programming language should I learn? Every new beginner has these kinds of questions at the first step. Because many development tools for machine learning are available today, the choice of development tools is sometimes based on personal preference and economic constraints.

Some essential development tools for machine learning will be introduced in this secession. Evey development tool has been proven useful not only for individuals but also for tech giants. We will begin with the programming language because this is the foundation of machine learning tools. Then we will introduce two machine learning frameworks used in this dissertation.

1.3.1 Python: The Most Popular Programming Language

Who uses Python today? Python was the top 1 in the IEEE Spectrum's annual interactive rankings of the top programming languages. According to the IEEE Spectrum, these rankings are created by weighting and combining 11 metrics from eight sources: CareerBuilder, GitHub, Google, Hacker News, the IEEE, Reddit, Stack Overflow, and Twitter. Python is considered to be among the top 5 or 10 most widely used programming languages today.

Figure 1.5 IEEE Spectrum top programming languages 2021.

Python has developed a large user base in the past ten years and gotten support from active developers' communities. From the Google Trend index, we can find Python was becoming increasingly popular worldwide.

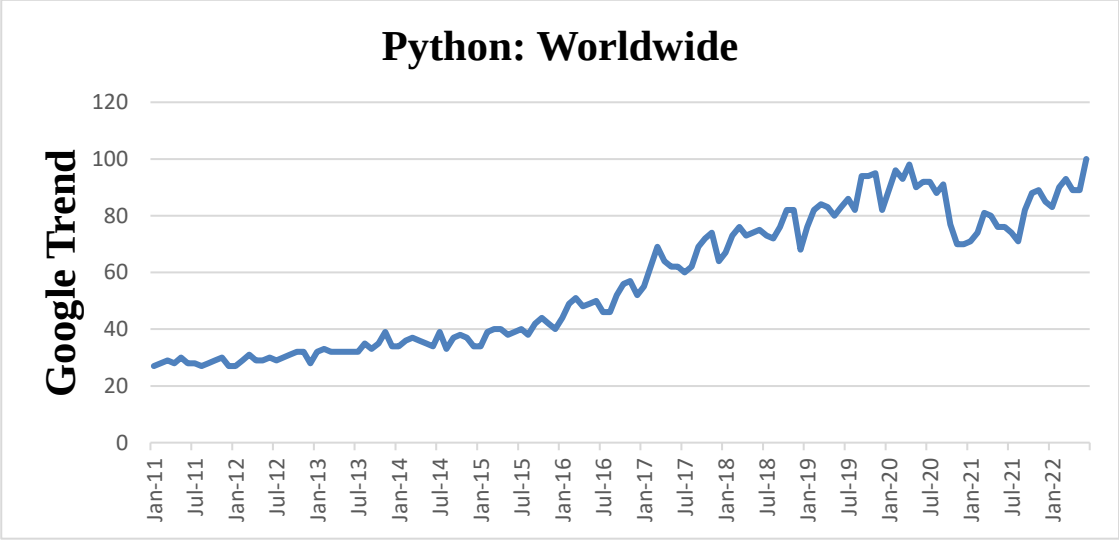


Figure 1.6 Python Google Trend index from Jan. 2011 to Jun. 2022.

Python has demonstrated its value for companies across different fields. Not only individual users benefit from Python. It has been a solid foundation to support commercial applications. For instance, here are some well-known companies are applying Python to their products and service:

1. Google uses Python in its web search system.
2. YouTube sharing service is largely written in Python.
3. Intel, Cisco, and IBM use Python for hardware testing.
4. JPMorgan Chase, and UBS apply Python to financial market forecasting.
5. NASA uses Python for scientific programming tasks.

And so on, the above companies are some representatives. We know Python is popular for both individual users and commercial companies. Then the question is, how does Python support machine learning studying? What are the main reasons for people to use Python? Let's focus on machine learning study and talk about more details behind Python's popularity.

Python is completely free and can compile and run on every major platform. Individual users and business companies both take a lot of benefit from Python's program portability. A machine learning model is usually built on a personal computer or an online cloud computing platform for an individual user. Below are the primary machine learning development platforms:

1. Linux and Unix systems
2. Microsoft Windows
3. Mac OS
4. Amazon Web Service

5. Microsoft Azure

6. Google Cloud

Python could run unchanged and stable across the above major development platforms. When running a Python program between local Windows and online cloud computing platforms, a developer just needs to make a copy of code between machines. We know that machine learning usually consumes a lot of computational power. When the machine learning model becomes more complex, the individual user always has the famous ran-out-of-memory error on a personal computer. It is impossible for everyone to invest in expensive machines, specifically machine learning. At this moment, cloud computing service is an easy solution to increase computational power. Today there are various cloud computing service options for individuals and business companies. The price is also getting more affordable.

Python comes with an extensive collection of prebuilt and third-party libraries. We have talked about how Python has a lot of communities. They give strong support for Python amid its development. One of Python fast grown use cases happens in scientific computing. Today, Python is heavily used in numeric programming. However, it is not a traditional domain for scripting languages. Compiled languages such as FORTRAN and C++ dominated this field for a long time. Also, some professional tools like MATLAB were widely used. Whatever may be a machine learning model, Python and numeric libraries are an inseparable combination to build it. Here are some main machine learning libraries used by Python developers:

1. NumPy

2. SciPy

3. Scikit-learn
4. Theano
5. TensorFlow
6. Keras
7. PyTorch
8. Pandas
9. Matplotlib

Among them, Scikit-learn has become the most popular Python machine learning library. Scikit-learn supplies a wide range of support for supervised and unsupervised learning algorithms. It also can be used for data mining and data analysis. The main machine learning algorithms that the Scikit-learn library can process are classification, regression, clustering, and dimensionality reduction.

TensorFlow or PyTorch is not just considered a Python machine learning library. Most Python machine learning developers consider them two significant machine learning frameworks. And Keras could run both as an API(application programming interface). TensorFlow and PyTorch have flexible and stable architectures which can run on different chips, including CPU, GPU, and TPU(tensor processing units)

Thanks to Python's easy-to-use advantage, productivity cross platforms, and efficient library support from communities, Python takes the leading role among machine learning development tools. Then we will introduce two machine learning frameworks, Caffe and Tensorflow.

1.3.2 Machine Learning Framework: Caffe

Caffe is an open-source deep learning framework. It was developed by the University of California, Berkley AI Research [7]. There is a lot of open-source documentation available on GitHub. It is widely used in the academic research project. It is written in C++ and has better support from Python and MATLAB. Caffe

Caffe is a flexible machine learning framework for many different machine learning models, especially supporting image classification and segmentation. Caffe could run on both CPU and GPU. It has taken advantage of other tech giants' assistance. Intel creates CPU-based acceleration computational kernel library MKL. And NVIDIA builds GPU-based cuDNN to improve the acceleration speed for GPU-based development environment.

Caffe has been more and more easily used by developers. It supports convolutional neural networks(CNN), long short-term memory(LSTM), and fully connected neural networks. In April 2017, Facebook updated Caffe to Caffe2. At the end of March 2018, Caffe2 was merged into PyTorch.It is very convenient to build neural networks with Caffe. We will explain the main inner architecture of Caffe using a simple neural networks example. Figure 1.7 stands the neural networks built with Caffe.

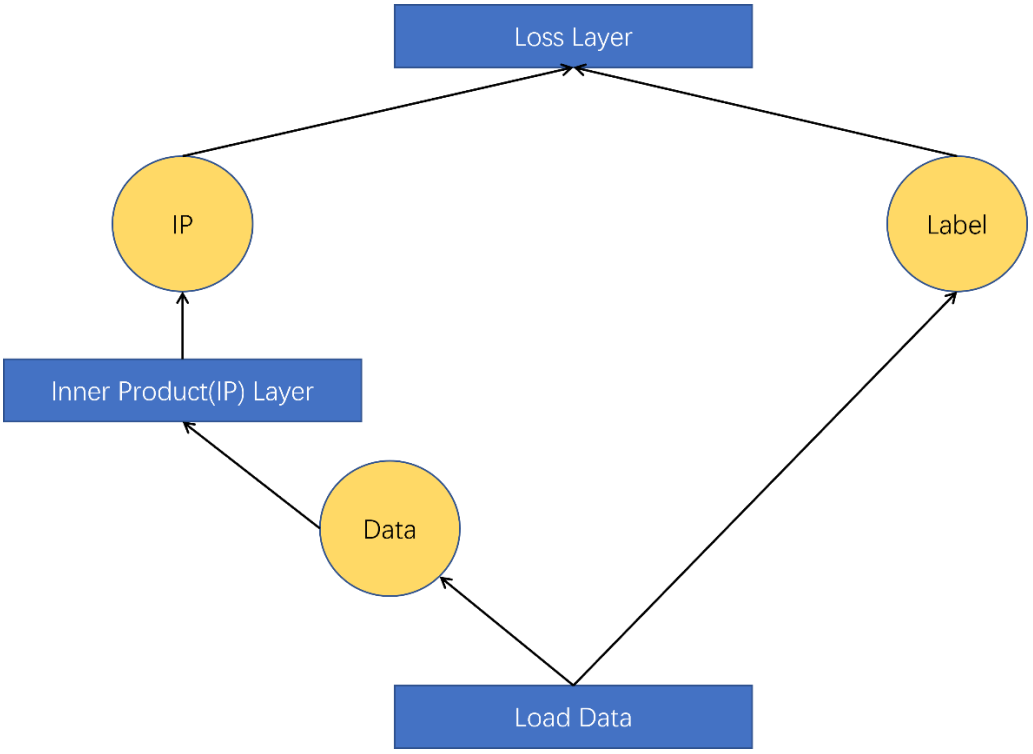


Figure 1.7 An Example of Neural Networks Built with Caffe.

Neural networks are built from bottom to top with Caffe. The architecture of the above neural networks is divided into four units. These terms are blob, layer, net, and loss. Caffe uses blob to store and transmit data. The data shape is an N-D array. Training and testing data, weights, and biases are all stored in blobs. The blob is also a bridge that links CPU and GPU. The data from the CPU is loaded into the blob, which is then passed to the GPU for computation. For large-scale data, LevelDB databases are used.

The layer is where computation happens in it. A set of layers and blobs connected together would create a net. A blob passes a layer as input. Then the layer will generate the corresponding output blob. Figure1.8 shows the layer protocol.

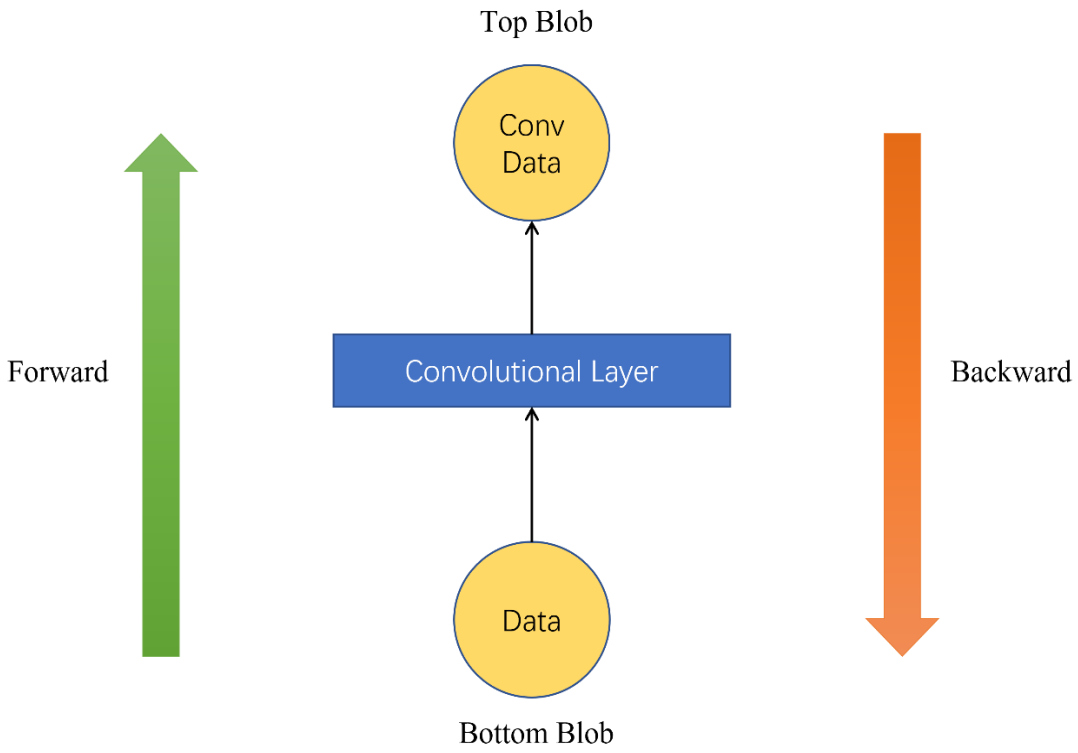


Figure 1.8 Caffe layer protocol.

A layer has the following operations, setup, forward pass, and backward pass. They are three basic concepts of a layer. We will introduce them step by step:

1. Setup: It initializes the parameters in a layer for the first time training a machine learning model. Caffe offers various layer setups, such as Convolution, Pooling, and nonlinear activations.

2. Forward pass: Inputs are passed and correspondingly outputs are generated.

3. Backward pass: This step computes gradients of output.

The loss could be thought of as a special kind of layer. Loss is at the end of the networks. Setting a suitable loss for a machine learning model is important because it defines the model type. If the model is used to make binary classification, the loss needs to be set to SoftMax with loss. For example, if the model is a regression model, we could use Euclidean Loss. Caffe supplies various losses to satisfy different machine learning models.

Also, developers could write their loss function for their models.

1.3.3 Machine Learning Framework: TensorFlow

TensorFlow is a powerful machine learning framework. It is a free and open-source software library. The Google Brain team developed TensorFlow. It was only used for Google internal researchers. Then Google released TensorFlow to the public in November 2015. From Google Trend, we know that TensorFlow attracted incredible developers in the following years.

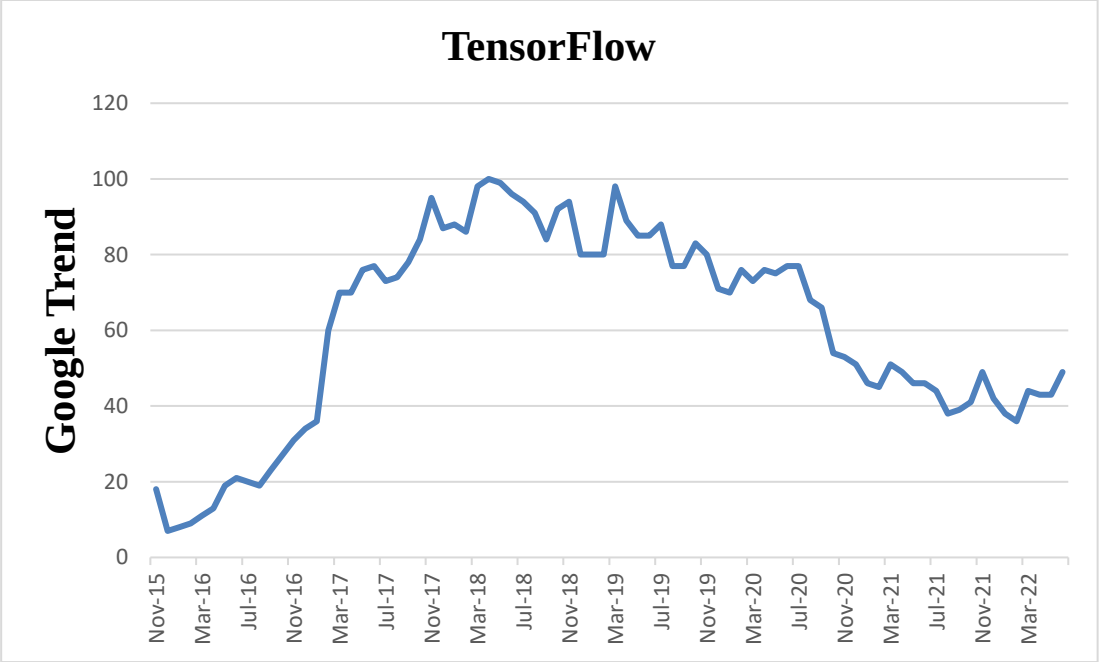
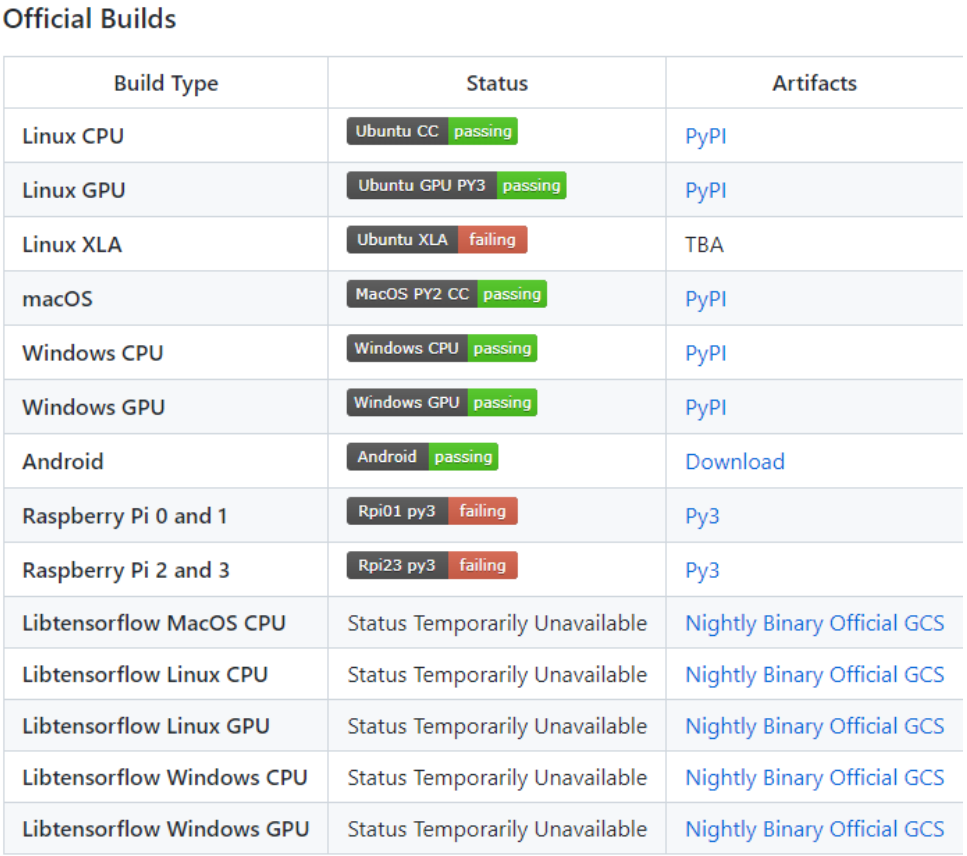


Figure 1.9 TensorFlow Google Trend index from Nov. 2015 to Jun. 2022.

TensorFlow is a breakthrough in the machine learning framework. Why? Because it develops amid the internet and mobile technology. TensorFlow works well with all popular programming languages such as Python, C++, Java, R, and Go.

Previous machine learning frameworks, such as Caffe, provided inadequate support for mobile computing platforms. According to the official TensorFlow GitHub account,

TensorFlow might be constructed on a variety of platforms. Thus, developers may construct TensorFlow on nearly all common platforms. That was previously impossible. Frameworks for machine learning were restricted to personal computers and cloud computing. However, TensorFlow may now be utilized for IoT (Internet of Things). Consequently, machine learning has more extensive uses. New concepts emerge, such as intelligent lighting and an intelligent outside camera.



Official Builds

Build Type	Status	Artifacts
Linux CPU	Ubuntu CC passing	PyPI
Linux GPU	Ubuntu GPU PY3 passing	PyPI
Linux XLA	Ubuntu XLA failing	TBA
macOS	MacOS PY2 CC passing	PyPI
Windows CPU	Windows CPU passing	PyPI
Windows GPU	Windows GPU passing	PyPI
Android	Android passing	Download
Raspberry Pi 0 and 1	Rpi01 py3 failing	Py3
Raspberry Pi 2 and 3	Rpi23 py3 failing	Py3
Libtensorflow MacOS CPU	Status Temporarily Unavailable	Nightly Binary Official GCS
Libtensorflow Linux CPU	Status Temporarily Unavailable	Nightly Binary Official GCS
Libtensorflow Linux GPU	Status Temporarily Unavailable	Nightly Binary Official GCS
Libtensorflow Windows CPU	Status Temporarily Unavailable	Nightly Binary Official GCS
Libtensorflow Windows GPU	Status Temporarily Unavailable	Nightly Binary Official GCS

Figure 1.10 Partial Community-Supported platforms for TensorFlow.

To use TensorFlow, it is important to clearly understand three major definitions. They are tensor, graph, and session. All TensorFlow codes contain these three parts.

What exactly is a tensor? Tensor is the data structure used by TensorFlow programs to represent all data types. Tensors could hold any type of information. A tensor is an array

with N dimensions. Different forms of scalar, vector, matrix, and high-dimensional array could be turned into tensors. Rank is the unit of dimension used to define tensors. Tensors may hold several forms of data, as described in Figure 1.10. The relationship between data type, tensor, and rank is described in Table 1.1.

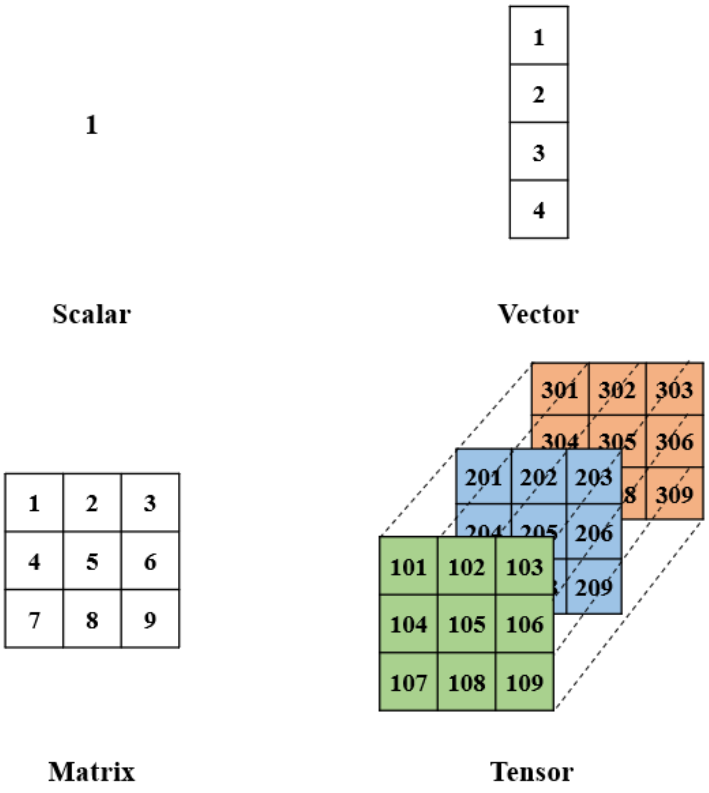


Figure 1.11 Different types of data stored in tensors.

Table 1.1 Relationship between Data, Tensor and Rank

Types of Data	Tensor	Rank
Scalar	0-D Tensor	0
Vector	1-D Tensor	1
Matrix	2-D Tensor	2
Tensors	N-D Tensor	N

The most unique aspect of TensorFlow is that its operations are represented by graphs. The graph consists of a collection of nodes linked by edges. Nodes serve two distinct purposes. A node is a location where one or more tensors are stored. A node could also perform tensor operations such as convolution computation. The edges indicate the movement of data. The best way to convey it is by a simple example. Suppose we want to calculate the result of the function listed below:

$$f(a,b,c) = ab + c + 1 \tag{1.1}$$

Let's now construct a graph for this function. The graph shown in Figure 1.12 was generated with TensorFlow program codes.

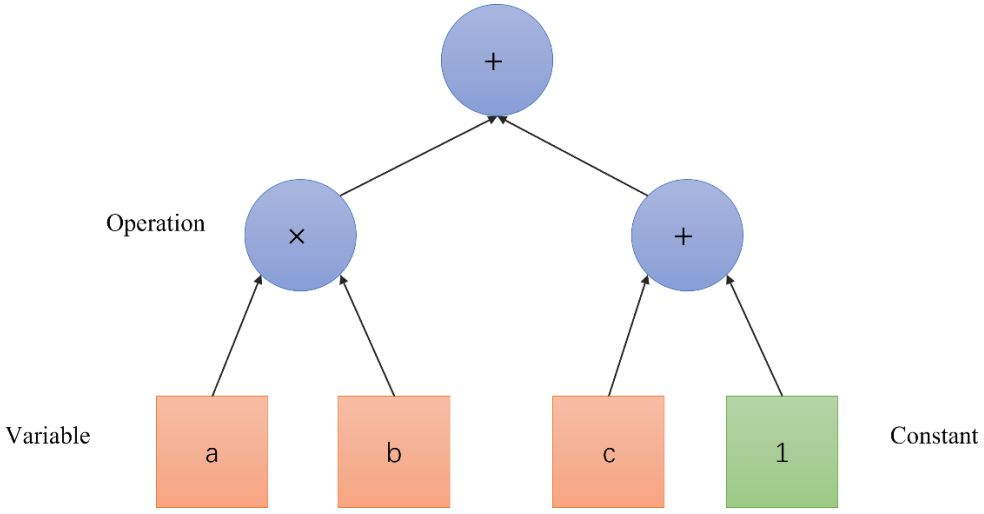


Figure 1.12 A simple graph built in TensorFlow program codes.

Tensorflow executes the graph operations. After constructing the graph, we can initiate a session. A session places graph operations on hardware like CPU and GPU. The majority of a session's execution depends on TensorFlow program codes. We will not go into detail on how to write the codes here.

1.4 Summary

This chapter begins with a brief overview of the evolution of neural networks. From the perceptron to the massive neural networks of today, the architecture of neural networks has become progressively complex. The development of hardware and software aids machine learning scientists in designing complicated machine learning models. Previously, hardware limitations were a significant drawback that reduced computing capability. Machine learning specialists may now deploy machine learning models using distributed and parallel computing. Additionally, software that can be programmed is becoming more user-friendly for both beginners and professional developers.

1.5 Outline of Dissertation

The remaining sections of this dissertation are formatted as follows: The second chapter demonstrates how to construct a convolutional network model. The third chapter describes the use of CNN models to detect seam carving operations on images. In Chapter 4, it is demonstrated that a CNN model can recognize various Gaussian filtering processes applied to images. In Chapter 5, a prototypical network model for detecting various Gaussian and average image filtering processes is shown.

CHAPTER 2

A ROADMAP FOR BUILDING CONVOLUTIONAL NEURAL NETWORKS

In earlier chapters, we presented a brief history of artificial neural networks and the fundamental development tools for machine learning. Step-by-step instructions for constructing convolutional neural networks are provided in this chapter.

In previous chapters, we have introduced the brief history of artificial neural networks and the essential development tools for machine learning. In this chapter, we will discuss how to build convolutional neural networks step by step.

2.1 Foundational Math behind Artificial Neural Networks

There are three different types of machine learning. They are supervised learning, unsupervised learning, and reinforcement learning. Figure 2.1 summaries the differences between them.

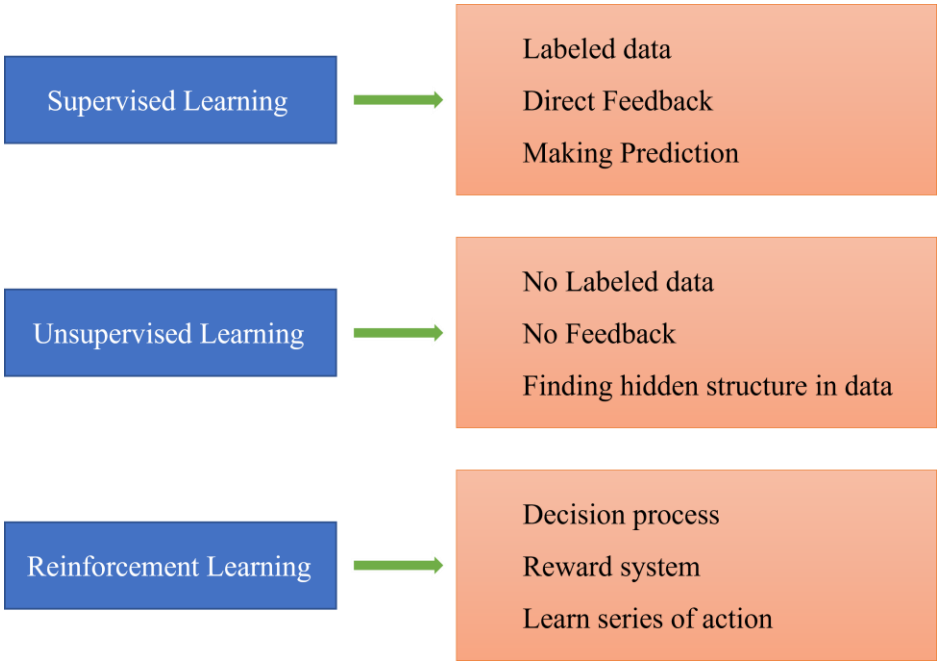


Figure 2.1 Three different types of machine learning.

The primary objective of supervised learning is prediction. Using labeled data, the supervised machine learning model is trained. Then, upon training, the supervised machine learning model could predict the label for unseen new data given the data's label. Figure 2.2 shows a typical procedure for supervised learning. The labeled training data is sent to a supervised machine learning algorithm with the purpose of fitting a predictive model capable of making predictions on new, unlabeled data. As the supervising machine learning algorithm, convolutional neural networks are used here.

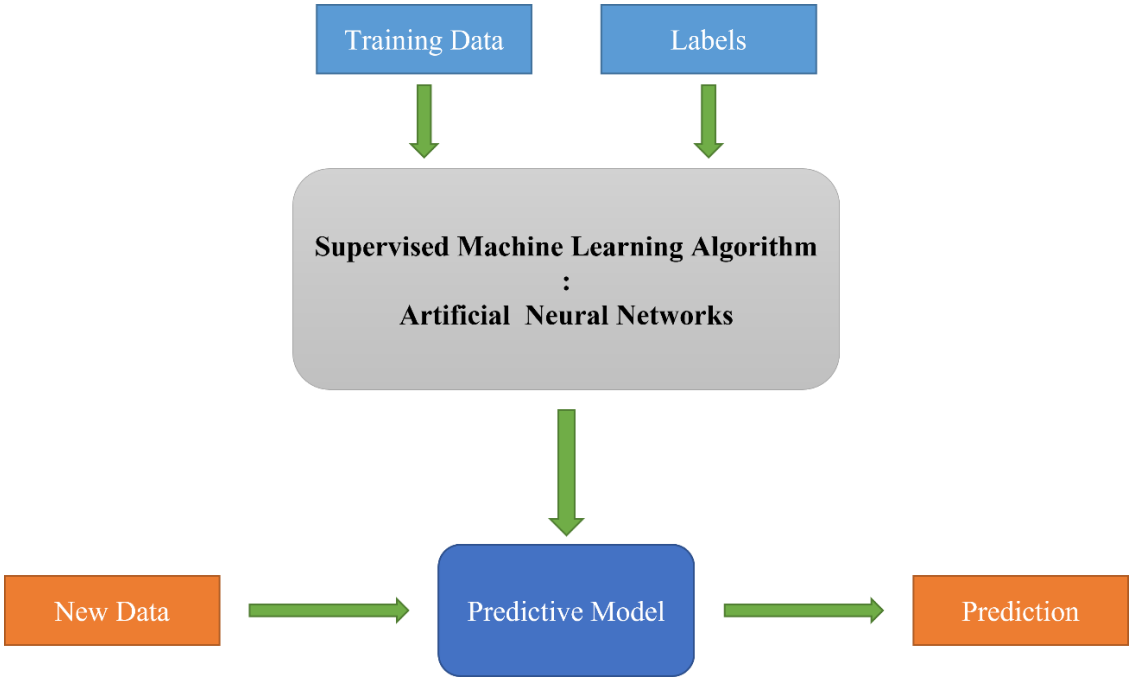


Figure 2.2 A typical workflow of artificial neural networks.

A classification task is a type of supervised learning problem that has discrete class labels. Recognizing cat and dog is an example of a traditional binary classification task. The number of classes does not have to be two. The number of classes is highly task dependent. Recognizing handwritten characters is an example of a common multiclass classification task. Regression is a subfield of supervised learning in which the output signal is a continuous value. The purpose of each convolutional neural network model utilized in this dissertation is classification. We will therefore concentrate on constructing convolutional neural networks for classification problems.

In Chapter 1, the fundamental structures of a single perceptron and multilayer perceptron were discussed. This chapter will begin with an explanation of the formal definition of an artificial perceptron before introducing convolutional neural networks in depth.

Let us construct a simple artificial neuron for binary classification. Our dataset $\boldsymbol{X}$

comprises both positive and negative values. A sample in dataset $\boldsymbol{X}$ is represented by x. Each input sample x is converted into an n-dimensional vector $\boldsymbol{x}$ Each input vector $\boldsymbol{x}$ has a corresponding weight vector **w**. The net input is defined as z:

$$z = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n \tag{2.1}$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.2}$$

We can then define a decision function to predict the class of the sample. If $\emptyset(z)$ is greater than a defined threshold θ, we predict class 1; otherwise, we predict class -1. In the perceptron algorithm, the decision function $\emptyset(z)$ is a unit step function.

$$\emptyset(z) = \begin{cases} 1, & if\ z \geq \theta \\ -1, & otherwise \end{cases} \tag{2.3}$$

For simplicity, we can bring the threshold, θ , to the left side of the equation and define a weight-zero as $w_0 = -\theta$ and $x_0 = 1$ so that we write z in a more compact form:

$$z = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = \boldsymbol{w}^T\boldsymbol{x} \tag{2.4}$$

$$\emptyset(z) = \begin{cases} 1, & if\ z \geq 0 \\ -1, & otherwise \end{cases} \tag{2.5}$$

Now we can find z = $\boldsymbol{w}^T\boldsymbol{x}$ is a linear function. It will always have the opportunity to cross origin. This could be undesirable. Figures 2.2, 2.3 and 2.4 give further details for visual depiction.

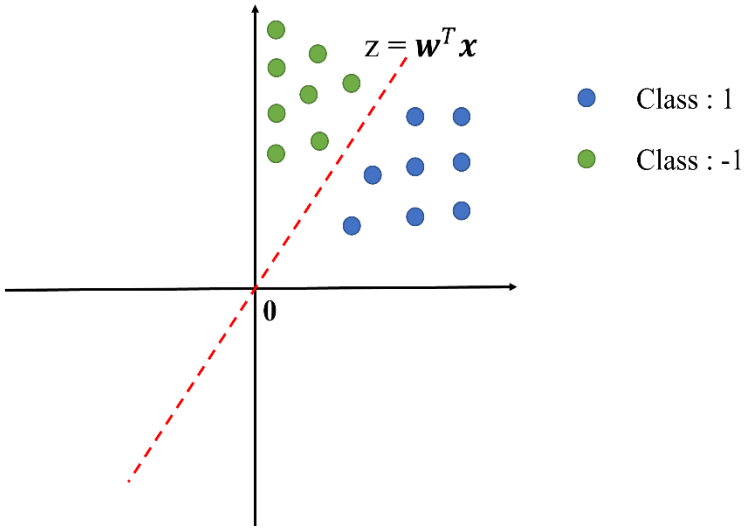


Figure 2.3 An ideal boundary line is made by z = $\boldsymbol{w^T x}$.

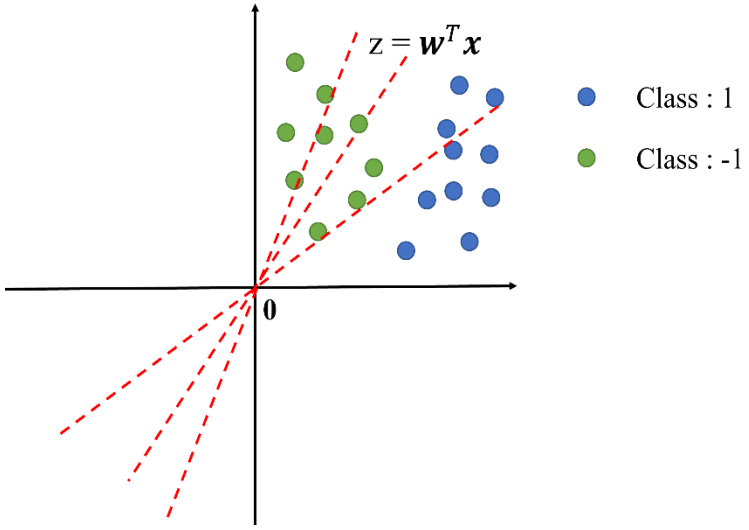


Figure 2.4 An ideal boundary line can never be made by z = $\boldsymbol{w^T x}$.

Figures 2.3 and 2.4 illustrate the various data distributions. Class 1 is denoted by blue points, while class -1 is shown by green points. In Figure 2.3, an ideal boundary is made by z = $\boldsymbol{w^T x}$. However, in Figure 2.4, no matter how we adjust, an ideal boundary cannot be created by z = $\boldsymbol{w^T x}$. Since z = $\boldsymbol{w^T x}$ will always passes through the origin, the range for adjusting the boundary line to distinguish two distinct data classes is constrained.

Now we add a constant value called bias to z = $\boldsymbol{w}^T\boldsymbol{x}$. It is represented by the small letter b. Then the function will change to:

$$z = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b = \boldsymbol{w}^T\boldsymbol{x} + b \qquad (2.6)$$

Following the new function 2.6, we can find the ideal boundary line under the data distribution in Figure 2.4, as shown in Figure 2.5. The range to adjust the boundary line extends. This will result in a more precise classification.

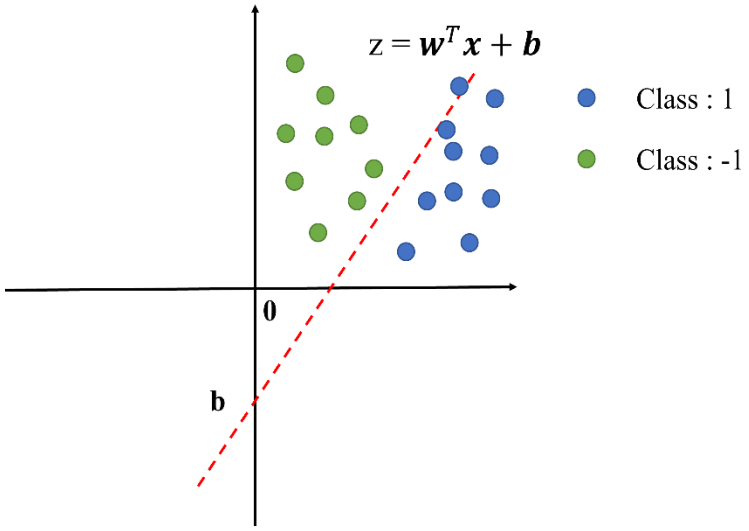


Figure 2.5 An ideal boundary line is made by z = $\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$.

Function 2.6 is the foundational math behind artificial perceptron. Connecting a lot of perceptron into a network shape we will build an artificial neural network. We shall construct an artificial neural network by connecting many perceptrons into a network structure. Matrix multiplication is the fundamental operation of artificial neural networks, as revealed by Function 2.6. The following section will demonstrate how to construct convolutional neural networks.

2.2 Building Convolutional Neural Networks

Initial inspiration for convolutional neural networks (CNN) came from how the visual cortex of the human brain recognizes objects. In 1995, Yann LeCun proposed the LeNet [8] family of convnets trained to recognize MNIST handwriting characters. It has a significant impact on CNN's further development. Since 1989, Yann LeCun and his colleagues have made significant contributions to the development of artificial neural networks [9-12]. In 2019, Yann LeCun earned the most prestigious award in computer science, the Turing Award. CNN has outstanding performance for image classification tasks. Nowadays, CNN is widely implemented for image recognition, computer version, textual documents analysis and so on. In the following sections, we will discuss the advantages of CNN. Then we will delve into operations in a typical CNN architecture.

2.2.1 Feature Hierarchies of CNN

The key to the performance of any machine learning algorithm is to extract important features from datasets. Traditional machine learning methods rely on domain-expert-provided input features. It takes a significant amount of time to manually identify important features.

CNN can automatically learn features from raw data. Earlier layers extract low-level features from raw data. CNN procedures will result in the formation of high-level features from low-level features. The subsequent layers, such as the fully connected layers, will then utilize these features for prediction. Combining low-level features to form higher-level features is referred to as feature hierarchy. This dissertation mainly focuses on the implementation of CNN for image-related problems. We will use an example to demonstrate how CNN extracts visual features.

When using CNN to identify a cat in an image, edges and blobs are taken from prior layers. These features ate low-level feature. Then, by combining these low-level features, we will create much more complex shapes, such as the head and body of a cat. These complex shapes are known as high-level features.

CNN extracts feature maps from an input image. As you can see in the following Figure 2.6. Each element in feature map comes from a local patch of pixels in the input image.

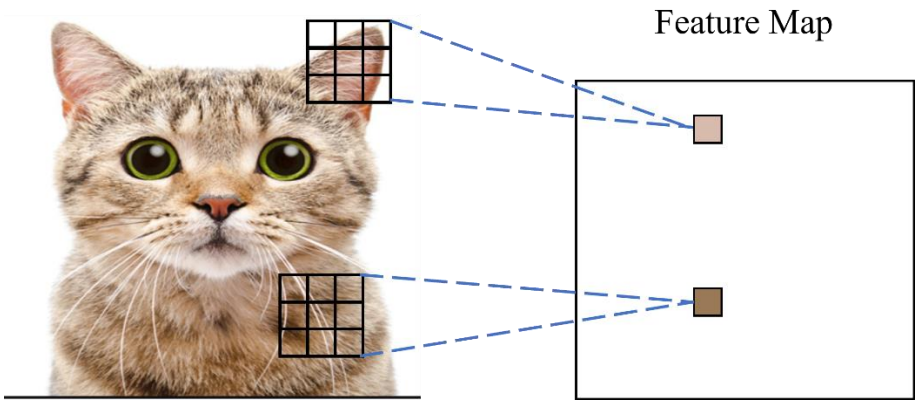


Figure 2.6 A feature map extracted by convolutional layers.

As explained in the previous sections, matrix multiplication is the fundamental operation of conventional neural networks. This operation is replaced with a convolution operation in a CNN, as illustrated below:

$$\mathbf{Z} = \boldsymbol{W} * \boldsymbol{X} + b \tag{2.7}$$

W: It is the weighted matrix. It is also called kernel matrix.

X: It is a matrix representing the pixels in a $height \times width$ region.

b**:** It is the bias.

As shown in Figure 2.7, we have an input image $I_{6\times6}$, and a kernel matrix $\mathbf{W}_{3\times3}$. The input matrix $X_{3\times3}$ has the same dimensions as the kernel matrix. It is really different from fully connected neural networks. In fully connected neural networks, the input matrix has the same dimensions as the entire image. Still using input image $I_{6\times6}$, the input matrix should be a 6×6 matrix rather than a 3×3 matrix.

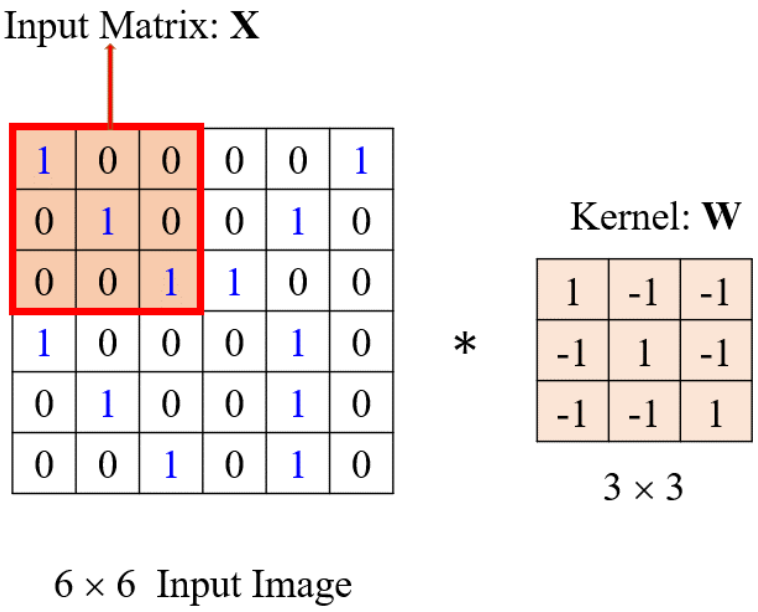


Figure 2.7 Input image, input matrix and kernel matrix.

Now, let's connect one input layer to one convolutional layer to describe the aforementioned convolution operations. The input layer is where the image is input. It converts an image to a matrix. We represent each value in the input matrix with a single node. The convolutional layer then has many kernels. That each kernel will generate a feature map from the image input. We express each weight in a kernel using edges. After convolution operations, the output Z consists of the convolutional layer's values. Figure 2.8 illustrates the transfer of an input image to an input layer. Figure 2.9 shows the relationship between the input layer and the convolutional layer.

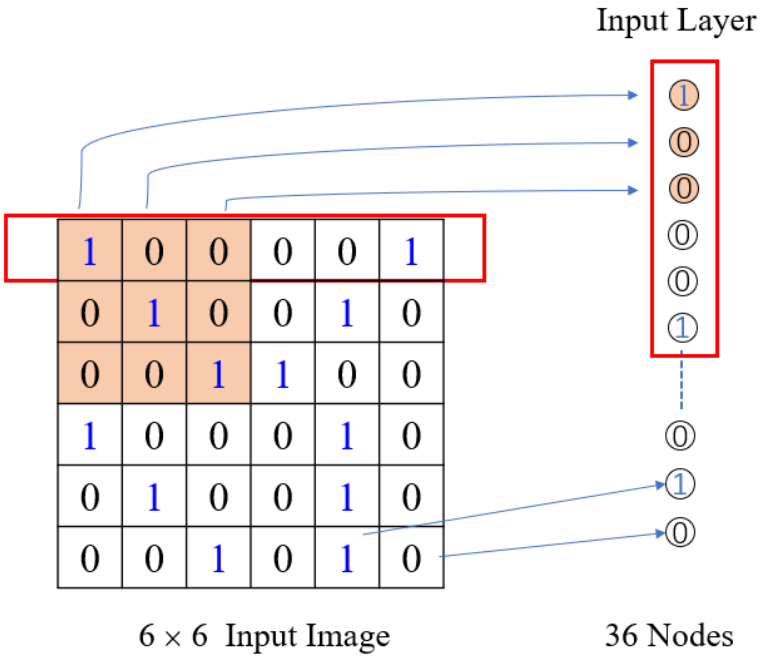


Figure 2.8 Building input layer from an input image.

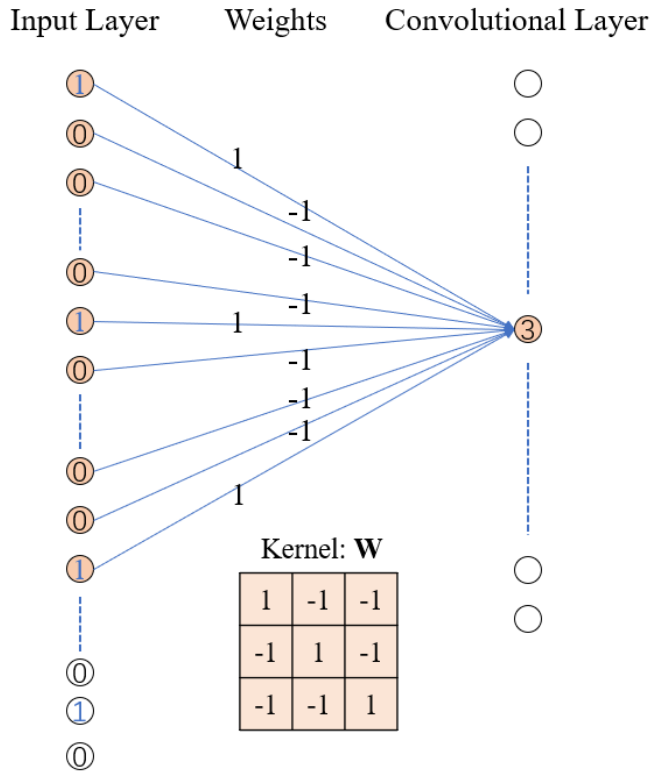


Figure 2.9 An input layer connected to a convolutional layer.

Here are two fundamental concepts that help CNN perform better than ordinary neural networks. The first is referred to as sparse connection. In other words, a node in one layer is not connected to all of the nodes in the previous layer. As shown in Figure 2.9, a convolutional layer node is only connected to nine input layer nodes. If the layer is fully connected, a node should connect 36 input layer nodes. Another benefit is known as parameter-sharing. The same weights are applied to distinct regions of the input image. In Figure 2.9, input matrix **X** shares common kernel **W**. They are marked by colored background.

As previously discussed, convolutional processes extract feature maps. Each node in the convolutional layer in Figure 2.9 corresponds to a value in the feature map. Now let's conclude our explanation of how accurate convolutional operations are for feature maps.

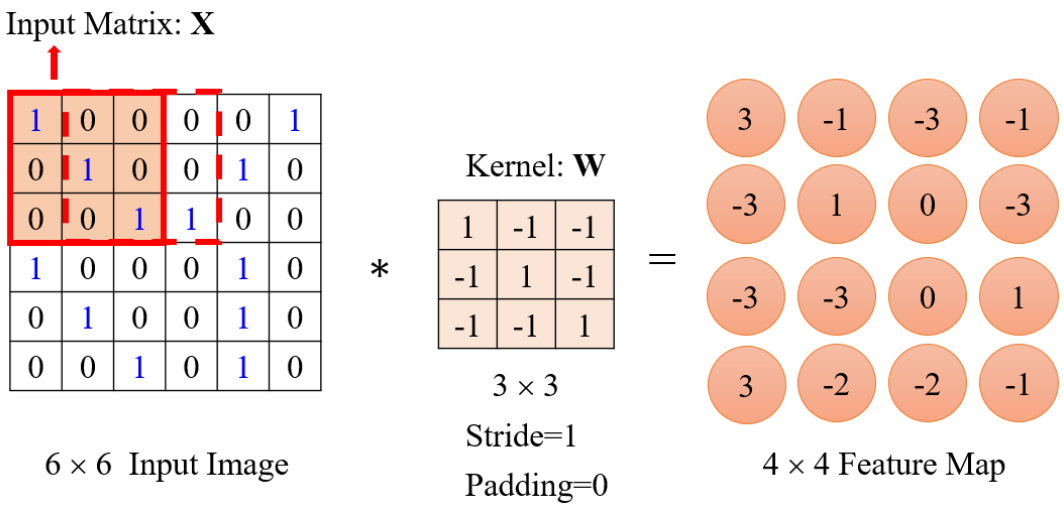


Figure 2.10 Extracting feature map by convolutional layer.

Given the input image **I** and the kernel **W** in the previous example, we have the convolved output. The 3×3 kernel **W** (also referred to as the filter) is multiplied elementwise with the input matrix **X** to produce one output matrix value. The remaining values are obtained by sliding the window across the image. There are three factors to

define a kernel:

Kernel size: It is represented by $height \times width$. It defines the covered region on image or feature map.

Stride: It is the step size of the kernel when it slides through the image.

Padding: Defines how the border of image or feature map is captured.

2.2.2 Common Layers Used for Building CNN

We've talked about convolutional layers and fully - connected layers. In this section, we will go through some of the most frequent layers utilized in CNN construction. There are three of them: the pooling layer, the activation layer, and the output layer.

Pooling layer decreases the dimensions of feature maps, resulting in increased computing efficiency. Additionally, lower feature map size assist prevent overfitting. CNN has two primary types of pooling operations: maximum pooling and average pooling. Similar to the kernel, we must define the region and stride to determine where the maximum or average operation is executed. The region is known as the pooling size. The pooling layer is usually defined as $P_{W \times H}$. Figures 2.11 and 2.12 illustrate the operation of maximum and average pooling, respectively.

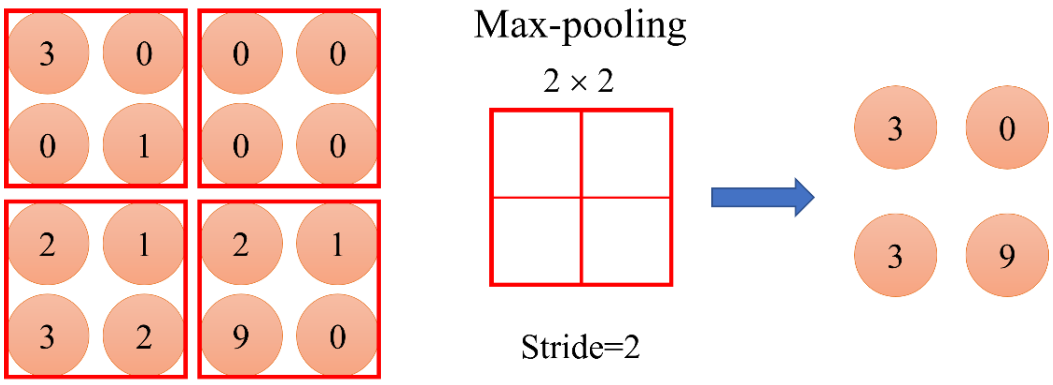


Figure 2.11 An example of max-pooling.

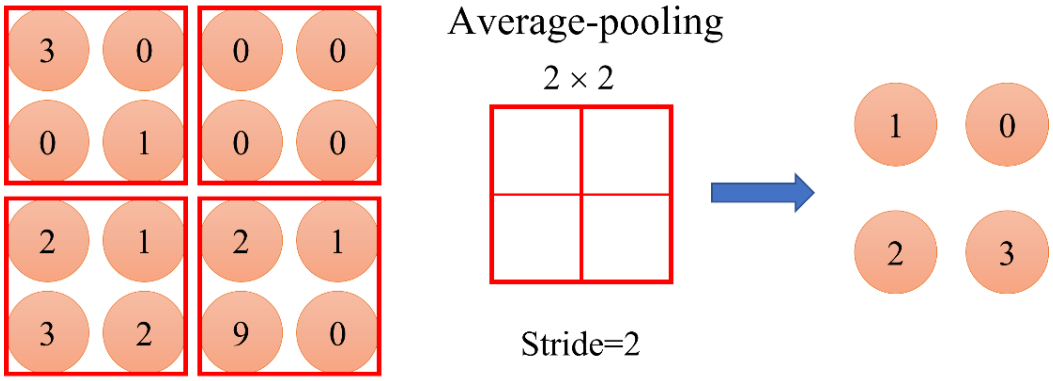


Figure 2.12 **A**n example of average-pooling.

CNN's activation layer is a crucial component of its design. Activation layers add extra nonlinear factors to CNN, allowing it to solve complicated problems more effectively. There are three typical activation functions used in CNN architecture for building activation layers. They are the Sigmoid, the TanH, and the ReLU. Their functions and figures are illustrated below.

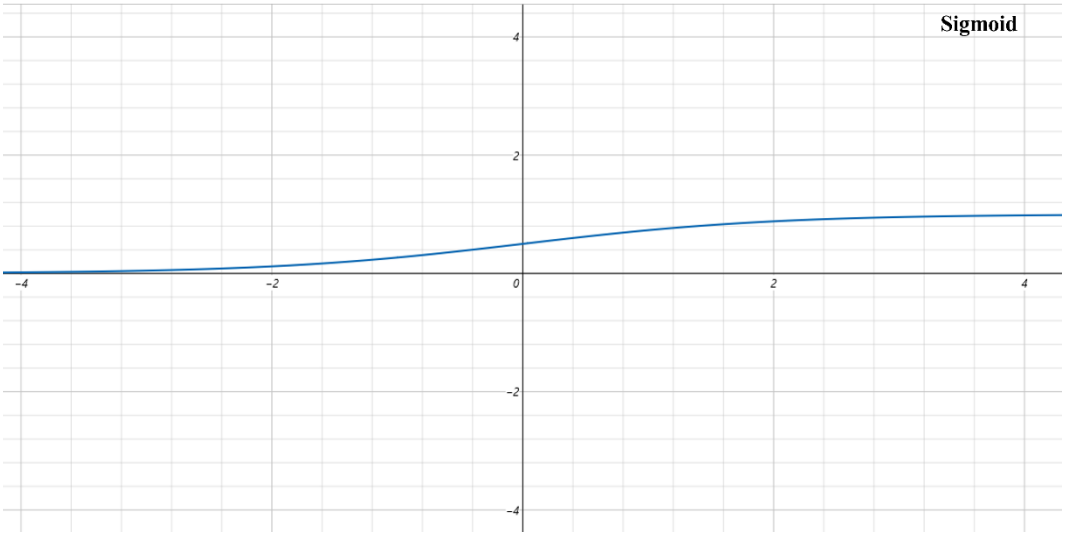


Figure 2.13 Sigmoid activation function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \text{.....(2.8)}$$

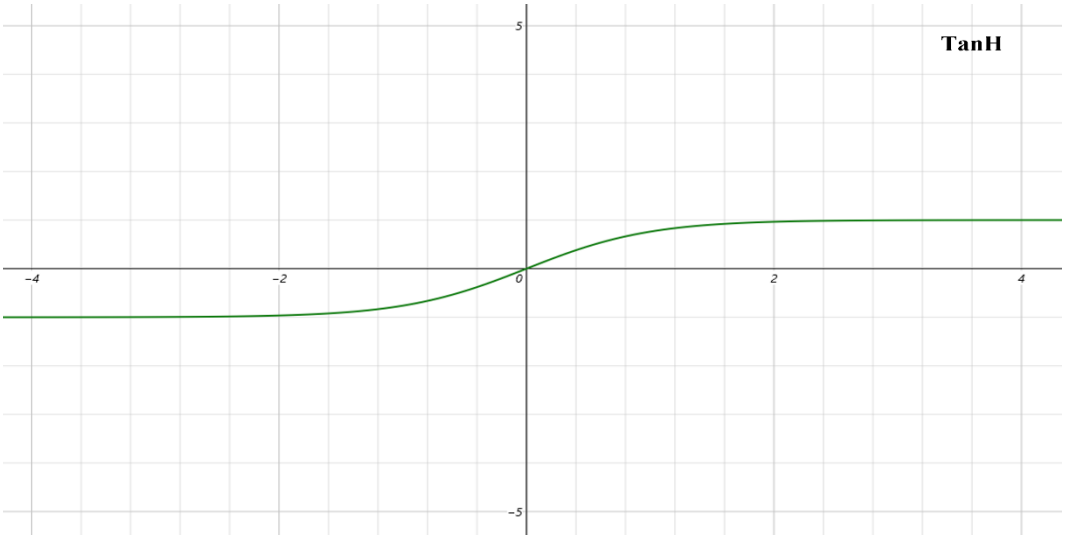


Figure 2.14 TanH activation function.

$$TanH(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \text{.....(2.9)}$$

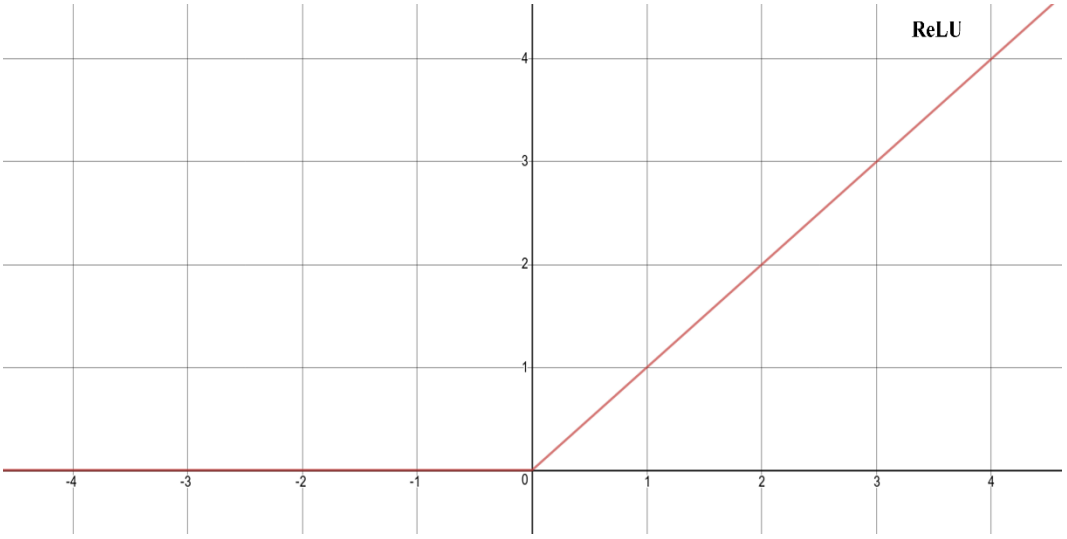


Figure 2.15 ReLU activation function.

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x \geq 0 \end{cases} \tag{2.10}$$

In CNN, the final layer is the output layer. Output layer will compute the outcome of the prediction. The outcome of the prediction could be logits or probabilities. However, how can we define a "good" output? We must create a loss function that penalizes outputs that deviate significantly from the predicted value. In CNN, the computation of outputs from inputs is referred to as forward propagation. After calculating the outputs, the outputs and loss function are used to update the CNN weights in order to improve CNN's performance. Since this process starts at the output layer and propagates backward through the CNN, we refer to it as backward propagation. In terms of classification issues, forward propagation and backward propagation are summarized as follows:

1. Initialize the weights to 0 or small random numbers.
2. For each training example $\boldsymbol{x}^{(i)}$ with label $\boldsymbol{y}^{(i)}$:
3. Compute the predictive label, $\hat{\boldsymbol{y}}^{(i)}$.
4. Define the loss function J($\boldsymbol{W}$)

5. Update the weights by using loss function.

We want to minimize the loss function. This indicates that the inaccuracy between the predictive output label $\hat{\boldsymbol{y}}^{(i)}$ and the actual label $\boldsymbol{y}^{(i)}$ will be minimal. The loss function should be convex when defined. We can calculate the partial derivatives of the loss function for each weight W_j, then identify the least loss value at this point. This powerful optimization process is known as gradient descent, as depicted in Figure2.16.\

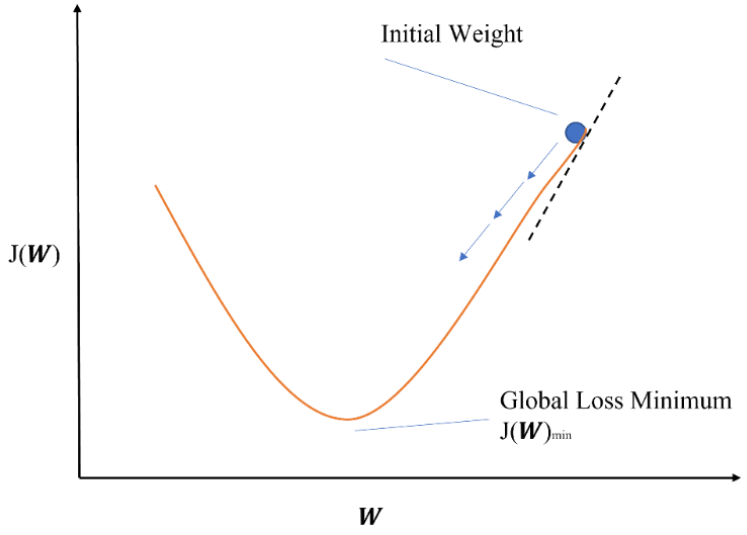


Figure 2.16 Gradient descent.

Using gradient descent, now we can update the weights by taking a step in the opposite direction. Also, we set a learning rate η to control how quickly the model is adapted. Usually, the learning rate is a constant between 0 to 1.

$$W := W + \Delta W \tag{2.11}$$

For each weight W_j :

$$W_j := W_j + \Delta W_j \tag{2.12}$$

$$\Delta W_j = -\eta \nabla J(W_j) \tag{2.13}$$

$$\Delta W_j = -\eta \frac{\partial J(W)}{W_j} \tag{2.14}$$

Due to the large number of parameters in CNN, updating all weights typically requires considerable time. CNN's performance cannot be significantly enhanced by updating all weights only once. The updating process may be repeated thousands of times in order to achieve a stable and satisfactory experimental result.

2.3 Summary

This chapter presents a roadmap to building a convolutional neural networks (CNN) model. At the beginning of this chapter, we illustrate the fundamental math behind CNN. Then we explain the basic functions of some typical layers in CNN. We could build a CNN model by organizing these typical layers into a hierarchical structure. At the end of this chapter, we explain how to update the parameters in the CNN model. Up to here, we have introduced the prior knowledge for CNN. Then in the following chapters, we will design various CNN models for some digital image forensics tasks.

CHAPTER 3

A CONVOLUTIONAL NEURAL NETWORK BASED SEAM CARVING DETECTION SCHEME FOR UNCOMPRESSED DIGITAL IMAGES

3.1 Introduction

Due to the rapid development of image-editing techniques in the past years, digital images can be easily edited or tampered with popular software such as Photoshop. To reveal malicious image editing, digital image forensics [13] have been extensively studied for the past decade. In this chapter, we present a novel forensic approach to detect the operation of seam carving [14] in digital images, specifically in uncompressed images. Seam carving, also known as content-aware scaling, is one popularly utilized image scaling algorithm and has been included in many predominant image editing software, such as Photoshop and GIMP. By recursively deleting a seam (a horizontal or vertical path of 8-connected pixels) with the lowest energy, the image size is altered, and the visually more important image contents can be well-preserved.

A few forensic works have been reported in the past several years to reveal traces of seam carving in digital images. In the first piece of forensic work for seam carving detection [15], Sarkar et al. proposed to utilize Markov transition probability to reveal the trace of seam carving in digital images, specifically in JPEG compressed images. Later in [16], a hybrid statistical feature model was proposed by Fillion et al. to track the operation of seam carving in uncompressed images based on energy distribution, seam behavior and wavelet absolute moments. In [17], Lu et al. proposed an active forensic approach to determine whether a received uncompressed image has been attacked by seam carving or not by comparing the SIFT features pre-extracted by the sender with the SIFT features

extracted at the receiver end. Chang et al. [18] later presented a series of statistical features based on the blocking artefact characteristics matrix to differentiate non-seam carved JPEG images from seam carved JPEG images. This work was further extended in [19]. In [20], Liu et al. proposed to employ the calibrated neighboring joint density of DCT coefficients for the detection of seam carving in JPEG images, and the extended works were reported in [21, 22]. In Ryu et al.'s work [23], the authors designed a set of features based on energy bias and noise level to unveil the operation of seam carving in uncompressed images. In [24], Wei et al. presented an interesting approach to detect seam carving in uncompressed images. By dividing images into 2×2 mini-squares and categorizing each of the squares into nine types of predefined patches, each square was possibly recovered to its original form. Then, Markov transition probability was applied to discriminate seam carved images from non-seam carved images. Yin et al. [25] proposed a blind forensic technique to detect seam carving in uncompressed images based on the similar idea pro-posed in [23]. In [25], twenty-four features consisting of six newly designed features and eighteen features proposed in [23] were extracted from the local binary pattern pre-processed images for seam carving detection in uncompressed images. In [26], Ye and Shi proposed to employ a set of energy features which extracted from local derivative pattern encoded images to identify seam carved images. In [27], an advanced statistical model, consisting of local derivative pattern, Markov transition probabilities, and subtractive pixel adjacency model, are designed to determine if an image has been gone through seam carving or not. The extended work of [26, 27] was presented in [28]. In [29], Zhang et al. designed forty-two features to unveil the statistical properties of spatial and spectral entropies (SSE). They were combined with local binary pattern (LBP)-based energy features to detect seam

carving image with low scaling ratio.

Most of the existing methods for seam carving detection as introduced, except [17], are focusing on feature engineering, a Support Vector Machines (SVM) based classification scheme is applied to ensure better performance. In this chapter, inspired by the substantial successes achieved by convolutional neural networks (CNN) in computer vision [30-32], and the success obtained by the CNN-based steganalysis work [33], we propose and report a CNN architecture that includes both the feature extraction and classification in a joint optimization framework to unveil the process of seam carving in uncompressed digital images. As far as we know, this is the first work that successfully applies deep learning for seam carving detection. Furthermore, as indicated by experimental results, the proposed approach achieves almost perfect results at higher scaling rates, and largely outperforms the state-of-the-art at lower scaling rates. The rest of the chapter is organized as follows: In Section 2, seam carving is briefly introduced. Then, the proposed CNN structures are described in Section 3. The experimental results are reported in Section 4. The conclusion is made in Section 5.

3.2 Background of Seam Carving

The image scaling is a process to resize a digital image so as to satisfy certain geometric requirement. However, the conventional image scaling schemes could not always provide a promising visual quality after resizing because the image content is not considered carefully by these algorithms. One example is shown in Figure 3.1. As a result, seam carving is designed to protect image content from being destroyed while scaling is conducted.

For a given energy function $e(\,.\,)$, e.g., gradient, the importance of a pixel in image I can be evaluated with its energy as shown below,

$$e\big(I(x,y)\big) = \left|\frac{\partial}{\partial x}I(x,y)\right| + \left|\frac{\partial}{\partial y}I(x,y)\right| \tag{3.1}$$

where x and y are the corresponding row and column coordinates, respectively. By assuming the less important image content consists of lower energy pixels, seam carving is to delete a seam with the lowest cumulative energy recursively so as to alter the size of a given image. Note that a seam is a path of 8-connected pixels crossing the image either from top to bottom (vertical seam), or from left to right (horizontal seam). For instance, a horizontal seam s^H in an n × m (height × width) image I can be defined as:

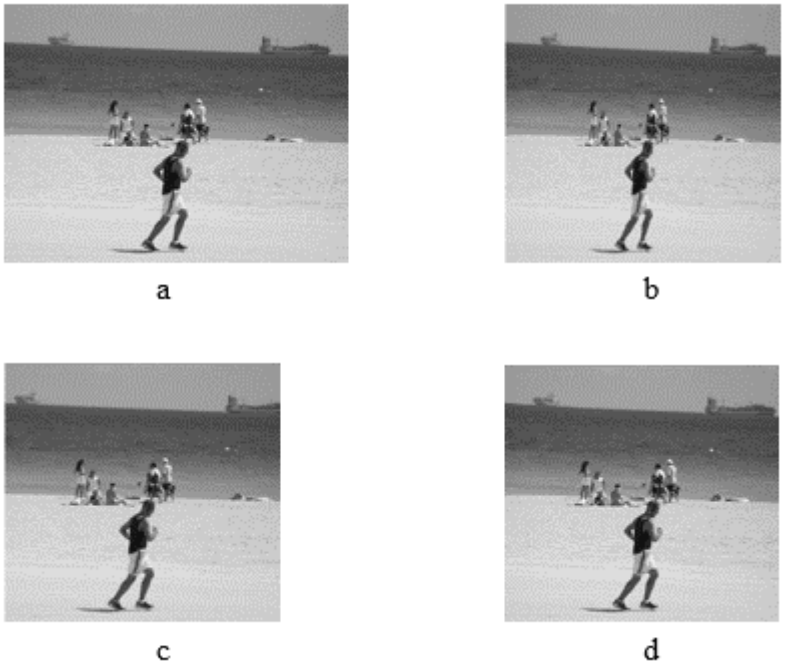


Figure 3.1 (a) An original image from UCID with a size of 384×512. (b), (c) and (d) are the resized copies of (a) with the same size of 384×411 but processed by different scaling techniques respectively: (b) bilinear interpolation, (c) cropping, (d) seam carving.

$$s^H = \{s_i^H\}_{i=1}^m = \{(x(i), i)\}_{i=1}^m, s.t. \forall i, |x(i) - x(i-1)| \leq 1 \quad (3.2)$$

where s_i^H represents the coordinates of each included pixel. Therefore, the optimal horizontal seam s^* can be shown below,

$$s^* = \min_s E(s) = \min_s \sum_{i=1}^m e\left(I(s_i^H)\right) \quad (3.3)$$

where $E(s)$ is the cumulative energy of seam s. As the optimal seam always has the lowest cumulative energy, it is considered to be the least visually important and unnoticeable in the image. Therefore, by removing multiple such optimal seams, either horizontal seams or vertical seams, not only can the image size be altered, but also the important image content could be well-preserved consequently.

3.3 Proposed CNN Architecture

CNN has aroused tremendous interests since a remarkable success was achieved in the ILSVRC-2012 competition by utilizing this advanced artificial intelligence technology [30]. A typical CNN hierarchical architecture starts with multiple stages of convolutional modules and ends with a classification module. A common convolutional module includes a convolutional layer, an activation layer, and a pooling layer. The convolutional layer is a trainable filter bank which can be considered as a feature extractor. The activation layer brings non-linearity to the network and bounds the extracted features. The pooling layer reduces the quantity of features extracted from immediately prior convolutional layer to avoid overfitting. By stacking a series of convolutional modules, hierarchical feature maps are extracted and then fed into the classification module composed of one or more fully connected layers, and the SoftMax layer with cross-entropy loss. The classification module can transform feature vectors to output probabilities for each class. Through back-

propagation, weights and biases in convolutional layers will be optimized so as to reduce the training loss, and the power of the network will then be enforced to predict the labels of unseen data.

The overall architecture of the proposed CNN is illustrated in Figure 3.2. Instead of directly feeding the original images into the network, a high-pass filtering (HPF) layer with kernel size of 5×5×1 (height × width × number of input feature maps) [33] is employed to pre-process input images. In this way, we use the first convolutional layer of CNN model as a pre-processing module. The trace of seam carving, i.e., imperceptible discontinuity of image content, is a kind of weak high frequency signal, which is greatly impacted by image content. Therefore, high-pass filter is employed at the beginning so as to boost the signal-to-noise ratio. This can provide a good initialization to drive the whole network, hence achieve good performance as compared to without doing it.

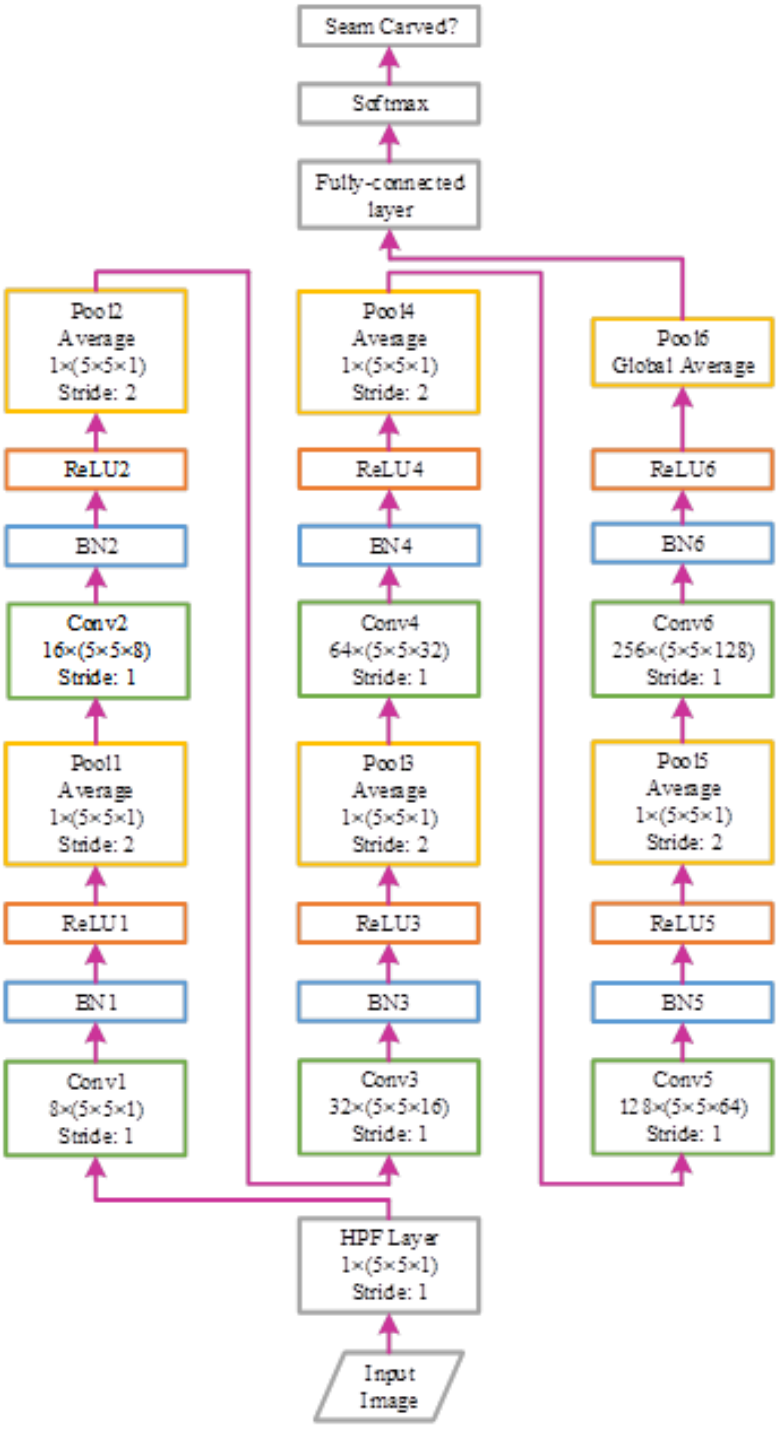


Figure 3.2 The proposed CNN architecture. Parametric setting of each layer is included in the corresponding box.

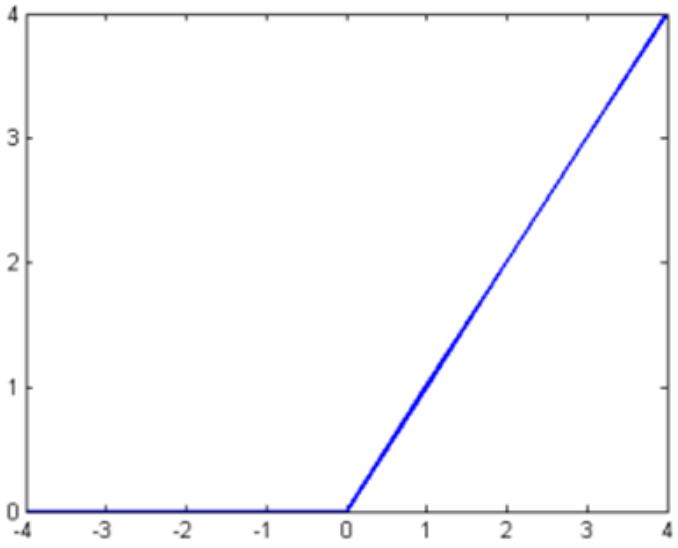
Following the HPF layer is the CNN hierarchical structure which consists of six convolutional modules and one fully- connected linear classification module. In the first convolutional layer (Conv1), the input, i.e., the pre-processed input image, is to be filtered by 8 kernels of size 5×5×1 each. In the following convolutional layers (Conv2 – Conv6), there are 16 kernels of size 5×5×8 in Conv2, 32 kernels of size 5×5×16 in Conv3, 64 kernels of size 5×5×32 in Conv4, 128 kernels of size 5×5×64 in Conv5 and 256 kernels of size 5×5×128 in Conv6 respectively so as to generate hierarchical feature maps.

Different from the conventional CNN module as introduced in [30], an additional layer, called batch normalization (BN) layer [35], is employed between each convolutional layer and the following activation layer. As the outputs generated by the convolutional layer are normalized by the corresponding BN layer, the so called 'internal covariate shift' [35] is reduced which helps to accelerate the training speed and to reduce the influence caused by poor initialization.

To increase the non-linearity of the proposed deep architecture, rectified linear units (ReLU) are served as the non- linear activation functions in each of the convolutional modules, as shown in Figure 3.3. Comparing with other popular non-linear functions, such as hyperbolic tangent and Sigmoid, ReLU has relatively simple form, i.e., gradient is 1 for positive inputs and 0 for negative inputs. Such characteristics could accelerate the speed on training deep neural networks, and also avoid the vanishing of gradient happens during the training stage [36].

Since the process of seam carving will remove lower energy pixels, those higher energy pixels which normally have large intensity value are more likely remained in the image. Due to this characteristic, focusing on the maximum pixel value of a local region

which is normally considered in computer vision intuitively insufficient to discover the trace of seam carving. Therefore, average pooling is employed in the proposed CNN framework for spatial sub-sampling instead of max pooling popularly utilized in computer vision. In the last pooling layer, namely, Pool6, the kernel size for pooling is fixed to the spatial size of the input feature maps. Each input feature map will be aggregated to one single number, which serves as a feature for the classification. As there are 256 input feature maps to Pool6, 256 features are generated and fed into the fully-connected linear classification module for each image.



$$ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

Figure 3.3 Rectified linear unit (ReLU).

3.4 Experimental Results

Since there is not any image database which is publicly available and designed for the forensic research on detecting seam carving, we implemented the seam carving algorithm in MATLAB and established 12 seam carved image sets based on the BOSSbase 1.01 [37], which is a benchmark image database for the research of steganalysis. It contains 10,000 never-compressed grayscale images with the size of 512×512. For each image from the BOSSbase, the pre-implemented seam carving algorithm was utilized to reduce the height by 5%, 10%, 20%, 30%, 40% and 50%, respectively. Therefore, 6 groups of seam carved copies were acquired. Similarly, by scaling the width of each original image with aforementioned various scaling rates, another 6 groups of seam carved copies were generated. Consequently, 12 seam carved copies were obtained for each image in the BOSSbase and thus 12 seam carved image sets were formed, i.e., '5%H', '10%H', '20%H', '30%H', '40%H', '50%H', '5%V', '10%V', '20%V', '30%V', '40%V' and '50%V'. Specifically, '5%H' stands for the height of each original image was scaled by 5%, '5%V' mean the width was decreased by 5%. As a result, each seam carved set contains 10,000 seam carved images.

To evaluate the performance of the proposed CNN architecture, the experiments were conducted to detect the 12 designed seam carving cases. In the experiments, the proposed CNN architecture was constructed with Caffe toolbox [38], and stochastic gradient descent was applied to train all the CNN with the batch size of 64 images. We fixed the momentum as 0.9 and the weight decay as 0.0005. The learning rate was initialized to 0.001 and forced to decrease 10% after each 5000 iterations. To fairly compare the performance with the state-of- the-art, we not only implemented and tested

methods proposed for seam carving detection [23, 25], but also examined the performance of rich model [24] which represents the state-of- the-art of steganalysis. Each method was tested on the 12 seam carving cases with linear SVM as the classifier [39]. Additionally, 2-fold cross validation was applied throughout the experiments.

As shown in Table 1, the proposed CNN architecture performs significantly better than the two state-of-the-art of seam carving detection [23, 25] when the scaling rate is below 30%. In particularly, our method achieves, respectively, 90% and 93% detection accuracies in the experiments of testing '5%H' and '5%V', the two toughest cases, which are 20% higher than performance achieved by both state-of-the-art.

The receiver operating characteristic curves (ROC) together with the corresponding area under ROC curves (AUC) shown in Figure 3.4 indicate that the proposed method outperforms the two seam carving forensic methods dramatically on detecting both '5%' and '10%' cases. It is also observed that rich model outperforms the [23, 25] on those low carving rate cases although it still underperforms the proposed CNN. Notably, the detection accuracy increases monotonically with the increased carving rate for all tested methods, and the gap between the proposed method and the tested prior arts is getting smaller as well. The reason behind is that overfitting is more significant for the methods which are more complicated and more powerful on modelling, such as proposed CNN and rich model as well, on detecting easy cases, i.e., detecting images in which a large number of seams are carved out.

In Figure 3.5, three samples are presented. The outputs of Conv5 for each sample and the corresponding seam carved copies are visualized as heat maps to illustrate what can be learnt by the proposed CNN. The region in the heat maps which has large value

indicates the ROI (region of interest) learnt by the deep neural network. It is observed that the trained deep neural network can effectively discover the region where the seams are deleted by learning from the seam carved copies, while irrelevant regions are learnt from the non-seam carved images. This also illustrated the effectiveness of the proposed CNN architecture on detecting seam carving.

Table 3.1. The Performance of Proposed CNN Architecture and The State-of-the-Art [23, 25, 34], on Detecting 12 Seam Carving Cases

	5%H	10%H	20%H	30%H	40%H	50%H
Ref. [11]	65.92%	72.88%	82.78%	90.31%	95.01%	97.77%
Ref. [13]	70.26%	83.60%	94.35%	97.90%	99.16%	99.71%
Ref. [22]	86.89%	93.22%	96.95%	97.98%	98.60%	99.07%
Proposed	90.37%	95.18%	97.84%	98.76%	99.21%	99.56%

	5%V	10%V	20%V	30%V	40%V	50%V
Ref. [11]	71.13%	79.83%	88.36%	93.18%	96.08%	97.79%
Ref. [13]	58.74%	71.50%	85.68%	93.31%	97.25%	98.97%
Ref. [22]	87.06%	94.74%	97.98%	98.82%	99.34%	99.60%
Proposed	93.99%	96.71%	98.55%	99.08%	99.45%	99.60%

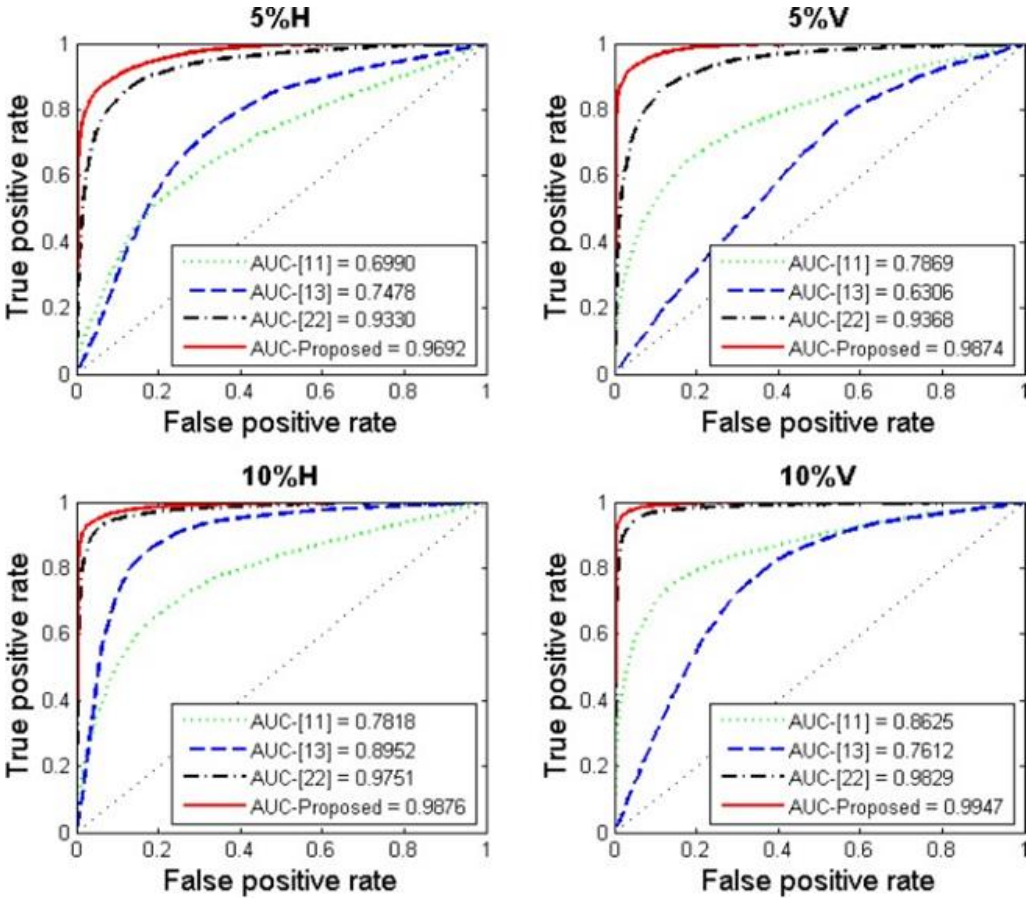


Figure 3.4 The ROC curves and their corresponding AUC curves.

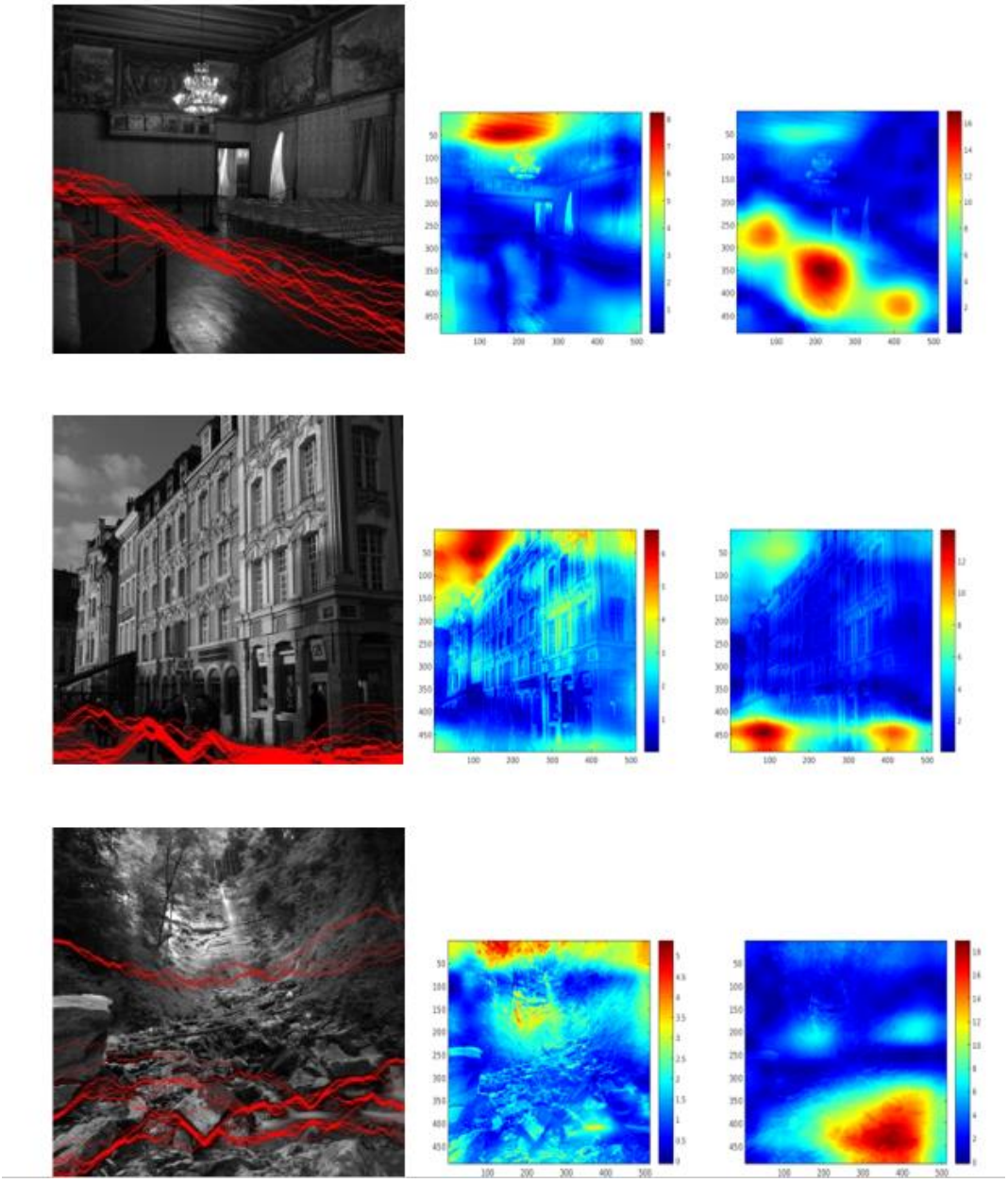


Figure 3.5 Heat maps.

Images in the first column illustrate the ground truth of carved seams in the original images with the carving rate equal to 5%. Heat maps in the second column are learnt from the original non-seam carved images by the proposed CNN, while heat maps learnt from the seam carved copies are shown in the third column.

3.5 Summary

In this chapter, a convolutional neural network architecture has been established and utilized for seam carving detection. It is the first deep learning framework on this research topic as far as we know. Indicated by experimental results, the proposed deep learning method can successfully detect seam carving in uncompressed digital images and outperform the state of-the-art in most of the experiments. In particular, the proposed deep convolutional neural network has achieved remarkable performance on detecting low carving rate cases, i.e., 5% and 10% carving rate cases. The performance of deep neural network on detecting seam carving in compressed images, i.e., JPEG images, needs to be further investigated. Therefore, the future work will be focusing on the remaining questions. Overall, through our work, it has been shown that deep learning could be a new direction for the forensic research on seam carving detection.

CHAPTER 4

REAL-TIME ESTIMATION FOR THE PARAMETERS OF GAUSSIAN FILTERING *VIA* DEEP LEARNING

4.1 Introduction

Intelligent devices and internet have made the digital images abundant and ubiquitous. It used to be difficult to edit or tamper with digital images by professional software, but in the last few years the cost and complexity of doing so have plummeted. Even the mobile devices now are equipped with powerful computational capability. Digital images can be easily edited or tampered just by using smart phone Apps instead of professional software. Usually, the forged images are difficult to be recognized by human eyes. Therefore, the authenticity of digital images may be suspicious. Digital image forensics [40–43] technique aims to verify the authenticity of digital images without the original source. Just like the other research area in information security such as privacy preserving [44, 45] and information hiding, they are all important to keep our community safe.

Digital images can serve as medium to carry information by displaying contents to human eyes. On the other hand, it can be also employed to carry hidden information. Words or even another image can be implanted into an image secretly without making any alternation. This technique is called as steganography [46, 47]. It could be dangerous if it is used by criminals or terrorists. Therefore, steganalysis [48], as a technique to detect the hidden information in images is crucial. Other than revealing forgeries or hidden messages in digital images, in image forensics, it is also important to completely understand the editing history of images to protect the integrity of images. Under such requirements, all possible manipulations occurred during the image forming history need to be verified. It

could also serve as supplement to tampering detection in many cases. This is the reason many forensics researchers dedicate themselves into the works to detect certain manipulations. The digital forensics has been an active research field of information security techniques and still has a prospect future.

So far, a lot of successful digital image forensics works have been reported in the past, such as tampering detection [49, 50], camera model identification [51, 52], enhancement detection [53–55], double JPEG compression [56–58] and so on. In image forensics, filtering operation plays an import role of image post-process. The verification of trace left by different filters [59–61] could provide sufficient evidence for further identifying the process of history. As a typical linear filter, Gaussian low-pass filter is widely utilized to eliminate noise and smooth images. These characters have been used for anti-forensics method [62, 63]. An image could be forged to hide the trace of copy and paste [64] by blurring the dis- continuities at the border of tampered objects. Median filter has the similar property to Gaussian low-pass filter; however, fewer works focus on Gaussian low-pass filter detection relative to median filter detection.

Perfect binary classification accuracy has been achieved for detection of Gaussian low-pass filter. Xu et al. [65] used the frequency residual function to detect Gaussian low-pass filtered images. In [66], the authors created feature vectors formed from both spectral and spatial domain to detect Gaussian low-pass filter. However, only a few research could detect the exact parameters of Gaussian low-pass filters. In [60], Boroumand and Fridrich detect the window size and standard deviation of different Gaussian filters from processing history of images. In their work, they detect four types of Gaussian low-pass filter, including window size 3 with standard deviation 0.5 and 1, window size 5 with standard

deviation 0.5 and 1. It is still a challenge to estimate the exact parameters of more different types of Gaussian low- pass filtering.

Except to refine the performance of forensics method from the aspect of accuracy, the other point scientists always mentioned is the computation speed. In the modern world, the smart devices are the most convenient tool for people to acquire and process new information. It is undoubtedly crucial to develop algorithms as real-time application for mobile devices. For such category as real-time application, in image forensics, other than validation accuracy, the computation speed is also a key element. It requires the algorithms to process data and make precise decisions immediately to assist people in real-time [67]. Unfortunately, little work has been done for the research of image forensics in real- time. As far as we can tell, the scientists were focusing on the validation accuracy over computation speed. However, it is believed that enabling algorithms towards the application in real-time is a potential prospective direction in forensics research. More and more scientists on image forensics are now paying attention to the application of image forensics in real-time to serve our community in realistic world.

Therefore, in this work, we focus on estimating the parameters of different types of Gaussian low-pass filtering in real-time. We propose a convolutional neural network (CNN) that is able to detect different types of Gaussian low-pass filtering in an extreme short time to serve as a real-time forensics tool. The CNN has been proved to be a useful tool for digital image forensics [68–71]. Conventional machine learning method such as support vector machine (SVM) [72] classify images based on handcraft features [73–75] extracted from images. However, there are limitations for handcraft features, as learning feature and classification are separate steps. Thus, these two steps cannot be optimized

simultaneously. Also, the manually extracted features may restrain the classification performance as it is fixed. Compared with these shallow machine learning methods, CNN is born with a superiority for classification. It is able to learn features and process classification automatically [76–78]. Besides, through back propagation, the classification results can be used to further optimize the procedure for feature extraction.

On the other hand, estimating the parameters of various types of Gaussian low-pass filtering is more difficult than just distinguishing the Gaussian low-pass filtered images from raw images. The Gaussian blur can be generated after the image is processed by Gaussian filter. The Gaussian filtered images can be easily distinguished from the original images by detecting the effect of Gaussian blur even with the traditional shallow linear classifier. However, if both images are processed by Gaussian filter, the traditional method to differentiate them may fail because Gaussian blur can be found in both of them. The only method can solve this problem is to estimate the intensity of Gaussian filter, in other words, to estimate the parameters applied in Gaussian filtering. The CNN with more powerful classification ability seems to be an idea candidate. What's more, a pre-trained CNN model is capable to analyze the given data quickly. The decision can be made immediately, that also makes it a perfect tool for real-time estimator.

Thus, we design a specific CNN architecture for estimating the parameters of Gaussian low-pass filtering in real-time. The experimental results demonstrate our method could successfully evaluate the parameters of different Gaussian low-pass filters. Besides, the model is also efficient in computation speed which makes it suit- able to serve as a real-time application. Some discussions are made as a guidance to build a proper CNN structure that can achieve a balance point in computation accuracy and computation speed to deal

with the given problem in real-time.

The rest of this chapter is organized as follows. Section 2 introduces the theoretical concept of Gaussian low-pass filter. Our proposed method is explained in Section 3. The experimental results and analysis appear in the fourth section. Then the summary is concluded in the last section.

4.2 Gaussian Filter

In image processing, Gaussian low-pass filter is widely used as a smoothing tool to remove noise or to produce Gaussian blur. The high-frequency component of an image will be eliminated by Gaussian filter. Thus, the noise with high-frequency components can be removed. The Gaussian blur is also welcomed in certain images to produce a pleasant view. Such as human facial or portrait images, the Gaussian filter can be used to remove wrinkles and freckles that is popular in social network. Gaussian smoothing also plays an important role in computer vision. It is an effective pre-processing stage to enhance image structures at different scales.

The Gaussian low-pass filtering uses a Gaussian function to calculate the transformation to apply to each pixel in the image. The Gaussian function in one dimension has the forms that displayed in the equation below.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{x^2}{2\sigma^2}\right)} \tag{4.1}$$

where the σ represents standard deviation of the Gaussian distribution. Figure 4.1 shows an example of one-dimensional Gaussian distribution curve with standard deviation of one.

In two dimensions, the Gaussian function is expressed as:

$$f(x) = \frac{1}{\sigma^2 2\pi} e^{(-\frac{x^2+y^2}{2\sigma^2})} \tag{4.2}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis. Figure 4.2 shows a two-dimensional Gaussian distribution curve.

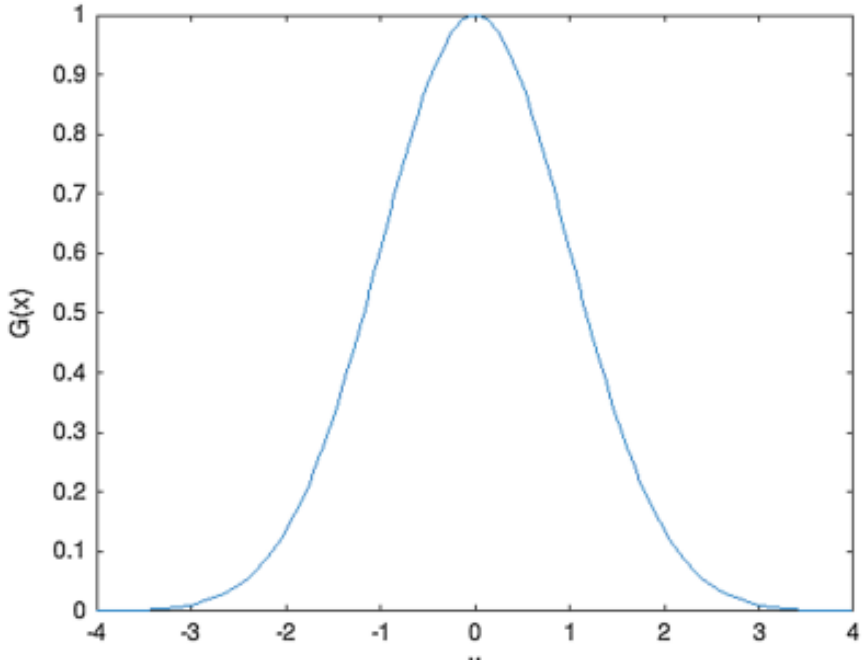


Figure 4.1 One-dimensional Gaussian distribution curve with standard deviation of 1.

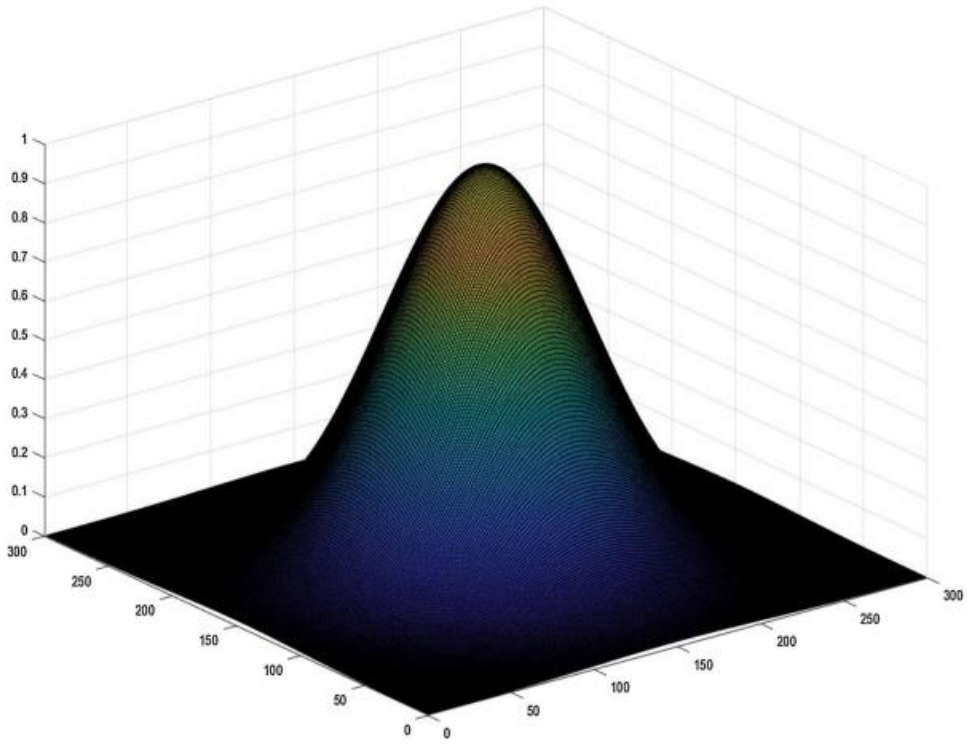


Figure 4.2 Two-dimensional Gaussian distribution curve.

The Gaussian low-pass filtering can be denoted by the expression below:

$$G(u,v) = F(u,v) \times H_G(u,v) \tag{4.3}$$

where G is the frequency response of the filtered images, F represents the original images, and H_G stands for the transfer function of Gaussian low-pass filter. When applied in digital image processing, the Gaussian function will create a convolution kernel with values correspond to Gaussian distribution from the center point. Based on theory analysis, the Gaussian distribution is non-zero everywhere, meaning that the convolution kernel will be infinite large. Note that, in most cases, we apply single dimension Gaussian distribution to process images. The default window size is fixed to 3. That leaves the σ to be the only adjustable parameters for Gaussian filtering. Theoretically, σ can be any positive numbers. The larger of σ, the intense Gaussian blur effect can be achieved. However, in practice, the

values at a distance of more than three standard deviation from center could be considered effectively zero because they only have a marginal effect. In other words, the Gaussian filters with larger standard deviation can be considered as homogeneous that is meaningless for analysis. The σ are recommended to be no larger than the window size. Therefore, we could ignore the values out of that range and focus on the effective standard deviation only.

After a Gaussian filter kernel has been created, the Gaussian low-pass filtering can be implemented by using convolution method to the original image. Every pixel's new value is calculated by weighted average of that pixel's neighborhood. The original pixel has the heaviest weight. The further away from the original pixel, the smaller weights are set for the neighboring pixels. Consequently, the filtered image looks smoother but still preserves boundaries and edges. The degree of smoothing is determined by the standard deviation of Gaussian function. As we discuss the Gaussian distribution above, a larger standard deviation needs a larger convolution kernel in order to achieve a better filtering performance. Figure 4.3 displays an image that has been filtered by three different Gaussian low-pass filters.

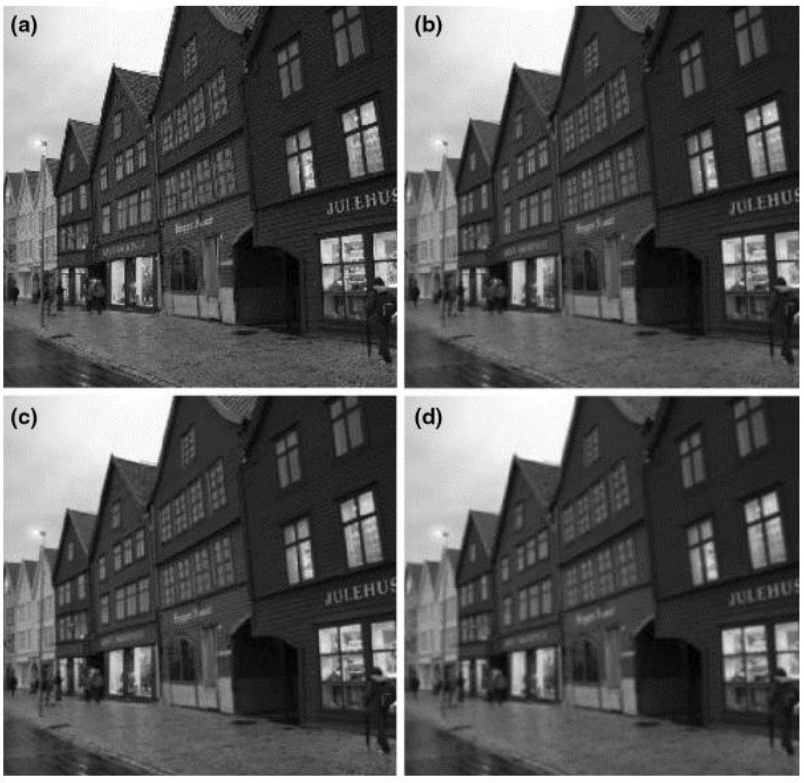


Figure 4.3 The effect of Gaussian blur: an original image; b Gaussian filtered image with window size 3 and standard deviation 1; c Gaussian filtered image with window size 3 and standard deviation 3; d Gaussian filtered image with window size 5 and standard deviation 1.

As shown in Figure4.3, with increasing standard deviation, the filtered image will be more blurred compared to the original images. In another aspect, a large window size may also bring a more obvious Gaussian blur. However, it is difficult to distinguish between any two of Figure 3b–d visually. Therefore, we cannot infer the exact Gaussian low-pass filter for each filtered image. To deal with this problem, we propose a CNN structure to identify different Gaussian low-pass filters. Next section will introduce our proposed method.

4.3 Proposed Method

Deep learning has a prosperous and successful development in recent years. Most deep learning methods are based on the neural networks which can be considered as a simulation to biological brain. Among all the neural networks, convolutional neural network (CNN) attracts most attention because of its amazing achievements. It is widely applied in variety fields to analyze data, such as object recognition, natural language processing, bioinformatics and so on. In our work, we employ it to solve the problem of approximately estimating the parameters of Gaussian filtered images.

A typical convolutional neural network consists of different types of layers. These layers are essential part of CNN. How to organize a proper CNN structure is the key to solve problems. Here, we introduce the proposed architecture. Figure 4.4 illustrates the overall architecture of our CNN.

The data layer, as the entrance of the network, controls the input data and makes necessary modification such as scaling or cropping to dataset. In our experiment, raw images are directly fed into the CNN without any modification in data layer.

After the fundamental processing of data layer, the vision layers are applied to produce feature images from original images. The most representative vision layers are convolutional layer and pooling layer. In fact, convolution and pooling can be regarded as the most important function in a neural network to learn and process features from images.

To be more specific, the convolutional layer could be considered as a set of feature extractors. It is composed of multiple filters. The filters are randomly created in the beginning and will be updated in CNN with the self-learning procedure. These filters are convoluted with the input data to the layer to generate the filtered output. Each filter can

generate a feature map, respectively. Therefore, the output of a convolutional layer are sets of arrays called feature maps. Each feature map represents a feature extracted from the input images. Generally, the filters in each convolutional layers increases with the network going deeper. In our structure, we set the numbers of filters doubles when moving to the next convolutional layer. In the shallow level of the network, the output of convolutional layer can be always found to be edges or texture of image contents which represent the intuitionistic features of objects. However, with the network going deeper, there are higher-order features that makes no sense for human eyes. These deeper features as well as the shallow features serve as the key for classification.

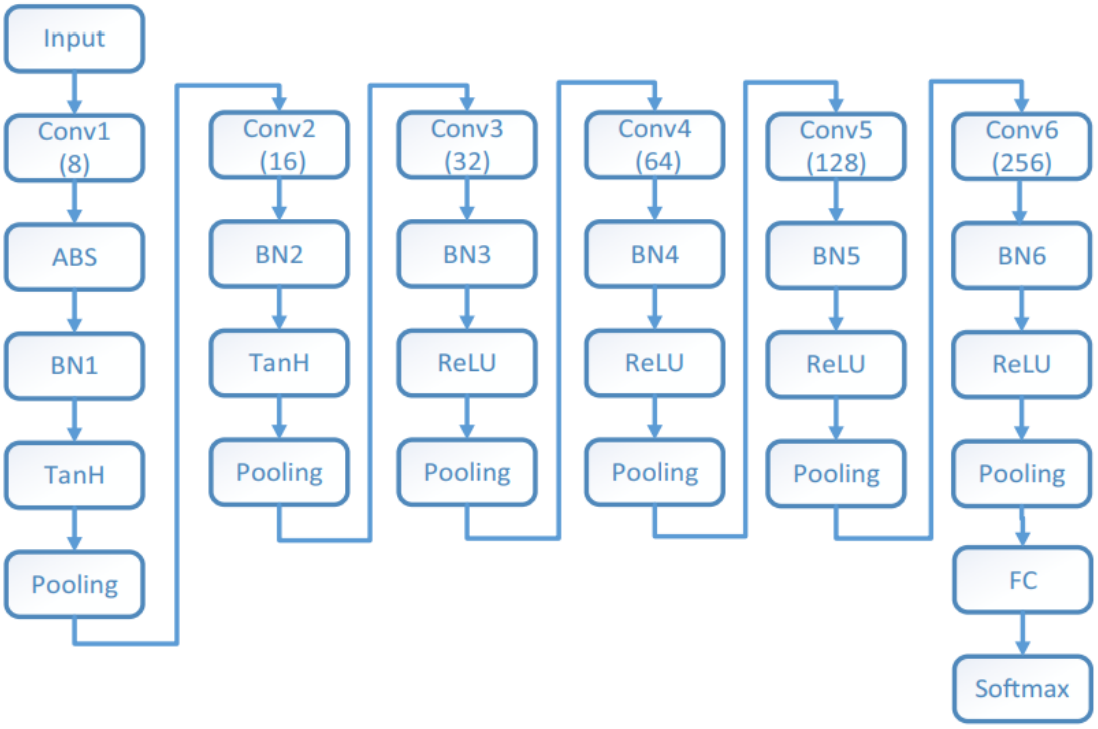


Figure 4.4 Proposed CNN architecture.

With several convolutional layers set in a network, considering there are multiple filters in each layer, the data size of generated feature map could be substantial. It requires

a lot of extra time to process that is not necessary. That is why the pooling layer is applied in neural network after convolutional layer. That is, to subsample the feature maps to decrease the feature dimension. It is also efficient to prevent overfitting. The down-sampling strategy can be chosen from ‘average’ and ‘maximum’ for pooling layer. The average pooling is widely used in computer vision for object or motion recognition. Generally speaking, the average pooling computes for the average value of all pixels in feature block to represents the given block while the maximum pooling employs the maximum intensity found in the block. The maximum pooling is more suitable for digital forensics with no details omitted. Hence, the maximum pooling is applied for all pooling layer in our network

In most cases, the trunk of a convolutional neural network is composed with multiple pairs of convolutional and pooling layers. A network with more vision layers is considered to be deeper that has a relative strong learning ability. However, in CNN, strong learning ability does always lead to exceptional performance. The phenomenon of over-fitting occurs when a strong network is adopted to solve a simple problem. When the network is overfitted, it cannot classify the images properly following the original purpose. For instance, when overfitting happens, a network designed for tampering detection may try to classify images by the objects in images that may suffer a failure for tampering detection. Hence, in order to avoid this phenomenon, the vision layers are selected with proper amounts to solve problems of different levels. Considering the classification for Gaussian filtered images is a challenging problem, we employ 6 pairs of convolution and pooling layers.

Other than the vision layers referred above, there is also the activation layer which

is crucial for CNN. It is always placed behind the convolution layer but before pooling layer. The activation layer brings nonlinear property into feature extracted by convolution layer, that is a brilliant function for classification. Comparing with the traditional linear only classifier, the classifier with both linear and nonlinear property is more suitable for classification, especially for multiple labels classification. It has been proved by many machine learning scientists. Until now, many activation functions are proposed to solve different problems. However, among these options, there are three basic forms which can be considered as classic activation functions, the Sigmoid, the ReLU and TanH. They can be described with the functions below, respectively. They are also illustrated in Figure 4.5

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{4.4}$$

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x \geq 0 \end{cases} \tag{4.5}$$

$$TanH(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5.6}$$

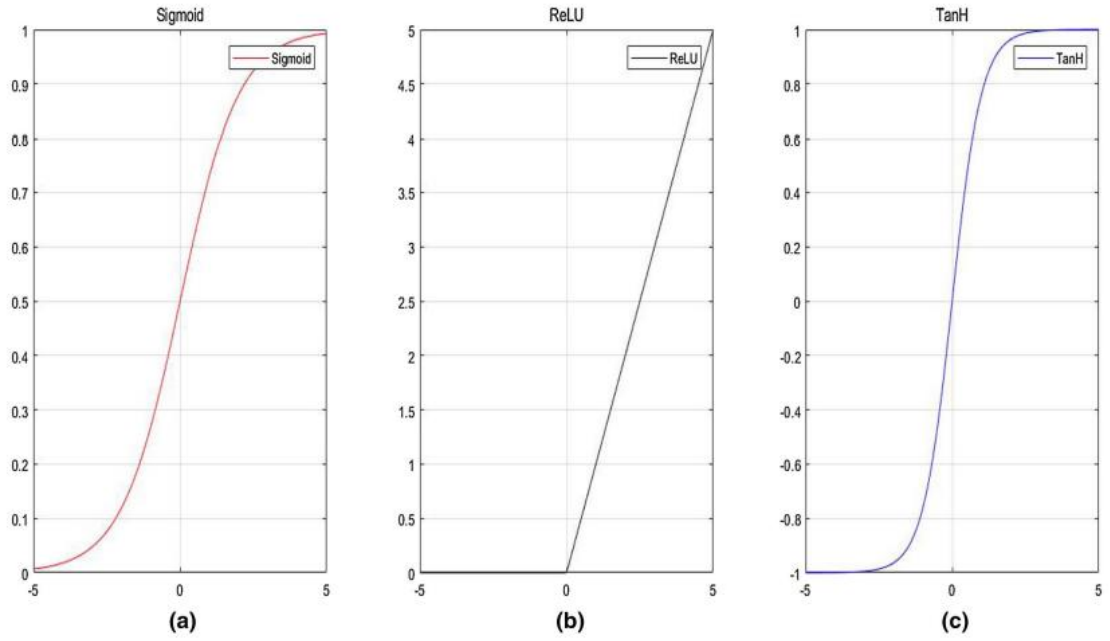


Figure 4.5 Sigmoid, ReLU and TanH functions.

Observed from Function 4.4, the output of sigmoid function is 1 when the input is close to positive infinite. And when the input is close to negative infinite, the output of sigmoid is 0. Consequently, the gradient of loss function tends to be 0 if default input is close to positive or negative infinite. This may bring in some issues to the CNN. In optimization stage, the strategy based on analyzing gradient of loss function, such as stochastic gradient descent, is preferred. Gradient of 0 may cause saturation which makes the network can hardly learn anything from the input data. Other than that, the output range of Sigmoid is between 0 and 1 which may also bring difficulty for optimization. It is nearly abandoned in recent years.

Instead, the ReLU layer is more popular as the choice for activation. From Function4.5, ReLU is simpler, the output remains the same if the input is larger than 0. Otherwise, the output is fixed to 0. It is computational efficiency and friendly for optimization. However, the ReLU layer is also sensible to the learning rate. It is vulnerable if the learning rate is high.

The TanH function is shown in Function4.6. It is a morph of Sigmoid function. However, the mean of TanH is 0 and the output range for TanH is between 1 and −1. This can be helpful for optimization. The ReLU activation has no upper threshold while the TanH activation has two thresholds. This property helps to cancel the propagation of dynamic range over layers. Thus, overflow could be prevented during the training phase and testing phase. Besides, it has been verified to be a hardware friendly method. Considering all the elements, TanH is chosen for the first two convolution stages in our proposed method, the ReLU is chosen for the rest to boost the computation speed. It is a tradeoff strategy between performance and computation burden. Another layer that needs

to be mentioned is batch normalization layer. Before this layer is introduced in CNN, it is highly possible that overfitting could occur because the self-learning ability of CNN is not controllable.

Although the overfitting can be prevented by applying drop-out layer to stop certain neurons, it also limited the self-learning ability of the CNN. Besides, the images fed into CNN are strictly controlled that all images must present the designated feature clearly. This high level of requirement is really difficult in data collecting procedure. With the batch normalization, the learning ability of CNN are controlled to concentrate on designated features. That makes collecting data more easily. Besides, it is no longer necessary to drop out neurons that the computation speed is also boosted. That is the reason we place batch normalization layer after each convolutional layer.

There are also layers in different categories to serve different purposes in our networks. The inner product layer also known as fully connected layer, is responsible to connect all the feature extracted from previous layers for classification. It also serves as a consultant for the entire network to decide what and how to learn from the input data. It is located after the last pooling layer in network. The SoftMax with loss layer is utilized to evaluate the performance of the network. It is placed at the end of the entire network as exit. Considering our goal is to identify images filtered with different parameters regardless of the image content, a scale layer is built with batch normalization layer after the first convolution layer to normalize the shallow feature learned in first convolution layer. Besides, an absolute value layer is also applied here to boost the performance of normalization. The network can reach convergence sooner with the assistance of the absolute layer. In the next section, we evaluate the proposed CNN in variety circumstances

for analyzing Gaussian filtering.

4.4 Experimental Results and Analysis

Given the fact that it requires large amount of data to boost the performance of CNN in training procedure, the BOSS image dataset [79] is selected to conduct our experiment. It consists of 10,000 grayscale images in the format of pgm. Since we only care for the effect of Gaussian filtering, the grayscale images are acceptable for our experiment. The size of input images is fixed to 512 × 512 without scaling or cropping in data layer.

The parameters that can control the smoothing effect through Gaussian filter are the window size h and the standard deviation σ. In certain cases, it is not difficult to distinguish the images with Gaussian blur from the other images even for human eyes. However, it is impossible for human eyes to distinguish the vision effect brought by different parameters. Hence, our experiment is designed to estimate the parameters applied in Gaussian filtered images based on h and σ.

First of all, before the network is used as a real-time estimator, we want to give an attempt for binary classifications to see if it can distinguish the images with different σ when only two standard deviations are applied. The discrimination of Gaussian filtered images and original images is also simulated here. In this stage, 6 groups of images are prepared by passing through Gaussian filters with the window size fixed to 3, while the standard deviation varies from 0.5, 1, 1.5, 2, 2.5 and 3. 0.5 is always applied for denoising as pre-processing while the filtered images are nearly the same when σ is larger than 3. Another group is prepared as original image that is untouched from any manipulation. For each group, 9000 random images are picked as training set while the rest 1000 images serve

as validation set. For each step in current stage, two groups are chosen for classification by our proposed method. The results can be found in Table 4.1. Similarly, we can also test the classification ability of proposed method when windows size is fixed to 5. When the window size is extended to 5, the boundary of standard deviation is also increased to 5, as discussed in Section4.2. Hence, the standard deviation σ = 5 is also included. The related results are shown in Table 4.2. Afterwards, we conduct another experiment to check if our proposed method is capable to differentiate the images smoothed by Gaussian filter with different window size when standard deviation is fixed. The result is shown in Table 4.3

From all the tables above, the proposed network is able to distinguish the Gaussian filtered images with different parameters. Besides, a larger σ is helpful to identify the window size. However, in order to be functional as an estimator rather than classifier, it requires our proposed network can distinguish more than two groups of images. The nonlinearity brought by activation functions in CNN makes it a perfect tool for multi-label classification. Therefore, in the second stage of our experiment, we employ all groups of images with window size either 3 or 5 to estimate the standard deviation that applied. The standard deviation is also picked from [0.5, 1, 1.5, 2, 2.5, 3]. The accuracy for estimation is shown in Table 4.4.

Table 4.1 Binary Classification Accuracy for Different Standard Deviations when Window Size is 3

Standard deviation	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$	$\sigma = 3$	Original
$\sigma = 0.5$	×	×	×	×	×	95.21%
$\sigma = 1$	99.33%	×	×	×	×	99.64%
$\sigma = 1.5$	99.57%	99.8%	×	×	×	99.85%
$\sigma = 2$	99.82%	99.50%	98.7%	×	×	99.73%
$\sigma = 3$	99.75%	99.32%	98.15%	97.33%	×	99.55%

Table 4.2 Binary Classification Accuracy for Different Standard Deviations when Window Size is 5

Standard deviation	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$	$\sigma = 3$	$\sigma = 5$	Original
$\sigma = 0.5$	×	×	×	×	×	×	95.50%
$\sigma = 1$	99.79%	×	×	×	×	×	99.80%
$\sigma = 1.5$	99.77%	97.32%	×	×	×	×	99.76%
$\sigma = 2$	99.23%	98.86%	93.50%	×	×	×	99.65%
$\sigma = 3$	99.81%	99.65%	95.57%	92.36%	×	×	99.72%
$\sigma = 5$	99.20%	98.53%	98.57%	96.96%	95.71%	×	99.83%

Table 4.3 Classification Accuracy for Window Size 3 and 5 when Standard Deviation is Fixed

Standard deviation	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$	$\sigma = 3$
Accuracy	91.39%	98.20%	99.00%	99.25%

Table 4.4 Estimation for Standard Deviations under Different Window Sizes

Window size	Size 3	Size 5
Accuracy	99.35%	97.75%

The estimation can reach accuracy over 97% for different window sizes that can be considered as a success. The performance of the proposed CNN is illustrated in Figures 4.6 and 4.7. We can see our proposed CNN start to converge after about 30 epochs and reach the convergence after about 60 epochs.

Furthermore, in order to serve as an estimator towards Gaussian filtering, our method should evaluate both window size and standard deviation simultaneously. Hence, images processed by all kinds of *h* and are fed into CNN together for this experiment.

Besides, we also want to test the performance of proposed CNN architecture with different pooling method as we discussed in Section 4.3. The comparison is made between two pooling strategies as displayed in Table 4.5.

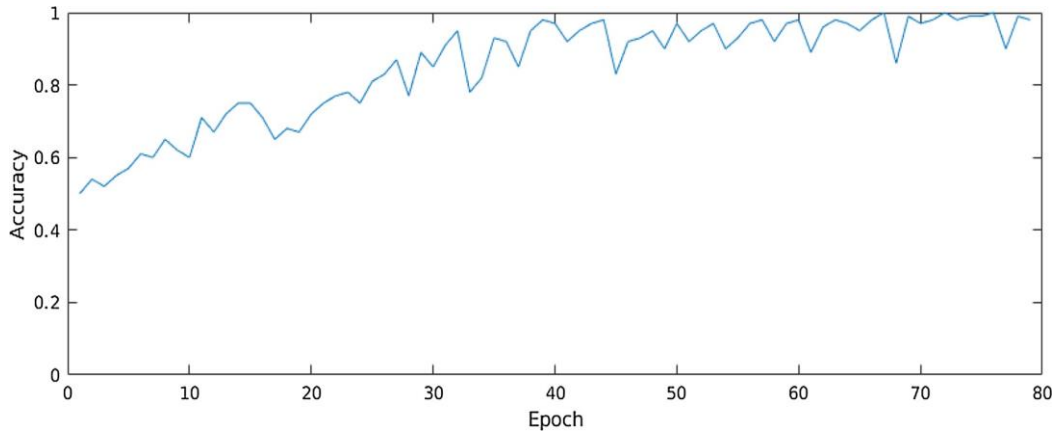


Figure 4.6 Performance of proposed CNN for window size 3 based on count of epoch.

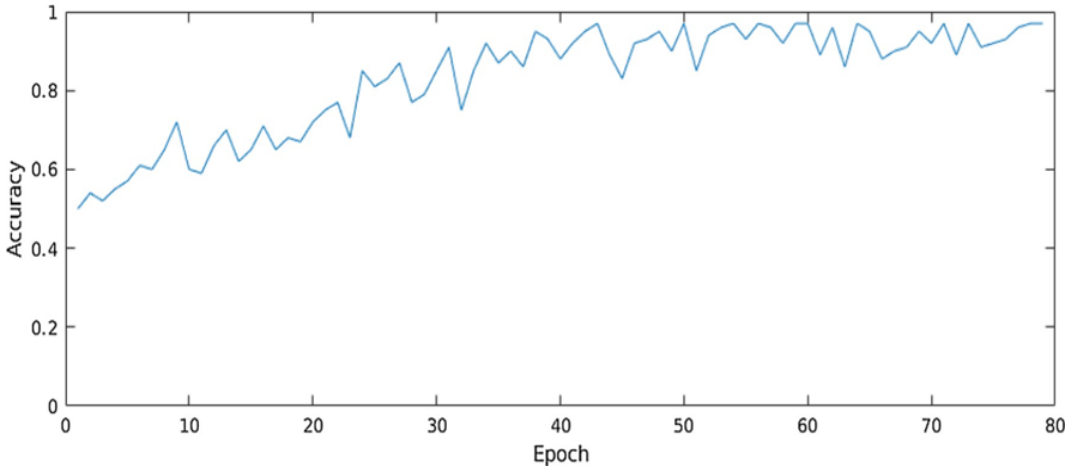


Figure 4.7 Performance of proposed CNN for window size 5 based on count of epoch.

Table 4.5 Estimation Accuracy for Same Model with Different Pooling Methods

Pooling	Max	Ave
Accuracy	96.95%	79.87%

From the results, it is apparent that the maximum pooling outperforms the average pooling as we assumed. In order to accurately analyze the texture in images, the maximum pooling is recommended while the average pooling is more suitable to get a brief view of objects in images. Other than that, it is easier to reach convergence for maximum pooling during our experiments. Therefore, the maximum pooling is verified as the best pooling strategy in our proposed methods

Besides the pooling, how to determine the depth of the CNN model can be also a critical problem in designing for CNN architecture. Generally speaking, the depth D of a CNN model is decided by the amount of convolutional layer applied. Here, in our experiment, we choose four models with depth equal to 5, 6, 7 and 8 for comparison. The results are reported in Table 4.6.

Table 4.6 Estimation Accuracy for Models with Different Ds

Depth	$D = 5$	$D = 6$	$D = 7$	$D = 8$
Accuracy	91.23%	96.79%	96.95%	94.33%

Based on the estimation accuracy, it is quite clear that when 7 convolutional layers are applied, the CNN model can reach the highest accuracy of 96.95% that outperforms all the other models. In most forensics research, D will be simply set to be 7 to acquire the most precise accuracy.

However, as referred in prior, another important point we have to mention is the computation speed of the proposed method. Including the consideration for estimation accuracy, computation speed is also a decisive component to make the proposed method to be a real-time tool. Hence, at last, we evaluate the computational efficiency to verify if

the proposed method qualifies to be employed in real-time scene. It requires the proposed method to be able to process large amount of data as soon as possible. Here, the time consumed to estimate the Gaussian parameter for 1000 images can be used as a measurement towards such purpose. The time consumption for 4 models with different numbers of convolutional layers on same computer with Caffe [38] on GPU GTX 1080ti are displayed in Table 4.7.

Table 4.7 Time Consumption for Models With Different Ds to Analyze 1000 Image

Depth	$D = 5$	$D = 6$	$D = 7$	$D = 8$
Time (s)	3.458s	3.590s	3.984s	4.353s

From the above table, it is quite obvious that more convolutional layers bring in more computation burden. Although the model with D to be 7 has the best estimation accuracy, it also takes more time for computation than the models with less convolutional layers. It is quite difficult to judge the performance of models from the above two tables. Hence, if the estimation accuracy is the major concern, we recommend the model with 7 convolutional layers. On the other hand, the model with 6 convolutional layers, or even the model with D = 5 are trade of models if it is required to complete the estimation sooner.

4.5 Summary

In this chapter, we proposed a method for real-time estimation of the parameters applied for Gaussian filtered images. This method is designed based on multiple labels classification ability of convolutional neural networks. The overall performance of the proposed method was evaluated by our experiments. Based on the data acquired and discussion, the designed network is able to reach high estimation accuracy in a short time that qualifies to serve as a real-time estimator. Some discussions are also made to evaluate the model with different amount of convolutional layers to satisfy different needs.

There are also some other thoughts summarized from this work. At first, the forensics research was concentrating on detecting the trace of designate manipulation. The detector was always designed to be a binary classifier. However, it is also important to distinguish the images even they are processed with same manipulation. With more information dug from images, it will help us to analyze the image more thoroughly. It is demanded if we want to completely understand the history of a given image. Gaussian filter is the one of the most common editing manipulations, we want to start with it as a break point to design estimator in image forensics for different purposes.

CNN has already been proved by many scientists to be a powerful tool to identify images with differences. In our work, we have proved that CNN can even distinguish images with minor differences, considering the only difference in our images is the Gaussian parameter. It is believed that there is still much potential we can explore from CNN and deep learning.

The training procedure for CNN models takes a lot of time and requires large amount of data. In order to makes it a real-time tool, the CNN models are always pre-

trained with fixed dataset to skip the training procedure. However, if the test image can be also immediately employed by CNN as a training set after validation, it can surely deliver a positive impact to optimize the CNN model. Hence, it is supposed to develop the model if it is dynamic during the real-time procedure.

Another point to enhance the CNN model as real-time tool is to boost the convergence speed. If a model can reach convergence with far less epochs, there is no need to employ pre-trained model against real-time issues. Besides, such model is capable to be trained in real-time towards different problems. It would be amazing if such technique can be developed.

Although we can understand how CNN works, what happens within it still remains a mystery. Unlike the features extracted in shallow layers, the higher order features make no sense for human eyes. Based on the reports of CNN, these deeper features are the key element of the powerful classification ability of CNN. It could be more interesting if we can establish a connection between these deeper features and real images.

CHAPTER 5

DIGITAL IMAGE FORENSICS BY USING PROTOTYPICAL NETWORKS

Digital image forensics investigates the anomalous patterns that might result from image manipulation. Over the past several years, machine learning techniques are successfully applied to the detection of image forgeries as a result of the extraordinary growth of machine learning. Convolutional Neural Networks (CNN) are frequently used in digital image forensics. A CNN model could distinguish original images from a certain kind of tampering images. However, when presented with a new image forgery detection task, each CNN model must be trained from scratch. Additionally, certain types of tampered image data are challenging to acquire or simulate.

Meta-learning is an alternative learning paradigm in which a machine learning model gains experience across numerous related tasks and uses this experience to improve its future learning performance. Few-shot learning is a method for acquiring knowledge from few data. Inspired by meta-learning and few-shot learning, we apply the proposed prototype networks to two image forgery detection tasks in this chapter. One is the detection of images with Gaussian filtering, while the other is the detection of images with average filtering. Our prototype networks do not need to be trained from scratch for a new task, unlike a traditional CNN model. Additionally, it drastically reduced the amount of images required for training. Our results reveal that the accuracy of the proposed method for two digital image forensics tasks is relatively high.

5.1 Introduction

In numerous fields, artificial intelligence (AI) has achieved remarkable progress [80-83]. The use of artificial intelligence is nearly ubiquitous in our daily lives. Due to the advancement of modern technology, artificial intelligence is playing an ever-increasing role in business and industry. These accomplishments have relied primarily on the fact that the development of these complicated models of artificial intelligence necessitates massive amounts of data. However, we will not always be able to construct a dataset from such a vast quantity of data. There are two novel concepts to generate machine learning model to solve this problem.

Few-shot learning [83, 84] or N-way learning is the process of learning from few data, where N represents the number of dataset classes and k denotes the quantity of data in each dataset class. For instance, we want to classify cats and dogs in our dataset. There are two data classes named cats and dogs. Two classes could be represented as 2-way. Each time we train the model with 5 images per class. Here, 5-shot refers to five images per class. Now, we can state that this is a 2-way 5-shot classification task.

A typical example of few-show learning in the real world is drug discovery. For the purpose of determining which drug poses the least risk to patients, medical researchers attempt to identify the beneficial features of test pharmaceuticals. In [85], H Altae-Tran et al utilized one-shot learning to drug discovery under low data constraint. Also, Few-shot learning advantages machines. It makes machine learning similar to human learning. Some research focuses on the implementation of few-shot learning on robots, such as one-shot imitation [86], multi-armed bandits [87], visual navigation [88], and continuous control [89].

Another concept is called meta-learning. Generally speaking, meta-learning is described as “learning to learn”. It doesn’t learn how to complete a specific task. It successively learns to solve many tasks. It will utilize its prior learning experience to lean the new tasks. It gradually improves at learning new tasks one by one. Typically, experts in machine learning divide meta learning into three categories: learning the metric space [90, 91], learning the initializations [92, 93] and learning the optimizer [94, 95]. Combined these two ideas, a machine learning model is designed for learning to learn from few data.

Another concept is called meta-learning. Generally speaking, meta-learning is described as “learning to learn”. It doesn’t learn how to solve a specific task. It successively learns to solve many tasks. It will use the previous learning experience to lean the new tasks. It becomes better at leaning new tasks one by one. Usually, machine learning scientists categorize meta-learning into the following three types: learning the metric space [90, 91], learning the initializations [92, 93] and learning the optimizer [94, 95]. Combined these two concepts, a machine learning model is design for learning to learn from few data.

Meta-learning is comprised of two steps: meta-training and meta-testing. The standard training set and testing have different names in meta-learning. The training set is referred to as the meta-training query set or meta-testing query set depending on application stage. Similar to training set, testing set is also referred to as meta-training query-set or meta-testing query-set. Meta-learning has evolved into a framework for few-shot learning. Figure 5.1 depicts the configuration of the meta-learning for few-shot image classification [96]. Each meta-training or meta-testing task is a few-shot learning task.

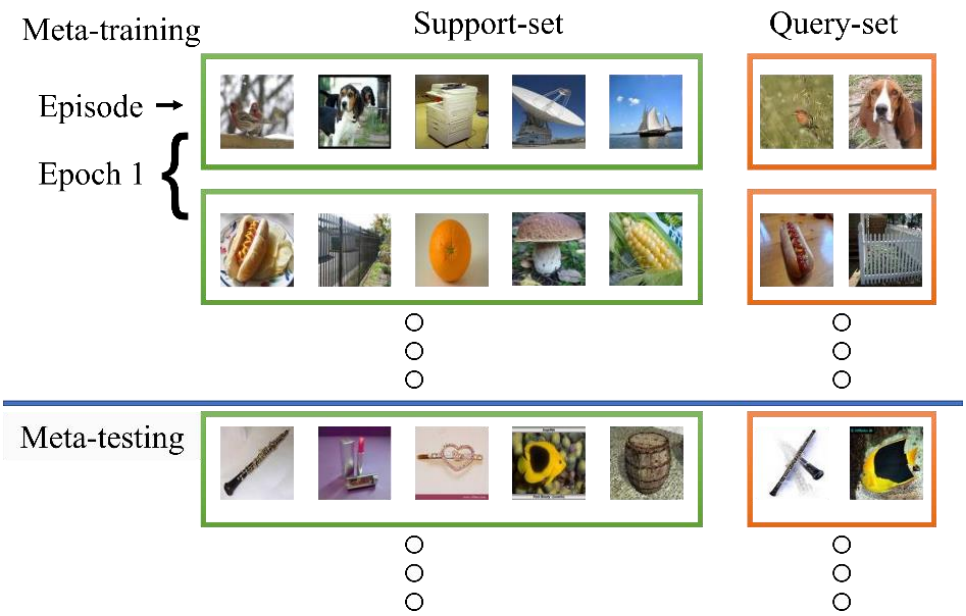


Figure 5.1 Meta-learning set-up for few-shot image classification.

In recent years, digital image forensics has drawn the interest of numerous scholars. It focuses on validating the authenticity of information associated with digital images. Nowadays, digital images could be easily edited or tempered just by using an application on a smart phone. To change digital images with high-performance image editing software such as Photoshop, no professional skills are required. Due to this, digital image forensics is now more crucial than ever before.

In the last few years, machine learning approaches to digital image forensics have grown rapidly [97-99]. One of the major machine learning approaches is convolutional neural networks (CNN). Detection of various filtering operations is a subfield of digital image forensics, since filtering operations are widely used to edit images. Both the Gaussian filter and the average filter are frequently used to remove noise and smooth images. Thus, many researchers pay attention to detect Gaussian filtering and average filtering in images. Some publications to detect Gaussian filter[100] and average filter [101] have proved CNN could be a valuable tool for digital image forensics.

In prior studies on CNN-based Gaussian filtering and Average filtering detection, thousands of images are fed to train CNN. Also, every CNN-based model could only determine one certain filter. When a CNN model is applied to two or more similar tasks, it must be fine-tuned to adapt to each task. Thus, even though these two filters have the similar effects on images, a CNN model cannot work perform optimally for both tasks without fine-tuning.

In this chapter, we propose a prototypical networks model to detect Gaussian filtering and average filtering. It is a few-shot learning model based on meta-learning framework. The novelty of our work is twofold. We detect Gaussian filtering and average filtering both in a same prototypical networks model. In this model design, fine-turning is no longer necessary Also, we use far less images to train our model. The experimental results denominate our model could obtain high accuracies on both tasks.

The rest of this chapter is organized as follows. In Section 5.2, the theoretical idea of prototype networks is introduced. Section 5.3 explains how to prepare dataset. Section 5.4 illustrates how to design CNN as an embedding function for prototypical networks. And our experimental results are presented in section 5.5.

5.2 Prototypical Networks

Prototypical networks [90] are simple but efficient few-shot learning algorithm. A prototypical network attempt to learn the metric space to make classification. The fundamental concept underlying prototypical networks is to compute a prototypical representation of each class based on an embedding function $f_{\emptyset}(\,)$. The embedding function can be any function that has the ability to extract features, such as convolutional

neural networks(CNN) and long-short term memory networks(LSTM). Then prototypical networks could make classification based on the distance between every class prototype and the data's embeddings.

As the embedding function in our prototypical networks, convolution neural networks are designed. The training and testing process for prototypical networks differs from that of conventional CNN but is otherwise identical. Figure 5.2 depicts our prototypical networks architecture.

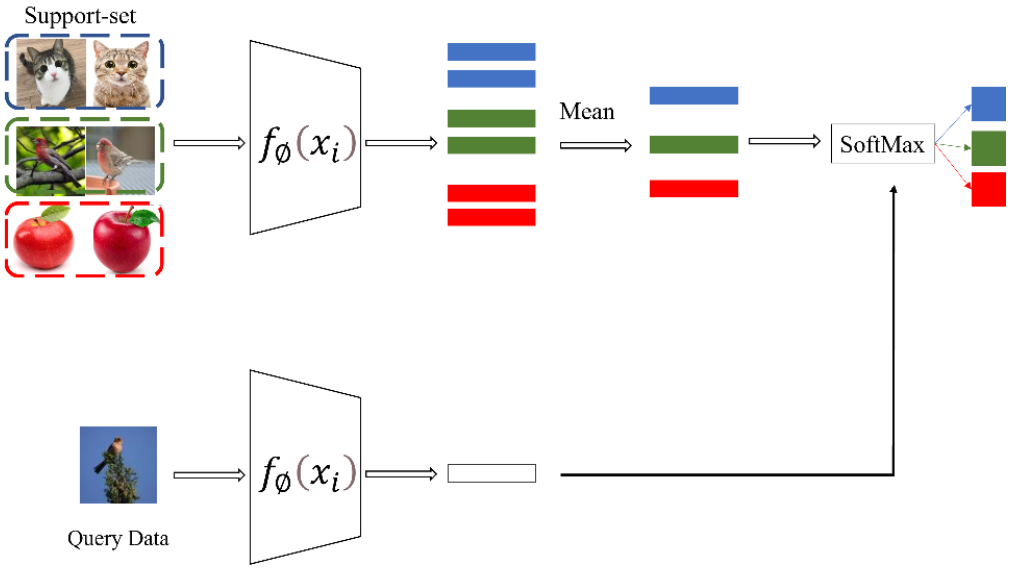


Figure 5.2 Proposed Prototypical networks.

The workflow of prototypical networks is shown below:

1. We have a support-set which contains N classes labeled data, comprising $S = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$ where $x_i \in \mathbb{R}^D$ is D-dimensional feature vector of an example and $y_i \in \{1,2, \ldots, n\}$ is the corresponding class label. There are k samples in each class.

2. The query-set contains samples from the same N classes. Every class has Q samples in it.

3. We use episodic training. In each episode, prototypical networks are trained on support set S and test on query set Q. prototypical networks aim to classify $N \times Q$ query data into N classes.

4. CNN work as an embedding function $f_{\emptyset}(x_i): \mathbb{R}^D \rightarrow \mathbb{R}^M$ with learnable parameters $\emptyset$, to compute each data's embedding. By the end of CNN extract each data's M-dimensional features, these features constitute each data's embeddings. After we have the embeddings for each data in support set S, we can calculate the class prototype of each class. S_n is a group of examples labeled with class n. For a certain class, the average value of all data's embeddings under this class is its class prototype:

$$\boldsymbol{c}_n = \frac{1}{|S_n|}\sum_{(x_i,y_i)\in S_n} f_{\emptyset}(\boldsymbol{x}_i) \tag{5.1}$$

5. Similarly, we calculate the data's embeddings in query set.

6. We calculate the Euclidean distance, $d: \mathbb{R}^M \times \mathbb{R}^M \rightarrow [0, +\infty)$, between a query data's embeddings and the class prototypes. Classification will be performed to a query data by finding its nearest class prototype.

7. We predict a query data's class by a probability function, $p_{\emptyset}((\mathrm{y} = \mathrm{n}|\mathrm{x})$ based on a SoftMax over distance to the prototypes in the embedding space:

$$p_{\emptyset}((\mathrm{y} = \mathrm{n}|\mathrm{x}) = \frac{\exp\left(-d(f_{\emptyset}(x), c_n)\right)}{\sum_n \exp\left(-d(f_{\emptyset}(x), c_n)\right)} \tag{5.2}$$

   Since we have n classes, we will have n probabilities. The query data belong to the class which has the highest probability.

8. We compute the loss function $j(\emptyset) = -\log\left[p_{\emptyset}((\mathrm{y} = \mathrm{n}|\mathrm{x})\right]$.Then we use the Adam optimizer to minimize the loss:

5.3 Preparation for the Dataset

5.3.1 Gaussian Filter and Average Filter

In digital image processing, Gaussian filtering and average filtering are typically used to remove image noise and blur detail. They are straightforward, intuitive, and simple to implement methods to smooth images. To utilize these two filters on an image, we firstly define the size of window *W*, which determines determine the range of filtering operation. The dimensions are typically odd numbers, such as 3×3 or 5×5. The value of each pixel is replaced by all the values in the kernel.

For average filter, the size of filter is the only one parameter that must be specified. For Gaussian filter, the standard deviation σ is another important parameter. Consequently, the filtering result produced by a Gaussian filter will be affected from both the window size and standard deviation σ. Functions of Gaussian filtering and average filtering are shown in Function 3 and Function 4 respectively. Where M is the total number of pixels in the kernel N. Figures 5.3 and 5.4 provide instances of Gaussian filtered and average filtered images.

$$G(x,y) = \frac{1}{\sigma^2 2\pi} e^{(-\frac{x^2+y^2}{2\sigma^2})} \quad \text{.....(3)}$$

$$h[i,j] = \frac{1}{M} \sum_{(x,y)\epsilon N} f[x,y] \quad (4)$$

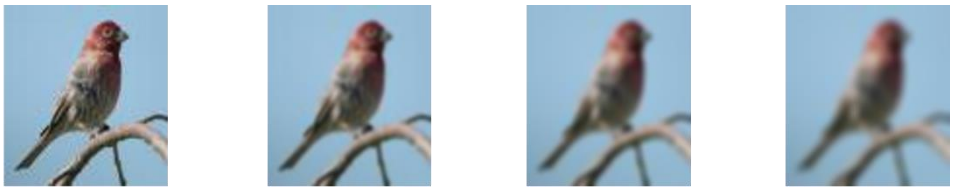


Figure 5.3 Gaussian filtered images with different window sizes and standard deviation.

Figure 5.4 Average filtered images with different window sizes.

5.3.2 Creating Support-set and Query-set

The objective of meta-training is to train prototypical networks with good classification performance. All images in meta-training step are not applied any filtering operation. We use an *mini*ImageNet [103] dataset for meta training classification. This dataset comprises 100 classes images sampled from ILSVRC-20 [104]. It splits to 64, 16, 20 classes as training, validation, and testing set respectively. In each class, there are 600 images of size 84×84.We construct our meta-training support-set from its 64 classes training set. In each meta-training episode, we randomly sample 5 classes from 64 classes. In these 5 classes images, we then sample 5 images per class to generate the meta-training support-set and 5 different images per class to build the meta-training query-set. This randomly sampling process is repeated at the beginning of each meta-training episode. Therefore, in each meta-training episode, the input support-set and query-set are 5 different images per class. Each meta-training task is a 5-way 5-shot classification task.

After meta-training step, the prototypical networks could make classification with a good accuracy. But we do not focus on this accuracy. Because the prototypical networks will continuously update during the meta-testing step. The classification accuracy in meta-testing step is what we must pay attention to.

For the meta-testing step, we use two datasets: Boss1.01[105] and Oxford-IIIT Pet [106]. Boss 1.01contains 10000 images includes size of 512×512. And Oxford-IIIT Pet has 37 categories dogs and cats with roughly 200 images for each class. Images in Oxford-IIIT Pet do not have a common image size. Since in the meta-training step, we utilize images with size of 84×84, here we need to resize all images in Boss 1.01 and Oxford-IIIT Pet appropriately to the size of 84×84. Images used in meta-testing step must have

the same size of images as those used in meta-training step. Otherwise, they are incompatible with prototypical networks.

The proposed prototypical networks are design to distinguish filtered images from original images. The idea of original images means these images have not been applied for any manipulation. We prepare two filtering operations. The first one is the Gaussian filtering and the second one is the average filtering. Our first experiment aims to recognize Gaussian filtered images from original images. Then, we try to classify average filtered images from original images. In meta-testing step, our proposed prototypical networks will perform binary classification instead of 5-class classification in meta-training step.

In the first experiment, images are filtered by three different Gaussian filters. The parameters that control the Gaussian filtering result are the widow size W and the standard deviation σ. There are three groups of images are prepared by passing through Gaussian filters with widow size of 3 and standard deviation of 0.5, widow size of 5 and standard deviation of 1, widow size of 7 and standard deviation of 1.5 respectively. Each group contains 200 Gaussian filtered images. At the end, we have three groups of various Gaussian filtered images in Boss 1.01 and Oxford-IIIT Pet dataset separately. Now we can create our meta-testing tasks. We totally have 6 meta-testing tasks. Three tasks created in Boss 1.01 and three tasks crated in Oxford-IIIT Pet. For each meta-testing task, we have two classes labeled images: Gaussian filtered images and original images. We randomly sample 5 images per class to build the meta-testing support-set and 5 different images per class to build the meta-testing query-set. Each meta-testing task is described as a 2-way 5-shot classification task.

In the second experiment, similarly, we create six average filtering datasets. The

only difference is the filtering operation. At this stage, images undergo an average filtering procedure. There is only one parameter influence the average filtering effect alone. That is the widow size of an average filter. We prepare three different average filters. Their window sizes vary from 3×3, 5×5 and 7×7. We develop six 2-way 5-shot meta-testing tasks to discern between the average filtered image and the original image.

In some previous digital image forensic research[100, 102], only using Boss1.01 dataset could prove their CNN models have good performance for their experimental purposes. In our experiment, we were also able to test our prototypical networks only on Boss 1.01. However, to strengthen the argument that our prototypical networks approach the idea of learning to learn from few examples, we test out model on additional dataset. Although the Oxford-IIIT Pet dataset is not commonly used for digital image forensics, it compensates for Boss 1.01's shortcomings. The majority of Boss 1.01's images are landscapes. They lack a prominent item in the center of an image. Even some central objects such as people or car are obscure. These landscape image will be very blur after filtering operation. It is challenging to distinguish any distinct item in an image. In contrast, images of cats and dogs in Oxford-IIIT Pet always have a clear central object. After performing filtering operation, we could still recognize the profile of a cat or dog. Boss 1.01 and Oxford-IIIT Pet contains two completely different styles of images. Our experiential results indicate that the proposed prototypical networks have good performance on both two datasets.

5.4 Designing the Convolutional Neural Networks as an Embedding Function

Convolutional neural networks have a good performance in processing tasks related to images. Convolutional neural networks take advantage to extract complex statistics and learn high levels features from datasets. Convolutional neural networks have demonstrated its successful applications in many research and commercial fields, such as image and video recognition, image classification, and self-driving system.

The key to the performance of any machine learning algorithm is to extract important features from datasets. Convolutional neural networks can automatically learn features from raw data. Earlier layers extract low-level features from input data. Low-level features will transform into high-level features throughout the operations of convolutional neural networks. The subsequent layers, such as the fully connected layers, will then utilize these features for prediction. Combining low-level features to form higher-level features is referred to as feature hierarchy.

Designing an effective embedding function $f_{\emptyset}(x_i)$ is the most crucial stage in building prototypical networks. As the embedding function in our experiment, convolutional neural networks (CNN) are utilized. In prototypical networks, each class prototype generated from meta-data embeddings is an expression of the class at a high level rather than a collection of labelled data.

The layers of a typical convolutional neural networks model comprise of many types. These layers are crucial components of convolutional neural networks. Figure 5.3 depicts the general structure of our CNN.

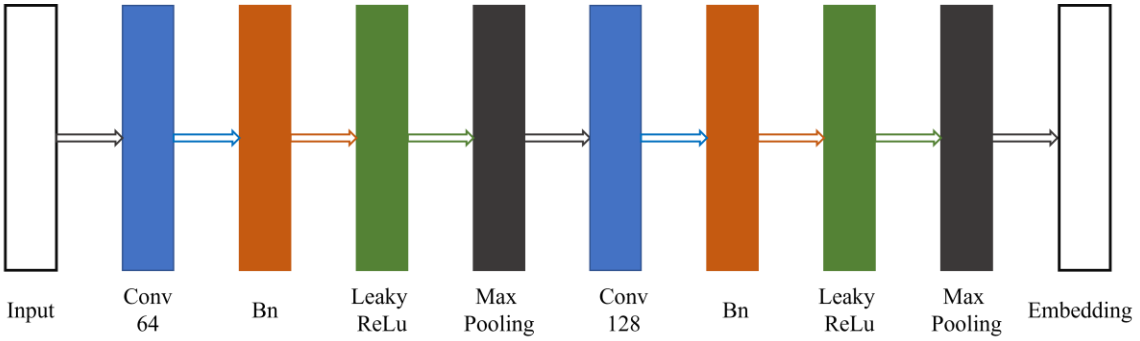


Figure 5.5 Proposed CNN architecture.

The input layer is the network's entrance. It evaluates whether the input image's dimensions match those we specify. Height, breadth, and channels denote the dimensions of an image's input. An RGB color image consists of three channels. A greyscale image has only one channel. In TensorFlow, the number of channels is one of the image dimensions. When inputted into CNN, all images must have identical dimensions. The TensorFlow program will generate errors if images with varying dimensions are input. Only the Boss 1.01 dataset contains grey images, while the other three datasets we utilized have color images, so we must convert color images to grey images before feeding them into CNN. Since we are mainly interested in how prototypical networks classify images rather than recognizing objects in an image, grey images are appropriate for our experiment. In addition, Boss 1.01 is widely utilized in digital image forensics; therefore, to validate our proposed prototypical networks, we must test our model on this dataset.

In CNN, the convolutional layer has the most important function for extracting feature maps. It consists of numerous filters. These filters have their own learnable parameters. The initial values of filter parameters were generated at random, and they will be updated during the training procedure. Our CNN architecture is composed of two convolutional layers. The first comprises 64 filters, while the second includes 128 filters.

Each filter has a window size of 3×3.

Typically, the number of convolutional layers determines the CNN's depth. However, the performance of a network with more convolutional layers may not be outstanding. When a complicated network is applied to solve a simple problem, overfitting will occur. In addition, the dimensions of feature maps are typically less than those of the input images. When an input image passes through multiple convolutional layers, feature maps with reduced dimensions are formed at the end of the final convolutional layer. How to determine the final dimensions of feature maps depends greatly on the specific problems. Training complex networks is always time-consuming. According to our experiments, increasing depth will not improve the precision of experimental results. In the proposed prototypical network, constructing two convolutional layers could achieve a balance between time and performance.

Batch normalization is a standard procedure in CNN construction. Batch normalization could make CNN training more stable and efficient. In our CNNs, we add a batch normalization layer following the convolutional layer.

CNNs require an activation layer as well. It is always following the convolutional layer but precedes the pooling layer. The activation layer imparts nonlinear properties to the extracted features via the convolution layer. A classifier with both linear and nonlinear properties will be more suitable for classification, particularly for classification of several classes. There are three classic activation functions, the Sigmoid, the ReLU and TanH which are commonly use in CNNs. Choosing an appropriate activation function for each activation will significantly impact the precision of an experiment. In our experiment, Leaky ReLU brings the best performance compared with other two classic activation

functions. Figure 5.4 illustrates the RuLu and Leaky ReLU functions.

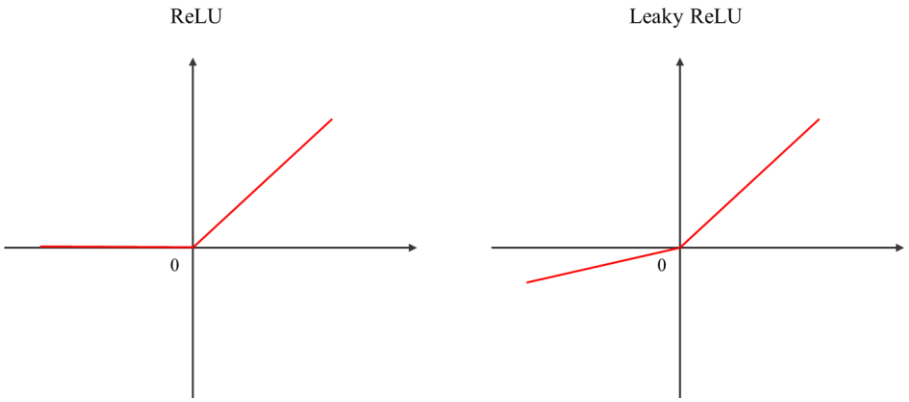


Figure 5.6 RuLu and Leaky ReLU functions.

$$\text{RuLU(x)} = \begin{cases} 0, & x \leq 0 \\ x, & x \geq 0 \end{cases} \tag{5.3}$$

$$\text{Leaky RuLU(x)} = \begin{cases} \alpha x, & x \leq 0 \\ x, & x \geq 0 \end{cases} \tag{5.4}$$

Leaky Relu is a variant of ReLU. Instead of being 0 when x<0, a leaky ReLU allows a small, non-zero, constant gradient α. From Function 5.3 and Function 5.4, we can find that Leaky ReLU could keep more information from features, especially useful for features with a lot of negative values. The constant gradient α is set to 0.2 in our experiments.

Pooling layers decreases the dimensionality of feature maps, which results in increased computational efficiency. Also, smaller dimensions of feature maps assist prevent overfitting. Average pooling and max pooling are the two basic down-sampling methods. In our experiments, we choose max pooling in every pooling layer. The window size of max pooling determines the scope of application of max pooling operation. It will calculate the maximum value within the range of the window size. We set 3×3 max pooling in both two pooling layers

After the last max pooling layer, CNN has finished extracting feature maps from the input images. These feature maps construct an image's embeddings. We then calculate the average value of all embeddings in one class. That is the prototype to represent one class of images.

5.5 Experimental Results

Systematic experimentation is a crucial component of applied machine learning. The essential experimental environment settings are arranged as below:

1. Jupyter Notebook
2. TensorFlow 1.15
3. Python 3.6.13
4. NVIDIA GeForce RTX with Max-Q Design

How to build support-sets and query-sets in meta-training and meta-testing steps are introduced separately before in section 5.3. In the meta-training step, all the meta-training tasks are 5-way 5-shot tasks. We use *mini*ImageNet dataset in this step. We set 10 epochs and each epoch contains 100 episodes for training. After meta-training step, the proposed prototypical networks model could achieve an accuracy of approximately 60% for 5-class classification. The accuracy in this range is good enough for few-shot learning algorithms applied on miniImageNet dataset. That indicates that our proposed prototypical networks model is already capable of classification. A solid meta-training accuracy lays the groundwork for further meta-testing.

In meta-testing step, the best advantage of prototypical networks is that when feed new class data which was unseen in previous meta-training step, the parameters

continuously update to accommodate the new data. In other words, when input new data to prototypical networks, it does not need to learn from sketch. Also, we do not need fine-tuning operation when we input new data. That is the most difference from training a traditional convolutional neural network with new data.

In the meta-testing step, we intend to train our proposed prototypical networks to distinguish filtered images from original images. All the meta-testing tasks are 2-way 5-shot tasks. In this step, we set 500 episodes for training.

The proposed prototypical networks are initially constructed to classify Gaussian filtered images from original images. We train and test our model on Boss 1.01 and Oxford-IIIT Pet datasets respectively. The small letter w represents the window size of a Gaussian filter. And the character σ represents standard deviation. The experimental results are show in Tables 5.1 and 5.2.

Table 5.1 Binary Classification Accuracy for Detection of Different Gaussian Filtered Images from Original Images in Boss 1.01

	Dataset: Boss 1.01		
Gaussian Filter	w:3 × 3 σ: 0.5	w:5×5 σ:1	w:7×7 σ:1.5
Accuracy	84.64%	85.98%	86.76 %

Table 5.2 Binary Classification Accuracy for Detection of Different Gaussian Filtered Images from Original Images in Oxford-IIIT Pet

	Dataset: Oxford-IIIT Pet		
Gaussian Filter	w:3 × 3 σ: 0.5	w:5×5 σ:1	w:7×7 σ:1.5
Accuracy	85.00%	85.2%	86.75%

It can be observed that the accuracy of binary classification could reach approximately 85%. Only using 5 images per class in a signal episode to train the

prototypical networks could reach fairly good experimental results. Training traditional CNN usually need prepare thousands of images. Our model totally only needs 400 images to train in meta-testing step. The proposed prototypical networks require significantly less images for training. Even our model are feed data from different sources, it could achieve relevant accuracy.

Then we conduct another experiment to test if our proposed method is capable to differentiate the average filtered images from original images. We just repeat the meta-training stage and then proceed to the meta-testing step. Similarly, we train our model on two datasets separately. Various window size of average filters could lead to different filtering effects. Tables 5.3 and 5.4 are our experimental results.

Table 5.3 Binary Classification Accuracy for Detection of Different Average Filtered Images from Original Images in Boss 1.01

	Dataset: Boss 1.01		
Average Filter	w:3 × 3	w:5×5	w:7×7
Accuracy	85.56 %	85.22 %	82.38%

Table 5.4 Binary Classification Accuracy for Detection of Different Average Filtered Images from Original Images in Oxford-IIIT Pet

	Oxford-IIIT Pet		
Average Filter	w:3 × 3	w:5×5	w:7×7
Accuracy	83.50 %	84.44%	84.64%

We can find that the accuracy could still reach to approximately 85%. We simply replace the input images from Gaussian filtered images to average filtered images. We do not need fine-tuning also. The fact that our proposed prototypical networks can tackle a variety of digital image forensics tasks despite varying datasets demonstrates their adaptability.

5.6 Summary

The meta-learning framework for few-shot learning is based on the principle of learning to learn with few data. Thanks to the development of meta-learning and few-shot learning, these novel machine learning models could be applied to a variety of tasks and achieve satisfactory accuracies. In addition, training these new types of machine learning models requires far less data.

In this chapter, we present a prototypical networks model for different digital image forensics tasks. Without a complicated neural network design, our model could classify Gaussian filtered and average filtered images from original images with high accuracy. From the perspective of datasets, our experiments demonstrate that the proposed prototypical networks have the flexibility to classify filtered images with high accuracies on different datasets. When utilizing datasets from different sources, there may be some limitations. For example, images have various sizes or channels. These constraints from datasets may require more consideration in future work.

CHAPTER 6

CONTRIBUTION AND FUTURE WORK

6.1 Major Contributions

In this dissertation, methods based on machine learning are used to address digital picture forensics challenges. In the first chapter, the history of neural networks is briefly discussed. From perceptrons to today's complex neural networks, neural network development has been a tremendous success. The second chapter describes the fundamental mathematical processes of neural networks. The convolutional layer, pooling layer, and activation layer are thoroughly analyzed.

In Chapter 3, a convolutional neural network architecture is developed and applied to the detection of seam carving. It is the first deep learning framework on this study issue. Experimental results demonstrate that the proposed deep learning method can successfully detect seam carving in uncompressed digital images and outperforms the state-of-the-art in the majority of research. In many instances, the experimental results indicate that the detection accuracy can approach 99%. In particular, the proposed deep convolutional neural network has demonstrated good performance in recognizing situations with low seam carving rates.

In Chapter 4, we propose a method for real-time estimation of Gaussian filtered image parameters. This method is based on the capability of convolutional neural networks to classify multiple classes. Experiments were conducted to assess the overall effectiveness of the proposed method. Based on the data and discussion, the proposed network is capable of achieving high estimating accuracy quickly, allowing it to serve as an parameters

estimator in real time.

In Chapter 5, we provide a prototypical networks model for various digital image forensics tasks. Our model could detect Gaussian filtered and average filtered images from original images without requiring a complex neural network architecture. In terms of datasets, our experiments show that the proposed prototypical networks can classify filtered images with good accuracies on a variety of datasets.

6.2 Limitations

Convolutional neural networks have seen tremendous success in recent years. The structure of neural networks is becoming increasingly complicated. Convolutional neural network models with more than 100 layers are available. Parallel architectures are also used in some convolutional neural network models.

Complex structures are not designed in neural network models in this dissertation. Even if simple neural network structures are sufficient for our studies. More complicated neural network structures, on the other hand, deserve consideration.

Also, the data source also has an impact on the experimental results. Only images from a signal dataset are used in some of the experiments in this dissertation. Testing images from various sources aids in the improvement of neural network model performance.

6.3 Future Research

Image manipulation is becoming more accessible. Ordinary individuals can now easily modify images without the need for professional skills. Image detection techniques that

work today may no longer be applicable in the future. As a result, image detection technology must be constantly updated. In the future research, More complicated neural network structures will be designed. In addition, more datasets will be tested in a neural network model. Thus, a neural network model could be effective for future digital image forensics tasks.

REFERENCES

[1] Ciregan, D., Meier, U., & Schmidhuber, J. (2012, June). Multi-column deep neural networks for image classification. In 2012 *IEEE conference on computer vision and pattern recognition* (pp. 3642-3649). IEEE.

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[4] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[5] Patrick, R. L. (1987). *General motors/North American monitor for the IBM 704 computer* (Vol. 7316). Rand.

[6] Lorenzo, M. J. (2019). *Abstracting Away the Machine: The History of the FORTRAN Programming Language (FORmula TRANslation).* Independently published.

[7] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678).

[8] LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

[9] LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, 19(143-155), 18.

[10] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541-551.

[11] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

[12] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013, May). Regularization of neural networks using drop connect. In *International conference on machine learning* (pp. 1058-1066). PMLR.

[13] Piva, A. (2013). An overview on image forensics. *International Scholarly Research Notices, 2013*.

[14] Avidan, S., & Shamir, A. (2007). Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 papers* (pp. 10-es).

[15] Sarkar, A., Nataraj, L., & Manjunath, B. S. (2009, September). Detection of seam carving and localization of seam insertions in digital images. In *Proceedings of the 11th ACM workshop on multimedia and security* (pp. 107-116).

[16] Fillion, C., & Sharma, G. (2010, January). Detecting content adaptive scaling of images for forensic applications. In *Media forensics and security II* (Vol. 7541, pp. 359-370). SPIE.

[17] Lu, W., & Wu, M. (2011, September). Seam carving estimation using forensic hash. In *Proceedings of the thirteenth ACM multimedia workshop on multimedia and security* (pp. 9-14).

[18] Chang, W. L., Shih, T. K., & Hsu, H. H. (2013, November). Detection of seam carving in JPEG images. In *2013 International joint conference on awareness science and technology & ubi-media computing (iCAST 2013 & UMEDIA 2013)* (pp. 632-638). IEEE.

[19] Wattanachote, K., Shih, T. K., Chang, W. L., & Chang, H. H. (2015). Tamper detection of JPEG image due to seam modifications. *IEEE transactions on information forensics and security*, *10*(12), 2477-2491.

[20] Liu, Q., & Chen, Z. (2014). Improved approaches with calibrated neighboring joint density to steganalysis and seam-carved forgery detection in JPEG images. *ACM transactions on intelligent systems and technology (TIST)*, *5*(4), 1-30.

[21] Liu, Q. (2016, December). Exposing seam carving forgery under recompression attacks by hybrid large feature mining. In *2016 23rd International conference on pattern recognition (ICPR)* (pp. 1041-1046). IEEE.

[22] Liu, Q. (2017). An approach to detecting JPEG down-recompression and seam carving forgery under recompression anti-forensics. *Pattern recognition*, 65, 35-46.

[23] Ryu, S. J., Lee, H. Y., & Lee, H. K. (2014). Detecting trace of seam carving for forensic analysis. *IEICE TRANSACTIONS on Information and systems*, *97*(5), 1304-1311.

[24] Wei, J. D., Lin, Y. J., & Wu, Y. J. (2014). A patch analysis method to detect seam carved images. *Pattern recognition letters*, 36, 100-106.

[25] Yin, T., Yang, G., Li, L., Zhang, D., & Sun, X. (2015). Detecting seam carving based image resizing using local binary patterns. *Computers & security*, 55, 130-141.

[26] Ye, J., & Shi, Y. Q. (2016, September). A local derivative pattern based image forensic framework for seam carving detection. In *International workshop on digital watermarking* (pp. 172-184). Springer, Cham.

[27] Ye, J., & Shi, Y. Q. (2017). An effective method to detect seam carving. *Journal of information security and applications*, *35*, 13-22.

[28] Ye, J., & Shi, Y. Q. (2017, August). A hybrid feature model for seam carving detection. In *International workshop on digital watermarking* (pp. 77-89). Springer, Cham.

[29] Zhang, D., Yin, T., Yang, G., Xia, M., Li, L., & Sun, X. (2017). Detecting image seam carving with low scaling ratio using multi-scale spatial and spectral entropies. *Journal of visual communication and image representation*, *48*, 281-291.

[30] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

[31] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

[33] Xu, G., Wu, H. Z., & Shi, Y. Q. (2016). Structural design of convolutional neural networks for steganalysis. *IEEE signal processing letters*, *23*(5), 708-712.

[34] Fridrich, J., & Kodovsky, J. (2012). Rich models for steganalysis of digital images. *IEEE transactions on information forensics and security*, *7*(3), 868-882.

[35] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[36] Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315-323). JMLR Workshop and Conference Proceedings.

[37] Bas, P., Filler, T., & Pevný, T. (2011, May). " Break our steganographic system": the ins and outs of organizing BOSS. In *International workshop on information hiding* (pp. 59-70). Springer, Berlin, Heidelberg.

[38] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on multimedia* (pp. 675-678).

[39] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 1-27.

[40] Farid, H. (2009). Image forgery detection. *IEEE signal processing magazine*, *26*(2), 16-25.

[41] Fridrich, J. (2009). Digital image forensics. *IEEE signal processing magazine*, *26*(2), 26-37.

[42] Piva, A. (2013). An overview on image forensics. *International scholarly research notices*, *2013*.

[43] Swaminathan, A., Wu, M., & Liu, K. R. (2008). Digital image forensics via intrinsic fingerprints. *IEEE transactions on information forensics and security*, *3*(1), 101-117.

[44] Qi, L., Wang, R., Hu, C., Li, S., He, Q., & Xu, X. (2019). Time-aware distributed service recommendation with privacy-preservation. *Information sciences*, *480*, 354-364.

[45] Qi, L., Zhang, X., Dou, W., Hu, C., Yang, C., & Chen, J. (2018). A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. *Future generation computer systems*, *88*, 636-643.

[46] Jung, K. H., & Yoo, K. Y. (2015). Steganographic method based on interpolation and LSB substitution of digital images. *Multimedia tools and applications*, *74*(6), 2143-2155.

[47] Meng, R., Rice, S. G., Wang, J., & Sun, X. (2018). A fusion steganographic algorithm based on faster R-CNN. *Computers, materials & continua*, *55*(1), 1-16.

[48] Lyu, S., & Farid, H. (2006). Steganalysis using higher-order image statistics. *IEEE transactions on information forensics and security*, *1*(1), 111-119.

[49] Silva, E., Carvalho, T., Ferreira, A., & Rocha, A. (2015). Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes. *Journal of visual communication and image representation*, *29*, 16-32.

[50] Lee, J. C. (2015). Copy-move image forgery detection based on Gabor magnitude. *Journal of visual communication and image representation*, *31*, 320-334.

[51] Lukas, J., Fridrich, J., & Goljan, M. (2006). Digital camera identification from sensor pattern noise. *IEEE transactions on information forensics and security*, *1*(2), 205-214.

[52] Li, C. T. (2010). Source camera identification using enhanced sensor pattern noise. *IEEE transactions on information forensics and security*, *5*(2), 280-287.

[53] Ding, F., Zhu, G., Yang, J., Xie, J., & Shi, Y. Q. (2014). Edge perpendicular binary coding for USM sharpening detection. *IEEE signal processing letters*, *22*(3), 327-331.

[54] Ding, F., Zhu, G., Dong, W., & Shi, Y. Q. (2018). An efficient weak sharpening detection method for image forensics. *Journal of visual communication and image representation*, *50*, 93-99.

[55] Zhu, N., Deng, C., & Gao, X. (2017). Image sharpening detection based on multiresolution overshoot artifact analysis. *Multimedia tools and applications*, *76*(15), 16563-16580.

[56] Guo, J. M., & Le, T. N. (2010). Secret communication using JPEG double compression. *IEEE signal processing letters*, *17*(10), 879-882.

[57] Barni, M., Bondi, L., Bonettini, N., Bestagini, P., Costanzo, A., Maggini, M., ... & Tubaro, S. (2017). Aligned and non-aligned double JPEG detection using convolutional neural networks. *Journal of visual communication and image representation*, *49*, 153-163.

[58] Yang, J., Xie, J., Zhu, G., Kwong, S., & Shi, Y. Q. (2014). An effective method for detecting double JPEG compression with the same quantization matrix. *IEEE transactions on information forensics and security*, *9*(11), 1933-1942.

[59] Zhang, Y., Li, S., Wang, S., & Shi, Y. Q. (2014). Revealing the traces of median filtering using high-order local ternary patterns. *IEEE signal processing letters*, *21*(3), 275-279.

[60] Boroumand, M., & Fridrich, J. (2017). Scalable processing history detector for jpeg images. *Electronic imaging*, *2017*(7), 128-137.

[61] Cao, G., Zhao, Y., Ni, R., Yu, L., & Tian, H. (2010, July). Forensic detection of median filtering in digital images. In *2010 IEEE international conference on multimedia and expo* (pp. 89-94). IEEE.

[62] Ravi, H., Subramanyam, A. V., & Emmanuel, S. (2015). ACE–an effective anti-forensic contrast enhancement technique. *IEEE signal processing letters*, *23*(2), 212-216.

[63] Stamm, M. C., Tjoa, S. K., Lin, W. S., & Liu, K. R. (2010, September). Undetectable image tampering through JPEG compression anti-forensics. In *2010 IEEE international conference on image processing* (pp. 2109-2112). IEEE.

[64] Li, J., Li, X., Yang, B., & Sun, X. (2014). Segmentation-based image copy-move forgery detection scheme. *IEEE transactions on information forensics and security*, *10*(3), 507-518.

[66] Rhee, K. H. (2016, October). Gaussian filtering detection using band pass residual and contrast of forgery image. In *2016 IEEE 7th annual information technology, electronics and mobile communication conference (IEMCON)* (pp. 1-4). IEEE.

[67] Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *ArXiv preprint arXiv:1606.02147*.

[68] Xu, G., Wu, H. Z., & Shi, Y. Q. (2016). Structural design of convolutional neural networks for steganalysis. *IEEE signal processing letters*, *23*(5), 708-712.

[69] Chen, J., Kang, X., Liu, Y., & Wang, Z. J. (2015). Median filtering forensics based on convolutional neural networks. *IEEE signal processing letters*, *22*(11), 1849-1853.

[70] Bondi, L., Baroffio, L., Güera, D., Bestagini, P., Delp, E. J., & Tubaro, S. (2016). First steps toward camera model identification with convolutional neural networks. *IEEE signal processing letters*, *24*(3), 259-263.

[71] Cui, Q., McIntosh, S., & Sun, H. (2018). Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs. *Comput. Mater. Continua*, *55*(2), 229-241.

[72] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 1-27.

[73] Ding, F., Shi, Y., Zhu, G., & Shi, Y. Q. (2019). Smoothing identification for digital image forensics. *Multimedia tools and applications*, *78*(7), 8225-8245.

[74] Yang, J., Zhu, G., Huang, J., & Zhao, X. (2015). Estimating JPEG compression history of bitmaps based on factor histogram. *Digital signal processing*, *41*, 90-97.

[75] Zhou, Z., Wu, Q. J., & Sun, X. (2019). Multiple distance-based coding: toward scalable feature matching for large-scale web image search. *IEEE transactions on big data*, *7*(3), 559-573.

[76] Ye, J., Shen, Z., Behrani, P., Ding, F., & Shi, Y. Q. (2018). Detecting USM image sharpening by using CNN. *Signal processing: image communication*, *68*, 258-264.

[77] Yuan, C., Li, X., Wu, Q. J., Li, J., & Sun, X. (2017). Fingerprint liveness detection from different fingerprint materials using convolutional neural network and principal component analysis. *Computers, materials & continua*, *53*(3), 357-371.

[78] Slavkovikj, V., Verstockt, S., De Neve, W., Van Hoecke, S., & Van de Walle, R. (2015, October). Hyperspectral image classification with convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 1159-1162).

[79] Bas, P., Filler, T., & Pevný, T. (2011, May). " Break our steganographic system": the ins and outs of organizing BOSS. In *International workshop on information hiding* (pp. 59-70). Springer, Berlin, Heidelberg.

[80] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[81] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484-489.

[82] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint arXiv:1810.04805*.

[83] Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, *28*(4), 594-611.

[84] Fink, M. (2004). Object classification from a single example utilizing class relevance metrics. *Advances in neural information processing systems*, *17*.

[85] Altae-Tran, H., Ramsundar, B., Pappu, A. S., & Pande, V. (2017). Low data drug discovery with one-shot learning. *ACS central science*, *3*(4), 283-293.

[86] Wu, Y., & Demiris, Y. (2010, May). Towards one shot learning by imitation for humanoid robots. In *2010 IEEE international conference on robotics and automation* (pp. 2889-2894). IEEE.

[87] Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., ... & Zaremba, W. (2017). One-shot imitation learning. *Advances in neural information processing systems*, *30*.

[88] Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126-1135). PMLR.

[89] Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., & Ahn, S. (2018). Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, *31*.

[90] Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, *30*.

[91] Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, *29*.

[92] Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126-1135). PMLR.

[93] Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, *31*.

[94] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., ... & De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, *29*.

[95] Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, *31*.

[96] Ravi, S., & Larochelle, H. (2016). Optimization as a model for few-shot learning.

[97] Xu, G., Wu, H. Z., & Shi, Y. Q. (2016). Structural design of convolutional neural networks for steganalysis. *IEEE signal processing letters*, *23*(5), 708-712.

[98] Bondi, L., Baroffio, L., Güera, D., Bestagini, P., Delp, E. J., & Tubaro, S. (2016). First steps toward camera model identification with convolutional neural networks. *IEEE Signal Processing Letters*, *24*(3), 259-263.

[99] Cui, Q., McIntosh, S., & Sun, H. (2018). Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs. *Comput. Mater. Continua*, *55*(2), 229-241.

[100] Ding, F., Shi, Y., Zhu, G., & Shi, Y. Q. (2020). Real-time estimation for the parameters of Gaussian filtering via deep learning. *Journal of real-time image processing*, *17*(1), 17-27.

[101] Chen, J., Kang, X., Liu, Y., & Wang, Z. J. (2015). Median filtering forensics based on convolutional neural networks. *IEEE signal processing letters*, *22*(11), 1849-1853.

[102] Ye, J., Shi, Y., Xu, G., & Shi, Y. Q. (2018, October). A convolutional neural network based seam carving detection scheme for uncompressed digital images. In *International workshop on digital watermarking* (pp. 3-13). Springer, Cham.

[103] Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, *29*.

[104] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

[105] Bas, P., Filler, T., & Pevný, T. (2011, May). " Break our steganographic system": the ins and outs of organizing BOSS. In *International workshop on information hiding* (pp. 59-70). Springer, Berlin, Heidelberg.

[106] Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2012, June). Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3498-3505). IEEE.