2019

Real-time Automated Weld Quality Analysis From Ultrasonic B-scan Using Deep Learning

Zarreen Naowal Reza
University of Windsor

Real-time Automated Weld Quality Analysis From Ultrasonic B-scan Using Deep Learning

by

Zarreen Naowal Reza

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2019

Real-time Automated Weld Quality Analysis From Ultrasonic B-scan Using Deep Learning

by

Zarreen Naowal Reza

APPROVED BY:

J. Wu,

Department of Electrical and Computer Engineering

B. Boufama,

School of Computer Science

D. Wu, Co-Advisor

School of Computer Science

R. Maev, Co-Advisor

Department of Physics

March 29, 2019

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Resistance spot welding is a widely used process for joining metals using electrically generated heat or Joule heating. It is one of the most commonly used techniques in automotive industry to weld sheet metals in order to form a car body. Although, industrial robots are used as automated spot welders in massive scale in the industries, the weld quality inspection process still requires human involvement to decide if a weld should be passed as acceptable or not. Not only it is a tedious and error-prone job, but also it costs industries lots of time and money. Therefore, making this process automated and real-time will have high significance in spot welding as well as the field of Non-destructive Testing (NDT). Research team in Institute of Diagnostic Imaging Research (IDIR) have developed technology to obtain grey-scale 2D images called ultrasonic b-scans in real-time during production in order to visualize the weld development with respect to time. They have demonstrated that by extracting and interpreting relevant patterns from these b-scans, weld quality can be determined accurately. However, current works combining conventional image and signal processing techniques are unable to extract those patterns from a wide variety of weld shapes with production-level satisfaction. Therefore, in this thesis, we propose to apply SSD, a single-shot multi-box detection based deep convolutional neural network framework for real-time embedded detection of components of cross-sectional weld shape from ultrasonic b-scans and interpret them to numeric parameters which are used as features to classify welds as good, bad or acceptable in real-time. Our proposed model has showed significant improvement in deciding weld quality compared to existing methods when tested on real industry facility.

Dedication

I would like to dedicate this thesis to my father Mr. M Murtozaa Reza. Without his continuous encouragement and immense support I would not be able to reach even close to where I am today.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to my supervisors Dr. Dan Wu and Dr. Roman Maev and also Dr. Andriy Chertov for believing in me and my capabilities. It meant a lot to me and worked as a great incentive. Their constant guidance and encouragement during my whole masters journey have been tremendously effective and valuable. Without them, this thesis would never be possible. I would also like to express my sincere gratitude to my thesis committee members Dr.Boubakeur Boufama, and Dr.Jonathan Wu for the precious time they have invested to evaluate my research and provide valuable feedback and appreciation.

Last but not the least, I would like to acknowledge the tremendous love, encouragement and support I had been given by my father, my uncle and aunt who I have stayed with in Canada and my friends and relatives which have helped me to keep myself focused and motivated towards the fulfillment of my target.

Zarreen Naowal Reza

Contents

List of Tables

List of Figures

Epigraph

"As a student, there appeared to me to be only three interesting problems in the world or in the world of science, at least. Genetics seemed to be pretty interesting, because nobody knew yet how it worked, but I wasn't sure that it was profound. The problems of physics seemed profound and solvable. It might have been nice to do physics. But the problem of intelligence seemed hopelessly profound. I can't remember considering anything else worth doing."

Marvin Lee Minsky (Co-founder of AI)

Chapter 1

Introduction

1.1 Context

1.1.1 Resistance Spot Welding

Resistance Spot Welding is the process of joining two to four gauge overlapping metal sheets to form car bodies. In Fig. 1.1, white arrows show some spot welding holes made to join parts of a car door. It is one of the oldest electric welding processes being used by automotive industry today. It started being used in welding field in between years 1900-1905. In modern cars, there are between 3 to 4 thousand weld joints.

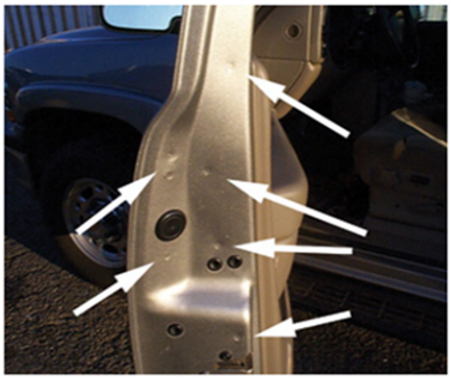


Figure 1.1: Resistance spot welding in car industry [54].

As shown in stage A and B in Fig. 1.2, in resistance spot welding process, two or more metal sheets are clamped together by heavy pressure F exerted by two copper electrodes from top and bottom. At stage C electrodes concentrate large welding current ranged from 5-13 kA into a small spot, marked as I. Forcing this large current for a duration of 200-300ms through the spot lets the metals to melt at the desired spot. This duration is determined by the material thickness and type, amount of the current to concentrate, and cross-sectional area of the welding tips and contact surfaces. After the small duration, current is turned off however metals are still held together so that the melted liquid can be solidified as a nugget, as seen in stage D. Once the nugget is formed and solidified, the metals involved in the process are joined to each other in stage E.

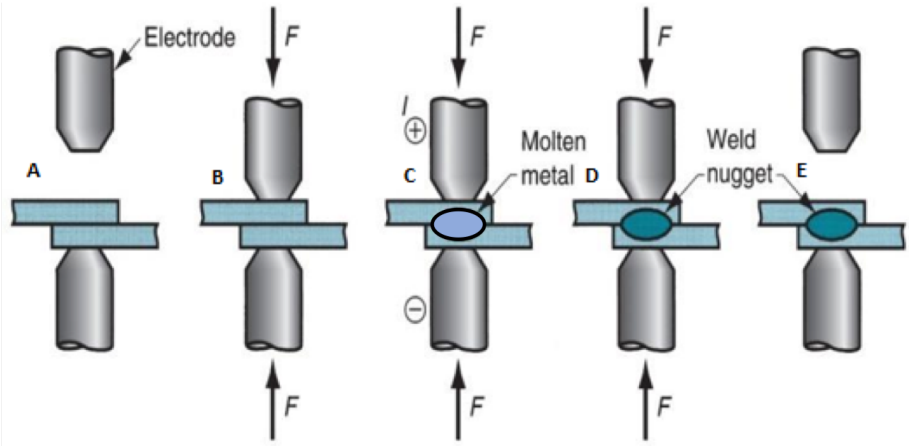


Figure 1.2: Process of spot welding [57].

1.1.2 Ultrasonic Non-Destructive Testing

Non-destructive testing (NDT), interchangeably known as non-destructive evaluation (NDE) and non-destructive inspection (NDI), is a group of analysis techniques used for inspecting material structures and detect any flaws or discontinuities in material composition without damaging or altering the test sample [15]. Because the test unit is not permanently tempered while being inspected, NDT is a highly valuable tech-

nique that can save both money and time in product evaluation, troubleshooting, and research [67]. The six most frequently used NDT methods are eddy-current, magnetic-particle, liquid penetrant, radiographic, ultrasonic, and visual testing. Among them, ultrasonic NDT is the most popular technique for weld joint inspection [68]. It is cost-efficient and also the critical method to inspect materials that cannot be exposed under radiation or electric field [68]. Ultrasonic NDT methods use ultrasonic waves that penetrates the metal sheets and sent back information of the internal structure of the material as a form of reflected waves. One of the ultrasonic NDT techniques is called through-transmission. In this process, as shown in Fig. 1.3 there are two transducers for the transmission and reception of ultrasonic waves set up at the opposite sides of the test sample. A transmitter sends ultrasonic wave energy through one surface whereas a separate receiver detects the amount that has reached it on another surface after traveling through the medium. Imperfections or any change in the space between the transmitter and receiver reduce the amount of sound or energy transmitted, and reveal their presence. However, if the joint is intact, most of the wave energy will go through the surface and be caught by the receiver signifying the absence of any discontinuities. In Fig. 1.3 ultrasound signal passes through the sample where it is undamaged, but signal scatters and lose strength when it hits a crack. Usually a couplant like water is used in the two surfaces of sample to reduce acoustic impedance mismatch between two mediums and facilitate the flow of ultrasound wave energy.

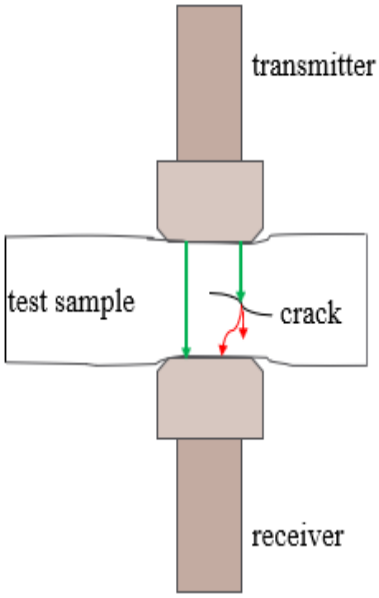


Figure 1.3: Detecting crack in test sample using ultrasonic through-transmission.

1.1.3 Weld Quality Inspection

Automotive industries started using industrial robots as means of automated spot welders in 1980. Since then the number of spot welding robots in industries has been increased extensively. According to a survey, in 2005 more than 120,000 robots were used in North Americans industries, where about half of them were used in spot welding [7]. In spite of this long history of spot welding being automated, the inspection of weld quality still requires human discretion. In spot welding, quality inspection is highly critical in order to prevent severe hazards. While welding, too little energy will not melt the metal enough and make poor weld resulting the joints falling apart. On the other hand, too much energy will melt the metal too much, resulting in making a hole instead of the nugget.

For decades, industries have relied on destructive methods as opposed to non-destructive ones for weld quality assurance. Some of the destructive techniques are peel test, chisel test, metallographic testing etc. Not only these techniques are expen-

sive and time-consuming, they require to selectively inspect only a small randomly chosen portion from a sample pool of thousands of welds. As a result, a large portion of the welds remain unverified. Later on, different nondestructive inspection techniques have been introduced to reduce the time and cost and increase the percentage of validated welds. At present, use of NDT in resistance spot weld quality inspection is a subject of high interest in automotive and other sheet metal assembly industries [8]. Although non-destructive testing methods have significant advantages over destructive methods, a large fraction of welds still get passed without verification. In addition to that, the process is still quite time consuming due to the fact that the test samples must be removed from the production line or at least production needs to be paused for inspection.

To overcome this limitation, many efforts have been made to develop a real-time spot weld quality assurance system which can be installed in the production line. One of the most successful attempts is initiated by the research team in the Institute of Diagnostic Imaging and Research (IDIR) led to the installation of half a dozen inline ultrasonic weld quality inspection prototypes all around the world [41]. In the inline ultrasound setup, they have installed piezoelectric transducer in the cooling water stream inside the welding copper electrodes as shown in Fig. 1.4.

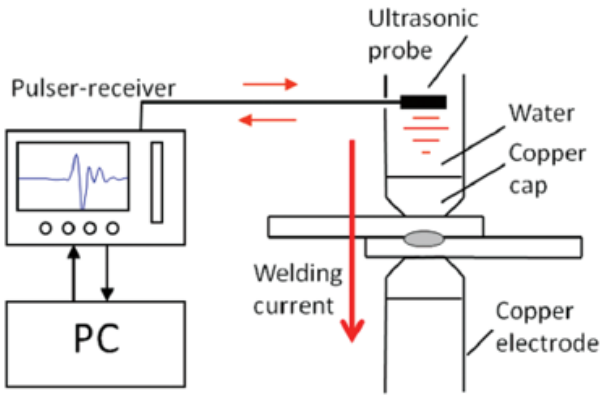
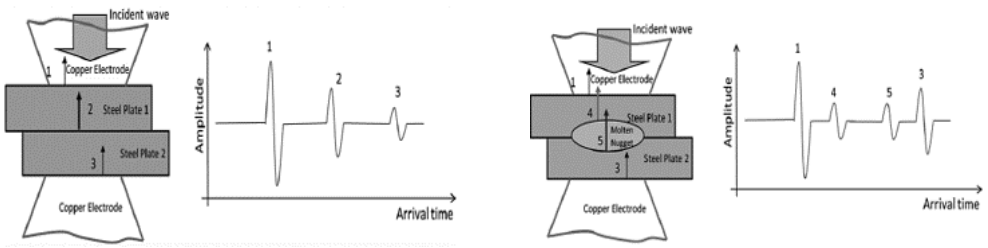


Figure 1.4: Schematic set-up of ultrasonic in-process spot weld analyzer [8].



(a) before welding (b) during welding

Figure 1.5: Two electrodes squeeze steel plates (left) and Gated A-scan (right) [8].

Sound waves propagate through the cooling water and copper electrode to reach the welded plates. Cooling water is used as a couplant to deliver sound from the transducer to the copper electrode cap. The dry contact between the electrode and metal sheets allows ultrasonic waves to penetrate further due to the high pressure exerted by the electrodes. The sound experiences partial reflections at every boundary, including the solid-liquid boundary of the molten nugget. In Fig. 1.5(a), sound waves are reflected from 1, 2 and 3 (copper-steel, steel-steel and steel-copper boundaries respectively) which are represented in the corresponding A-scan as peaks. Similarly, as shown in Fig. 1.5(b), after current is turned on and metal sheets start melting, reflections are visible in the A-scans from 4 and 5 (solid-liquid and liquid-solid boundaries respectively). Due to the high impedance mismatch at the liquid nuggets boundary (i.e. 4 and 5), enough sound energy is reflected which can be detected reliably [41].

The transducers on both side collect information from every reflecting boundary on the wave path and forms properly time-gated A-scans. Each A-scan is a plot of time vs voltage acquired by the receiving transducer. Multiple A-scans of the same point on the weld process captured successively in time with time interval of 3 ms are stacked vertically to form a 2D grey-scale image called b-scan. Fig. 1.6 presents a real ultrasonic b-scan of a properly welded spot weld of two plates having 1.2mm of thickness each. Here, current is turned on at 50ms shown by the red vertical line. Yellow line refers to the moment when melting starts and growth of the nugget has

started to become visible. Gradually nugget depth increases and at the middle it reaches the maximum. Then after 200ms current is turned off, as a result molten liquid starts to solidify. At this point we can see nugget depth decreasing and finally meeting to a closing point. Closing of the nugget ensures that molten liquid has become completely solidified and whole weld nugget is formed successfully.

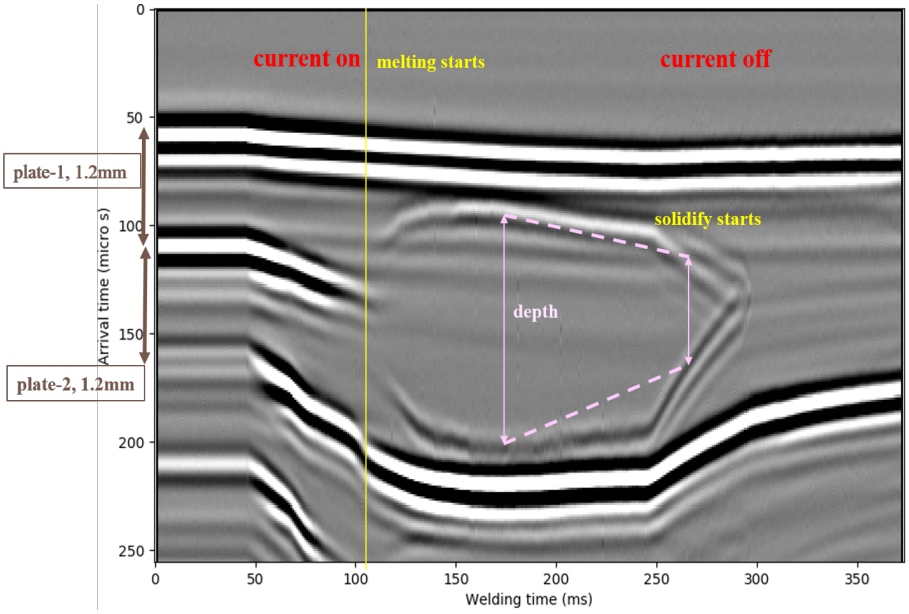


Figure 1.6: Original ultrasonic b-scan.

In [8], authors have reported that spot weld quality inspection is done using different pattern recognition methods on the b-scan to extract important features which are crucial to determine weld quality. For instance, as shown in Fig. 1.7, d2 refers to the degree of liquid penetration or the nugget height, d1 and d3 are the distance of weld nugget from top and bottom plate respectively. If nugget height is bigger than a pre-defined penetration threshold, the weld quality can be considered as acceptable.

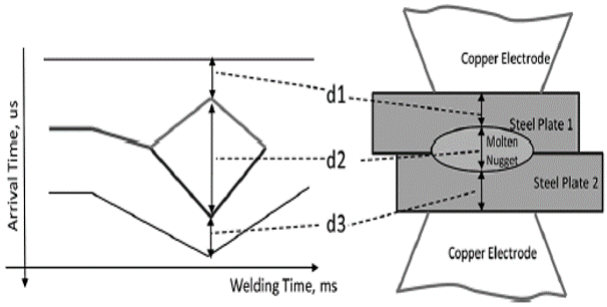


Figure 1.7: Relating an ultrasonic B-scan to features [8].

1.1.4 Deep learning in Computer Vision

Computer vision is a quintessential field of computer science that defines the ability of the computer or any machine to receive and identify an image and perceive the knowledge portrayed by the elements present in the image, and then make analytic decision accordingly. It is close to artificial intelligence because human intelligence and instincts are being conferred to a computer .

The accessibility to Web 2.0 or World Wide Web containing user-generated contents has paved the way to generate visual data in abundance. According to a survey from photo-printing service Photoworld, it would take a person 10 entire years to merely look at all the photos shared on social media sharing site Snapchat in just the last hour. At the same rate, in those 10 years, another 880,000 years worth of photos would have been already been spawned. As it is already well past human capabilities to keep up, computers must take part in making sense of all these abundance of images. As a result, computer vision has become ubiquitous in our society, allowing computer-controlled systems used in factories and farms to autonomous cars and drones, to run more efficiently, intelligently and safely. Core to many of these applications are visual recognition tasks namely image classification, object localization and object detection. While this seems natural to humans, those tasks are difficult due to the large number of objects in the world, the continuous set of viewpoints from

which they can be viewed, the lighting in scene, color variations, background clutter, or occlusion [6].

To a computer, an image is nothing but a huge array of matrices or tensors of pixels, where each pixel is usually a value between 0 and 255, although the representation of pixels may vary. Computers read the numeric pixel values and try to associate them to the objects present in a particular image. There are several image processing and signal processing algorithms like SIFT and SURF that extract features like edges and contours from images. These features are the key descriptors of images, which can be passed to other machine learning algorithms for detection or classification task.

On the other hand, deep learning, which is a sub-domain of machine learning, simplifies the process of feature extraction through the process of convolution where the filter weights are learned from data as opposed to user defined rules. Convolution is a mathematical operation, which maps out an energy function to the input function in order to measure the similarity between two signals, or images [12]. Deep learning algorithms are data-driven which gives them the flexibility to fit more accurately to the data or images they have been passed to. Therefore, given large amount of data and huge processing power, these algorithms have been proved to outperform humans in domain-specific tasks like image classification, object detection, object localization, object tracking, pose estimation, image segmentation, image captioning and so on. Since a breakthrough in speech and natural language processing in 2011, and in image classification in the scientific competition ILSVRC in 2012 [34], deep learning has conquered many fields and won challenges leaving the conventional image and signal processing methods far behind. Therefore, it seems justified to apply deep learning in solving computer vision problems where conventional methods fail to provide satisfactory result.

1.2 Thesis Motivation

In inline real-time automated weld quality inspection, the decision whether the weld is acceptable or not is made right after the weld is made and before the robot moves to the next weld position. This allows the implementation of real-time feedback to the robot to either stop the line or perform re-welding of the unacceptable weld. Establishment of such system is highly desired by industries as it saves them tons of time and money and make the inspection process less error-prone. However, making sense of the weld phenomena captured by ultrasonic b-scan is the most crucial part in this type of system. For computers, this is a highly challenging task due to the large variety of weld shapes of different aspect ratio, shape and size. The pattern recognition software must learn right patterns from b-scans irrespective of elevated noise level, partially missing patterns, distorted patterns disproportional scaling etc. Currently, traditional image and signal processing algorithms are being applied to fulfill this task. However, they have failed to provide industry-level satisfactory result in terms of accuracy. Therefore, in this thesis, we have applied deep learning on ultrasonic b-scan of spot welds to perform object detection in order to detect important components of the weld shape which are related to weld quality inspection. To the best of our knowledge this is the first work in the direction of applying deep learning in analyzing the quality of any kind of weld.

1.3 Problem Statement

In this thesis, we attempt the following

- Apply a deep convolutional neural network based framework for in-line detection of components (or objects) of weld shape from ultrasonic b-scans.

- Interpret detected components to numeric features and classify welds as good, acceptable or bad in real-time during production.
- Improve existing automated system to produce high weld quality classification accuracy that matches with production-level satisfaction.
- Demonstrate how the result obtained from deep learning model can be interpreted to understand the physical phenomena taking place inside materials during welding.

1.4 Thesis Organization

Chapter 2 describes some background of spot welding, the underlying physics behind the process and then change its gear towards deep learning and object detection task which we will be applying in our proposed model. Chapter 3 includes literature review of some existing works both in weld quality inspection and deep learning in computer vision. Chapter 4 describes our proposed methodology in detail. Then chapter 5 discusses about the experiments we have conducted along with their outcome. Then in the following chapter we provide a comprehensive result analysis and some insight we get out of the experiments. Finally, chapter 7 discusses some future direction and draws conclusion to our work.

Chapter 2

Background

2.1 Building Blocks of a Neural Network

Although the original intention of designing a neural network was to simulate the human brain system to solve general learning problems in a principled way, in practice neural network algorithms work in much simpler way than a human brain. Current neural networks can be compared to statistical models having the higher level purpose of mapping inputs to their corresponding outputs. By the means of lots of complex matrix computations going on under the hood, they try to find correlations by approximating an unknown function $f(x) = y$ between any input x and any output y. In the process of learning, they drive to discover the most accurate function f which is also the best approximation of transforming x into y. These networks can be both linear or non-linear. Non-linear networks can approximate any function with an arbitrary amount of error. For this reason, they are also called "universal approximator". In this section, the building blocks of a non-linear shallow neural network followed by a deep neural network is described.

2.1.1 A Neuron

An artificial neuron (derived from the basic computational unit of the brain called neuron), also referred to as a perceptron is a mathematical function that takes one or more inputs and returns their weighted sum as output. Weighted sum is passed through an activation function to transform the otherwise linear function to a non-linear function. This is the reason why neural networks perform well in deriving very complex non-linear functions. Basically, weights are values associated with each input and are parameters to be learned by the network from input data in order to accurately map them to desired output.

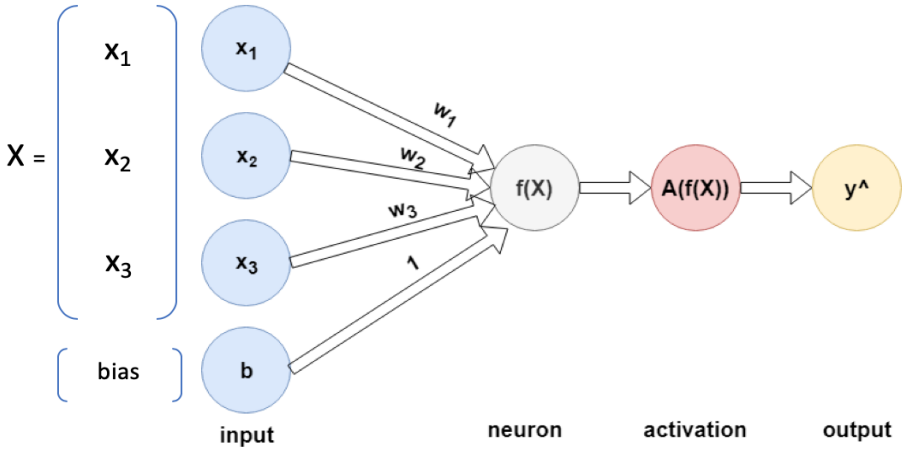


Figure 2.1: Illustration of a neuron as a function.

In Fig. 2.1, input X is a vector containing three elements x_1, x_2 and x_3. Each element have weights w_1, w_2 and w_3 associated to them respectively. In addition to weights, there is also a bias term b which helps the function better fit the data. The neuron *f(x)* calculates the weighted sum using Equation (2.1) [18]. *f(x)* is also defined by z interchangeably.

$$f(x) = z = w_1x_1 + w_2x_2 + w_3x_3 + b \tag{2.1}$$

The resultant output is then passed to an activation function *A(f(x))* which returns

the final output of the neuron.

2.1.2 Activation Function

An activation function is a function applied on a neuron to instill some kind of non-linear properties in the network. A relationship is considered as linear if a change in a variable initiates a constant change in the consecutive latter variables. On the other hand, a non-linear relationship between variables is established when a change in one variable does not necessarily ignites a chain of constant changes in the consecutive latter variables, although they can still impact each other in an unpredictable or irregular manner. Without the injection of some non-linearity, a large chain of linear algebraic equation will eventually collapse into a single equation after simplification. Therefore, the significant capacity of the neural network to approximate any convex and non-convex function is directly the result of the non-linear activation functions. In a small visual example in Fig. 2.2, we can see that when data points make non-linear patterns, a linear line can not fit them accurately, thus it misses some data points. However, a non-linear function captures the difficult pattern efficiently and fits all data points.

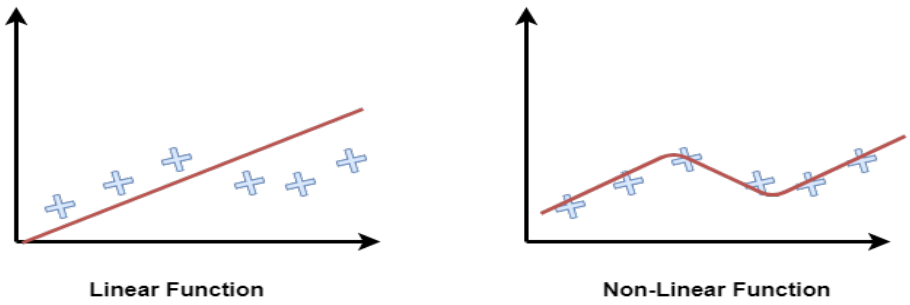


Figure 2.2: Best fit linear and non-linear models.

Every activation function takes the output of *f(x)* as a vector as input and performs a pre-defined element-wise operation on it. Among many activation functions, three of them are most commonly used.

Sigmoid

The sigmoid non-linearity function has the following mathematical form [20],

$$sigmoid(f(x)) = \frac{1}{1 + e^{-f(x)}} \tag{2.2}$$

It takes a real value as input and squashes it between 0 and 1 so that the range does not become too large. However, for large values of neuron activation at either of the positive or negative *x-axis*, the network gets saturated and the gradients at these regions get very close to zero (i.e. no significant weight update in these regions), causing "vanishing gradient" problem [5]. Fig. 2.3 shows the sigmoid function graph that converges in the positive and negative domain as values get bigger.

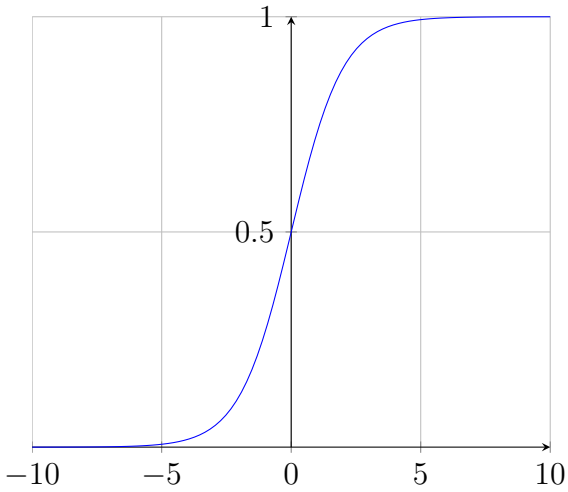


Figure 2.3: A sigmoid activation function.

Hyperbolic Tangent

The hyperbolic tangent or *tanh* non-linearity function has the following mathematical form [53],

$$tanh(f(x)) = \frac{e^{f(x)} - e^{-f(x)}}{e^{f(x)} + e^{-f(x)}} \tag{2.3}$$

It takes a real value as input and squashes it between -1 and 1. However, it suffers from the "vanishing gradient" problem in the positive and negative domain like sigmoid function for similar reason. Fig. 2.4 shows the graph for hyperbolic tangent function.

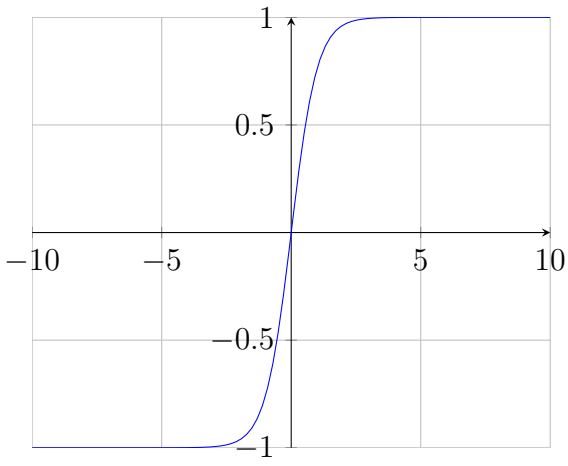


Figure 2.4: A tanh activation function.

Rectified Linear Unit (ReLU)

The ReLU has the following mathematical form [37],

$$ReLU(f(x)) = max(0, f(x)) \tag{2.4}$$

ReLU has gained huge popularity as an activation function due to its edge over sigmoid and tanh in couple of ways. It takes a real value as input and squashes it between 0 and +infinity. Because it does not saturate in the positive domain, it avoids the vanishing gradient problem. As a result, it also accelerates the convergence of stochastic gradient descent. Another big advantage of ReLU is that it involves computationally cheaper operations compared to the expensive exponentials in sigmoid and tanh. However, ReLU saturates in the negative domain which allows it to disregard all the negative values. Thus it may not be suitable to capture patterns in all different datasets and architectures. One solution to this problem is LeakyReLU. Instead of zeroing out the negative activation, LeakyReLU applies a small negative

slope of 0.01 or so. Therefore, LeakyReLU has the following mathematical form [37], where α is the slope.

$$LeakyReLU(f(x)) = \begin{cases} max(0, x) & for \quad f(x) > 0 \\ \alpha x & for \quad f(x) < 0 \end{cases} \tag{2.5}$$

Fig. 2.5 shows ReLU graph where slope keeps increasing only in the positive domain.

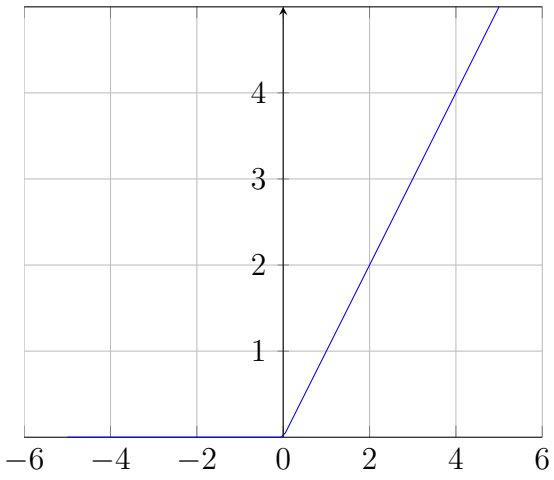


Figure 2.5: A ReLU activation function.

Fig. 2.6 shows LeakyReLU graph where slope keeps increasing in the positive domain but also allows gradients to flow by a restricted amount in the negative domain. Here, α is 0.1.

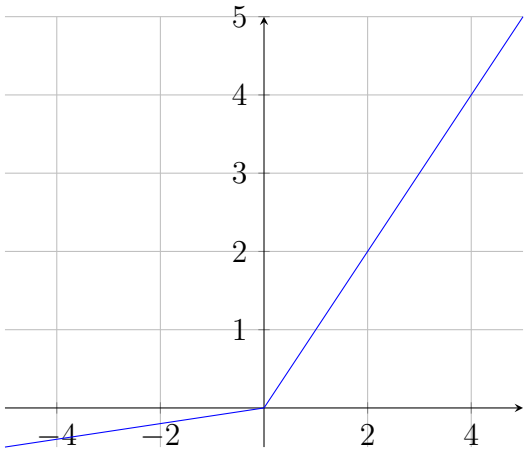
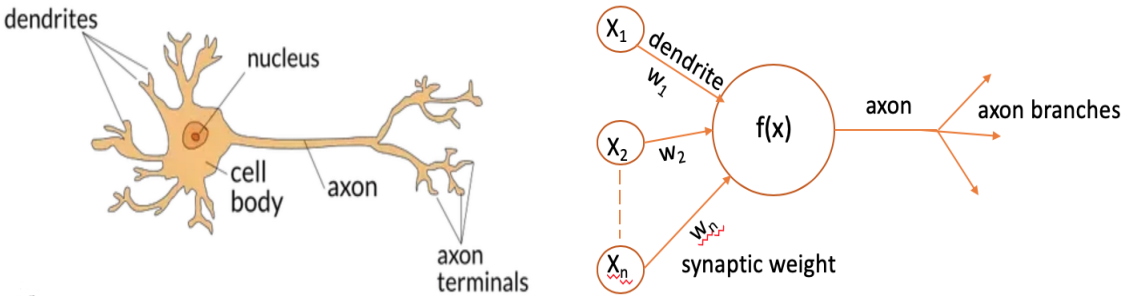


Figure 2.6: A LeakyReLU activation function.

2.1.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are motivated by the nervous system of the human brain where approximately 86 billion neurons are interconnected with 10^{14} to 10^{15} synapses [22]. In human nervous system, each neuron receives input signals from its dendrites and produces output signal along its axon. Comparatively, ANNs are a way smaller and simplified computing model of the nervous system with tens to as many as thousands of neurons interconnected as processing elements, which process information by their dynamic state response triggered by external inputs (i.e. user given data). As seen in Fig. 2.7, just like a biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons through axon branches, the artificial neuron has a number of input channels, a processing stage called *f(x)*, and one output analogous to axon branches that can fan out to multiple other artificial neurons. ANNs are comprised of stacks of layers, where each layer contains a pre-defined number of neurons just like the one in the figure. Neurons from each layer are connected by edges which are similar to synapses. Axons are analogous to the output obtained from a single layer.



(a) human neuron (b) ANN neuron

Figure 2.7: A human neuron vs an artificial neuron.

Any shallow neural network has at least three layers, namely as an input layer, hidden layer(s) and an output layer. Hidden layers are intermediate layers between input and output layer. In Fig. 2.8, the network has 7 neurons altogether. These neurons are stacked in 3 fully connected layers, that is each neuron from one layer is connected to each neuron from the following layer. The number of layers in the network, the number of neurons in each layer, how they are connected (as example, fully-connected or locally connected) - all these are usually pre-defined while designing the network architecture. First layer, consisting of 3 neurons, is the input layer where these neurons do not participate in any kind of calculation, rather they just propagate the input to the next hidden layer with their associated weights and bias. For simplicity, in this network the bias term is omitted. The hidden layer calculates the weighted sum from each neuron and passes the result as a 3×3 matrix to the ReLU activation function. The output of ReLU is used as input to the output layer. In the output layer, sigmoid activation function is used to transform the resultant matrix into a categorical or numerical value which is the final output of the network.

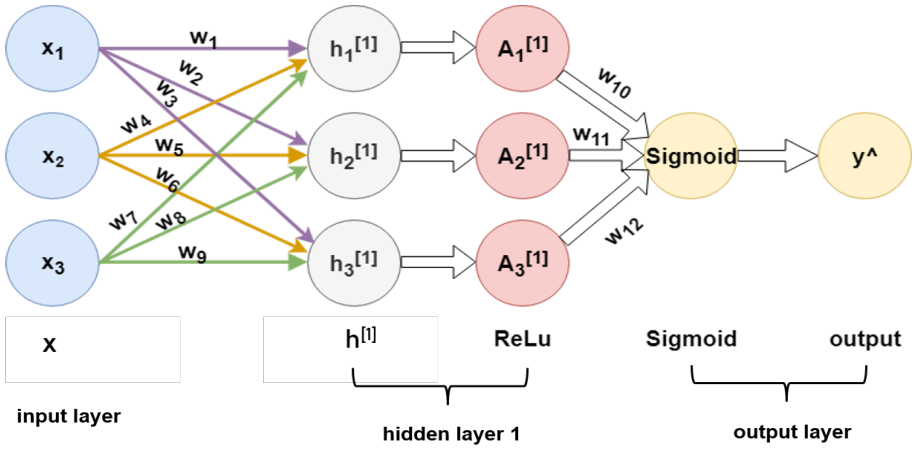


Figure 2.8: Diagram of a shallow ANN.

In Fig. 2.8, ${h_1}^{[1]}$, ${h_2}^{[1]}$ and ${h_3}^{[1]}$ are the neurons of hidden layer 1. From the network, they are calculated individually from the weighted sum of x_1, x_2 and x_3 as following,

$$\begin{aligned} h_1^{[1]} &= w_1 \cdot x_1 + w_4 \cdot x_2 + w_7 \cdot x_3 \\ h_2^{[1]} &= w_2 \cdot x_1 + w_5 \cdot x_2 + w_8 \cdot x_3 \\ h_3^{[1]} &= w_3 \cdot x_1 + w_6 \cdot x_2 + w_9 \cdot x_3 \end{aligned} \tag{2.6}$$

Now $h_1^{[1]}$, $h_2^{[1]}$ and $h_3^{[1]}$ are passed to corresponding ReLU activation function ${A_1}^{[1]}$, ${A_2}^{[1]}$ and ${A_3}^{[1]}$ for non-linear transformation. The weighed sum of the activations generates the final output $\hat{y}$. w_{10}, w_{11} and w_{12} are weights associated to ${A_1}^{[1]}$, ${A_2}^{[1]}$ and ${A_3}^{[1]}$ respectively. Therefore, the function for calculating $\hat{y}$ is defined in Equation (2.7) [18].

$$\hat{y} = f(x_1, x_2, x_3) = Sigmoid(w_{10} * ReLU({h_1}^{[1]}) + (w_{11} * ReLU({h_2}^{[1]}) + (w_{12} * ReLU({h_3}^{[1]})) \tag{2.7}$$

2.1.4 Deep Neural Networks

Deep neural networks (DNNs) are basically ANNs with many hidden layers ranges from 3 to 152 (ResNet [21]). The more layers stacked up in between the input and output layer, the deeper the network is considered. The deeper the network, the more complex representation from the data can be learned. The hidden layers learn data representation in hierarchical order, which means that the output of a hidden layer is passed as input to its next consecutive hidden layer and so on.

Many recent breakthrough outcomes are obtained by using very deep models. However, stacking up layers after layers does not always guarantee better model accuracy. In [21], authors have reported that with network depth increasing, the accuracy gets saturated after one point. Also, deeper models require to compute larger number of parameters which makes them computationally expensive and slower. Fig. 2.9 shows a deep neural network with n hidden layers.

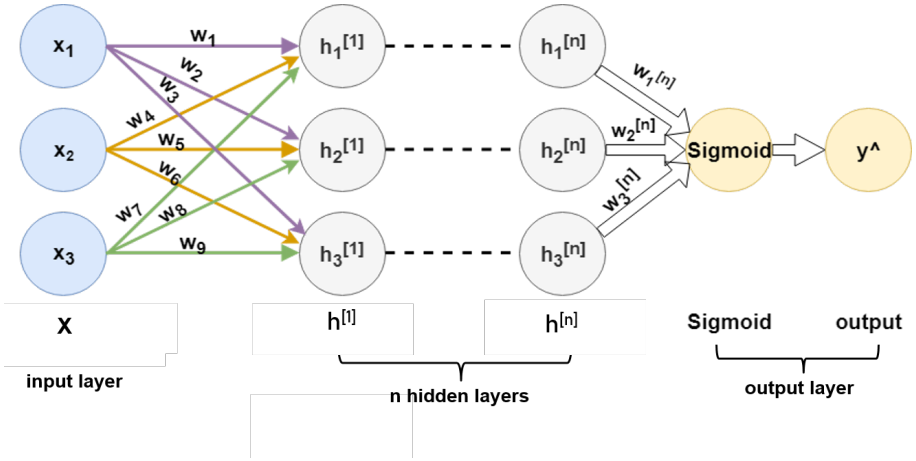


Figure 2.9: Diagram of a Deep NN.

2.2 Training a Neural Network

Neural networks are used for three main categories of learning, namely as supervised, unsupervised and reinforcement learning. In this thesis, neural network has been used to solve a supervised learning problem, where the input is spot weld b-scans along with corresponding annotations and target is prediction of the annotations. Training of a neural network involves the combination of several individual algorithms and helper components like a base architecture, optimization algorithm called gradient descent, convolution, pooling, softmax classifier etc. In this chapter, three of the most important components are explained.

2.2.1 Loss function

The loss function, also called a cost function, is a measure of quantifying the capacity of a network to approximate the true function that produces the desired output from given data. In supervised learning, desired output is the ground truth labels or classes $\hat{y}$ of each sample in the dataset. In a single forward pass, the network takes as inputs the weights, biases and examples from the training set and predicts the labels $\hat{y}$ for each sample. The difference between the predicted labels and the ground truth labels are calculated which is the total loss of the network. Although there are many loss functions that exist, all of them essentially penalize on the difference or error between the predicted $\hat{y}$ for a given sample and its actual label y. Mean squared error or MSE is a commonly used loss function. For m number of samples in the training set, the equation for calculating total loss over weight parameters, $J(w_1, ...w_n)$ using MSE is shown in Equation (2.8) [66].

$$Loss(y, \hat{y}) = J(w_1, ...w_n) = \frac{1}{m} \sum_{i=1}^{m} (y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2 \tag{2.8}$$

2.2.2 Gradient Descent

The purpose of training a neural net is to minimize the loss by adjusting the parameters, like weights, biases etc. The smaller the loss, the more accurately the true function is approximated. Gradient descent algorithms are the most commonly used methods for loss function optimization. At the beginning of the training, network parameters are randomly initialized. Therefore, the loss is very high. Gradient descent algorithms calculate the first order derivative of each of the parameters with respect to the loss function and directs in which direction the parameter values have to be adjusted in order to reduce the loss. Eventually, it finds the global minimum of the loss function curve which is the point at which the loss is minimum. All model parameters including weights are defined as θ. Therefore, for each θ_j, gradient descent algorithms find the derivative of each θ_j for each input vector X_0 to X_n with respect to total loss $J(\theta_{j0}, ...\theta_{jn})$ and update each parameter using Equation (2.9) [43].

$$\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_{j0}, ...\Theta_{jn}) \tag{2.9}$$

Here, α is the learning rate that defines the step size in the direction the parameters are adjusted to. The algorithm iteratively performs this operation until either the global minimum is reached or the model converges.

The feature space of the loss function can be convex or even highly non-convex with tens of millions of parameters depending on model complexity. Figure 2.10 and 2.11 show examples of gradient descent in both convex and non-convex loss surface having only two parameters Θ_0 and Θ_1. The black line shows the gradient started from an initial point flowing all the way down to the global minimum on convex and local minimum on non-convex surface.

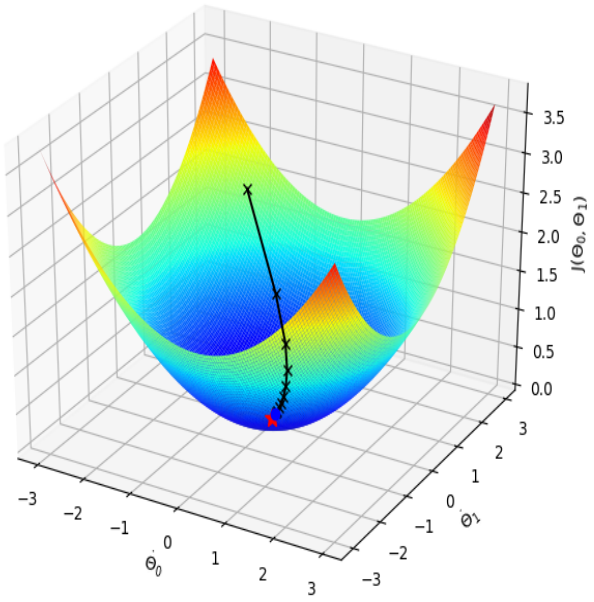


Figure 2.10: Gradient descent on convex surface.

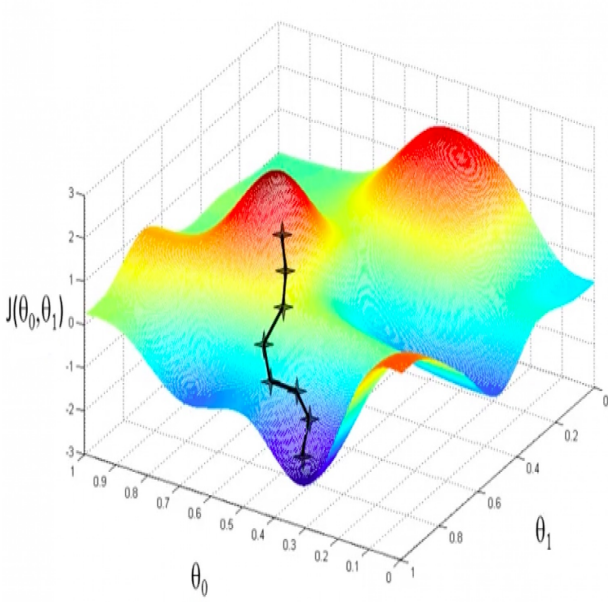


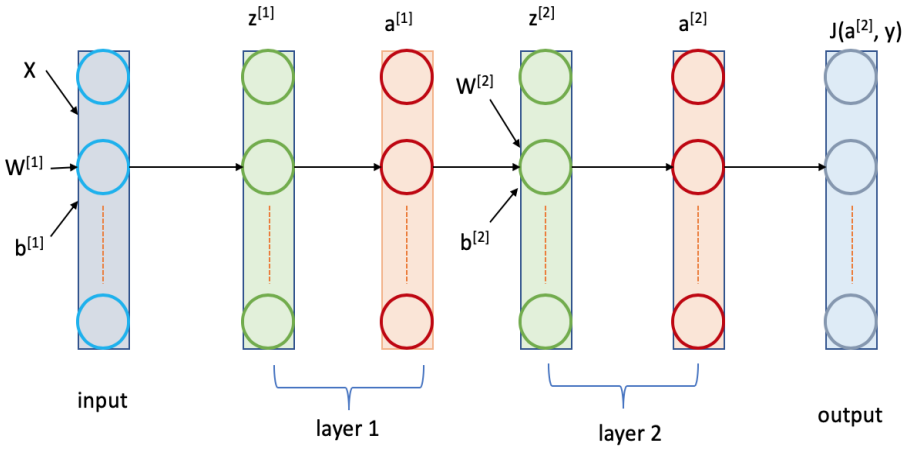
Figure 2.11: Gradient descent on non-convex surface (Source: Andrew Ng).

2.2.3 Backpropagation

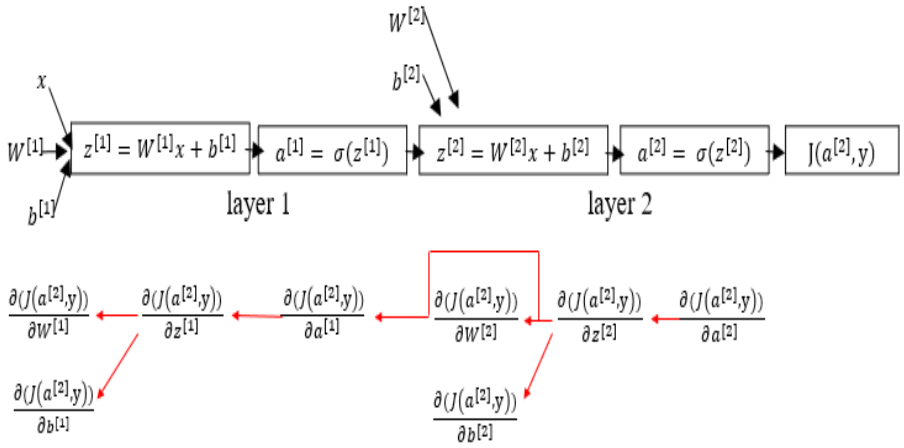
Calculating the derivatives of millions of parameters with respect to total loss and find the optimal point in a high-dimensional convex or non-convex surface is analogous to finding a needle in a haystack. Accomplishment of this task using the naive approach

is extremely unfeasible. That's where backpropagation comes to the rescue. Backpropagation is a fast way of calculating all the parameter gradients in a single forward and backward pass through the network rather than having to do one single pass for every single parameter. It utilizes the chain rule of partial derivatives to compute gradients on a continuously differentiable multivariate function in a neural network. The chain rule enables gradient descent algorithms to decompose a parameter derivative as a product of its individual functional components. Use of backpropagation allows a backward pass take roughly the same amount of computation as a forward pass and makes doing gradient descent on deep neural networks a feasible problem by speeding up the training dramatically.

Fig. 2.12(b) illustrates backpropagation in action in a neural network with two hidden layers *layer 1* and *layer 2* as shown in Fig. 2.12(a). After a forward pass (shown with black arrow) the loss function $J(a^{[2]}, y)$ is evaluated. Then, in the backward pass (shown in red arrow) the partial derivative of each of the four parameters $W^{[1]}$, $b^{[1]}$, $W^{[2]}$ and $b^{[2]}$ with respect to $J(a^{[2]}, y)$ is calculated using chain rule. As example, to calculate the partial derivative of $W^{[1]}$, we have to first partially derive $z^{[1]}$ (weighted sum function in layer 1) with respect to $J(a^{[2]}, y)$ as $\frac{\partial J(a^{[2]},y)}{\partial z^{[1]}}$. Again, in order to find $\frac{\partial J(a^{[2]},y)}{\partial z^{[1]}}$, $\frac{\partial J(a^{[2]},y)}{\partial a^{[1]}}$ has to be calculated where $a^{[1]}$ is the layer 1 activation function. Consequently, the derivative of $a^{[1]}$ comes from the derivative of functions $z^{[2]}$ and $a^{[2]}$ in layer 2. In this way, backpropagation propagates the total loss backward all the way down to each parameters and adjust them accordingly using gradient descent. Therefore, it gives a detailed insight into how changing the weights and biases changes the overall behaviour of the network a.k.a the change in total network loss [45].



(a) neural net with two hidden layers



(b) backpropagation in each layer

Figure 2.12: Backpropagation in action.

2.3 Convolutional Neural Networks

Convolutional neural networks or CNNs are deep neural networks considered as the most representative model of deep learning [69]. CNNs extend on traditional neural networks by combining both fully-connected hidden layers and locally-connected convolutional layers. In traditional neural networks, each neuron in the hidden layer is fully-connected to all neurons in the preceding layer. However, in case of large 2D input images full-connectivity gets highly computationally expensive. It also does

not take advantage of the local correlations of pixels in real-world images. Inspired by the biology of human vision, CNNs take advantage of the spatial correlation of objects present in real-world images and restrict the connectivity of each node to a local area known as its receptive field [13]. In general, CNNs use four key ideas that greatly reduce the number of parameters to be learned and improve the approximation accuracy compared to traditional DNNs. These are: local connections, shared weights, pooling and the efficient use of many layers [69].

2.3.1 Basic Architecture

A typical CNN architecture is demonstrated in Fig. 2.13.

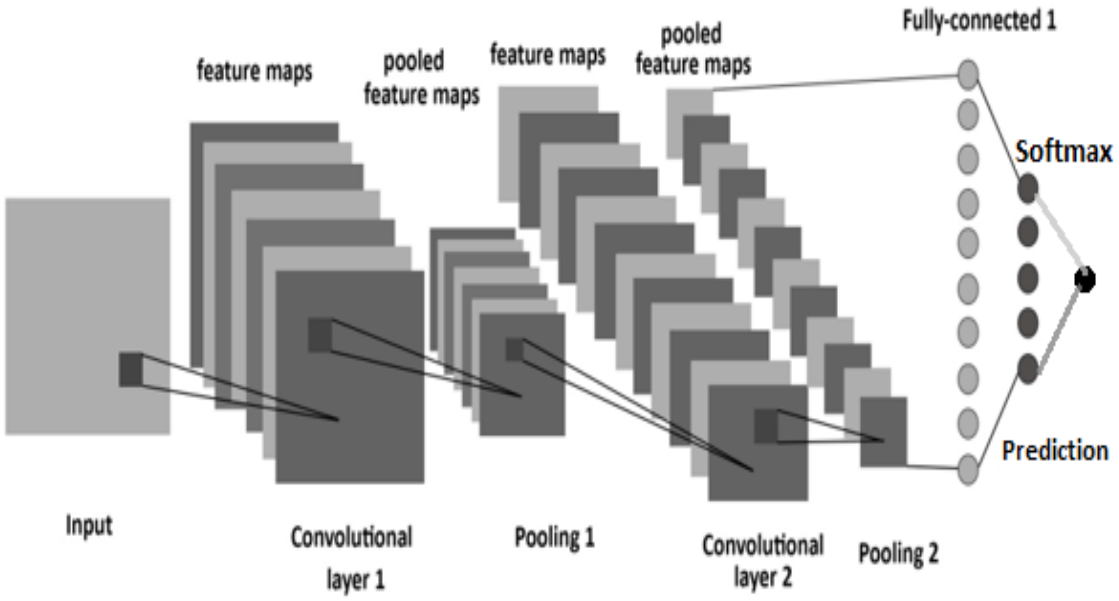
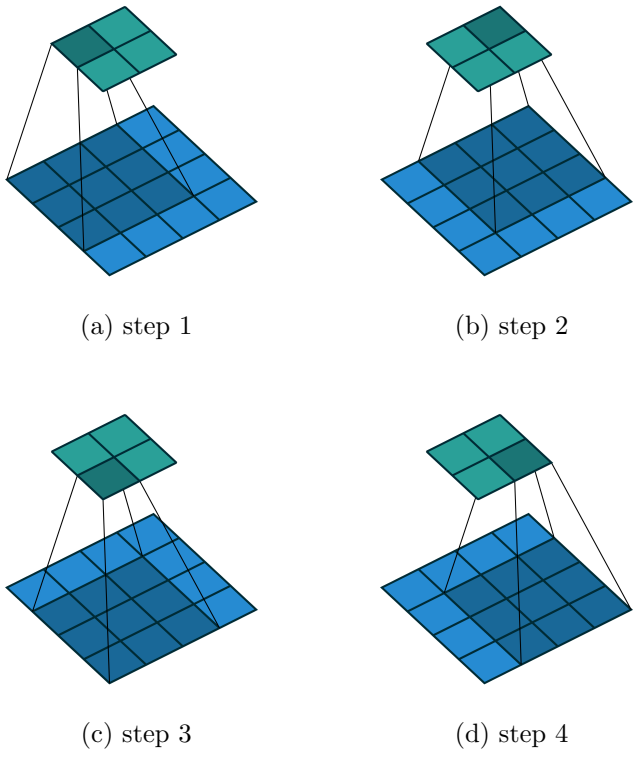


Figure 2.13: Architecture of a typical CNN [47].

The hidden layers in a CNN can be any of the three types from convolutional layer, pooling layer and fully-connected layer. Each neuron in the input layer takes each image pixel as input variables and passes them to a small number of adjacent neurons of the next layer they are connected to. Image region covered by these adjacent neurons is called the receptive field as mentioned earlier. Each neuron in the



(a) step 1 (b) step 2

(c) step 3 (d) step 4

Figure 2.14: Convolution in action [12].

hidden layer takes a receptive field from the previous layer as input and transforms that into many different representations by applying convolution and pooling filters. Filters are 2D matrices of weights which are analogous to the weight parameters of the traditional DNN described in previous section. For each filter, a different representation of the output of previous layer is obtained which is called a feature map. In Fig. 2.13, stacked images in the convolutional and pooling layers are the feature maps or channels or layer depths. Filter weights are shared across a single feature map. On the other hand, different shared filter weights are used for different feature maps. Activation functions are applied to each of the feature maps individually in each layer.

Convolution

Convolutional filters convolute a weight matrix with the values of a receptive field of neurons in order to extract useful features or patterns. These filters learn more high-level features as the network gets deeper. At the beginning of the training, weights are randomly initialized. However, during training process, weights are adjusted through backpropagation in order to learn the features which are most representative for correct prediction.

In Fig. 2.14, an input matrix in light blue is convolved by a filter matrix in dark blue and generates a feature map in green by taking the summation of the element-wise product of each cell of the filter and each cell of the area covered by the filter of the input matrix. In this convolution, no zero-padding [31] on the input matrix and stride [31] of 1 on the filter is used.

Pooling

Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one [69]. In other words, pooling layers compress the responses of a receptive field into a singleton value to produce more robust feature representations. A typical neuron in the pooling layer (MaxPool) takes the maximum of a local patch or region of cells in a feature map resulting in a lower dimensional representation of the particular feature map. As a result, locally connected pooling neurons take features from patches that are already shifted by more than one row or column, thereby creating an invariance to small shifts and distortions to new unseen images.

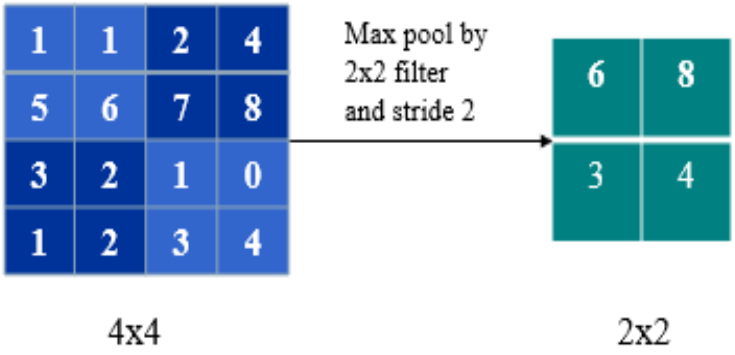


Figure 2.15: Max Pool in action.

In Fig. 2.15, a 4×4 feature map is reduced to a 2×2 map after taking the maximum value from each of the four patches. Pooling filter weights are not learnable parameters thus they don't take part in backpropagation.

2.3.2 CNNs as Feature Extractor

CNNs have the great advantage of very deep feature extraction through hierarchical representation of images from pixel to high-level semantic features [72]. In CNNs, features extracted from one convolution layer followed by a pooling layer are passed as input to next layer along with all the different transformation learned by feature maps. More importantly, CNNs learn this representation automatically from the given data without any domain-expert defined rules. Along with multi-layer non-linear mappings using activations, many hidden pattern of the input data can also be discovered which makes CNNs exponentially expressive compared to traditional computer vision algorithms. Therefore, currently CNNs are the most common model to be used as the base feature extractor in any computer vision tasks including image classification, object detection, face recognition etc [72]. Detailed explanation of CNNs and their mechanism in our work is included in the following chapters.

Chapter 3

Literature Review and Related Work

This chapter is divided into two main parts. First part describes some of the existing works on weld quality analysis using various traditional image and signal processing techniques. The limitations of the existing methods are also discussed. On the other hand, second part demonstrates the literature related to applying deep learning to object detection task along with some existing works of detecting objects from ground penetrating radar (GPR) b-scans using deep learning. All the existing works involving deep learning is about object detection on dataset that are completely unrelated to our weld b-scan data. Since, our work is the first attempt to use deep learning in this special type of data, no similar or related existing literature has been found.

3.1 Weld Quality Analysis with Existing Methods

In [40], Chertov et. al. have used image and signal processing techniques to determine spot weld quality in ultrasonic through-transmission mode which we already explained briefly in Chapter one. In this mode, as shown in Fig. 3.1(a), short longi-

tudinal acoustic pulses are sent through the metal sheets to be welded from one end and are picked up by the receiving transducer in the other end. Once weld current is turned on, weld zone gets heated and as a result the media change their properties and modify the transmitted signal differently at different stages of the welding process. Analyzing this signal collected by receiver, useful features can be extracted that are related to the entire weld phenomena. Authors have attempted to approximate different weld parameters including signal delay, phase, spectrum, amplitude, temperature distribution etc. by solving differential equations using finite difference method (FDM) described in [38]. However, they have reported that among half a dozen of these different parameters of the wave signal, the time of flight or TOF delay is most reliably correlated to the weld nugget diameter. Time of flight is the time a signal takes from being emitted to returning back to the transmitter after being reflected by each interface (i.e. copper-steel, steel-steel, steel-copper etc.) in Fig. 3.1 of the weld zone. TOF delay is occurred because of the acoustic wave velocity decrease with the elevation of heat at different stages and thermal expansion of the heated metals. FDM model helps track the TOF for each signal reflected from different interfaces. It results in measuring the degree of heating during the welding and better prediction of the weld nugget size. Fig. 3.2 shows the correlation between TOF delay and weld diameter. From the figure we can see that TOF delay is directly proportional to weld diameter.

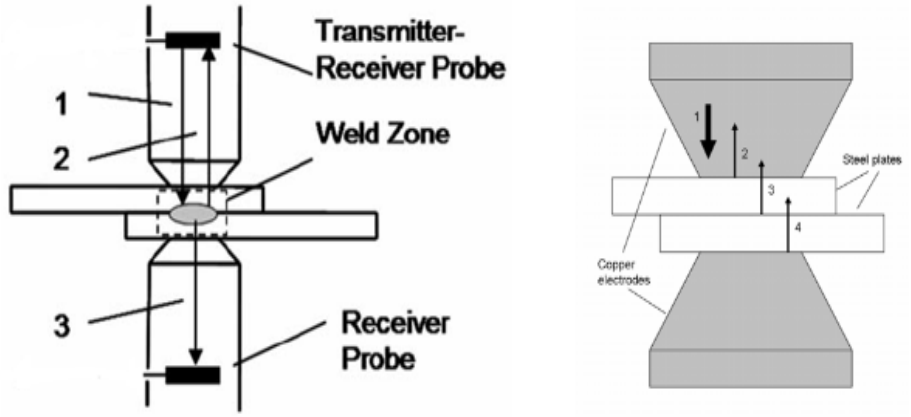


Figure 3.1: Schema setup of (a) Through-transmission (b) Reflection mode [39]

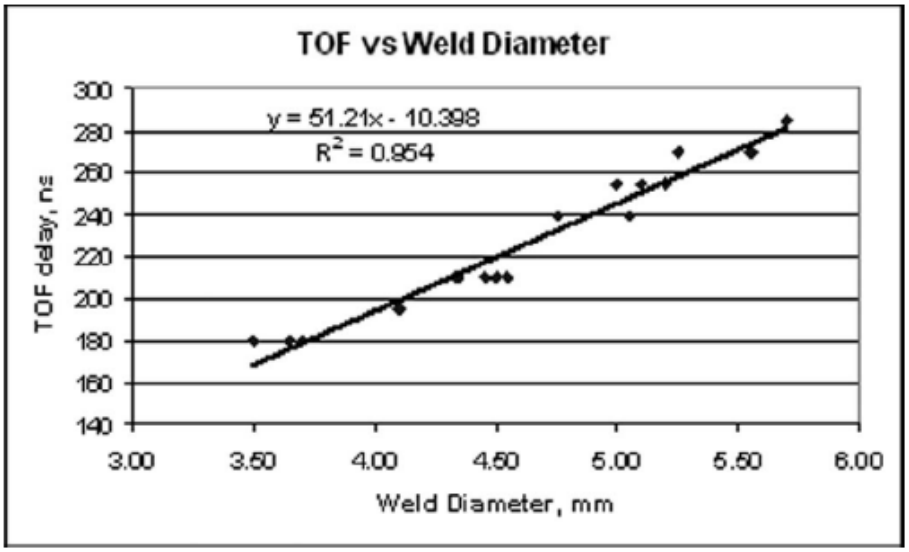


Figure 3.2: TOF delay versus weld diameter [40].

In [39], authors have empirically proved their observation found in [40] by conducting same experiment on ultrasonic reflection mode. In reflection mode, as shown in Fig. 3.1(b) only one transducer is used as opposed two like through-transmission which works as both the transmitter and receiver. Authors have reported that in the reflection mode as long as the wave passes through the weld zone twice the accuracy of measuring, the total TOF delay is higher than that of the through-transmission mode. As a result, the reflection method is capable of providing more accurate results in measuring the degree of heating of the welding process and thus the better

prediction of the weld nugget size.

However, most of the real weld scans are very noisy, having signal-to-noise ratio (SNR) close to 1 which creates challenges in determining dynamically changing positions of the reflected signals. To overcome this problem, different types of signal pre-processing techniques are applied. In [1], authors have taken an approach of subdividing the interfaces as straight line segments *2*, *3* and *4* as shown in Fig. 3.3 using Radon Transform. In this method, an image is smoothly rotated by the center and its normalized projection is measured with respect to the horizontal axis. The signal of the projection gains maximum peak when a line segment becomes perpendicular to the horizontal axis. Then the position of the line segment causing maximum peak and the image orientation is tracked and mapped to the original image which enhances low-contrast interface lines having weak signals. Along with Radon Transform, some horizontal and diagonal filters are also applied in order to remove irrelevant signals due to noise.

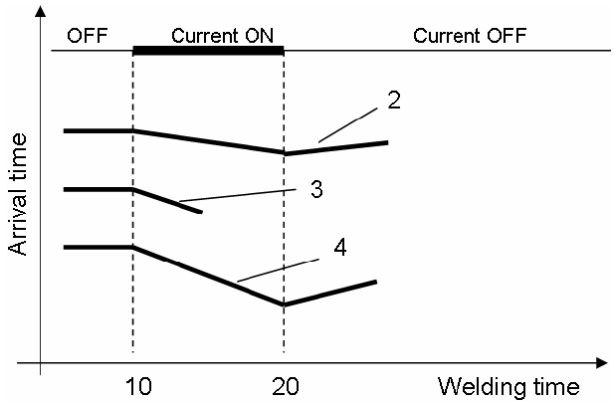


Figure 3.3: Interfaces 2, 3 and 4 represented by straight line segments [1].

Apart from the above mentioned methods, many other image and signal processing techniques are being tried on in order to find the right patterns or features relevant to weld quality. Some of them are Wavelet-based method, Short time Fourier transform, Center frequency phase tracking, sliding Hough Transform etc [2]. How-

ever, none of them have succeeded to match the expected accuracy due to their own limitations. Therefore, to improve the current weld quality control system, we have taken a different approach using deep learning. We have proposed an object detection model capable of finding all desired patterns as objects from weld b-scans with high accuracy. These patterns are highly correlated to the underlying physical phenomena that takes place during welding. Therefore, interpreting them geometrically we can decide the weld quality very reliably.

3.2 Object Detection using Deep Learning

3.2.1 Object Detection Task

Object detection is a supervised learning problem which is a combination of object classification and object localization done for multiple objects present in an image. Given an input image, it draws rectangular bounding boxes around each existing object (localization) and labels each of the boxes with the object label or class (classification) like “cat”, “dog”, “car” etc. Along with labels, it also provides the model confidence, or in other words the probability of the object inside the bounding box being correctly classified. Fig. 3.4 illustrates the difference between classic image classification and object detection problem.

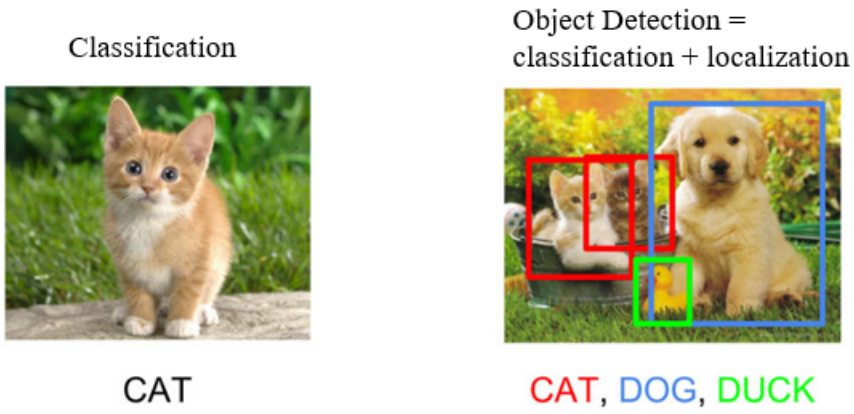


Figure 3.4: Object detection task [26].

3.2.2 Brute Force Approach - Sliding Window Detectors

Since a very deep neural net called AlexNet [34] won the ILSVRC image classification challenge in 2012, CNN became one of the most dominated algorithms in computer vision. At the initial stage, there was a brute force approach used for object detection called Sliding Window Detectors. The steps for the approach includes following,

- Given an input image, take a window of size smaller than the input.
- Slide the window through the image from left to right or top to bottom.
- Cut out patches of the input image from each of the region covered by the window as it slides through.
- Each image patch is fed into a CNN model for feature extraction. The patches are warped to fit a fixed size as CNN takes fixed size inputs only.
- Features extracted by CNN is passed to a Support Vector Machine (SVM) [9] classifier to identify the class of the object present in the patch if any and another linear regressor for providing the boundary box.
- Repeat all previous steps with windows of varied size and aspect ratios in order to detect objects with different sizes due to being captured from various viewing distances.

Although this exhaustive approach can locate all possible positions of the objects, it is computationally very expensive and produces too many redundant windows as candidates. On the contrary, limiting the number of window templates to a fixed boundary will let the algorithm miss or inaccurately locate some of the objects.

3.2.3 Current Approach

The current frameworks for object detection task can be categorized into two main types [72]. One category of frameworks uses a two-step pipeline - first it generates

region proposals using improved algorithms other than sliding windows, then passes the regions to CNN and SVM for feature extraction followed by classification and localization. The other category unifies the region proposal generation and classification and regression task as one multi-task learning problem. The two-step pipeline methods mainly include R-CNN [49], Fast-RCN [17], Faster R-CNN [51], R-FCN [70], FPN [56], SPP-net [30] etc. The unified methods include Multibox [10], YOLO [28], YOLOv2 [50], SSD [61], DSSD [16], DSOD [71] etc. In this work, we have extensively studied Faster R-CNN and SSD as representatives of each of the two types. As the final solution to our problem, we have chosen SSD over Faster R-CNN for it's relatively better speed-vs-accuracy trade-off.

Faster R-CNN

In Faster R-CNN, input image is passes to a deep CNN which outputs multiple feature maps. Regions proposals are generated on these convolutional feature maps instead of the raw input image. Also, Sliding Window and its variant methods are replaced by a Region Proposal Network (RPN) for region proposal generation. RPN is another deep fully-convolutional network which is trained to predict object bounding boxes and objectness score (the probability of an object being present in a location) at each position of the feature map grid simultaneously.

In Fig. 3.5, the complete architecture of Faster R-CNN is illustrated. Given an input image, it is passed to a deep CNN for generating multiple feature maps. The output of the CNN is passed to RPN. RPN takes the output feature maps and slides 3×3 filters over them to make and generate class-agnostic region proposals. The CNN in RPN passes 256 feature values to 2 separate fully connected layers where one predicts a boundary box with 4 coordinates and other one predicts 2 objectness score - one for object and one for background. For each location in the feature maps, RPN

makes k predictions. As a result, it generates $4 \times k$ coordinates and $2 \times k$ scores per location. Thus, for feature maps of shape $h \times w$ with a 4×3 filter, RPN proposes total $h \times w \times k$ regions or bounding boxes, also called Region of Interest (RoI). One important aspect of RPN is that it generates proposals in terms of predicting offsets in coordinates relative to some pre-selected references boxes called anchors or priors. Anchors are carefully pre-selected all over the input image to cover real-life objects of diverse scales and aspect ratios reasonably well. In order to make k predictions, k anchor boxes are created at each location as each prediction box has to be associated with a specific anchor whom it will be adjusted to. This approach leads the initial training with better predictions rather than random or brute-force way of proposals and allows each prediction to specialize in a certain shape.

Next, the proposed RoIs and corresponding feature maps are passed to RoI pooling layer to warp the variable size ROIs into in a predefined size shape to fit the fully connected layers. FC layers take warped RoIs as input and pass them to last two separate FC layers tagged to a softmax classifier for classification and linear regressor for bounding box prediction.

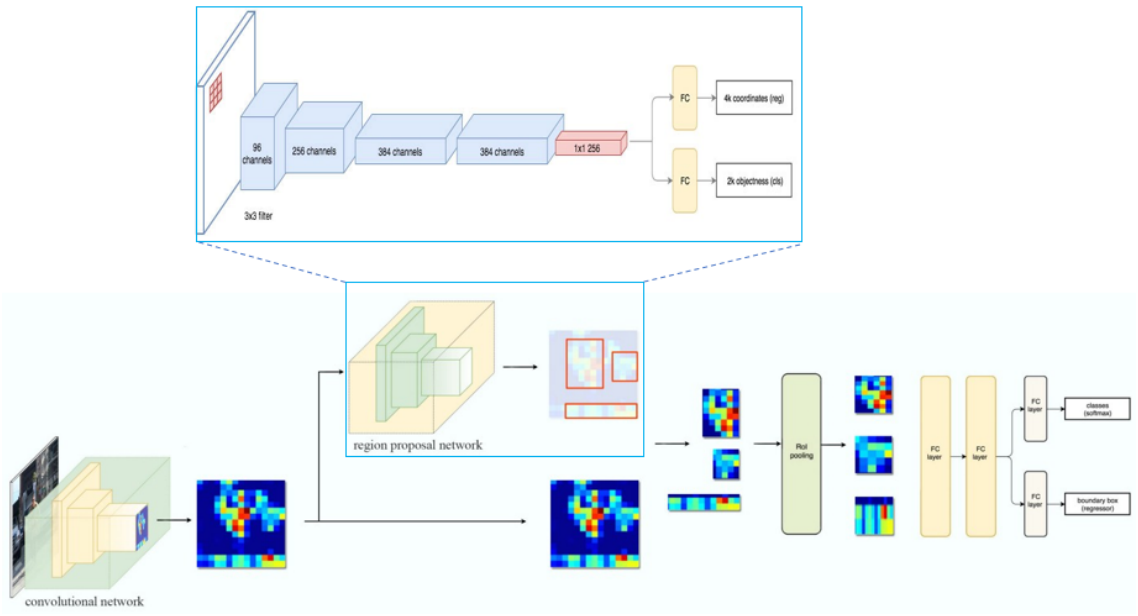


Figure 3.5: Architecture of Faster R-CNN [25].

Although Faster R-CNN incorporates two separate CNNs, it can be trained in an end-to-end way as a multi-task learning problem where useful representation for multiple correlated tasks are learned from the same input. It is also at least 150× faster in inference time than its predecessors [51]. However, an alternate training is still required to obtain shared convolution parameters between RPN and rest of the detection network which is very time-consuming. In addition to that, RPN produces object-like regions (including backgrounds) instead of real object instances and is not skilled in dealing with objects with extreme scales or shapes [25].

Single Shot Multibox Detector (SSD)

SSD is a one-step framework that learns to map a classification-and-regression problem directly from raw image pixels to bounding box coordinates and class probabilities, in single global steps, thus the name "single shot". Fig. 3.6 illustrates the architecture of SSD. It incorporates a deep CNN of 19 convolutional layers as the base feature extractor similar to Faster R-CNN. However, the fully-connected layers from the extractor is discarded and replaced by a set of auxiliary convolutional layers

as shown in blue. These auxiliary layers are responsible for extracting features at multiple scales and progressively decrease the size of the input at each subsequent layer. Instead of generating anchor boxes at each feature map separately like in Faster R-CNN, SSD generates anchor boxes directly on the multi-scale feature maps coming from the base CNN. In [61], authors have used six auxiliary layers for producing feature maps of size 8×8, 6×6, 4×4, 3×3, 2×2 and 1×1 respectively. Due to multiple scaling, generated anchor boxes better cover all the diverse shapes of objects including very small objects and very large ones. Features extracted by each region proposal or anchor from each feature map is simultaneously passed to a box regressor and a classifier. Finally, the network outputs coordinate offsets of predictions relative to anchor boxes, objectness score (here confidence score) of each predicted boxes and probabilities of an object inside a predicted box being each of the C classes. For instance, for feature maps of size $f = m \times n$, b anchors per cell in each of the feature maps and C total classes, SSD will output total $f \times b \times (4 + c)$ values for each of six feature maps. Anchors boxes are generated with different aspect ratios and scales similar to Faster R-CNN, however, due to applying them on multiple scaled feature maps objects with more variety of sizes and scales are detected more accurately.

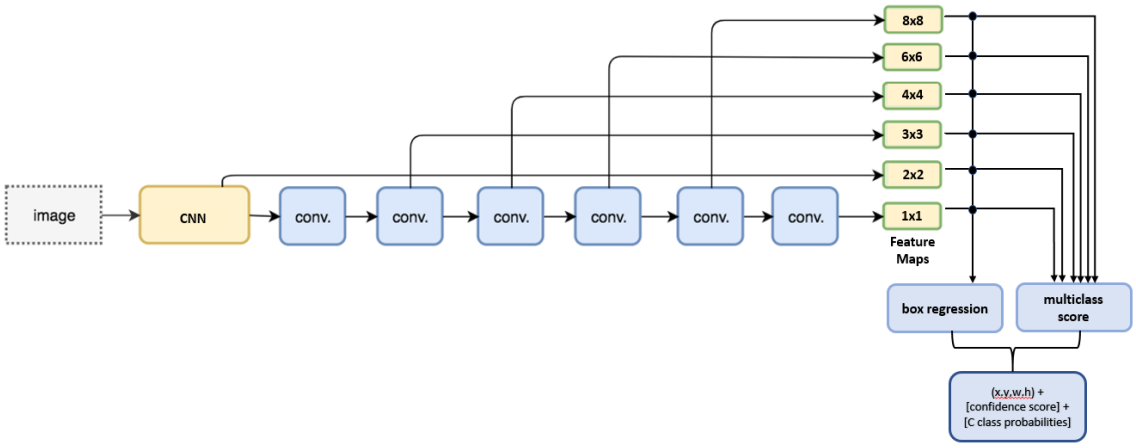


Figure 3.6: Architecture of SSD.

Integrating with some pre and post-processing algorithms like non-maximum sup-

pression and hard negative mining, data augmentation and a larger number of carefully chosen anchor boxes, SSD significantly outperforms the Faster R-CNN in terms of accuracy on standard PASCAL VOC and COCO object detection dataset, while being three times faster [72].

3.2.4 Related Works

There are very few studies have been conducted that have used machine learning or deep learning methods on ultrasonic b-scans. Ours is a contributory addition to this area and to the best of our knowledge, ours is the first work applying deep learning for weld quality analysis, that too from ultrasonic weld b-scans (which have different frequency than other ultrasonic images).

In [35], Besaw et. al. (2015) have used a CNN for detecting Buried Explosive Hazard (BEH) from Ground Penetrating Radar (GPR) b-scans. This is a classic two-class classification problem. Their training set have 2D GPR b-scans collected by GPR sensor. The samples are divided into two classes - BEH and False Alarm (FA). BEH samples are also categorized into two types - buried anti-tank (AT) and anti-personnel (AP) landmine targets. Fig. 3.7 shows representative samples of AT and AP used for training.

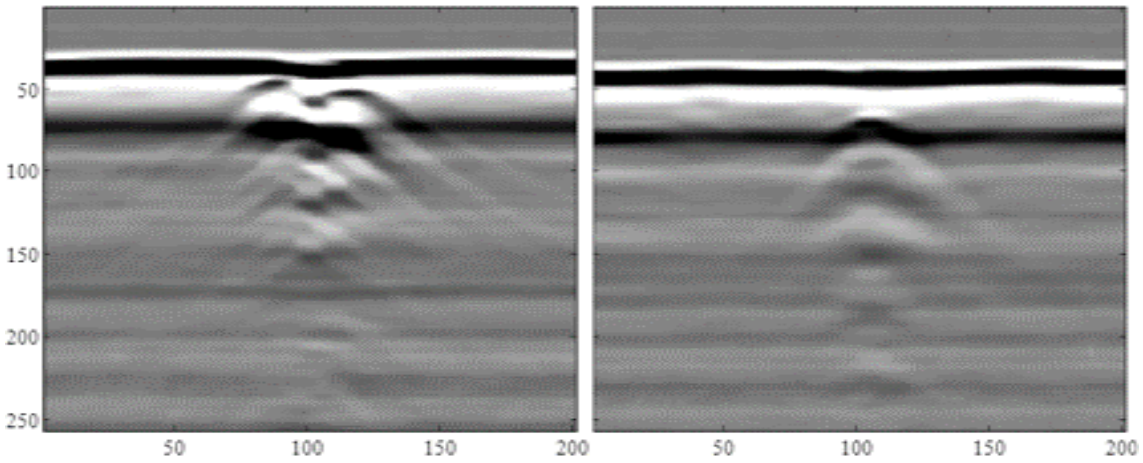


Figure 3.7: Two types of GPR b-Scans - AT and AP [35].

GPR b-scans are prepossessed before being passed to CNN for classification. Pre-processing steps include spatial re-sampling, ground-bounce tracking and alignment, and A-scan phase alignment. A median filtering and ZCA whitening method is also used to smoothen and whiten the b-scan. Also, a Deep Belief Network and a series of post-processing steps are applied to identify anomalous GPR signatures. Finally, 49×49 patches are extracted from the resultant anomalous signatures which are used as input to the CNN.

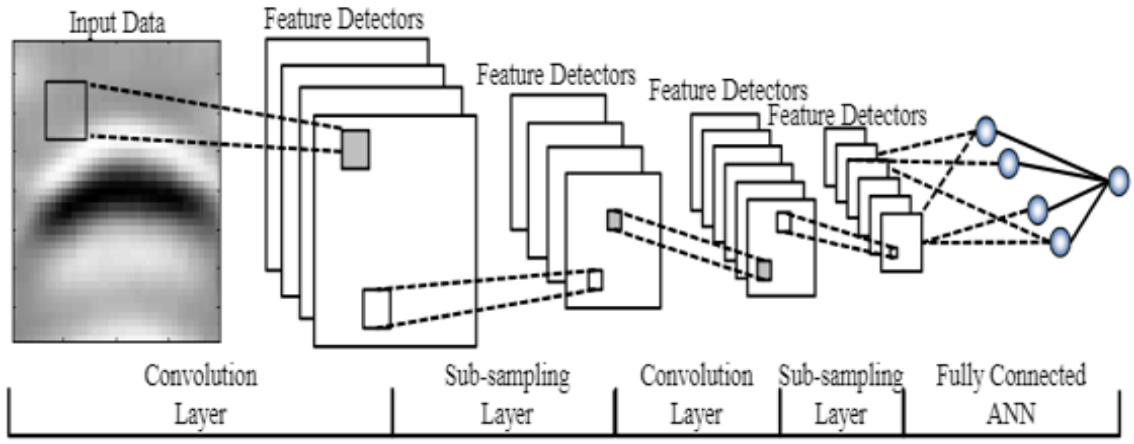


Figure 3.8: Architecture of CNN used in [35].

As shown in Fig. 3.8, the CNN used in this work consists of two convolutional layers, each followed by max-pooling layers, followed by a single fully-connected layer of

100 neurons. $Tanh$ activation function is used at each hidden layer whereas a softmax activation is used in the output layer to label each input as a BEH or FA. Stochastic gradient descent over mini-batch is used for optimizing the network. Several regularization methods are used to reduce overfitting on the training data including L1 and L2 regularization and dropout. They compared the result with traditional feature extraction algorithms like texture feature coding method (TFCM) [58] and edge histogram descriptors (EHD) [19]. In the ROC curve, CNN-based model has shown 8-12% improvement in accuracy compared to both TFCM and EHD .

Pham et. al. (2018) in [48] have used deep Faster R-CNN framework to identify underground buried objects by detecting hyperbolic reflections from GPR b-scans. They used both real and simulated GPR b-scans for training. They also used transfer learning [32] which is the way of using convolutional layer weights learned from a large image dataset and then fine-tuned them on the task-specific dataset. As shown in Fig. 3.9, they first trained a CNN on grey-scale CIFAR-10 dataset [33] which includes 50K training samples and 10K test images of 10 classes of objects including cat, dog, frog, automobile etc. The convolution layer weights are then transferred to the equivalent layers in the Faster R-CNN frameworks. These weights, along with the entire framework weights are then fine-tuned on the GPR b-scan dataset. In case of small or even moderately large dataset, transfer learning has been proven to improve model accuracy in a significant manner [11]. Instead of color dataset, the grey-scale set of CIFAR-10 is used in order to better resemble the single-channel GPR b-scans.

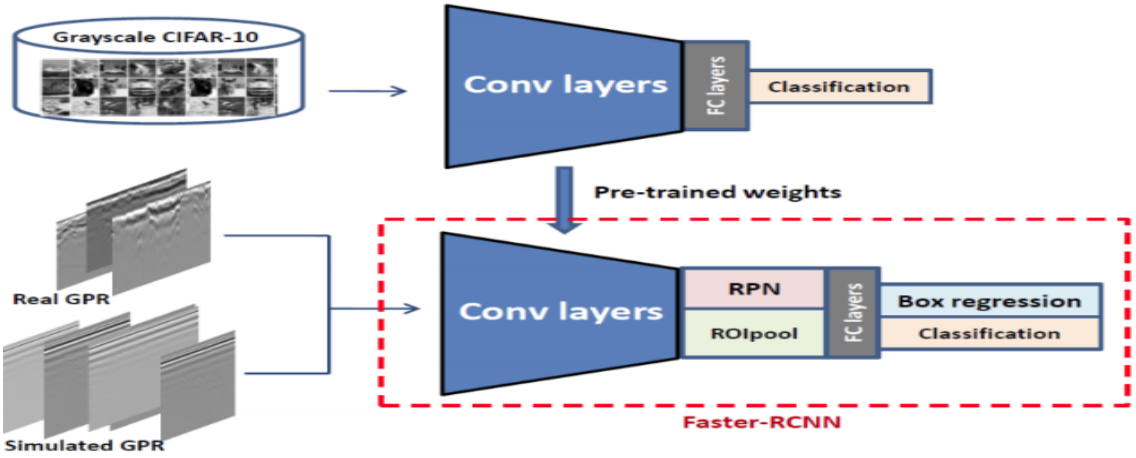


Figure 3.9: Proposed framework for buried object detection from GPR b-scans in [48].

The CNN model used consists of 3 convolution layers of 16, 32 and 64 filters of size 5×5 pixels and a max-pooling layer of 2×2 filters, each layer followed by a ReLu activation function. One fully-connected layer contains 64 neurons. Authors have compared their model performance to Cascade Object Detector (COD) based on Viola-Jones [60], HOG [46] and Haar-like [42] features and reported that their Faster R-CNN model outperformed each of them significantly. However, they did not report on the inference time comparison.

Chapter 4

Weld Quality Analysis using Deep Learning

In this work, we have incorporated the classic object detection task as a base to solve a larger problem of weld quality analysis. Our proposed solution uses cutting-edge deep learning based object detection frameworks to detect geometrical components of weld shape from ultrasonic b-scans. These components are representative to the physical phenomena taking place during the weld process. Analysis of these detected components in terms of their presence and position relative to other components guides the ultimate decision of the weld quality. This is a new application of the decades old object detection task in enabling machines perform a higher level job of quality analysis.

4.1 Data collection and pre-processing

4.1.1 Data Collection

For this research, we have used real-world spot-weld b-scans. Majority of them are collected from different automotive companies, whereas rest of the data are collected

by replicating the original ones in the IDIR laboratory using a spot-welder robot. B-scans are stored in SQLite database (.db3 files) as binary strings of 4-byte floats along with some metadata. Metadata contains additional information of each weld scan including weld ID, weld scan dimension along x and y-axis, current on and off position along x-axis, plate thicknesses etc. In Fig. 4.1, current on and off position is shown on a particular weld b-scan. Along the x-axis, current is turned on at 50ms, and kept flowing for 200ms. After that, at 250ms current is turned off.

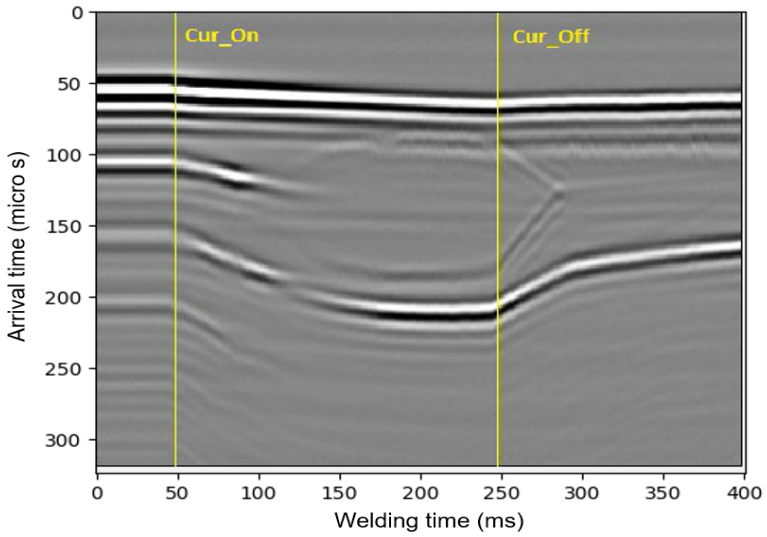
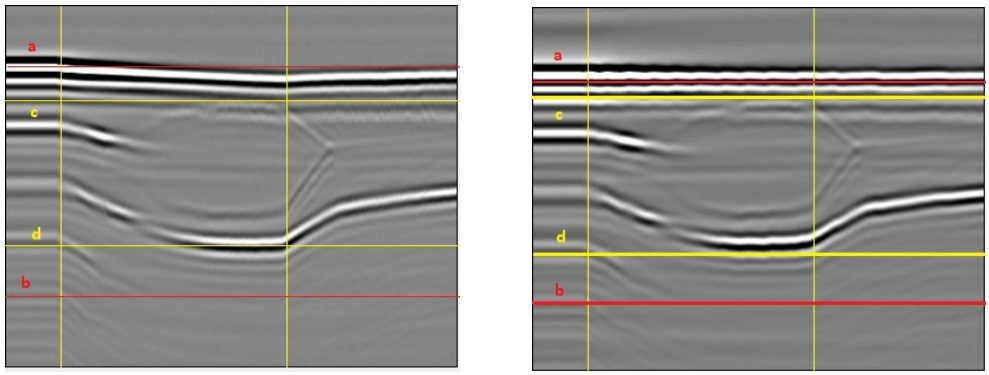


Figure 4.1: Current on and off position in a b-scan.

For our work, 2-D grey-scale weld images are used which are obtained by converting binary strings to 4-byte floating point numbers and then converting to pixel values ranged from 0-255. Our dataset contains weld samples of different plate thicknesses including both symmetric and asymmetric scans. Symmetric scans are weld scans of metal sheets of same thickness (1.2mm + 1.2mm) whereas asymmetric scans are weld scans of metal sheets with different thickness (1.2mm + 0.8mm). Besides, apart from two sheet welds, we have collected samples of three sheet welds as well (1.2mm + 1.2mm + 1.2mm).



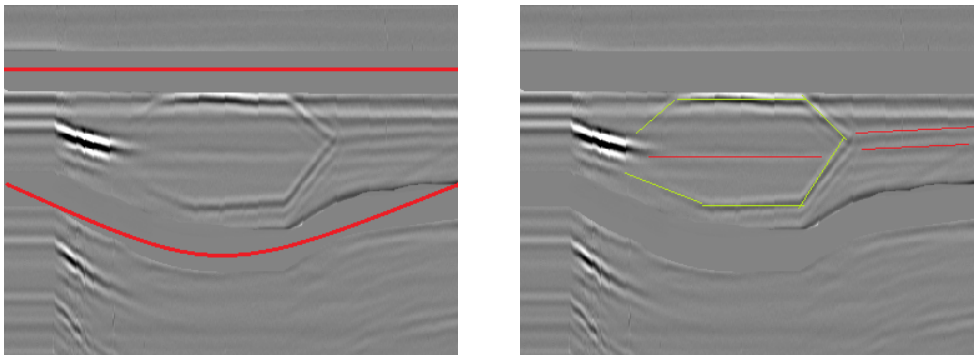
(a) original (b) normalized

Figure 4.2: Normalize original b-scan based on top and bottom interface.

4.1.2 Data Pre-processing

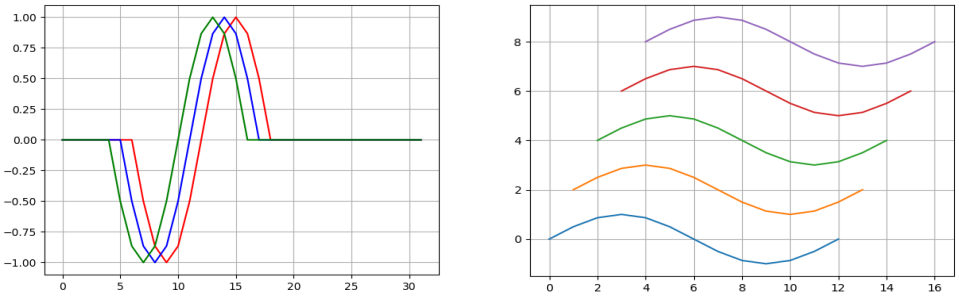
Different image and signal processing methods are used on the original b-scans in order to get rid of redundant features and enhance the properties to be detected. At first, b-scans are normalized based on the top and bottom interfaces. During welding process, pressure exerted by copper caps make the weld shape in the b-scan deviate from it's original coordinates and shift a bit in the place of pressure exertion, the portion between current on and off as shown in Fig. 4.2. In Fig. 4.2(a), horizontal lines a and b refer to the original coordinates of top and bottom interface respectively. Along the y-axis, each pixel is shifted by the offset of top interface from line a and then shifted by half the offset of bottom interface from line b. Fig. 4.2(b) shows the resultant image that is straightened up or normalized based on both top and bottom interfaces.

Next, we can see that the top and bottom interface have very high amplitude signals compared to the nugget shape in between which is our region of interest. In order to enhance these low amplitude signals, some filtering methods are applied. At first, high amplitude signals are removed by replacing the pixel values with mean pixel value, as shown by the highlighted parts in Fig. 4.3(a). At this point, we would



(a) high amplitude pixels removed (b) components to detect

Figure 4.3: Processed image after high amplitude pixels removal.

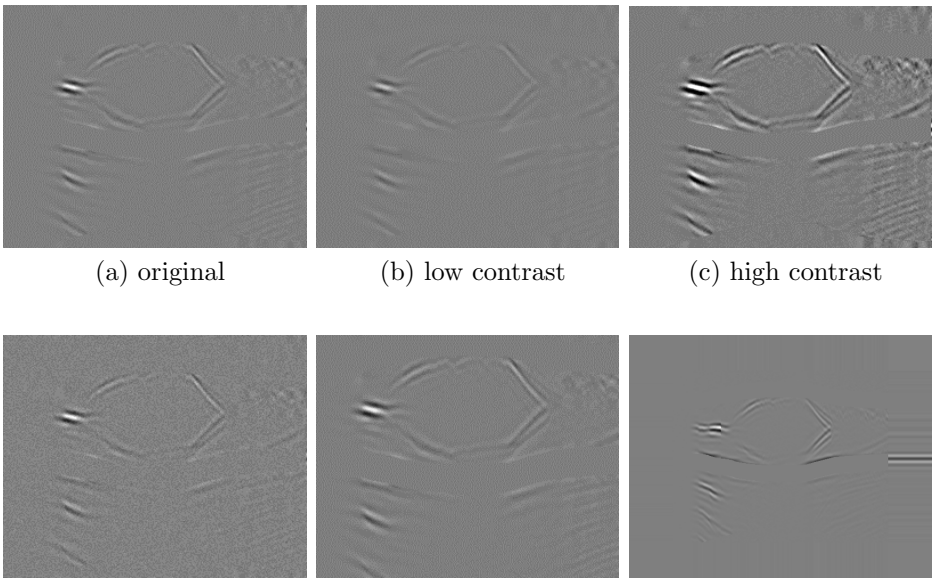


(a) horizontal filter (b) 45° diagonal filter

Figure 4.4: Horizontal and diagonal filter made of sinusoidal waves.

like to magnify the signals of the green highlighted parts of the image and get rid of redundant components like the lines highlighted in red as shown in Fig. 4.3(b).

In order to achieve this, we have applied customized convolutional filters containing sinusoidal waves. Multiple filters like the one shown in Fig. 4.4(a) are convolved to b-scans so that the horizontal components in red get removed. Similarly, diagonal filters like the one in Fig. 4.4(b) are convolved which eventually enhance the green-highlighted fork-shaped components of the nugget . These highlighted parts are our region of interest to be detected from each b-scan.



(a) original (b) low contrast (c) high contrast

(d) random noise (e) random crop by 25% (f) aspect Ratio by 15%

Figure 4.6: Different data augmentation methods.

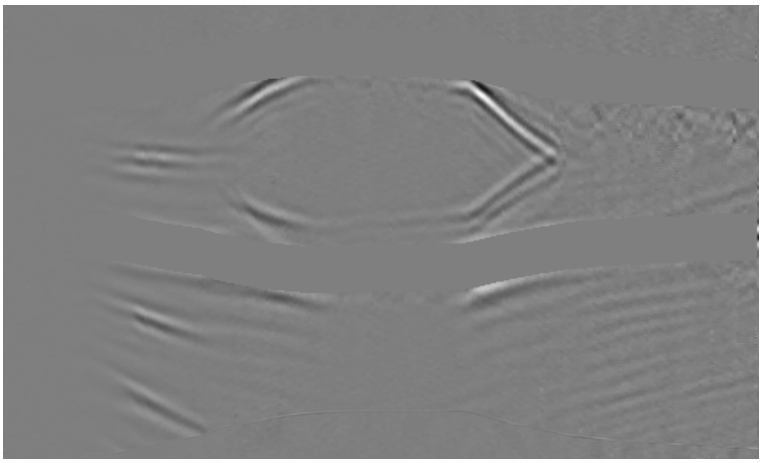


Figure 4.5: Final b-scan after pre-processing.

Finally, Fig. 4.5 shows the final processed b-scan that is used as training image in our deep learning model.

4.1.3 Data Augmentation

In order to increase the number of weld b-scans in our dataset, several data augmentation techniques are applied. From each original b-scan, five different augmented

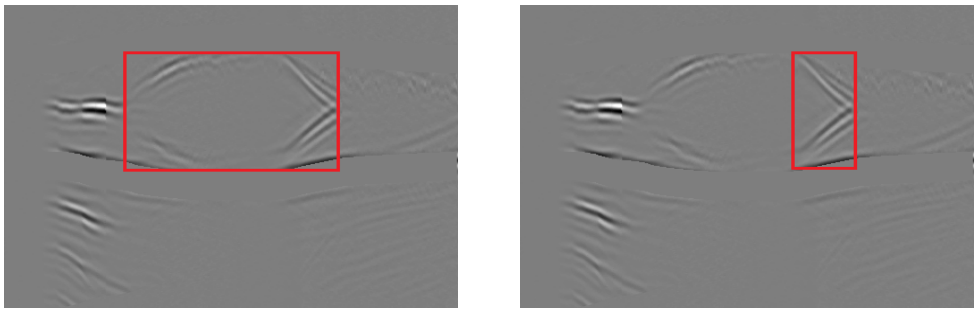
samples are generated using the following five methods.

- Apply 25% low contrast, as shown in Fig. 4.6(b)
- Apply 25% high contrast, as shown in Fig. 4.6(c)
- Apply random noise, as shown in Fig. 4.6(d)
- Crop by 25% and 50% in any of the randomly chosen side from top, bottom, left and right, as shown in Fig. 4.6(e)
- Change aspect ratio by a percentage randomly sampled from 0.5% to 30% along either horizontal or vertical axis, as shown in Fig. 4.6(f)

Augmented samples are generated by applying the above-mentioned techniques independently, as well as a combination of two or more of them. As a result, the number of weld samples in our dataset has been increased by 10 times.

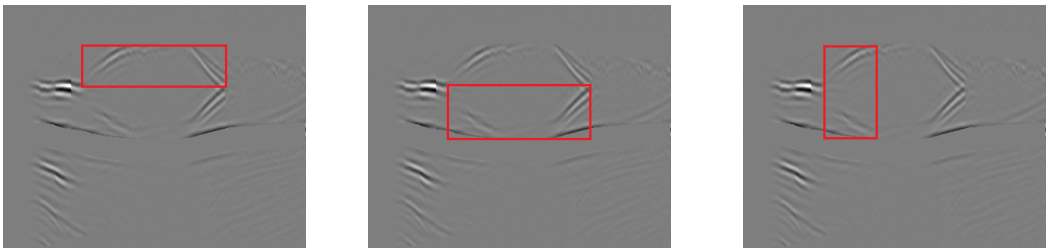
4.1.4 Building Labeled Dataset

In order to train deep learning model to detect our desired classes or objects, samples are required to be well-labeled. Previous research conducted in IDIR have reported that the presence of certain components of the weld nugget and their geometric position in the b-scan is highly representative of the weld quality (i.e. whether it is a good weld or bad weld etc.). Fig. 4.7 shows two of the most important components called *nugget* and *solid*. Detection of class *nugget* tells us that a whole nugget is formed, which highly indicates that the welding is complete. Similarly, class *solid* indicates the proper solidification of the molten nugget after current is turned off. Therefore, it is justifiable to train our model to detect these two classes from b-scan. However, due to hardware limitation and other external influence most of the cases b-scans fail to visualize the whole nugget even if it is formed. To tackle those cases, the whole



(a) nugget (b) solid

Figure 4.7: Nugget and Solid class.



(a) top (b) bottom (c) grow

Figure 4.8: Top, Bottom and Grow class.

nugget is divided into three fragments called *grow*, *top* and *bottom*, as shown in Fig. 4.8. As a result, even if the whole nugget is not prominent enough to be captured in the b-scan, we can still collect useful information from the evident components. By creating these additional classes we have managed to detect weld quality of partially visible or distorted b-scans, which has made our model more robust.

In order to build labeled dataset, each b-scan is manually annotated to detect any or all the five classes based on if they are visible. We have used *labelImg* [59] annotation tool for annotating the classes. It generates annotation record where each row holds annotation for a single class for each b-scan. A sample of the annotation file is shown in Fig. 4.9. Here, each row contains an image name, image dimension (i.e. height and width), a single class and its coordinates. ($xmin$, $ymin$) is the position of the left-most corner and ($xmax$, $ymax$) is the position of the right-most corner of

the bounding box around that specific class.

```
filename,width,height,class,xmin,ymin,xmax,ymax
pstv_10_41_57_719.png,333,400,nugget,157,98,193,140
pstv_10_41_57_719.png,333,400,solid,177,98,193,140
pstv_10_41_57_719.png,333,400,top,157,98,193,117
pstv_10_41_57_719.png,333,400,bottom,157,117,193,140
pstv_10_41_57_719.png,333,400,grow,157,98,177,140
```

Figure 4.9: Annotation record.

4.1.5 Signal Processing - Envelope Detection

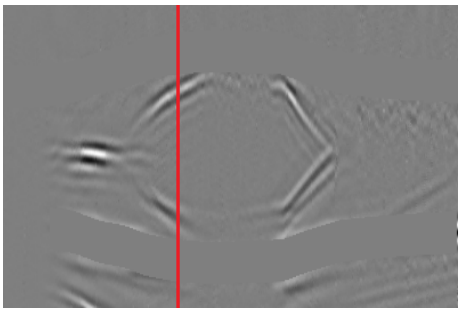
As mentioned in Chapter 1, each b-scan is made of multiple a-scans stacked together in the order of time. These a-scans are mainly real-valued ultrasonic signals which means that the Fourier Transform [63] of these signals contain both positive and negative frequency components [62]. However, negative frequency components hold redundant information and therefore can be discarded with no loss of signal. They can be filtered out by converting a real-valued a-scan to complex-valued function that has negative half of the frequency spectrum zeroed out. This complex-valued function is called *analytic signal*. The real and imaginary parts of an analytic signal are real-valued functions related to each other by Hilbert transform [62]. Hilbert Transform is a linear function that convolves the Fourier Transform of a real-valued function *u(t)* with a convolution function $1/(\pi t)$ as defined in Equation (4.1) [64].

$$H(u)(t) = \frac{1}{\pi} \int_{\infty}^{-\infty} \frac{u(\tau)}{t-\tau} d\tau \tag{4.1}$$

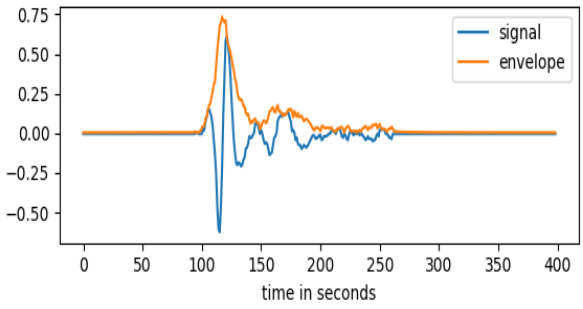
Therefore, analytic signal $x_{a(t)}$ of any signal x_t is obtained from Equation (4.2) [52], where F is the Fourier Transform, U is the unit step function and y is the Hilbert Transform.

$$x_{a(t)} = F^{-1}(F(x_t)2U) = x + iy \tag{4.2}$$

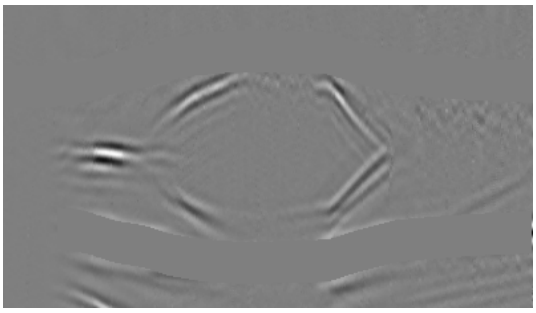
The purpose of using Hilbert transform is to determine the amplitude envelope of



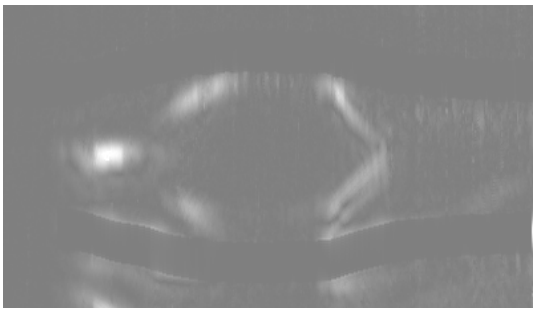
(a) original b-scan



(b) envelope signal



(c) original b-scan



(d) envelope b-scan

Figure 4.10: Envelope Detection.

signal by magnitude of the analytic signal (i.e. taking the absolute value of analytic signal). Envelope of a signal concentrates all signal energy into the center around a single carrier frequency. Therefore, it is very useful for feature extraction from signals as it enables high-amplitude signals be more prominent to be detected and suppress noisy signals. In our work, we have performed envelope detection individually on each a-scan, thus generated an enhanced version of the b-scans. Fig. 4.10(b) shows the original signal and enveloped signal of a single a-scan located at the position of the red line in the b-scan in Fig. 4.10(a). Fig. 4.10(d) shows the corresponding b-scan after converting its a-scans or signals to envelope signal.

4.2 Component Detection and Localization

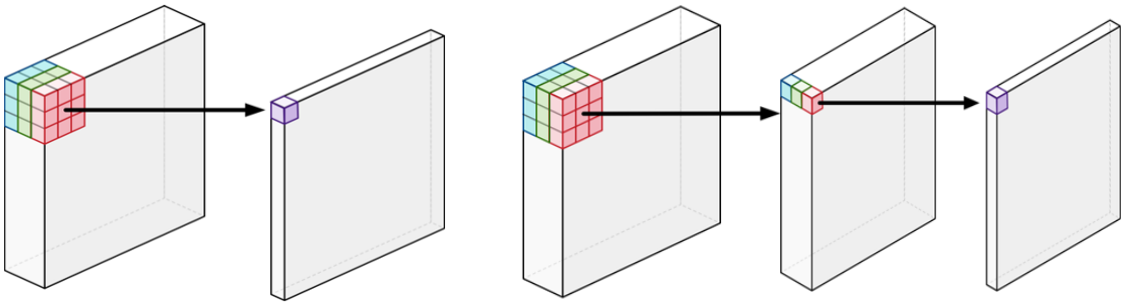
A customized model of Single Shot Multibox Detector (SSD) framework [61] has been implemented for component detection and localization from b-scans. Given an input image or in our case a b-scan, the model provides a list of bounding boxes, a label or class name assigned to each of the boxes and a confidence score of each box being labeled as the assigned class. Detected bounding boxes are provided in terms of coordinates. Using them some numerical values or parameters are calculated which help to analyze and ultimately decide the weld quality.

SSD framework is a single feed-forward convolutional neural network which performs both object localization and classification in a single forward pass. Yet the whole network is composed of six major parts or algorithms and each of them plays crucial role in providing the final output.

- Base Network for Feature Extraction
- Convolutional Box Predictor
- Priors generation
- Matching priors to ground-truth boxes
- Hard Negative Mining
- Post-processing - Non-maximum suppression

4.2.1 Base Network for Feature Extraction

SSD architecture is based on a deep CNN which works as the feature extractor. Deep CNNs has showed high performance in image classification on many benchmark



(a) regular convolution (b) depthwise separable convolution

Figure 4.11: Regular VS depthwise separable convolution.

datasets, which indicates that the features learned by these networks can be very useful in performing other more sophisticated computer vision tasks like object detection as well. The architecture that we have used in this work is built on a deep CNN called MobileNet [3] which is 10 times faster than its contemporary regular deep CNNs like VGGNet-16.

In MobileNet model, each standard convolutional layer is replaced by two separate layers called a depthwise convolution followed by a pointwise convolution, combinedly called as depthwise separable convolution. A standard convolution layer both filters and combines inputs, no matter how many channels it has, into a new set of single channel output pixels in one step. Whereas the depthwise separable convolution splits this process into two layers, a separate layer for filtering independently on each channel and a separate layer for combining the filtered channels into a single output channel. This factorization has the effect of drastically reducing computation and model size, therefore resulting in a very fast model for real-time performance [3]. In Fig. 4.11(a), regular convolution with 3×3 filters on a 3-channel input generates an output of a single channel image pixel in a single go. On the other hand, depthwise separable convolution in Fig. 4.11(b) at first applies 3×3 filters on each of the three channels separately, generating output image that also has 3 channels where each channel gets its own set of weights. In the next step, pointwise convolution is applied

which is same as a regular convolution but with a 1×1 kernel. Pointwise convolution simply adds up all the output channels of the depthwise convolution as a weighted sum in order to create new features.

Although, the end result of both regular and depthwise separable convolution is similar, a regular convolution has perform much more mathematical computation to obtain the result and needs to learn more weights compared to the latter one. For instance, the number of computation required to perform convolution with 32 3×3 filters on a 16-channels input using regular convolution is following,

$$16 \times 32 \times 3 \times 3 = 4608$$

Whereas the number of computation required by depthwise separable convolution is following (7 times faster),

$$16 \times 3 \times 3 + 16 \times 32 \times 1 \times 1 = 656$$

The original MobileNet architecture uses one regular convolutional layer in the first layer and 13 consecutive depthwise convolution layers each followed by a pointwise separable layer. Batch Normalization (BN) and ReLU are applied after each convolution. After the convolutional layers, there is an Average Pooling layer of kernel 1×1 followed by a Fully-connected layer linked to a softmax classifier for classification. Complete architecture of MobileNet is described in [3].

L2-Regularization

L2-regularization is a method used for reducing model complexity and overfitting on training data. During training, few weights can get too large and thus influence the output prediction heavily by overpowering rest of the weights. It makes the model lose its generalization ability and over fit on training data. L2-regularization penalizes

large weights and makes sure the model learns smaller but consistent weights throughout all the layers instead of clustering around few activations. L2-regularization is added to the loss function like Equation (4.3) [44].

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda \sum_{i}\sum_{j} W_{i,j}^2 \tag{4.3}$$

Here, N is the number of classes, L_i is the loss of class i, $W_{i,j}^2$ is the L2-regularization function that calculates sum of squares of each weight from the weight matrix W and λ is regularization hyperparameter which controls the amount or strength of regularization to be added.

Batch-normalization

Batch-normalization is another important regularization method that accelerates the model training by allowing it to use higher learning-rates. While training, our large dataset is passed as batches due to memory constraint. The distribution of the batches usually differ from each other which significantly affects the behavior of the model. This change in input distribution is called *Covariate shift*. Neural networks change the weights of each layer at each iteration at each mini-batch of input. Due to covariate shift, layer activations change quickly to adapt to its changing inputs. Since the activations of a previous layer are passed as output to the next layer, eventually all the layers are affected inconsistently for each batch. This results in longer time for model convergence as well as oscillating around a local minima forever. One way to avoid this case is to normalize activations of each layer to have zero mean and unit variance. It helps each layer to learn on a more stable distribution of inputs in all the batches which speeds-up the model training. However, forcing activations of all layers to be fixed mean and variance can limit the underlying representation of input data. Therefore, instead of keeping mean and variance fixed, batch normalization allows

the model to learn parameters γ and β that can convert the mean and variance to any value based on input data and model architecture. The equations for calculating γ and β are explained in [27].

4.2.2 Convolutional Box Predictor

For object detection task, original base extractor is extended to a larger network by removing and adding some successive layers. From the MobileNet architecture, the last fully-connected layer is discarded and a set of auxiliary convolutional layers are added which are called convolutional box predictor layers. These layers produce multiple feature maps of sizes 19×19, 10×10, 5×5, 3×3 and 1×1 in successive order which are stacked after the feature map coming from the last convolutional layer of MobileNet of size 38×38. All six auxiliary layers extract features from b-scans at multiple scales and progressively decrease the size of the input scan to each subsequent layer. Performing convolution on multiple scales enables detecting components from weld shapes of various sizes. As example, feature maps of size 19×19 performs better in detecting small shapes whereas 3×3 performs better for larger shapes. Each of the box predictor layers is connected to two heads, one regressor to predict bounding boxes and one sigmoid classifier to classify each of the detected boxes. Fig. 4.12 shows the SSD architecture that consists of base feature extractor and auxiliary predictor layers. Each layer predicts some bounding boxes for the class *solid*. Eventually predictions from all the layers are accumulated in the post-processing layer which provides the final prediction for *solid* as the output.

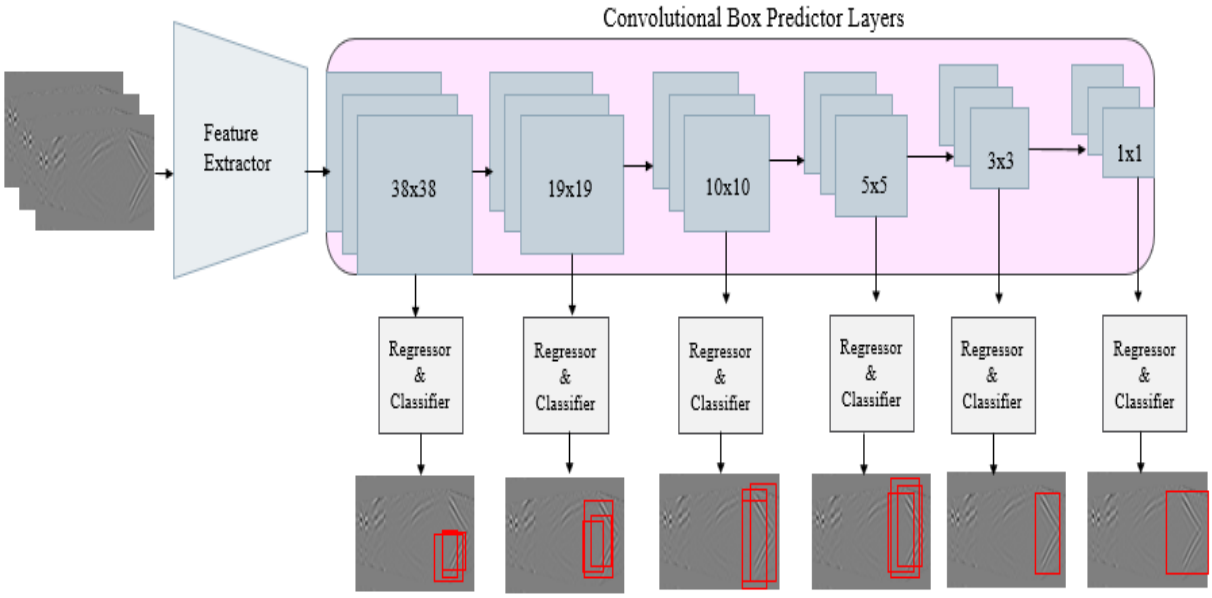


Figure 4.12: SSD framework with Convolutional Box Predictor layers.

4.2.3 Priors Generation

Priors or anchor generation is one of the most important tasks that can significantly affect the overall model performance. Priors are a collection of pre-defined boxes overlaid on the input scan as well as the predictor feature maps at different spatial locations, scales and aspect ratios that act as reference points for the model to predict the ground truth boxes. Ground truth boxes are the boxes that we have annotated while labeling the dataset. Prior boxes are really important because they provide a strong starting point for the bounding box regression algorithm to predict the ground truth boxes as opposed to starting predictions with completely random coordinates. Each feature map is divided into number of grids and each grid is associated to a set of priors or default bounding boxes of different dimensions and aspect ratios. Each of the priors predict exactly one bounding box. For our task, we have priors of six aspect ratios for each grid cell in each of the six feature maps. As shown in Fig. 4.13 each grid cell of a 5×5 feature map have six priors which mostly matches with the classes or objects we want to detect. As example, red priors are most likely to capture

nugget, *top* and *bottom* whereas blue priors are there to detect *grow* and *solid* of both smaller and larger sizes. Also, feature maps represent the dominant features of the b-scans at different scales, therefore generating priors on multiple scale feature maps increase the likelihood of any object of any size to be eventually detected, localized and appropriately classified.

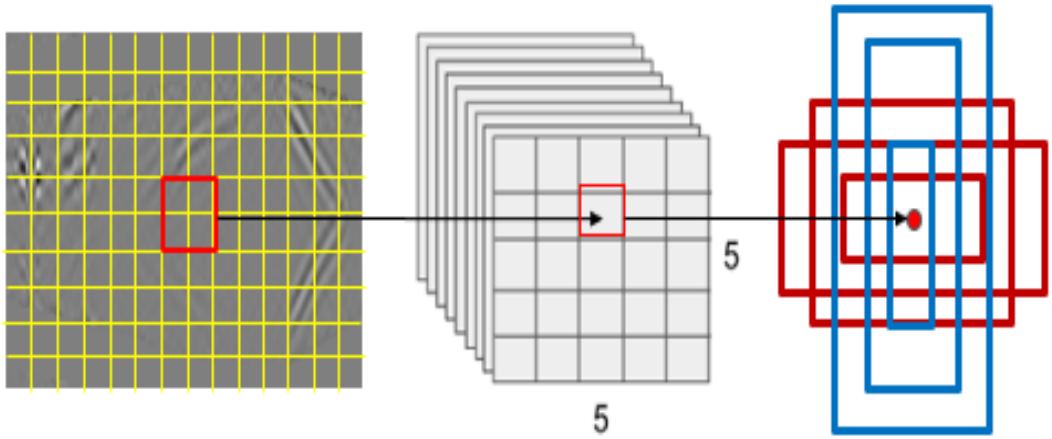


Figure 4.13: Prior generation on 5×5 feature map.

4.2.4 Matching Priors to Ground-truth Boxes

Next, each ground truth box is matched to priors having the biggest Jaccard Index [65] or Intersection over Union (IoU). In addition, unassigned priors having IoU greater than a threshold are also matched to ground truth boxes, for our case the threshold is 0.5. At this point, all the matched priors are called positive training examples and rest are considered as negative examples. In Fig. 4.14 blue prior is a positive example since it has greater IoU with the ground truth box of *solid* and red prior is a negative example because it does not have IoU greater than 0.5 with any of the ground truth boxes.

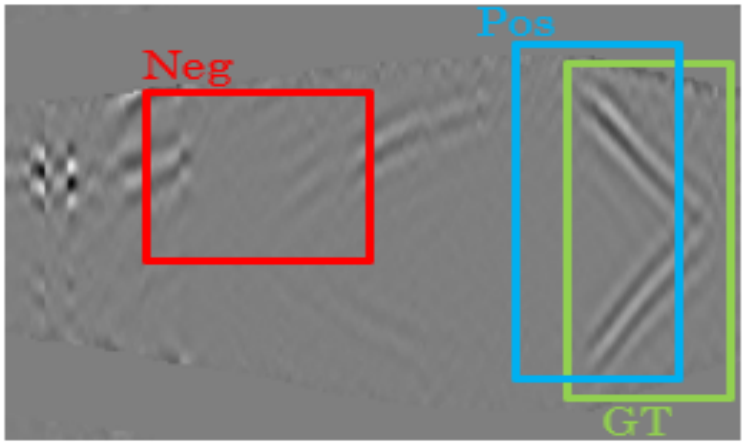


Figure 4.14: Matching priors to GT.

4.2.5 Hard Negative Mining

Among a large number of priors, only a few will be matched as positive samples and rest of the large number of priors will be left out as negatives. From the training point of view, there will be a class imbalance between positive and negative samples which may make the training set biased towards negatives. To avoid this, we randomly sample some of the negative examples from the negative pool in such a way so that the training set have a negative to positive ratio of 3:1. This process is called hard negative mining. The reason why we need to keep negative samples is because the network also needs to learn and be explicitly told about the features that lead to incorrect detection.

4.2.6 Post-processing - Non-maximum Suppression

At the prediction stage while both training and inference, every prior will have a set of bounding boxes and labels. As a result, there will be multiple predictions for the same object. Non-maximum suppression algorithm is used to filter out duplicate and redundant predictions and keep the top K most accurate predictions. It sorts the predictions by their confidence score for each class. Then it picks predictions

from the top and removes all the other predictions that have IoU greater than 0.5 with the picked one. In this way, it ends up with having maximum one hundred top predictions per class. This ensures that only the most likely predictions are kept by the model, while the more noisier ones are discarded. Fig. 4.15 shows complete SSD architecture with the non-maximum suppression layer which filters out all the less accurate predictions and generating the one as output with the highest confidence score.

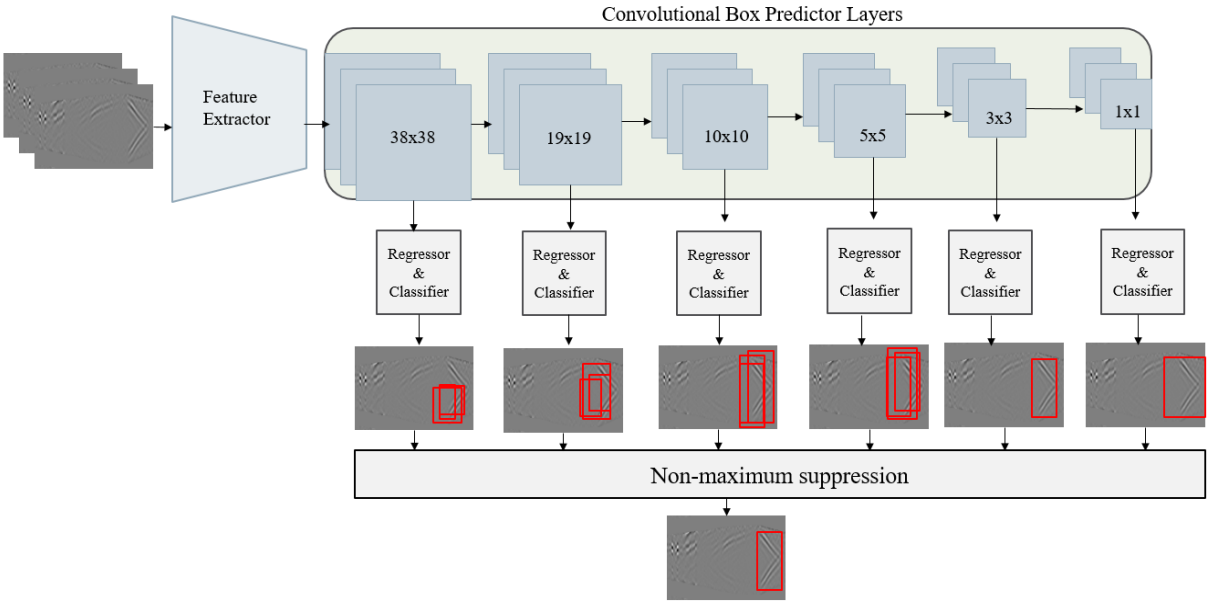


Figure 4.15: SSD framework with Non-maximum suppression.

4.2.7 Training

SSD network is trained end-to-end for multi-learning of classification and localization simultaneously. It starts with positive and negative training samples obtained from the matched priors. The global objective is to start with positive samples as predictions and being trained to regress the predictions closer to the matched ground-truth boxes. At each step for each positive sample for each grid cell it generates the following output,

- A probability vector of length $c + 1$, where c is the number of classes and one background class that indicates no object
- A vector with 4 elements $[x, y, width, height]$ representing the offset required to move the prior position to fit the ground-truth bounding box

After each training step, it keeps the priors for those the global loss function is reduced and re-adjust them to better match with the ground-truth boxes. Our model converges when the difference between the priors and ground-truth boxes are close to zero.

Loss Function

The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf) [61] where localization loss is the mismatch between the predicted boundary box and the ground truth box and confidence or classification loss is the loss in assigning class labels to predicted boxes. Therefore, the equation for total objective loss is defined in Equation (4.4) [61].

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \tag{4.4}$$

Here, x is 1 if the prior is matched to the determined ground truth box, and 0 otherwise, N is the number of matched priors, l is predicted bounding box, g is ground-truth bounding box, c is class, L_{conf} is confidence loss, L_{loc} is localization loss, and α is the weight for localization loss. We have used Smooth-L1 loss [17] for localization on l and g and Softmax loss [4] for optimizing confidence loss over multiple class confidences c. Details of the derivation of the loss function is described in [61].

4.2.8 Transfer Learning

One caveat of using very deep DNN is that it requires huge dataset to be trained and evaluated on. Fortunately, transfer learning is a popular methodology to use as a compensation for inadequate data. Transfer learning is the process of improving the learning in a new task through the transfer of knowledge learned from a related task. In practice, in computer vision deep neural networks like VGGNet-19 or Inception [55] are trained on large benchmark datasets with millions of images for image classification. Empirical studies have shown that the earlier layers in deep CNNs contain features that are generic to most of the images like edges, blob, lines etc. and later layers become progressively more refined to catch the details of the classes from the specific dataset [32]. Therefore, the weights learned by the earlier layers on one large dataset can be reused as a base of initialization of similar model for a different dataset. Instead of training a large network from scratch with random weight initialization, a pre-trained model starts off with previously learned weights and then whole or part of the network is retrained to fine-tune weights for a second dataset. Not only pre-trained models help in case of smaller database, it also reduces the training time significantly by converging faster. Authors in [23] have reported that in case of limited dataset, pre-training model on very large dataset improves deep CNN performance significantly.

For our task, we have used pre-trained weights learned from MS COCO dataset [36] that consists of more than 2 million images of random common objects from everyday use. We removed the final fully-connected layers and fine-tuned the earlier depthwise separable convolution layers weights by training on our b-scan dataset. Although, this dataset is completely different than our b-scan dataset, initializing the model with these pre-trained weights have significantly improved our model performance. Fig. 4.16 shows the diagram of the transfer learning methodology used in our

work.

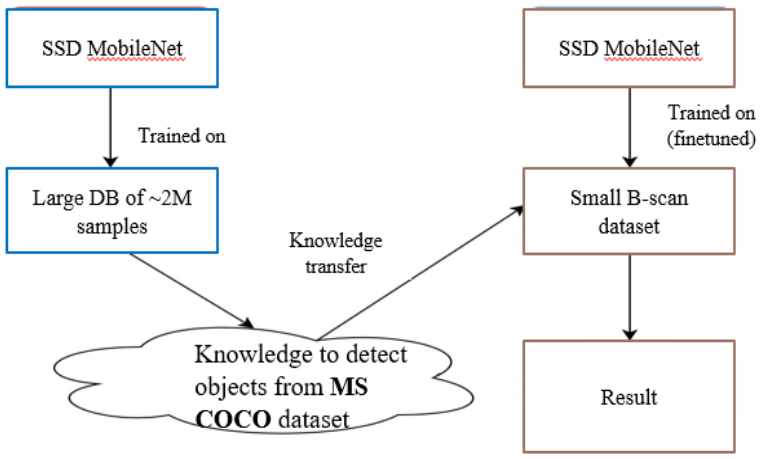
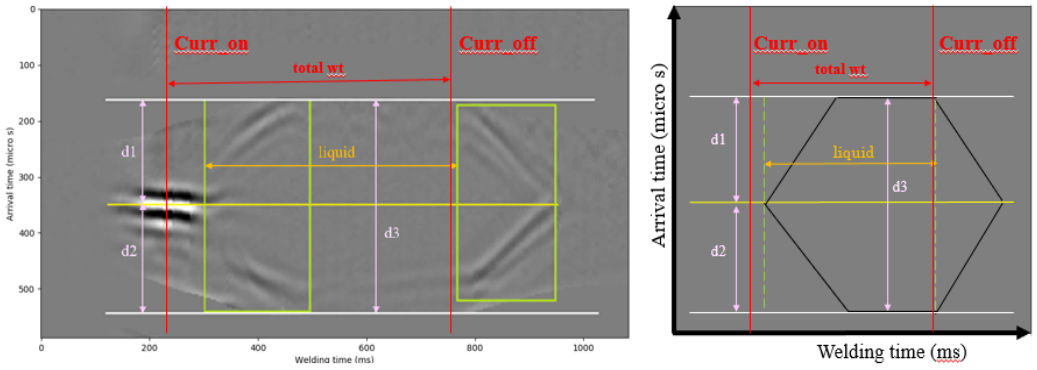


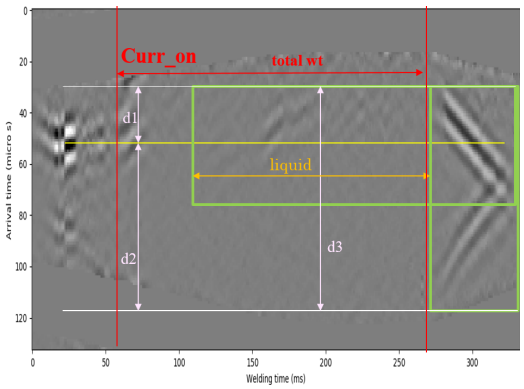
Figure 4.16: Transfer learning.

4.3 Decide Weld Quality

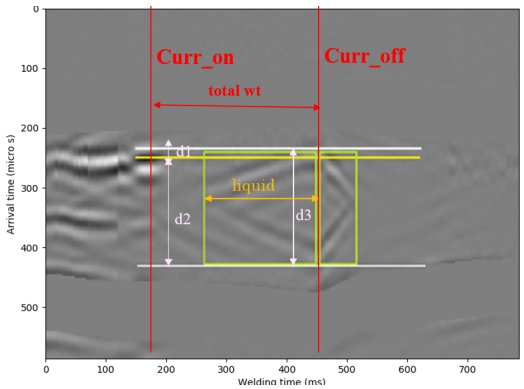
Once our proposed model detects and localizes the classes, we can use the coordinates of the bounding boxes and interpret them to decide on the weld quality. There are several parameters that can be measured from bounding box coordinates which are highly correlated with the overall weld quality. As example, in Fig. 4.17 *d1*, *d2*, *d3*, *liquid* all are weld parameters which we can calculated. *d1* and *d2* are the amount of liquid penetration happened in plate-1 and plate-2 respectively. They are calculated by taking the distance from top and bottom line (showed in white) from the solid-solid interface line as shown in yellow. The position of solid-solid interface line is already obtained from weld meta-data. The top and bottom lines are basically the *ymin* and *ymax* coordinates of *grow* and *solid*. If *ymin* and *ymax* don't align for both the classes, then the greater (or deeper) value is considered. Also, in cases where *grow* is not visible, *ymin* coordinate of *top* is taken instead, as seen in Fig. 4.17(b) and *ymax* is taken from *solid*. If all the classes are visible, then average of *grow*, *top* and *solid* for *ymin* and *grow*, *bottom* and *solid* for *ymax* is calculated. *d3* is the distance from *ymin* of *top* and *ymax* of *bottom* which indicates the depth of



(a) good weld parameters corresponding to its schematic view



(b) acceptable weld



(c) bad weld

Figure 4.17: Decide weld quality from bounding boxes.

the nugget formed. Another important parameter is *liquid* which defines the time metal has been melting for into liquid. It is the time spent since growth of nugget started until it starts to solidify. Empirical observation shows that good welds usually require liquid to flow for at least $40 - 60\%$ of the total welding time *total wt*. Total welding is the time passed between current on and off as shown by the red arrow, which is already known. Thus, liquid time is calculated by taking the difference of *xmin* of *grow* and *xmin* of *solid*. If the difference is at least $40 - 60\%$ of *total wt*, it can be considered as good or acceptable. Length of other parameters *d1*, *d2* and *d3* also have to be over a certain threshold pre-defined by the weld operator for a weld to be passed. As example, weld in Fig. 4.17(a) has very well proportion between *d1* and *d2* thus can be considered as good. However, weld in Fig. 4.17(b) has smaller

penetration in plate-1 and still can be considered as acceptable. On the other hand, weld in Fig. 4.17(c) has almost failed to penetrate plate-1, thus marked as bad.

Based on the value obtained for each of the parameters, each weld scan is given a quality ranking between 0.0 to 1.0, 0.0 defining the worst and 1.0 defining the best quality. Weld samples ranked under 0.40 is considered as bad, between 0.40 and 0.55 as acceptable and over 0.55 as good.

4.4 Model Complexity

Majority of the literature report time complexity of deep learning models in terms of total time taken by the model for training and inference when run on specific hardware. This is different than traditional algorithms where time-complexity is defined by the order of input size using big-oh notation. Standard deep learning models perform millions of matrix multiplication, convolution, product and summation which is computationally very expensive. However, most of these calculations can be performed in parallel because at each layer of the network thousands of identical artificial neurons perform the same computation. This uniformity can be leveraged by using Graphical Processing Unit (GPU) due to its parallel computing architecture, large number of computational unit and higher bandwidth to retrieve from memory. In this work, training our model in GPU has sped up total training time by 10 times than training on five-core CPU.

Apart from calculating running time on specific hardware, we also make an approximation of the asymptotic complexity to get an idea of which segment of the model takes up the largest amount of time while training and inference. Forward pass of the base CNN is dominated by the time complexity of matrix multiplication.

Number of multiplication to perform depends on size of input channel (or image resolution), number of convolutional layers, number of neurons in each layer, number of filters and size of each filter in each layer and the size of output feature map. Except for the input layer, size of input channel and output channel is determined by the number of neurons in each layer. In addition, at each layer there is an activation function which can be linear or quadratic based on the function. In our case, we use ReLu which performs element-wise calculation, thus runs in quadratic time on each neuron on each layer. All these are parameters or weights to be learned during training. Therefore, runtime of a convolutional model scales linearly with the total number of learnable weights. The total time-complexity of convolution layers for forward pass is as following,

$$\begin{aligned} time_{forward} &= time_{convolution} + time_{activation} \\ &= O(\sum_{l=1}^{L} n_{l-1} \cdot (f \cdot f) \cdot n_l \cdot (m_l \cdot m_l)) + O(L \cdot n_c) \\ &= O(weights) \end{aligned} \tag{4.5}$$

Here, l is the index of a convolutional layer, L is the total number of layers, n_{l-1} is the number of input channels of the l_{th} layer, f is filter height and width, n_l is the number of filters in l_{th} layer, m_l is the size of output feature map and n_c is number of neurons in a layer which in practice varies for each layer. For training, there is also back-propagation cost which also runs to calculate weights for all the parameters using chain-rule. Thus, $time_{backprop}$ is also $O(weights)$.

Apart from the above, there are additional computations like batch normalization, dropout, regression and classification using sigmoid function etc. but they take up only $5 - 10\%$ of the total training time, as tested with Chromium Event Profiling Tool [29]. Once the model is trained and all the weights are learned, inference just

performs summation of the product of weights in linear time. Memory complexity of the model is linear to the number of weights to be stored, thus $O(weights)$.

4.5 Model Evaluation

Accuracy of the trained models are measured for both classification and localization. The accuracy metric used in called Mean Average Precision or mAP [14]. AP is calculated for each class from the area under precision-recall curve. Precision measures the percentage of the model predictions that are made correct and recall measures the percentage of all the possible ground-truth boxes being detected.

$$\begin{aligned} Precision &= \frac{TruePositive}{TruePositive + FalsePositive} \\ Recall &= \frac{TruePositive}{TruePositive + FalseNegative} \end{aligned} \tag{4.6}$$

Here, *TruePositive* is the number of predictions that has IoU greater than 0.5 with ground-truth boxes, *FalsePositive* is the number of predictions with IoU less than 0.5 with ground-truth boxes and *FalseNegative* is the number of ground-truth boxes that are not detected by the model.

Now predictions are sorted by their confidence score from highest to lowest. Then 11 different confidence thresholds called ranks are chosen such that the recall at those confidence values have 11 values ranged from 0 to 1 by 0.1 interval. The thresholds should be such that the Recall at those confidence values is 0, 0.1, 0.2, 0.3 till 0.9 and 1.0. Average Precision or AP is now computed as the average of maximum precision values at these chosen 11 recall values. Therefore, AP for class c is defined as Equation

(4.7) [14] where *P(r)* is the precision value for one of the 11 recalls r.

$$AP_c = \frac{1}{11} \sum_{r\epsilon\{0.0,...1.0\}}^{R} max(P(r)) \tag{4.7}$$

Then mAP is just the average of APs over all the classes as shown in Equation (4.8), where C is the total number of classes and AP_c is AP for class c.

$$mAP = \frac{1}{C} \sum_{c}^{C} AP_c \tag{4.8}$$

The greater the mAP, the better the model is in terms of accuracy.

Chapter 5

Experiments and Result

This chapter includes all the experiments conducted to train different models with different settings. Model performance in terms of inference speed and accuracy and detection result on real b-scans are also reported.

5.1 Implementation Details

Our models are implemented and modified using Tensorflow Object Detection API released by Google under open-source license [24]. We have run lots of empirical analysis in order to design the best model for our task. Experimentation setup includes various combination of hyperparameter values, use of regularization methods, batch size, number of layers, number of filters and size of the filters in each layer, number of training samples, input image resolution, IoU threshold for Non-maximum suppression and mAP, whether or not to use transfer learning, whether or not use enveloped version of the b-scans etc. While selecting the best model, we take into account the best trade-off between three factors listed in the order of importance,

- Detection accuracy
- Real-time inference speed

- Required memory

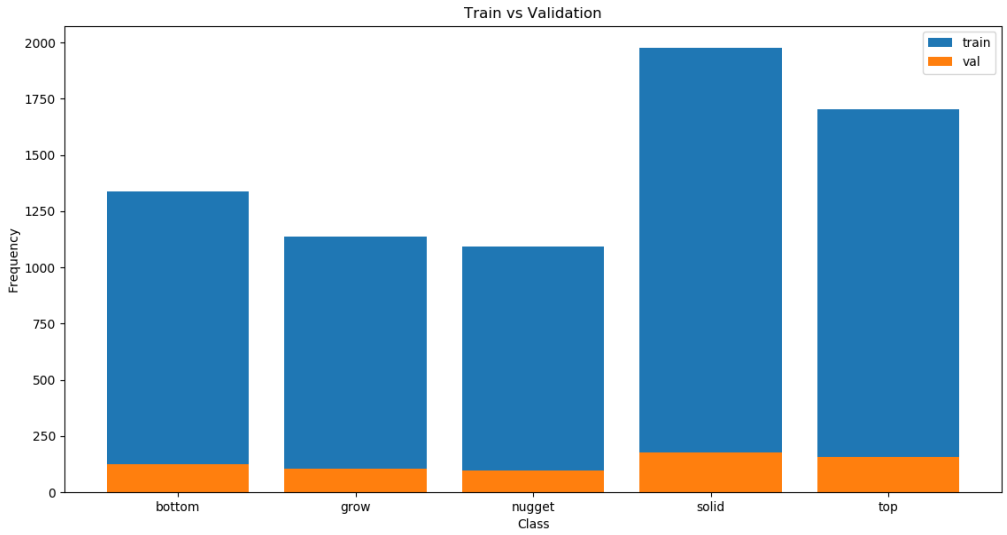
Also, while optimizing the model, we took the following steps in no fixed order,

- Reduce number of convolution layer as much as possible
- Use 3×3 filters in as many places as possible, as opposed to larger sized filters
- Reduce number of convolutional predictor layers as much as possible
- Collect as many original weld samples of different variety as possible

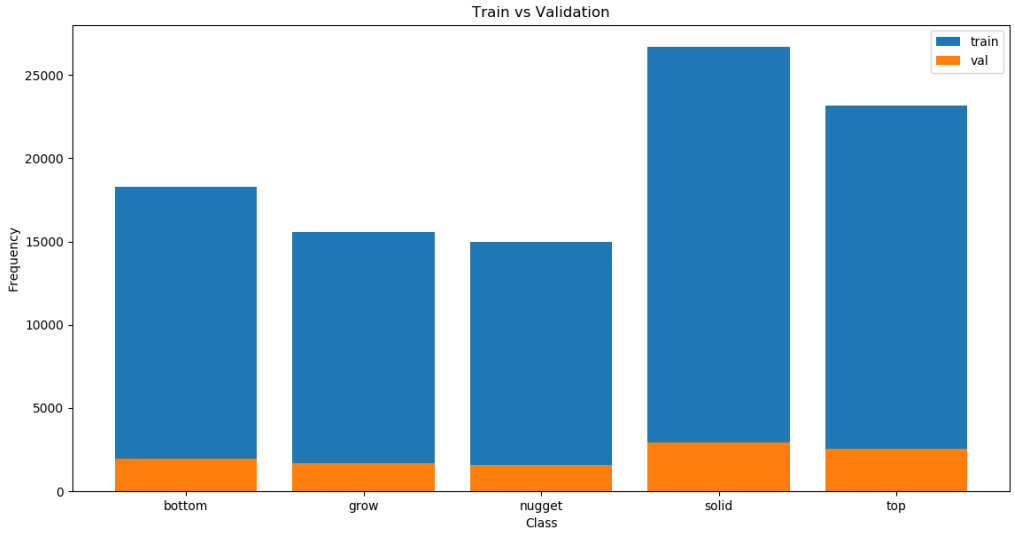
One important point to mention is that, any change in the model architecture removes the possibility of using transfer learning because the weights are learned for a specific model. In those cases, we trained model from scratch without initializing with pre-trained weights. Our original dataset consists of 2,375 training and 105 validation images. After data augmentation, training set contains 27,000 images and validation set contains 3,000 images. Each trained model is evaluated on a hold-out test set of 98 images. Fig. 5.1 shows the class distribution in the original and augmented dataset.

Our training dataset consists of weld b-scans of five categories which are notably different from each other in terms of weld size, weld shape, aspect ratio and image quality. They are so different due to the thickness and number of the plates to be welded. As example, along with two-plate welds, our dataset contains weld b-scans of three plates as well. There are also symmetric weld scans with both plates of same thickness and asymmetric weld scans with different plates of different thickness. Thickness of each of the plates in each category is described below with the thickness of plate-1, plate-2 and plate-3 (in case of three-plate welds) respectively.

- 1.2mm + 1.2mm (symmetric)

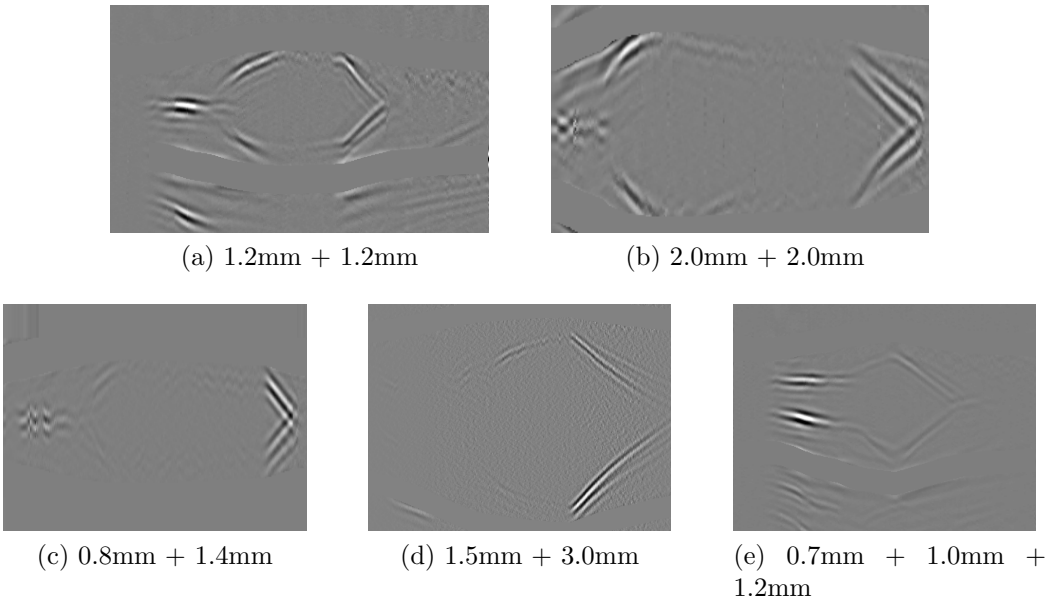


(a) original dataset



(b) augmented dataset

Figure 5.1: Distribution of five classes.



(a) 1.2mm + 1.2mm (b) 2.0mm + 2.0mm

(c) 0.8mm + 1.4mm (d) 1.5mm + 3.0mm (e) 0.7mm + 1.0mm + 1.2mm

Figure 5.2: Five types of weld samples.

- 2.0mm + 2.0mm (symmetric)
- 0.8mm + 1.4mm (asymmetric)
- 1.5mm + 3.0mm (asymmetric)
- 0.7mm + 1.0mm + 1.2mm (asymmetric 3-plate)

Fig. 5.2 shows a sample of each of the five categories of weld b-scans. Each of our models are trained to generalize well on all these different types of welds which is a challenging task. After running several experiments finally we obtained the following set of hyperparameters that works best on our dataset,

- Adam optimizer initial learning rate: 0.0004
- Batch size: 24
- λ of L2-regularization: 0.00004
- ϵ of Batch normalization: 0.001

- IoU threshold: 0.6
- Input image resolution: 100×100

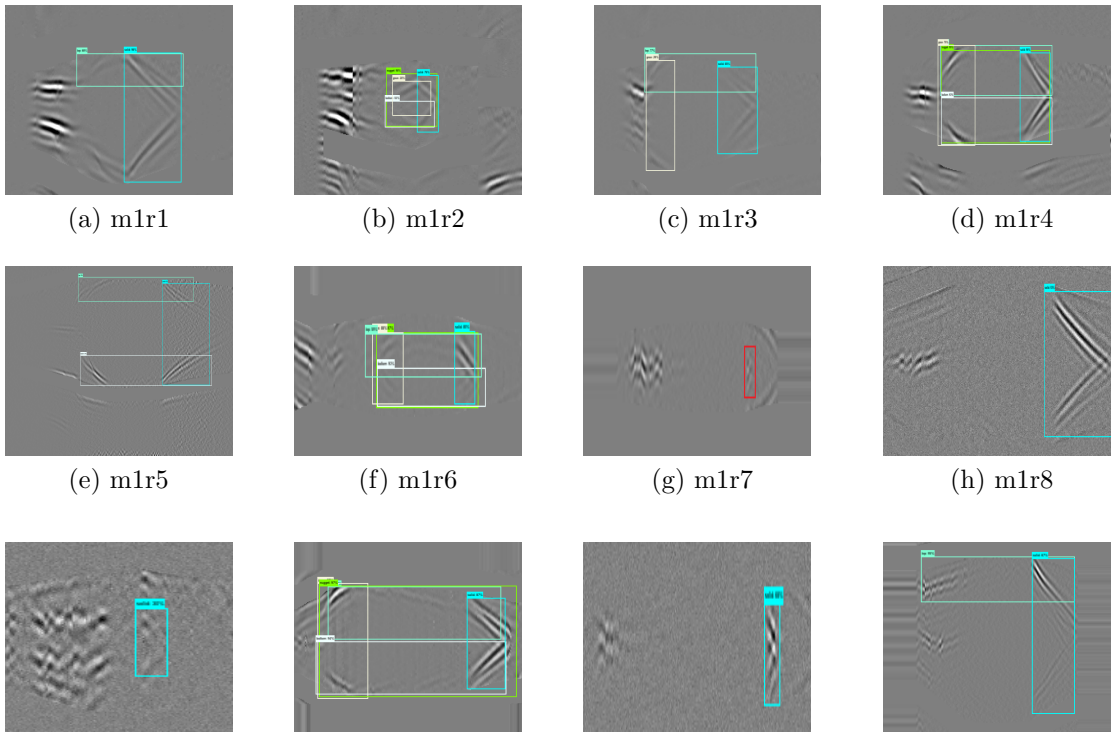
All the models are trained for around 50 to 90 thousand epochs. According to our observation, in most of the cases model loss has converged after these many epochs. Loss curve is monitored in Tensorboard. Training is done on Nvidia GeForce GTX 1070ti GPU and inference is done on Intel Corei5 CPU.

5.2 Different Models

We have compared the performance between five different models. The hyperparamters mentioned above are fixed for each of the model, however they are significantly different from each other in terms of framework, use of transfer learning, data augmentation and enveloped inputs. Except for Model 5, transfer learning is used for all the models.

- Model 1: SSD framework with original set of regular b-scans (without envelope detection)
- Model 2: SSD framework with original set of enveloped b-scans
- Model 3: Faster R-CNN framework with original set of regular b-scans
- Model 4: SSD framework with augmented set of regular b-scans with transfer learning
- Model 5: SSD framework with augmented set of regular b-scans without transfer learning

As our original dataset is too small to train on from scratch, SSD and Faster R-CNN framework with original set without transfer learning could not be tested.



(a) m1r1 (b) m1r2 (c) m1r3 (d) m1r4

(e) m1r5 (f) m1r6 (g) m1r7 (h) m1r8

(i) m1r9 (j) m1r10 (k) m1r11 (l) m1r12

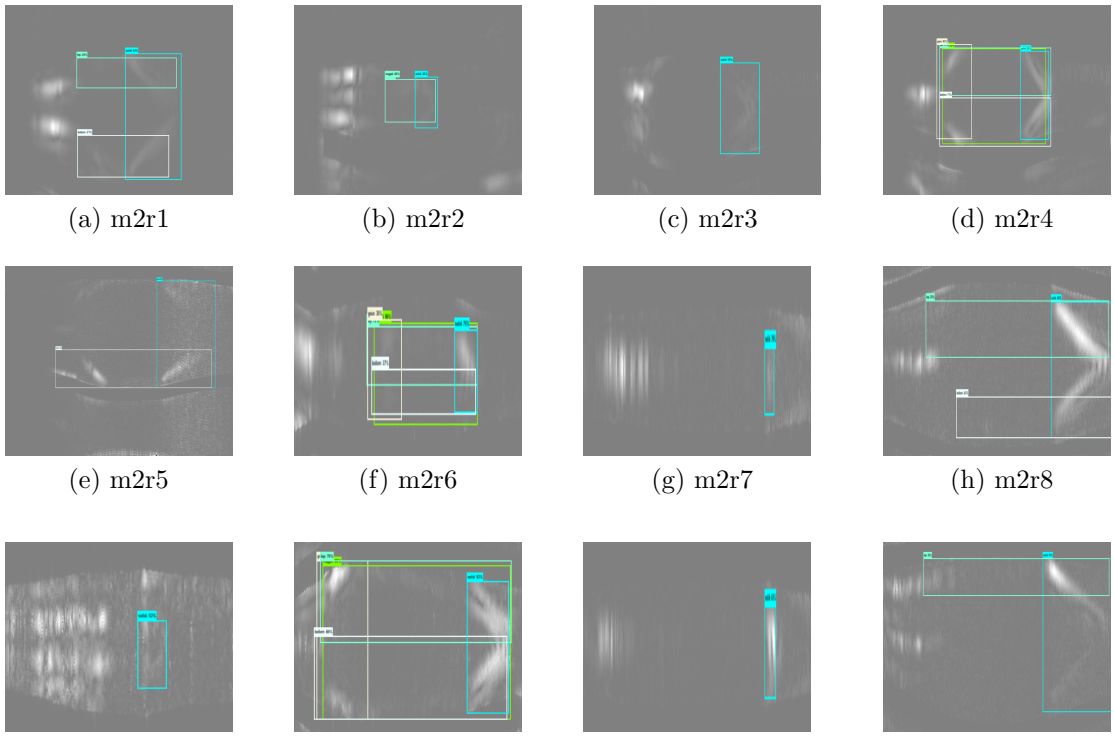
Figure 5.3: Sample results of model 1.

5.3 Result

Below are some results from the hold-out test dataset predicted by all the five models. Same set of b-scans are provided for each model in the same order for the sake of comparison. Results are hand-picked so that they represent various types of weld scans.

Model 1 - SSD framework with original set of regular b-scans

Fig. 5.3 shows some of the results generated by model 1. As seen, this model successfully detects most of the classes including cases when only few of the classes are visible in the b-scan and when classes have various shapes or aspect ratio. However, model 1 could not detect the *solid* class in Fig. 5.3(g) highlighted as red.



(a) m2r1 (b) m2r2 (c) m2r3 (d) m2r4

(e) m2r5 (f) m2r6 (g) m2r7 (h) m2r8

(i) m2r9 (j) m2r10 (k) m2r11 (l) m2r12

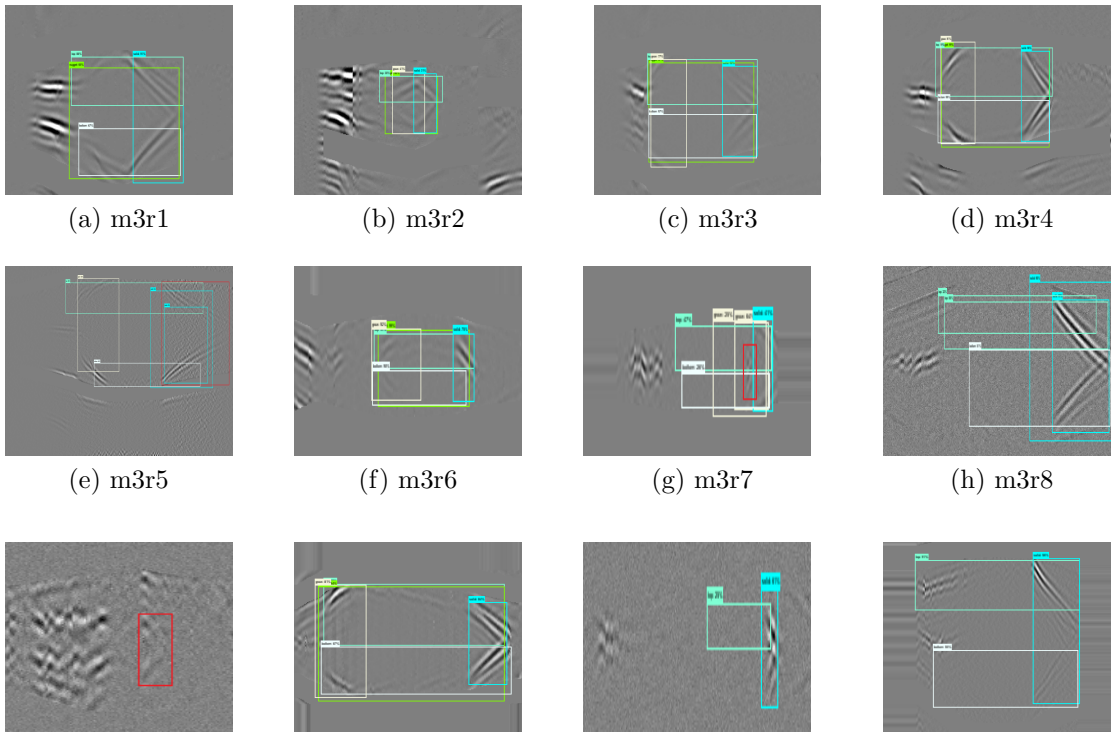
Figure 5.4: Sample results of model 2.

Model 2 - SSD framework with original set of enveloped b-scans

Fig. 5.4 shows results generated by model 2. As seen, this model performs better in some of the scans than model 1. As example, model 1 detects two classes in Fig. 5.3(a) but model 2 detects all the three classes including *bottom* in Fig. 5.4(a). It also gives better result for Fig. 5.4(h) compared to Fig. 5.3(h). Moreover, it successfully detects the *solid* class in Fig. 5.4(g) which is missed by model 1. However, model 2 detects less number of classes in Fig. 5.4(b) and Fig. 5.4(c) than model 1.

Model 3 - Faster R-CNN framework with original set of regular b-scans

Fig. 5.5 shows results generated by model 3. Result shows that overall it performs worse than model 1 and 2. As example, it misses the *solid* class in Fig. 5.5(i). More importantly, its localization of bounding boxes are less accurate than the previous



(a) m3r1 (b) m3r2 (c) m3r3 (d) m3r4

(e) m3r5 (f) m3r6 (g) m3r7 (h) m3r8

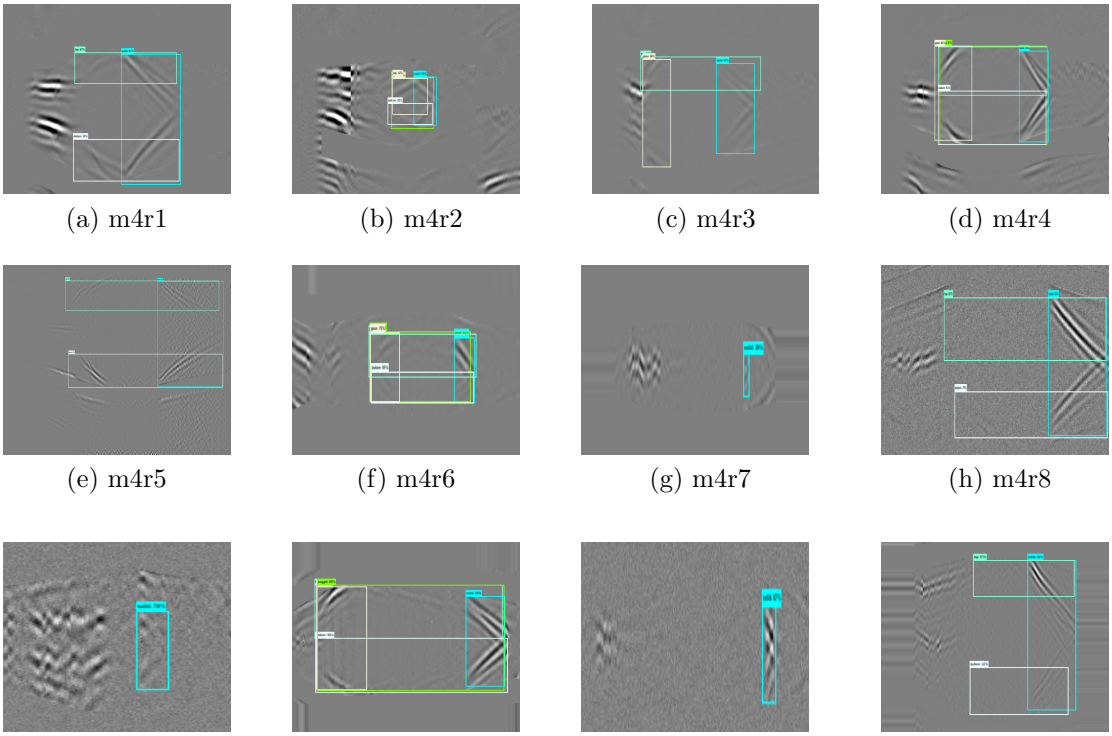
(i) m3r9 (j) m3r10 (k) m3r11 (l) m3r12

Figure 5.5: Sample results of model 3.

models. As shown in Fig. 5.5(e), it has detected two bounding boxes for *solid* but both of them are largely off than the ground truth box drawn in red. Besides, in Fig. 5.5(g) it missclassifies all the classes when there is just one ground-truth box for *solid* as shown in red.

Model 4 - SSD framework with augmented set of regular b-scans with transfer learning

Fig. 5.6 shows results generated by model 4. According to the result, this model has outperformed all the previous three models in terms of both detection and localization. It has detected 100 percent of the total ground-truth boxes including some challenging ones like Fig. 5.6(b), (g), (i) and (k) where other models have difficulty in detecting the boxes accurately.

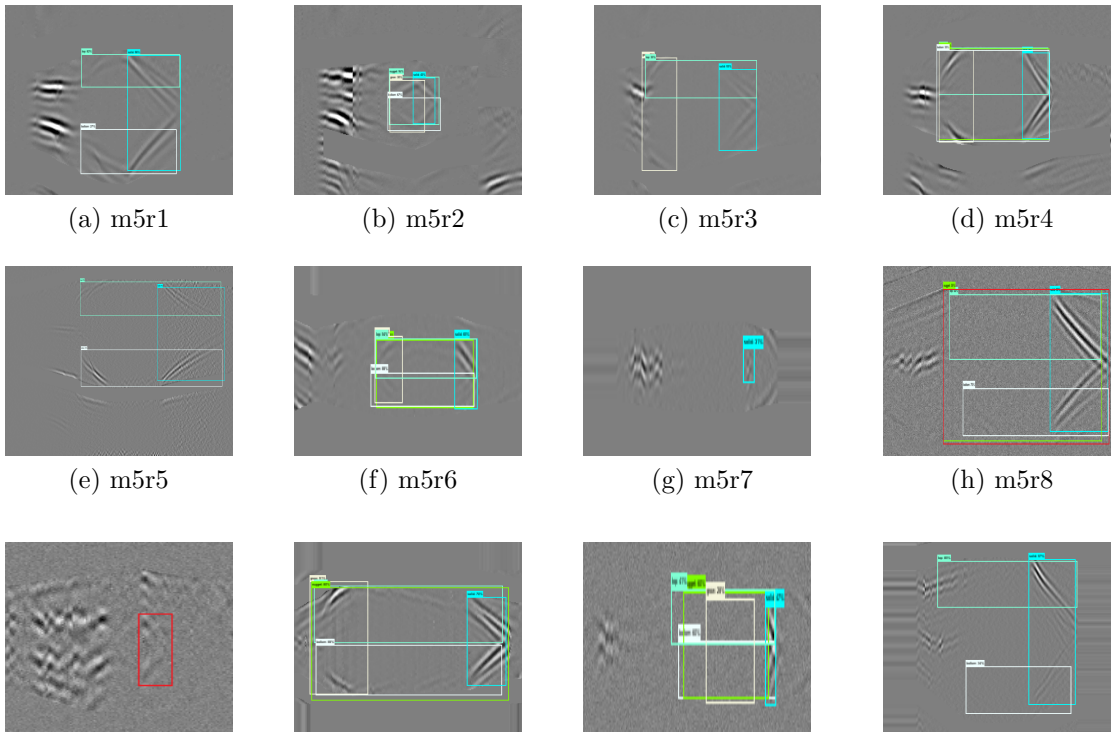


(a) m4r1 (b) m4r2 (c) m4r3 (d) m4r4

(e) m4r5 (f) m4r6 (g) m4r7 (h) m4r8

(i) m4r9 (j) m4r10 (k) m4r11 (l) m4r12

Figure 5.6: Sample results of model 4.



(a) m5r1 (b) m5r2 (c) m5r3 (d) m5r4

(e) m5r5 (f) m5r6 (g) m5r7 (h) m5r8

(i) m5r9 (j) m5r10 (k) m5r11 (l) m5r12

Figure 5.7: Sample results of model 5.

Model 5 - SSD framework with augmented set of regular b-scans without transfer learning

Fig. 5.7 shows results generated by model 5. Result shows that it misses *solid* in Fig. 5.7(i), incorrectly localize *nugget* in Fig. 5.7(h) as opposed to the ground-truth box in red and missclassifies all the classes other than *solid* in Fig. 5.7(k).

According to above results, model 4 performs best out of all the five models which has been trained on augmented dataset along with transfer learning. More comprehensive analysis of the models is included in Chapter 6.

Chapter 6

Result Analysis and Discussion

This chapter includes more comprehensive study of the performance of each of the five models. As mentioned earlier, models are evaluated based on mAP metric and inference time. In this chapter, we have reported mAP and inference time of each model and discussed the impact of using different settings for training. Finally, we have compared the weld quality analysis result provided by our best model with the result given by existing system in IDIR and reported the improvement achieved from using our deep-learning based model.

6.1 mAP Comparison

mAP is calculated on hold-out test dataset of 180 b-scans. Table 6.1 shows class-wise AP and all-class mAP for all the models. AP scores are measured at IoU threshold 0.5. According to mAP scores, model 4 is again the best one out of all five. It also has the highest AP score for detecting individual classes. One important insight is that apart from model 3 all the models show higher AP score in detecting *nugget* and *solid* than other three classes. Also, AP score for *bottom*, *top* and *grow* are the lowest in majority of the models including model 4. One reason behind this can be the large variety of aspect ratio of these classes and absence of them in majority of

Model 1

Class	AP@0.5IoU
nugget	95
solid	94
top	95
bottom	85
grow	90
mAP	92

Model 2

Class	AP@0.5IoU
nugget	96
solid	96
top	87
bottom	88
grow	91
mAP	91

Model 3

Class	AP@0.5IoU
nugget	92
solid	87
top	94
bottom	89
grow	84
mAP	89

Model 4

Class	AP@0.5IoU
nugget	**99**
solid	**98**
top	**97**
bottom	**97**
grow	**96**
mAP	**97**

Model 5

Class	AP@0.5IoU
nugget	96
solid	95
top	95
bottom	89
grow	96
mAP	94

Table 6.1: mAP of five model.

b-scans compared to *nugget* and *solid* as already shown in Fig. 5.1 which may have made the models little biased towards the latter two classes.

Classification and localization loss is another metric to evaluate model performance during training. The less the loss, the better the model has been trained to predict the ground-truth outcome. Fig. 6.1 shows a comparison of all the models for both the losses. These losses are measured on the validation set of 3000 b-scans during training, instead of hold-out test set. Models have converged at their corresponding loss values. According to the diagram, model 3 has significantly low classification and localization loss than the other, although it have not performed well on the hold-out test set as we see from detection samples and mAP scores. It is a good indication that the model is overfitted on training and validation set and as a result it could not generalize well on new images. On the other hand, model 4 has the second lowest classification and localization loss but it has showed great result in new unseen images, indicating it has generalized well.

Model	Avg. inference time (sec)
model 1	0.370
model 2	0.367
model 3	2.267
model 4	0.375
model 5	0.368

Table 6.2: Average inference time of five models.

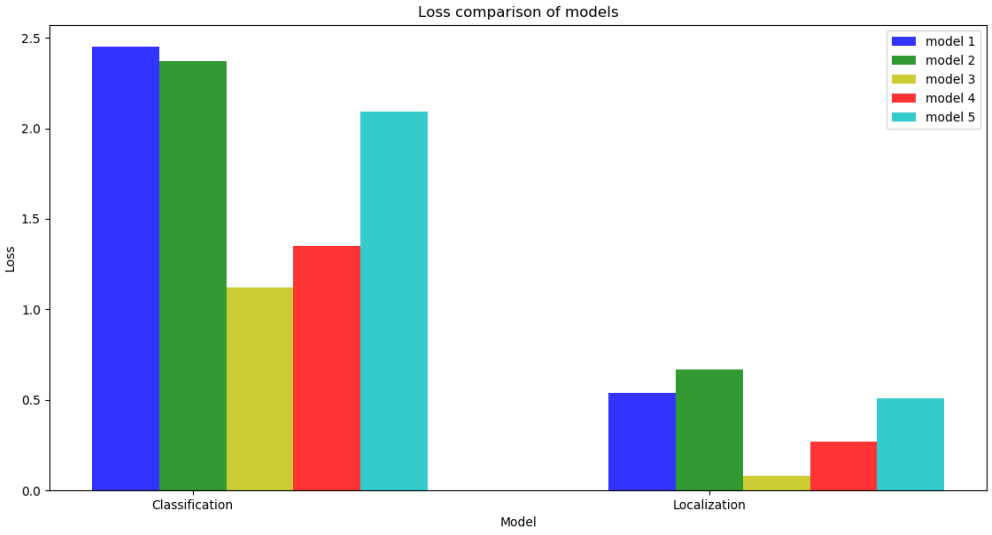


Figure 6.1: Loss comparison of five models.

6.2 Inference Time Comparison

Inference speed is very important for our work because our model has to provide feedback on the weld quality in real-time. Table 6.2 shows the average inference time taken by each of the five models for a single prediction on Intel Corei5 CPU. The lesser the time a model takes for prediction, the faster it is.

Table 6.2 shows that apart from model 3, all other models are quite fast in predicting weld classes on a single b-scan, which suffices the condition of performing in real-time. Model 3 takes too much time compared to others because of the network

architecture of Faster R-CNN, which incorporates a separate network for generating proposals or anchor boxes as discussed in Chapter 2. Use of additional region proposal network has increased the inference time to great extent. On the other hand, model 1, 2, 3 and 4 have MobileNet as their base extractor which takes up 85 to 90 percent of the total inference time and therefore they have similar time complexity. Replacing the base network with a smaller neural net will reduce inference time, however it may also reduce the model accuracy.

6.3 Observations

From the result analysis of all the five models, we have observed the following,

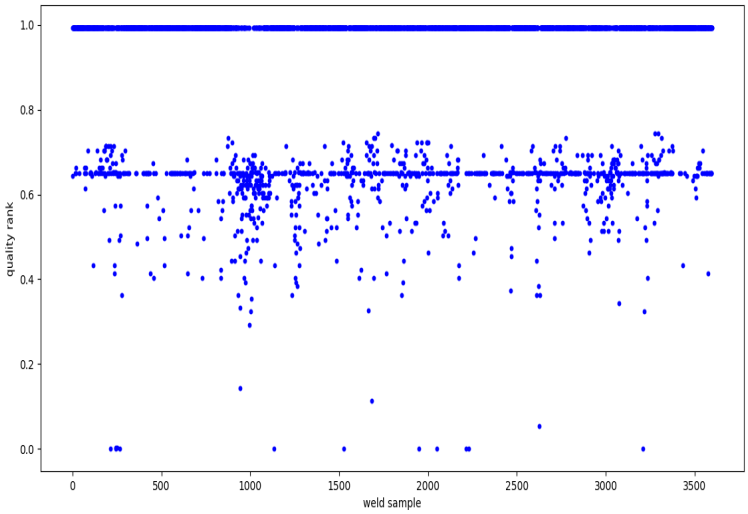
- Application of data augmentation in training set has improved the model accuracy.
- Application of transfer learning has improved the model accuracy significantly.
- Application of data augmentation along with transfer learning has provided the best performance out of all five models.
- Application of Signal Processing has not helped improve the performance of deep learning model. One probable reason behind this could be that signal processing has been done on 1-D signals of vertically stacked a-scans whereas CNN applies filter on 2-D images by sliding through horizontally. Therefore, CNN could not leverage the advantage of simplified 1-D signals.
- Faster R-CNN is not feasible for real-time application due to slow inference speed.

6.4 Comparison to Existing System

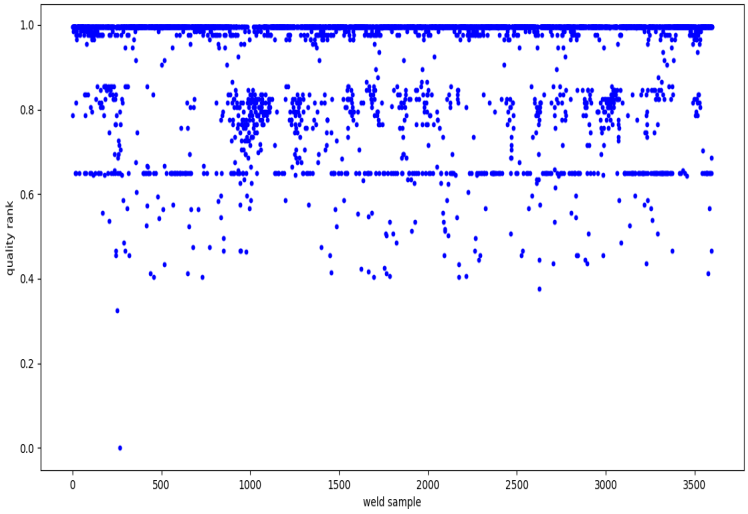
As mentioned earlier, the main motivation of incorporating our proposed model in weld quality inspection is to improve the performance of the existing model in order to achieve factory-level accuracy. Therefore, it is essential to compare our model to the existing system and measure how much it has been able to increase the accuracy. For comparison, our prototype model has been incorporated to the weld quality control software called RIWA and tested on 3600 weld b-scans under production facility in Narmco plant in Gadsden, Alabama. Fig. 6.2(a) shows the distribution of weld quality ranked by existing system as opposed to our proposed model as shown in Fig. 6.2(b). According to the two graphs, we can see that the existing system has under-performed in ranking good welds over 0.65, thus resulting in predicting more false negatives (classifies good and acceptable welds as bad). On the other hand, our proposed model has been substantially able to lift up the overall weld quality and reduce the number of wrong decisions in detecting good and acceptable weld to a great extent.

Fig. 6.3 shows a small sample of weld quality analysis result provided by our proposed model as opposed to the existing system during testing. As we can see, existing system has given 5 false decision on weld quality as shown with red cross whereas our proposed model has made only one false prediction.

We have also compared the average inference time of both the systems in same machine. Fig. 6.4 shows that our proposed model takes almost 10 times more time for providing decision on a single weld. Although, it is slower than the existing model, still the time taken by our model is acceptable to the industry due to its high accuracy. Nonetheless, our model can be optimized by reducing the complexity and replacing the very deep base network with a simpler one without dropping the accuracy.

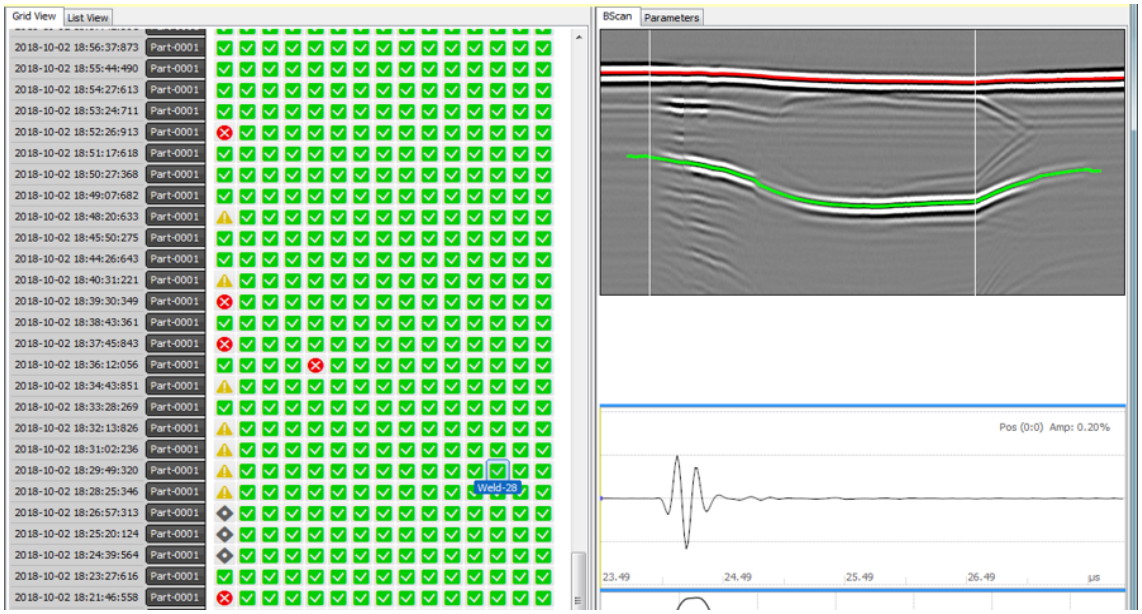


(a) existing system

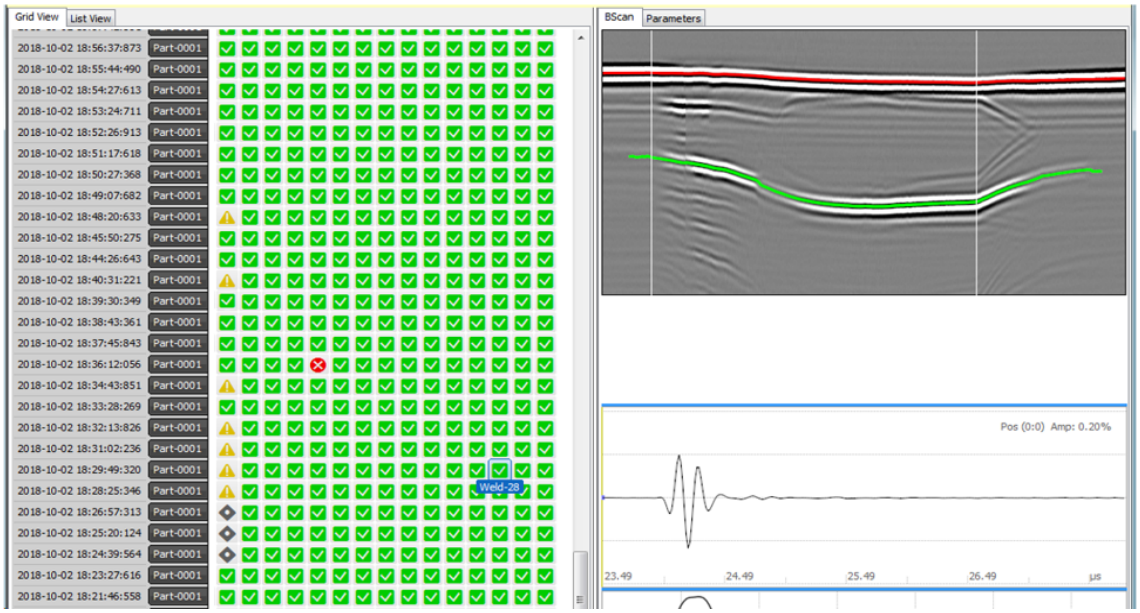


(b) proposed model

Figure 6.2: Performance comparison of existing system against our proposed model.



(a) existing system



(b) deep learning system

Figure 6.3: Sample of performance of existing system against our proposed model.

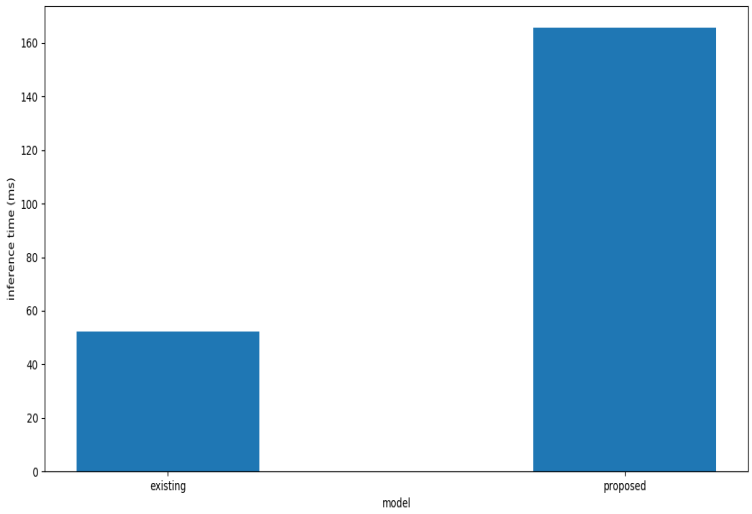


Figure 6.4: Inference time comparison of existing and our proposed model.

Chapter 7

Conclusion and Future Work

7.1 Summary of Contribution

In this thesis, our major contributions are following.

Applied deep learning for weld quality analysis

We have proposed a deep learning model to decide spot weld quality from ultrasonic b-scans which have not been tried before. Thus ours is the baseline work in this domain which can be extended to other industrial quality inspection using appropriate dataset.

Used object detection task to solve a more higher-level problem

Usually the use of object detection models are limited to detecting objects from images and notify the machine if a certain object is present in its visual space or not. However, in this work, we have utilized the information provided by object detection model and interpret them geometrically to allow the machine to take a more higher level decision of weld quality inspection.

Improved existing system significantly

Our proposed model have outperformed the existing system which was struggling to meet the industry-level accuracy. Our model has reduced the number of false decisions to great extent still performing real-time.

Compared different models

We have tested the performance of five deep learning models which are quite different by incorporating various methodology like data augmentation, signal processing, transfer learning etc. This extensive comparison has helped us to understand the impact of these different methods have on our data.

7.2 Future Work

As mentioned earlier, our proposed model can be optimized by implementing a much simpler architecture. Along with changing the network itself, some type of hybrid method of feature-engineering can also be incorporated to reduce the complexity in our data. Feature-engineering may include generating custom anchor boxes for different regions of the b-scan divided by current on and off position.

Bibliography

[1] Chertov A. and Maev R. Extraction of straight line segments from noisy images as a part of pattern recognition procedure. *Advances in Signal Processing for NDE of Materials*, 2004.

[2] R. Gr. Maev A. M. Chertov, A. C. Karloff. Improvements in pattern recognition on noisy ultrasonic b-scans, 2011.

[3] Bo Chen Dmitry Kalenichenko Weijun Wang-Tobias Weyand Marco Andreetto Andrew G. Howard, Menglong Zhu and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[4] E. Bendersky. The softmax function and its derivative. `https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/`, 2016.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[6] Remi Cadene. *Deep Learning for Visual Recognition*. PhD thesis, Master's thesis, Sorbonne Universits, Paris, France, 2016.

[7] Howard B Cary and Scott C Helzer. *Modern Welding Technology*. 2005.

[8] Karloff A. C. Perez W. Lui A. Maev R. G. Chertov, A. M. In-process ultrasound nde of resistance spot welds. insight - non-destructive testing and condition monitoring. pages 257–261, 2012.

[9] C. Cortes and V. Vapnik. Support vector machine. *Machine Learning*, 20(3):273–297, 1995.

[10] A. Toshev D. Erhan, C. Szegedy and D. Anguelov. Scalable object detection using deep neural networks. *CVPR*, 2014.

[11] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[12] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[13] H. El Khiyari and H. Wechsler. Face recognition across time lapse using convolutional neural networks. 7:141–151, 2016.

[14] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[15] The American Society for Non-destructive Testing. Introduction to non-destructive testing. `https://www.asnt.org/MinorSiteSections/AboutASNT/Intro-to-NDT`, 2017.

[16] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. Dssd : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017.

[17] R. Girshick. Fast r-cnn. *ICCV*, 2015.

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

[19] Frigui H. and P. Gader. Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-

nearest neighbor classifier. *Fuzzy Systems, IEEE Transactions*, 17(1):185199, 2009.

[20] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*, IWANN '96, pages 195–201, London, UK, UK, 1995. Springer-Verlag.

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

[22] Suzana Herculano-Houzel. The human brain in numbers: A linearly scaled-up primate brain. *Frontiers in human neuroscience*, 3:31, 11 2009.

[23] Marcia Hon and Naimul Mefraz Khan. Towards alzheimer's disease classification through transfer learning. *CoRR*, abs/1711.11117, 2017.

[24] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[25] Jonathan Hui. What do we learn from region based object detectors (faster r-cnn, r-fcn, fpn)? `medium.com/@jonathan_hui/what-do-we-learn-from-region-&based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9`, 2018 (Accessed: 2018-12-25).

[26] Lars Hulstaertj. A beginner's guide to object detection. `https://www.datacamp.com/community/tutorials/object-detection-guide`, 2018 (Accessed: 2018-12-25).

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

[28] R. Girshick J. Redmon, S. Divvala and A. Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[29] M. Kearney K. Basques. Analyze runtime performance. `https://developers.google.com/web/tools/chrome-devtools/rendering-tools/`.

[30] S. Ren K. He, X. Zhang and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell*, 37(9):19041916, 2015.

[31] Andrej Karpathy. "neural networks part 1: Setting up the architecture." notes for cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/neural-networks-1/`, 2015 (Accessed: 2018-12-25).

[32] Andrej Karpathy. "transfer learning." notes for cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/transfer-learning/`, 2015 (Accessed: 2018-12-25).

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[35] Philip J. Stimac Lance E. Besaw. Deep convolutional neural networks for classifying gpr b-scans, 2015.

[36] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[37] Danqing Liu. A practical guide to relu. `https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7`, Accessed: 2018-12-25.

[38] Chertov A. M. and Maev R. Gr. Inverse problem solution to find real-time temperature distribution inside the spot weld medium using ultrasound time of flight methods. *Quantitative Nondestructive Evaluation*, 2003.

[39] Chertov A. M. and Maev R. Gr. Determination of resistance spot weld quality in real time using reflected acoustic waves. comparison with through-transmission mode. *16th World Conference on Nondestructive Testing*, 2004.

[40] Chertov A. M. and Maev R. Gr. A one-dimensional numerical model of acoustic wave propagation in a multilayered structure of a resistance spot weld. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 52(10), 2005.

[41] Chertov A. M. Regalado W. P. Tchipilko A. Lichaa P. Clemente D. Phan T Maev, R. G. In-line inspection of resistance spot welds for sheet metal assembly. pages 58–62, 01 2014.

[42] K. Nasrollahi and T. B. Moeslund. Haar-like features for robust real-time face recognition. In *2013 IEEE International Conference on Image Processing*, pages 3073–3077, Sep. 2013.

[43] Andrew Ng. Cs229 lecture notes. `http://cs229.stanford.edu/notes/cs229-notes1.pdf`, Accessed: 2018-12-25.

[44] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[45] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015 [Online].

[46] R. Sakaguchi P. A. Torrione, K. D. Morton and L. M. Collins. Histograms of oriented gradients for landmine detection in ground-penetrating radar data. *IEEE Trans. Geosci. Remote Sens.*, 52(3):15391550, 2014.

[47] Lefvre S Pham, M. Buried object detection from b-scan ground penetrating radar data using faster-rcnn. 2018.

[48] Minh-Tan Pham and Sébastien Lefèvre. Buried object detection from b-scan ground penetrating radar data using faster-rcnn. *CoRR*, abs/1803.08414, 2018.

[49] T. Darrell R. Girshick, J. Donahue and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[50] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *CVPR*, 2017.

[51] R. Girshick S. Ren, K. He and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[52] SciPy.org. scipy.signal.hilbert. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.hilbert.html#scipy-signal-hilbert`, Accessed: 2018-12-25.

[53] Sefik Ilkin Serengil. Hyperbolic tangent as neural network activation function. `https://sefiks.com/2017/01/29/hyperbolic-tangent-as-neural-network-activation-function/`, Accessed: 2018-12-25.

[54] J Sprovieri. Resistance spot riveting. https://www.assemblymag.com/articles/93677-resistance-spot-riveting, 2017 (Accessed: 2018-06-23).

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

[56] R. B. Girshick K. He B. Hariharan T.-Y. Lin, P. Dollar and S. J. Belongie. Feature pyramid networks for object detection. *CVPR*, 2017.

[57] techminy.com. Resistance spot welding working process its application. http://techminy.com/resistance-spot-welding, 2017 (Accessed: 2018-06-23).

[58] P. Torrione and L.M. Collins. Texture features for antitank landmine detection using ground penetrating radar. *IEEE Transactions on Geoscience and Remote Sensing*, 45(7):2374–2382,, 2007.

[59] Tzutalin. Labelimg. https://github.com/tzutalin/labelImg, 2015 (Accessed: 2018-12-25).

[60] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 1:II, 2001.

[61] D. Erhan C. Szegedy S. Reed C.-Y. Fu W. Liu, D. Anguelov and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.

[62] Wikipedia. Analytic signal. https://en.wikipedia.org/wiki/Analytic_signal, Accessed: 2018-12-25.

[63] Wikipedia. Fourier transform. https://en.wikipedia.org/wiki/Fourier_transform, Accessed: 2018-12-25.

[64] Wikipedia. Hilbert transform. https://en.wikipedia.org/wiki/Hilbert_transform, Accessed: 2018-12-25.

[65] Wikipedia. Jaccard index. https://en.wikipedia.org/wiki/Jaccard_index, Accessed: 2018-12-25.

[66] Wikipedia. Mean squared error. https://en.wikipedia.org/wiki/Mean_squared_error#cite_ref-pointEstimation_1-0, Accessed: 2018-12-25.

[67] Wikipedia. Nondestructive testing. https://en.wikipedia.org/wiki/Nondestructive_testing, Accessed: 2018-12-25.

[68] L. Wong. *Capacitive Micromachined Ultrasonic Transducers for Non-destructive Testing Applications*. PhD thesis, PhD thesis, University of Waterloo, ON, Canada, 2014.

[69] Y. Bengio Y. LeCun and G. Hinton. Deep learning. 521(7553):436444, 2015.

[70] J. Sun et al. Y. Li, K. He. R-fcn: Object detection via region-based fully convolutional networks. *NIPS*, 2016.

[71] J. Li Y. G. Jiang Y. Chen Z. Shen, Z. Liu and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. *ICCV*, 2017.

[72] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018.

Vita Auctoris

NAME:	Zarreen Naowal Reza
PLACE OF BIRTH:	Dhaka, Bangladesh
EDUCATION:	Bachelor of Science in Computer Science and Engineering, BRAC University, Dhaka, Bangladesh, 2016.
	Master of Science in Computer Science, University of Windsor, Windsor, Ontario, Canada, 2019.