DETECTING HEAD IN PILLOW DEFECT (HIP) BY USING DEEP LEARNING AND IMAGE PROCESSING TECHNIQUE

BY

TAN WEI JIN

SUPERVISED

BY

DR. AUN YICHIET

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2021

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: DETECTING HEAD IN PILLOW DEFECT (HIP) BY USING DEEP LEARNING AND IMAGE PROCESSING TECHNIQUE

Academic Session: JAN 2021

I TAN WEI JIN

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
57, Taman Nyaman Indah,
Lorong Ponduk Upih 3, 11000,
Balik Pulau, Pulau Pinang

Aun Yichiet
Supervisor's name

Date: 16 April 2021

Date: 16 April 2021

DETECTING HEAD IN PILLOW DEFECT (HIP) BY USING DEEP LEARNING AND IMAGE PROCESSING TECHNIQUE

BY

TAN WEI JIN

SUPERVISED

BY

DR. AUN YICHIET

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2021

DECLARATION OF ORIGINALITY

I declare that this report entitled "**DETECTING HEAD IN PILLOW DEFECT (HIP) BY USING DEEP LEARNING AND IMAGE PROCESSING TECHNIQUE**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _________________________

Name : TAN WEI JIN

Date : 16 April 2021

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr Aun Yicheit who has given me this bright opportuning to engage in an AI development project with the collaboration of ViTrox Technologies Sdn. Bhd. A million thanks to you for all the support and advice to make this project possible.

Besides that, I would like to thank my Vitrox supervisor, Mrs Lim Lay Ngor, my mentor Ms. Ooi Yi Fong in sharing their experience on dealing with the HIP defects which gave me some insight in dealing with the issues I faced in this project. Also would like to thank Mr Lim Seng Yew for his approval and final review in this FYP collaboration with ViTrox.

Last but not least, I would also like to thank my friends and family for their support and everlasting encouragement throughout the entire project. I would also like to thank Vitrox Technologies Shd. Bhd. for providing all the datasets and equipment used in this project. This project is under Memorandum of Argreement with Vitrox Technologies Sdn. Bhd.

ABSTRACT

Deep learning is an Artificial Intelligence (AI) method that mimics the ways human brain processing data and recognizes the data or objects. It is a subset of machine learning which utilizes the hierarchical level of artificial neural networks (ANN) to perform the process of machine learning (Hargrave, 2019). Deep learning has very great potential of wide adoption in various industries. In fact, deep learning has already been used by corporations and start-ups such as Google, Facebook, Amazon, Tesla etc for several different tasks such as filtering fake news, analysing shopping trends and developing self-driving cars. In the manufacturing sectors, deep learning techniques were usually used to aid the engineers or inspectors in making decisions in the production line or the quality inspection phase. However, there are still various reasons why deep learning was not largely implemented in the manufacturing sector especially in the detection of Head in Pillow (HIP) defects that occurred in the Ball Grid Array (BGA) of a printed circuit board (PCB). This project aims to design a robust deep learning model that could be implemented to speed up and ease the process of detecting the HIP defects. The 3 Dimensional (3D) Convolutional Neural Network (CNN) will be the foundation of the deep learning model which will deal with the grayscale BGA slice images that were stacked together. The outcome of the project will be a robust deep learning model that could classify the HIP defects on BGA joints in greyscale which have not more than 9 slices. Over 200 of 3D CNN models with different hyperparameters and architecture are created in this project to achieve the objectives of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

MB	Megabyte
ms	millisecond
T_g	Glass transition temperature

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
AXI	Automated X-ray Inspection
BGA	Ball Grid Array
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DNN	Deep Neural Network
e.g.	Exempli gratia
et al	And others
etc	Et cetera
FC	Fully Connected
FYP	Final Year Project
GPU	Graphics Processing Unit
HIP	Head-In-Pillow
HOP	Head-On-Pillow
IDE	Integrated Development Environment
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LSTM	Long Term Short Memory
n.d.	No date
NLP	Natural Language Processing
OS	Operating System
p./pp.	Page/Pages
PCB	Printed Circuit Board
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
ROI	Region of Interest
SGD	Stochastic Gradient Descent
VGGNet	Visual Geometry Group Neural Network

Chapter 1 Introduction

1.1 Problem Statement and Motivation

With the developing use of leadless soldering and reduction of size and pitch of solder joints in gadgets producing, the Head-in-Pillow (HIP) defects which also known as the ball-and-socket is a solder joint defect had become a common issue during the Ball Grid Array (BGA) assembly. The HIP defects could be caused by several issues such as the oxidation of BGA ball or the oxygen barrier in solder pastes individually/ jointly, warpage, misalignment of the components, et cetera. (Chen et al., 2014). A printed circuit board (PCB) with HIP defects will result in joints with sufficient association of electrical integrity yet missing the adequate machinal strength. This conceivably expensive imperfection isn't generally distinguished in practical testing and would just appear as a disappointment in the field after the components are exposed to some physical or thermal stress as they could barely withstand the stress strength due to insufficient mechanical strength (Seelig, 2008).

Despite various ways or precaution steps such as verified and measure the print definition and the print height consistency before the solder paste are applied onto the solder bump with the print definition, using a square or rounding opening aperture with excessive print volume reducing time above the glass transition temperature (Tg) and ensuring minimum delta temperature difference between BGA components and the rest of the components on the PCB which suggested by Alpha Assembly Solutions (n.d., p.2) to reduce the occurrence of the HIP defects, the inspection for HIP defects in BGA is still unavoidable even though the defects ratio is 0 for almost all of the time as it would be a loss to the company if there are defects board produced.

Moreover, the time required for HIP defects inspection could range from seconds to half an hour depending on the PCB board size, the components size, and the number of joints. Speed is a factor that could affect the overall productivity and the efficiency in manufacturing. Thus, the HIP defects inspection speed should be fast enough so that the following inspections are not delayed. Furthermore, the accuracy on detecting the HIP defects with the Automated X-ray Inspection (AXI) machine currently is not ideal/ perfect as many defects and good joints were classified wrongly by the machine. Therefore, the inspectors would have to manually inspect the false calls from the machine again with the help of a real-time X-ray viewing machine. This process is necessarily as by viewing the 2D images captured from the AXI machine is insufficient for decision making. Thus, the BGA HIP defects inspection process would be decelerated which would cause a delay to the following inspections.

1.2 Project Scope

The aim of the project is to build a 3D neural network that could receive BGA images and separate or classify to two classes which are HIP defects or non-defects and an image stacking technique. The model should be able to receive up BGA joints with not more than 9 slices that are in grayscale.

1.3 Project Objectives

The objectives of this project are:

i. To design a 3D CNN for BGA HIP defect detection
ii. To design an image stacking technique using salient layers based on domain heuristics
iii. To retrain BGA dataset on the proposed 3D CNN model to improve upon existing Manufacturer 2D CNN architecture
iv. To optimize a 3D CNN model that could inference the BGA HIP defects in less than 200ms per inference.

1.4 Impact, significance, and contribution

With the help of the 3D CNN model, the time required for the HIP defects inspection could be reduced significantly. In fact, the inference speed is expected to be at least 200ms or below per inference. Since the 3D CNN is expected to have an accuracy of 99%, the false calls from the AXI machines are expected to reduce drastically too. With a lesser false calls rate, the manual inspection process is expected to be sped up and the workload of the inspector could be reduced as the use of the real-time X-ray viewing machine could be reduced. Assuming there are 1, 000 BGA joints per inspections, one single inspection could be done in just under 5 mins with 10 false calls leftover at max which required manual inspection with the help of the real-time X-ray viewing machine. The result obtained is significant, as this feat had not been able to be achieved by any previous reviewed literature. Besides that, the image stacking technique could help understand which slices are more important in detecting the HIP defects for this particular dataset.

1.5 Background information

1.5.1 Introduction to image processing

Image processing usually consists of several stages, image importation, image analysis, image manipulation and image output. There are two methods of image processing which is digital image processing and analogue image processing. In analogue image processing, the processing is done on the two-dimensional (2D) analogue signals. Generally, the images such as television images which are manipulated by electricity are used in this image processing method. On the contrary, digital image processing usually deals with digital images which are matrices of small pixels and elements. The images are manipulated through software and algorithms to solve different tasks such as image detection, image reconstruction, image restoration, compression, enhancement, and etc. Since digital image processing has a wider range of applications, digital image processing has dominated over analogue image processing as the time goes by. There are a few major techniques of digital image processing such as image editing which altered the image through graphic software tools, independent component analysis which separates the multivariate signal into additive subcomponents, pixilation which refers to turning printed images into digitized images such as GIF, and etc. (Adoriasoft, 2017)

1.5.2 Convolutional Neural Network (CNN)

According to 'Understanding of Convolutional Neural Network (CNN) – Deep Learning written by Prabhu (2018):

> Convolutional neural network (CNN) is one the deep neural networks which are widely used for image recognition, image classification, objects detections, face recognition, etc. The concept of the convolutional networks was inspired by the connectivity pattern between neurons that assemble the organization of animal visual cortex. The cortical neurons are triggered by different stimuli and only a restricted region of the visual field known as the receptive field will respond to it. When there is a presence of a collection of reception fields overlapping each other that cover the entire visual area, a vision will be formed.

When CNN model is trained/ tested, the input images will have to pass through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and a SoftMax function that normalized the output of the network to a probability distribution with probabilistic values between 0 and 1 as shown in Figure 1.5.2.1.

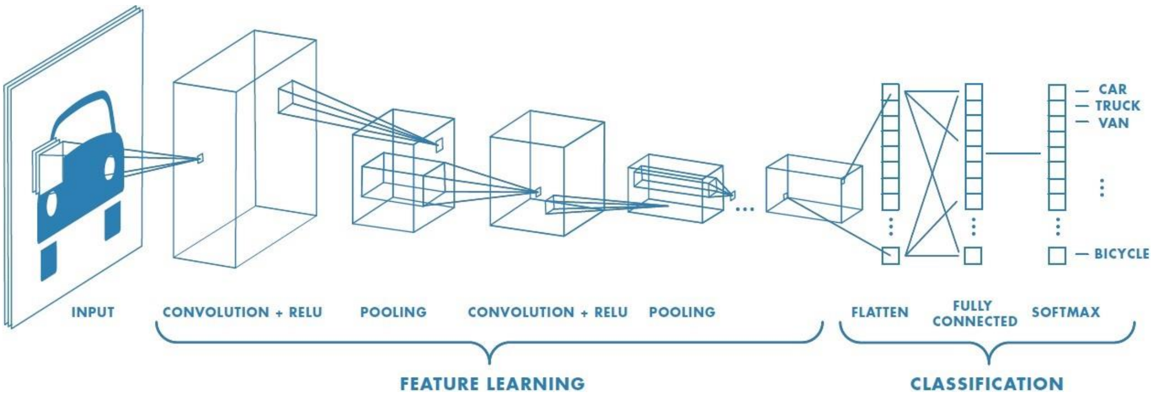


Figure 1.5.2.1 Overview of a convolution neural network on classifying an image

The convolution layers refer to layers that extract the features of the input images which is actually a mathematical operation that requires two inputs, input matrix and a filter/kernel as shown in Figure 1.5.2.2. The relationship between the pixels of the images are preserved as convolution learns the image features with the help of small squares of input data. Operations such as edge detection, blurring, sharpening could be done with convolution by applying different filters/ 'Feature Map' on the images.

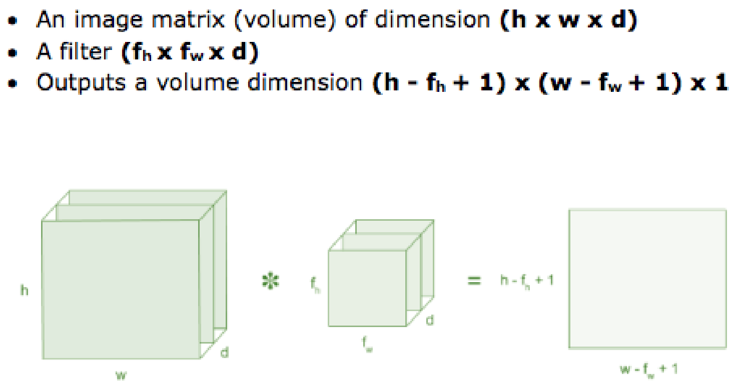


Figure 1.5.2.2 Image matrix multiplies with kernel to obtain the output matrix

Padding will be done on the input matrix when the filter does not fit the input images perfectly. There are two type of padding which is the zero-padding and also the valid padding. Whenever zero-padding is applied, zeros will be padded to the image until the filter could fit the image perfect. On the other hand, valid padding will cause any part of the images that the filter could not fit in to be dropped and only valid parts of the image remain. In addition, Rectified Linear Unit (ReLU) will also be introduced at this stage of the model to pump in some non-negative linear values to the matrix. ReLU is chosen instead of other non-linear functions such as tanh or sigmoid and is prefered by most of the data scientists as ReLU outperforms most of the non-linear functions.

After the input matrix goes through iterations of convolutions, it will be arrived at the pooling layer and the input matrix parameters will be cut down if the matrix is too large. Pooling is referred to a sample-based discretization process where the matrix will be subsampled or down sampled without discarding the important information in the matrix. There are a few different types of pooling such as max or min pooling, average pooling and also sum pooling. In max pooling, the largest element in the rectified feature map will be chosen while in min pooling, the smallest element in that particular rectified feature map will be chosen.

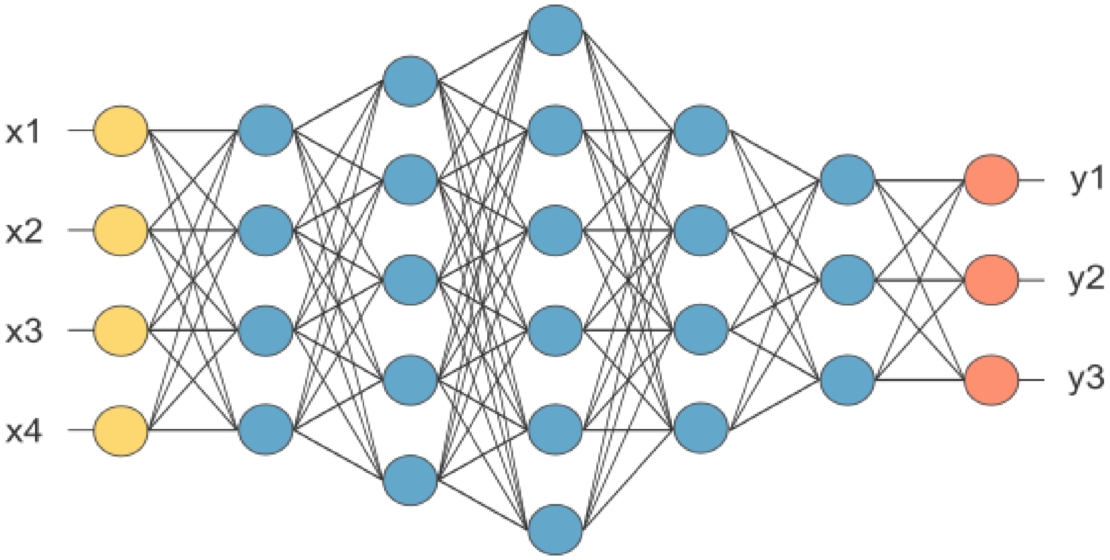


Figure 1.5.2.3 Fully Connected (FC) Layer

Finally, the pooled matrix will enter the fully connected (FC) layer where it would be flattened into a vector and being fed into a neural like network. The filter matrix will be converted as a vector (e.g. x1) as shown in Figure 1.5.2.3 and the features will be combined to form a model. The SoftMax function will then classify the output into specific classes.

1.5.3 Deep Neural Network (DNN)

The name neural network is a derivation from the neural connections of the human brain. The basic building blocks of a neural network are known as neurons which is similar to the biology term neuron which is the basic working unit of the brain which is a specialized cell that was designed to transmit information from a nerve cell to the other nerve cells, muscle cells or the gland cells. The neurons in neural networks can be said as a mathematical function or method which takes in inputs and produces a specific number of outputs. These functions which are contained within the neurons are generally referred as activation functions.

When these artificial neurons are arranged in together in a specific way, a layer is formed. These layers could then be stacked together in desired ways to form a neural network. The output from the previous layer of neurons would act as the inputs and will be fed into the following or next layer for further processing. This process forms a complex chain which impersonates the inner workings of a human brain. A neural network with more layers is usually more complex and more powerful to solve a more complex problem. However, the number of the layers should be determined based on the complexity of the problem.

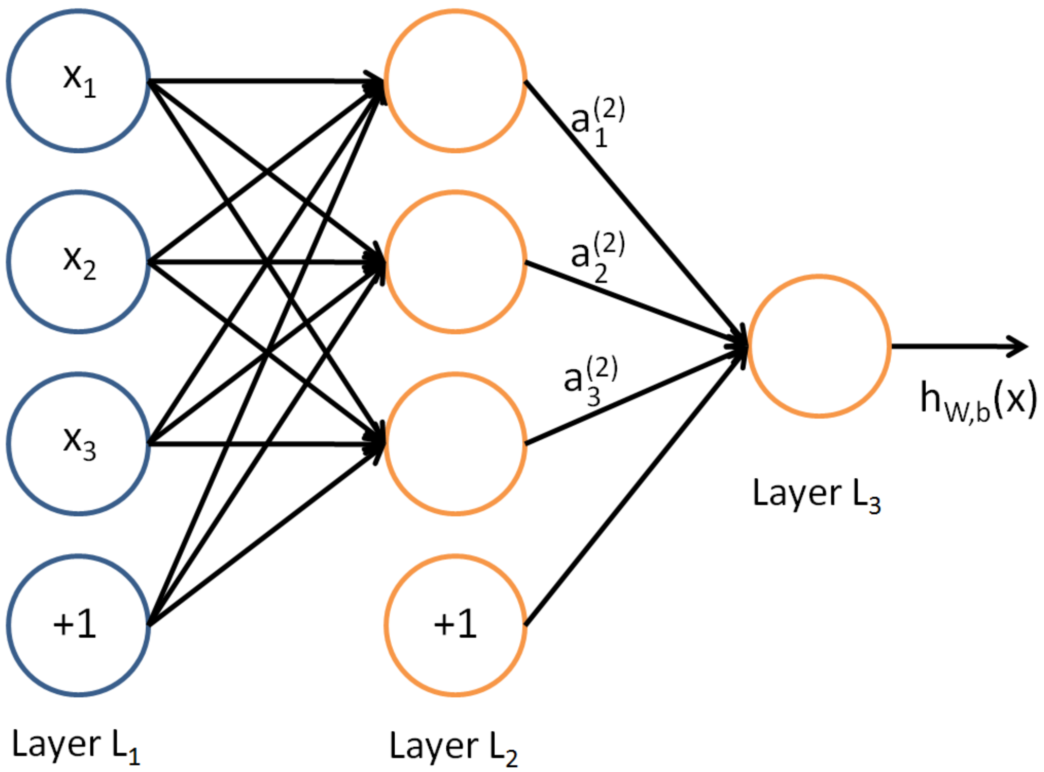


Figure 1.5.3.1 Example of layers in Convolutional Neural Network (CNN)

Based on Figure 1.5.3.1, the input layer, L1 which is located at the leftmost layer will be receiving input from the previous layer or the raw input. These data will be passed along the neural network and the rightmost layer will be outputting the prediction. The layers between the output layers and the input layers are known as the hidden layer as it computes those intermediate values that are invisible throughout the training phase. A Deep Neural Network (DNN) could be formed by having multiple or more than one hidden layer.

1.5.4 Ball Grid Array (BGA) Head-In-Pillow (HIP) Defects

The Ball Grid Array (BGA) Head-In-Pillow (HIP) or Head-On-Pillow (HOP) defects is a solder joint defects that refers to a phenomenon which occurred when pre-deposited solder ball on the chip and the solder paste applied to the circuit board does not join together even though they are melted. The name HIP derived from the distinct boundary between the solder ball on the chip and the solder paste on the circuit board which is somewhat similar to a head resting on a pillow.

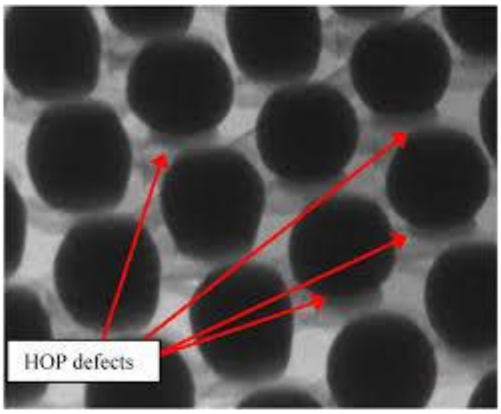


Figure 1.5.4.1 Grayscale HIP defects in 3D

Figure 1.5.4.1 shows the image of the BGA and the solder joint with HIP defects that will normally be seen in the real-time X-ray machine. This machine is usually used by manufacturers for board inspection after the AXI machines classified the joints. The difference between this machine and the AXI machines is that AXI machine can do auto inspection while manual inspection had to be done by the inspector when the real time machine is used. However, the real-time X-ray machine could generate images of the board in different angels much faster compare to the AXI machines as the AXI machines are required to inspect all the components that are present on the PCB. Despite the image generation process being much faster for the real-time X-ray machine, inspection had to be done manually and usually this machine would not be used if defects inspection could be done with the help of images generated by the AXI machines.

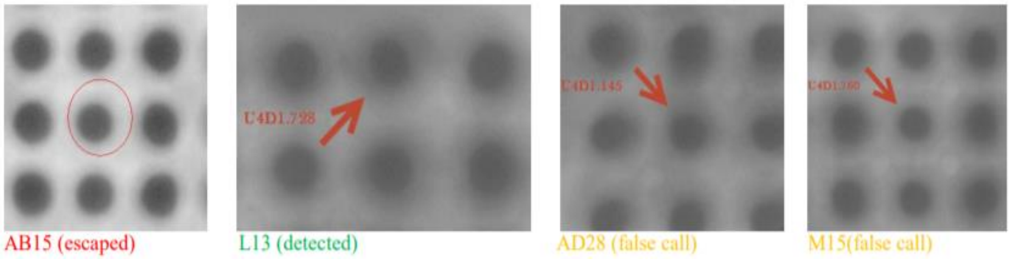


Figure 1.5.4.2 Grayscale HIP defects slices

The slice image of the BGA based on the offsets that are predefined by the user before the machine starts to inspect the board. Based on Figure 1.5.4.1, it is very hard for the inspectors to classify the defects from the good BGA joints. This is because there are various environmental issues that might cause the slice images to be blurred or unclear.

1.6 Proposed approach

The models with the best result obtained are built with 2 3D convolution layers, 2 3D batch normalization layers, 2 3D max pooling layers, 1 1D batch normalization layer, 1 dropout layer with 2 fully connected layers. Leaky ReLU activation function is used in this model instead of the common ReLU activation function. Dummy images are padded to those BGA images that have slices less than 9 to create an input volume of 9x224x224.

1.7 Highlight of what has been achieved.

A total of 244 models are trained in this project and the highest accuracy achieved on 3 different combinations of datasets are 86.90%, 95.41% and 93.41%. These models are able to classify BGA images with a number of slices of 5,6 and 9. However, the models are able to accept BGA images with different number of slices which are not more than 9 as those images will be padded with dummy images to 9 slices before the input enters the model for classification. A special padding technique where the dummy images are padded in front of the BGA slice images instead of padding it after the slice images are used in this project. Moreover, the models could also inference a BGA joint in less than 200ms. Significant improvement could also be seen as compared to the 2D CNN model produced by Manufacturer.

1.8 Report Organization

The research of this project will be discussed in the following chapter. Channel-Wise Pre-processing method, some model architectures and review on convolution layers could be expected in the following chapter. The proposed model architecture, tools and technologies used in this project, details on the datasets used in this project and the project timeline is located in chapter 3. Hyperparameter used, model summary, result of the preliminary models and also the evaluation of the preliminary models could be found in chapter 4. Experiment and Evaluation of the models on 3 different combinations of datasets could be seen in Chapter 5. The comparison of the proposed 3D CNN models and the Manufacturer 2D CNN models could also be observed in the same chapter. Chapter 6 of the report will be in charge of concluding the whole report. Steps and ideas for improving the model will also be stated in Chapter 6.

Chapter 2 Literature Review

2.1 Channel-Wise Pre-Processing method

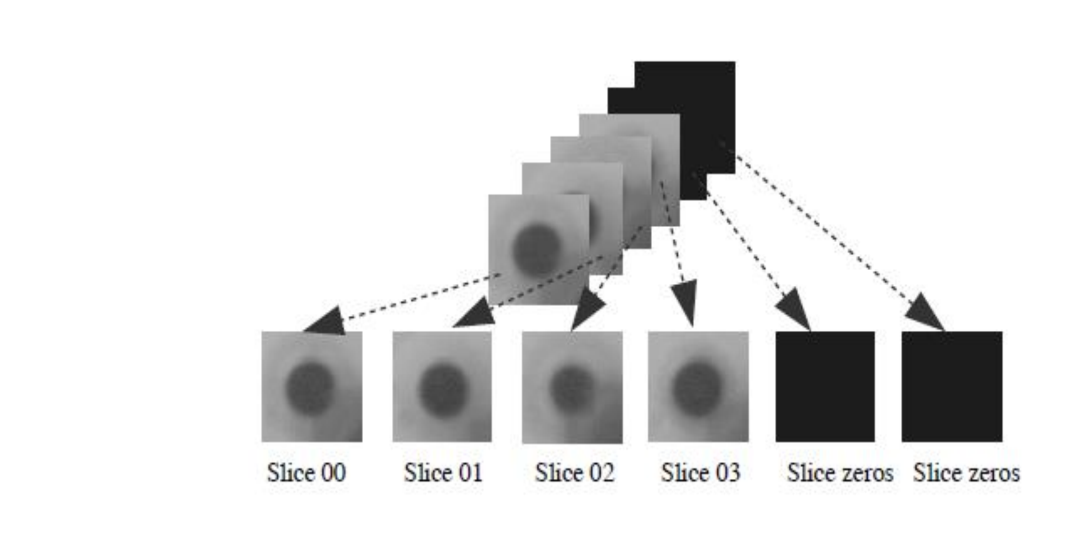


Figure 2.1.1 Channel-wise pre-processing output

The channel-wise pre-processing method proposed by Zhang et al. (2020) is similar to a pipeline process where the accumulated instructions are executed in an expected order. According to Zhang et al. (2020), the channel-wise pre-processing method is able to process raw solder joint X-ray images and output them into six channels as shown in Figure 2.1.1. Due to the inconsistent number of BGAs' X-ray imaging slices and the fact that the region of interests (ROIs) for the BGAs might not be as accurate as expected when the values are applied on the images, the channel-wise pre-processing method could help in addressing such problems. The channel-wise pre-processing method will deal with the inconsistency of BGA's X-ray imaging slices first by applying both deep and shallow depth of the imaging or either of them. Subsequently, the zero-slices will be used to pad those joints that do not have 6 slice images to the maximum number of slices which is six. A ROI based cropping will then be implemented to the image to segment the joints out from the image that might consist of other surrounding solders or components. The cropped images should only be focused on one single solder joint instead of having additional noise in it. Each individual slice of the solder will then be sent to 6 different channels accordingly for training the model.

2.2 Residual Neural Network (ResNet)

The residual neural network (ResNet) is one of the variants of deep convolutional neural networks proposed by He et al. (2015) that won the 1st place of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 organized by ImageNet. The Resnet is proposed by the authors as another method in addressing the degradation problem that occurred when more layers are used in the neural network. Although the problem were largely addressed before the emergence of ResNet through normalized initialization and intermediate normalization layers which enable the networks with an additional of tens of layers to converge the stochastic gradient descent (SGD) with backward-propagation, ResNet is able to surpass the other methods in term of optimization and accuracy from considerably increased depth.

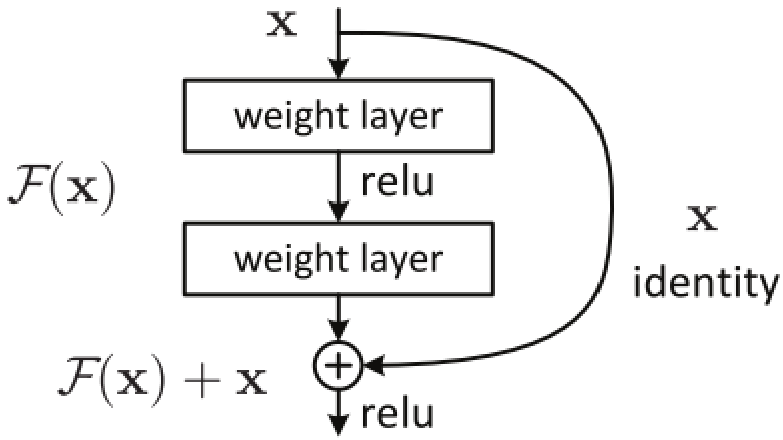


Figure 2.2.1 A residual block of ResNet

According to “An Overview of ResNet and its Variants” by Fung (2017), the core idea of the ResNet is the introduction of “identity shortcut connection” that enables the skipping of multiple layers as shown in Figure 2.2.1. Thus, instead of hoping the stacked layers directly fit into a desired underlying mapping, the stacked layers will be fitted into a residual mapping with the help of residual blocks. However, the Highway Network proposed by Srivastava et al. (2015) cited in Fung (2017) which introduce the concept of gated shortcut connections which control the volume of information that are allowed to flow across the shortcut which is similar to the Long Term Short Memory (LSTM) cell proposed by Hochreiter (1997) in terms of concept where the parameterized forget gate is used to controls the flow could not performs as well as the

ResNet. This is something extraordinary as the solution space of the Highway Network contains ResNet, but it could not even perform as good as the ResNet could.

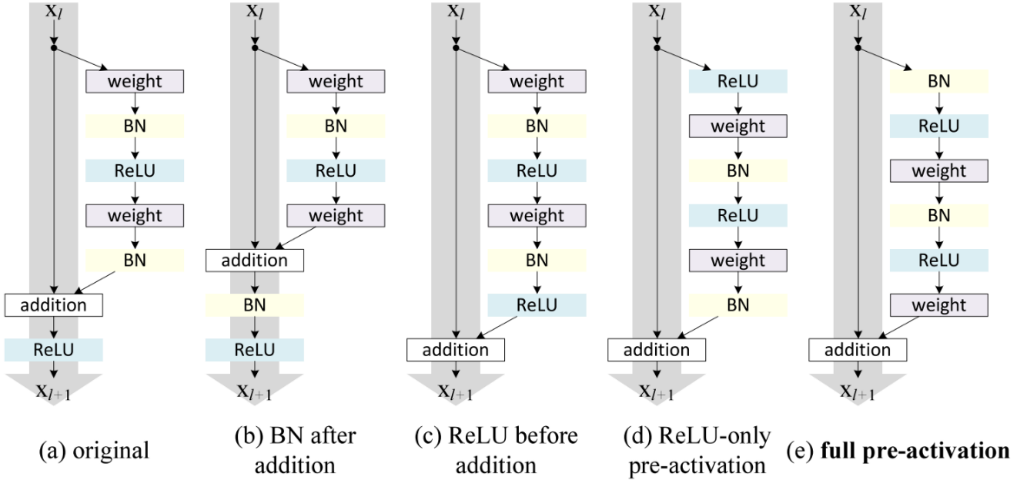


Figure 2.2.2 Variants of residual blocks

A pre-activation variant of residual block is then introduced by He et al. (2017) where the gradients could now flow through the shortcut connections to any other previous layers unimpededly. With the help of the pre-activation variant residual block, the Resnet-110 which contains only 110 of pre-activation variant residual layers is able to outperform a ResNet-1202 that has more than 10 times that of the layers in Resnet-110. However, the residual layers in ResNet-1202 are non-pre-activation variant residual layers as in the Resnet-110. He et al. (2017) also further demonstrated that a Resnet-1001 with 1001 of pre-activation variant residual layers is able to outperform its counterpart which have lesser layers. This proof that stacking of residual layers will further improve the performance of the neural networks and ResNet is further escalated as one of the widely implemented deep learning architectures in various computer vision projects.

2.3 Visual Geometry Group Neural Network (VGGNet)

In 2014, Simonyan and Zisserman (2015) from the Visual Geometry Group, Department of Engineering Science, University of Oxford proposed the VGGNet which is one of the most remarkable CNNs of that year. Even though VGGNet is the 1st runner-up instead of the winner of the ILSVRC 2014 in classification task, VGGNet is still able to defeat GoogLeNet the winner of ILSVRC 2014 on the localization task. The VGGNet not only had a significant improvement over the ZFNet which is the winner of the ILSVRC 2013 and the winner of the ILSVRC 2012, AlexNet, VGGNet is one of the deep learning models that is able to obtained an error rate of under 10% in 2014.

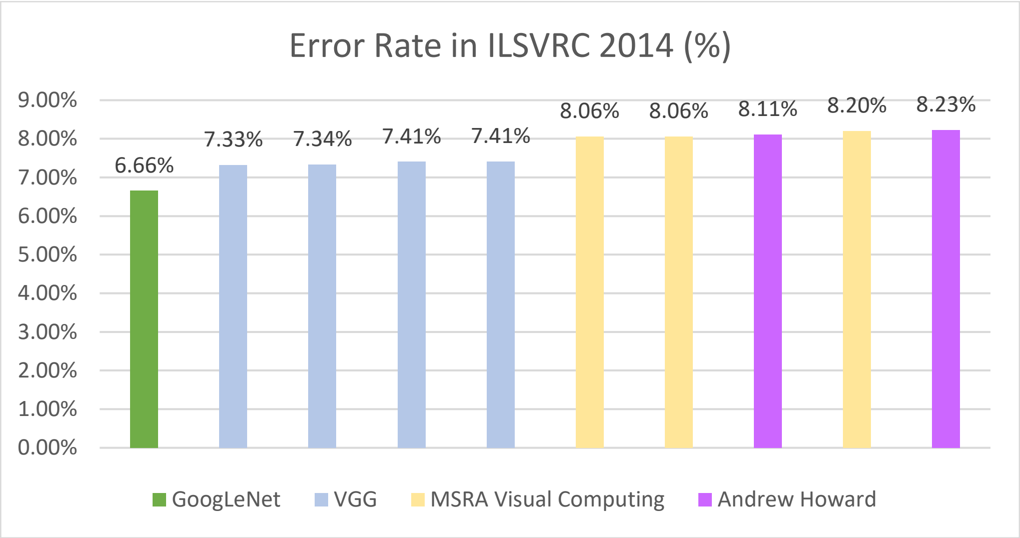


Figure 2.3.1 Error Rate of models in ILSVRC 2014

Instead of directly addressing the vanishing gradient problem faced by CNNs as convolution layers in the neural network increased, Simonyan and Zisserman (2015) proposed VGGNet to address it in another way. Instead of using larger filter matrices, multiple layers of 3 x 3 filter matrices could help in converging the neural network faster and reducing the overfitting problem. Multiple layers of small filters are able to reduce the number of parameters which is better for faster convergence and reduce overfitting problems of the models. For example, 3 layers of 3 x 3 filters are able to reduce 45% of the parameters of the input matrix with a similar effective area of a 7 x 7 filter (Tsang, 2018).

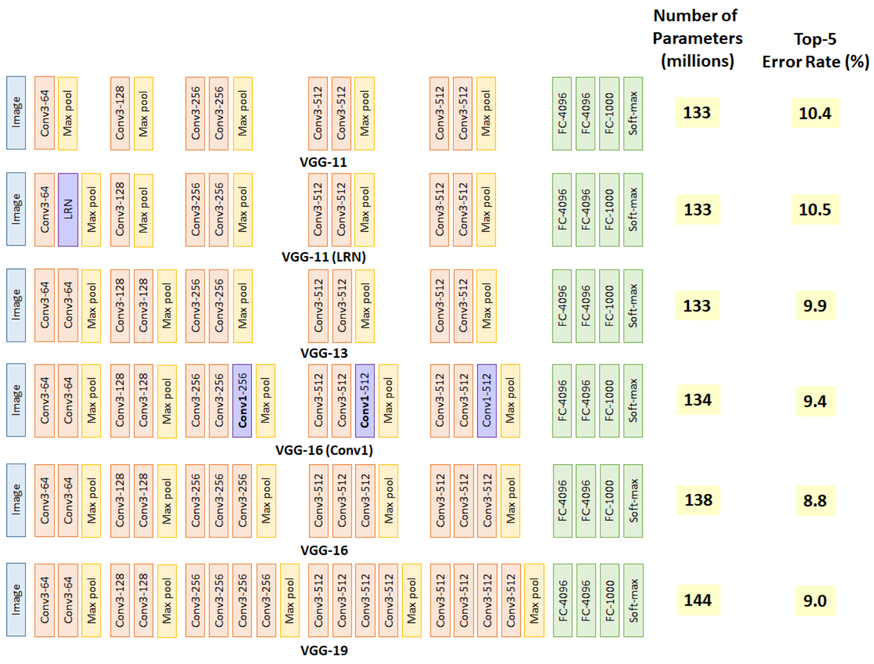


Figure 2.3.2 Different VGG Layer Structures using single scale (256) Evaluation

With the concept of multiple smaller filters are better than one big filter, an ablation study is done by the author on VGGNet (Tsang, 2018). Based on Figure 2.3.2, we can see that the performance of the model is increasing as convolution layers increase. However, the result obtained from VGG-19 is contradicted with the statement stated above, as the model had started to converge when 16 convolution layers were added to 19 convolution layers. This is where multi-scale training and testing come into play.

The phrase multi-scale is referring to multiple sizes / dimensions of the images. When the neural network is trained with images with the same scale/ size, the model's performance might seem very good in classifying images with the same scale as the training data. However, when an image with a larger scale as compared to the training images, the model might be biased towards the training data and classify that larger scale images wrongly. Thus, training a neural network with multi-scale images is necessary. During a multi-scale training or testing, an image is scaled with smaller-size equal to a range from 256 to 512 and then the image will be cropped to 224 x 224 before it is used for training or testing. The results obtained by Simonyan and Zisserman showed that VGGNets with multi-scale training or/ and multi-scale training are performing much better than networks with single-scale training and testing.

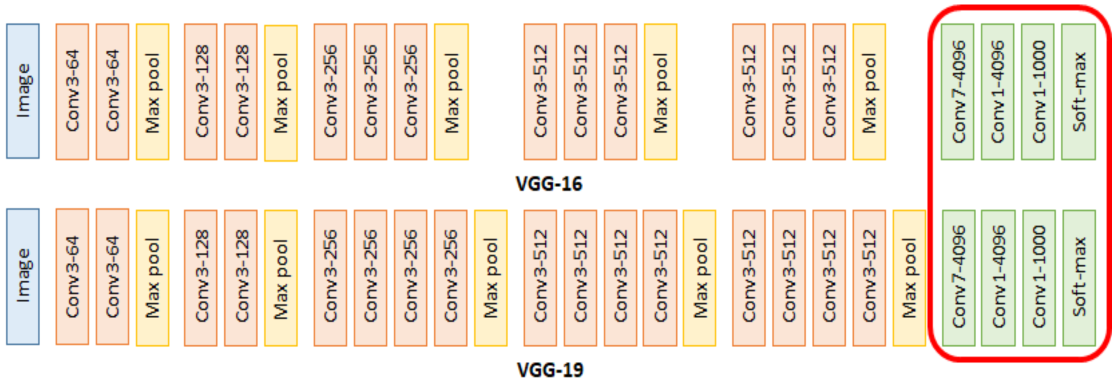
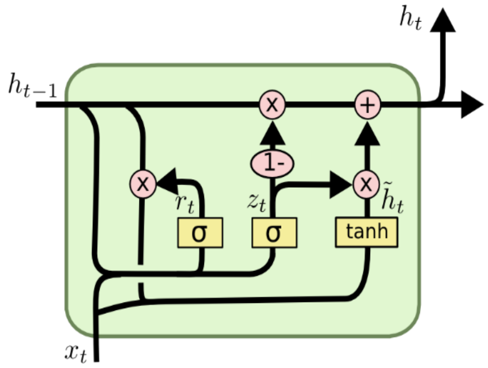


Figure 2.3.3 VGGNet During Testing

AlexNet introduced us to the concept of multi-cropping during testing which could increase the accuracy of the models. The idea of multi-cropping is referred to multiple cropping of corners, centre and the horizontal flips of the images. The cropped images will then be outputted as a probability vector which will be added or averaged and act as additional features to the models to obtain a better result. In VGGNet testing the concept of replacing the first fully connected layer (FC) with a 7x7 convolutional layer while replacing the second and third FC layers with 1 x 1 convolutional layers is known as convolutionalized/ dense testing. With the concept of multi-cropping and dense testing, the performance of the VGGNets are improved. In order to further improve the VGGNets, fusion of concepts as mentioned in the past few paragraphs is involved. By combining multi-scale training, multi-scale testing, multi-cropping dense as well as VGG-16/ VGG-19, an accuracy of 93.2% could be achieved.

2.4 Long Short-Term Memory (LSTM) Networks

Recurrent neural networks (RNNs) are one of the neural networks that are able to recognize previous state sequences and utilize them to get a better result. Yet, RNNs are also facing the vanishing gradients problems during back-propagation (Wenninger et al. 2015) which is similar to the problems faced by CNNs. Long Short-Term Memory networks which are usually known as LSTMs are variants of the RNNs which have the capability of learning long-term dependencies are implemented to solve the vanishing gradients problem faced by RNNs. This concept of LSTM network was first introduced by Hochreiter and Schmidhuber (1997) and was further refined and popularized by other people who are in the computer vision field.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.4.1 Underlying concepts of LSTMs' memory cells

According to Olah (2015):

> By replacing the hidden nodes of RNNs with memory cells containing input, out and forget gates that have the ability to control the flows of the information of the cells, LSTMs are able to alleviate the vanishing gradients problem faced by RNNs. In addition, the memory cells are able to modify the data stored within them. The gradient of value 1 is recurred in the connections between the memory cells helps to prevent the gradient from vanishing even though the model is backward propagated from time to time.

2.5 Types of Convolutions

There are many different types of convolutions including 1D convolutions, 2D convolutions, 3D convolutions, dilated convolutions, transposed convolutions etc. 2D and 3D convolutions are the most common convolutions types that were applied in various Convolutional Neural Networks (CNN) for image or object recognition.

1D convolutions are the most simplistic convolutions that are usually used on sequential datasets. Extracting the 1D subsequence from the input sequences and identifying the local patterns within the window of convolution is one of the major usages of the 1D convolutions. 1D convolutions are usually used in Natural Language Processing (NLP) where each sentence is represented as a sequence of words. Figure 2.5.1 shows how a 1D convolution filter is applied to a sequence of data to obtain new features from it.

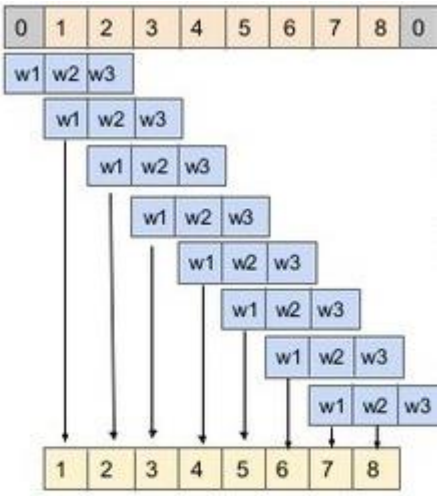


Figure 2.5.1 1D convolutional filtering

2D Convolutions are the most common type of convolutions as it had been widely used in CNN architectures on image datasets. In the 2D convolutions, the filter moves in a 2-directions (x, y) to compute the features from the spatial dimensions. The output shape of a 2D convolutions layer will be a 2-Dimensional matrix.

In 3D CNNs, convolutions are applied on a 3-dimensional filter that moves 3-direction (x, y, z) instead of 2-dimensional filter that moves 2-direction (x, y) to compute the low-level feature representations. The output shape of the 3D convolution layers is a 3-dimensional volume space such as cube or cuboid. 3D convolutions are useful in event detection in videos, 3D medical images, etc and they are not limited to 3D space only, 3D convolution could also be applied to 2D space inputs such as images.

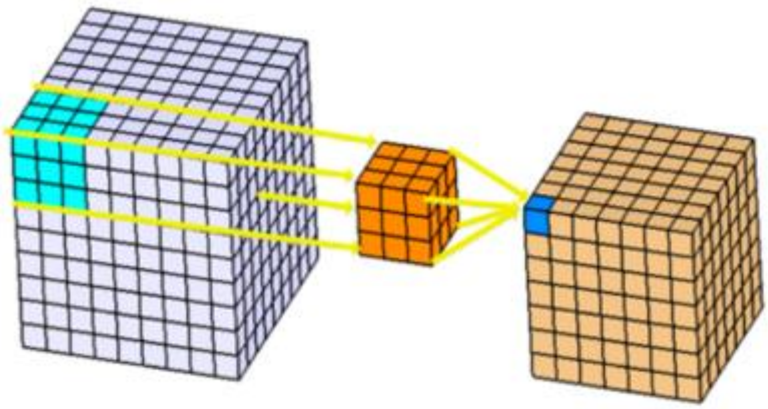


Figure 2.5.2 3D convolutions demonstration

Chapter 3 Proposed Method/ Approach

3.1 Methodology and General Work Procedures

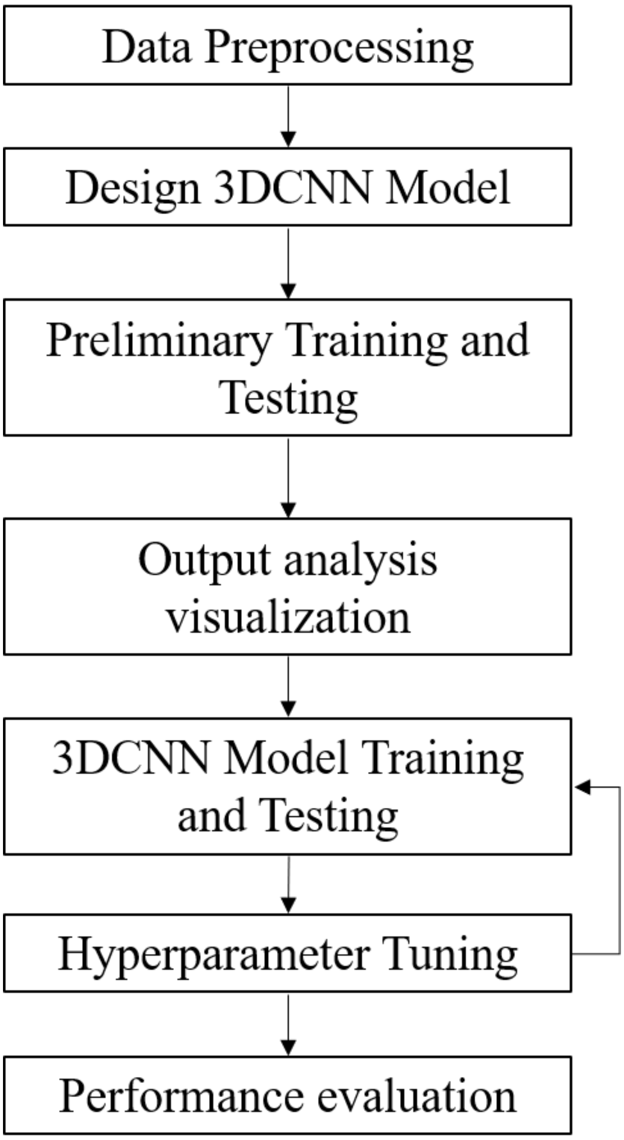


Figure 3.1.1 Research Methodology

During the early phase of the project methodology, datasets were received, and pre-processing is done on the image sets. The pre-processing of datasets includes, splitting the image sets into training and testing set, ensuring that each slice images of the joint are available and stacking the slice images together in various combinations.

Once the datasets are split and stacking combinations are planned, the 3DCNN model will be designed based on existing literature as references with some modification to fit the datasets. During the preliminary phase, only part of the datasets will be trained with the model. For instance, the datasets with 9 slice images per joint will be prioritized first in the preliminary phase. The reason 9 slice BGA images are prioritized first is because the amount of 9 slices BGA images are larger compared to 5 slices or 11 slices BGA images. The models will be trained with the training data that were prepared during the pre-processing phase, 20% of the training data will be used to validate the model performance first before testing them on the testing data. Amendments such as hyperparameter tuning and stacking combination were made to get a more desirable result or outcome.

Once the results for 9 slice BGA images are acceptable, the models are trained with different slices of BGA images at once. Some modifications were done on the models for it to accept input with different numbers of slices. When dealing with input that doesn’t have a total of 9 slices, dummy images were padded to the slices so that the input could fit the model. The models were trained with a combination of training data on stressing the model’s limit. This particular step is used to ensure the model is not overfitting the training data which will cause the test accuracy to drop when facing new data. In the final phases of the project methodology, amendments such as hyperparameter tuning and stacking combination were made to get a more desirable result or outcome in order to achieve the objective of the project.

3.2 Tools and technologies used

Python is selected as the programming language of choice for the proposed deep learning model in this project. Pytorch which is a Python-based scientific computing package that targeted audiences who wish to get a replacement for Numpy to use the power of GPUs and a deep learning research platform that provides maximum flexibility and speed will be used in this project. The Pytorch library is used for building the 3DCNN while the matplotlib library is used for plotting the graphs and also creating the confusion matrix for model evaluation. Other libraries such as the os, Numpy and Pandas libraries will be used as well in this project. Jupyter Notebook which is generally used for developing open-source software, open-standards, and services for interactive computing across multiple programming languages will be the Integrated development environment (IDE) for this project.

Hardware details:

OS	Windows 10 Pro 64-bit
CPU	Intel®Xeon®Silver 4210
RAM	128.0 GB
Storage	1.0 TB

Table 3.2.1 Hardware details

Python library version:

Pandas	1.1.1
Numpy	1.19.1
PIL	7.2.0
Pytorch	1.2.0
Torchvision	0.4.0
Sklearn	0.23.2

Table 3.2.2 Python library version

3.3 System Design / Overview

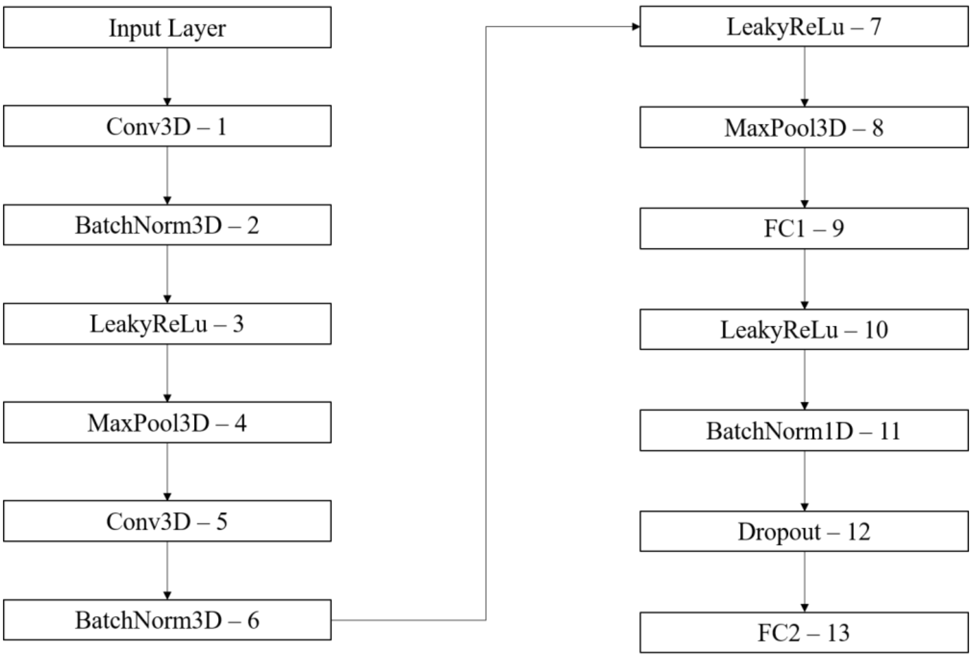


Figure 3.3.1 Proposed 3D CNN Architecture

Figure 3.3.1 provides the general overview of the proposed 3D CNN. This CNN consists of 2 3D convolution layers, 2 3D Batch Normalization layers, 2 Max Pooling layers, 4 LeakyRelu layers, 1 1D Batch Normalization layer, 1 Dropout layer and 2 fully connected layers. During the training phase, augmentation such as random flipping and random rotating would be done on the data with the help of torch vision transforms library. Data augmentation was done on the training data to increase the features captured by the model. Besides augmentation, the training data would also be shuffled thus the sequence of training data during the forward propagation of the model for each batch would be different. No data augmentation would be done during the validation phase where the validation data are actually part of the training data that were sampled for validating the model. During the testing phase, the testing data will be used to test the model's performance. All three data used in this project are individual BGA joints that have no repeated occurrence in three of the data. Fine tuning such as adding a fully connected layer, changing the number of neurons in the fully connected layer, changing the number of filters in the convolution layers, using a different optimizer, adding initialization to the fully connected layers, etc. were done to improve the model's performance.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1     [-1, 32, 10, 225, 225]             288
       BatchNorm3d-2     [-1, 32, 10, 225, 225]              64
         LeakyReLU-3     [-1, 32, 10, 225, 225]               0
         MaxPool3d-4      [-1, 32, 5, 112, 112]               0
            Conv3d-5      [-1, 64, 6, 113, 113]          16,448
       BatchNorm3d-6      [-1, 64, 6, 113, 113]             128
         LeakyReLU-7      [-1, 64, 6, 113, 113]               0
         MaxPool3d-8        [-1, 64, 3, 56, 56]               0
            Linear-9                  [-1, 128]      77,070,464
        LeakyReLU-10                  [-1, 128]               0
      BatchNorm1d-11                  [-1, 128]             256
          Dropout-12                  [-1, 128]               0
           Linear-13                    [-1, 2]             258
================================================================
Total params: 77,087,906
Trainable params: 77,087,906
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.72
Forward/backward pass size (MB): 502.93
Params size (MB): 294.07
Estimated Total Size (MB): 798.72
----------------------------------------------------------------
```

Figure 3.3.2 Model summary

Models that performed best on three different combinations of datasets respectively are created with the same model architecture. The model summary of the models is shown in Figure 3.3.2. Even though three of the models have the same architecture but the hyperparameters used in each of the models are slightly different to each other.

3.4 Implementation Issues and Challenges

There are several issues that arise during the preliminary phase of the project. Due to the slicing feature provided to the AXI machines, there is a chance that HIP defects are not present in any of the slices for the particular joint. The accuracy of the model might be influenced by this issue as the label for that particular joint is incorrect based on the interpretation of the slice images.

Moreover, the size of the model is one of the factors that would influence the inference speed. The larger the size of the model, the longer the inference speed will be. Thus, the number of layers and the number of neurons in the fully connected layers must be controlled even though models with more layers are usually more suitable for such complicated project. Besides that, more resources will be allocated to the model when the size of the model is huge. This would cause lesser resources being allocated to the datasets which might cause error due to insufficient memory during the training phase.

Furthermore, due to various environmental issues, the BGA slice images might be blur, unclear or even shady. Additional pre-processing might be required to be done on the datasets before they were used for training and testing the model. Besides that, more time are required for standardizing the datasets which would reduce the time available for model training and testing.

Besides that, the models could only receive BGA images with slices ranging from 1 to 9. This is because the input layer of the model is fixed to receive 9 slices only. BGA images with slices less than 9 will be padded to 9 slices with dummy images. This limits the ability of the model to classify BGA joints with slices more than 9. Padding is the main issue that causes this problem as the expected number of slices had to be configured first before the number of dummy images required to be padded could be determined.

Moreover, the resolution of BGA images is limited to 224 x 224 only because images with different resolutions will result in different activation volumes produced by the convolution layer. A different activation volume would not fit the fully connected layer as the volumes are hardcoded and should be consistent. In order to enable the model to received images with different resolution, a global pooling could be used to replace the first fully connected layer. However, the results produced by the models with global pooling layer are not as good as the models with 2 fully connected layers. Therefore, the models are created with 2 fully connected layers instead of 1 global pooling layer and 1 fully connected layer.

Lastly, due to the variance in the number of BGA slices, experiments must be done on the various stacking combinations in order to optimize the output of the model. Time allocation for each of the sections is very crucial or extremely important for this project as there are various factors that would delay the progression of the project.

3.5 Dataset

Recipes	Good Joint	Defects	Number of Slices	Slice Sequences
A	84	68	5	Pad s2 Mid slice s1 Chip
B	368	319	9	s6 Pad s5 s4 s3 Mid slice s2 s1 Chip
C	329	239	6	s3 Pad s2 s1 Mid slice Chip
D	12	2	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5
E	62	49	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5

F	185	159	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5
G	166	159	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5
H	381	264	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5
I	419	334	9	s6 Pad s5 s4 s3 Mid slice s2 s1 Chip
J	175	239	6	s3 Pad s2 s1 Mid slice Chip
K	282	126	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5
L	124	14	9	s6 s4 Chip s3 Mid slice s1 Pad s2 s5

Total	2587	1972

Table 3.5.1 Datasets

There is a total amount of 12 datasets that will be used for training and testing in this project. These datasets are directly collected from the AXI machines. The total number of BGA joints in this dataset is 4,559 joints with 2,587 good joints and 1,972 defect joints. There are 3 different slices settings which are 5, 6 and 9.

Combination	Recipes	Good Joint	Defects	Train	Validate	Test
A	A, B, C, D, E, F, G	1206	995	1326	333	542
B	A, B, C	781	626	898	226	283
C	A, B, C, H	1162	890	1137	397	518
D	I, J, K, L	1000	713	1084	280	349

Table 3.5.2 Combinations

Table 3.5.2 shows the combination of data that will be used to train, test and validate the models. Combination B is created because data in Recipe D, E, F and G are having shading issues that would affect the result of the model. The statement is proven as the model trained with combination B has a significant improvement in terms of accuracy. Recipe H is actually the recaptured and processed version of Recipe D, E, F and G. Since the setting of the AXI machine is different, result return by the machine will be different thus the total amount of data in combination A and B are different. Therefore, combination C is created. Model trained, validated and tested with combination C are slightly better than results obtained with combination A however, the accuracy couldn’t surpass the results obtained with combination B. This is quite odd as the images in Recipe H are better than those in Recipe D, E, F, G. In order to investigate this issue, a decision to standardize the capturing method and processing method is done on the other Recipes too. This result in the production of combination D. Recipe I is the recaptured and processed version of Recipe B, Recipe J is the recaptured and processed version of Recipe C, Recipe K is the recaptured and processed version of Recipe D and E while Recipe L is the recaptured and processed version of Recipe F and G. In other words, combination D was the recapture and processed version of combination A with the exclusion of Recipe A. The processing method used in Recipe I, J, K and L are similar to Recipe H so that this would not be a factor that will affect the result obtained by the models trained, validated and tested with this combination. Results show that the accuracy did improve using combination D however it does not exceed the accuracy obtained with combination B. Due to time constraint, the project had to be put to and end thus no more combinations and investigations are produced and done to solve the issue. Combination A, B, D will be the main focus in this project as the images in combination C are not standardized.

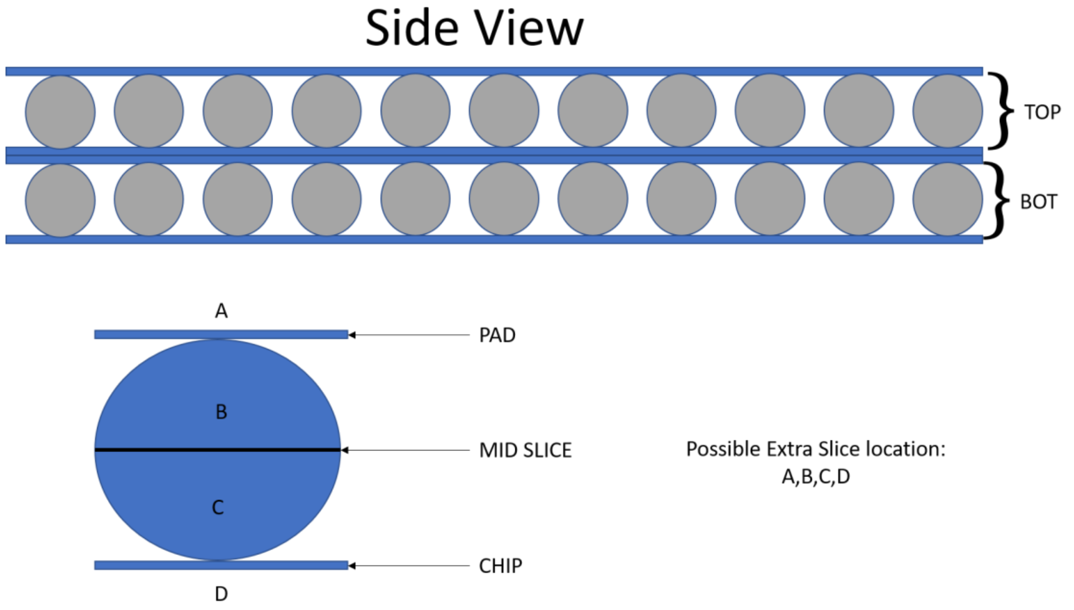


Figure 3.5.1 BGA details

Figure 3.4.1 shows the possible slicing location of the BGA. The sequence of the pad, Mid slice and chip is based on the location of the BGA. The pad slice will come first if the BGA is located on top of the chip. On the contrary, the chip slice will come first if the BGA is located below the chip. The default slicing setting will be pad, Mid slice and chip as HIP occurred more frequently in these few locations. The slicing setting is not fixed to the settings shown in the datasets. Users of the AXI machine could increase the number of slices that would be produced per BGA joint to increase the chances of detecting the HIP defects.

3.6 Timeline

TASK NAME	START DATE	END DATE	DAYS COMPLETE
Literature Review			
Research on 3D Convolution Neural Network	26-Oct-20	22-Nov-20	28
Dataset Preparation			
Dataset pre-processing	23-Nov-20	25-Nov-20	3
Design 3D Convolution Model			
Preliminary training	25-Nov-20	29-Nov-20	5
Preliminary testing	26-Nov-20	29-Nov-20	4
Output analysis visualization	27-Nov-20	29-Nov-20	3
Model training and testing	18-Jan-21	28-Feb-21	42
Hyperparameter tuning	25-Jan-21	28-Feb-21	35
Performance evaluation	1-Feb-21	28-Feb-21	28
Miscellaneous			
Bug-fixing	1-Mar-21	28-Mar-21	28
Documentation restructuring	29-Mar-21	11-Apr-21	14
Report Writing			
FYP Report 1	30-Nov-20	6-Dec-20	7
FYP Report 2	12-Apr-21	16-Apr-21	5

Table 3.6.1 Timeline

Research on the 3D convolution neural network (CNN) had been done this semester before receiving the datasets. After receiving the dataset, some analysis had been done to understand the datasets. Pre-processing steps such as labelling, splitting, and stacking had been done to split the datasets to training and testing sets. After a preliminary 3DCNN model is designed and built based on existing literature as references, the model is trained and tested with a portion of the datasets. Evaluation is done based on the training loss and the accuracy of the model on predicting the testing sets. This pretty much wrapped up the progression of the project in phase 1.

In the second phase, more hyperparameter tuning is done on the models to produce a better result before it has any contact with other portions of the data. After the data are combined with the combination as mentioned in Table 3.5.2, the models are trained, validated and tested. Results are recorded and more hyperparameter tunings are done to the models. Some time was spent for bug-fixing during the hyperparameter tuning and model architecture redesigning phase. The project is halted as restructuring and editing are required to be done on the FYP report 1 to fit it into the format specified for the FYP report 2.

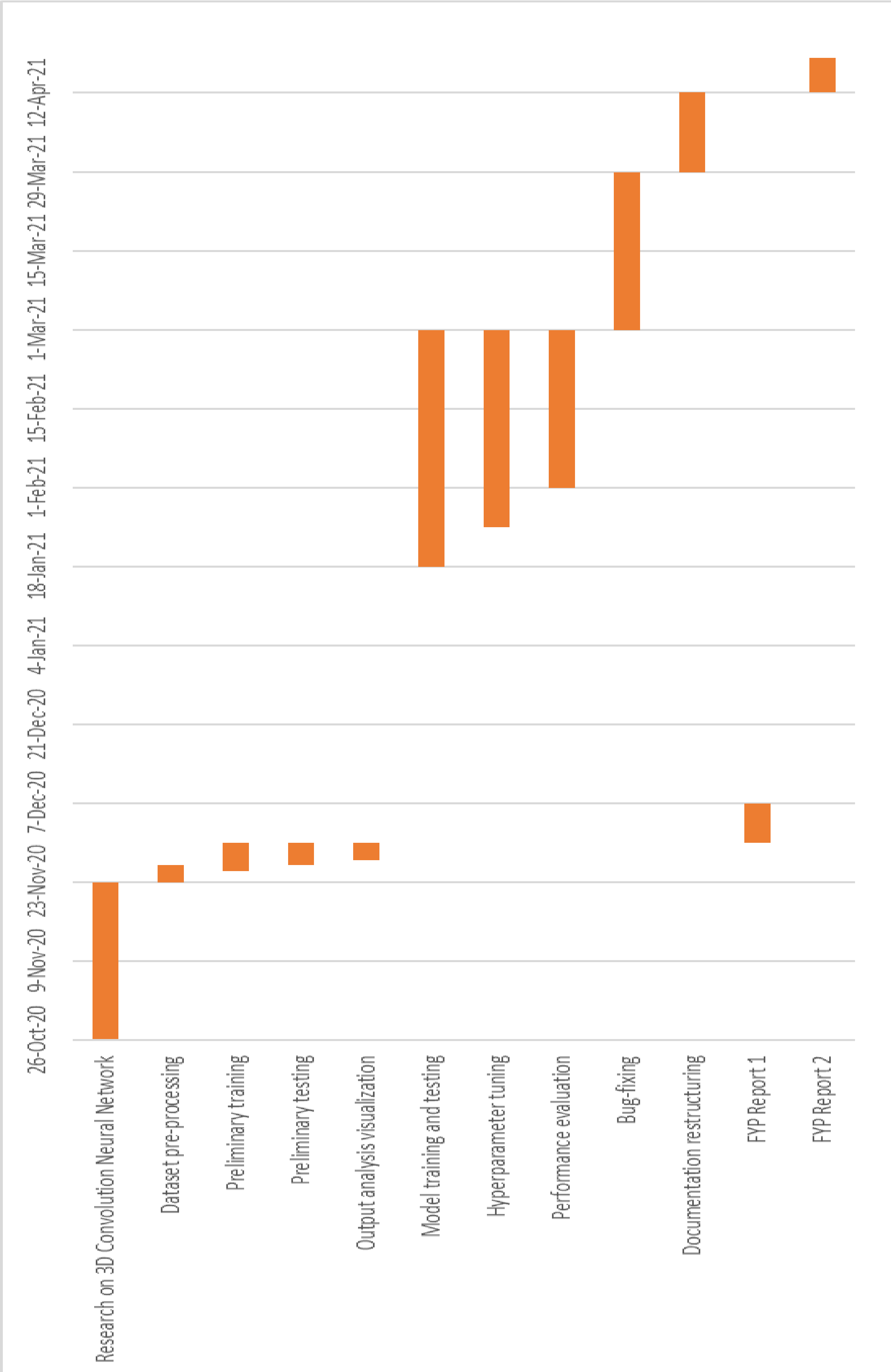


Figure 3.6.1 Gantt Chart

Chapter 4 Preliminary Work

4.1 Preliminary Work Experimental Setup

Layer Name	Output shape	Parameters
Input	[224 x 224 x 9 x 1]	-
Conv3D – 1	[224 x 224 x 9 x 32]	896
BatchNorm3D – 2	[224 x 224 x 9 x 32]	128
LeakyRelu – 3	[224 x 224 x 9 x 32]	0
MaxPool3d - 4	[56 x 56 x 9 x 32]	0
Conv3D – 5	[56 x 56 x 9 x 64]	55,360
BatchNorm3D – 6	[56 x 56 x 9 x 64]	256
LeakyRelu – 7	[56 x 56 x 9 x 64]	0
MaxPool3d – 8	[14 x 14 x 9 x 64]	0
Linear – 9	[128 x 1]	14,450,816
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0
Linear – 13	[2 x 1]	258

Table 4.1.1 3DCNN Architecture 1

Layer Name	Output shape	Parameters
Input	[224 x 224 x 6 x 1]	-
Conv3D – 1	[224 x 224 x 6 x 32]	896
BatchNorm3D – 2	[224 x 224 x 6 x 32]	128
LeakyRelu – 3	[224 x 224 x 6 x 32]	0
MaxPool3d - 4	[56 x 56 x 6 x 32]	0
Conv3D – 5	[56 x 56 x 6 x 64]	55,360
BatchNorm3D – 6	[56 x 56 x 6 x 64]	256
LeakyRelu – 7	[56 x 56 x 6 x 64]	0
MaxPool3d – 8	[14 x 14 x 6 x 64]	0
Linear – 9	[128 x 1]	9,633,920
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0

Linear – 13	[2 x 1]	258

Table 4.1.2 3DCNN Architecture 2

Layer Name	Output shape	Parameters
Input	[224 x 224 x 3 x 1]	-
Conv3D – 1	[224 x 224 x 3 x 32]	896
BatchNorm3D – 2	[224 x 224 x 3 x 32]	128
LeakyRelu – 3	[224 x 224 x 3 x 32]	0
MaxPool3d - 4	[56 x 56 x 3 x 32]	0
Conv3D – 5	[56 x 56 x 3 x 64]	55,360
BatchNorm3D – 6	[56 x 56 x 3 x 64]	256
LeakyRelu – 7	[56 x 56 x 3 x 64]	0
MaxPool3d – 8	[14 x 14 x 3 x 64]	0
Linear – 9	[128 x 1]	4,817,024
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0
Linear – 13	[2 x 1]	258

Table 4.1.3 3DCNN Architecture 3

Layer Name	Output shape	Parameters
Input	[224 x 224 x 9 x 1]	-
Conv3D – 1	[224 x 224 x 9 x 32]	896
BatchNorm3D – 2	[224 x 224 x 9 x 32]	128
LeakyRelu – 3	[224 x 224 x 9 x 32]	0
MaxPool3d - 4	[112 x 112 x 4 x 32]	0
Conv3D – 5	[112 x 112 x 4 x 64]	55,360
BatchNorm3D – 6	[112 x 112 x 4 x 64]	256
LeakyRelu – 7	[112 x 112 x 4 x 64]	0
MaxPool3d – 8	[56 x 56 x 2 x 64]	0
Linear – 9	[128 x 1]	51,380,352
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0

Linear – 13	[2 x 1]	258

Table 4.1.4 3DCNN Architecture 4

Layer Name	Output shape	Parameters
Input	[224 x 224 x 6 x 1]	-
Conv3D – 1	[224 x 224 x 6 x 32]	896
BatchNorm3D – 2	[224 x 224 x 6 x 32]	128
LeakyRelu – 3	[224 x 224 x 6 x 32]	0
MaxPool3d - 4	[112 x 112 x 3 x 32]	0
Conv3D – 5	[112 x 112 x 3 x 64]	55,360
BatchNorm3D – 6	[112 x 112 x 3 x 64]	256
LeakyRelu – 7	[112 x 112 x 3 x 64]	0
MaxPool3d – 8	[56 x 56 x 1 x 64]	0
Linear – 9	[128 x 1]	25,690,240
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0
Linear – 13	[2 x 1]	258

Table 4.1.5 3DCNN Architecture 5

Layer Name	Output shape	Parameters
Input	[224 x 224 x 3 x 1]	-
Conv3D – 1	[224 x 224 x 3 x 32]	896
BatchNorm3D – 2	[224 x 224 x 3 x 32]	128
LeakyRelu – 3	[224 x 224 x 3 x 32]	0
MaxPool3d – 4	[112 x 112 x 3 x 32]	0
Conv3D – 5	[112 x 112 x 3 x 64]	55,360
BatchNorm3D – 6	[112 x 112 x 3 x 64]	256
LeakyRelu – 7	[112 x 112 x 3 x 64]	0
MaxPool3d – 8	[56 x 56 x 3 x 64]	0
Linear – 9	[128 x 1]	77,070,464
LeakyRelu – 10	[128 x 1]	0
BatchNorm1D – 11	[128 x 1]	512
Dropout – 12	[128 x 1]	0

Linear – 13	[2 x 1]	258

Table 4.1.6 3DCNN Architecture 6

There is a total of 6 3DCNN model architectures that are designed and tested during the preliminary phase. Table 3.3.1 to Table 3.3.3 are having a 1 x 4 x 4 kernel size in the 3D Max Pooling layers while the architectures shown in Table 3.3.4 to Table 3.3.6 are having a 4 x 4 x 4 kernel size in the 3D Max Pooling layers. The filter size in the 3D Max Pooling layers control the depth of the image, with the first three architectures the depth of the images are not reduced before it reaches the Fully Connected (FC) layers while the depth of the images in the architecture 4 and 5 will be reduced. The input 3DCNN architecture 6 will have constant depth before reaching the FC layers, however the width and the height of the input will be increased as the size of the kernel has decreased. In other words, the input of the first FC layer for the model architecture 4, 5 and 6 will be bigger than the remaining 3 architectures.

The hyperparameter used for the models:

i. Number of epochs: 100
ii. Batch size: 16
iii. Optimizer: Adam / SGD, weight_decay: 0.0005
iv. Learning rate: 0.01
v. Scheduler: ReduceLROnPlateau, factor: 0.1, patience: 5, eps: $1e^{-06}$
vi. Loss Function: CrossEntropyLoss

Data Augmentation on training data:

i. RandomHorizontalFlip: 0.5
ii. RandomVerticalFlip: 0.5
iii. RandomRotation: 0.5

4.2 Training, Validation and Testing dataset

	Training data (B)	**Validation data (B)**	**Testing data (F)**
Good Joints	382	66	185
Defect Joints	247	72	159

Table 4.2.1 Datasets for preliminary training and testing

The train_test_split from sklearn.model_selection library is used to split the training and validation data. The training and validation data were split from the dataset B with the ratio of 80:20. Dataset F was chosen to be the testing data for this phase as the number of the data in dataset F is the highest among the remaining 9 slices dataset. Dataset B and F are the best preliminary training and testing candidates as the slicing sequence and the location of the BGA is different for both datasets.

4.3 Performance evaluation

Model	Slices	Sequence	Optimizer	Training Time	Testing Time	Accuracy	Confusion Matrix	
A	9	PMPS1S6	Adam	2075.9	6.3	50.29%	1	158
							13	172
B	9	PMPS1S6	SGD	1960.7	5.4	60.47%	131	28
							108	77
C	9	Ori	Adam	1978.5	5.7	62.50%	130	29
							100	85
D	9	Ori	SGD	2013.9	6.2	62.79%	111	48
							80	105
E	3	Ori	Adam	722.1	2	45.93%	7	152
							34	151
F	3	Ori	SGD	698.2	2.2	46.22%	2	157
							28	157
G	3	PadMP	Adam	704	2.2	50.87%	158	1
							168	17
H	3	PadMP	SGD	647.4	2	53.78%	158	1
							158	27
I	3	PackageMP	Adam	650.5	1.9	49.42%	159	0
							174	11
J	3	PackageMP	SGD	689.3	2.1	57.27%	149	10
							137	48
K	6	S1S6	Adam	1372.3	4.5	47.38%	55	104
							77	108
L	6	S1S6	SGD	1342	4.4	60.76%	125	34
							101	84
M	6	Ori	Adam	1357.5	4.4	66.57%	121	38
							77	108
N	6	Ori	SGD	1368	5	60.76%	141	18
							117	68
O	9	Ori	Adam	2363.8	6	55.52%	85	74
							79	106
P	9	Ori	SGD	2208.6	10.1	59.01%	96	63
							78	107
Q	6	Ori	Adam	1642.7	5	68.60%	125	34
							74	111
R	6	Ori	SGD	1514.2	3.7	63.66%	127	32
							93	92
S	3	Ori	Adam	1100.5	2.1	50.29%	2	157
							14	171
T	3	Ori	SGD	1023	7.6	47.97%	40	119
							60	125

Table 4.3.1 Model Performance

Model Architecture	Model
1	A, B, C, D
2	K, L, M, N
3	E, F, G, H, I, J
4	O, P
5	Q, R
6	S, T

Table 4.3.2 Model Architecture to Model

Based on Table 4.3.1, the top 3 highest accuracy is 68.6% by model Q, 66.57 by model M and 63.66% by model R. The statistic also shows that the model trained with original stacking sequence could produce a better accuracy compared to those with random sequence. Moreover, the statistic also shows that the SGD optimizer is dealing better with BGA stacks with random sequence, but it is outperformed by the Adam optimizer when dealing with original BGA stacking sequence. Despite having a better training time, models with SGD optimizer required more time for making predictions. Besides that, model trained with only 3 slices often misclassified the defect joints when compared to model trained with 6 or 9 slices. Based on this statistic, it is strongly believed that the model proposed in this project should be trained with the original stacking sequence with Adam optimizer.

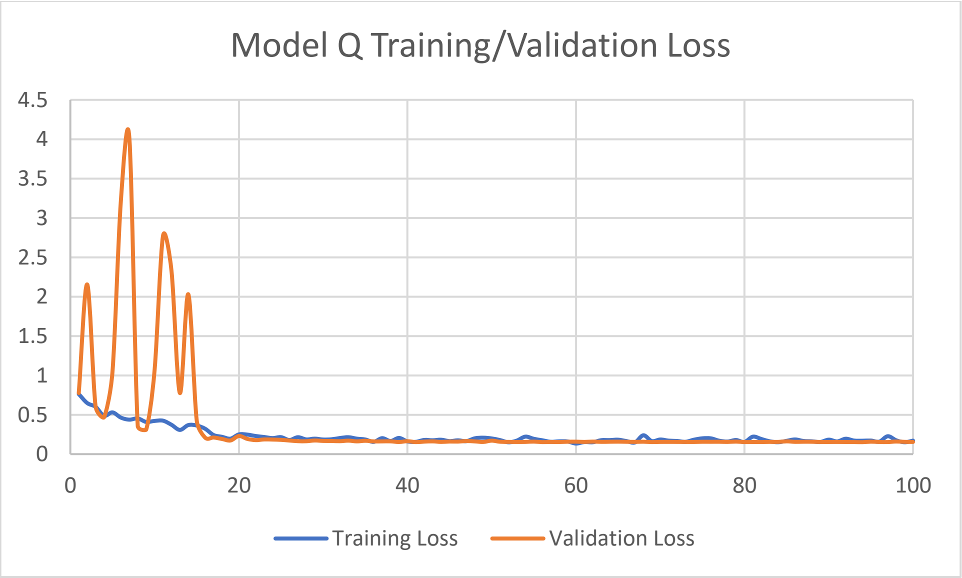


Figure 4.3.1 Model Q Training/Validation Loss

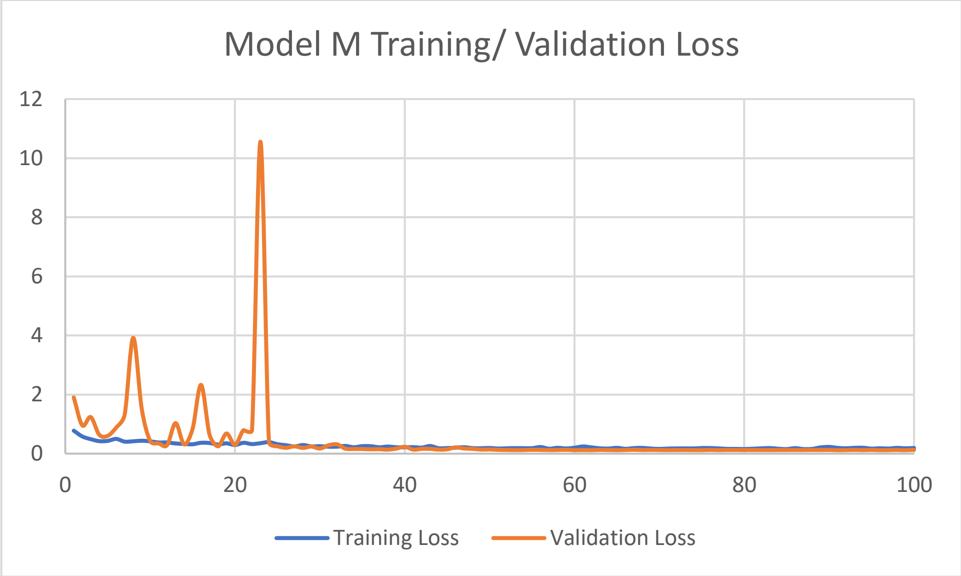


Figure 4.3.2 Model M Training/Validation Loss

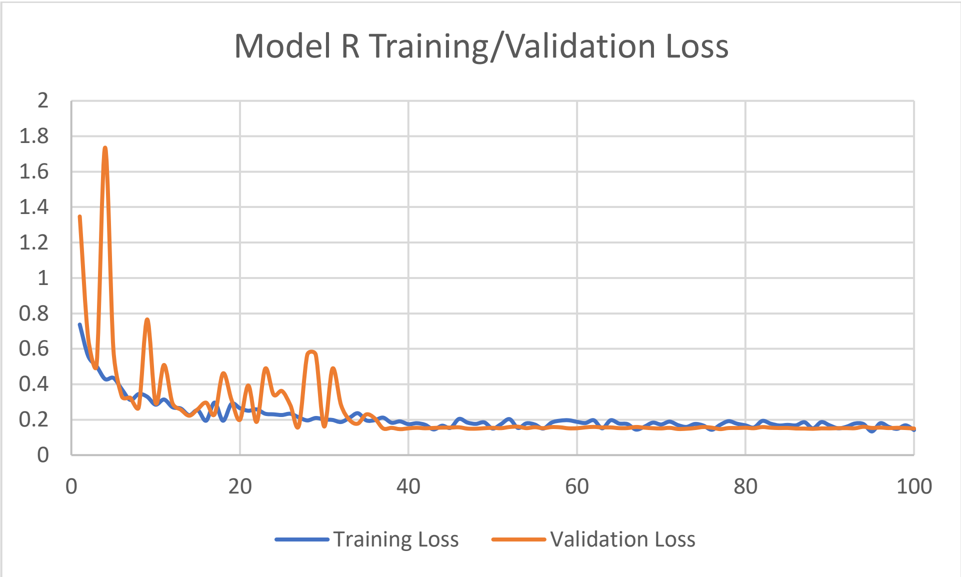


Figure 4.3.3 Model R Training/Validation Loss

Despite having a higher loss in the first 20 to 40 epochs, model Q and M still outperformed model R in the long run. Based on the three figures above, none of the models can achieve 0 loss in the training and validation process. However, the lowest loss value for model R is still significantly higher when compared to model Q and M. This results in a poorer performance when model R is used to predict the testing data. Since none of the models can achieve 100% accuracy in predicting validation data, it is strongly believed that all of them are still slightly underfitting the training data. The result is expected to be better when a deeper model or model with a different hyperparameter or different number of neurons in the FC layers is used

Chapter 5 Experiments and Evaluation.

5.1 Experimental setup

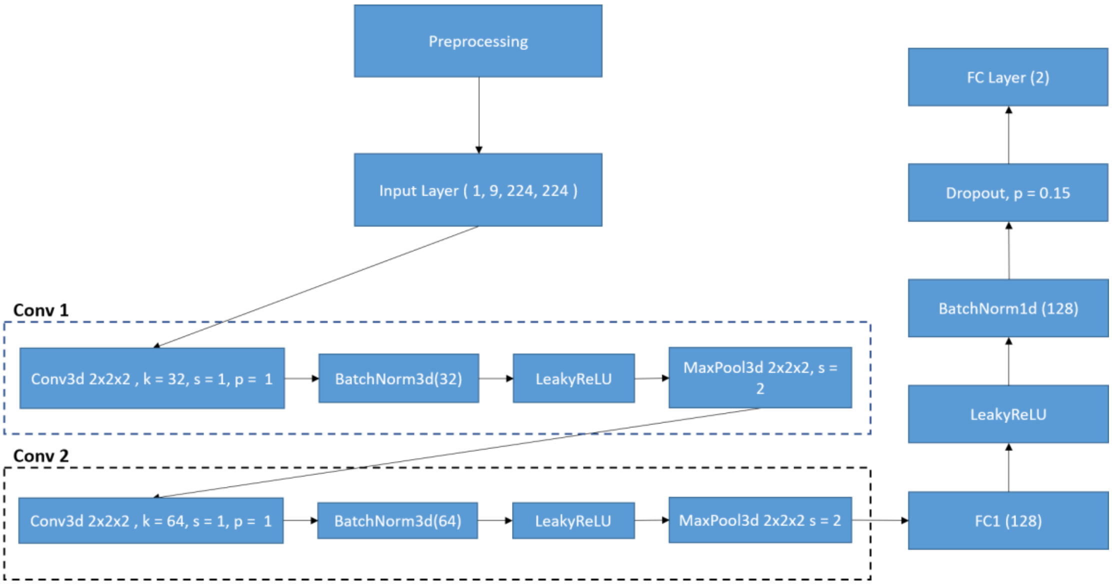


Figure 5.1.1 Model Architecture

Figure 5.1.1 shows the model architecture that is used to build the models that obtained the best result in 3 different combinations of datasets in this project. Layers, filter size, number of neurons and activation function used are specified in the figure to help others to replicate the model. The input volume will have a shape of (1, 9, 224, 224) where 1 represents the channel of the image, 9 represents the number of slices after padding, followed by the resolution of the image. The models are trained with different hyperparameters to suit different combinations of datasets as shown below.

Hyperparameters for best model in combination A:

i. Number of epochs: 100
ii. Batch size: 16
iii. Optimizer: Adam, weight_decay: 0.0005
iv. Learning rate: 0.01
v. Scheduler: ReduceLROnPlateau, factor: 0.1, patience: 5, eps: $1e^{-06}$
vi. Loss Function: CrossEntropyLoss
vii. Initialization on fully connected layer: Xavier Normal initialization for weight, zero for bias.
viii. Image stacking method used: Dummy images first followed with BGA images stacked with original sequence.

There are two models that could achieve the same accuracy in classifying BGA images in combination B, however the model with the least number of false positives is selected here.

Hyperparameters for best model in combination B:

i. Number of epochs: 100
ii. Batch size: 16
iii. Optimizer: Adam, weight_decay: $1e^{-06}$
iv. Learning rate: 0.01
v. Scheduler: ReduceLROnPlateau, factor: 0.1, patience: 5, eps: $1e^{-06}$
vi. Loss Function: CrossEntropyLoss
vii. Image stacking method used: Dummy images first followed with BGA images stacked with original sequence.

There are multiple models that could achieve the same accuracy in classifying BGA images in combination D, however the model with the least number of false positives is selected here.

Hyperparameters for best model in combination D:

i. Number of epochs: 100
ii. Batch size: 16
iii. Optimizer: Adam, weight_decay: 0.0005
iv. Learning rate: 0.01
v. Scheduler: ReduceLROnPlateau, factor: 0.1, patience: 5, eps: $1e^{-06}$
vi. Loss Function: CrossEntropyLoss
vii. Initialization on fully connected layer: Xavier Uniform initialization for weight, zero for bias.
viii. Image stacking method used: Dummy images first followed with BGA images stacked with original sequence. (Normalized version)

The data augmentation used in this phase remains unchanged as in the preliminary stage. Only training data will be augmented while testing and validating data will not.

5.2 Result evaluation

<table>
<tr><th>Combination</th><th colspan="2">Validation Confusion Matrix</th><th>Validation Accuracy</th><th colspan="2">Testing Confusion Matrix</th><th>Testing Accuracy</th><th>Inference Speed per Joint (ms)</th></tr>
<tr><td rowspan="2">A</td><td>129</td><td>15</td><td rowspan="2">89.69</td><td>214</td><td>45</td><td rowspan="2">86.90</td><td rowspan="2">16.7</td></tr>
<tr><td>18</td><td>158</td><td>26</td><td>257</td></tr>
<tr><td rowspan="2">B</td><td>94</td><td>8</td><td rowspan="2">95.13</td><td>137</td><td>5</td><td rowspan="2">95.41</td><td rowspan="2">19.5</td></tr>
<tr><td>3</td><td>121</td><td>8</td><td>133</td></tr>
<tr><td rowspan="2">D</td><td>104</td><td>7</td><td rowspan="2">96.69</td><td>143</td><td>9</td><td rowspan="2">93.41</td><td rowspan="2">21.3</td></tr>
<tr><td>2</td><td>159</td><td>14</td><td>183</td></tr>
</table>

Table 5.2.1 Result obtained with 3D CNN models

<table>
<tr><th>Combination</th><th colspan="2">Testing Confusion Matrix</th><th>Testing Accuracy</th></tr>
<tr><td rowspan="2">A</td><td>191</td><td>68</td><td rowspan="2">79.15</td></tr>
<tr><td>45</td><td>238</td></tr>
<tr><td rowspan="2">B</td><td>125</td><td>17</td><td rowspan="2">92.23</td></tr>
<tr><td>5</td><td>136</td></tr>
<tr><td rowspan="2">D</td><td>0</td><td>152</td><td rowspan="2">56.45</td></tr>
<tr><td>0</td><td>197</td></tr>
</table>

Table 5.2.2 Result obtained with Manufacturer 2D CNN models

Table 5.2.1 shows the result obtained by the 3D CNN models created in this project while Table 5.2.2 shows the performance of the manufacturer 2D CNN models on 3 of the datasets. The tables show that the performance of the 3D CNN models is must better when compared to the 2D CNN models. Both of the 2D CNN models and 3D CNN models are trained, validated and tested with the similar training, validating and testing data. However, since the hyperparameters in the 2D CNN are not tuned to fit each of the dataset while in the 3DCNN each models' hyperparameters are tuned to fit the datasets. Therefore, to make it fair, the average accuracy score is used compared with the 2D CNN models instead of the best results obtained by the 3D CNN models in each dataset.

Combination	Average 3DCNN Models Accuracy	Manufacturer’s 2D CNN models Accuracy
A	82.63	79.15
B	82.11	92.23
D	91.43	56.45

Table 5.2.3 Performance comparison

As can be seen in Table 5.2.3, the 3D CNN models are outperforming the 2DCNN models in combination A and combination D. The average accuracy for 3D CNN models in combination B doesn’t exceed the accuracy obtained with the 2D CNN as there are a few models that are very underfitting the datasets which causes the average to be low. When those outliers are excluded, an average accuracy score of 92.58% could be obtained by the 3D CNN models in combination B. Besides that, the models are able to classify a BGA joint within 200ms as shown in Table 5.2.1 More details regarding the performance, the setting used and the inference speed for the models could be seen in the appendix.

Chapter 6 Conclusion

6.1 Project Review

HIP defect is always a pain for the inspector when inspecting the PCB as HIP defects are sometimes misclassified by the AXI machine. This led to the need of manual inspection with the help of a real-time X-ray machine which had a lower efficiency rate when compared to the AXI machine. Despite having a longer processing time, manual inspection is having a higher accuracy rate when compared to machine inspection. Hence, a more efficient way to classify the HIP defects is required. With the help of the 3D CNN model that achieved all the objectives in this project, it is strongly believed that the model would cause a huge impact in the manufacturing sector. The model is able to classify one BGA joint in less than 200ms with a better performance when compared to the 2D CNN.

6.2 Future work

More experiments are expected to be done with different BGA slices images as the current models could only accept BGA joints with slices less than or equal to 9. A specific stacking technique together with the solution for different input size are expected to be proposed in the coming future. In addition, the size of the best models at the moment is approximately 210 MB which is considered quite huge when compared to models trained with 2D-squeezenet or 2D-mobilenet which is only about 20 – 40 MB. Therefore, a solution for reducing the model size is required as deploying a huge model would require a better and powerful machine which would increase the cost for the manufacturing organizations who wish to have this model. Moreover, there is still room for improvement as the models are not able to achieve 99% in terms of accuracy. This is crucial as the HIP defects would cause a huge damage to humans and the PCB manufacturer when the defects are not being detected.

BIBLIOGRAPHY

Adoriasoft (2017). *Image Recognition and Image Processing Techniques.* [Online] Available from: https://medium.com/@Adoriasoft/image-recognition-and-image-processing-techniques-fe3d35d58919 (Accessed 1 December 2020).

Alpha Assembly Solutions (n.d., pp. 1-2). *How to Avoid BGA Head-in-Pillow Defects.* [Online] Available from: https://www.alphaassembly.com/-/media/Files/CooksonElectronics/Tech-Tips/Solder-Paste/TechTip---How-to-Avoid-BGA-Head-in-Pillow-Defects.pdf (Accessed 10 September 2020).

Bansal, S. (2018). *3D Convolutions: Understanding + Use Case.* [Online] Available from: https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case (Accessed 1 December 2020).

Castellanos, A., Gutierrez, A., Martin, G., Vandiver, M., Dematampitiya, R., Le, H., Le, E., Chau, P., Cui, H., Zhao, A.Q. and Qian, W.B., 2019. Head in Pillow X-ray Inspection at Flextronics.

Chen, C., Cai, J., Wang, Q., Wang, Y., Zou, G., Zhao, Z. and Park, C.Y. (2014) Characterization of after-reflow misalignment on Head-in-Pillow defect in BGA assembly. In *2014 15th international conference on electronic packaging technology* (pp. 1177-1180). IEEE.

Farooq, U. (2018). *Model summary in PyTorch.* [Online] Available from: https://medium.com/@umerfarooq_26378/model-summary-in-pytorch-b5a1e4b64d25 (Accessed 1 December 2020).

Fung, V. (2017). *An Overview of ResNet and its Variants.* [Online] Available from: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035 (Accessed 10 September 2020).

Hargrave, M. (2019). *Deep Learning.* [Online] Available from: https://www.investopedia.com/terms/d/deep-learning.asp#:~:text=Deep%20learning%20is%20an%20AI,is%20both%20unstructured%20and%20unlabeled (Accessed 1 December 2020).

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

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Ji, S., Xu, W., Yang, M. and Yu, K., 2012. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, *35*(1), pp.221-231.

Koczera, B. and Zhao, A. (2012). *Using Automated 3D X-Ray Inspection to Detect BTC Defects.* [Online] Available from: https://smtnet.com/library/files/upload/Detect-BTC-Defects.pdf (Accessed 10 September 2020).

Mallick, S. and Nayak, S. (2018). *Number of Parameters and Tensor Sizes in a Convolutional Neural Network (CNN)* [Online] Available from: https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/ (Accessed 1 December 2020).

Olah, C.,2015. *Understanding LSTMs.* [Online] Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accessed 10 September 2020).

Prabhu (2018). *Understanding of Convolutional Neural Network (CNN) – Deep Learning.* [Online] Available from: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (Accessed 10 September 2020).

Prove, P. (2017). *An Introduction to different Types of Convolutions in Deep Learning.* [Online] Available from: https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d (Accessed 1 December 2020).

Seelig, K., 2008. Head-in-Pillow BGA Defects. *AIM Solder, Montreal, Canada, http://www. aimsolder. com/sites/default/files/head-in-pillow_bga_defects. pdf.*

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

Tsang, S. (2018). *Review: VGGNet – 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014.* [Online] Available from: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11 (Accessed 10 September 2020).

Zhang, Q., Zhang, M., Gamanayake, C., Yuen, C., Geng, Z., Jayasekaraand, H., Zhang, X., Woo, C.W., Low, J. and Liu, X., 2020. Deep Learning Based Defect Detection for Solder Joints on Industrial X-Ray Circuit Board Images. *arXiv preprint arXiv:2008.02604*.

APPENDIX A: Poster

UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology

Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique

PROFILE

Project Developer
Tan Wei Jin

Programme
Bachelor Of Computer Science

Project Supervisor
Dr. Aun Yichiet

MODEL HIGHLIGHTS

ROBUST ●●●○○

ACCURACY ●●●●○

INFERENCE SPEED ●●●●●

MODEL SIZE ●●●○○

INTRODUCTION

The Head-in-Pillow (HIP) defects had always been an issue during the Ball Grid Array (BGA) assembly. This conceivably expensive imperfection isn't generally distinguished in practical testing, and would just appears as a disappointment in the field after the components are exposed to some physical or thermal stress. Hence, a 3-Dimensional Convolutional Neural Network model is proposed to ease the HIP defects inspection process.

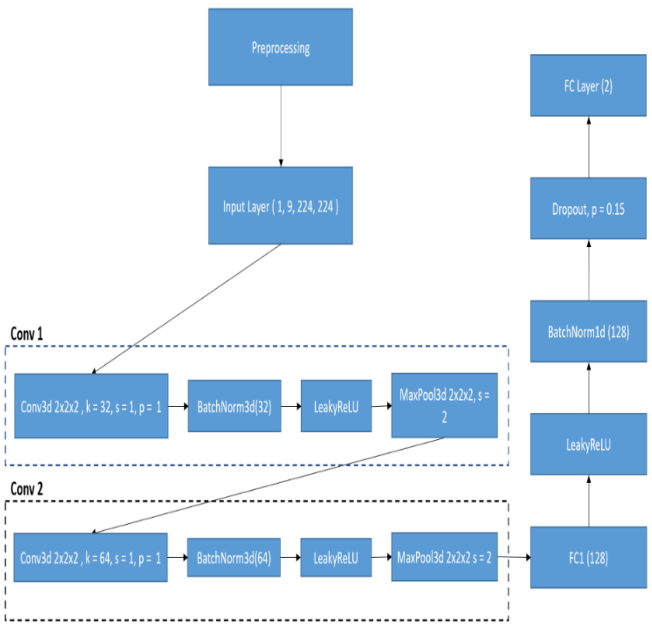
PERFORMANCE COMPARISION

Combination	Average 3DCNN Models Accuracy	Vitrox 2D CNN models Accuracy
A	82.63	79.15
B	82.11	92.23
D	91.43	56.45

PROJECT OBJECTIVES

- To design a 3D CNN for BGA HIP defects detection
- To design an image stacking technique using salient layers based on domain heuristics
- To retrain BGA dataset on the proposed 3D CNN model to improve upon existing Vitrox 2D CNN architecture
- To optimize a 3D CNN model that could inference the BGA HIP defects in less than 200ms per inference.

MODEL ARCHITECTURE



APPENDIX B: Final Year Project Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 2**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE [Please write the details of the work done in the last fortnight.] i. Reversing the image stacking sequence ii. Retrain the model with the stacking method
2. WORK TO BE DONE i. Hyperparameter tuning for the models ii. Try to test the model with a different dataset.
3. PROBLEMS ENCOUNTERED i. Accuracy is still fairly low for the current models
4. SELF EVALUATION OF THE PROGRESS Self-assigned tasks are completed within expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. Hyperparameter tuning and new stacking method did improve the accuracy of the model.

ii. The model is tested with new datasets from the similar PCB board. 90% accuracy in classifying is achieved for the first time.

2. WORK TO BE DONE

i. Try to add more neurons in the fully connected layers

ii. Try to add a fully connected layers to increase the complexity of the model

3. PROBLEMS ENCOUNTERED

i. Models couldn't classify the defects correctly even if its from the same PCB board.

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE
[Please write the details of the work done in the last fortnight.] i. More neurons are added to the fully connected layer, not much improvement in term of accuracy. ii. An additional fully connected layer is added to the model, not much improvement in term of accuracy.
2. WORK TO BE DONE
i. Hyperparameter tuning on model with new architecture.
3. PROBLEMS ENCOUNTERED
i. Models could only classify BGA images with 9 slices.
4. SELF EVALUATION OF THE PROGRESS
Self-assigned tasks are completed within expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 8**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. Hyperparameter tuning on new model architecture didn't have much improvement in terms of accuracy.
ii. Model is able to classify BGA defects with 5,6 and 9 slices through dummy image padding method.
iii. Model's accuracy is stable at 80%.

2. WORK TO BE DONE

i. Perform hyperparameter tuning on the models.
ii. Train the model by excluding some of the datasets that might cause the accuracy to be low.
iii. Train the model on new datasets received from Vitrox

3. PROBLEMS ENCOUNTERED

i. Have to allocate some time on other assignments,

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 10**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. Models' accuracy improved slight with hyperparameter tuning.
ii. Model train without certain datasets could achieved 95% accuracy.
iii. Model train with dataset received by Vitrox could only achieved about 89% accuracy.

2. WORK TO BE DONE

i. Fully connected layer initialization.
ii. Try to pad the dummy images in front of the slice images.
iii. More hyperparameter tuning.
iv. Train model with newly collected data by Vitrox.

3. PROBLEMS ENCOUNTERED

i. Model accuracy is currently cap at 95% and few of the dataset for a particular PCB is excluded.
ii. Model accuracy is cap at 85% if all data from 4 different PCB boards are used.

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

Project II

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Tan Wei Jin 17ACB02302	
Supervisor: Dr. Aun Yichiet	
Project Title: Detecting Head-In-Pillow Defect (HIP) By Using Deep Learning and Image Processing Technique	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. Padding dummy images in front of slice image could improve the accuracy of the model.

ii. Normalization and initialization in fully connected layer are able to help the model trained with new data collected by Vitrox to achieved 93% accuracy.

iii. Finalization meeting to freeze the project with Vitrox.

2. WORK TO BE DONE

i. Start to prepare FYP report 2.

ii. Finalizing all the results obtained through out the process.

3. PROBLEMS ENCOUNTERED

i. Had to spend more time in assignments that due by this week.

ii. Accuracy of model is not as high as expectation.

4. SELF EVALUATION OF THE PROGRESS

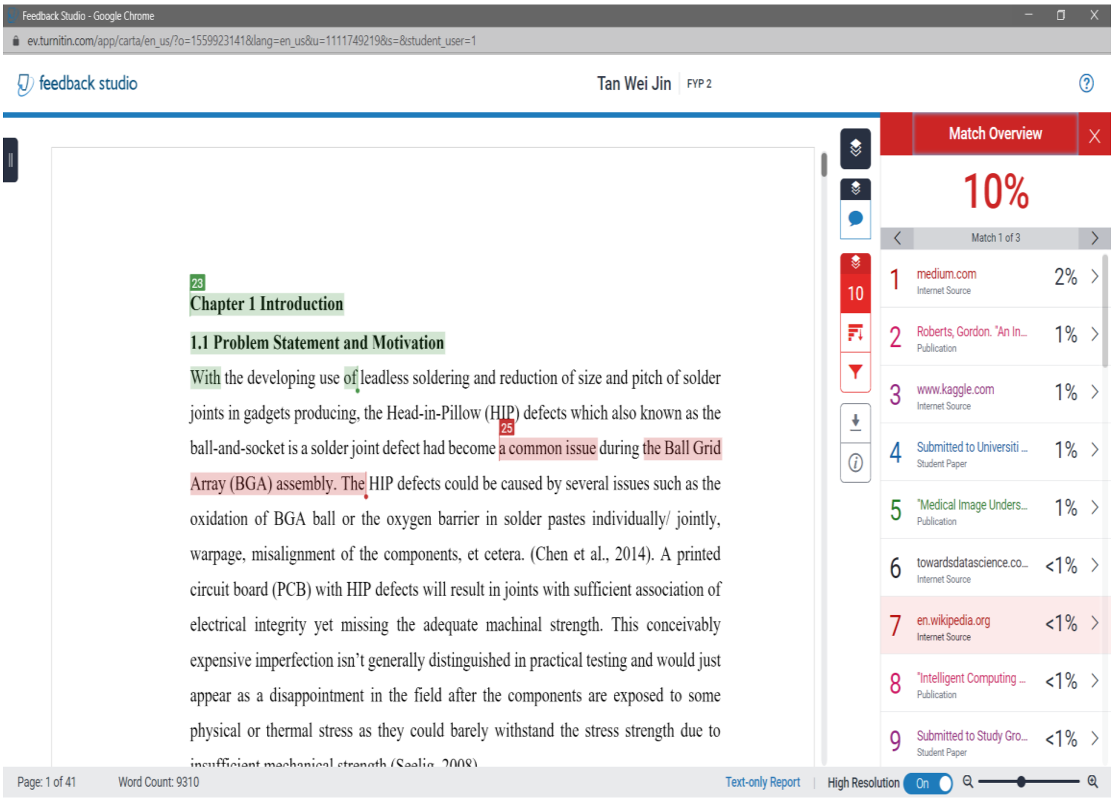
Self-assigned tasks are completed within expected timeframe.

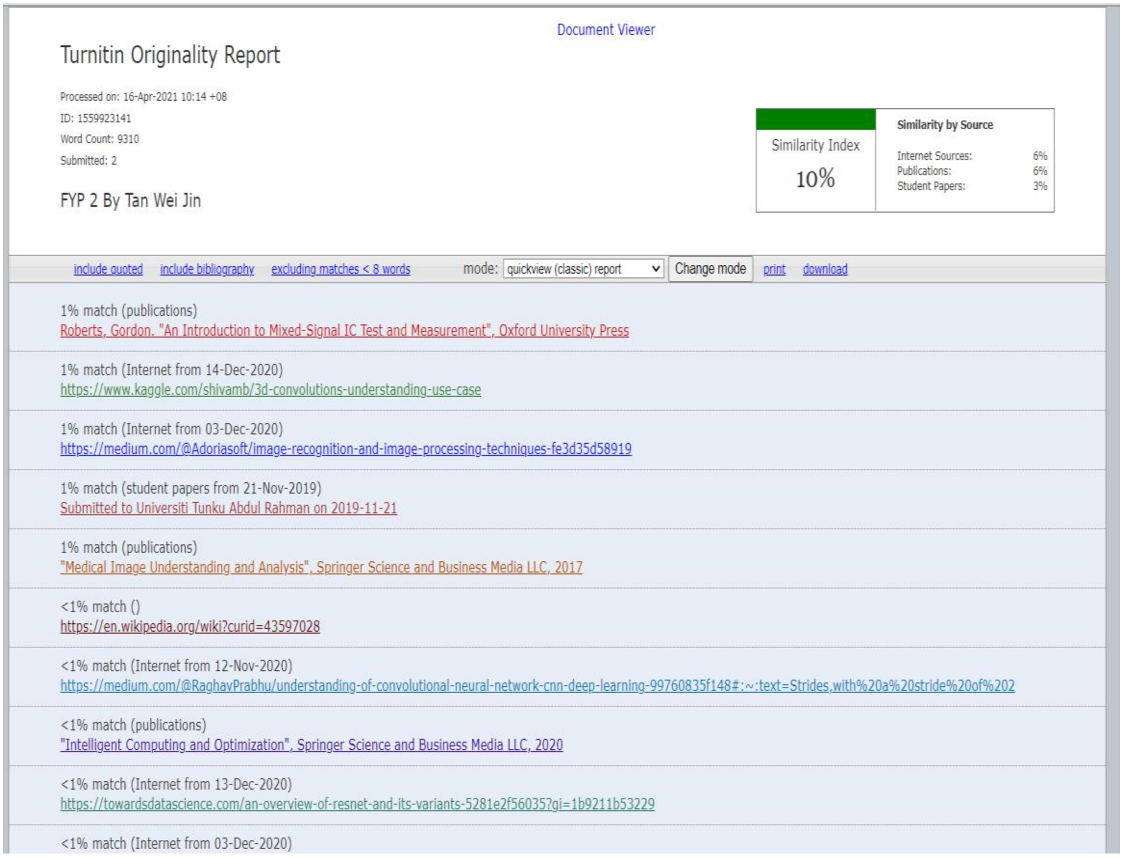
Supervisor's signature

尉進 陳

Student's signature

APPENDIX C: Plagiarism Check Result




Document Viewer

Turnitin Originality Report

Processed on: 16-Apr-2021 10:14 +08
ID: 1559923141
Word Count: 9310
Submitted: 2

FYP 2 By Tan Wei Jin

Similarity Index	Similarity by Source	
10%	Internet Sources:	6%
	Publications:	6%
	Student Papers:	3%

include quoted include bibliography excluding matches < 8 words mode: quickview (classic) report Change mode print download

1% match (publications)
Roberts, Gordon. "An Introduction to Mixed-Signal IC Test and Measurement", Oxford University Press

1% match (Internet from 14-Dec-2020)
https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case

1% match (Internet from 03-Dec-2020)
https://medium.com/@Adoriasoft/image-recognition-and-image-processing-techniques-fe3d35d58919

1% match (student papers from 21-Nov-2019)
Submitted to Universiti Tunku Abdul Rahman on 2019-11-21

1% match (publications)
"Medical Image Understanding and Analysis", Springer Science and Business Media LLC, 2017

<1% match ()
https://en.wikipedia.org/wiki?curid=43597028

<1% match (Internet from 12-Nov-2020)
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148#:~:text=Strides,with%20a%20stride%20of%202

<1% match (publications)
"Intelligent Computing and Optimization", Springer Science and Business Media LLC, 2020

<1% match (Internet from 13-Dec-2020)
https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035?gi=1b9211b53229

<1% match (Internet from 03-Dec-2020)

APPENDIX D: Models' Performance

Model	Recipe	Slices	Sequence	Normalization	MaxPool Shape	FC Layers / Global Pooling	Initialization	Dropout	Activation Function	Optimizer	Weight_Decay	Patience	Validation CM		Validation Accuracy	Testing CM		Testing Accuracy	Testing Time	Testing Time per Joint
1	Training: B, Validating: B, Testing: F	9	PMCS1S6	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	67	5	94.20289855	1	158	50.29069767	6.369995594	0.018517429
													3	63		13	172			
2	Training: B, Validating: B, Testing: F	9	PMCS1S6	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	71	1	97.10144928	131	28	60.46511628	5.418999434	0.015752905
													3	63		108	77			
3	Training: B, Validating: B, Testing: F	9	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	72	0	96.37681159	130	29	62.5	5.711996078	0.01660464
													5	61		100	85			
4	Training: B, Validating: B, Testing: F	9	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	96.37681159	111	48	62.79069767	6.235998869	0.018127904
													2	64		80	105			
5	Training: B, Validating: B, Testing: F	3	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	70	2	94.92753623	7	152	45.93023256	2.002996445	0.005822664
													5	61		34	151			
6	Training: B, Validating: B, Testing: F	3	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.20289855	2	157	46.22093023	2.229994774	0.006482543
													4	62		28	157			
7	Training: B, Validating: B, Testing: F	3	PMC	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	70	2	95.65217391	158	1	50.87209302	2.237996578	0.006505804
													4	62		168	17			
8	Training: B, Validating: B, Testing: F	3	PMC	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	94.92753623	158	1	53.77906977	1.986996412	0.005776152
													4	62		158	27			
9	Training: B, Validating: B, Testing: F	3	CMP	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	159	0	49.41860465	1.896998405	0.00551453
													4	62		174	11			
10	Training: B, Validating: B, Testing: F	3	CMP	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.20289855	149	10	57.26744186	2.071990967	0.00602323
													4	62		137	48			
11	Training: B, Validating: B, Testing: F	6	S1S6	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	71	1	96.37681159	55	104	47.38372093	4.54199338	0.013203469
													4	62		77	108			
12	Training: B, Validating: B, Testing: F	6	S1S6	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	71	1	96.37681159	125	34	60.75581395	4.383995295	0.012744172
													4	62		101	84			
13	Training: B, Validating: B, Testing: F	6	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	71	1	96.37681159	121	38	66.56976744	4.40600419	0.012808152
													4	62		77	108			
14	Training: B, Validating: B, Testing: F	6	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	70	2	95.65217391	141	18	60.75581395	5.028992176	0.014619163
													4	62		117	68			
15	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	94.92753623	85	74	55.52325581	5.951997519	0.017302318
													3	63		79	106			
16	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	93.47826087	96	63	59.01162791	4.983996868	0.014488363
													5	61		78	107			
17	Training: B, Validating: B, Testing: F	6	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	67	5	94.20289855	125	34	68.60465116	2.110995293	0.006136614
													3	63		74	111			
18	Training: B, Validating: B, Testing: F	6	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.20289855	127	32	63.6627907	7.609423876	0.022120418
													4	62		93	92			
19	Training: B, Validating: B, Testing: F	3	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	94.92753623	2	157	50.29069767	10.06244397	0.029251291
													3	63		14	171			
20	Training: B, Validating: B, Testing: F	3	Original	-	Irregular	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	94.20289855	40	119	47.96511628	3.719548941	0.010812642
													5	61		60	125			
21	Training: B, Validating: B, Testing: F	6	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	25	134	60.1744186	8.953121424	0.026026516
													4	62		3	182			
22	Training: B, Validating: B, Testing: F	6	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.92753623	44	115	65.11627907	4.10931015	0.011945669
													3	63		5	180			
23	Training: B, Validating: B, Testing: F	6	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	94.20289855	30	129	60.46511628	4.531249762	0.013172238
													4	62		7	178			
24	Training: B, Validating: B, Testing: F	6	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	71	1	97.10144928	75	84	68.60465116	5.562501431	0.016170062
													3	63		24	161			
25	Training: B, Validating: B, Testing: F	9	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.20289855	158	1	61.3372093	12.1093235	0.035201522
													5	61		132	53			
26	Training: B, Validating: B, Testing: F	9	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	93.47826087	121	38	69.76744186	6.562499523	0.019077033
													5	61		66	119			
27	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	94.92753623	98	61	72.96511628	6.609327078	0.01921316
													3	63		32	153			
28	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	70	2	95.65217391	108	51	79.06976744	6.640563488	0.019303964
													4	62		21	164			
29	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	15	144	56.68604651	5.187435389	0.015079754
													4	62		5	180			
30	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.92753623	81	78	69.47674419	5.109373093	0.014852829
													3	63		27	158			

31	Training: B, Validating: B, Testing: F	9	Reverse Both	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	25	134	58.72093023	4.453072309	0.012944978
													4	62		8	177			
32	Training: B, Validating: B, Testing: F	9	Reverse Both	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	94.92753623	133	26	63.08139535	4.921866894	0.014307753
													4	62		101	84			
33	Training: B, Validating: B, Testing: F	9	Reverset Train	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	70	2	95.65217391	37	122	62.20930233	5.078069687	0.01476183
													4	62		8	177			
34	Training: B, Validating: B, Testing: F	9	Reverse Train	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.92753623	58	101	65.40697674	4.843690872	0.014080497
													3	63		18	167			
35	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	70	2	94.20289855	12	147	56.39534884	4.609367371	0.013399324
													6	60		3	182			
36	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	94.92753623	52	107	65.40697674	4.781251907	0.013898988
													4	62		12	173			
37	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	3	156	50.58139535	5.662103891	0.016459604
													4	62		14	171			
38	Training: B, Validating: B, Testing: F	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	68	4	94.92753623	89	70	56.97674419	6.5	0.018895349
													3	63		78	107			
39	Training: B, Validating: B, Testing: F	9	Reverse Train	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	93.47826087	159	0	59.30232558	5.484319687	0.01594279
													5	61		140	45			
40	Training: B, Validating: B, Testing: F	9	Reverse Train	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	70	2	94.20289855	135	24	70.05813953	6.453063726	0.018758906
													6	60		79	106			
41	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	68	4	94.92753623	101	58	66.27906977	5.656187296	0.016442405
													3	63		58	127			
42	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	67	5	94.92753623	114	45	75.87209302	6.249943495	0.01816844
													2	64		38	147			
43	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	24	9	76.8115942	138	21	75.38461538	5.234585285	0.016106416
													7	29		59	107			
44	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	30	3	86.95652174	148	11	90.15384615	5.421914577	0.016682814
													6	30		21	145			
45	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	24	9	73.91304348	124	35	79.38461538	8.40400362	0.025858473
													9	27		32	134			
46	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	26	7	78.26086957	133	26	82.15384615	8.208001852	0.02525539
													8	28		32	134			
47	Training: F, Validating: F, Testing: G	9	Reverse Both	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	24	9	79.71014493	113	46	83.07692308	8.712999821	0.02680923
													5	31		9	157			
48	Training: F, Validating: F, Testing: G	9	Reverse Both	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	30	3	91.30434783	131	28	89.84615385	8.29500103	0.02552308
													3	33		5	161			
49	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	21	12	73.91304348	126	33	68	8.711999655	0.026806153
													6	30		71	95			
50	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	22	11	76.8115942	127	32	82.46153846	8.472999573	0.026070768
													5	31		25	141			
51	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	69	3	94.92753623	131	28	63.37209302	15.52599788	0.045133715
													4	62		98	87			
52	Training: B, Validating: B, Testing: F	9	Reverse Test	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	69	3	92.02898551	143	16	77.61627907	8.858999968	0.025752907
													8	58		61	124			
53	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	10	29	4	88.4057971	139	20	90.15384615	15.74899793	0.048458455
													4	32		12	154			
54	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[256, 128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	10	29	4	89.85507246	129	30	89.84615385	8.294002295	0.025520007
													3	33		3	163			
55	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	Adam	5.00E-04	10	30	3	91.30434783	134	25	91.07692308	8.238997221	0.025350761
													3	33		4	162			
56	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	SGD	5.00E-04	10	24	9	84.05797101	111	48	77.23076923	7.084997416	0.021799992
													2	34		26	140			
57	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	Adam	1.00E-04	5	29	4	89.85507246	120	39	86.76923077	12.57199264	0.038683054
													3	33		4	162			
58	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	SGD	1.00E-04	5	22	11	78.26086957	120	39	83.38461538	5.991999149	0.01843692
													4	32		15	151			
59	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	Adam	1.00E-05	5	27	6	85.50724638	133	26	88	202.6725807	0.623607941
													4	32		13	153			
60	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	SGD	1.00E-05	5	24	9	78.26086957	126	33	81.53846154	323.6339989	0.99579692
													6	30		27	139			

61	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	Adam	1.00E-06	5	26	7	86.95652174	120	39	86.76923077	162.6448882	0.50044581
													2	34		4	162			
62	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	-	LeackyReLU	SGD	1.00E-06	5	27	6	85.50724638	132	27	84.61538462	5.650994778	0.017387676
													4	32		23	143			
63	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	28	5	86.95652174	127	32	89.23076923	7.366995096	0.022667677
													4	32		3	163			
64	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	26	7	69.56521739	119	40	69.84615385	6.213994265	0.019119982
													14	22		58	108			
65	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	29	4	91.30434783	133	26	84	6.266002178	0.019280007
													2	34		26	140			
66	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	24	9	75.36231884	135	24	85.53846154	6.403991699	0.01970459
													8	28		23	143			
67	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	27	6	86.95652174	120	39	87.07692308	6.425001383	0.019769235
													3	33		3	163			
68	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	29	4	86.95652174	145	14	90.46153846	6.344992399	0.019523054
													5	31		17	149			
69	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.1	LeackyReLU	Adam	5.00E-04	5	28	5	88.4057971	119	40	86.46153846	6.136000633	0.018880002
													3	33		4	162			
70	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.1	LeackyReLU	SGD	5.00E-04	5	31	2	84.05797101	149	10	85.84615385	6.09399271	0.018750747
													9	27		36	130			
71	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.2	LeackyReLU	Adam	5.00E-04	5	30	3	88.4057971	139	20	90.15384615	6.685991526	0.020572282
													5	31		12	154			
72	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.2	LeackyReLU	SGD	5.00E-04	5	27	6	79.41176471	147	12	82.15384615	6.266992092	0.019283053
													8	27		46	120			
73	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.3	LeackyReLU	Adam	5.00E-04	5	26	7	85.50724638	121	38	85.23076923	6.926992178	0.021313822
													3	33		10	156			
74	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.3	LeackyReLU	SGD	5.00E-04	5	23	10	76.8115942	136	23	86.46153846	5.877006769	0.018083098
													6	30		21	145			
75	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.4	LeackyReLU	Adam	5.00E-04	5	28	5	86.95652174	126	33	88.30769231	6.030950308	0.01855677
													4	32		5	161			
76	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.4	LeackyReLU	SGD	5.00E-04	5	24	9	68.11594203	123	36	77.53846154	6.817998648	0.020978457
													13	23		37	129			
77	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.5	LeackyReLU	Adam	5.00E-04	5	30	3	89.85507246	127	32	89.23076923	6.420997143	0.019756914
													4	32		3	163			
78	Training: F, Validating: F, Testing: G	9	Original	-	Cube	[128, 2]	-	0.5	LeackyReLU	SGD	5.00E-04	5	25	8	76.8115942	134	25	83.38461538	6.206000328	0.019095386
													8	28		29	137			
79	Training: F, Validating: F, Testing: B	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	22	11	78.26086957	51	268	45.99708879	17.84374213	0.025973424
													4	32		103	265			
80	Training: F, Validating: F, Testing: B	9	Original	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	24	9	75.36231884	50	269	50.80058224	42.20465183	0.061433263
													8	28		69	299			
81	Training: F, Validating: F, Testing: B	9	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	23	10	72.46376812	56	263	51.81950509	101.5539534	0.147822348
													9	27		68	300			
82	Training: F, Validating: F, Testing: B	9	Reverse Train	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	27	6	84.05797101	0	319	53.56622999	55.57138515	0.080889935
													5	31		0	368			
83	Training: F, Validating: F, Testing: B	9	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	23	10	69.56521739	40	279	48.61717613	93.28233361	0.135782145
													11	25		74	294			
84	Training: F, Validating: F, Testing: B	9	Reverse Test	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	25	8	79.71014493	0	319	53.56622999	107.2328646	0.156088595
													6	30		0	368			
85	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	121	28	87.08708709	176	83	79.33579336	39.10896444	0.072156761
													15	169		29	254			
86	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	138	11	89.78978979	206	53	85.42435424	9.109267473	0.016806767
													23	161		26	257			
87	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	10	125	24	87.38738739	184	75	80.07380074	25.3281281	0.046730864
													18	166		33	250			
88	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	10	137	12	90.99099099	204	55	84.87084871	9.89062953	0.018248394
													18	166		27	256			
89	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	133	16	88.58858859	190	69	82.10332103	12.93749762	0.023869922
													22	162		28	255			
90	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	139	10	89.48948949	200	59	83.76383764	10.89062214	0.020093399
													25	159		29	254			

91	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	139	10	91.29129129	209	50	85.42435424	57.12505746	0.105396785
													19	165		29	254			
92	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	135	14	89.18918919	196	63	82.65682657	30.64056492	0.056532408
													22	162		31	252			
93	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	133	16	89.18918919	194	65	83.2103321	49.34370971	0.091040055
													20	164		26	257			
94	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	131	18	88.28828829	188	71	81.73431734	30.18750143	0.055696497
													21	163		28	255			
95	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.1	LeackyReLU	Adam	5.00E-04	5	125	24	83.78378378	199	60	83.02583026	10.23431778	0.018882505
													30	154		32	251			
96	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.1	LeackyReLU	SGD	5.00E-04	5	127	22	86.18618619	195	64	81.73431734	10.57806897	0.019516732
													24	160		35	248			
97	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.2	LeackyReLU	Adam	5.00E-04	5	126	23	84.08408408	198	61	73.98523985	10.40624714	0.019199718
													30	154		80	203			
98	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.2	LeackyReLU	SGD	5.00E-04	5	134	15	87.38738739	197	62	81.73431734	8.999993563	0.016605154
													27	157		37	246			
99	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.3	LeackyReLU	Adam	5.00E-04	5	133	16	87.08708709	193	66	81.91881919	10.01561952	0.018479003
													27	157		32	251			
100	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.3	LeackyReLU	SGD	5.00E-04	5	133	16	88.25301205	194	65	82.10332103	9.093698502	0.016778042
													23	160		32	251			
101	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.4	LeackyReLU	Adam	5.00E-04	5	115	34	86.18618619	154	105	78.04428044	77.78119612	0.143507742
													12	172		14	269			
102	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.4	LeackyReLU	SGD	5.00E-04	5	135	14	89.18918919	193	66	82.84132841	25.35931683	0.046788407
													22	162		27	256			
103	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.5	LeackyReLU	Adam	5.00E-04	5	122	27	87.98798799	179	80	81.73431734	9.500032425	0.017527735
													13	171		19	264			
104	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.5	LeackyReLU	SGD	5.00E-04	5	136	13	88.28828829	195	64	82.28782288	9.609374285	0.017729473
													26	158		32	251			
105	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	124	25	84.38438438	199	60	81.18081181	9.359359741	0.017268191
													27	157		42	241			
106	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	133	16	88.88888889	196	63	82.65682657	9.640628099	0.017787137
													21	163		31	252			
107	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	129	20	87.98798799	201	58	83.39483395	31.17187119	0.057512677
													20	164		32	251			
108	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	133	16	89.18918919	192	67	82.28782288	9.515566111	0.017556395
													20	164		29	254			
109	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	126	23	89.18918919	190	69	83.39483395	9.046868086	0.016691639
													13	171		21	262			
110	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	140	9	90.69069069	202	57	84.31734317	11.07800007	0.020439115
													22	162		28	255			
111	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	134	15	88.88888889	200	59	83.02583026	25.01562023	0.046154281
													22	162		33	250			
112	Combination A	9	Original, Black Image Last	-	Cube	[256, 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	137	12	90.09009009	202	57	83.57933579	27.62500072	0.050968636
													21	163		32	251			
113	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	131	18	85.88588589	198	61	81.5498155	25.18749571	0.046471394
													29	155		39	244			
114	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	138	11	89.78978979	201	58	83.94833948	24.40619659	0.045029883
													23	161		29	254			
115	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	125	24	87.38738739	189	70	82.65682657	11.04687524	0.020381689
													18	166		24	259			
116	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	136	13	87.38738739	201	58	82.47232472	24.53124332	0.045260597
													29	155		37	246			
117	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	132	17	88.88888889	190	69	82.10332103	24.57807469	0.045347001
													20	164		28	255			
118	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	132	17	87.08708709	193	66	81.5498155	10.51562047	0.019401514
													26	158		34	249			
119	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	137	12	89.48948949	195	64	82.84132841	24.29682755	0.044828095
													23	161		29	254			
120	Combination A	9	Original, Black Image Last	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	136	13	89.48948949	191	68	83.39483395	26.89056325	0.049613585
													22	162		22	261			

121	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	140	9	92.79279279	203	56	85.2398524	841.9843698	1.553476697
													15	169		24	259			
122	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	138	11	91.29129129	197	62	83.39483395	9.45306325	0.017441076
													18	166		28	255			
123	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	138	11	88.28828829	198	61	81.36531365	11.56249642	0.021333019
													28	156		40	243			
124	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	136	13	90.99099099	195	64	83.76383764	9.28125	0.017124077
													17	167		24	259			
125	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	136	13	89.18918919	194	65	83.57933579	9.843698502	0.018161805
													23	161		24	259			
126	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	140	9	91.29129129	200	59	84.87084871	9.749938488	0.017988816
													20	164		23	260			
127	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	135	14	90.09009009	201	58	84.68634686	9.640572309	0.017787034
													19	165		25	258			
128	Combination A	9	Original, Black Image First	-	Cube	[256 , 128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	140	9	89.78978979	207	52	84.31734317	9.984318256	0.018421251
													25	159		33	250			
129	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	134	15	89.48948949	219	40	85.97785978	11.15626216	0.02058351
													20	164		36	247			
130	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	142	7	89.78978979	202	57	82.84132841	10.81250215	0.019949266
													27	157		36	247			
131	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	139	10	90.99099099	197	62	83.2103321	10.51562786	0.019401527
													20	164		29	254			
132	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	137	12	90.69069069	198	61	84.13284133	10.46869659	0.019314938
													19	165		25	258			
133	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	137	12	89.78978979	207	52	85.2398524	9.312433958	0.017181612
													22	162		28	255			
134	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	136	13	89.78978979	196	63	83.39483395	10.49994278	0.019372588
													21	163		27	256			
135	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	141	8	91.29129129	202	57	83.57933579	10.1561892	0.018738356
													21	163		32	251			
136	Combination A	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	138	11	91.59159159	196	63	83.94833948	10.65625048	0.019660979
													17	167		24	259			
137	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	96	6	96.01769912	135	7	95.40636042	5.406249523	0.019103355
													3	121		6	135			
138	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	98	4	93.80530973	134	8	92.22614841	4.624946833	0.016342568
													10	114		14	127			
139	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	97	5	94.69026549	136	6	95.05300353	4.765566111	0.016839456
													7	117		8	133			
140	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	96	6	93.80530973	136	6	94.34628975	5.343693972	0.018882311
													8	116		10	131			
141	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	95	7	94.24778761	137	5	94.69964664	4.828066349	0.017060305
													6	118		10	131			
142	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	96	6	95.57522124	135	7	94.34628975	5.437446117	0.019213591
													4	120		9	132			
143	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	94	8	95.13274336	137	5	95.40636042	5.515680552	0.019490037
													3	121		8	133			
144	Combination B	9	Original, Black Image First	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	98	4	93.36283186	138	4	93.63957597	4.703066587	0.01661861
													11	113		14	127			
145	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	5.00E-04	5	95	7	94.24778761	134	8	93.28621908	5.109318495	0.018054129
													6	118		11	130			
146	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	5.00E-04	5	94	8	93.36283186	133	9	92.93286219	4.968741894	0.017557392
													7	117		11	130			
147	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-04	5	97	5	93.36283186	137	5	92.22614841	5.109319448	0.018054132
													10	114		17	124			
148	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-04	5	99	3	87.16814159	138	4	89.75265018	5.171867847	0.018275151
													26	98		25	116			
149	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	Adam	1.00E-05	5	96	6	95.57522124	138	4	94.34628975	4.734321356	0.016729051
													4	120		12	129			
150	Combination B	9	Original, Black Image Last	-	Cube	[128 , 2]	-	0.15	LeackyReLU	SGD	1.00E-05	5	99	3	93.80530973	139	3	93.28621908	5.203128099	0.018385612
													11	113		16	125			

151	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	Adam	1.00E-06	5	100	2	12	112	93.80530973	139	3	16	125	93.28621908	5.031195879	0.017778077
152	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeackyReLU	SGD	1.00E-06	5	98	4	12	112	92.92035398	136	6	14	127	92.93286219	4.843694448	0.017115528
153	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	5.00E-04	5	92	10	10	114	91.15044248	117	25	13	128	86.57243816	15.03124285	0.053113932
154	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	5.00E-04	5	98	4	12	112	92.92035398	138	4	14	127	93.63957597	5.718503237	0.020206725
155	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-04	5	82	20	11	113	86.28318584	103	39	18	123	79.85865724	5.781186342	0.02042822
156	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-04	5	98	4	11	113	93.36283186	138	4	17	124	92.5795053	5.859307051	0.020704265
157	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-05	5	97	5	13	111	92.03539823	138	4	17	124	92.5795053	5.390564919	0.019047933
158	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-05	5	100	2	13	111	93.36283186	138	4	16	125	92.93286219	5.74994421	0.020317824
159	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-06	5	96	6	7	117	94.24778761	137	5	13	128	93.63957597	5.734313488	0.020262592
160	Combination B	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-06	5	101	1	16	108	92.47787611	141	1	19	122	92.93286219	5.531188726	0.019544836
161	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	ReLU	Adam	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.749943256	0.020317821
162	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	ReLU	SGD	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.499997139	0.019434619
163	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	LeakyReLU	Adam	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.749942064	0.020317816
164	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	LeakyReLU	SGD	5.00E-04	5	16	86	1	123	61.50442478	21	121	1	140	56.89045936	5.874995232	0.0207597
165	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	ReLU	Adam	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.578182459	0.019710892
166	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	ReLU	SGD	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.484374046	0.019379414
167	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	LeakyReLU	Adam	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.359372854	0.018937713
168	Combination B	9	Original, Black Image Last	-	Cube	GAP [2]	-	0.15	LeakyReLU	SGD	5.00E-04	5	0	102	0	124	54.86725664	0	142	0	141	49.82332155	5.624940872	0.019876116
169	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	Adam	5.00E-04	5	153	18	22	191	89.58333333	197	37	23	261	88.41698842	23.12499928	0.044642856
170	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	SGD	5.00E-04	5	162	9	19	194	92.70833333	206	28	19	265	90.92664093	10.60936713	0.020481404
171	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	5.00E-04	5	164	7	13	200	94.79166667	213	21	19	265	92.27799228	9.609373331	0.018550914
172	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	5.00E-04	5	166	5	24	189	92.44791667	210	24	28	256	89.96138996	10.39062333	0.020059118
173	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	Adam	1.00E-04	5	155	16	23	190	89.84375	201	33	29	255	88.03088803	10.57811642	0.020421074
174	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	SGD	1.00E-04	5	163	8	18	195	93.22916667	208	26	19	265	91.31274131	10.90630364	0.02105464
175	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	Adam	1.00E-05	5	159	12	14	199	93.22916667	208	26	19	265	91.31274131	8.843753576	0.017072883
176	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	SGD	1.00E-05	5	164	7	18	195	93.48958333	204	30	21	263	90.15444015	10.28124642	0.019847966
177	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	Adam	1.00E-06	5	163	8	21	192	92.44791667	214	20	22	262	91.89189189	10.82811928	0.020903705
178	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	LeakyReLU	SGD	1.00E-06	5	166	5	19	194	93.75	209	25	22	262	90.92664093	10.42181969	0.020119343
179	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-04	5	162	9	23	190	91.66666667	208	26	21	263	90.92664093	8.781252384	0.016952225
180	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-04	5	164	7	21	192	92.70833333	210	24	24	260	90.73359073	24.04686403	0.046422517

181	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-05	5	160	11	93.22916667	199	35	89.96138996	9.687443733	0.018701629
													15	198		17	267			
182	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-05	5	165	6	94.53125	210	24	91.89189189	9.140619755	0.017645984
													15	198		18	266			
183	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	Adam	1.00E-06	5	157	14	92.1875	195	39	89.18918919	9.359375715	0.018068293
													16	197		17	267			
184	Combination C	9	Original, Black Image Last	-	Cube	[128, 2]	-	0.15	ReLU	SGD	1.00E-06	5	162	9	94.01041667	208	26	91.31274131	10.64062524	0.020541748
													14	199		19	265			
185	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	KMN	0.15	LeakyReLU	Adam	5.00E-04	5	101	10	93.75	128	24	90.25787966	30.73240042	0.088058454
													7	154		10	187			
186	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	KMN	0.15	LeakyReLU	SGD	5.00E-04	5	102	9	91.54411765	130	22	87.96561605	6.012312412	0.017227256
													14	147		20	177			
187	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	KMN	0.15	LeakyReLU	Adam	5.00E-04	5	104	7	94.85294118	133	19	92.26361032	6.156127691	0.017639334
													7	154		8	189			
188	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	KMN	0.15	LeakyReLU	SGD	5.00E-04	5	105	6	94.48529412	138	14	92.83667622	6.093354464	0.017459468
													9	152		11	186			
189	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	-	0.15	LeakyReLU	Adam	5.00E-04	5	102	9	95.22058824	139	13	92.26361032	7.763175011	0.022244054
													4	157		14	183			
190	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	-	0.15	LeakyReLU	SGD	5.00E-04	5	104	7	95.95588235	138	14	91.97707736	7.296400785	0.020906593
													4	157		14	183			
191	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	KMN	0.15	LeakyReLU	Adam	5.00E-04	5	105	6	96.32352941	140	12	91.69054441	7.676266432	0.021995033
													4	157		17	180			
192	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	KMN	0.15	LeakyReLU	SGD	5.00E-04	5	105	6	95.22058824	141	11	91.11747851	195.5606706	0.560345761
													7	154		20	177			
193	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	Adam	5.00E-04	5	105	6	93.75	134	18	91.11747851	6.15097332	0.017624565
													11	150		13	184			
194	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	SGD	5.00E-04	5	103	8	92.64705882	131	21	88.53868195	6.656793833	0.019073908
													12	149		19	178			
195	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	Adam	5.00E-04	5	102	9	94.85294118	143	9	93.12320917	7.734848976	0.022162891
													5	156		15	182			
196	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	SGD	5.00E-04	5	105	6	97.05882353	141	11	93.40974212	27.28977609	0.078194201
													2	159		12	185			
197	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	104	7	93.75	137	15	93.40974212	6.041602373	0.017311182
													10	151		8	189			
198	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	105	6	93.38235294	130	22	89.11174785	7.30421257	0.020928976
													12	149		16	181			
199	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	103	8	93.01470588	139	13	92.55014327	6.516180992	0.018671006
													11	150		13	184			
200	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	92	19	88.97058824	114	38	85.67335244	6.30916667	0.018077841
													11	150		12	185			
201	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	104	7	96.69117647	143	9	93.40974212	7.444838285	0.021331915
													2	159		14	183			
202	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	102	9	95.22058824	139	13	92.83667622	7.858870506	0.022518254
													4	157		12	185			
203	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	105	6	95.58823529	143	9	92.55014327	7.576664209	0.02170964
													6	155		17	180			
204	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	102	9	95.22058824	139	13	91.11747851	7.542482376	0.021611697
													4	157		18	179			
205	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Normal	0.15	ReLU	Adam	5.00E-04	5	103	8	95.22058824	142	10	92.55014327	7.266129732	0.020819856
													5	156		16	181			
206	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Normal	0.15	ReLU	SGD	5.00E-04	5	104	7	94.48529412	141	11	92.26361032	7.27589798	0.020847845
													8	153		16	181			
207	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Normal	0.15	ReLU	Adam	5.00E-04	5	104	7	95.58823529	141	11	91.11747851	8.30903101	0.023808112
													5	156		20	177			
208	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Normal	0.15	ReLU	SGD	5.00E-04	5	103	8	95.95588235	138	14	92.83667622	8.039522886	0.023035882
													3	158		11	186			
209	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	ReLU	Adam	5.00E-04	5	104	7	95.22058824	141	11	92.83667622	7.228046179	0.020710734
													6	155		14	183			
210	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	ReLU	SGD	5.00E-04	5	106	5	96.32352941	142	10	92.55014327	7.402848482	0.0212116
													5	156		16	181			

211	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Unifrom	0.15	ReLU	Adam	5.00E-04	5	106	5	96.69117647	143	9	91.97707736	7.500495911	0.021491392
													4	157		19	178			
212	Combination D	9	Original, Black Image Last	Norm	Cube	[128, 2]	Xavier Unifrom	0.15	ReLU	SGD	5.00E-04	5	103	8	93.01470588	143	9	90.83094556	7.905737638	0.022652543
													11	150		23	174			
213	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Normal	0.15	ReLU	Adam	5.00E-04	5	103	8	94.11764706	136	16	92.55014327	6.178309441	0.017702892
													8	153		10	187			
214	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Normal	0.15	ReLU	SGD	5.00E-04	5	103	8	92.64705882	128	24	89.11174785	6.754443169	0.019353705
													12	149		14	183			
215	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	ReLU	Adam	5.00E-04	5	106	5	94.85294118	139	13	93.40974212	7.972138166	0.022842803
													9	152		10	187			
216	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	ReLU	SGD	5.00E-04	5	103	8	94.11764706	129	23	90.83094556	6.12070179	0.017537827
													8	153		9	188			
217	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	ReLU	Adam	5.00E-04	5	106	5	94.48529412	142	10	93.40974212	12.046875	0.034518266
													10	151		13	184			
218	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	ReLU	SGD	5.00E-04	5	104	7	94.11764706	131	21	91.40401146	6.92181325	0.019833276
													9	152		9	188			
219	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	ReLU	Adam	5.00E-04	5	89	22	88.97058824	113	39	85.10028653	6.718693018	0.019251269
													8	153		13	184			
220	Combination D	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	ReLU	SGD	5.00E-04	5	102	9	94.11764706	133	19	91.40401146	15.32000089	0.043896851
													7	154		11	186			
221	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	Adam	5.00E-04	5	126	18	88.125	192	67	81.91881919	10.08594298	0.018608751
													20	156		31	252			
222	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	SGD	5.00E-04	5	133	11	87.8125	199	60	81.91881919	9.719000101	0.017931735
													28	148		38	245			
223	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	Adam	5.00E-04	5	128	16	86.875	198	61	81.91881919	12.2159977	0.022538741
													26	150		37	246			
224	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	SGD	5.00E-04	5	132	12	90	199	60	83.39483395	13.0200007	0.024022142
													20	156		30	253			
225	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	123	21	86.5625	186	73	80.44280443	11.90625191	0.021967254
													22	154		33	250			
226	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	132	12	88.75	199	60	82.28782288	10.73431826	0.019805015
													24	152		36	247			
227	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	116	28	86.5625	180	79	80.81180812	47.97403121	0.088512973
													15	161		25	258			
228	Combination A	9	Original, Black Image Last	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	127	17	87.5	191	68	81.91881919	10.13406754	0.018697542
													23	153		30	253			
229	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	Adam	5.00E-04	5	133	11	90.9375	202	57	83.94833948	12.89062428	0.02378344
													18	158		30	253			
230	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	SGD	5.00E-04	5	136	8	90	197	62	82.10332103	10.90624952	0.020122232
													24	152		35	248			
231	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	Adam	5.00E-04	5	129	15	89.6875	214	45	86.900369	9.078061342	0.016749191
													18	158		26	257			
232	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	SGD	5.00E-04	5	131	13	89.6875	193	66	82.84132841	10.54687595	0.019459181
													20	156		27	256			
233	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	133	11	90.9375	216	43	85.2398524	9.984374762	0.018421356
													18	158		37	246			
234	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	136	8	90	200	59	83.39483395	9.296819687	0.017152804
													24	152		31	252			
235	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	141	3	90.9375	207	52	83.57933579	13.42187071	0.024763599
													26	150		37	246			
236	Combination A	9	Original, Black Image First	-	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	132	12	90	192	67	82.65682657	13.17188096	0.024302363
													20	156		27	256			
237	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	Adam	5.00E-04	5	133	11	87.8125	196	63	81.36531365	29.98437166	0.055321719
													28	148		38	245			
238	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Kaiming Normal	0.15	LeakyReLU	SGD	5.00E-04	5	122	22	87.5	182	77	81.73431734	14.890625	0.027473478
													18	158		22	261			
239	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	Adam	5.00E-04	5	122	22	87.5	183	76	80.81180812	14.65625072	0.027041053
													18	158		28	255			
240	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Xavier Normal	0.15	LeakyReLU	SGD	5.00E-04	5	116	28	84.375	170	89	79.33579336	15.79687595	0.029145528
													22	154		23	260			
241	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	118	26	85.3125	176	83	78.78228782	15.76555443	0.029087739
													21	155		32	251			
242	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Kaiming Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	119	25	88.4375	182	77	82.65682657	10.95311904	0.020208707
													12	164		17	266			
243	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	Adam	5.00E-04	5	123	21	88.4375	184	75	82.47232472	11.42187524	0.021073571
													16	160		20	263			
244	Combination A	9	Original, Black Image First	Norm	Cube	[128, 2]	Xavier Uniform	0.15	LeakyReLU	SGD	5.00E-04	5	118	26	86.875	177	82	81.91881919	14.53118992	0.026810314
													16	160		16	267			

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Tan Wei Jin
ID Number(s)	17ACB02302
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	DETECTING HEAD IN PILLOW DEFECT (HIP) BY USING DEEP LEARNING AND IMAGE PROCESSING TECHNIQUE

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 10 **%** **Similarity by source** Internet Sources: 6 % Publications: 6 % Student Papers: 3 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required, and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Signature of Co-Supervisor

Name: Dr. Aun Yichiet

Name:

Date: 16 April 2021

Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB02302
Student Name	TAN WEI JIN
Supervisor Name	Dr. Aun Yichiet

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Front Cover
√	Signed Report Status Declaration Form
√	Title Page
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.
尉進 陳	
(Signature of Student) Date: 16 April 2021	(Signature of Supervisor) Date: 16 April 2021