WELD PENETRATION IDENTIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORK

Chao Li
University of Kentucky, cli284@uky.edu

Digital Object Identifier: https://doi.org/10.13023/etd.2019.003

WELD PENETRATION IDENTIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORK

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Chao Li
Lexington, Kentucky
Director: Dr. Yuming Zhang, Professor of Electrical and Computer Engineering
Lexington, Kentucky
2018

ABSTRACT OF DISSERTATION

WELD PENETRATION IDENTIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORK

Weld joint penetration determination is the key factor in welding process control area. Not only has it directly affected the weld joint mechanical properties, like fatigue for example. It also requires much of human intelligence, which either complex modeling or rich of welding experience. Therefore, weld penetration status identification has become the obstacle for intelligent welding system. In this dissertation, an innovative method has been proposed to detect the weld joint penetration status using machine-learning algorithms.

A GTAW welding system is firstly built. Project a dot-structured laser pattern onto the weld pool surface during welding process, the reflected laser pattern is captured which contains all the information about the penetration status. An experienced welder is able to determine weld penetration status just based on the reflected laser pattern. However, it is difficult to characterize the images to extract key information that used to determine penetration status. To overcome the challenges in finding right features and accurately processing images to extract key features using conventional machine vision algorithms, we propose using convolutional neural network (CNN) to automatically extract key features and determine penetration status.

Data-label pairs are needed to train a CNN. Therefore, an image acquiring system is designed to collect reflected laser pattern and the image of work-piece backside. Data augmentation is performed to enlarge the training data size, which resulting in 270,000 training data, 45,000 validation data and 45,000 test data. A six-layer convolutional neural network (CNN) has been designed and trained using a revised mini-batch gradient descent optimizer. Final test accuracy is 90.7% and using a voting mechanism based on three consequent images further improve the prediction accuracy.

KEY WORDS: Gas tungsten arc welding (GTAW), computer vision, weld penetration, machine learning, convolutional neural network (CNN)

Chao Li

Nov 29, 2018

WELD PENETRATION IDENTIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORK

By
Chao Li

Yuming Zhang, Ph.D.
Director of Dissertation

Aaron Cramer, Ph.D.
Director of Graduate Studies

Nov 29, 2018
Date

Acknowledgements

I would firstly thank to my advisor Dr. Yuming Zhang for his invaluable guidance, encouragement and instructions. In addition, I thank to my co-advisor Dr. Michael T. Johnson and Dr. Qiang Ye, Dr. Dan Inoel for their great support in my study in University of Kentucky. I want to thank all of my colleagues in the welding lab: Wenhua Jiao, Qiyue Wang for their great helpful suggestions in my work.

In addition, I want to thanks to my parents for their endless love and support throughout my life.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background

Gas tungsten arc welding (GATW) and gas metal arc welding are the two prevalent welding processes in industrial manufacturing. The theory behind them is similar, generating heat by using electric arc between the electrode and the work-piece. In GTAW, a non-consumable tungsten is used as electrode to emit electrons, which established stable arc with work-piece. The work-piece being welded forms a liquid weld pool by the heat of the arc and joints. The two pieces of the work-piece are welded together after cooling. An optional filler metal maybe used, if necessary. During the welding process, an inert gas, argon for example covers the weld pool surface protecting it from contamination. This process is illustrated in Figure 1.1. Unlike GTAW, gas metal arc welding (GMAW) uses a consumable electrode wire, consistently generating weld droplet into the arc zone. After the droplet solidification, the work-piece are jointed together.

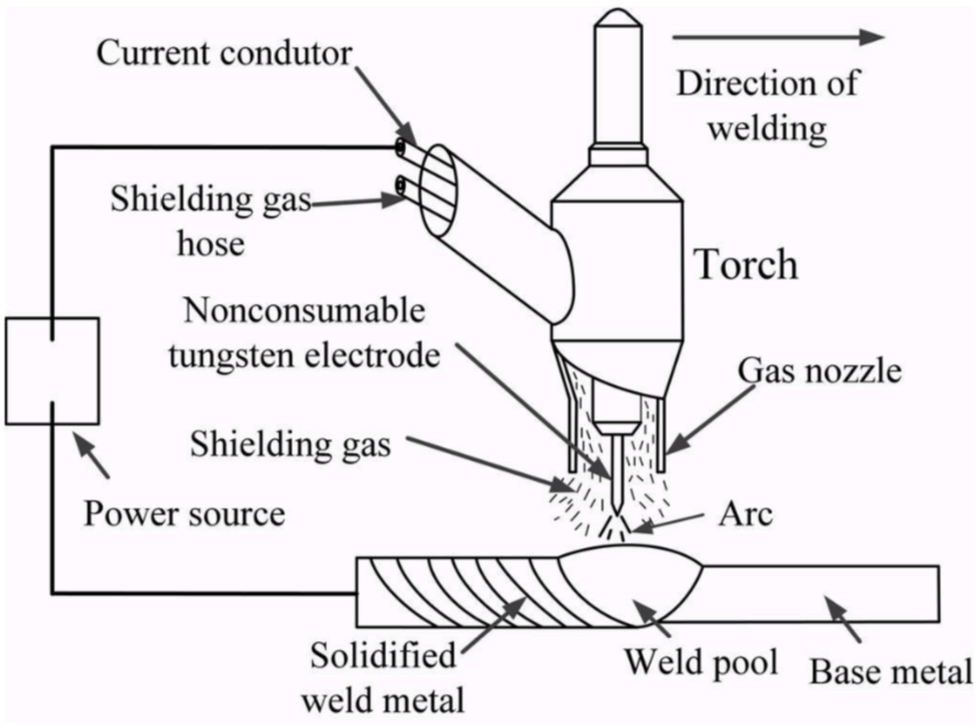


Figure 1. 1 GTAW welding process

GTAW is commonly used in critical cases including pressure vessels, aerospace, etc. due to its stability and high-quality weld joints produced. In these cases, the degree of penetration status is an important criterion to judge weld joint integrity and affects mechanical properties especially fatigue properties and service life of weld structure. Therefore, an experienced welder is crucial since he/she is able to appraise penetration status of backside and make adjustments (weld current, weld speed, etc.) based on the observation of weld pool surface during welding. However, manually welding requires the welder to keep concentrating for long time during welding. Health issues including high stress, dry eyes, etc. become obvious and that dramatically affects welder's reaction time, concentration time resulting in weld quality degrade.

On the other hand, welding robots are designed to weld long time consistently without quality issue. In addition, unlike human, weld robots can be placed and perform well under harsh environments such as high temperature, strong arc light. However, current welding robots are lack of intelligence: the movement of the robots are pre-programmed, the welding parameters are pre-set, even the position of the weld-piece is strictly limited under small variation. Even some weld robots are equipped with sensing equipment like camera, the degree of weld penetration cannot be precisely determined. What is worse, the sensing equipment are highly cost and hard to be setup, which means the flexibility of welding robots is degraded with sensing equipment. Therefore, a welding system that determines weld penetration status automatically is urgent needed in current manufacturing industry.

1.2 Objective and Approach

As discussed in last section, human being is able to make adjustments during welding, which is very important for critical parts, but the weld quality degrades along the time. Welding robot ensures the weld quality but only good for simple tasks with no complicated adjustments needed in welding process. The purpose of this research is to propose a welding system that combines human's intelligence and robot's consistency, which means automatically determine the weld penetration status during welding process, step closer to intelligent manufacturing. To endow the weld robot with human being's intelligence, we propose to use machine learning algorithms especially convolutional neural network (CNN). Therefore, the objectives of this study are:

1. To build an GTAW welding system to run welding process fast and consistently;
2. To establish an image acquiring system to collect data about the weld penetration status during welding process. Transfer the human being's knowledge into a format that weld robots are able to recognize;
3. To design and train a convolutional neural network using collected data, precisely predict weld penetration status offline;
4. To apply the trained convolutional neural network into the welding system. Build an online control mechanism using CNN, precisely determine penetration status during welding process.

The biggest challenge is to "teach" a weld robot to justify different weld penetration status. Conventional methods try to solve this by finding certain key features that directly related to the weld penetration status. This process contains but not limited to creating complex models, numerous mathematical operations, etc. In summary, using human's intelligence to simplify welding process into several key features that easily to be tracked by welding robots. However, no such model has been designed so far and the prediction accuracy is not good enough. Another way is to improve sensing method, such as using infrared cameras[1-3], ultrasonic sensing[4, 5], X-ray[6]. However, besides the high cost on equipment, the welding robots equipped by equipment will lost their flexibility and harsh environment endurance. As a sub-method of machine learning, convolutional neural network has achieved impressive performance in computer vision area, including classification[7, 8], segmentation[9-11],etc. Recall that an experienced welder is able to determine penetration status by observing weld pool surface. Inspired by animal's visual cortex[12], a well-trained CNN has the ability to determine penetration status. The next challenge is letting the welding robot “see” the weld pool like human. We use two cameras to capture both the weld pool surface and the backside of the weld pool. The images of the backside are used as labels when we train the convolutional neural network. The creating label process is crucial since we transferred human knowledge into a format that the CNN understands. We will discuss in next chapters. Another challenge is about the data size. Current machine learning learners share the same dataset, like MINST[13], Caltech-256[14], ImageNet[15] etc. However, no dataset contains the weld images we need. Therefore, we need to create our own dataset. Collect all data by welding seems impossible since a typical dataset contains more than 10,000 samples (MINIST contains 70,000 samples, Caltech-256 contains 30,607 samples, ImageNet contains over 14 million samples). Therefore, data augmentation is needed to create enough data.

1.3 Dissertation Outline

In this dissertation, an intelligent welding system that automatically collect data, determine penetration status, control welding process is developed. The main research approach and results are discussed in the following chapters. The dissertation is organized as follows.

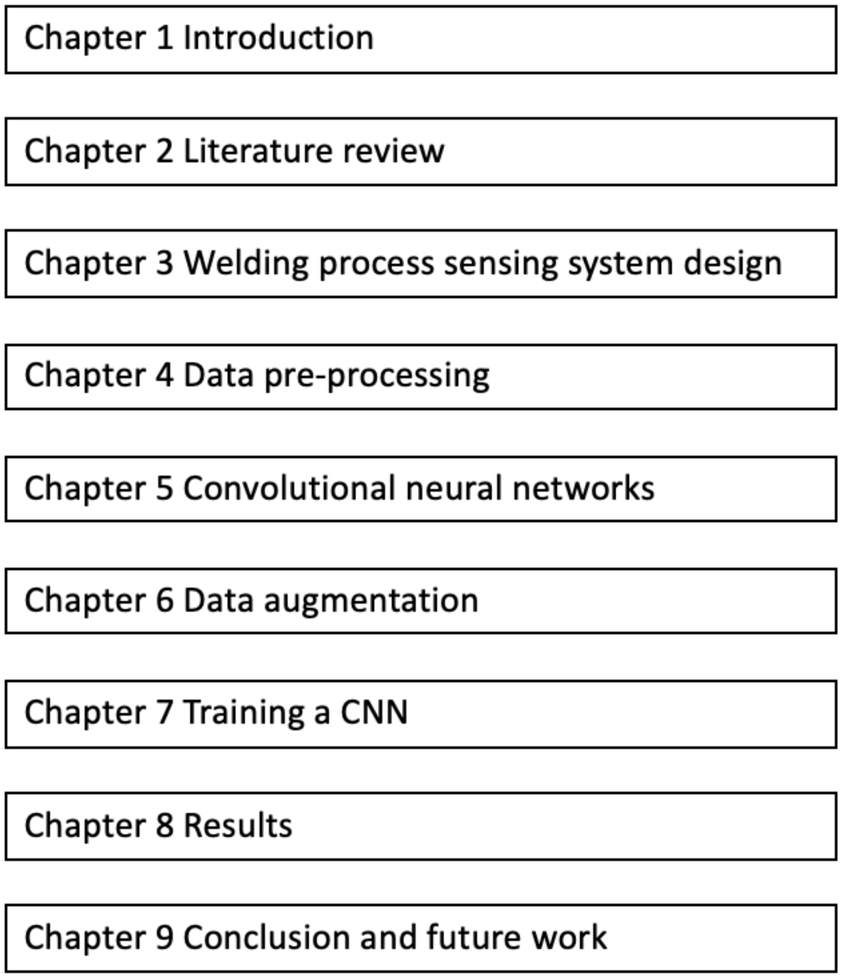

Chapter 1 Introduction
Chapter 2 Literature review
Chapter 3 Welding process sensing system design
Chapter 4 Data pre-processing
Chapter 5 Convolutional neural networks
Chapter 6 Data augmentation
Chapter 7 Training a CNN
Chapter 8 Results
Chapter 9 Conclusion and future work

Figure 1. 2 Organization of dissertation

Chapter 1: Introduction

The background and motivation of this dissertation is discussed, as well as the objective of this study.

Chapter 2: Literature Review

In this chapter, the conventional sensing methods are discussed, including pool oscillation, infrared, ultrasonic, acoustic emission, and vision-based sensing method.

Chapter 3 Welding process system sensing design

A GTAW welding system is built and a machine vision-based sensing system is designed based on that welding system, which includes two cameras, a dot matrix laser pattern, and a screen. A dot matrix structured laser is projected onto the weld pool surface, showing the changes of the weld pool during welding process. On the other side, the reflected laser is collected by a screen, which is placed, on the exact reflected path. A high-speed camera is used to capture the images on the screen. At the same time, another camera is capturing the backside of the weld pool. The image pairs between surface and backside of weld pool have been collected.

Chapter 4 Data pre-processing

Images of both weld pool surface and weld pool backside are captured in last chapter. Before these images are sent to train the neural network, pre-processing needs to be done. Different weld penetration status are identified using human's knowledge. Human being's knowledge are transforming into the way computer understands. Data-label pairs are established.

Chapter 5 Convolutional neural networks

The prevalent machine learning method in computer vision area, convolutional neural network is discussed in this chapter. Four basic components: convolutional layer, pooling layer, neural network and regression layer are presented in theory. A six-layer CNN is designed to learn weld penetration status. Details of the architecture including neuron numbers in each layer, number of parameters are discussed.

Chapter 6 Data augmentation

Even weld over 300 times, the data size is far smaller than training this six-layer CNN needed. It is impossible to run over 10,000 welding process to collect enough data. Data augmentation is performed to enlarge data size, resulting in 270,000 training set, 45,000 validation set and 45,000 test data.

Chapter 7 Training a convolutional neural network

Different optimizers (mini-batch gradient descent and Adam) are discussed. Learning rate annealing is performed to get more accurate result. Early stopping ensures the training efficiency. Finally, batch normalization is added as a way of pre-processing, decrease the over-fitting risk.

Chapter 8 Results

Preliminary results are showed and discussed in this chapter. Further propose a voting method based on three continuous images improve the predict accuracy. Apply the trained CNN to control real welding process.

Chapter 9 Conclusion and future work

The main finding and contributions are concluded and the future work to improve this method is discussed.

Chapter 2 Literature Review

Welding process has been widely applied in current industry manufacturing, including automotive assembly, aircraft production, micro-electric components, etc. Big batch manufacturing is not the trend right now; small batch, personalized manufacturing requires innovative intelligent welding system. However, the extreme brightness of the arc light makes sensing the weld process hard. Direct way of sensing is to observe the backside of the weld pool, but work piece position makes it even harder. Therefore, huge methods have been proposed to sense the welding process, including pool oscillation, infrared-based method, ultrasonic-based method and computer vision-based method. In this chapter, all of them will be discussed.

2.1 Full penetration and partial penetration

As discussed, GTAW has been widely used in industrial manufacturing especially in critical cases. In these critical cases, the weld joint penetration status is the most important criterion. Typically, there are three-penetration status: partial penetration, full penetration and over penetration. Moreover, at the beginning of welding there exist a status, no penetration at all. Since partial penetration and full penetration are desired in real industry welding, researchers have paid much attention on distinguishing them. Figure 2.1 shows partial penetration and full penetration.

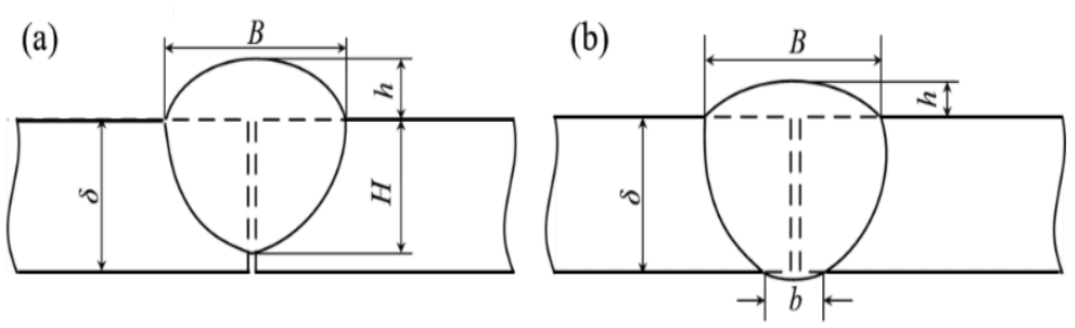


Figure 2. 1 Weld joint penetration (a) partial penetration (b) full penetration

Under full penetration, shown in figure (b) in 2.2, the weld bead reached the backside of the weld joint, causing the work-piece completely weld together. Welding joints mechanical properties like fatigue property are better than partial penetration. But achieving full penetration is much harder than partial penetration. Precisely control of the welding process are required: making sure no partial penetration or over penetration. Partial penetration on the other hand, is easily to achieve. So, for less critical part, paritial penetration is preferred.

2.2 Welding process sensing

Sensing welding process is the basic of weld process control, the information it collected directly determine the complexity of control methods. As discussed, the invisible of the weld pool backside makes it hard to sensing weld process. In this chapter, we will discuss these sensing methods.

2.2.1 Pool Oscillation method

Track back to 1972, Kotecki et al. [16] firstly found the oscillation phenomena of the weld pool (diameter of weld pool is correlate with the natural frequency) by doing stationary

GTA welding process. Following researchers like Richardson et al[17]. proposed the natural frequency is strongly dependent on the inverse of the square root of pool mass. However, these pioneering works have poor accuracy and cannot been applied in moving welding process. Later, the abrupt transition of the weld pool's natural oscillation frequency from partial penetration to full penetration has been found and applied to monitor and control the weld joint penetration by Xiao and Ouden [18, 19]. The finding that natural oscillation frequency in partial penetration is much higher than that in full penetration pave the way for the following researchers. K. Andersen et al[20]. proposed a closed-loop feedback welding control system by implementing synchronous weld pool pulsing method. B.Y.B.Yudodibroto[21] et al. further discussed the weld pool oscillation method successfully applied GTAW with cold filler wire addition. However, the accuracy of the oscillation methods are affected by the moving speed of the welding robot. In addition, the work-piece surface need to be carefully cleaned in case the dirt or oxide causing natural frequency changes. Therefore, the application of pool oscillation sensing methods is under small range.

2.2.2 Infrared-based sensing method

The infrared-based sensing uses thermal sensor for example infrared camera to track the weld pool properties, like penetration status, weld bead width, etc. during welding. Chen et al. [1-3] proposed that the depth of welding joint penetration was determined by construct the thermal distribution of the weld pool surface based on the infrared thermal images that are captured by the infrared camera. The infrared has been widely used, but the infrared sensors are expensive. What is worse, the accuracy of sensing directly affected by the environment, like lighting conditions.

2.2.3 Ultrasonic-based sensing method

The ultrasonic wave is project to the find the boundaries between the liquid weld pool and the work-piece[4, 5]. The ultrasonic wave transmission speed is different in different materials; therefore, by calculating the time reflected ultrasonic wave is received, the depth of weld penetration is determined. However, to accurately measure the depth, the work-piece material need to be uniform and contains low percentage of impurity. In addition, the surface of work-piece must be clean and even to ensure effective coupling. Although non-contact ultrasonic-based sensing has been proposed, such as laser ultrasonic[22] to remedy contact ultrasonic-based sensing, these systems requires special calibration and not easy to be applied in industry.

2.2.4 Computer vision-based sensing method

Computer vision-based sensing has been widely used since its cheap cost, easy to setup and relatively acceptable accuracy. Unlike the methods discussed before, computer vision-based method cannot provide information directly relate to weld pool penetration status. Extra steps like weld pool reconstruction are needed. A typical vison sensing system uses one or multiple cameras to capture the weld pool surface during welding process. For some cases, like tube welding, the backside of the weld pool is hard to capture. Optional optical filters are needed to filter the strong arc light. The images of the weld pool surface contain enough information to reconstruct a 2D or 3D weld pool. R.Kovacevic et al. [23] propose an on-line welding pool edge detection sensing system, show in figure 2.2.

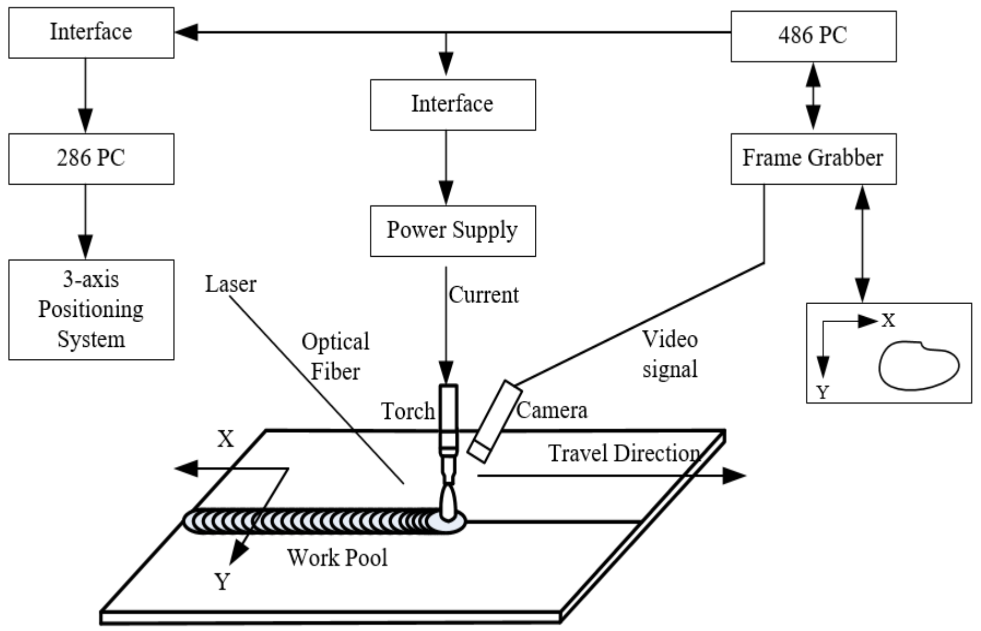


Figure 2. 2 On-line weld pool edge detection system

The camera is capturing the weld pool surface each time the laser is paused to avoid strong arc light affecting the captured images. The welding process is controlled by an adaptive method. Another research that is done by University of Kentucky proposed to reconstruct weld pool using computer vision-based sensing [24-26]. Project the dot-matrix structured laser pattern onto the welding pool, on the other side, a screen is placed on the path of reflected pattern to collect the reflected pattern. Details in figure 2.3. An iterative algorithm has been designed that only based on the width, length, convexity extracting from the reflected pattern.

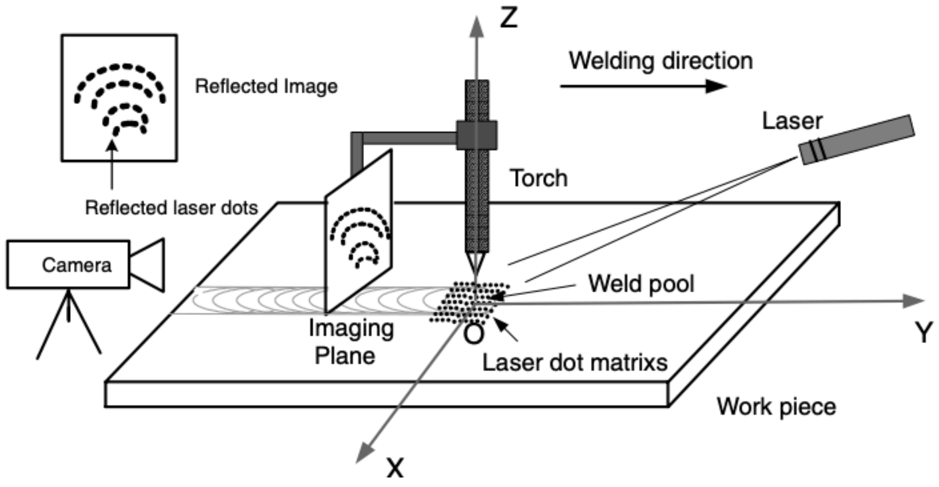


Figure 2. 3 Weld pool reconstruct by width, length and convexity

The computer vision-based is widely used in current industry due to its cheap cost and simple setup. However, extra work are needed including complex modeling process, mathematical operations. Researchers have been trying to find better algorithms to extract the key features that related with the weld penetration status.

Chapter 3 Welding process system sensing design

As discussed, numerous sensing methods have been proposed to accurately monitor the welding process. The information sensing methods provided directly affects the control algorithm. A good sensing system should have at least three properties: accuracy, fast response and robustness. Accuracy means the sensing system provide correct information about the welding process, which is the basic requirement for a sensing system. Welding process is a dynamic process, which means precise control requires real-time information. Therefore, the response time of the sensing system must be short. What is more, the sensing system should be able to be applied in various cases. Taking much time to set up a sensing system but only be applied to some particular cases is useless.

In this chapter, we propose a computer vision-based sensing method to collect welding process information. An experienced welder is able to determine weld penetration based on his/her observation, our sensing system plays the role of eyes for the intelligent welding system.

3.1 GTAW welding system

All the GTAW welding experiments are done in the lab at the University of Kentucky. The welding system contains welding torch, power supply, workstation, work piece. Inert gas is also used. A robot arm UR5 is firstly used in our system, shown in figure 3.1.

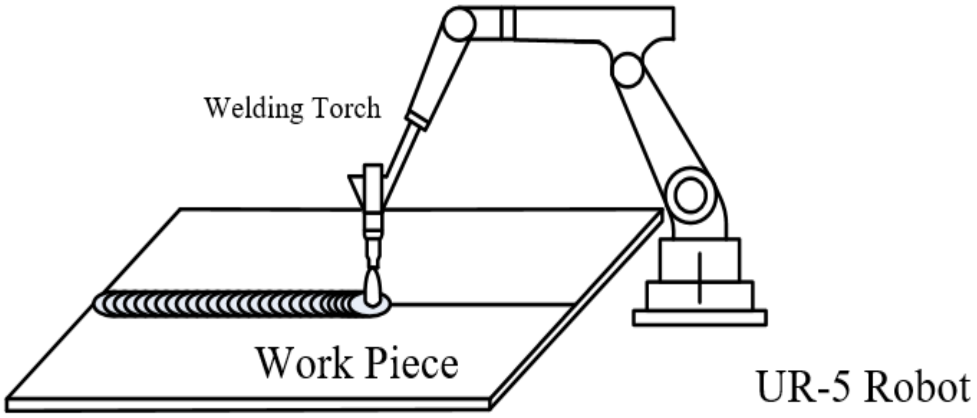


Figure 3. 1 GTAW welding system using UR5

The welding torch is attached to the robot arm and perpendicular to the work-piece. During welding process, the welding torch along the robot arm UR5 travels to continuously welding. Power supply and inert gas are not showed in this figure. UR5 has four joints and be able to carry up to 11lbs programmable robot, which makes it perfect for welding. However, when adding image acquiring system (two cameras) on that robot, the jitters of cameras are so dramatically which causes the unclearness of the captured images. The specific of the cameras will be discussed in next section. The reason of jitters is not the weight of the cameras, but the object distance. We use a high-speed camera (Point Grey GZL-CL-22C5M-C) to capture the weld pool surface image. However, the minimum distance that camera can capture clear image is 400mm. Therefore, that camera must be set 400mm away the welding torch. In addition, the relative position of high-speed camera and welding torch must be unchanged in order to capture the weld pool surface. Therefore, the high-speed camera deployed like figure 3.2. In this way, when the robot is moving, the camera shakes obviously.

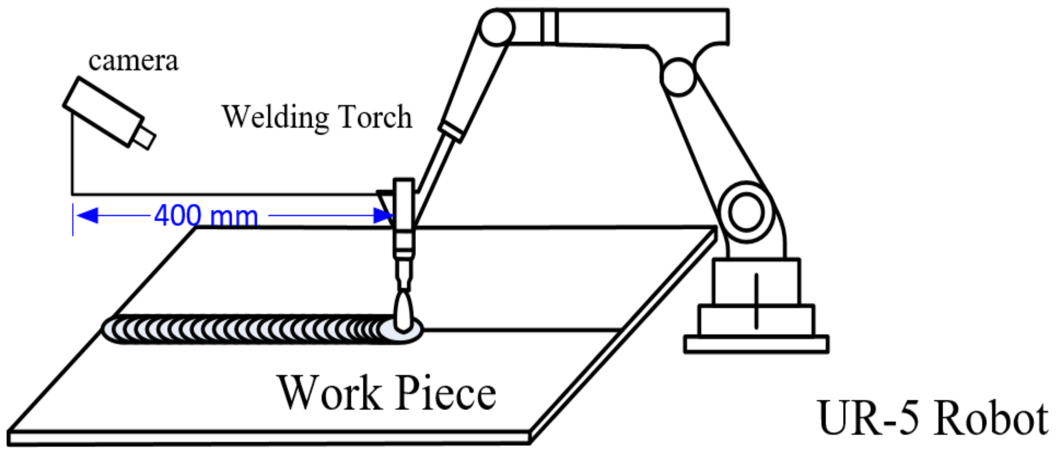


Figure 3. 2 UR5 carrying a high-speed camera

Therefore, we improved the welding system. Instead of using the programmable robot UR5, we added a motion control device (figure 3.3) onto the workstation. The motion control device is programmable and controls the workstation to move. During the welding process, the work-piece moves along the workstation as programmed. The welding torch and two cameras are set at the specific location where during welding process remains unchanged. The power supply is Miller PM200 DC, which is able to output direct current up to 200 Ampere. Figure 3.4 shows the welding system.

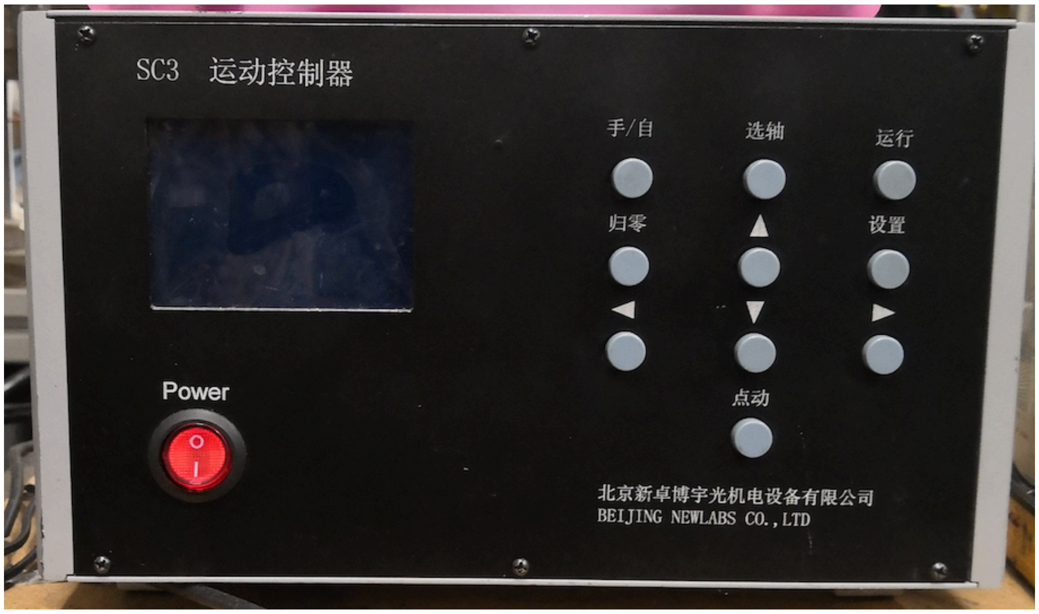


Figure 3. 3 Motion control device

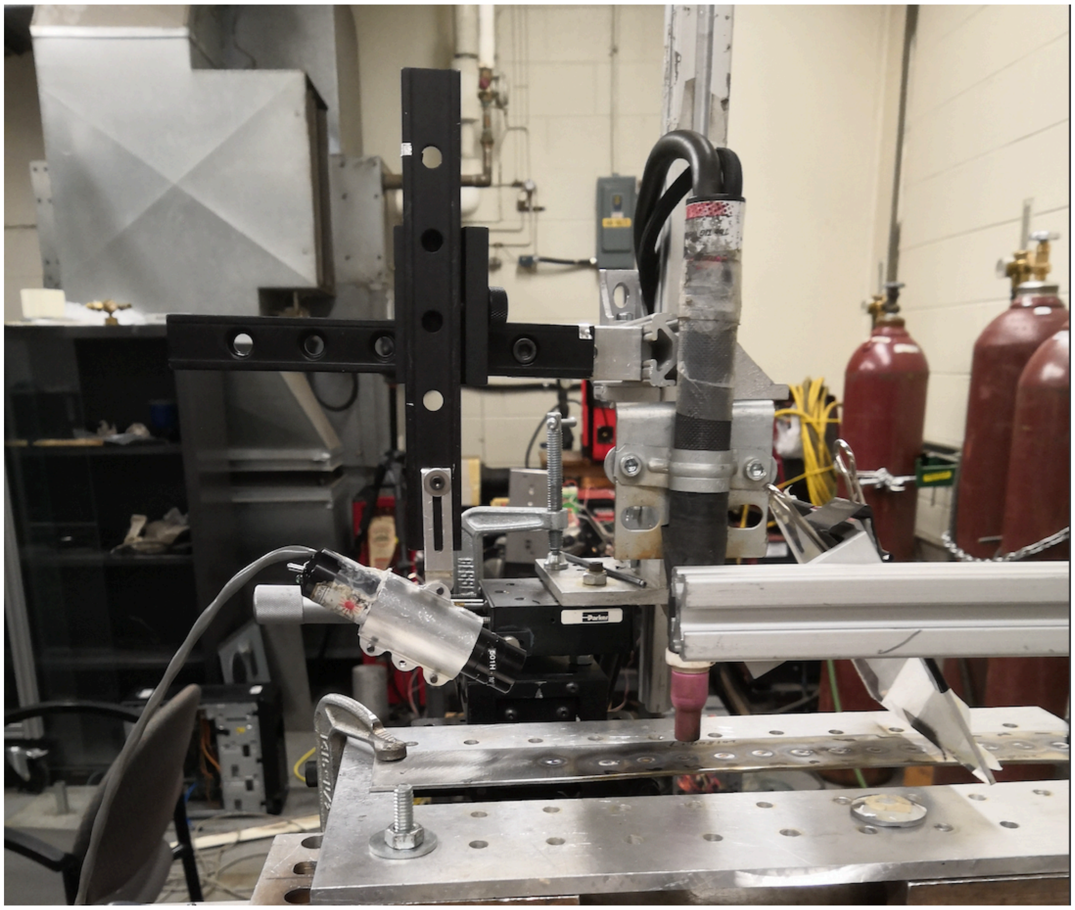


Figure 3. 4 GTAW welding system

3.2 Image acquiring system

Two cameras including one high-speed, one standard, a laser with optical head and an image screen forms the image acquiring system. As discussed, the sensing system is designed to capture the information of welding process and the weld pool surface contains enough information about the penetration status. To collect data for the convolutional neural network, we use two cameras to capture images about welding process: the images from high-speed camera are used for data the images from the standard camera are used for label. To obviously show the changes of weld pool, we project a 19 by 19 dot-matrix structured laser pattern onto the weld pool surface during welding. The wavelength of that laser is 650nm, therefore, a camera equipped with a 650nm center-wavelength band-pass optical filter catches the entire laser pattern without disturbance of other light source. A screen is placed on the path of the reflected laser pattern. Instead of directly capture the weld pool surface, the camera captures the reflected pattern on the screen. In this way, further reduces the strong arc light disturbance. The high-speed camera we use in our experiments is Point Grey GZL-CL-22C5M-C. To accurately track the welding process, the frame rate is set to be maximum: 1000fps. This high-speed camera is equipped with a 650 nm center-wavelength band-pass optical filter, precisely capturing the reflected laser pattern on the screen. Another camera (Point Grey FL-3-FW-0251C) is used to capture the backside of the weld pool. Unlike the weld pool surface, the backside of the weld pool changes less significantly, so that camera is set 30fps. The whole system is shown in 3.6.

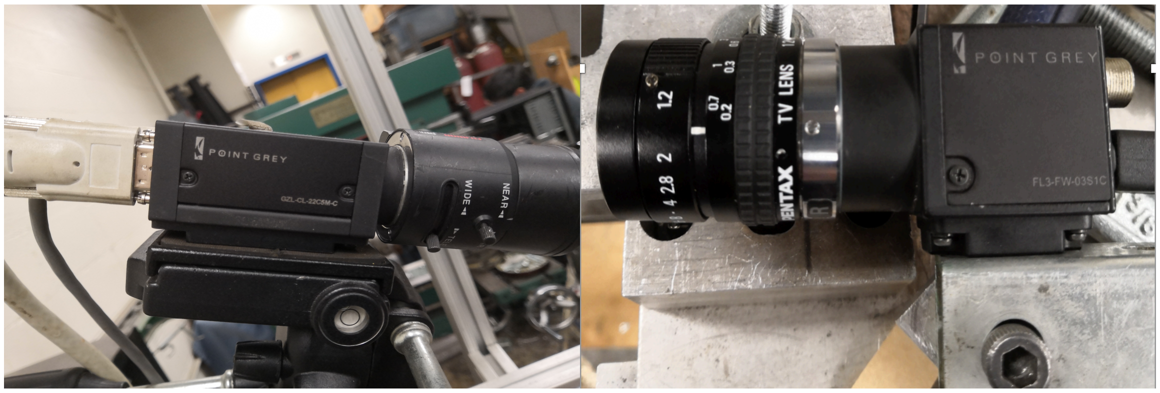


Figure 3. 5 High-speed camera (left) and standard camera (right)

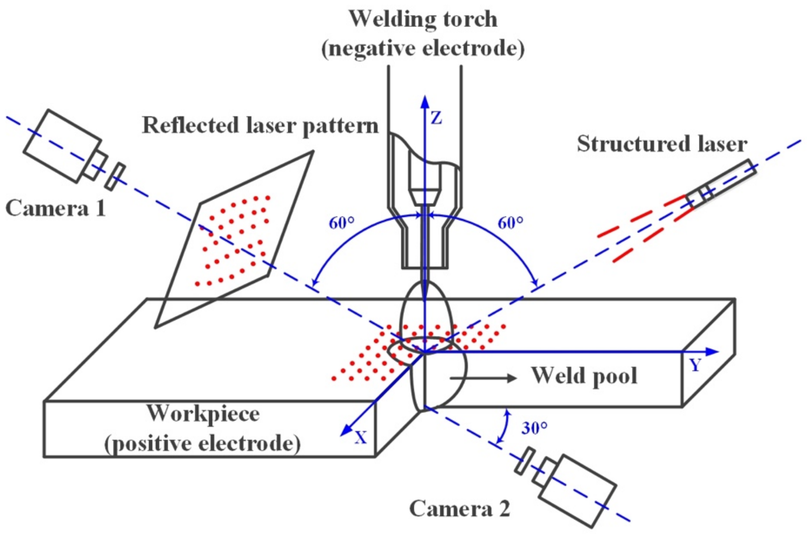


Figure 3. 6 GTAW welding system with sensing system

Camera 1 refers to the high-speed camera, camera 2 refers to the standard camera. The structured laser is placed 50mm away from the welding torch with 30 degree by horizontal. The screen is placed 50mm away from the welding torch with same degree as structured laser. The high-speed camera pointing out the screen capturing the reflected laser pattern while the standard camera points on the backside of the weld pool during welding.

3.3 Observation results

During the whole experiments, the material of the work-piece is 0.125-inch thickness 304 stainless steel. The welding current is pulsed with 60 Ampere as peak current and 20 Ampere as base current. Every cycle, use peak current weld 47 milliseconds and base current for 3 milliseconds, shown in figure 3.7.

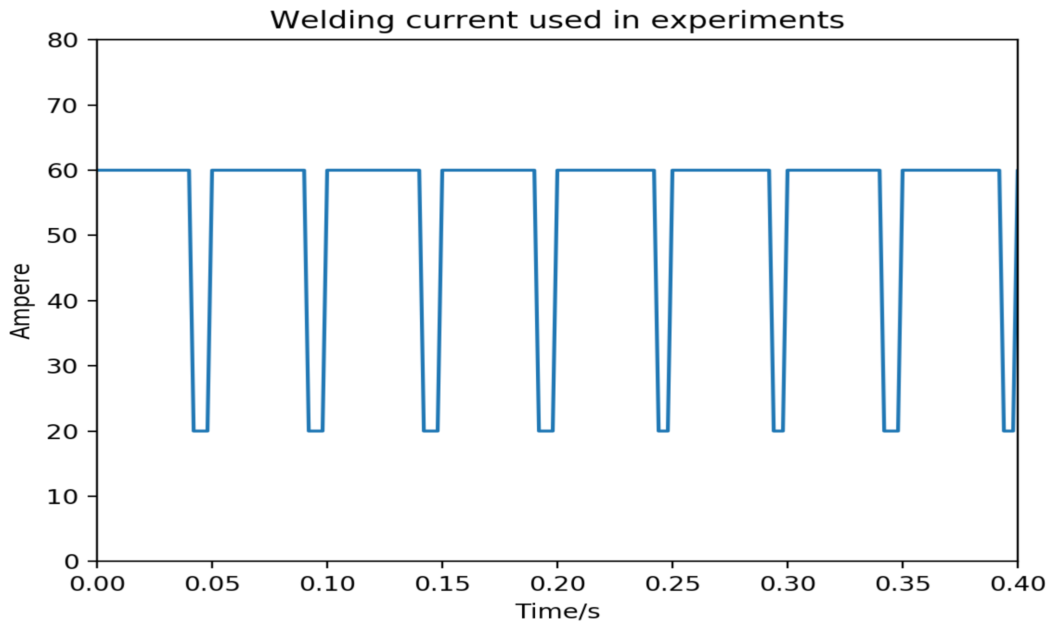


Figure 3. 7 Welding current used in experiments

Compared with continuous one level direct current, the pulsed welding current uses less energy, improves mechanical properties[27]. What is more, during the 3ms base current, the arc light is dramatically weak, the captured images are clearer. The capture speed for high-speed camera is 1000 fps and 30 fps for standard camera. Instead of tracking all the time, both two cameras taking images during the base current. Therefore, under base current period, the high-speed camera captures three images while the standard capture one image at the same time. During the peak current, the strong arc light makes the captured image hard to distinguish, and welding process is a dynamic process, there is no need to track every 1ms. The typical captured images are shown in figure 3.8.

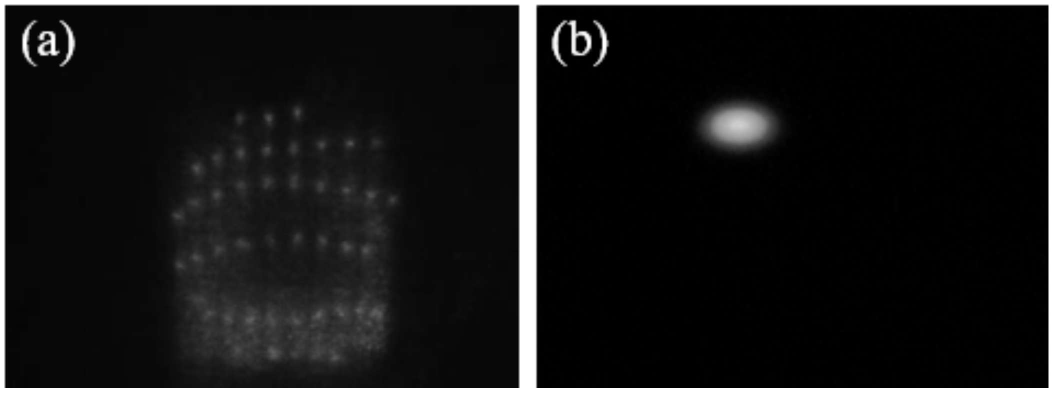


Figure 3. 8 Typical captured images: (a) high-speed camera (b) standard camera

In figure 3.8 (a) is captured by the high-speed camera. As discussed, a 19 by 19 dot-matrix laser pattern is projected onto the weld pool surface, which generates the reflected laser pattern. Based on reflection rule, the dots close to the laser generator will on the top of the reflected image, but this makes no sense, all the information about the weld pool surface have been included in that reflected image. (b) is captured by camera 2 in figure 3.5. At each beginning of the base current (20 Ampere), this camera captures one image as corresponding image of the weld pool surface at that time.

Therefore, the GTAW welding process sensing system has been established, it welds, collects data automatically. In the next chapter, we will discuss the methods used to process these images.

Chapter 4 Data pre-processing

The sensing system endows the vision ability for the welding system. However, for an intelligent welding system, only watching the welding process is not enough. As discussed, the convolutional neural network will endow the learning ability for the welding system. Unfortunately, the captured images by these two cameras are not enough as dataset to train the convolutional neural network. Therefore, in this chapter, we will discuss the data pre-processing methods.

4.1 Reflected laser pattern image processing

In our experiments, the welding process is divided into six stages. Figure 4.1 shows the typical images of these stages.

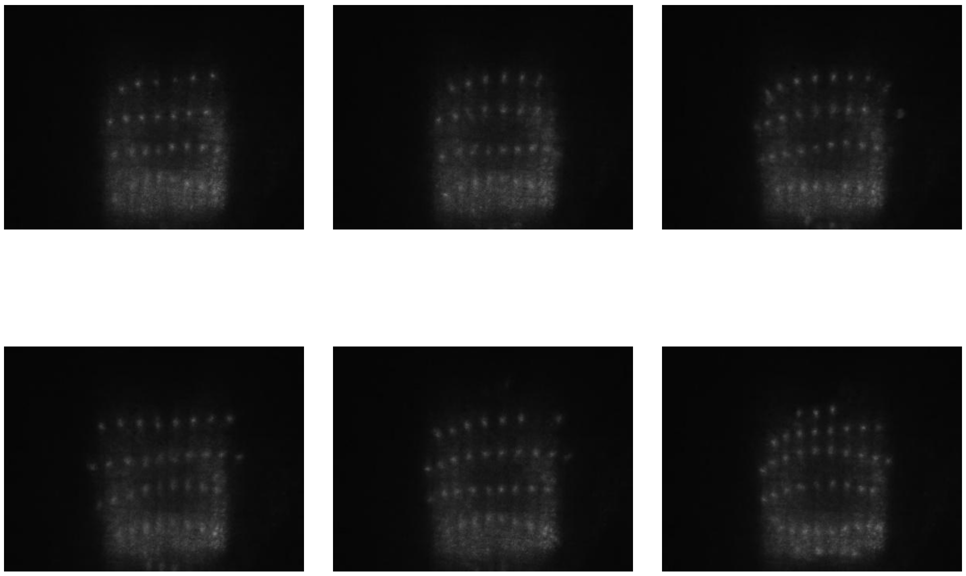


Figure 4. 1 Weld pool surface images under six stages

An experienced welder is able to determine penetration status only by observing the weld pool surface. Therefore, these images contain enough information, and be as training data for convolutional neural network. Nevertheless, for hardware side, we need to do sampling process. The images from high-speed camera are 384 pixels in width and 288 pixels in height. Even we use the best graphic card at that time GTX 1080 with 8 GB memory, the image size is too big. Therefore, the image size must be reduced.

To avoid image distortion, the width-height ratio is kept with 48 pixels in width and 36 pixels in height. The method we use to resize is the bilinear interpolation. Bilinear interpolation is a widely used sampling method in image processing area. It is simple, running fast and achieves good performance. The theory is the same as linear interpolation. Figure 4.2 shows the process of bilinear interpolation.

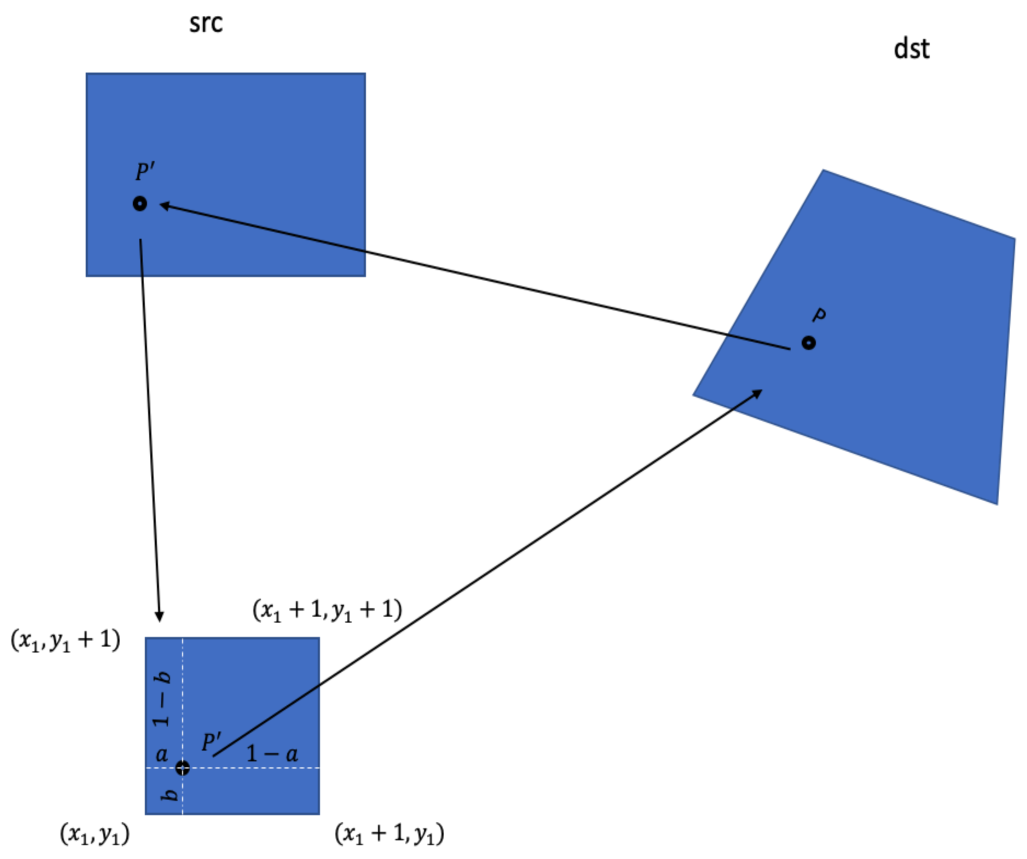


Figure 4. 2 A diagram shows the steps of bilinear interpolation

As shown in the figure 4.2, the pixel $P(x, y)$ in the destination image is what we want. Firstly, mapping $P(x, y)$ back to the source image:$P'(x, y)$. Then, use the nearby four pixels $((x_1, y_1)$,$(x_1 + 1, y_1), (x_1, y_1 + 1), (x_1 + 1, y_1 + 1))$ to represent the$P'(x, y)$. The bilinear interpolation is the linear interpolation works in two directions. Figure 4.3 shows the detail how the bilinear interpolation works.

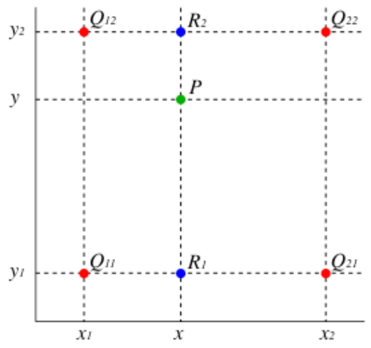


Figure 4. 3 The theory of bilinear interpolation[28]

For the point $P(x, y)$ four nearby points are: $Q_{11}, Q_{21}, Q_{22}, and\ Q_{12}$. Firstly in x direction, do the linear interpolation. We use a function f to represent the pixel value of position(x, y).

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} * f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} * f(Q_{21}) \tag{4.1}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} * f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} * f(Q_{22}) \tag{4.2}$$

Where$R_1 = (x, y_1)$, $R_2 = (x, y_2)$.

Then, in y direction, do the linear interpolation.

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} * f(R_1) + \frac{y - y_1}{y_2 - y_1} * f(R_2) \tag{4.3}$$

Therefore, we got $f(P)$:

$$f(P) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)} * (x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)} * (x - x_1)(y_2 - y) + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)} * (x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)} * (x - x_1)(y - y_1)$$

Equation 4.4: Four nearby points represents one particular point

The same results if we linear interpolate y direction first. In this way, the 384-pixel width, 288-pixel height images are resized to 48-pixel width, 36-pixel height, further used as the data for training a convolutional neural network.

We use bilinear interpolation to do the down-sampling operation. It is unavoidable that some details in the source image are lost, but as we say, an experienced welder is able to determine penetration status based on his/her observation. Human's eyes are far more advanced than any cameras in market, but when a welder wear the protection helmet, he/she cannot see much clear images of the weld pool surface under the strong arc light. Therefore, the resized smaller images contain enough information that correlated with weld penetration status. Following training convolutional neural networks proves that is right.

Besides bilinear interpolation, many other interpolations have been applied in image processing area. For example, unlike bilinear interpolation nearest interpolation chooses the nearest point pixel value as the destination value, while bicubic interpolation takes the weight sum of nearby sixteen points as the destination value. The bilinear interpolation balances the running time and the accuracy, which fits most for our research.

4.2 Backside image processing

As discussed, the welding process is divide into six stages; figure 4.4 shows the typical images of these stages:

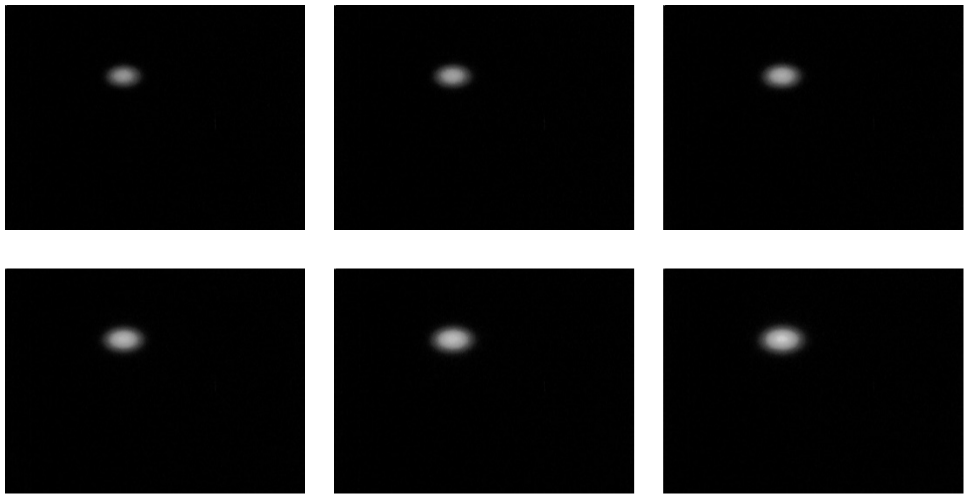


Figure 4. 4 Backside images of weld pool under six stages

In industry manufacturing, partial penetration and full penetration status are desired. The degree of welding penetration states are usually characterized as the width b, shown in figure 4.5[24, 25].

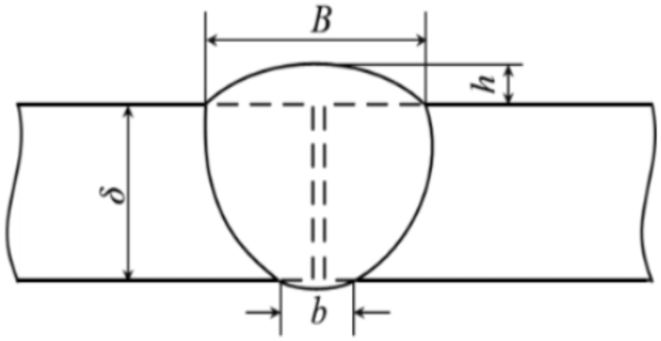


Figure 4. 5 Backside weld pool width b

For partial penetration, the width b equals to zero, non-zero for full penetration. However, for irregular shape, it is not easy to find out the length. For the image we captured from the backside of weld pool, it is an oval shape, see figure 4.6.

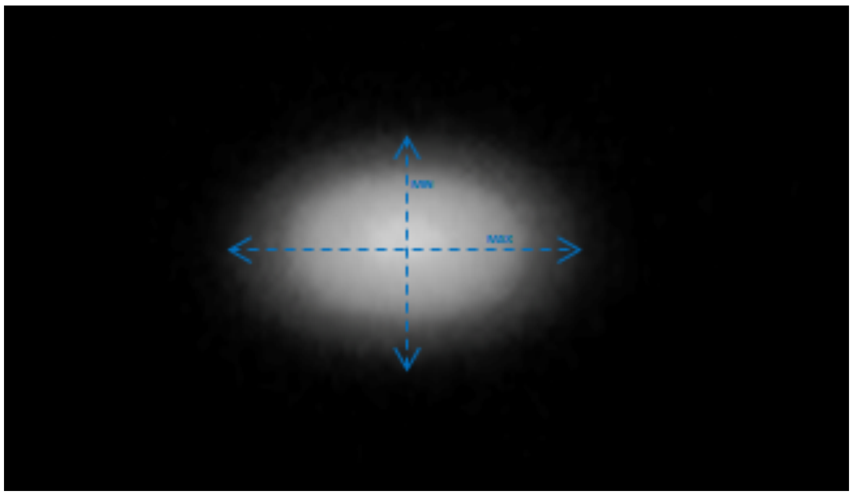


Figure 4. 6 Calculate width for an irregular shape

Traditional way is to measure the maximum length and the minimum length of this irregular shape. The width can roughly be the average of maximum and minimum. However, this process requires additional time to find the maximum and minimum length, which is not good for real-time control of welding process. Therefore, we proposed to use the area to evaluate the penetration status of the weld pool. The process is shown in figure 4.7.

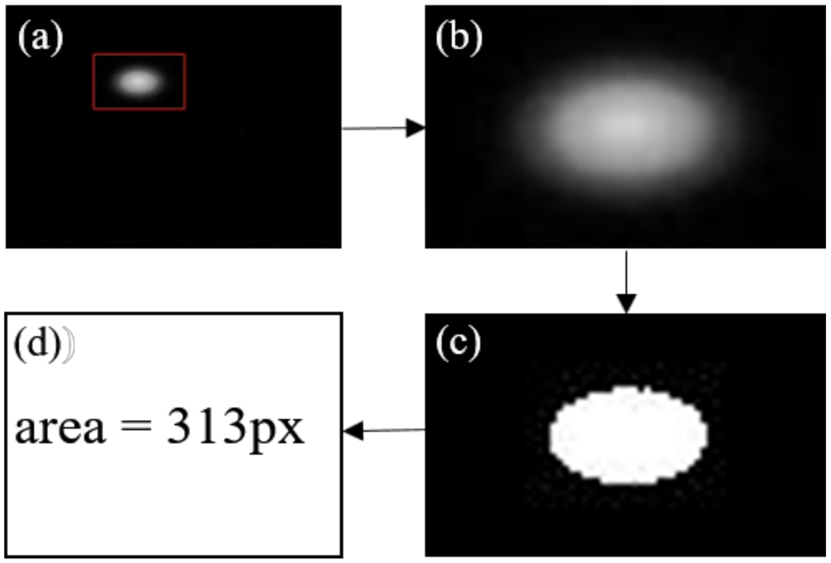


Figure 4. 7 Weld penetration status identification process

Firstly, the region of interest (ROI) is selected around the light area which reflecting the welding penetration status. Binary operation with threshold 110 is performed to filter out light pixels that comes from the lighting, or unmelted base metal. The pixels that larger than 110 is considered as the welded bead. We welded over 300 times, the lighting conditions and cameras' settings are remaining the same. Essential experiments have been done, like weld for 1 seconds, 1.5 seconds, 2 seconds, 2.5 seconds, to set the threshold to 110. When the threshold is too small, even the weld joint is not penetrated at all, there still light some dots on figure 4.7(c) due to lighting, or the reflection of the metal. While when the threshold is too big, the beginning of the welding process is dismissed. In addition, the bigger threshold causes the area small, which makes us hard to distinguish the partial penetration and full penetration. Finally, according to the accumulated number of pixels kept, we got the area. Based on the area, different weld penetration status are identified.

Table 1 Weld penetration status with labels

Area (pixels)	Label
650-950	0
950-1350	1
1350-1650	2
1650-1950	3
1950-2250	4
2250-2500	5

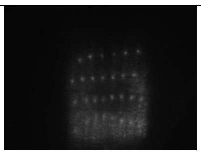
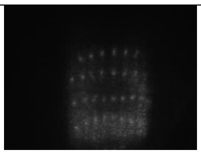
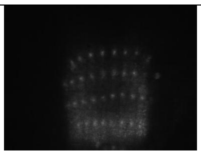
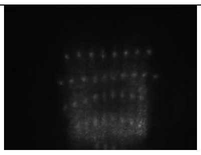
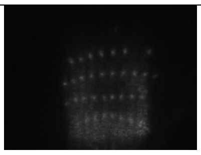
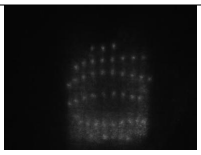
The beginning of the welding process is not considered in our research, due to partial and full penetration is what we want. For welding process physical meaning, we can roughly say the label 0 through 2 stands for partial penetration and label 3 through 5 stands for full penetration.

The reason we set six labels is that we want to more precisely control the welding process. Even the weld joint is under partial or full penetration, the degree of weld pool is different. For example, some critical parts needs full penetration, and then we weld until label 5. For some less critical parts, partial penetration label 3 works. Even label 0 works for saving welding time and consuming less power. Another reason we set six labels is for the voting mechanism to improving the prediction accuracy. The voting method will discuss in the chapter 8.

4.3 Summary

In this chapter, we discuss the methods we use to process the images captured using two cameras. The raw images cannot be directly sent to the convolutional neural networks to do the training mainly because these images have not been processed by human, or we can say have not been added the human's intelligence. Chapter 3 endows the welding system vision ability. Chapter 5 through 7 endows learning ability to the welding. In this chapter, we transforming the human's intelligence into a format that a machine understands. We divided the welding process into six stages, and based on the experiments' results we determine the penetration status and give the different stages different labels. The labeling method that is more exact is to find an experienced welder. When he/she welded, he/she identified the welding process based on his/her experience and his/her intelligence. Unfortunately, with so many experiments to do it is extremely hard to find an experienced welder to do that. Therefore, after data pre-processing step, we have the data and corresponding labels to train a convolutional neural network. Table 2 summaries the data and label currently we have.

Table 2 Data with corresponding labels

Weld pool surface	Weld pool backside	Label	Size
		0	457
		1	495
		2	540
		3	570
		4	626
		5	862

Chapter 5 Convolutional neural network

5.1 Introduction

The convolutional neural networks are designed to mimic the animal's vision system. Track back to 1968[12], a study of monkey's visual cortex shows that different lateral geniculate nucleus (LGNs) are responsible to different stimulations, shown in figure 5.1.

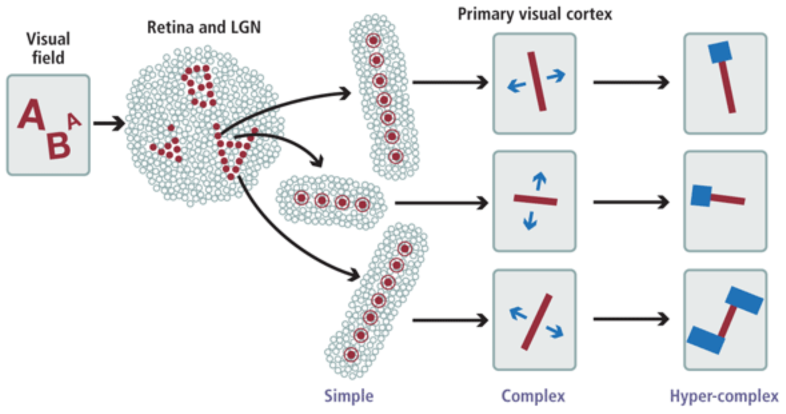


Figure 5. 1 A simplified visual cortex system[29]

The retina and the very first LGNs responsible to the light dots, then simple cells being activated encountered with edges or other stimulation. Complex cells and hyper-complex cells further processing. In 1981[30, 31], Fukushima proposed the term "neocognitron" and firstly create a network to represent the human's vision cortex system.

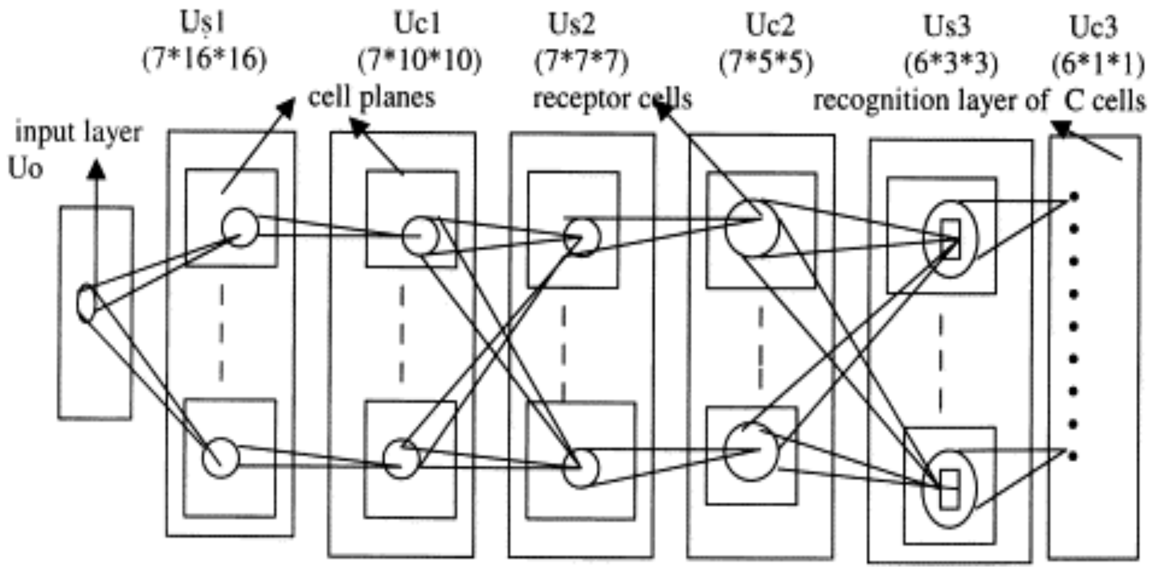


Figure 5. 2 Fukushima's network[31]

This network can be seen as the prototype of current convolutional neural networks (CNNs). In his network, two cells are included: simple cells and complex cells. For simple cells, they receive one plane of the previous step, while for complex cells; they receive multiple plans from previous step. Therefore, as network going deeper, the reception field keeps enlarging. Many ideas in that network like systematic filter, ReLU activation function, average pooling and sparse connection are still widely used in modern CNNs. However, the weights and bias in this network cannot be changed, and it is based on Winner Take All (WTA) unsupervised learning algorithm. The practical use of that network is limited.

The breakthrough occurs on the 1985, DE Rumelhart, GE Hinton et al.[32, 33] proposed the Back Propagation (BP) algorithm. The convolutional operation is redefined by using weights sharing method. The parameters in a network reduced significantly, making training a network applicable. Based on BP, LeCun et al.[13] proposed the first modern CNN. The architecture is shown in figure 5.3.

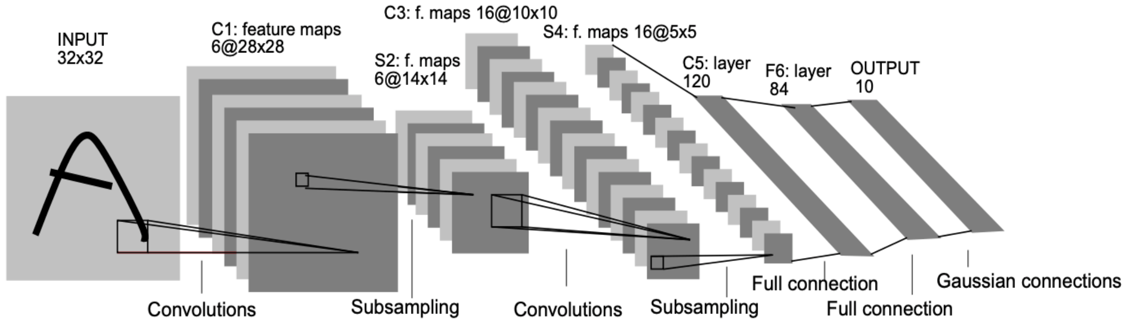


Figure 5. 3 The architecture of LeNet-5[13]

LeNet-5 contains seven layers, with two convolutional layer, two pooling layers, two fully connection layers and a Euclidean radial basis function (RBF) layer as the final output layer. The input is a hand-written digit number 0 through 9. The first layer is convolutional layer C1 with six neurons, which generates six feature maps. The second layer is subsampling layer S2. This CNN uses max-pooling method. The third layer C3 is also a convolutional layer with 16 neurons following a max-pooling layer S4 as well. Partial connections are between the C1 S2 and C3 S4 to decrease the calculation burden. The last layers are fully connected layers and Euclidean radial basis function (RBF) layer that outputs the predicted number. LeNet-5 achieves 0.95% test error on MNIST dataset and has been successfully commercial used.

However, despite its impressive performance on MNIST dataset, the LeNet-5 did not get much attention. The computation time is too much on that age and what is worse; the support vector machine (SVM) achieves close or even better results.

Until 2012, A.Krizhevsky et al.[7] won the ILSVRC-2012 competition with 15.3% top 5 test error, more than ten percentage than second, the CNNs becomes a hot topic again. The architecture of AlexNet is shown in figure 5.4.

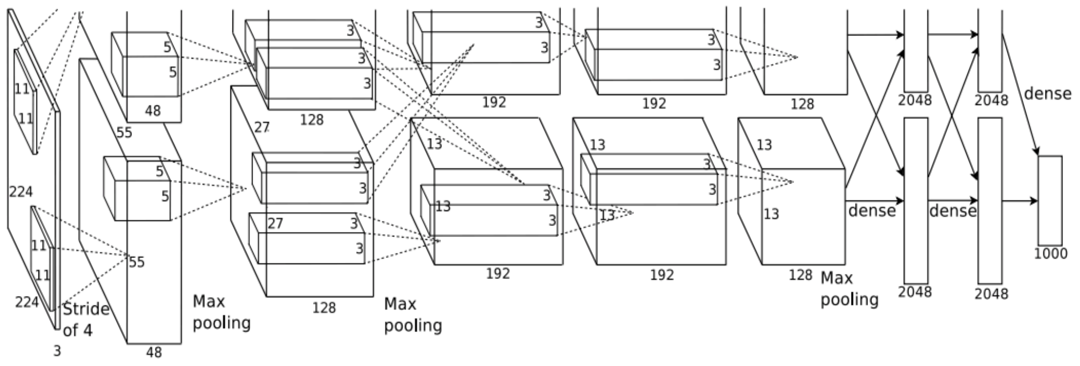


Figure 5. 4 The architecture of AlexNet[7]

The CNNs are getting deeper resulting much more parameters in a CNN; therefore, Dropout has been proposed to control the overfitting. Data augmentation is used to enlarge the dataset to give the robustness for the CNN. Re-using ReLU activation function is due to the hardness of converging when using traditional activation functions like Tanh, Sigmoid. Another boost is the hardware, especially the GPU accelerating running speed.

Inspired by the AlexNet, deeper CNNs have been proposed to improve the performance of CNNs such as R-CNN[9, 34, 35], ZF Net[36], VGGNet[37], GAN[38], GoogLeNet[8] etc. ResNet[11] achieves top 5 error 3.57%, better than human being's 5.1% error rate. So far, convolutional neural networks (CNNs) have become the prevalent method in computer vision area. In the next part, we will discuss the basic components that construct a convolutional neural network.

5.2 Basic components

Recently convolutional neural networks (CNNs) have become deeper and deeper, the ResNet has reached over 1,000 layers. Nevertheless, the basic components of CNNs are the same: a CNN contains multiple convolutional layers, some pooling layers, fully connected layers, and regression layers. Some CNNs have other special layers but not all the CNNs have.

5.2.1 Convolutional layers

Convolution layers are designed to process gird like topology 2D data like images. Multiple small patches are in the convolutional layers which we call them feature maps[39]. The number of these feature maps are dependent on the number of neurons we set. Two important properties: local connections and weights sharing makes convolutional layers powerful in processing image. Firstly, we discuss the convolutional operation.

In signal and processing area, convolutional operation is widely used. By convolutional operation, the signal in time domain is transferred into frequency domain signal, which we call the Fourier Transform[40], see equation 5.1.

$$f(s) = \int_{-\infty}^{\infty} f(t)e^{-2\pi its}dt \tag{5.1}$$

Fourier Transform (FT) is a kind of the one dimension (1D) convolutional operation. For 1D data like audio, language, the FT is essential to do processing like designing low-pass filter, etc. The general format of 1D convolutional operation can be written in equation 5.2.

$$f(t) ** g(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{5.2}$$

Where $**$ denotes convolution

The two dimension (2D) convolutional operation is quite the same as 1D convolutional, a 2D filter matrix called Kernel is replacing the $g(t)$ in the equation 5.2. Like equation 5.2, the kernel firstly flip over 180 degree then multiply with the source. Equation 5.3 shows the 2D convolutional operation.

$$S^l(i,j,k) = \sum_{m,n} I^{l-1}(i+m,j+n){K_k}^l(m,n) \qquad (5.3)$$

Where S^l is the calculated value after convolutional layers where $i\,,j$ indicates the position; I^l is lth layer; ${K_k}^l$ is the kth kernel used in lth layer.

Below in figure 5.5 shows how 2D convolutional operation works.

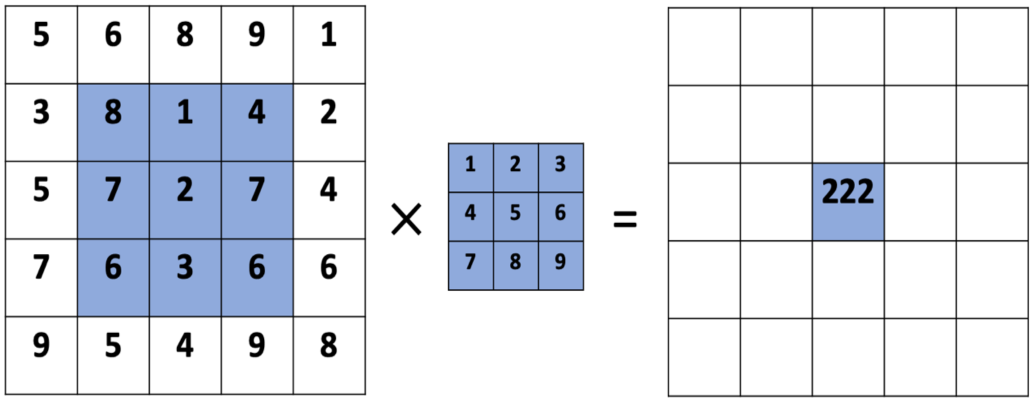


Figure 5. 5 Illustration of the 2D convolutional operation

A 3 by 3 kernel is designed to show how 2D convolutional operation works. This kernel has no actual meaning, and the weights in this kernel are set randomly. The left table represents a patch of a picture with numbers mean the pixels value on each position. The area in blue is used for convolution. Therefore, the results is $8*1+1*2+4*3+7*4+2*5+7*6+6*7+3*8+6*9=222$, the position in the output should be the same as the input.

As we can see, the 2D convolutional operation needs many multiply operation and add operation. Actually, it needs four double-loop. Therefore, the size of the kernel is usually 3 by 3 or 5 by 5, for larger kernel size, the running time becomes large. The kernel needs to be $2N + 1$ by $2N + 1$ size with the center is (N, N) and radius is N.

A problem occurs when kernel meets the boundaries. The 2D convolutional operation take the sum of the around specific position. Therefore, what if we want to calculate the left top position. Some of the nine points around that point are missing. Typically, four ways to deal with this boundary problem: zero padding, original padding, cycle padding and no padding. Zero padding puts zeros beyond the boundaries, while the original padding puts the very last number to expand the image. Cycle padding is a little complex, treat the same images are next to that image, the beyond the boundaries, put the exact same values on that image. Figure 5.6, 5.7 and 5.8 illustrate these three padding methods.

0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**5**	**6**	**8**	**9**	**1**
0	**0**	**3**	**8**	**1**	**4**	**2**
0	**0**	**5**	**7**	**2**	**7**	**4**
0	**0**	**7**	**6**	**3**	**6**	**6**
0	**0**	**9**	**5**	**4**	**9**	**8**

Figure 5. 6 Zero padding

5	**5**	**5**	**6**	**8**	**9**	**1**
5	**5**	**5**	**6**	**8**	**9**	**1**
5	**5**	**5**	**6**	**8**	**9**	**1**
3	**3**	**3**	**8**	**1**	**4**	**2**
5	**5**	**5**	**7**	**2**	**7**	**4**
7	**7**	**7**	**6**	**3**	**6**	**6**
9	**9**	**9**	**5**	**4**	**9**	**8**

Figure 5. 7 Original padding

6	**6**	**7**	**6**	**3**	**6**	**6**
9	**8**	**9**	**5**	**4**	**9**	**8**
9	**1**	**5**	**6**	**8**	**9**	**1**
4	**2**	**3**	**8**	**1**	**4**	**2**
7	**4**	**5**	**7**	**2**	**7**	**4**
6	**6**	**7**	**6**	**3**	**6**	**6**
9	**8**	**9**	**5**	**4**	**9**	**8**

Figure 5. 8 Cycle padding: on the left, top left and top there are three same images next to the original image, so the cycle padding is like this.

No padding will decrease the size of the input. Take a 3 by 3 kernel for example; the very left, right, top and bottom cannot get output values because there are no enough surrounding pixels near them. So a m by n input image, taking convolutional operation with a 3 by 3 kernel using no padding method. The size of the output image is $m-2$ by $n-2$. In general, a m by n input image convolves with a k by k kernel with no padding, the size of the output image is $m-k+1$ by $n-k+1$.

In our research, we use no padding method in convolutional operation. Recall the image we used as the input for CNN, shown in figure 5.9.

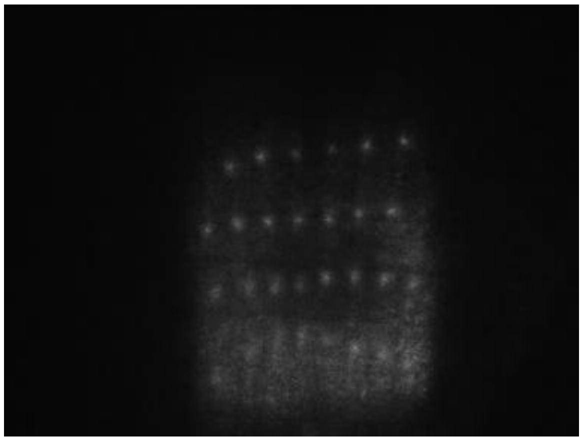


Figure 5. 9 A typical input image for CNN

The reflected laser pattern, which correlated with the weld pool surface, is located on the center of the image. All other areas are black with pixel value equals to zero. Zero padding and original padding works the same in our images, both put zeros beyond the boundaries. Only the center of the image is useful for us, the decrease part causing by no padding is just the useless part, which good for decreasing calculation burden.

Along with CNN has become the prevalent method in computer vision area. Many researchers proposed different methods to improve its performance. Atrous Convolution[11] is proposed to enlarge the reception field in deep convolutional neural networks. Unlike the traditional convolutional operation that performs in one-step, atrous convolution performs convolutional operation in every other r position. The convolution equation is shown in equation 5.4.

$$S^l(i,j,k) = \sum_{m,n} I^{l-1}(i+\mathrm{r}m, j+\mathrm{r}n) {K_k}^l(m,n) \tag{5.4}$$

Where rate parameter r corresponding to the stride length. $r = 1$ means the standard convolution. By setting the rate parameter larger than 1, the reception field enlarged. Figure 5.10 shows how atrous convolution works when $r = 2$.

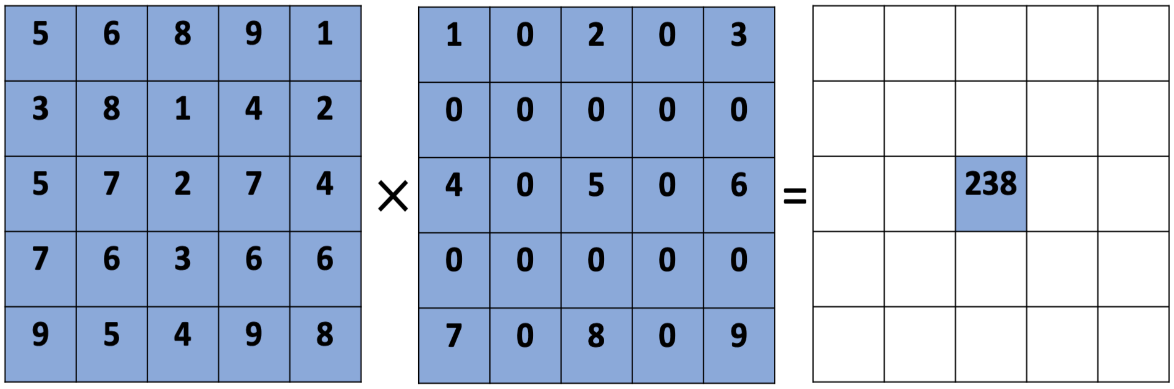
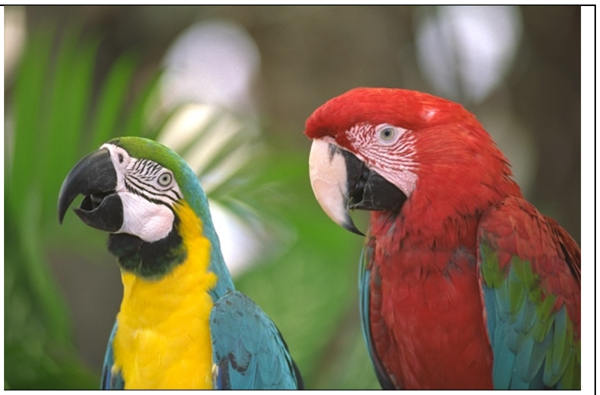
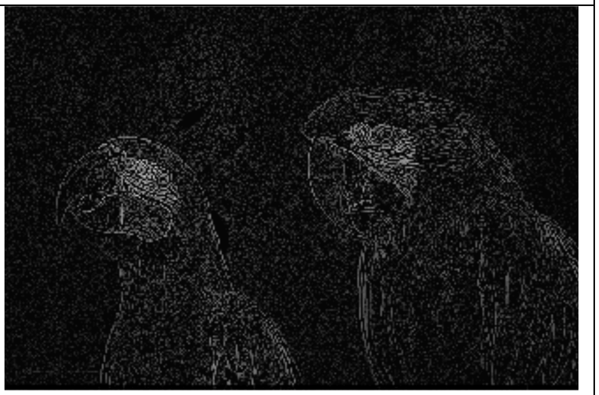
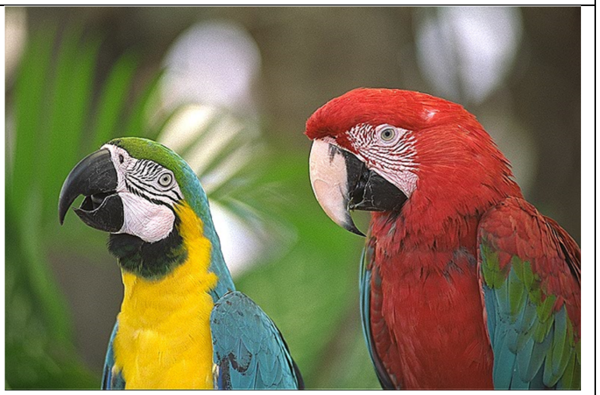


Figure 5. 10 Illustration of 2D atrous convolution when $r = 2$.

Compare with standard convolutional operation in figure 5.5. All the pixels in the input image are involved in the atrous convolution, we call it reception field enlarged. At the same time, although the kernel size is larger (5 by 5 compare 3 by 3), the parameters in the kernel remains the same. Using standard convolutional operation to get the same reception field, we need a 5 by 5 kernel with 25 parameters. Therefore, atrous convolution is useful in very deep CNNs to deal with subsampling causes smaller reception field. However, our designed CNN is relatively shallow (six layers), we use standard convolutional operation instead.

It is clear that 1D convolutional operation is used to design specific filters like low-pass, band-pass, and high-pass etc. 2D convolutional operation can even do more. Different kernels gives us different results such as blurring, sharpening, embossing, and more. Table three shows different kernels applied in the same image.

Table 3 Different kernels generate different results

Original		
Edge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Blur	$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$	

Therefore, in the CNN, each neuron generates one feature map, in other way the parameters in one feature map are the same. This is called weight sharing, which significantly decreases the parameters in the convolutional neural network. Different neurons generates different feature maps, for our research, we use 75 neurons in the first convolutional layer, which generates 75 different feature maps.

One of the primary task in training a CNN is to find the best weights for these kernels. Traditional image processing methods design these kernels by human; this requires complex modeling and calculation. For example, the researchers want to find an edge of a specific part in an image. The kernel needs to be designed to ignore all the other edges in the same image but except for that edge, which means the edge detection we seen in the table cannot be directly used. Back to our research, we want to know the penetration status based on the weld pool surface. The difficulty of that the key features are still unclear so far. We cannot say some edges or some positions that directly related with penetration status. That makes weld penetration status sensing hard and not accuracy as we expected. In CNN on the other hand, it is much better than traditional methods. We need not to know the exact key features in advance but let the CNN to extract the key features that related to penetration status itself. The training process will be discussed in next chapters.

5.2.2 Pooling layers

Another part of a CNN is the pooling layers. The pooling layers are always following the convolutional layers. The convolutional operation including multiply and add operations, these operations are linear operation. However, the combination of linear operation is also linear operation. Therefore, the pooling layer is along with the activation function to add the nonlinear factor into the CNN. Three kinds of pooling methods are used in current CNNs: mean-pooling, max-pooling and stochastic-pooling.

Mean-pooling takes the average of a small patch usually 2 by 2 as the output of that area, shown in figure 5.11.

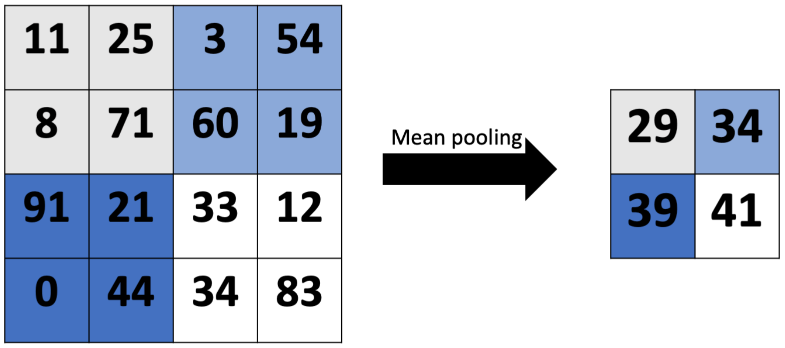


Figure 5. 11 Mean-pooling

Therefore, after mean-pooling, the image is largely smaller, from 4 by 4 to 2 by 2. The parameters in the CNN are significantly decreased. However, two main issues about mean-pooling. The output value is not always the integer. Even if we can make it to integer, that integer makes no sense. For example, for a 2 by 2 small patch, there are four colors, red, yellow, grey and blue. We wish to pick one color to represent this small 2 by 2 patch. After mean-pooling, we got an orange color, a color even not shown in that small patch. By doing mean-pooling we are not removing details but removing all but creating a new picture. Another issue is related to the first one: the performance. Using mean-pooling layers is worse than same architecture but using max-pooling layer, so current CNNs use max-pooling operation as pooling layer [7, 39, 41-45].

Max-pooling does the same way as mean-pooling but outputs the max value of that small patch, shown in figure 5.12.

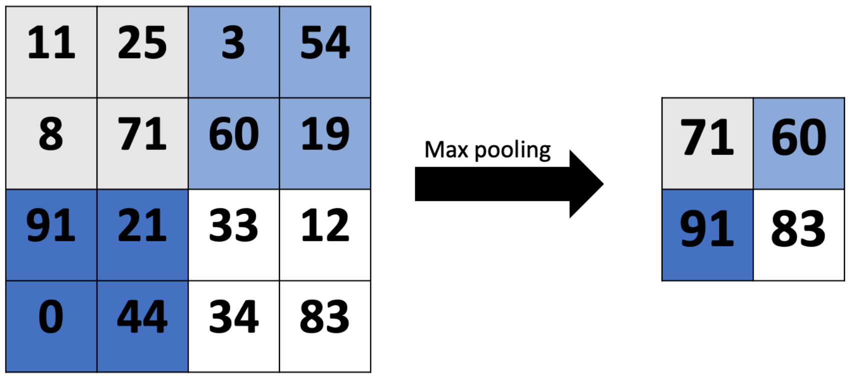


Figure 5. 12 Max-pooling

Like mean-pooling, the max-pooling operation decrease the input image size by half one side, thus significantly decreasing the number of parameters a CNN has. The less parameters further reducing the risk of overfitting. In addition, the max-pooling keeps the most significant feature in a small area like 2 by 2, removes irrelevant details. Therefore, the max-pooling is a way to reorganize the features. More importantly, the max-pooling is used to endow the CNN the ability of the invariance to the image transformation such as rotate, shift, shrink, etc. [44-46]. Therefore, the CNN is more robustness to disturbance like noise. As discussed, the performance of max-pooling is better than the mean-pooling, which makes the max-pooling method the most widely used as the pooling layer.

The max-pooling takes the maximum number as output, for a 2 by 2 area, three values are dismissed, as the area becomes larger, more values will be dismissed: doing max-pooling for 5 by 5, 24 values dismissing. To solve this problem, stochastic-pooling is proposed.

Unlike max-pooling just dismisses the smaller values, the stochastic-pooling gives each value a probability of being picking as output based on their value[47].

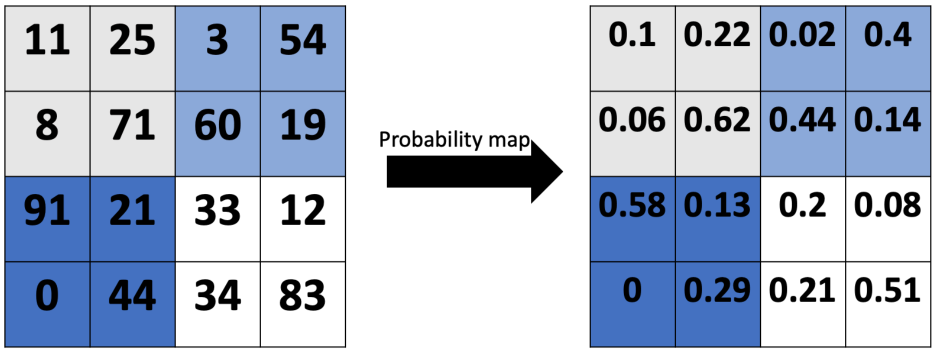


Figure 5. 13 Stochastic-pooling

In each of the pooling area, first calculate its probability using equation 5.5.

$$p_i = \frac{a_i}{\sum_{k \epsilon R} a_k} \tag{5.5}$$

Next, based their probabilities, choose one value as the output. The max-pooling can be treat as the special version, where the maximum value with 1 probability others have 0. In stochastic-pooling, the smaller values have a chance to be the output, which for some cases gives us better results[47].

The pooling methods we discuss so far have no overlapping area. There is another type of pooling called overlapping pooling. The theory is simple: the pooling area we set covers part of the nearby pooling area. By increasing the calculation burden, more details are collected. A. Krizhevsky et al[7] proposed that by using overlapping max-pooling, the top-5 error decreases 0.3%. Recently, spatial pyramid pooling[48] is proposed to transform any size of the feature maps into the same dimension. Despite these new pooling methods are proposed to improve the CNN's performance. Our research chooses the max-pooling operation as pooling layer.

5.2.3 Fully connected layers

The fully connected layers usually follow the convolutional layers and max-pooling layers. Using the convolutional layers and the max-pooling layers we have mapped the features into the specific hidden multi-dimension space, the fully connected layers are used to map the learned distributed-feature representations into the space where labels are. Fully connected layers are the basic component of standard neural networks.

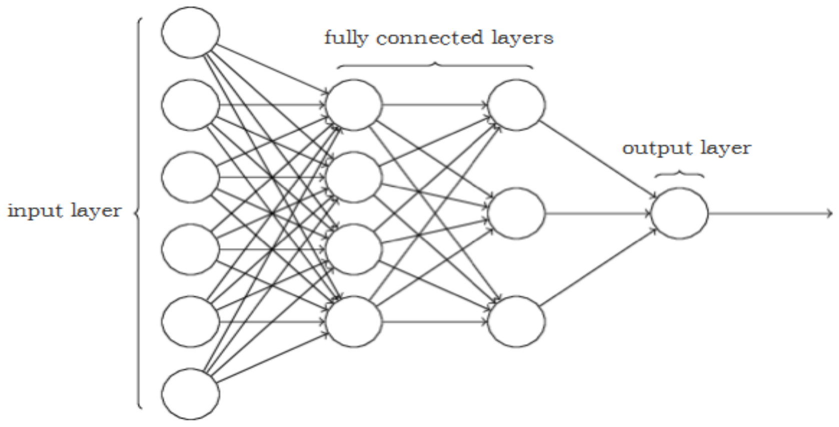


Figure 5. 14 A fully connected layer network

A typical fully connected layer network is shown in figure 5.14. The circle in the figure stands for the neuron, all the neurons in each layer are connected with all the previous and next layer's neurons, the neurons in the same layer are not connected[49, 50].

The neurons are designed to mimic human's neurons. Before we actually discuss the neurons, we need to know about the perception. In 1958, F Rosenblatt[51] proposed the perceptron.

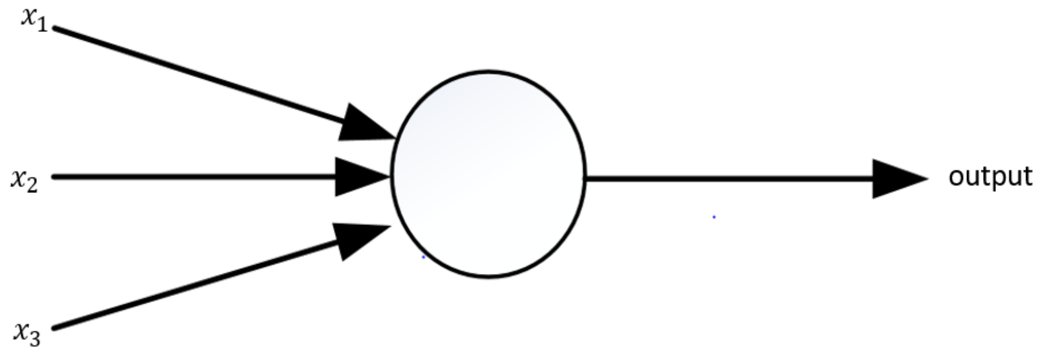


Figure 5. 15 Perceptron

The input of the perceptron is binary number 0 or 1, which corresponding to activated or not activated. Weights $(w_1, w_2, w_3, \dots w_k)$ are used to measure the importance of each input. Calculate each input with its weights then compare with θ (the threshold) gives the output. Given the input $x_1, x_2, \dots\ x_k$ the output of the neuron is

$$\begin{cases} 0 & if\ \sum_{i\in k}(w_i x_i + b_i) \le\ \theta \\ 1 & if\ \sum_{i\in k}(w_i x_i + b_i) >\ \theta \end{cases} \tag{5.6}$$

The perceptron describes how human beings make decisions[51]. However, for more complicated cases, a simple perceptron is not enough: perceptron has only two status 0 and 1, cannot reflect small changes on the input. To deal with this problem, an activation function $f(w, b)$ is used, where w means the weights and b means the bias. Therefore, the output of the neural given input $x_1, x_2, \ldots\ x_k$ is $f(w_i x_i + b_i)$. The activation function can be sigmoid[52], tanh[53], ReLU[39], etc. Among these activation functions, the tanh and ReLU are most widely used in current CNNs, shown in figure 5.16.

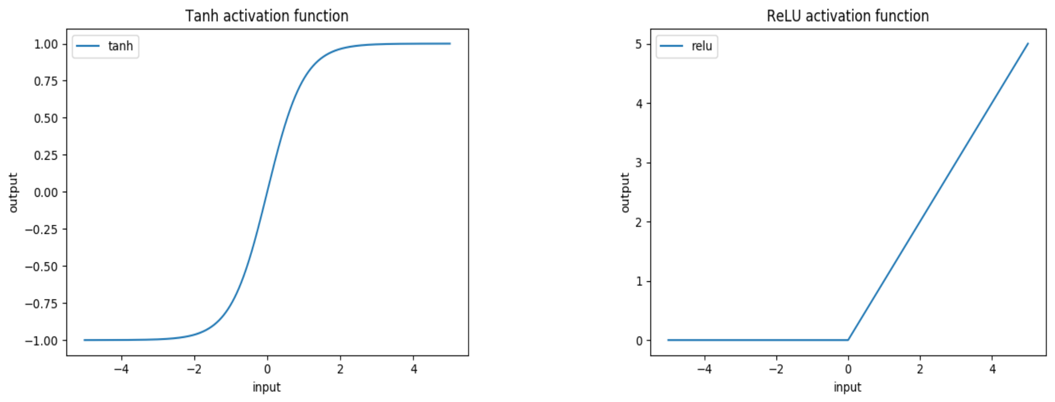


Figure 5. 16 Left tanh: $y_{tanh} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; right ReLU: $y_{relu} = \max\ (0, x)$.

Both of these functions are non-linear and both of them output 0 given 0 input. The difference is obvious, the range of tanh is $(-1\ , 1)$ while the ReLU is able to output larger than 1. The squash-like tanh results in the saturation risk, which means if the weights or bias are big enough, no matter how small the input, the output of tanh function is always 1. In addition, ReLU has the better performance [54-56] and faster converge[7]. In our research, we use ReLU as activation function, and at the same time, we use tanh activation function for compare.

The activation is not only used in fully connected layers, in convolutional layers and max-pooling layers, activation is used after the convolutional operation or max-pooling operation.

The reason for using fully connected layers is to combine all the features convolutional layer learned, and to learn non-linear combinations of these features. In the fully connected layer, the position information is dismissed, since all the features are transformed into one-dimension. The learned CNN has robustness for position variance, which means it is able to detect a cat in an image regardless the cat is on the top left or right corner. However, for segmentation task, which requires detection and position information, fully connected layers cannot be used. Another issue about fully connected layer is the parameters it has. The parameters of fully connected layers can take up to 90% of the whole CNNs, which will show in next part. Resulting longer running time and high risk of over fitting. Therefore, global average pooling (GAP) is proposed and achieves good results[10, 57, 58] such as ResNet[11], GoogLeNet[8]. However, our research goal is classification not segmentation, thus we keep fully connected layers as high reasoning method.

5.2.4 Regression layers

The final layer of the CNN are the regression layer or classification layer depending the output is continuous number or the group number. In chapter 4, we have created the data and corresponding labels. There are six labels in our research; therefore, a classification layer is the choice. In our research, softmax regression is classifier in the final layer, which is a special kind of the binary logistic regression (LR) classifier. Even we call it regression layer, it is a multiple class classifier. Therefore, we will discuss the binary logistic regression first.

The binary logistic regression uses a logistic function to make a prediction 0 or 1 for the given variable. The logistic function is Sigmoid, shown in figure 5.17.

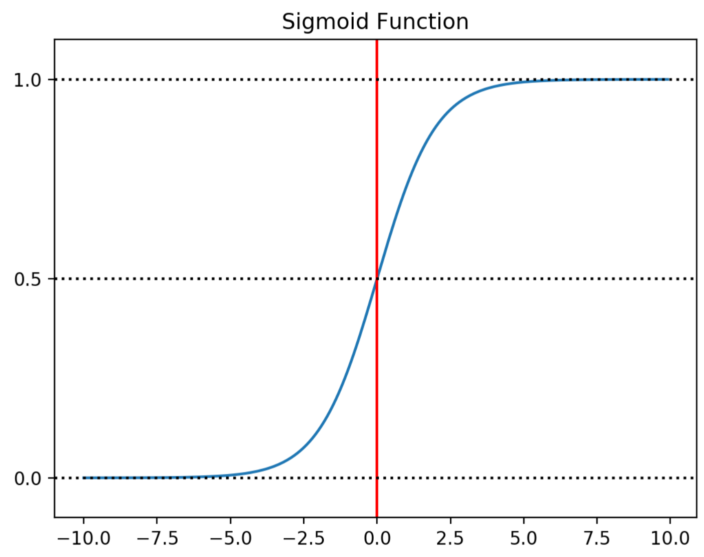


Figure 5. 17 Logistic function $\frac{1}{1+e^{-x}}$

The logistic function output is between 0 and 1, which can be treated as a probability. Therefore, the probability of 1 given a input x_i is

$$P(y_i = 1|x_i; w) = \frac{1}{1+\exp(-w*x_i)} \tag{5.7}$$

Therefore, given a dataset contains N data, the likelihood function is:

$$\prod_{i=1}^{N}[P(Y = 1|x_i; w)]^{y_i}[1 - P(Y = 1|x_i; w)]^{1-y_i} \tag{5.8}$$

Then, we get the minimum negative log likelihood as the loss function:

$$min_w \quad L = -log \prod_{i=1}^{N}[P(Y = 1|x_i; w)]^{y_i}[1 - P(Y = 1|x_i; w)]^{1-y_i} \tag{5.9}$$

$$= -\sum_{i=1}^{N}\left[y_i logP(Y = 1|x_i; w) + (1 - y_i)\log\big(1 - P(Y = 1|x_i; w)\big)\right]$$

$$= -\sum_{i=1}^{N}[y_i(w * x_i) - log(1 + exp(w * x_i))]$$

Thus this loss function is the calculate the sum, in case too much number, we take the average of that sum. The loss function is

$$L = -\sum_{i=1}^{N} [y_i(w * x_i) - \log(1 + \exp(w * x_i))] \tag{5.10}$$

Using the gradient descent (GD) algorithm, which we will discuss in chapter 7, the parameters w can be calculated. Thus, based on equation (5.7), the probability of 1 is calculated.

The binary logistic regression distinguishes two class every time. For our research, which has six classes need to be classified, five times binary logistic regression will work. However, better choice is softmax regression. Unlike logistic regression, the softmax regression is designed to deal with multiple labels classification[59]. The softmax function is shown in equation 5.11.

$$f_i(y) = \frac{\exp{(y_i)}}{\sum_k \exp{(y_i)}} \tag{5.11}$$

Where k is the number of labels.

Similarity, given input x_i and parameters w, the probability of outputs label j $(y_i = j)$ is

$$P(y_i = j|x_i; w) = \frac{\exp{(w_j * x_i)}}{\sum_{n=1}^{k} \exp{(w_n * x_i)}} \tag{5.12}$$

Therefore, like binary logistic regression, the likelihood function is

$$\prod_{i=1}^{N} \prod_{j=1}^{k} (\frac{\exp{(w_j * x_i)}}{\sum_{n=1}^{k} \exp{(w_n * x_i)}})^{1[y_i=1]} \tag{5.13}$$

The loss function can be defined like:

$$min_w \quad L(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k}[1[y_i = 1]log\frac{\exp(w_j * x_i)}{\sum_{n=1}^{k}\exp(w_n * x_i)}] \qquad (5.14)$$

Then, use the gradient descent (GD) algorithm to calculate the parameters w. Therefore, for a given input x_i, the softmax regression layer outputs all the probabilities of all the labels $P(y_i = j|x_i; w)$, then choose the label with maximum probability as the final output.

5.3 Architecture of CNN

In the last section, the basic components that construct the CNN are discussed. However, how many convolutional layers are needed, how the convolutional layers are connected to the max-pooling layer, where to put fully connected layers, these questions are still unclear. In this section, we will discuss the architecture of CNN.

5.3.1 Basic rules for setting a CNN

The convolutional neural network is designed to mimic human's visual system. However, there are billions of neurons in human being's brain. Even in current hardware, the billions of neurons are so enormous, and it is extremely hard to implement. Therefore, researchers step back to design different architectures for different cases.

It is believed that a neural network with only one hidden layer can approximate any continuous function as long as the hidden layer has enough neurons [60-62]. Therefore, the reason for going deeper is reducing the parameters networks have[59, 63]. In convolutional layers, as discussed, one feature map sharing the same parameters: weights and bias of one neuron. In addition, for max-pooling layer, the input image size is reduced by one out of fourth. Using a shallow for example one hidden to approximate a function takes exponential number of that represented by deep rectifier network[64].

Another reason for using multiple layers is that it has more generalizing ability, the learned weights and bias are smaller, the functions are more smooth[59, 63]. Using multiple layer, we can say that we divide a complex problem into several small problems, each layer corresponding one small problem. The learning process can be easier than that using shallow network. Take CNN for example, the convolutional layer collects key features, the max-pooling further reorganizing these key feature, then convolutional collect higher-level features, the following max-pooling do the same job, after several cycles, the input image are transferred into a set of features, then using hidden layer or global average pooling to do higher reasoning, outputs the final result. This process is more reasonable than using hundreds or even thousands of neurons to do the job in one shallow network. Even though some researcher questioning the deeper and deeper neural network[65], the deeper neural networks actually achieves better results. The ResNet[11] with more than 100 layers outperforms human being in some cases.

The deeper neural networks actually cause the overfitting problem, but the overfitting problem is even worse in shallow networks. Therefore, it is better to use methods like regulation, dropout[7] to control the overfitting, not use shallow network because the concern of overfitting.

Therefore, based on the complexity of the task, choosing a slightly larger and deeper neural network is much better using a smaller neural network.

5.3.2 Architecture details

Our task is to identify the penetration status using CNN. There are six labels corresponding different penetration status. Before setting our architecture, we will discuss some classical CNNs' architectures.

In chapter 5, we have discussed two CNNs: LeNet and AlexNet. LeNet[13] has seven layers including two convolutional layers, two max-pooling layers, two hidden layers and one Euclidean radial basis function. It achieves over 99% identification accuracy on MNIST dataset with ten labels. AlexNet[7] contains 13 layers with five convolutional layers, five max-pooling layers, three fully connected layers. It won the ILSVRC-2012 competition with 15.3% top-5 test error based on the famous ImageNet dataset, which has 1,000 labels. Later in the year of 2014, VGG has been proposed, further reduce the top-5 to 7.3% [8, 37]. VGG has 5 convolutional layers, 5 max-pooling layers, three fully connected layers with a softmax layer to output the result. The architecture is shown in figure 5.18.

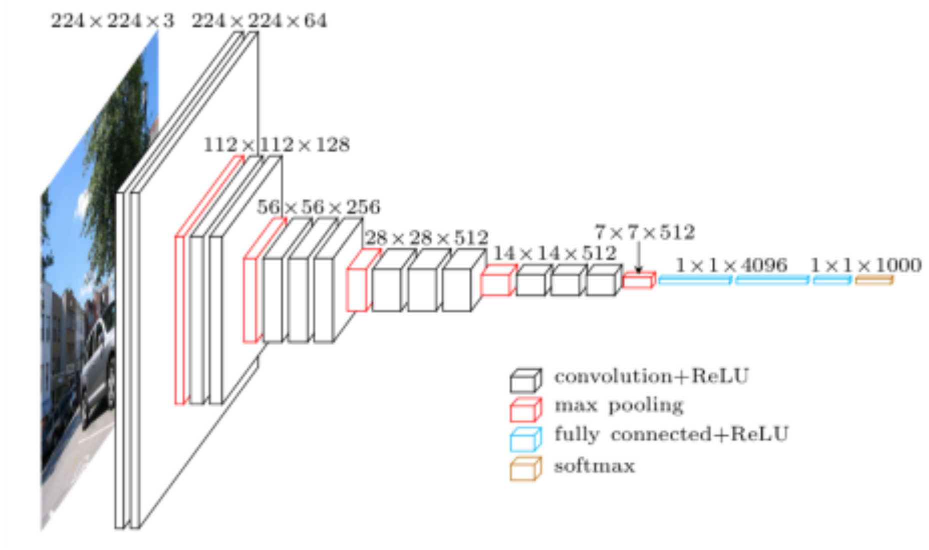


Figure 5. 18 The architecture of VGG[66]

A year later, GoogLeNet further reduce the top-5 error to 6.7% with 22 convolutional layers, 4 max-pooling layers, one average pooling layer, one fully connected layer and softmax layer to give the result. Shown in figure 5.19.

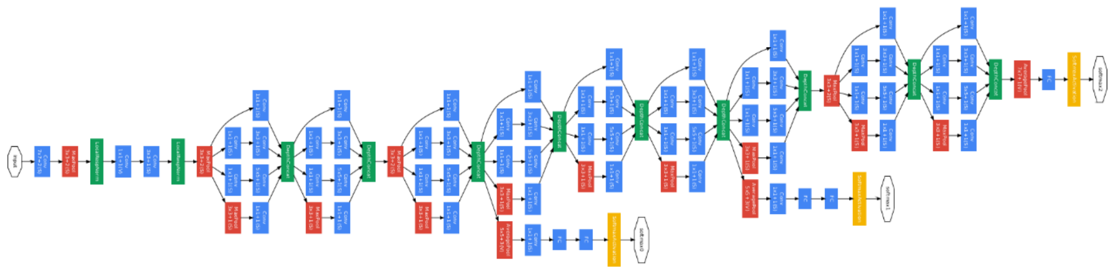


Figure 5. 19 The architecture of GoogLeNet[8]

ResNet[11] has 151 convolutional layers which has the state of art 3.57% top-5 error on the ImageNet, outperform human being's 5.1% top-5 error.

It is clear that for ImageNet with 1,000 labels the deeper network gives better results. Back to our research, comparing with these architectures, we design a six-layer convolutional neural network with two convolutional layers, two max-pooling layers, one fully connected layer and a softmax regression as logistic regression layer.

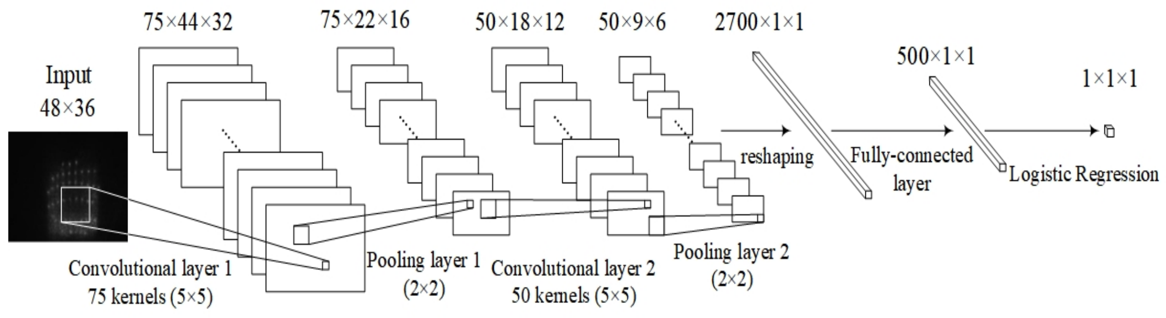


Figure 5. 20 The architecture of CNN used in our research

Input is the image we captured using high-speed camera, size is 48 width by 36 height. The first layer is convolutional layer with 75 neurons to make sure collect enough information for further use. Kernel size is 5 by 5. Convolutional operation is standard with no padding. Therefore, after convolution the size is 44 ($= 48 - 5 + 1$) by 32 ($= 36 - 5 + 1$). Each neuron generates one feature map, thus, there are 75 feature maps. Different neuron settings have been tested on the same dataset. See table 4.

Table 4 Different neuron settings performance

Neuron numbers in Conv layers: (first, second)	Validation error in the same dataset
(300,200)	22.35 %
(75,50)	22.467%
(60,40)	24.367%

Original neurons are 300 for the first convolutional layer, 200 for the second convolutional layer. It achieves the best performance, 0.117% less validation error than (75, 50) setting, and 2.017% less than that of (60, 40). However, the cost is longer running time. Our goal is real-time control welding process, which requires fast response. Therefore, for practical view, we prefer 75 neurons as first convolutional layer, 50 neurons as second convolutional layers which balancing the accuracy and the running time. Therefore, for the first convolutional layer, the parameter number that needs to be learned is 1950 ($= (5 * 5 + 1) * 75$).

Following the convolutional layer is the max-pooling layers. A 2 by 2 max-pooling operation performed, reducing the data into 22 width by 16 height. As discussed, max-pooling largely decreases the parameters and endows robustness for position variation. The parameters number that need to be learned is 150 (=$(1 + 1) * 75$).

Batch normalization is performed after the max-pooling layer. Batch normalization is used as re-distribute the input data, which can be treated as a way of pre-processing. The batch normalization will discuss in chapter 7. Rectified Linear Unit (ReLU) is the activation function, which gives 0.492% less validation error on the same dataset (20.875% validation error minus 21.367% validation error).

The second convolutional layer and the second max-pooling do the exact same process. The kernel size is 5 by 5, no padding for the convolutional operation. Max-pooling area is 2 by 2. Batch normalization has been performed, activation function is ReLU. After the second max-pooling layer, the size is further reduced to 9 width by 6 height. The parameters that need to be learned in the second convolutional layer is 93800 ($= (75 * 5 * 5 + 1) * 50$)). The second max-pooling layer need to learn 100 ($= (1 + 1) * 50$) parameters.

Fully connected layer is used to do high reasoning work. As discussed, the position information is dismissed by convert the second max-pooling output into a vector. Therefore, the input size is 2700 by 1 matrix, shown in figure 5.21.

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,2699} & w_{1,2700} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,2699} & w_{2,2700} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{499,1} & w_{499,2} & \cdots & w_{499,2699} & w_{499,2700} \\ w_{500,1} & w_{500,2} & \cdots & w_{500,2699} & w_{500,2700} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2699} \\ x_{2700} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{499} \\ b_{500} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{499} \\ y_{500} \end{bmatrix}$$

Figure 5. 21 Multiply process in fully connected layer

Therefore, the parameter number that is need to be learned is 1,350,500 ($= 2700 * 500 + 500$)), as discussed, the parameters of fully connected layer take up to 93.337% of all the parameters so far.

The final layer is a logistic layer, using a softmax regression to calculate the probability of each label's, then outputs the maximum one as the label. Softmax regression gives every sample the probabilities of each labels: $\frac{\exp{(y_0)}}{\sum_{k=0}^{5} y_k}, \frac{\exp{(y_1)}}{\sum_{k=0}^{5} y_k}, \frac{\exp{(y_2)}}{\sum_{k=0}^{5} y_k}, \frac{\exp{(y_3)}}{\sum_{k=0}^{5} y_k}, \frac{\exp{(y_4)}}{\sum_{k=0}^{5} y_k}, \frac{\exp{(y_5)}}{\sum_{k=0}^{5} y_k}$.

Outputs the label with the maximum probability. The parameter number needs to be learned is 3006 ($= 500 * 6 + 6$). Table 5 summarize number of parameters in each layer.

Table 5 Parameters in each layer

Layer	Number of parameters
Convolutional-1	(5×5+1)×75 = 1950
Pooling-1	(1+1) ×75=150
Convolutional-2	(75×5×5+1)×50 = 93800
Pooling-2	(1+1) ×50 = 100
Fully-connected	2700×500 + 500 = 1,350,500
softmax regression	500×6 + 6 = 3006
Total	1,449,506

5.4 Summary

In this chapter, we discussed the history of convolutional neural network (CNN), the basic components that construct a CNN including convolutional layer, pooling layer, fully connected layer and softmax regression layer. Classical architecture of CNNs such as LeNet, AlexNet, VGG, GoogLeNet are discussed. Then, proposed our six-layer CNN, discuss the parameters that need to be learned. Before we discuss the training method, we will discuss the data augmentation in next chapter.

Chapter 6 Data augmentation

In last chapter, we have discussed the parameters we need to learned for our six-layer CNN. A total of 1,449,506 parameters in our CNN, but the dataset size so far is only 3,550 so far, which is not enough to train the CNN. Therefore, the data augmentation is performed in our research.

6.1 Necessity for doing data augmentation

Recent CNNs take the data augmentation as a way to control the overfitting[67], such as AlexNet[7], VGG[37], GoogLeNet[8], ResNet[11]. On the other hand, training neural networks based on small dataset resulting serious overfitting, a little change like position, sizes will decrease the accuracy. Pinto et al.[68] designed a V-1 like model with limited images to train. Results shows the performance degrease when variations are added into the test set. As discussed, LeNet is training based on the MNIST dataset which contains 60,000 training samples and 10,000 test samples[13]. Caltech-256[14] contains 30,607 samples. In addition, most CNNs are based on the ImageNet[15] dataset, which contains 21,841 synsets with over 14 million samples so far. Therefore, 3,550 samples is far below the requirement to train a six-layer CNN, which contains 1,449,506 parameters. Data augmentation is needed in our research.

6.2 Data augmentation methods

Many data augmentation methods have been proposed and been proved effective in certain cases. In summary, data augmentation methods can be concluded into two ways: one is affine transformation the other one is Generative Adversarial Nets (GANs).

6.2.1 Affine transformation

The affine transformation takes the form:

$$y = w * x + b \tag{6.1}$$

Based on this equation, the processing methods includes shift, horizontal or vertical flip, rotation or reflection. Figure 6.1 summarize these transformations.

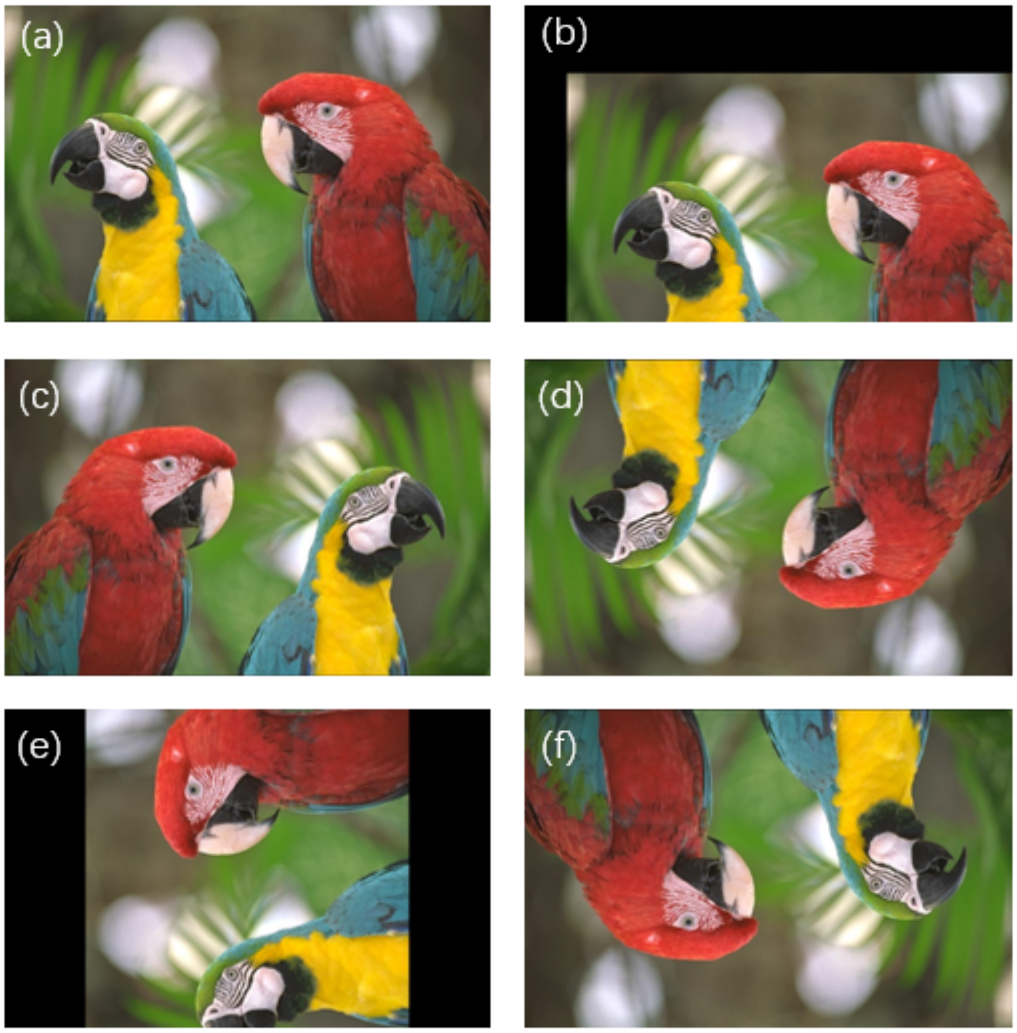


Figure 6. 1 Typical affine transformations: (a). original image (b). shift (c). horizontal flip (d). vertical flip (e). rotation- 90 degree, (f). rotation-180 degree

These methods enlarge the dataset fast and easy, but for some cases, some of them are not appropriate. For example, the trained CNN is about face detection, the 180 degree rotation cannot been used since no need to recognize a face in that direction. The next common methods is scale jittering[11, 37]. The crop size is fixed n by n, for example VGG use 224 by 224. The input image is isotopically scaled. The shorter side (width or height) is chosen as the training scale S. S is randomly chosen in the range of $(min, \max)$, where

min must be greater or equal to the cropping size n. Finally randomly crop a n by n area from the scaled image. Figure 6.2 shows the whole process.

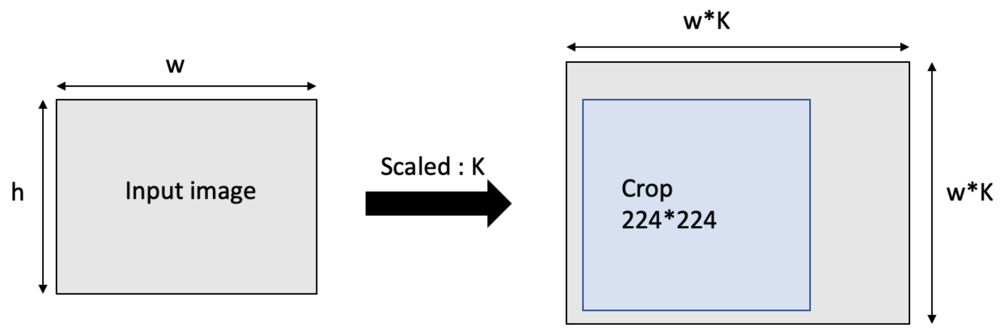


Figure 6. 2 The process of scale jittering

Another method is scale aspect ratio augmentation[8]. Unlike the scale jittering method which keeps the input width-height ratio. The ratio in scale aspect ratio changes in a range of $[\frac{3}{4}, \frac{4}{3}]$ to generate more images.

For colored images, color jittering and PCA jittering can also be applied to do data augmentation. All the images in our research is grey mode, these methods are inappropriate and we will not discuss them.

6.2.2 Generative Adversarial Nets (GANs)

Another widely used way of data augmentation is Generative Adversarial Nets (GANs). Unlike the affine transformations, GANs[38, 69] actually generates new images. It achieves impressive results in image translating[58, 70], representation learning[58, 71], etc. A Discriminator[38] (D) is proposed to estimate the reality of the generated images that generated by the generator (G), the loss function is defined as adversarial loss which forces the generated images are different from the real images. Training process is based on back propagation (BP), but always update one (G or D) at the same keep the other unchanged. Figure 6.3 to 6.5 shows some images generated from the GANs.

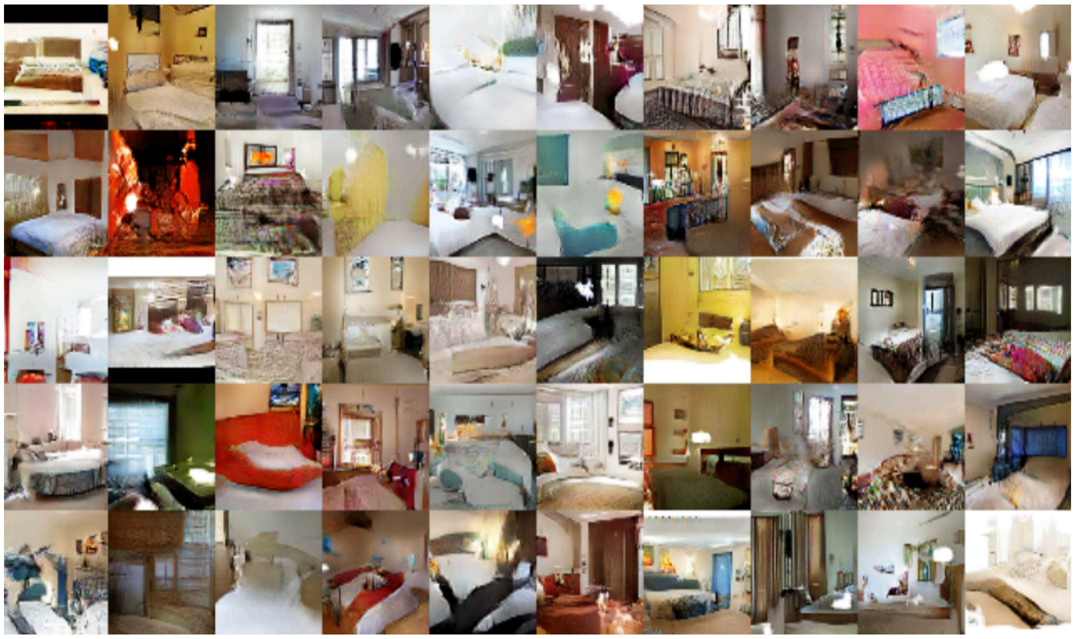


Figure 6. 3 Generated different bedrooms[58]

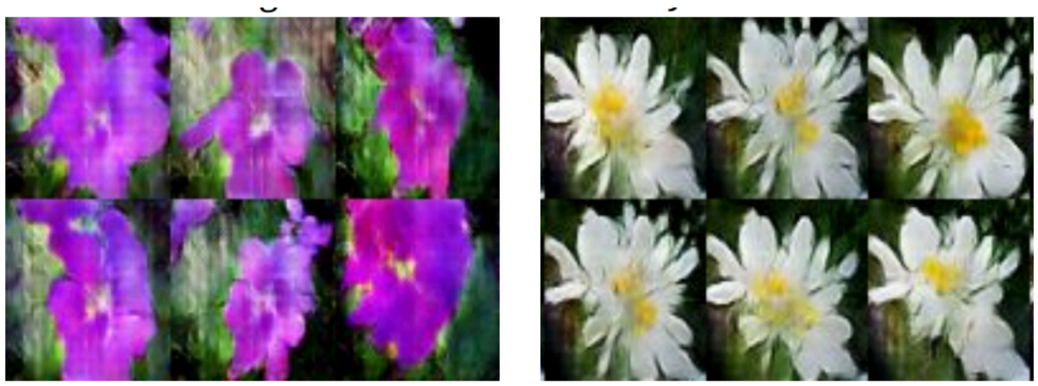


Figure 6. 4 Generated different flowers[72]

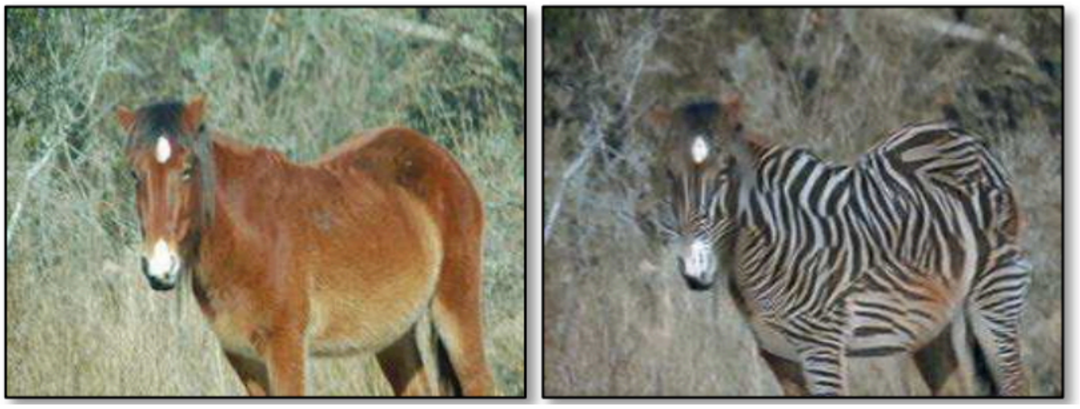


Figure 6. 5 Generate zebra from horse[73]

6.3 Summary

The affine transformation and generative adversarial nets (GANs) are the two methods doing data augmentation, both of them achieve impressive results. In our research, an experienced welder is able to determine the weld penetration status form different position and under different view. Therefore, affine methods including shift, rotation and resizing are used as the data augmentation method. To ensure the entire reflected pattern is kept after the affine transformation, we firstly select the region of interest (ROI). Experiment results shows the shift operation results in part of the ROI missing. As discussed, all the images are in grey scale, the color augmentation and PCA jittering are not useful. In addition, GANs performance on our grey scale images are not well. Figure 6.6 summarizes the data augmentation methods we use in our research.

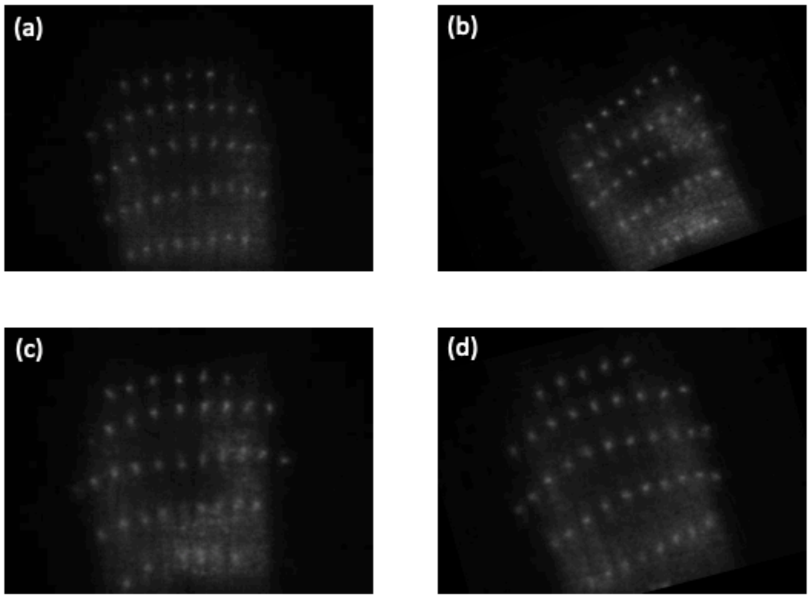


Figure 6. 6 Data augmentation (a) original image, (b) rotated image, (c) scaled image, (d) rotated and scaled image.

In table 6, we summarized the data size before and after data augmentation.

Table 6 Data size after data augmentation

Label	Number of raw images	Number of images after augmentation
0	457	59,868
1	495	64,846
2	540	70,696
3	570	74,101
4	626	70,739
5	862	87,063

To make sure the training results of the neural networks are convincing, the training dataset, validation set and test set must be completely different. Since if the test set is the images that used to training, we cannot tell this neural network has actually learned or just remember all the images. Therefore, when creating the test set, we randomly pick one image from each of the six labels and do not put them back. Do this cycle for 7,500 times, we create a test set that contains 45,000 images. Same thing with validation set and training set. Therefore, we create a training set, which contains 270,000 images, a validation set, which contains 45,000 images and a test set, which contains 45,000 images.

Chapter 7 Training a CNN

In the previous chapters, we have designed a six-layer CNN, created three independent dataset: training data, validation set and test set. In this chapter, we will discuss how to train a convolutional neural network (CNN). The loss function is firstly discussed; the optimization methods including mini-batch gradient descent and Adam are compared. Learning tricks like learning rate annealing and early stopping are used in our training process. Finally, we will discuss the batch normalization, which controls the overfitting.

7.1 Loss function

Loss function needs to be firstly define in the training process, since the loss function evaluates the degree of consistency between the results got from the CNN and the ground truth table[59]. Many loss functions have been proposed, such as gold standard[74], hinge loss[75], log loss including cross entropy error[76, 77], squared loss[78] and exponential loss[79].

The gold standard loss is also called 0-1 loss; it is used to record the times that prediction result match the truth, see equation 7.1.

$$L(y, f(x)) = \begin{cases} 0 & if\ y = f(x) \\ 1 & if\ y \neq f(x) \end{cases} \tag{7.1}$$

Where y is the true label and $f(x)$ is the predict label. In our research, the softmax regression gives the probabilities of the six labels (0 to 5) given an input. The maximum is picked as the predict label, when calculating the validation error and test error, the 0-1 loss is used. For a mini batch (600 in our research), each time we match the real label, we add 1 to the sum S. Therefore, the validation/test error is defined in equation 7.2.

$$Error = 1 - \frac{S}{N} \tag{7.2}$$

Where S is the sum, N is the mini-batch size.

The hinge loss function is widely used in support vector machine (SVM)[75]. For a classification problem, define y is the prediction value, not the label, t is either 0 or 1. The hinge loss function is defined:

$$L(y) = \max(0, 1 - t * y) \tag{7.3}$$

Where in SVM, $y = w * x + b$.

The support vector machine achieves impressive results in classification[80-82], but the deep convolutional neural network outperform by the accuracy[7] and the running speed[83]. Therefore, in our research, we use the convolutional neural networks to do the classification.

Log loss function is discussed in the regression layer part. The loss function for softmax regression is

$$min_w \quad L(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k}[1[y_i = 1]log\frac{\exp(w_j * x_i)}{\sum_{n=1}^{k}\exp(w_n * x_i)}] \tag{7.4}$$

Where the equation 7.4 is the same as equation 5.14

In some cases, the regularization term is added to the loss function to control the overfitting. L-2 norm regularization is most used which can be written in equation 7.5[59].

$$\lambda \sum\nolimits_{k \in K} (w_k)^2 \tag{7.5}$$

Where λ is the penalty factor, controls the degree regularization.

Thus, the loss function can be written

$$\begin{aligned} min_w \quad L(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k}[1[y_i = 1] log \frac{\exp(w_j * x_i)}{\sum_{n=1}^{k} \exp(w_n * x_i)}] \\ + \lambda \sum\nolimits_{k \in K} (w_k)^2 \end{aligned} \tag{7.6}$$

Calculating the minimum, the sum of the parameters should be minimized, which six. Avoiding the situation like one large parameters along with many zeros. The training results are smoother and have more generalization ability.

In our research, batch normalization is applied to control the overfitting, thus the details of the overfitting will not be discussed.

7.2 Optimizer

The goal of the training process is to minimize the loss function, in physical meaning, to make the predict results the same as the true label. Rumelhart et al [33] proposed the error BackPropagation (BP) algorithm to train a three layers neural network. Currently, BP algorithm has become the most common algorithms to train a neural network. Mini-batch gradient descent and Adaptive moment estimation (Adam) are both based on the BP algorithm.

7.2.1 Mini-batch gradient descent

As discussed, the mini-batch gradient descent is based on the BP algorithm. Therefore, we discuss the BP algorithm first. The BP algorithm is to change the parameters (weights, bias) according to the input samples, to make the output close the desire truth. In summary, the training process can be divide into two parts: feedforward pass and backpropagation pass. Figure 7.1 shows a three layers BP neural networks[84].

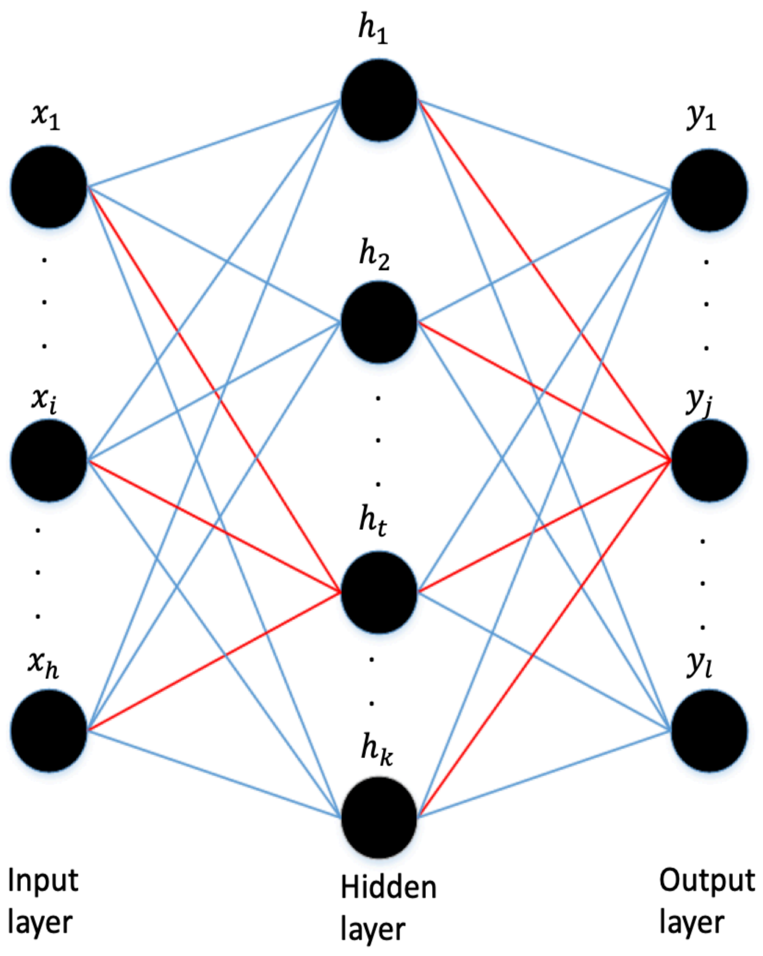


Figure 7. 1 A three-layer BP network

This neural network contains three layers: input layer, hidden layer and the output layer. Input layer has h neurons, hidden layer has k neurons and the output layer contains l neurons. Given the input data $D: (x_1, y_1),\ (x_2, y_2), \dots (x_n, y_n),\ x_i \in R^h, y_i \in R^l$. The weights between the input layer neuron i and hidden layer neuron h is indicated byv_{ih}, and w_{hj} means the weights between the hidden layer neuron h and the output layer neuron j. The threshold of the output is written as θ_j for neuron j and γ_j for neuron j in the hidden layer. The activation function shown is sigmoid function, and can be other types of activation function.

Therefore, in the feedforward pass, initialize the parameters (weights, bias) for each layer, gives an input (x_k, y_k), outputs the results $\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \dots\ \hat{y}_l^k)$.

$$\hat{y}_j^k = f(h_i^o - \theta_j) \tag{7.7}$$

Therefore, the square error is defined as:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2 \tag{7.8}$$

If the error is acceptable, the parameters need not to be optimized. The training process is end. However, when the error is not desirable, the backpropagation pass works. BP algorithm optimizing the parameters (weights and bias) based the error defined in equation 7.8. Two methods are typically used in optimizing, one is gradient descent the other is least square. Both of them take derivative to find the minimum of the loss function. However, the least square is non-iterative and trying to find the global minimum. The gradient descent is iterative and after several iterations, find the local minimum.

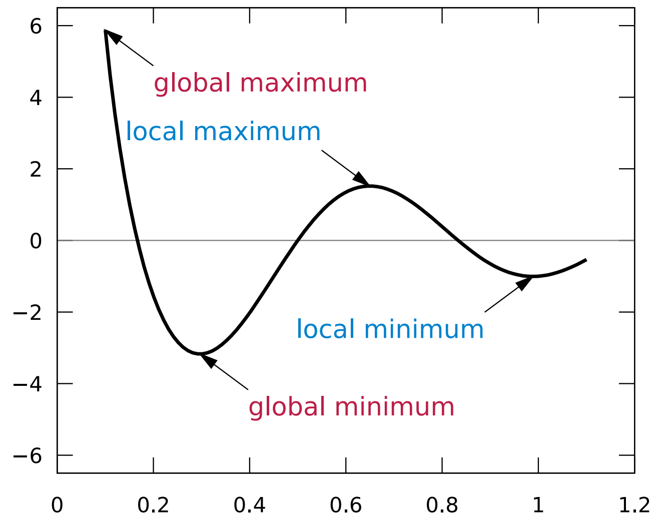


Figure 7. 2 Global minimum and local minimum[85]

In our research, the mini-batch gradient descent is used, the following steps are based on the gradient descent method.

For the t_{th} neuron in the hidden layer, the input of it is

$$\alpha_h = \sum_{i=1}^{h} v_{ih} * x_i$$

Similar way the input of the j_{th} neuron in the output layer is

$$\beta_j = \sum_{h=1}^{k} w_{hj} * h_h$$

For a given learning rate η, we have

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} \tag{7.9}$$

Based on the chain-rule[86], we have

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} * \frac{\partial \hat{y}_j^k}{\partial \beta_j} * \frac{\partial \beta_j}{\partial w_{hj}} \tag{7.10}$$

From the definition of β_j, we have

$$\frac{\partial \beta_j}{\partial w_{hj}} = h_h \tag{7.11}$$

Therefore, based on equation 7.8 and 7.7, we have

$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} * \frac{\partial \hat{y}_j^k}{\partial \beta_j}$$

$$= -\left(\hat{y}_j^k - y_j^k\right) f'\left(\beta_j - \theta_j\right)$$

$$= \hat{y}_j^k (1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k) \tag{7.12}$$

Where for Sigmoid function, $f'(x) = f(x)\left(1 - f(x)\right)$

Apply equation 7.12 and 7.11 into the equation 7.10 and 7.9, we have

$$\Delta w_{hj} = \eta g_j h_h \tag{7.13}$$

Similarly, we have

$$\Delta \theta_j = -\eta g_j \tag{7.14}$$

$$\Delta v_{ih} = \eta e_h x_i \tag{7.15}$$

$$\Delta \gamma_h = -\eta e_h \tag{7.16}$$

Where $e_h = \frac{\partial E_k}{\partial h_h} * \frac{\partial h_h}{\partial \alpha_h}$

$$e_h = \frac{\partial E_k}{\partial h_h} * \frac{\partial h_h}{\partial \alpha_h} = -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} * \frac{\partial \beta_j}{\partial h_h} f'(\alpha_h - \gamma_h)$$

$$= \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h)$$

$$= h_h(1 - h_h) \sum_{j=1}^{l} w_{hj} g_j \tag{7.17}$$

Next, do iteration process based on these update equations until reach the optimum. The process above shows the way using one sample to do the gradient descent. Based on how many samples are used in gradient descent, three methods are defined. Batch gradient descent uses the whole data to do the gradient descent. Each iteration the whole data are used to calculate the gradient makes it most accurate [87]. However, when the dataset is too large, the speed will be slow. On the other hand, stochastic gradient descent uses only one sample each time to calculate the gradient, but the accuracy is not good. Mini-batch gradient descent uses a small batch (for example 600 in our research) to calculate the gradient each time. It balances the accuracy and the speed, and becomes most common used [7, 9-11, 15, 35, 37, 43, 56]. Therefore, the loss function can be written as equation 7.18

$$E = \frac{1}{2N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 \tag{7.18}$$

Where N is the mini-batch number, 600 in our research. In addition, to increase the convergence speed and escape the saddle point, the momentum term is added in our mini-batch gradient descent[59].

7.2.2 Adaptive moment estimation

Besides the mini-batch gradient descent, adaptive moment estimation (Adam) is another important optimizer. Adam combines the advantages of two methods: AdaGrad[88] and RMSProp[89], for each iteration, the learning rate is bounded in a certain range[90]. It calculated the gradient's first moment estimate and second moment estimate to adjust each parameter's learning rate η. The details will not be discussed in our research, readers can refer the paper[90]. Equation 7.19 to 7.23 shows the updating rule for the parameters.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{7.19}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{7.20}$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{7.21}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{7.22}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{7.23}$$

Where g_t is the gradient respect to the parameters, m_t is the first moment estimate, and $m_0 = 0$, v_t is the second moment estimate and $v_0 = 0$. By default, $\alpha = 0.001$, $\beta_1 = 0.9$, β_2=0.999 and $\epsilon = 10^{-8}$.

The revised Adam has been proposed and achieves good performance, like AdaMax and Nadam[87], but will not be discussed in our research.

The performance in our research is slightly worse than the revised mini-batch gradient descent (22.750% validation error compared to 22.083% validation error in the dataset), but the convergence speed is much faster. For performance view, we use the revised mini-batch gradient descent, and the details will show in next part.

7.3 Initialization

The activation function used in our research is Rectified Linear Unit (ReLU), cause it converges faster[7] and has better performance[54, 56]. Therefore, the initialization process based on the ReLU activation function. Initialization set the starting point of the training process, a bad initialization results in sticking on saddle point[91], much longer training time, etc. The Gaussian distribution initialization is widely used in current neural networks. Recall that smooth functions have more generalization ability. In our research, zero mean and 0.01 variance Gaussian distribution for weights, bias set to 0 is firstly tried. The validation error is 22.083%. Xavier initialization[92, 93] is another widely used initialization method. Unlike the Gaussian distribution initialization, the size of the previous layer is considered. The bias are kept 0 while the weights are initialization as equation 7.24.

$$w_{ij} \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}] \tag{7.24}$$

Where U means uniform distribution and n is the size of last layer.

However, the Xavier initialization is not fit for nonlinear activation functions like ReLU[55]. Therefore, the revised Xavier initialization[55] is proposed to deal with nonlinear activation functions and to solve the hard convergence for very deep neural networks. Revised Xavier initialization use a Gaussian distribution with zero mean and $\sqrt{2/n_i}$ standard deviation. Where n_i denotes the input dimension of the layer. However, in our six-layer CNN, the revised Xavier initialization ends with a saddle point. Further reduce the revised initialization by 0.1, validation error is 22.483% validation error, which is worse than the Gaussian distribution initialization.

7.4 Learning rate annealing and early stopping

The learning rate set the step size each time updating the parameters. Too much learning rate hinder the training process or even converge to a fake minimum while small learning rate makes training process extremely slow to converge, see figure 7.3. However, for a training process choosing an appropriate learning rate can be difficult. A better way is to use learning rate annealing method.

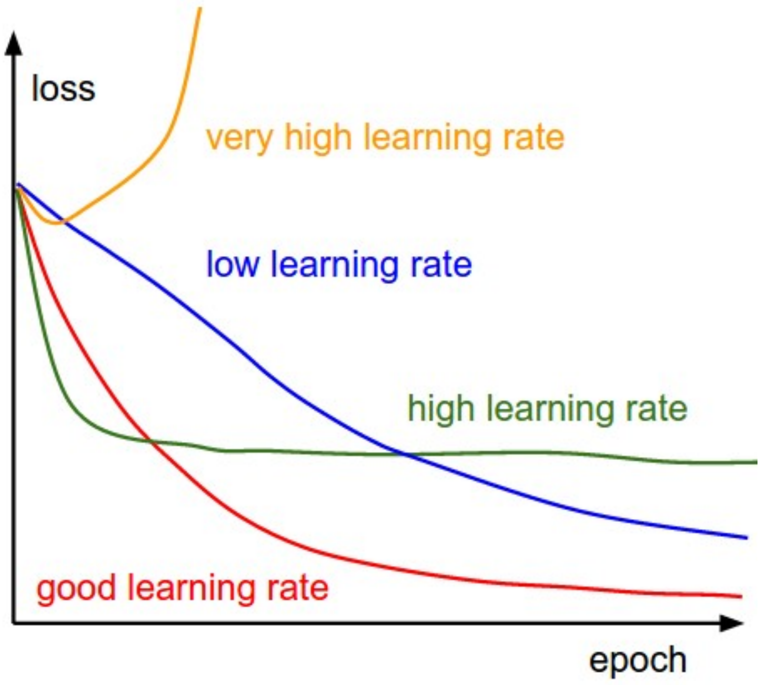


Figure 7. 3 Different learning rates affect the training process[59]

Learning rate annealing firstly set a slightly bigger learning rate to make sure fast convergence in the beginning of the training process. Then decaying the learning rage based on time or on the number of epochs[94]. In [95], an adaptive learning rate method (ADADELTA) has been proposed. The learning rate is calculated based on the L2 norm of all the gradients of previous, which makes the learning rate larger when the gradient smaller. However, the initial learning rate for ADADELTA is critical. Therefore, in our research, step decay is used where the initial learning rate is set to 0.01, and when the validation error stops decreasing for three consecutive epochs, decrease the learning rate by 0.5. Figure 7.4 shows the learning rate annealing in our training process.

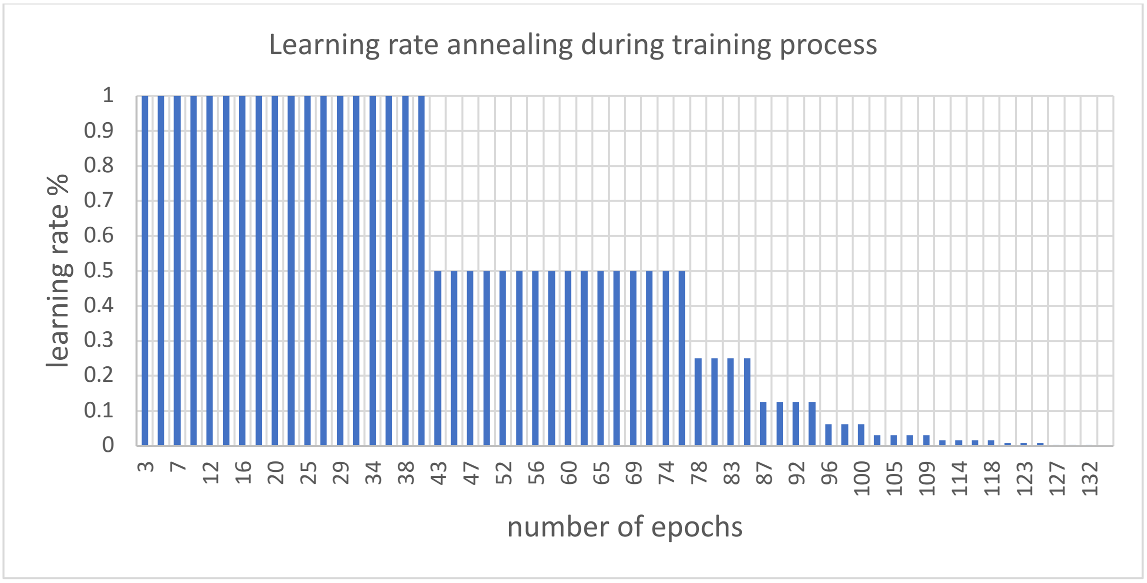


Figure 7. 4 Learning rate annealing in our training process

As discussed, the mini-batch gradient descent is an iteration process. Traditional way to control the end of an iteration process is to set a tolerate threshold, when the error is below this threshold, the iteration process stops. However, for CNN, that threshold is hard to determine, which means the epoch number is critical. The training process terminates without reach to the optimum when the epoch number is set too small. On the other hand, too big epoch number causes the training process taking much time. To solve this dilemma, we use set a big epoch number to make sure training process will not terminates before reaching the optimum but at the same apply the early stop mechanism. The training process will terminates when the validation error stops decreasing for ten consequent epochs.

7.5 Batch Normalization

In our defined loss function, the L2 norm regularization is not added to control the overfitting. The overfitting should be avoided since at that time, the trained neural network matches too closely to training data even the noise or bad samples are considered. Although the loss function is minimized, the generalization ability is decreased.

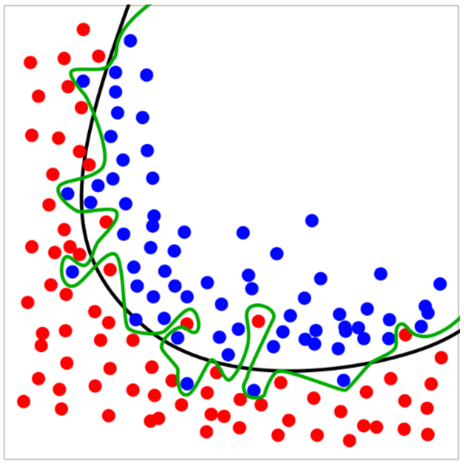


Figure 7. 5 Overfitting decreased the generalization ability[96]

The dropout[7] and batch normalization[97] are the two widely used methods controlling the overfitting. Dropout adds a probability that determine each neuron work or not work. For one neuron view, the dropout forces it to work with different neurons each time, decreasing the combination between the neurons and increasing the generalization ability[98]. However, in our research, we use the batch normalization as the way controlling the overfitting. In the training process, the parameters (weights and bias) need to be update to minimize the loss function. However, the updates of parameters from all the previous layers affect the distribution of the input of next layer. Sergey Ioffe and Christian Szegedy [97] defined this distribution change as internal covariate shift[99]. Batch normalization is proposed to deal with the internal covariate shift. As a way of Principal Components Analysis (PCA), whitening is widely used for data preprocessing[100, 101], but whitening requires much computation. So in batch normalization, the authors a close way to whitening preprocessing method.

For each dimension of the input[97]

$$\hat{x}^k = \frac{x^k - E(x^k)}{\sqrt{Var(x^k)}} \tag{7.25}$$

Where the mean E and variance Var are calculate based on the mini-batch data.

However, by doing this, the features learned from last layer are distorted, thus two learnable parameters are used to restore the features. For each activation x^k, $\gamma^k and\ \ \beta^k$, we have

$$y^k = \gamma^k * \hat{x}^k + \beta^k \tag{7.26}$$

Where γ^k and β^k are learned during training process.

In this way, BN layer forces the unordered distribution into an ordered distribution, which we need to learn.

Besides controlling the overfitting, BN layer speeds up the completely training process, and is able to save a bad initialization to some extent.

7.6 Summary

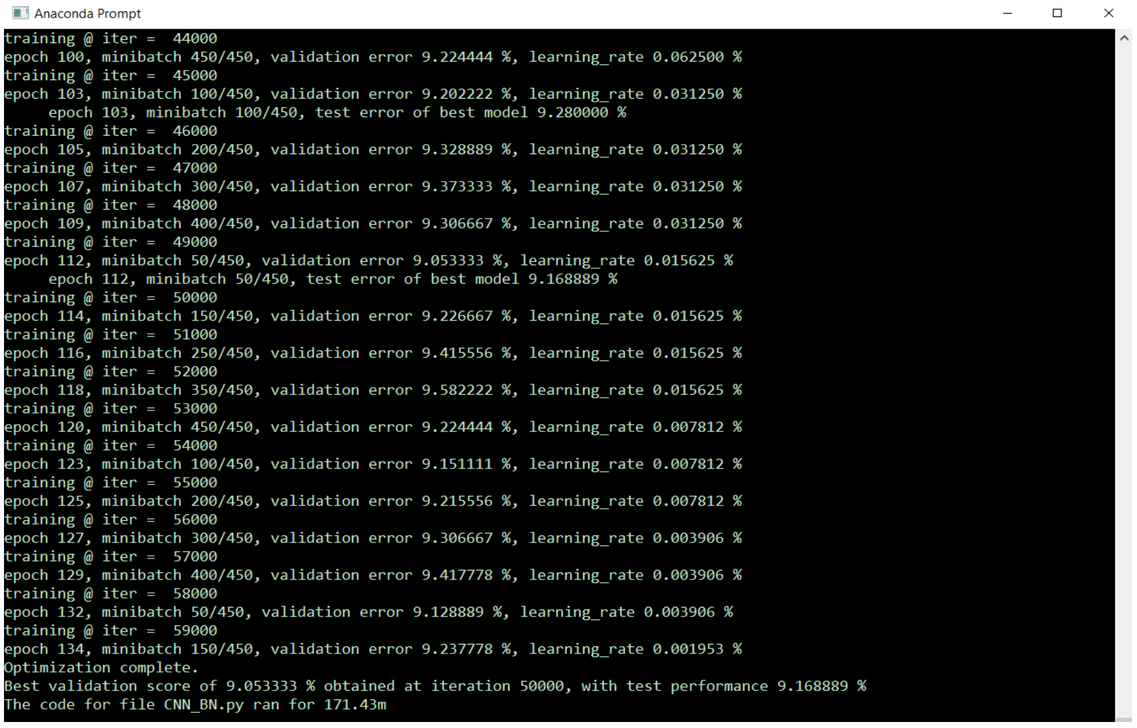
In this chapter, the training method used in our research has been discussed. For the performance view, we use mini-batch gradient descent with learning rate annealing. The initialization is from the Gaussian distribution with zero mean and 0.01 variance. The training process terminates when the validation error stop decreasing for ten consequently periods. Batch normalization is used to control overfitting.

Chapter 8 Results

In this chapter, we will discuss the results of our trained CNN. The plot of training error, validation error and test error are given. The six-layer CNN give us 90.83% prediction accuracy about the identification of weld penetration. A voting mechanism has been proposed based on three consequent images, which increase the prediction accuracy to over 97%.

8.1 Preliminary results

All the training work is based on the GTX1080 GPU. After 134 epochs, the training process terminates with best validation error is 9.053%, and the test error 9.169%. Shown in figure 8.1.



```
training @ iter =  44000
epoch 100, minibatch 450/450, validation error 9.224444 %, learning_rate 0.062500 %
training @ iter =  45000
epoch 103, minibatch 100/450, validation error 9.202222 %, learning_rate 0.031250 %
     epoch 103, minibatch 100/450, test error of best model 9.280000 %
training @ iter =  46000
epoch 105, minibatch 200/450, validation error 9.328889 %, learning_rate 0.031250 %
training @ iter =  47000
epoch 107, minibatch 300/450, validation error 9.373333 %, learning_rate 0.031250 %
training @ iter =  48000
epoch 109, minibatch 400/450, validation error 9.306667 %, learning_rate 0.031250 %
training @ iter =  49000
epoch 112, minibatch 50/450, validation error 9.053333 %, learning_rate 0.015625 %
     epoch 112, minibatch 50/450, test error of best model 9.168889 %
training @ iter =  50000
epoch 114, minibatch 150/450, validation error 9.226667 %, learning_rate 0.015625 %
training @ iter =  51000
epoch 116, minibatch 250/450, validation error 9.415556 %, learning_rate 0.015625 %
training @ iter =  52000
epoch 118, minibatch 350/450, validation error 9.582222 %, learning_rate 0.015625 %
training @ iter =  53000
epoch 120, minibatch 450/450, validation error 9.224444 %, learning_rate 0.007812 %
training @ iter =  54000
epoch 123, minibatch 100/450, validation error 9.151111 %, learning_rate 0.007812 %
training @ iter =  55000
epoch 125, minibatch 200/450, validation error 9.215556 %, learning_rate 0.007812 %
training @ iter =  56000
epoch 127, minibatch 300/450, validation error 9.306667 %, learning_rate 0.003906 %
training @ iter =  57000
epoch 129, minibatch 400/450, validation error 9.417778 %, learning_rate 0.003906 %
training @ iter =  58000
epoch 132, minibatch 50/450, validation error 9.128889 %, learning_rate 0.003906 %
training @ iter =  59000
epoch 134, minibatch 150/450, validation error 9.237778 %, learning_rate 0.001953 %
Optimization complete.
Best validation score of 9.053333 % obtained at iteration 50000, with test performance 9.168889 %
The code for file CNN_BN.py ran for 171.43m
```

Figure 8. 1 Training results

Figure 8.2 shows the training error, validation error and test error during the training process.

The training error is 0.195%, so the training process makes the prediction very close to the truth.

The gap between the training error and test error is neither too large nor too small which indicates the overfitting is acceptable. Remember that validation set and test set are different, the validation error and test error is close also shows this trained CNN performs close on two different data set, showing that the overfitting is not serious.

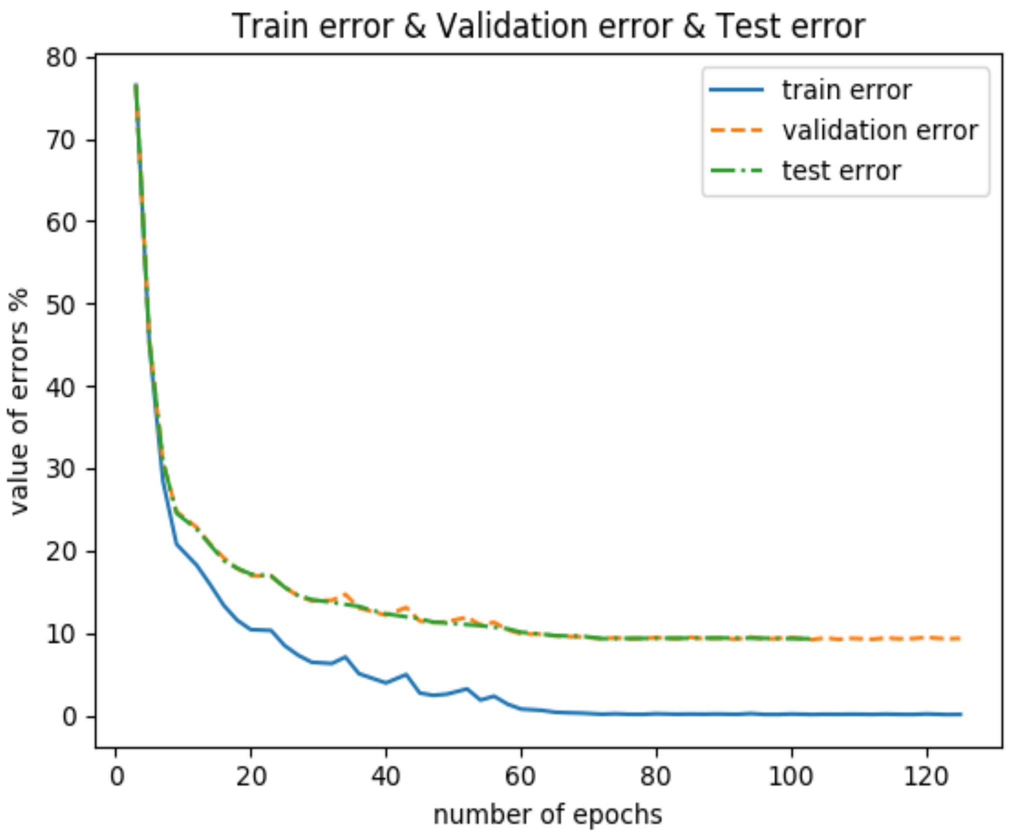


Figure 8. 2 The training error, validation error and test error during training process

8.2 Voting mechanism and control

As discussed in chapter 3, each low current period, and the high-speed camera capture three consequent images. Welding process is a dynamic process, the wrong prediction happens only between close labels, which means the true label is 3, but the CNN gives us 2 or 4, the possibility it gives 0 can be dismissed. Therefore, a voting mechanism based on three images has been proposed to further increase the prediction accuracy. There are two cases when voting:

1. At least two images output the same label, then output that label as the final label;
2. All the three images output different labels. At that time we do not use any of them, wait for next cycle.

For one image, the prediction accuracy is 90.83%, so based on the voting mechanism, the prediction accuracy is

$$0.9083^3 + 3 * 0.9083^2 * (1 - 0.9083) = 0.9763 = 97.63\%$$

However, these three images are not independent, the prediction accuracy is improved after voting, but cannot get 97.63%.

The control system in our research can be designed in figure 8.3.

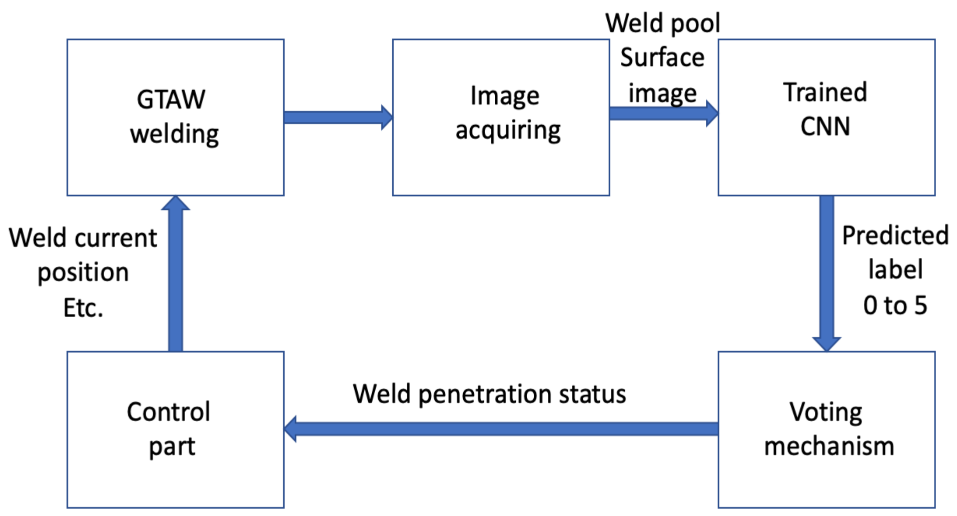


Figure 8. 3 Weld process control based on CNN

Chapter 9 Conclusion and future work

In our research, we propose an innovative approach identify weld penetration status using convolutional neural network (CNN). A sensing system, which collect both weld pool surface and backside of the weld pool, has been built to generate train data. Label preserving data augmentation including rotation, scale has been done to create over 360,000 data. A six-layer CNN has been designed and trained based on augmented dataset. Final test accuracy rate of the proposed CNN model is 90.83% and the accuracy rate can be further improved by using a voting mechanism, which is regarded to be good enough in practical industrial welding manufacturing.

The control system has been designed like figure 8.3. However, the hardware limitation stopped us. The computer in the control part is equipped with an AMD A6-3650 2.6 GHz CPU. Running our trained CNN takes over 300 milliseconds, which is too long for real welding control. Running on a GPU is much faster, an eight-layer CNN gets results in 1.2 milliseconds on a GPU[10]. However, the control computer has been collected with two data collection cards, no room to install a GPU. In future, the welding sensing and control system will be established with a GPU.

Reference:

[1] W. Chen and B. Chin, "Monitoring joint penetration using infrared sensing techniques," *Welding Journal,* vol. 69, pp. 181s-185s, 1990.

[2] S. Nagarajan, W. Chen, and B. Chin, "Infrared sensing for adaptive arc welding," *Welding Journal,* vol. 68, pp. 462-466, 1989.

[3] S. Nagarajan, P. Banerjee, W. Chen, and B. A. Chin, "Control of the welding process using infrared sensors," *IEEE Transactions on Robotics and Automation,* vol. 8, pp. 86-93, 1992.

[4] N. Carlson and J. Johnson, "Ultrasonic sensing of weld pool penetration," *Welding Journal (Miami);(USA),* vol. 67, 1988.

[5] D. Hardt and J. Katz, "Ultrasonic measurement of weld penetration," *Welding Journal,* vol. 63, pp. 273s-281s, 1984.

[6] A. Aendenroomer and G. Den Ouden, "Weld pool oscillation as a tool for penetration sensing during pulsed GTA welding," *WELDING JOURNAL-NEW YORK-,* vol. 77, pp. 181-s, 1998.

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

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov*, et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

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

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

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915,* 2016.

[12] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology,* vol. 160, pp. 106-154, 1962.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[14] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 248-255.

[16] D. Kotecki, D. Cheever, and D. Howden, "Mechanism of ripple formation during weld solidification," *WELD J,* vol. 51, p. 368, 1972.

[17] R. Renwick and R. Richardson, "Experimental investigation of GTA weld pool oscillations," *WELDING J.,* vol. 62, p. 29, 1983.

[18] Y. H. Xiao and G. D. Ouden, "A study of GTA weld pool oscillation," *Welding Journal,* vol. 69, p. 289, 1990.

[19] Y. H. Xiao and G. D. Ouden, "Weld pool oscillation during GTA welding of mild steel," *Welding Journal,* vol. 72, pp. 428-s, 1993.

[20] K. Andersen, G. E. Cook, R. J. Barnett, and A. M. Strauss, "Synchronous weld pool oscillation for monitoring and control," *IEEE Transactions on Industry Applications,* vol. 33, pp. 464-471, 1997.

[21] B. Yudodibroto, M. Hermans, Y. Hirata, and G. den Ouden, "Influence of filler wire addition on weld pool oscillation during gas tungsten arc welding," *Science and technology of welding and joining,* vol. 9, pp. 163-168, 2004.

[22] P. A. Kotidis, J. F. Cunningham, P. F. Gozewski, C. Borsody, D. E. Klimek, and J. A. Woodroffe, "Laser ultrasonics-based material analysis system and method using matched filter processing," ed: Google Patents, 1997.

[23] R. Kovacevic, Y. Zhang, and S. Ruan, "Sensing and control of weld pool geometry for automated GTA welding," *Journal of Engineering for Industry,* vol. 117, pp. 210-222, 1995.

[24] W. Zhang, "Machine-human Cooperative Control of Welding Process," 2014.

[25] W. Zhang, Y. Liu, X. Wang, and Y. Zhang, "Characterization of three-dimensional weld pool surface in GTAW," *Welding Journal,* vol. 91, pp. 195s-203s, 2012.

[26] H. S. Song and Y. M. Zhang, "Three-dimensional reconstruction of specular surface for a gas tungsten arc weld pool," *Measurement Science and Technology,* vol. 18, p. 3751, 2007.

[27] T. S. Kumar, V. Balasubramanian, and M. Sanavullah, "Influences of pulsed current tungsten inert gas welding parameters on the tensile properties of AA 6061 aluminium alloy," *Materials & design,* vol. 28, pp. 2080-2092, 2007.

[28] Wikipedia. *Bilinear interpolation*.

[29] M.-v. s. e. h. v. s. u. F.-b. p. processing. (2010). *System Stimulus*.

[30] K. Fukushima, "Neocognitron--a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *NHK 放送科学基礎研究所報告,* pp. p106-115, 1981.

[31] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron," *IEICE Technical Report, A,* vol. 62, pp. 658-665, 1979.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science1985.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature,* vol. 323, p. 533, 1986.

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

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

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

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

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair*, et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, pp. 436-444, 2015.

[40] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications* vol. 31999: McGraw-Hill New York, 1986.

[41] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *ICDAR*, 2003, pp. 958-962.

[42] J. Bouvrie, "Notes on convolutional neural networks," 2006.

[43] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly*, et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine,* vol. 29, pp. 82-97, 2012.

[44] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111-118.

[45] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," 2010.

[46] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding,* vol. 106, pp. 59-70, 2007.

[47] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557,* 2013.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European conference on computer vision*, 2014, pp. 346-361.

[49] A. Zell, *Simulation neuronaler netze* vol. 1: Addison-Wesley Bonn, 1994.

[50] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural network design* vol. 20: Pws Pub. Boston, 1996.

[51] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review,* vol. 65, p. 386, 1958.

[52] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks,* vol. 2, pp. 183-192, 1989.

[53] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[54] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013.

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

[56] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315-323.

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

[58] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434,* 2015.

[59] A. Karpathy, "Cs231n: Convolutional neural networks for visual recognition," *Neural networks,* vol. 1, 2016.

[60] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks,* vol. 2, pp. 359-366, 1989.

[61] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks,* vol. 4, pp. 251-257, 1991.

[62] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems,* vol. 2, pp. 303-314, 1989.

[63] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning* vol. 1: MIT press Cambridge, 2016.

[64] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 2924-2932.

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

[66] leonardblier. (2016). *A BRIEF REPORT OF THE HEURITECH DEEP LEARNING MEETUP #5*.

[67] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," *arXiv preprint arXiv:1609.08764,* 2016.

[68] N. Pinto, D. D. Cox, and J. J. DiCarlo, "Why is real-world visual object recognition hard?," *PLoS computational biology,* vol. 4, p. e27, 2008.

[69] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126,* 2016.

[70] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a￼ laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486-1494.

[71] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234-2242.

[72] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396,* 2016.

[73] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint,* 2017.

[74] V. Vapnik, *Statistical learning theory. 1998* vol. 3: Wiley, New York, 1998.

[75] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications,* vol. 13, pp. 18-28, 1998.

[76] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression* vol. 398: John Wiley & Sons, 2013.

[77] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected papers of hirotugu akaike*, ed: Springer, 1998, pp. 199-213.

[78] W. S. Lee, P. L. Bartlett, and R. C. Williamson, "The importance of convexity in learning with squared loss," *IEEE Transactions on Information Theory,* vol. 44, pp. 1974-1980, 1998.

[79] H. Masnadi-Shirazi and N. Vasconcelos, "On the design of loss functions for classification: theory, robustness to outliers, and savageboost," in *Advances in neural information processing systems*, 2009, pp. 1049-1056.

[80] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters,* vol. 9, pp. 293-300, 1999.

[81] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, 1998, pp. 137-142.

[82] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research,* vol. 2, pp. 45-66, 2001.

[83] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks,* vol. 13, pp. 415-425, 2002.

[84] 周志华, *机器学习*: Qing hua da xue chu ban she, 2016.

[85] Wikipedia. *Maxima and minima*.

[86] S. Hilger, "Analysis on measure chains—a unified approach to continuous and discrete calculus," *Results in Mathematics,* vol. 18, pp. 18-56, 1990.

[87] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747,* 2016.

[88] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research,* vol. 12, pp. 2121-2159, 2011.

[89] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning,* vol. 4, pp. 26-31, 2012.

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

[91] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in neural information processing systems*, 2014, pp. 2933-2941.

[92] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249-256.

[93] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick*, et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675-678.

[94] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, 1992, pp. 3-12.

[95] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701,* 2012.

[96] Wikipedia. *Overfitting*.

[97] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448-456.

[98] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of machine learning research,* vol. 15, pp. 1929-1958, 2014.

[99] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference,* vol. 90, pp. 227-244, 2000.

[100] A. Hyvärinen, J. Hurri, and P. O. Hoyer, "Principal components and whitening," in *Natural Image Statistics*, ed: Springer, 2009, pp. 93-130.

[101] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, ed: Springer, 2012, pp. 9-48.

VITA

Chao Li

EDUCATION

Master of Science in Electrical and Computer Engineering at Ohio State University (OSU), August 2013 – December 2014.

Bachelor of Engineering in Electrical Engineering at Beijing Jiaotong University (BJTU), August 2009 – July 2013.

PUBLICATIONS

Machine Learning Based Detection of Weld Joint Penetration from Weld Pool Reflection Images – IEEE Transactions on Automation Science and Engineering (Revised and submitted)

A tutorial for deep learning applied in manufacturing area – (in progress)