C O V E N T R Y
U N I V E R S I T Y

Faculty of Engineering, Environment and Computing

School of Computing, Electronics and Mathematics

MSc Data Science & Computational Intelligence

M08CDE Individual Project

Development of a Convolutional Neural Network for the detection of Railway Track Maintenance

Author: Thomas Staite

SID: 6422510

Supervisor: Dr. Mauro S. Innocente

Submitted in partial fulfilment of the requirements for the Degree of Master of Data Science & Computational Intelligence

Academic Year: 2018/19

M08CDE Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed:  Date: 14/08/2019

Please complete all fields.

First Name:	Thomas
Last Name:	Staite
Student ID number	6422510
Ethics Application Number	P91330
1st Supervisor Name	Dr Mauro S. Innocente
2nd Supervisor Name	Dr Sara Sharifzadeh

This form must be completed, scanned and included with your project submission to Turnitin. Failure to append these declarations may result in your project being rejected for marking.

Abstract

Railway track components have an on-going need to be periodically inspected, monitored, and maintained to ensure the safety and reliance of both the railway track itself as well as the trains that traverse along them. An exploration into the current approach used to complete the track inspection task reveals just how tedious, time-consuming, and labour-intensive it can become, with Rail Engineers being frequently required to manually navigate sections of the track, and make informed decisions on maintenance requirements, with further difficulties and repercussions; including unexpected delays to public transport service, as well as increased risk to work-related injuries, also adding to the pressure and main concerns of fulfilling this critical job-role.

This paper explores one possibility of meliorating this task by adopting deep-learning technologies along with relevant image processing tools to design and develop machine learning models, whom are capable of successfully automating the detection of maintenance requirements in track components.

A selection of Data Pre-processing tasks was applied to the dataset and proved paramount to the success of the project by ensuring the developed models were trained correctly and could generalise and perform well on new data. The specific pre-processing tasks that were employed were, dataset expansion through image slicing, affine transformations, and elastic distortions, labelling of the dataset into one of five classes, handling of the class imbalance issue, and down sampling.

The proposed machine learning solution involves a set of Convolutional Neural Network (CNN) architectures whose designs were inspired by previous studies that have been reviewed, have been developed in Python using Google's Colaboratory research tool, were trained and tested on an expanded dataset of 10,000 images, and then were compared against each other for evaluation of model performance. The best-performing model from the experiment managed to yield a test accuracy of 93.4% - misclassifying only 65 out of the 1,000 test images. A short Java application was then developed to consolidate the trained Python model with an attractive Graphical User Interface (GUI) into a single program.

Table of Contents

Acknowledgements

The delivery of this project report, the experimentations, and the final application would not have been possible without the help and guidance of my supervisor **Dr Mauro S. Innocente,** whom had provided constant support and feedback which has improved my work greatly.

I would also like to thank the writers and editors of various publications, listed in the bibliography and references section of this report. Their work has formed the basis for my research throughout the process of the project.

1 Introduction

1.1 Background to the Project

The inspection of railway track for the detection of maintenance needs remains a critical task in ensuring the smooth running of day-to-day railway operations and the safety of railway traffic, passengers, and the workforce. Traditionally, such a task becomes a job role for a trained railway engineer whom would periodically walk along the track lines performing necessary inspections of the railway components and forming a report of any maintenance that is required. As one can imagine, this manual process of inspection is expensive both in monetary and in time and can be very dangerous if scheduled during busy periods or on frequently used track lines. Furthermore, failure to undertake track inspection or miss-detecting existing defeats due to human error or misjudgement can have negative repercussions in the long-term - including further costly; and sometimes permanent, damage to the track and the trains that run on it, and disruptions; lateness and cancellations, to passenger journeys due to unexpected faults that could have been prevented through appropriate track maintenance. There are also increased safety concerns for the passengers who use these trains and the workforce that operate on the lines if the track is not being maintained properly.

The Office of Rail and Road (ORR) is the independent safety and economic regulator for Britain's railways. They continuously gather data and publish frequent official reports and summary statistics on five important divisions (usage, passenger experience, performance, finance, and health and safety) and can be used as a guidance for analyses of the overall performance of Britain's rails through time. In one recent report; published 24th May 2019, it was calculated that nationally, 13.7% of trains in Q4 (1st October – 31st December) of the year 2018 were late by more than 5 minutes to their destination, with 4.3% of trains being reported as 'Cancelled or Significantly Late'. (Office of Rail and Road, 2019) Although no specific details have been released explaining the exact reasoning behind the levels of lateness and cancellations, it can be at least suggested that a proportion of these cases is somewhat related to maintenance issues; such as unexpected track or train component failure, resulting in the need for emergency, unscheduled maintenance.

Furthermore, separate reports by the same company look at levels of injuries and fatalities to both train passengers and the workforce that operate on the rails. In the case of the workforce section, ORR reports 2 cases of fatalities (death) in the year 2017 with a further 6,661 workforce injuries - down 2.1% from 6,801 the previous year, but still averaging **over 18 injuries a day**. 164 of these reported injuries were classified as major injuries and it is likely that additional injuries may have occurred but were not reported, perhaps for personal reasons or because of the magnitude of the injury. (Office of Rail and Road, 2018) Again, no specific information has been published regarding 'why' and 'how' these accidents occurred, but it can be at least suggested that a small proportion of these incidents may have occurred when staff were performing inspection or maintenance duties.

Considering the statistics highlighted in the above reports, it comes as no surprise that railway companies and the Government alike are constantly researching and developing new solutions to improve the safety, efficiency, and value of the public rail transport service across the five aforementioned divisions.

The research in this report focuses on one specific area of Data Science that has become increasingly prominent across many businesses in various sectors, with public transport being no exception to this – the introduction of Machine Learning techniques for constant improvement and

personalisation of products and services. More specifically, this study looks at the possibilities of automating the process of inspecting track segments for maintenance needs through an Image Processing application encoded in the form of a Convolutional Neural Network (CNN). The primary benefit of automating such task will be the obsoleting of the engineer's job role to manually inspect the track, providing many benefits for the business:
First and foremost, there are associated cost reductions since engineers will no longer be required to inspect the track themselves and so this time and energy can be focused on other critical tasks that are not yet automated. Development and installation of the automated system will likely incur some initial expense, however the running costs of maintaining this system over time will be much lower than the running costs of recruiting, training, and paying extra engineer salaries. With that being said, in the long run, the business will expect to report a reduction in running costs for this particular task.

Secondly, the time required to complete the inspection task will be much lower as track segments can be constantly reviewed by the CNN at much faster rates than it would take for engineers to manually traverse and examine the track, and then build reports. Moreover, if the CNN is able to perform at near-human level, then it could even completely omit errors like poor judgement or miss-detection on the engineer's behalf. Furthermore, constant real-time results from the CNN will allow for quicker maintenance scheduling which will help in the prevention of lateness/cancellations of train journeys caused by track or train component failure.

Finally, since engineers will no longer need to physically inspect the railway track, one could expect a positive impact to the number of reported workforce injuries that have occurred when inspecting or maintaining track components. This helps make the railway transport service safer for everyone and could also save the company on costs like compensation being paid due to work-related injuries.

Research will be carried out to identify some existing attempts in Image Processing with Neural Networks and the selected relevant literature will be reviewed to understand what has worked well in other academic's use cases and if there are any gaps in the research that could be addressed. The experiments that will follow will be based on this literature review and the final CNN architectures will be trained and tested, with the approaches taken compared with those studied in the literature review. The performance of the CNN will be analysed, and a conclusion will be made as to how well the network is able to automate this track inspection task.

In terms of the actual maintenance requirements that are targeted in this report, the CNN architecture will aim to identify the sleepers of the track that are either missing one or both fasteners (the clips that secure the track to the sleeper). In addition to this, if there is a moderate amount of excess ballast (the layer of rock the sleepers sit on) which makes the fastener visible but prevents us from determining whether the condition of the fastener is intact, then these images will be placed into a separate class. Likewise, if there is a substantial amount of excess ballast that covers the entire section of the sleeper and thus the entire fastener, then it is impossible to detect the fastener, let alone evaluate its condition. These images will be placed into another separate class.

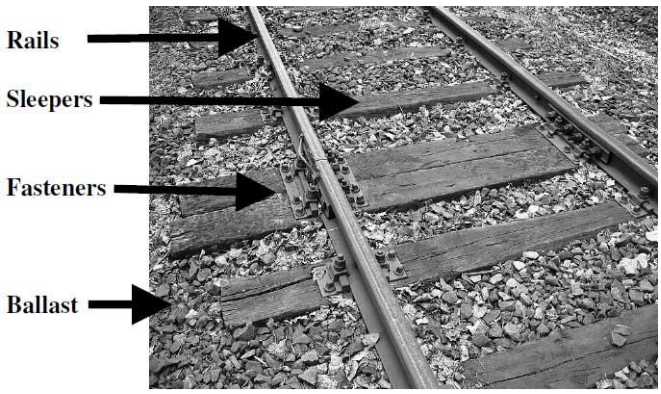


Figure 1 Components of a railway track. [fig. 1]

1.2 Project Aims and Objectives

1.2.1 Project Aim

The aim of the project is to design, development, and test a Convolutional Neural Network for an Image Processing classification task which aims to classify images of railway track into those that require maintenance and those that do not.

1.2.2 Objective – Gather and Prepare dataset

A suitable data set of images will be gathered and pre-processing techniques will be applied to images in the dataset to help prepare it for training the models.

1.2.3 Objective – Design and Develop CNN

Convolutional Neural Network architectures will be designed, developed, trained, tested, and compared to identify the best performing model for this use case.

1.2.4 Objective – Design and Develop GUI

A Graphical User Interface will be designed and developed in Java.

1.2.5 Objective – Consolidate CNN and GUI

The CNN and GUI will be consolidated into one product.

1.3 Overview of This Report

The remainder of this report will be set out as follows:

- **Section 2**. Literature Review – The literature review will explore previous studies and attempts in similar problem areas relating to data preparation, image processing, and artificial neural network. The literature will be reviewed and related to the work that is intended to be carried out in this project. Any gaps in the literature will be identified and addressed.

- **Section 3**. Methodology – The methodology section will discuss the steps that are to be taken during the process of the project. This will include the important sub-tasks including data gathering, experiment setup, model design, development, training, and testing, experiment results, and GUI development.

- **Section 4**. Experiment Setup – This section will describe how the experiment will be conducted. It will provide vital information on the number of tests to be performed, the details of each test, the objectives and reasoning's behind each test, and what performance metrics will be employed to measure and compare the results obtained.

- **Section 5**. Data Pre-processing – Here the image dataset that has been acquired will be introduced and an overview will be given to inform the reader of some of its key details. Limitations of the dataset in regard to using it for training of a machine learning will be identified and necessary data preparation tools and techniques that will combat these limitations will be introduced, discussed, and applied.

- **Section 6**. Solution Design – This section will cover the specific design of each of the CNN architectures and will give information about model configurations and hyper-parameters

- **Section 7**. Implementation – The solutions that have been designed will be developed in Python using an Anaconda Jupyter Notebook to document the work and store the code. Details of the model implementations and training will be provided in this section.

- **Section 8.** Experiment Results - Each of the trained CNN architectures will undergo testing on a 'test dataset'. The performance of each model will be documented in this section in the form of accuracy and loss graphs, Confusion Matrices, and Performance Histograms.

- **Section 9**. GUI Development - A Graphical User Interface will be designed and developed and once the experiment has concluded, the best performing model will be consolidated with this GUI. Work relating to this will be documented in this section.

- **Section 10**. Project Management – Project management covers any materials created, tools used, and approaches taken to help with the management and smooth-running of the project. Materials like meeting records, project Gantt chart, and report versioning are discussed in this section.

- **Section 11**. Critical Appraisal – This section will discuss my approach to this project, what work was completed and the results that were achieved, and any issues that were faced throughout the process of the project.

- **Section 12**. Conclusions – The results obtained from the experiment will be reviewed and conclusions will be drawn from the work completed. Any further improvements to the work will be identified and listed under a 'future work' section.

- **Section 13**. Student Reflection - The benefits and limitations of both the approach taken to the project as well as the content of the project itself will be identified and areas of potential improvement will be discussed. This section can be used for future reference to aid with personal skill development.

- Bibliography and References – Here a list of references link the external materials that have been pivotal to the development of the project.

- List of Figures – Every figure that has been used in this report will be listed here. Any figures from external sources will be referenced and hyperlinks to the source website will be provided.

- List of Tables – Every table that has been used in this report will be listed here. Any tables from external sources will be referenced and hyperlinks to the source website will be provided.

- Appendix – The final section of the report is the appendix. This will include any other figures and tables that are relevant to the project.

2 Literature Review

2.1 Convolutional Neural Network Architecture

Convolutional Neural Networks, CNNs, or ConvNets; based on the classical convolutional neural network proposed by LeCun et al (LeCun, 1989), are a class of deep neural networks that have proven very effective in areas such as image recognition and classification. Previous studies have seen various applications of CNNs that can identify and separate faces as part of powering vision in robots, as well as recognition of various traffic signs in self-driving cars.

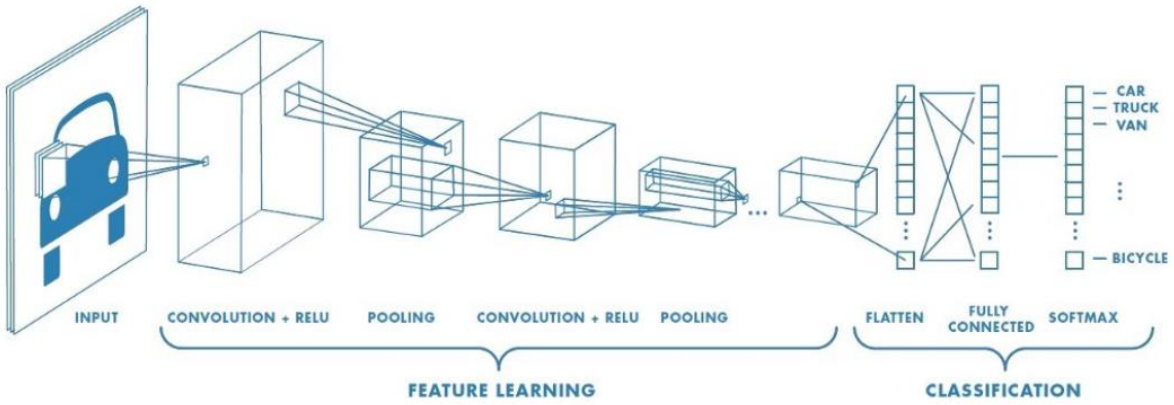


Figure 2 An example CNN for the classification of transport modes. [fig. 2]

The basic concept behind a CNN is to take in an input image, assign importance (learnable weights and biases) to various aspects of the image and be able to differentiate one from the other.

CNNs are able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. This architecture performs a better fitting to the image dataset than a standard Neural Network due to the reduction in the number of parameters involved as well as the reusability of weights. (Sumit Saha, 2018)
There are four main operations performed by the CNN which help it to effectively carry out its classification task. These form the basic building blocks of *every* CNN architecture:

1. Convolution
2. Non-Linearity (ReLU)
3. Pooling / Sub Sampling
4. Classification (Fully Connected Layer)

- The Convolution Step

The objective of the Convolution operation is to extract the high-level features such as edges and curves, from the input image. CNN architecture does not have to be limited to just one Convolutional Layer, conventionally the first convolutional layer is responsible for capturing low-level features like edges, colour, gradient orientation, and subsequent convolutional layers allows the architecture to adapt to the high-level features as well, resulting in a network that has understanding of images in the dataset, similar to our understanding.

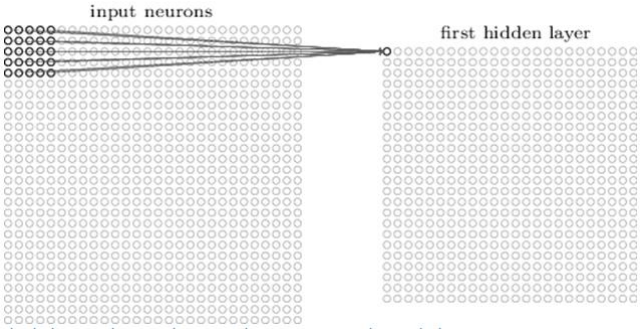


Figure 3 Applying the Convolution operation on the input image. [fig. 3]

There are three main components that make up the convolutional layer. The first component is the **input image**. In the case depicted by the image above, our image has dimensions 32 x 32 x 1 (32 pixels wide, 32 pixels high, and only one colour channel).

The second component is a **filter** which is a matrix of weights (or parameters) and can be thought of as a feature identifier. It covers an area of the input image called a **receptive field**. In the case above, the filter has dimensions 5 x 5 and so covers a total of 25 pixels for each position it is placed on the input volume. It is important to note that the depth of the filter has to be the same as the depth of the input. The filter that has been mentioned is first positioned in the top left corner of the input image and convolves around the image, where it multiplies the values in the filter with the original pixel values of the image (it computes the element wise multiplication). These multiplications are summed up to obtain a single number.

This process is repeated for every unique location in the input volume that the filter can cover (once we have calculated the first number, we move the filter one unit to the right, then one unit to the right again, and so on). Every unique location on the input volume produces a number. After convolving the filter over all possible locations, we are left with a 28 x 28 x 1 matrix of numbers – this is our third component and is called a **feature map**. (Adit Deshpande, 2016)
The reason that the above feature map is a 28 x 28 matrix is because there are 784 different locations that our 5 x 5 filters can fit on our 32 x 32 input image. These 784 numbers are mapped to our 28 x 28 array.

The size of the resulting feature map is controlled by three parameters that can be decided by the programmer during development of the network:

Depth - Increasing the number of filters that will convolve over the input image will increase the depth of the feature map. For example, if we had 10 filters, then we would have 10 feature maps.

Stride – this is the number of pixels by which we move the filter matrix across the input matrix. When the stride is 1 then we move the filters one pixel at a time, a stride of 2 will move the filters two pixels at a time, and so on. The larger the stride, the smaller the size of the feature map.

Zero-padding – Sometimes, it is convenient to pad the input matrix with zeros so that we can preserve features located at the borders of our images. Zero-padding can also help us control the size of the feature maps and make them equal to the size of the input matrix if it is desirable. (Madhi Varman, 2018)

The following formula can be used to calculate the dimensions of the resulting feature map:

$$output\ width\ = \left[\frac{W - F_w + 2P}{S_w}\right] + 1$$
$$output\ height\ = \left[\frac{H - F_h + 2P}{S_h}\right] + 1$$

W: $input\ width$
H: $input\ height$
F_w: $convolution\ filter\ width$
F_h: $convolution\ filter\ height$
P: $convolution\ padding\ size$
S_w: $horizontal\ stride\ amount$
S_h: $vertical\ stride\ amount$

Additional information on this formula can be studied by reading the following article published by Dang Ha The Hien: https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807

Additional information on Strides and Padding can be studied by reading the following article published by Jason Brownlee: https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/

- Non-Linearity

ReLU (Rectified Linear Unit) is a non-linear operation that is usually applied after every Convolution operation. Its output is given by:

$$Output = Max(0, input)$$

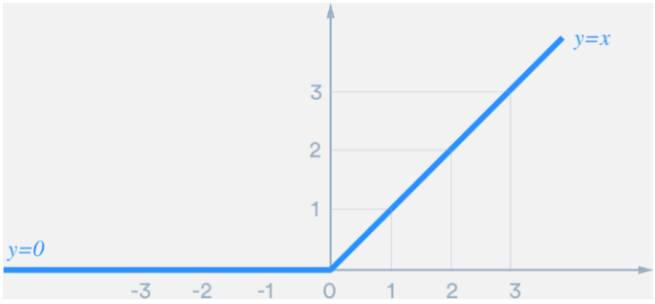


Figure 4 ReLU activation function. [fig. 4]

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map to zero. The purpose of doing this is to introduce non-linearity into our CNN, since most of the real-world data that we would be working with would require our CNN to learn non-linearity (the convolution step previously explained is a linear operation; element wise matrix multiplication and summation, so we account for non-linearity by introducing a non-linear function like ReLU). ReLU has been found in practice to outperform other non-linear function like **tanh** and **sigmoid** in most situations as gradient computation is very simple (either 0 or 1 depending on the sign of x), the computational step is also easy: any negative values are just set to 0.0 – no exponentials, multiplication, or division operations (allowing the network to train much faster), and it helps alleviate the vanishing gradient problem.

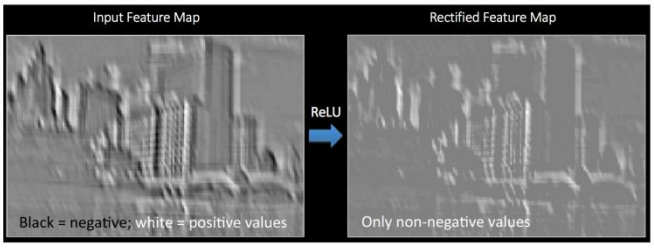


Figure 5 A Rectified Feature Map after applying the ReLU operation. [fig. 5]

Additional information on the Vanishing Gradient problem can be studied by reading the following article published by Nikhil Garg: https://www.quora.com/What-is-the-vanishing-gradient-problem

- The Pooling Step

Pooling (also known as subsampling or down-sampling) reduces dimensionality of each of our feature maps while still retaining most of the significant information. The three most common types of pooling that are used are: Max, Average, and Sum.

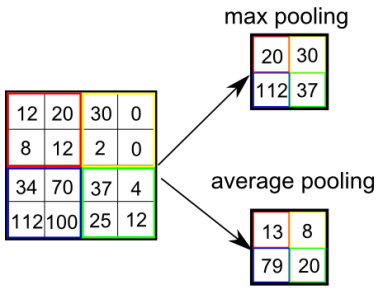


Figure 6 Two possible Pooling operation – Max and Average. [fig. 6]

In the case of Max Pooling, we create a spatial neighbourhood (for example, a 2x2 window) and select the largest element from the rectified feature map within that window – average pooling will take the average of all elements, whilst sum pooling will take the sum of all the elements. It has been found in practice that Max Pooling tends to work the best for CNN application.

It is important to note that the pooling operation is applied separately to each feature map – so if we have 10 feature maps, then we would receive 10 output maps after applying the pooling operation.

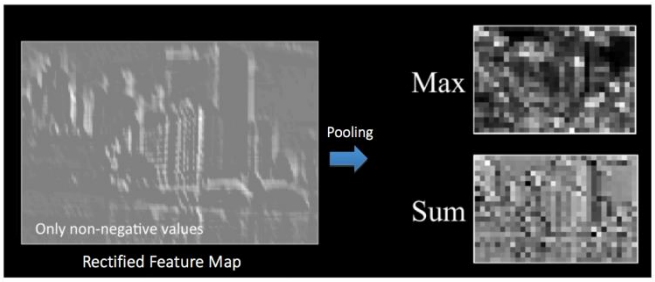


Figure 7 Outputs after applying Max and Sum pooling. [fig. 7]

As well as reducing the dimensionality of our feature maps, the pooling operation also performs a few other important tasks, this includes: reducing the number of parameters and computations in the network (preventing the introduction of overfitting), making the network invariant to small transformations, distortions and translations in the input image (small distortions in the input images will not change the output of the Pooling operation since we take the maximum / average value in a local neighbourhood), and also helping us to arrive at an almost scale invariant representation of the input image – this is very important as it gives the CNN the capability to detect objects regardless of their exact location within the image.

- The Fully Connected Layer (FC)

This type of layer is similar to one you would find in a standard Multi-Layer Perceptron (MLP) network. The output from the convolutional and pooling layers will represent high-level features of the input image. The purpose of the fully-connected layer is to take these features and use them for classifying our image into one of the pre-defined classes.

In CNN architecture we tend to use the **Softmax** activation function in the output layer to allow for the classification to take place – the Softmax function produces a vector that represents the probability distributions of a list of potential outcomes, which, as you can expect, have probabilities that sum up to one.

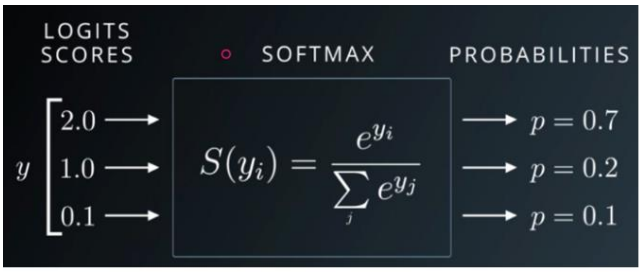


Figure 8 Softmax function. [fig. 8]

- Training using Backpropagation

Training of the Convolutional Neural Network can briefly be summarized by the five steps below:

Step 1: We initialize all filters and parameters (weights and biases) to random values and select values for our hyper-parameters (for example: number of filters, filter sizes, architecture of the network, etc.).

Step 2: The network completes a **forward pass** by taking a training image as input and passing it though each layer of the network (performing Convolution, ReLU, Pooling, and finally Classification operations) to obtain the output probabilities for each of the pre-defined classes – Since our parameters have been initialized with random values, we can expect our output probabilities to also be random.

Step 3: The total error for each class at the output layer is calculated and a summation across all classes is completed – it is common to use the Mean Square Error (MSE) error function which is defined below:

$$COST\ (C) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

n: $number\ of\ output\ classes$
Y_i: $target\ probability$
$\hat{Y}_i$: $output\ probability$

Step 4: Using backpropagation we calculate the gradients of the error with respect to each individual weight in the network and use **gradient descent** techniques to perform updates to the values of the filters, the weights, and the biases. Re-training the network with the same input image should yield a probability vector that contains values which are closer to the desired values. This means that the network has *learnt* to classify this particular image correctly by adjusting its trainable parameters.

Step 5: The network now has to repeat steps 2 – 4 with all images in the training set to try and find the best parameter values that yield the lowest overall cost value.
This training process can be repeated for a number of iterations (epochs), starting with different initial parameter values in an attempt to find the globally optimal settings for parameter values that will result in the best performing network.

The hyper-parameters of a Neural Network model help define the networks behaviour and its approach to the learning process. Convolutional Neural Networks (CNNs) have an extended set of hyper-parameters (in comparison to the standard MLP) to cater for the behaviour of the 'Convolutional' and the 'Pooling' layers. The set of tuneable hyper parameters in a Convolutional Neural Network architecture is as follows:

- Model Design Hyper-parameters

<u>Convolutional Layer & Activation function</u>
- Filter dimensions
- No. of filters
- Stride size
- Padding size (if applicable)
- Activation function (ReLU, Leaky ReLU, ELU, etc.)

<u>Pooling Layer</u>
- Type of pooling (Max, Average, Sum, etc.)

<u>Fully Connected Layer</u>
- Number of fully connected layers
- Number of neurons in each FC layer
- Initial weight and bias values
- Type of activation function used
- Loss Function used

<u>Training Hyper-parameters</u>
- Learning rate
- Number of epochs
- Batch size

2.2 Review of previous studies

Choosing the class of neural network to implement for a given use case is somewhat intuitive, deciding on the actual design of the architecture and determining optimal hyper-parameter values the model will possess, along with careful, considerate pre-processing and preparation of the dataset, on the other hand, is some of the most difficult aspects of neural network design and development and remains at the forefront of research within the field of Artificial Neural Networks, with many researchers and enthusiasts experimenting with various combinations of network layouts and hyper-parameter values in attempt to discover architecture combinations that provide optimal performance for given circumstances.

Patrice Y. Simard et al from Microsoft Research wrote a paper titled 'Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis'. In this paper they explained how there is currently a "*confusing plethora of different neural network methods that are used in the literature and in industry*" and demonstrated some of the current 'best practices' in ANN work through implementations of their own networks, which served the application of recognising handwritten digits using images from the MNIST dataset – a benchmark dataset of segmented images of handwritten digits, each with dimensions 28 x 28 pixels. (Patrice Y. Simard et al, 2003)

Optimal performance of their models was achieved by following two 'essential' practices. The first being related to the data preparation phase. They created a "*new, general set of elastic distortions that vastly expanded the size of the training set*" – in machine learning, we favour large amounts of training data as it reduces opportunities to introduce overfitting and also helps with model generalization capabilities, leading to sophisticated and accurate data modelling. The second 'essential' practice they followed in their work was the selection of the Convolutional Neural Network for an image processing task. They decided to compare the developed CNN with a standard Multi-Layer Perceptron Model to compare model performance, demonstrate the advantages of the CNN over the MLP for image processing, and emphasize the general importance of careful network class selection.

In regard to the first practice, Patrice and his colleges experimented by building networks that trained on three separate datasets - firstly on a dataset that had underwent no type of distortion (i.e. the original dataset), then on a dataset whose instances had underwent affine distortion, and then finally on a separate dataset whose instances had underwent elastic distortion. The results were compared and have been re-created in table 1. For the affine distortions, Patrice et al explained that the application of simple distortion techniques (translations, rotations, skewing) could be generated by applying affine displacement fields to the images: -

"*This is done by computing a new target location for every pixel in the image with respect to its original location. For instance, if* $\Delta x(x,y)=1$, *and* $\Delta y(x,y)=0$, *this indicates that the new location of every pixel is shifted by 1 to the right. Similarly, if the displacement field was* $\Delta x(x,y)= \alpha x$, *and* $\Delta y(x,y)= \alpha y$, *then the image would be scaled by the amount* α".

From the experiments conducted with affine transformations it was found that these greatly improved the results on the MNIST database in comparison to just using the original dataset. However, they found that they were able to obtain even better results when elastic deformations were used.

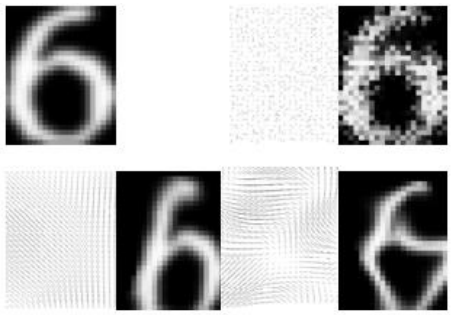


Figure 9 Top Left: Original Image. Top Right and Bottom: pairs of displacement fields with various smoothing levels, and the resulting image from application of the displacement field. [fig. 9]

Their elastic image deformations were created
"*by first generating random displacement fields, that is* $\Delta x(x,y) = rand(-1,+1)$ *and* $\Delta y(x,y) = rand(-1,+1)$*, where rand(-1,+1) is a random number between -1 and +1, generated with a uniform distribution. The fields* Δx *and* Δy *where then convolved with a Gaussian of standard deviation* σ*(in pixels). If* σ *is large, the resulting values are very small because the random values average 0."*

They then chose to normalise the displacement field (to a norm of 1) making the field close to constant, with a random direction. If σ is small, the field looks like a completely random field after normalization (as depicted in Figure 1, top right). For intermediate σ values, the displacement values look like elastic deformation, where σ is the elasticity coefficient. Finally, the displacement fields were multiplied by a scaling factor α that controls the intensity of the deformation.

Figure 9 top right shows an example of a pure random field($\sigma = 0.01$), bottom left shows an example of a smoothed random field ($\sigma = 8$) and bottom right shows an example of a smoothed random field with too much variability($\sigma = 4$). Patrice and his colleges indicated that if σ is large, the displacements become close to affine, and if it is very large, the displacements become translations. After experimenting with various values of σ and α, it was found that their best results were obtained when $\sigma = 4$ and $\alpha = 34$.

As mentioned above, the second 'essential' practice they defined was demonstrated by considering and comparing two types of architectures for their MNIST data set. The two architectures they compared were a two-layered multi-layer perceptron network and a Convolutional Neural Network. The error functions they employed were cross-entropy (CE) and mean squared error (MSE), however the latter was only used once during their experiments and so it seems rather redundant in this study. With that being said, when it comes to evaluating the results they obtained, the focus will only be on those that used CE as the error function. During the experiment, they that the CNN outperformed the MLP on every experiment that was conducted. This perhaps does not come as a surprise as the CNN had been found on many occasions to be much better suited for image processing tasks than other ANN architectures, due to the spatial topology that is well captured by the CNN as well as the Local Connectivity and Parameter Sharing capabilities of the network - both of which help reduce the number of learnable parameters; preventing the introduction of overfitting as well as significantly reducing network learning time.

The CNN architecture that was developed in this study was composed of the following layers:
One input layer which was a 29 x 29 pixel matrix. This represented the input image. The original input image size was 28 x 28, however due to their configurations (as seen below), the nearest value which generated an integer size after 2 layers of convolution is 29 x 29.

One Convolutional layer followed by a Pooling/sub-sampling layer. The convolutional layer consisted of five filters each with pixel dimensions 5 x 5. Patrice and his colleges found that fewer than 5 different features decreased performance, while using more than 5 did not improve it. They also used a stride of one, and no padding; as according to their study, it did not improve the performance significantly. It was decided that a sub-sampling of the layer by a factor of 2 would also be applied. With these hyper parameter values, we can calculate the dimensions of their feature maps after both convolution and pooling operations had been completed. Each feature map had size (29-3) / 2 = 13. This then, resulted in five 13 x 13 feature maps.

There was then a second combination of Convolutional and Pooling layers. This time the convolutional layer consisted of fifty filters (as opposed to five in the previous layer), again they experimented with this value and found that 25 features decreased the performance while 100 features did not improve it. Each filter had dimensions 5 x 5, a stride of one, and no padding. The sub-sample factor also remained at two. The size of the feature maps once the convolution and pooling operations had been applied were calculated to be (13-3) / 2 = 5. This then, resulted in fifty 5 x 5 feature maps.

A trainable classifier in the form of 2 fully connected layers formed the penultimate and final layer of the network. The number of hidden units in the penultimate layer is paramount in controlling the capacity, and generalization of the overall classifier. After experimenting with a range of values it was found that 100 hidden units seemed to provide the best performance for their case.

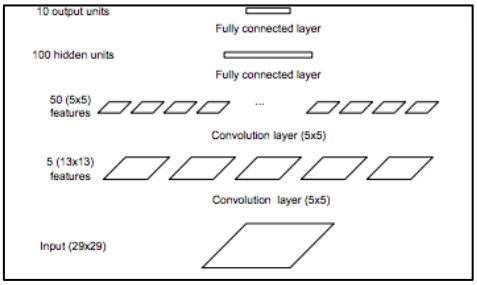


Figure 10 The CNN architecture developed by Patrice et al. [fig. 10]

The MNIST dataset that has been gathered to train both the CNN and the MLP consisted of 70,000 instances. The first 50,000 instances were used to train the model, the next 10,000 were used for validation and parameter adjustment, and the final 10,000 were used to test the model. The results reported below on the test set were done with the parameter values that were optimal on validation.

Algorithm	**Distortion**	**Error**
2-layer MLP (CE)	None	1.6%
2-layer MLP (CE)	Affine	1.1%
2-layer MLP (CE)	Elastic	0.7%
Simple Conv (CE)	Affine	0.6%
Simple Conv (CE)	**Elastic**	**0.4%**

Table 1 Comparison of MLP and CNN with various distortion techniques [tbl. 1]

Several interesting results have been captured in the results table. First and foremost, it is evident that elastic deformations have had a considerable impact on the performance, both for the MLP

and the CNN architectures. A reduction in errors of 0.4% and 0.2% was achieved when using elastic deformations over affine transformations for MLP and CNN, respectively. In addition to this, we have further evidence to prove that in general CNN architectures outperform MLP when it comes to image processing tasks as both of the CNN results obtained have lower error values than the results obtained for the MLP architectures.

The comparison between two techniques used in synthesizing training instances and the levels of performance they can provide on differing network architectures is relevant and useful to the work that will be carried out in my own study. There is however a gap in this study that has not been addressed by Patrice and his colleges – how would the performance of the CNN architecture compare on a single dataset which contains a proportion of instances that have underwent no transformations, a proportion of instances that have underwent affine transformations, and a proportion of instances that have underwent elastic transformations? Could an even further expansion of the dataset through applying both of these synthetic manufacture techniques yield even lower error rates?

One approach that will be taken in my research is to experiment by training the CNN architecture with three separate datasets - one which contains instances that have underwent only affine transformations, one which contains instances that have underwent only elastic transformations, and one that contains instances that have underwent BOTH affine and elastic transformations. Calculating the error values obtained for each of these experiments will help me to compare the results with those obtain by Patrice and his colleges and see whether similar reductions in error occur when using elastic distortions as opposed to affine transformations. Moreover, by experimenting with the third dataset, it will help me answer the questions as to whether applying both affine and elastic distortions to the same dataset provides even further reduction in the error values.

A paper titled ‘Deep convolutional neural network for detection of rail surface defects’, published by S. Faghih-Roohi et al in July 2016 looked an application of CNN similar to the one central to this study. In this paper, S. Faghih-Roohi and his colleges proposed a deep convolutional neural network (DCNN) solution to the analysis of image data for the detection of rail surface defects. This paper focused solemnly on the actual railway lines that trains run on. (S. Faghih-Roohi, 2016) The image dataset that they built to train their CNN architecture came from ‘*many hours of automated video recordings’* which were collected from a camera that was mounted on a measurement vehicle and captured the top view of the railway tracks. They estimated that this data covered approximately 350 kilometres of track, equivalent to 700 kilometres of rail. Among the collected frames, they manually labelled 22,408 objects as belonging to 1 out of 6 classes (normal, weld, light squat, moderate squat, severe squat, and joint).

‘*The weld class corresponds to those parts of the track surface where the rails are welded together to form one continuous rail, and in most of the cases in the images, these are hardly distinguishable from the normal rail surface even by the human eye when the weld is in good health condition. Squats are a type of surface-initiated track defects. (Maria Molodova et al, 2014) There are different classifications of the squat types based on their severity and size. Insulated joints electrically separate two consecutive track sections with an insulating material.’*

A sample of images depicting each of these classes can be seen below. In terms of the class distribution, S. Faghih-Roohi and his colleges report 985 welds, 938 light squats, 562 moderate and severe squats, and 755 rail joints. Out of each class, 10% of the samples were reserved for testing while the remaining 90% were reserved for training.

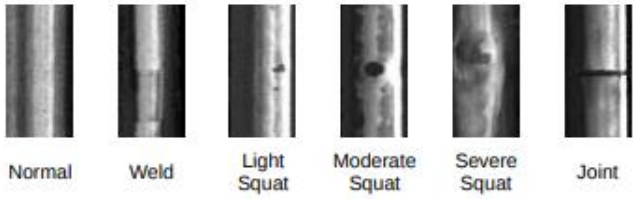


Figure 11 A sample of the images used in S. Faghih-Roohi and his colleges study [11]

When preparation of the data was underway, it was found that there was a '*huge imbalance of the class sizes'* and so to combat this S. Faghih-Roohi and his college chose to randomly under sample the majority class (the 'normal' class) so that it contained a similar number of instances as the other 5 classes.

Random under-sampling is a data preparation technique that is used in machine learning to modify unequal data classes in an attempt to create more balanced data sets. Random under-sampling is performed by preserving all of the instances in the minority classes while randomly and uniformly discarding a number of instances from the majority class, resulting in a balancing of the number of instances in each class. One negative repercussion of doing this is that it can potentially lead to loss of valuable information that can be used to help the model fit and generalize to the data.

Alternative methods to create balanced datasets included taking the approach that Patrice and his colleges did, which was to synthetically manufacture new instances through the means of affine transformations and elastic distortions. The quantity of new instances to manufacture can be adjusted for each class to both expand the dataset and tackle the class imbalance problem.
One of the other main focal points of this research is to review the specific CNN architectures that were used in this research. It was reported that three DCNN structures (small, medium, and large) were considered for implementation. The approach taken here was to find the set of model hyper-parameters that lead to the highest classification accuracy and then to use these hyper-parameters for building the DCNN model. Training parameters such as the learning rate were also adjusted during initial test runs. Table 2 summarizes the details of each DCNN architecture that was developed in this study.

Deep convolutional neural network				
Type of layer		Small	Medium	Large
Convolutional layer 1	Feature maps Filter size	6 17 × 9	10 9 × 5	20 9 × 5
Max-pooling layer 1	Filter size	3 × 3	2 × 2	2 × 2
Convolutional layer 2	Feature maps Filter size	10 8 × 3	20 9 × 6	40 9 × 6
Max-pooling layer 2	Filter size	3 × 3	2 × 2	2 × 2
Convolutional layer 3	Feature maps Filter size	- -	30 4 × 2	60 4 × 2
Max-pooling layer 3	Filter size	-	2 × 2	2 × 2
Fully-connected layer 1	Number of nodes	70	120	320
Fully-connected layer 2	Number of nodes	18	30	80
Fully-connected layer 3	Number of nodes	-	-	8
Estimated number of parameters (weights)		22000	120600	644200

Table 2 Summary of the architecture and parameter settings in the study by S. Faghih-Roohi et al [tbl. 2]

To understand the structure of the architectures built we can take the case of the medium DCNN. This model consists of three convolutional layers, three max-pooling layers, and three fully-connected layers. Each input image is of size 100 x 50 pixels with just the single colour channel (greyscale). The first convolutional layer in this architecture takes a normalized input and convolves over it with a filter of dimensions 9 x 5 pixels. The second convolutional layer takes the pooled feature map from the first layer and convolves it with a filter of dimensions 9 x 6 pixels. The filter size for the third convolutional layer has dimensions 4 x 2 pixels. As for the max-pooling layers, the window size that was defined has dimensions 2 x 2 pixels. What was interesting about this study was that for the non-linearity operation, both Tanh and the ReLU activation function was used at some point in the model. Finally, appending onto the end of the three convolutional and max-pooling layers were two fully-connected layers which have 120 nodes and 30 nodes, respectively. The task of these FC layers is to classify the input image into one of the 6 pre-determined classes.

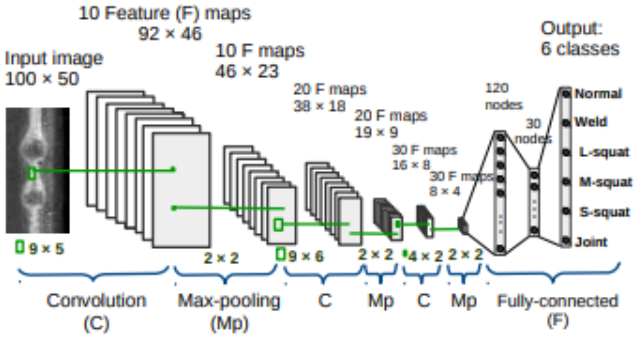


Figure 12 The design for the medium DCNN that S. Faghih-Roohi et al developed [fig. 12]

Similar to the study by Patrice Y. Simard et al that was reviewed earlier on in the section, S. Faghih-Roohi also had a learning rate that implemented a decay factor to help avoid introducing overfitting to the training data. The learning rate that was used in this study was initially set to 10^{-3} with a decay factor of 10^{-5}.

Two types of experiments were conducted on the three network architectures. The first type of experiment was to train the networks to classify the data into the 6 classes and Confusion Matrices have been produced to aid the analyses of the performance of each of these models.
The second type of experiment that was conducted was a retrain of the models using both Tanh and ReLU activation functions to see how the selection of the non-linearity operation affects the performance of each model.

The confusion matrices for the first type of experiment have been included below. The rows of the matrices correspond to the correct classes and the columns correspond to the predicted classes. For example, in Table 4, 48.13% of classified severe squats were correct, while 33.12% of the actual severe squats are classified as being in the medium squat class. Similarly, in Table 5, the percentage of correctly classified welds is 61.95%, while 30.97% of the actual welds are classified in the normal class. These false classification values are relatively high compared to the total number of welds and therefore this comparison shows that through the assessment of the multi-class classification accuracy, it can be suggested that the two classes of normal and welds should be integrated into just one class (Normal). Likewise, it can be suggested that all three classes of light, medium, and severe squat should be combined together as a single defect class (Defect). As a result of doing this there are now three classes (Normal, Defect, and Joint) considered for the performance evaluation of the DCNN models

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	94.62	3.06	1.82	0.19	0.08	0.23
Weld	31.85	59.62	3.20	1.55	0.29	3.49
L-squat	21.09	3.27	66.93	7.42	0.10	1.19
M-squat	5.95	3.81	27.62	53.57	7.62	1.43
S-squat	6.25	5.62	2.50	37.50	44.38	3.75
Joint	3.13	5.75	1.12	1.75	2.12	86.13

Table 3 Confusion Matrix for the small DCNN (%) [tbl 3]

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	95.74	1.94	1.78	0.11	0.04	0.39
Weld	29.61	63.01	3.11	0.88	0.19	3.20
L-squat	22.47	3.17	65.05	7.72	0.30	1.29
M-squat	8.81	2.86	22.86	56.19	6.90	2.38
S-squat	8.12	4.37	3.13	33.12	48.13	3.13
Joint	2.25	5.50	1.62	1.00	1.00	88.63

Table 4 Confusion Matrix for the medium DCNN (%) [tbl 4]

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	96.32	1.82	1.32	0.15	0.08	0.31
Weld	30.97	61.95	2.52	0.68	0.29	3.59
L-squat	22.08	3.17	64.36	9.40	0.10	0.89
M-squat	6.90	3.33	24.76	56.67	7.15	1.19
S-squat	8.12	2.50	3.13	34.37	50.00	1.88
Joint	2.50	4.37	1.00	1.75	2.00	88.38

Table 5 Confusion Matrix for the medium DCNN (%) [tbl 5]

In addition to the reduction of the number of classes to three classes, it was also decided to include a binary classification of normal samples versus anomalies – to cater for this they simply regarded the normal class as one class and all of the non-normal classes and another class and computed the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FP). The binary classification accuracy and F1-Score can then be defined based on the reduced classification matrices. The accuracy is calculated as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

While the F1-Score is calculated as follows:

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where the precision – the fraction of relevant instances among the retrieved instances, is defined by

$$Precision = \frac{TP}{TP + FP}$$

And the recall – the fraction of relevant instances that have been retrieved over the total amount of relevant instances, is defined by

$$Recall = \frac{TP}{TP + FN}$$

The performance results of the here DCNN architectures for the multi-class (3 classes) and binary (normal and anomaly) classifications are summarized in Table 6. The type of activation function that was used is also included allowing for the comparison of the results that were obtained in this study.

Activation Function	DCNN	Binary accuracy	Binary F1-score	Multi-class accuracy	Relative run time
Tanh	Small	0.9216	0.8995	0.9117	1.005
	Medium	0.9250	0.9030	0.9180	1.370
	Large	0.9290	0.9091	0.9195	2.022
ReLU	Small	0.9293	0.9095	0.9221	1.000
	Medium	0.9277	0.9077	0.9203	1.375
	Large	0.9304	0.9107	0.9247	2.027

Table 6 Performance Results when using Tanh and ReLU activation functions [tbl 6]

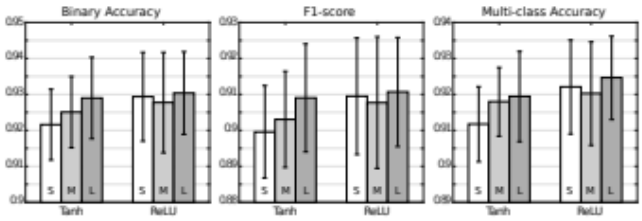


Figure 13 Visualising performance of Tanh and ReLU through histograms [fig. 13]

The histogram plots in figure 12 compares the results published in Table 6 based on the means of the computed performance metrics and the corresponding standard deviations. We can see from these results that, as expected, the ReLU activation function tends to yield better results than tanh across all three network architectures. It is also useful to note that the computational running time of using the ReLU function is not significantly different to using the tanh function. This helps us to conclude that from the results the ReLU function seems to outperform the tanh function and so when it comes to implementing my own CNN, I should keep this in mind.

In addition to this, when comparing the three network architectures, it can be observed that the large DCNN in general outperforms both of the other types of DCNN. However, such an improvement comes at the cost of almost doubling the computation time required as opposed to the small DCNN. It could be concluded from this that there is an obvious trade-off between the network complexity/size and the time required to train it. When it comes to designing my own network, I could control the trade-off by opting for one of the following models:

- A small-size network that will train relatively faster than larger networks but may not have high levels of performance

- A large-size network that will likely achieve higher levels of performance than smaller networks but will take considerably longer to train

- A medium-size network that will likely perform better than a small network but worse than a large network and will train faster than a large network but slower than a small network.

3 Methodology

Every scientific or engineering-based project requires the developer to devise a step-by-step mental walkthrough of how they will go about approaching each step in their methodology. Some of the main focuses here are to understand: what tasks will need to be completed in order to reach the end goal, in which order will they need to be completed, how long each task should take, and which process model they believe would be most beneficial for their project. While devising this methodology, the developer should also remember to consider the aims and objectives of the project, whom are the criteria of success for the project.

The specific tasks required to fulfil the aims and objectives of this project have been identified as follows:

- Gather a relevant image-based dataset
- Research the data pre-processing tasks related to image processing and apply the necessary ones to the gathered dataset
- Design the Neural Network architecture models
- Develop the Neural Network architecture models
- Train and Test the models with the prepared dataset and compare the results of each model.
- Design and Develop the Graphical User Interface
- Consolidate the CNN with the GUI into a single application

A project management consideration that requires attention early-on in the project is to decide the process model that will be implemented. Each task in the list above can be viewed as an individual step in the project, where the first step must be completed before the second, the second step must be completed before the third, and so on. It can be said then, that this list has an incremental nature, where each step adds something different to the project. With that being said, it was decided that the most suitable process model to adopt was the incremental approach, whereby each task will be worked on until full completion before any progress is made on the next task. One other consideration that was made when choosing the process model was the time; or lack thereof, to conduct the project – An iterative approach would require repetition of the steps listed above for constant improvement and adjustments. This can be quite a time-consuming method and it was evident from the start of the project that there just wasn't enough time to seriously consider this as the approach to be used.

4 Experiment Setup

This experiment aims to progress the research carried out by authors of the papers reviewed in section 2. As identified, Patrice and his colleges managed to find a way to reduce error through the means of expanding his data set with affine transformations and elastic distortions. They found that applying affine transformations to their dataset reduced the overall training error, while applying elastic distortions to the dataset resulted in an either further loss (Table 1). I decided that to further this research, they could create a third dataset which contains a number of original images, a number of images that have gone through affine transformations, and a number of images that have gone through elastic distortion. The error results from testing the CNN with this dataset can then be compared to the other two datasets to see whether an even further reduction in error occurs. In order prepare for this experiment, I will need to create 3 separate datasets and expand each one with one/both type/s of transformation.

In addition to testing three separate datasets on CNN architecture, I also decided that I would like to take a similar approach to the work carried out by S. Faghih-Roohi et al in regard to the actual CNN architecture. Finding the balance between the depth of a CNN architecture and size of its hyper-parameters, the network training time, and its classification performance is one of the biggest concerns in Neural Network design. Experimenting with three architectures of progressive sizes will help me find this balance and determine which of the three architectures provide the best overall results. I will also be able to determine if any of the networks suffer from under-/over-fitting.

The three architectures that have been chosen for this study are represented in the following table.

Convolutional Neural Network Architecture				
Layer Type	**Hyper Prameter**	**Small**	**Medium**	**Large**
Convolutional Layer 1	Feature Maps	6	10	32
	Filter Size	5 x 5	5 x 5	3 x 3
	Stride	1	1	1
	Padding	valid	valid	valid
Max-Pooling Layer 1	Window Size	3 x 3	2 x 2	2 x 2
Convolutional Layer 2	Feature Maps	10	20	64
	Filter Size	5 x 5	5 x 5	3 x 3
	Stride	1	1	1
	Padding	valid	valid	valid
Max-Pooling Layer 2	Window Size	3 x 3	2 x 2	3 x 3
Fully Connected Layer 1	No. of neurons	40	40	70
Fully Connected Layer 2	No. of neurons	-	20	40
Fully Connected Layer 3	No. of neurons	-	-	15
Model Parameters		137,111	1,470,245	3,879,701

Table 7 Model Architectures [tbl. 7]

The three CNN architectures listed in the table above, along with the three separate training datasets results in a total of 9 tests to be conducted. The results from these tests should allow me to compare the different size architectures to determine which one performs the best across

these datasets and what the difference in terms of training time cost is. Moreover, I can compare whether; like the study from Patrice et al, the elastic distortions yield a lower error than affine transformations, and also whether the third dataset is able to yield even lower error than the elastic distortions dataset.

5 Data Pre-Processing

5.1 Gathering the Dataset

The original raw image dataset for this project has been provided by my supervisor whom works in the Research Institute for Future Transport and Cities at Coventry University. The dataset contains a total of 997 images, each with dimensions width: 6432, height: 2048, and colour channels: 1 (greyscale). Each image in the dataset depicts an aerial view of a section of the railway track. An example image from the dataset can be viewed below.

Figure 14 One of the 997 images in the gathered dataset [fig. 14]

5.2 Dataset Expansion through slicing

It was understood that the volume of the original dataset is not sufficient for the training and testing of a Machine Learning algorithm, and so it was decided that expansion of the dataset would need to take place. One approach taken to perform such expansions is to slice the images into a number of sub-images that each depict one specific sleeper. It was decided that 6, 7, and 8 equal sized slices were to be chosen and experimented on, some of these slice examples can be seen in the following images. It is important to note that since the width of the original image (6432) is not perfectly divisible by 7, some of the sub-images created from the slice size of 7 contain a degree of overlap.

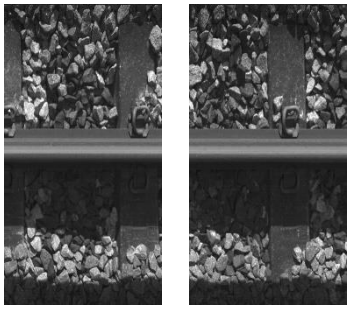
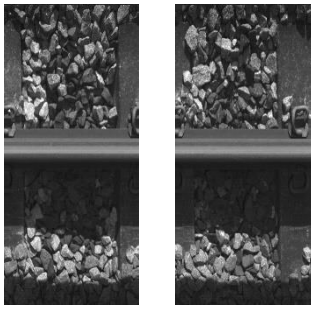


Figure 15-16 two example slices as a result of splitting an image into 6 separate slices [fig. 15-16]

Figure 17-18 two example slices as a result of splitting an image into 7 separate slices [fig. 17-18]



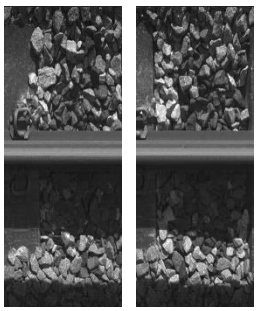


Figure 19-20 two example slices as a result of splitting an image into 8 separate slices [fig. 19-20]

An experiment was conducted to see which quantity of slices (6, 7, or 8) resulted in the highest number of resulting sub-images that perfectly depicted only one whole sleeper. For example, figure 15 contains two sleepers in one image and so would not be suitable for our dataset as only part of each sleeper is present in the image, meaning the network classifier may not be able to correctly extract the important features from this image.

Figure 19 contains only one sleeper but again the whole sleeper is not depicted in the sub-image and so this could cause similar issues when training a network.

Figure 16, on the other hand, is a perfect example of the type of sub-image we are looking to create from slicing the images. This sub-image depicts one and only sleeper from the railway track and the sleeper is allocated inside the slice frame.

20 images were split into 6 equal slices for a total of 120 sub-images. Of these 120 images, it was found that 85 of them perfectly matched the criteria of the training set. This meant that across these 20 sample images, 70.83% of the sub-images were useful for our study.

The same 20 images were then split into 7 equal slices for a total of 140 sub-images. Of these 140 sub-images, it was found that 70 of these images perfectly matched the criteria. This resulted in an estimation of exactly 50% of the training data being useful for training and 50% being discarded – a reduction of roughly 20% on the previous test.

Lastly, the 20 images were split into 8 equals slices for a total of 160 sub-images. Of these 160 images, it was found that only 78 of them perfectly depicted a single sleeper. This meant that when split this way, only 48.75% of the sub-images were useful for our study – an either further reduction from the previous tests.

Concluding this experiment, it can be reported that in this case the best number of slices to split the dataset into is 6, with approximately just over 70% of the resulting images being useful for training. 7 equal slices reduced this figure down to 50%, and 8 equal slices further reduced this to just over 48%.

Applying this cropping function to each individual image from the dataset result in a new dataset containing 5,982 (997*6) images.

5.3 Dataset Expansion through slicing – Code

Below is the Python code that was written to automate the process of slicing each image.

```
def importImages():

    X_data = []
    for i in range(1,21):
        image = cv2.imread(r'A:/Documents/Coventry University/MSc/MSc - Individual Project/raw images/RailwayImage (' +
        X_data.append (image)

    npImageArray = np.array(X_data)

    print('X_data shape:', np.array(X_data).shape)

    return npImageArray


imageArray = importImages()
```

Select the first 20 images from the image dataset and convert to an array of pixel values.

```
def cropImage(images, cropSize):
    fileNumber = 1
    boundary = int(6432/cropSize)

    for j in images:
        for k in range(0,cropSize):
            imageSlice = j[:, (k*boundary):((k+1)*boundary), :]
            writeToFile = Image.fromarray(imageSlice)
            writeToFile.save(r"A:/Documents/Coventry University/MSc/MSc - Individual Project/CroppedImages/"+str(cropSi
            fileNumber = fileNumber + 1


# Slice images into 6, 7, 8 equal slices
cropImage(imageArray, 6)
cropImage(imageArray, 7)
cropImage(imageArray, 8)
```

Divide the width of the image by either 6, 7, or 8, and split the image vertically by that amount to create either 6, 7, or 8 new slices. Save these slices to a folder

Figure 21 Dataset Expansion through Slicing [fig. 21]

5.4 Labelling the Dataset

Once the dataset had been expanded, it needed to be manually labelled into one of 5 possible classes. While doing this, I also needed to discard images that did not meet the specification described above. From the experiment conducted above, we could expect to discard roughly 30% of the data, however this was only an experiment on 20 out of 997 images and so may not be a true representation of the proportion of images that can expect to be discarded.

The five class labels for this study are:

- Class 1 – No Maintenance Required
- Class 2 – Missing Fastener
- Class 3 – Moderate Ballast
- Class 4 – Excess Ballast
- Class 5 – Deletion Class

Class 1 represents those images that depict once sleeper in the track whose fasteners are existent and intact (i.e. no maintenance is needed). It also contains the images that show only a very small amount of ballast (the layer of rocks that the sleepers lay on) which does not cover the fasteners and prevents us from identifying the existence and condition of the fasteners. Example images from this class are as follows.

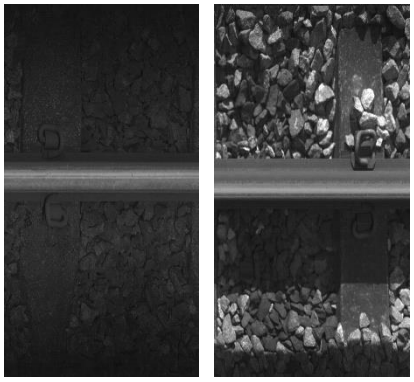


Figure 22-23 Class 1 example images [fig. 22-23]

Class 2 represents those images that depicted one sleeper in the track where either one or both fasteners for that sleeper are damaged or completely missing. This are the sections of the track that require maintenance to re-secure the track, making it safer for all that use it. Example images from this class are as follows.

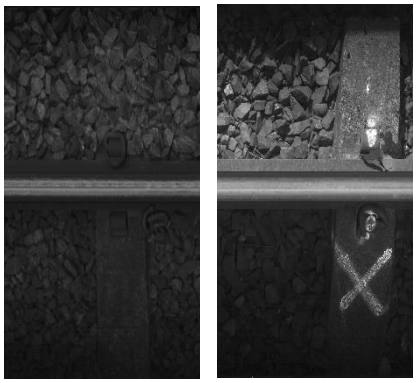


Figure 24-25 Class 2 example images [fig. 24-25]

Class 3 contains the images that depict one sleeper but there is a ‘moderate’ amount of ballast on and/or around either or both of the fasteners, preventing us from being able to detect whether that fastener is intact or whether it requires maintenance – it may be easy to identify the fasteners in these images, but it isn’t possible to determine if they are at a safe, working condition. Example images from this class are as follows.

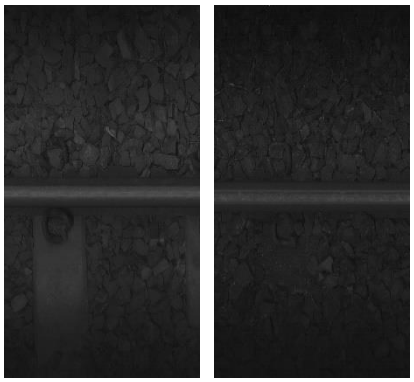


Figure 26-27 Class 3 example images [fig. 26-27]

Class 4 contains the images that depict one sleeper where either one or both fasteners are completely covered in ballast, preventing us from being able to identify the existence of the fastener.

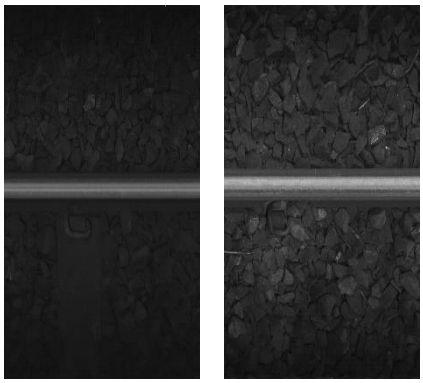


Figure 28-29 Class 4 example images [fig. 28-29]

Finally, **Class 5** contains the selection of images from the slicing process that did not meet the criteria – i.e. those that either contained more than one sleeper in the image, or those that only display a portion of one sleeper in the image. This class is required as new images that are classified by the CNN will need to be sliced beforehand and the unusable slices of the image will need to be discarded – these images will be classified into this deletion class.

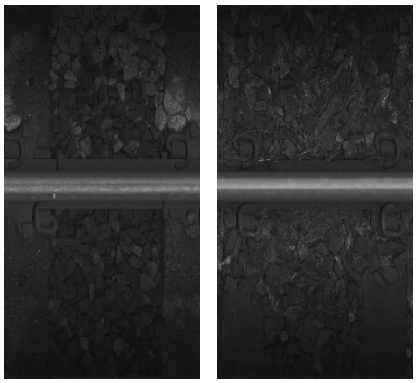


Figure 30-31 Class 5 example images [fig. 30-31]

These five classes have been chosen for one primary reason. Moderate and large amounts of excess ballast in the images can prevent engineers from being able to determine whether the fasteners are there and whether they require maintenance, making it somewhat difficult to accurately categorize these images into one of the first two classes - *definitely* requiring maintenance or *definitely* not requiring maintenance. Due to this issue of excess ballast, it was decided that two new classes will be created which will contain images the track that cannot be confidently and accurately placed into one of the first two classes. Those images that contain a moderate level of ballast is classified into Class 3, and those images that contain a substantial amount of excess ballast are classified into Class 4.

It would then be up to railway engineers to review the images in classes 3 and 4 and decide whether they would like to take any necessary actions in regard to removing this excess ballast from the track. Doing this will help reveal the presence/absence of the fasteners, along with their condition. The last class is the deletion class which will store all of the slices of each image that do not meet the training/testing image criteria.

5.5 Class Imbalance Problem

After each image had been manually labelled, it was evident from the class size distributions that there was a 'class imbalance' problem (where the majority class; in this case the first class, contains an overwhelming number of instances compared to the other classes). There were various methods that could be adopted to help combat this problem, such as random oversampling, and random under-sampling, and as described in the literature review and the experiment setup, the experiment to be conducted would include three datasets (one with just affine transformations, one with just elastic distortions, and one with both affine and elastic), and so intuitively we could control the class imbalance problem by adjusting the number of new instances to create through applying transformation. For example, much fewer instances would be created for the majority class (Class 1) than would be created for either of the other minority classes.

5.5.1 Affine Transformation

The affine transformations that were included in the synthesis of new instances included: Rotations, Scaling, Shearing, and Translations. Suitable parameter values (such as the maximum quantity to scale an image, the maximum quantity to translate an image in either axis by, etc.) were tested on a number of sample images from the dataset – this was to ensure that important features like the sleepers and fasteners were not accidentally removed from the image or over-transformed up to the point where they are indistinguishable.

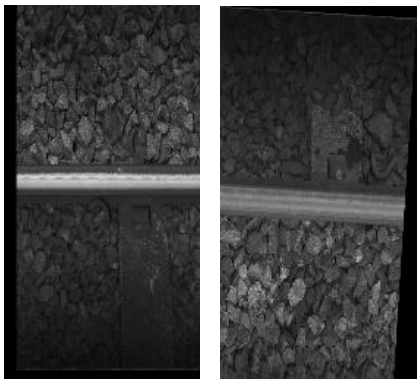


Figure 32-33 Examples of images that have underwent Affine Transformation [fig. 32-33]

5.5.2 Affine Transformation - Code

```
def randomTransformationChoice(image):
    choice = random.randrange(0, 4)
    if choice == 1:
        transformedImage = ApplyRotation(image)
    elif choice == 2:
        transformedImage = ApplyScaling(image)
    elif choice == 3:
        transformedImage = ApplyTranslation(image)
    else:
        transformedImage = ApplyShearing(image)

    return (transformedImage)
```

Randomly choose one of the types of Affine Transformation to apply to the image.

Figure 34 Randomly choosing an Affine Transformation type [fig. 34]

Rotation

```
# RANDOM ROTATION BETWEEN -0.1 AND 0.1
def ApplyRotation(newImg):
    choice = random.uniform(-0.1, 0.1)
    tform = AffineTransform(scale=(1, 1), rotation=choice, shear=0, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Rotation amount has lower and upper bounds of -0.1 and 0.1, respectively.

Scaling

```
# RANDOM SCALING BETWEEN 0.92 AND 1.02
def ApplyScaling(newImg):
    choice = random.uniform(0.92, 1.02)
    tform = AffineTransform(scale=(choice, choice), rotation=0, shear=0, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Scaling coefficient has lower and upper bounds of 0.92 and 1.02, respectively.

Translation

```
# RANDOM TRANSLATION BETWEEN -100 AND 100 (BOTH AXIS)
def ApplyTranslation(newImg):
    xchoice, ychoice = random.randrange(-100, 100), random.randrange(-100, 100)
    tform = AffineTransform(scale=(1, 1), rotation=0, shear=0, translation=(xchoice, ychoice))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Translation (by pixel amount) has lower and upper bounds of -100 and 100, respectively.

Shearing

```
# RANDOM SHEARING BETWEEN -0.1 AND 0.1
def ApplyShearing(newImg):
    choice = random.uniform(-0.1, 0.1)
    tform = AffineTransform(scale=(1, 1), rotation=0, shear=choice, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Shearing coefficient has lower and upper bounds of -0.1 and 0.1, respectively.

Figure 35 Affine Transformation code [fig. 35]

5.5.3 Elastic Distortion

As for the elastic distortions, the exact same method and Python code was used as the one described in the literature review. In similar fashion to the affine transformations above, the upper and lower bounds of the elastic distortion parameters (sigma – the elasticity coefficient, and alpha – the scaling factor that controls the intensity of the deformation) were tested on a number of sample images and the best performing boundary values were selected to be used for the whole dataset.

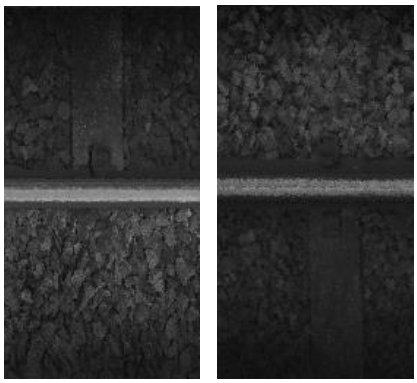


Figure 36-37 Examples of images that have underwent Elastic Distortion [fig. 36-37]

5.5.4 Elastic Distortion – Code

```
def elastic_transform(image, alpha, sigma, random_state=None):

    """Elastic deformation of images as described in [Simard2003]_.
    .. [Simard2003] Simard, Steinkraus and Platt, "Best Practices for
       Convolutional Neural Networks applied to Visual Document Analysis", in
       Proc. of the International Conference on Document Analysis and
       Recognition, 2003.
    """
    if random_state is None:
        random_state = np.random.RandomState(None)

    # Random_state.rand - creates an array of the given shape and populates with random samples from a uniform dist. ov
    # Sigma - Elasticity coefficient
    # Alpha - Scaling factor to control intesity of deformation

    shape = image.shape
    dx = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma, mode="constant", cval=0) * alpha
    dy = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma, mode="constant", cval=0) * alpha

    # Return co-ordinate matrices from co-ordinate vectors
    x, y = np.meshgrid(np.arange(shape[1]), np.arange(shape[0]))
    indices = np.reshape(y+dy, (-1, 1)), np.reshape(x+dx, (-1, 1))

    distored_image = map_coordinates(image, indices, order=1, mode='reflect')
    return distored_image.reshape(image.shape)
```

Figure 38 Elastic Distortion code [fig. 38]

Upon completion of further dataset expansion through synthetic generation of instances with affine transformations and elastic distortions, the resulting dataset consisted of a total of 10,000 instances (2,000 in each class).

5.6 Dimension Reduction

Each of these 2,000 images were then reduced in dimensions from 2048 x 1072 down to 256 x 134 in order to dramatically decrease the disk size of the training data. By doing this, the whole dataset could be uploaded to Google Drive within an acceptable time frame and linked to 'Google Colab' without running into any insufficient RAM issues. Moreover, this would allow each of the networks to train on a much larger batch size as each batch will contain a higher quantity of lower dimension instances as opposed to a lower quantity of higher dimension instances.

5.7 Dimension Reduction – Code

```
from matplotlib import pyplot as plt
from PIL import Image # used for loading images
import numpy as np
import random
import cv2


from skimage.feature import corner_harris, corner_subpix, corner_peaks
from skimage.transform import warp, AffineTransform
from scipy.ndimage.interpolation import map_coordinates
from scipy.ndimage.filters import gaussian_filter
```

```
def Resize(img):
    height = 256
    width = 134
    dim = (width, height)

    resized = cv2.resize(img, dim, interpolation=cv2.INTER_LINEAR)
    return resized
```

```
i = 1
while i <= 1573:
    img  = np.array(Image.open(r'A:/Documents/Coventry University/MSc/MSc - Individual Project/Classes/Transformations/AffineAnd
    newImg = Resize(img)
    imageToWrite = Image.fromarray(newImg)
    writeToFile = imageToWrite
    writeIcon = i + 427
    writeToFile.save(r"A:/Documents/Coventry University/MSc/MSc - Individual Project/Classes/AffineAndElastic/Class 4/Class4Imag
    i=i+1
```

Figure 39 Dimension Reduction code [fig. 39]

6 Solution Design

This section is concerned with the choices made in relation to the design of the architectures that will be developed as part of the project requirement. Inspirations taken from the literature review have helped inform some of the choices made in the data preparation stage and have also aided the set-up of the experiment by providing example work on how different architecture configurations can be designed, developed, and compared. As discussed, this project will compare the performance of three varying size Convolutional Neural Network architectures whom will be trained on 3 separate datasets, to produce a total of 9 results. These can be analysed using various performance metrics and graphs. As part of the Solution Design, it was important that diagrams of each of the chosen architectures were created to help the readers and myself visualise the details of each network.

6.1 Small Architecture Design

The smallest CNN model will be constructed with **two Convolutional Layers**, **two Max-Pooling layers**, and **one Fully-Connected hidden layer**. C1 (as depicted below) will have 6 filters each with dimensions 5 x 5 and will produce 6 feature maps, each having dimensions 130 x 252. These feature maps will undergo the first iteration of down sampling by being fed into a Max-Pooling layer (MP1) whose Pooling Windows have dimensions 3 x 3, reducing each Feature Map to have dimensions 43 x 84. C2 will have similar Filter dimensions to C1, however it will have 10 Filters; and therefore 10 Feature Maps, as opposed to just 6 in C1. These will each have dimensions 39 x 80. MP2 will once again down sample this data so that each feature map has dimensions 13 x 26. The 10 Feature Maps each with dimensions 13 x 26 can be flattened into a single vector of 3380 nodes (13 * 26 * 10). This vector will then be treated as the input layer into the two Fully-Connected layers where classification through the SoftMax function will allow the network to make its decision.

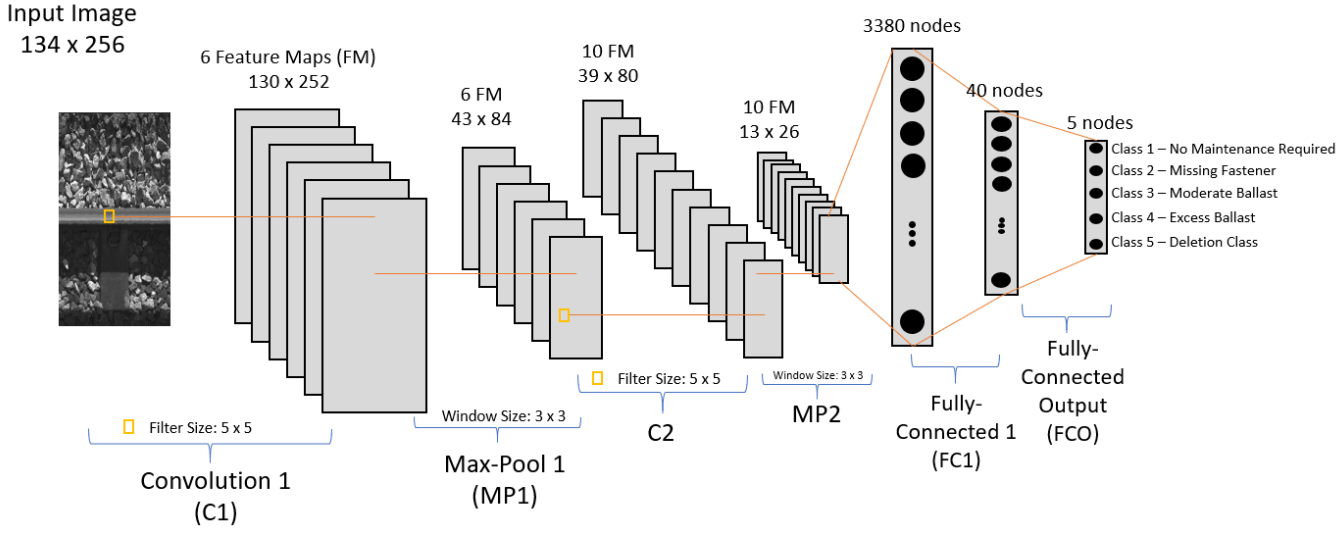


Figure 40 Design of the Small CNN Architecture [fig. 40]

6.2 Medium Architecture Design

The medium-sized CNN model will be constructed with **two Convolutional Layers**, **two Max-Pooling layers**, and **two Fully-Connected hidden layers**. C1 (as depicted below) will have 10 filters each with dimensions 5 x 5 and will produce 10 feature maps, each having dimensions 130 x 252. These feature maps will undergo the first iteration of down sampling by being fed into a Max-Pooling layer (MP1) whose Pooling Windows have dimensions 2 x 2, reducing each Feature Map to have dimensions 65 x 126. As we have used smaller Pooling Window dimensions than the previously described architecture, we can expect the resulting Feature Maps from MP1 to be larger (as more information is retained). C2 will have similar Filter dimensions to C1, however it will have 20 Filters; and therefore 20 Feature Maps, as opposed to 10 in C1. These will each have dimensions 61 x 122. MP2 will once again down sample this data so that each feature map has dimensions 30 x 61. The 20 Feature Maps each with dimensions 30 x 61 can be flattened into a single vector of 36600 nodes (30 * 61 * 20). This vector will then be treated as the input layer into the two Fully-Connected layers where classification through the SoftMax function will allow the network to make its decision.

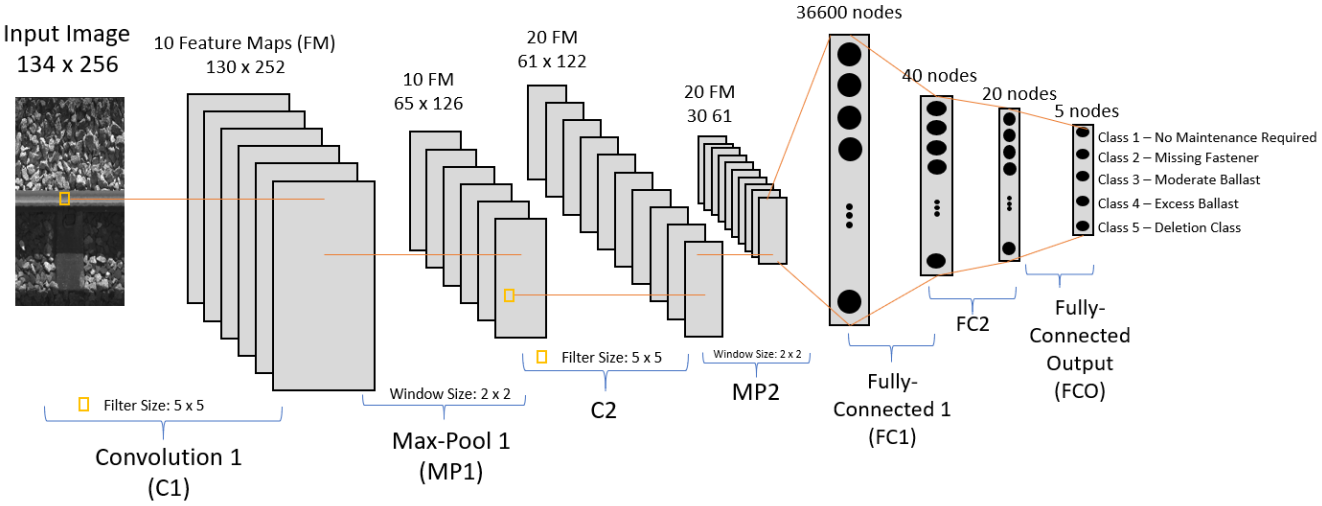


Figure 41 Design of the Medium CNN Architecture [fig. 41]

6.3 Large Architecture Design

The largest CNN model will be constructed with **two Convolutional Layers**, **two Max-Pooling layers**, and **three Fully-Connected hidden layers**. C1 (as depicted below) will have 32 filters each with dimensions 3 x 3 and will produce 32 feature maps, each having dimensions 132 x 254. These feature maps will undergo the first iteration of down sampling by being fed into a Max-Pooling layer (MP1) whose Pooling Windows have dimensions 2 x 2, reducing each Feature Map to have dimensions 66 x 127. C2 will have similar Filter dimensions to C1, however it will have 64 Filters; and therefore 64 Feature Maps, as opposed to 32 in C1. These will each have dimensions 64 x 125. MP2 will once again down sample this data so that each feature map has dimensions 21 x 41. The 64 Feature Maps each with dimensions 21 x 41 can be flattened into a single vector of 55104 nodes (21 * 41 * 64). This vector will then be treated as the input layer into the two Fully-Connected layers where classification through the SoftMax function will allow the network to make its decision.

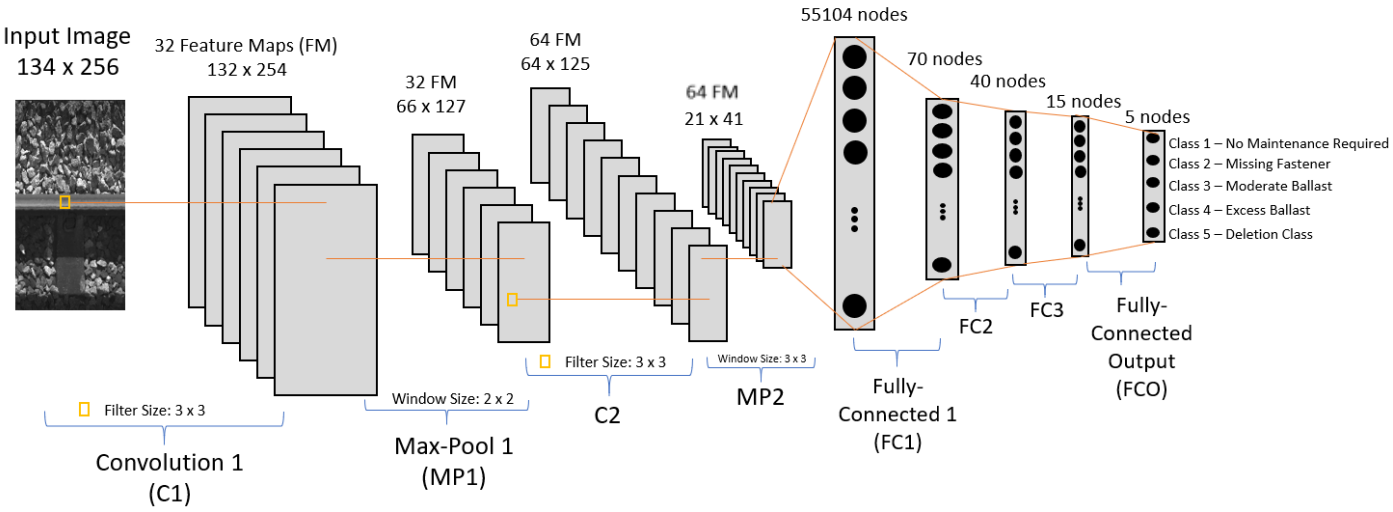


Figure 42 Design of the Large CNN Architecture [fig. 42]

7 Implementation

Working in an incremental manner meant that once it came to this section where actual implementation of the models and conduction of the experiment can commence, all previous work including data gathering, data pre-processing, the literature review, the experiment setup, and the solution designs, were complete – the expectation was that no more work relating to these areas of the project will be carried out beyond this point.

Considering the previous knowledge and experience of the Python programming language that had been acquired during my studies at university, the selection of the programming language to implement the models in was rather intuitive. Moreover, choosing Python meant that I could become more exposed to a selection of Data Science-based libraries and packages whom provide useful, optimised, and very well-maintained resources and functions for common Data Science tasks – some of which may become very helpful in a job role, post-university. The specific extensions that were used are:

- **NumPy** (Numerical Python) – a large collection of high-level mathematical functions to operate on matrices and arrays.
- **OpenCV** (Open Source Computer Vision) – an image processing library with a useful set of functions that can be applied to images.
- **Keras** – an open-source neural network library (explained in more detail below).
- **Matplotlib** – an object-oriented API for embedding plots (graphs and tables) in Python, with support for the NumPy extension.
- **Seaborn** – a data visualization library that provides a high-level interface for drawing informative statistical graphics.
- **Scikit-learn** – a machine learning library that provides data pre-processing operations, and summary statistics on model performance.

Choosing the environment in which the implementations would take place was also a straightforward choice. Initially it became a decision between the standard Python IDLE and a more sophisticated ‘playground’ named ‘Anaconda Jupyter Notebook’ – this program allows for the enclosure of fully customizable text along with live python code into a single notebook. My experiences with Jupyter Notebook in the past have been mostly positive and so my mind was set on using the latter environment. However, when this was discussed in a casual conversation with a fellow student, he suggested that I look at ‘Google Colab’ - a cloud-based environment that was similar in nature to Jupyter Notebook (it allowed the mixture of text and code in a single document), but also allowed for free GPU hardware acceleration. This essentially enabled me to use a much higher-level of computing power than the standard desktop PC that I was currently using to execute my program code. By utilising GPU acceleration, the execution time required to load and process the data, and then train and test the models was greatly reduced.

As for the models themselves, all three have been programmed using one of the most popular frameworks for Neural Network implementation – Keras. Keras is a high-level Neural Networks API that is written in Python and allows for fast experimentation conduction. The documentation for this API states that it is ‘*being able to go from idea to result with the least possible delay [which] is key to doing good research*’ (Keras Documentation Website, n.d.) and this API aims to achieve this by introducing development tools that make the prototyping and development of Neural Network models fast and easy.

The models were initially trained on 80% of the prepared data and tested on the remaining 20%. The results; including model train and test accuracy, and model loss, were captured in an Excel table, and the Confusion Matrix and accuracy/loss graphs were screenshotted and added in to

the same Excel document. These results were evaluated and compared against each other and the common issue of ‘overfitting’ became apparent during this evaluation stage. To mitigate this issue, **the experiment was repeated** with the introduction of a validation dataset and the ‘early stopping’ tool.

Validation Dataset – a sample of data from the training set that is used to give an estimate of model performance while tuning its hyper-parameters (the weights and biases)

Early Stopping – This tool monitors the progression of model training and constantly saves the ‘best version’ of the model to an external file. This file is updated on each epoch that improves the model’s accuracy/loss on the testing data. Model overfitting is also prevented as this tool can choose to revert to the last epoch before the model began to overfit to the training data.

Once again, all the statistics from this experimentation were recorded into the Excel document and evaluated. The ‘Experiment Results’ section discusses the findings of the experiment.

A Graphical User Interface to act as a gateway between the user and the now-trained optimal model was developed in the Java programming language. My experience with this programming language at this moment in time was rather limited and so at first, I found it difficult to use but remained persistent and resourceful and eventually developed a solid basic understanding of the language. The ‘NetBeans’ program was used to design and develop each interface in the application and code was written to allow it to execute various Python scripts that performed the necessary data pre-processing task on the image that was uploaded by the user. The prepared image was then evaluated by the model; which is saved externally, and the results were obtained by the Java application. These results were then communicated back to the user through a suitable interface.

8 Experiment Results

This experiment aimed to answer two fundamental questions that were derived through the research of previous academic work and supporting literature.

First and foremost, I wanted to know which architecture (small, medium, or large) generally yields the best-performing model for this particular application. There is a trade-off between the selection of architectures as the small architecture will tend to train much quicker than the other two, but its feature extraction capabilities may not be as powerful or as accurate, likewise, the large architecture may be able to extract greater levels of features but trains much slower and comes with the risk of overfitting to the training data.

In addition to testing the specific network architectures, I also wanted to follow up the research into dataset expansion techniques by querying whether a mixed dataset (containing instances that have undergone Affine Transformation as well as instances that have undergone Elastic Distortion) was able to outperform datasets that have **only** undergone Affine Transformation or **only** undergone Elastic Distortions.

After implementation and training of each of the models, I recorded the obtained training accuracy and loss as well as the testing accuracy and loss. These statistics were all consolidated into a single Excel table that can be viewed below.

Dataset	Architecture	Train Accuracy	Train Loss	Test Accuracy	Test Loss	Computation time/epoch
Affine	Small	66.62%	0.6921	73.30%	0.6921	7 seconds
	Medium	82.58%	0.4015	80.30%	0.6219	12 seconds
	Large	91.81%	0.2333	82.10%	0.848	19 seconds
Elastic	Small	80.44%	0.5166	84.20%	0.4663	7 seconds
	Medium	95.98%	0.1084	93.10%	0.3867	12 seconds
	Large	93.58%	0.1469	92.90%	0.3762	19 seconds
Mixed	Small	78.87%	0.5103	78.30%	0.5293	7 seconds
	Medium	92.66%	0.1745	84.70%	0.5938	12 seconds
	Large	93.99%	0.1516	83.70%	0.8614	19 seconds

Table 8 Experiment 1 Results [tbl. 8]

The evaluation and comparison of these results have unveiled some very interesting information. Firstly, the average test accuracy for the affine dataset is 78.56%, for the elastic dataset it is 90.06%, and for the mixed dataset it is 82.23%. A similar trend occurs for the loss values obtained for the test data. This information tells us that a) the elastic dataset; just like in the work by Patrice et al, manages to outperform the affine dataset, and b) the mixed dataset **does not** further improve the model accuracy/loss and yield greater results than the elastic dataset.

We can also see that in the case of the large architectures for the Affine and Mixed dataset, the training loss is significantly lower than the testing loss. This indicates that the model has continued to be successful in extracting features from the training set but has not been able to generalise these feature extraction capabilities to the test set – therefore the model is said to have 'overfit' to the training data. The introduction of a validation test set (as discussed in the 'Implementation' section) can help to prevent overfitting from occurring.

In all cases, the Small architecture only managed to achieve sub-par results compared to the two larger models but had a much shorter training epoch time of only 7 seconds. Depending on how you view this trade-off, you may choose to opt for a slighter poorer-performing model, but one that trains much quicker. I decided that in this case, the Small architectures just simply were not powerful enough to perform the classification task effectively.

Moreover, for the elastic and mixed datasets, the medium and large network architectures have been able to yield very similar accuracy results. This tell us that the simpler model (the medium model) performs just as well; and in these cases, even better, than the larger-sized architectures – while training at 7 seconds quicker per epoch! With that being said, I believe the optimal model for this first iteration of experiments is the one with Medium-sized architecture that was trained on the Elastic dataset.

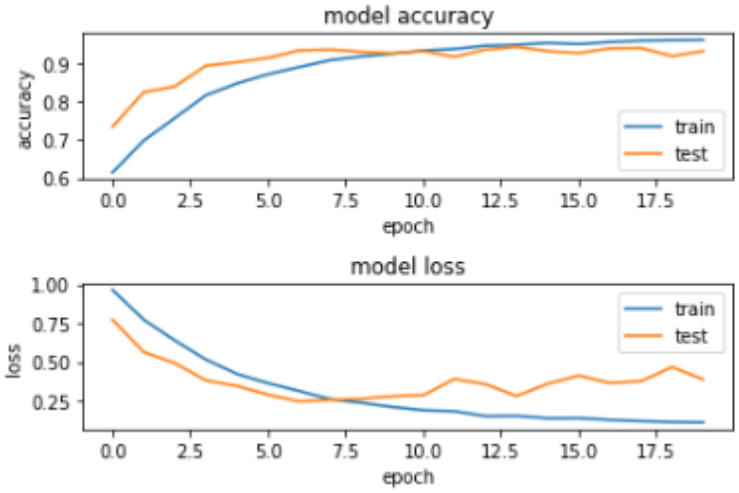


Figure 43 Model Accuracy/Loss graph for Elastic/Medium architecture [fig. 43]

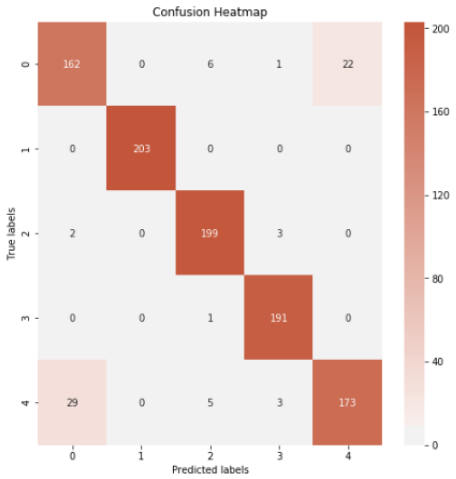
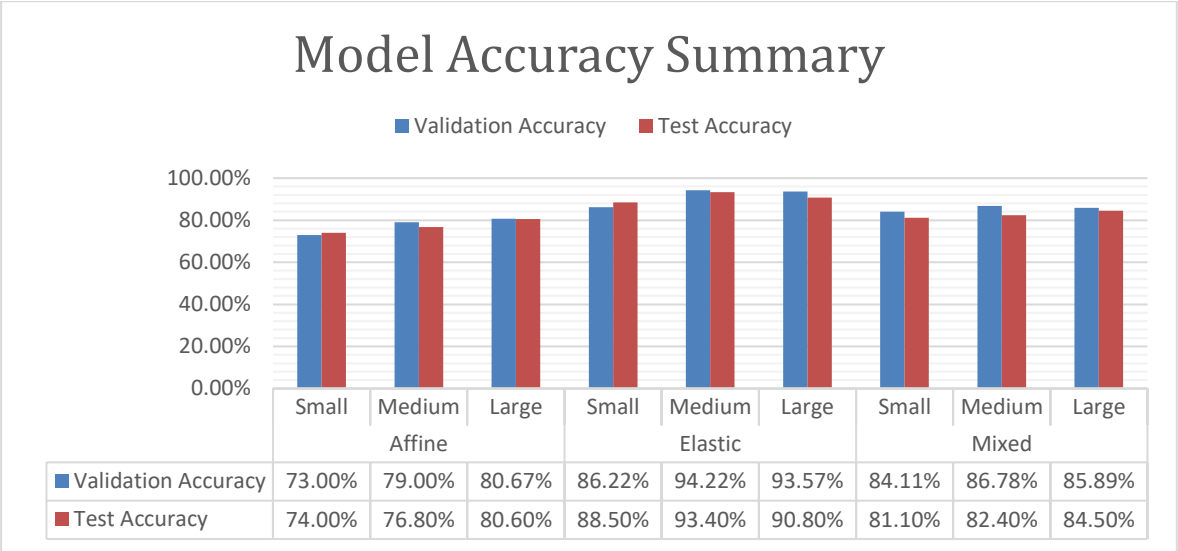


Figure 44 Confusion Matrix for Elastic/Medium architecture [fig. 44]

The second iteration of experiments incorporated a validation test set and 'early stopping' techniques. The statistical results table and corresponding graphs can be seen below.

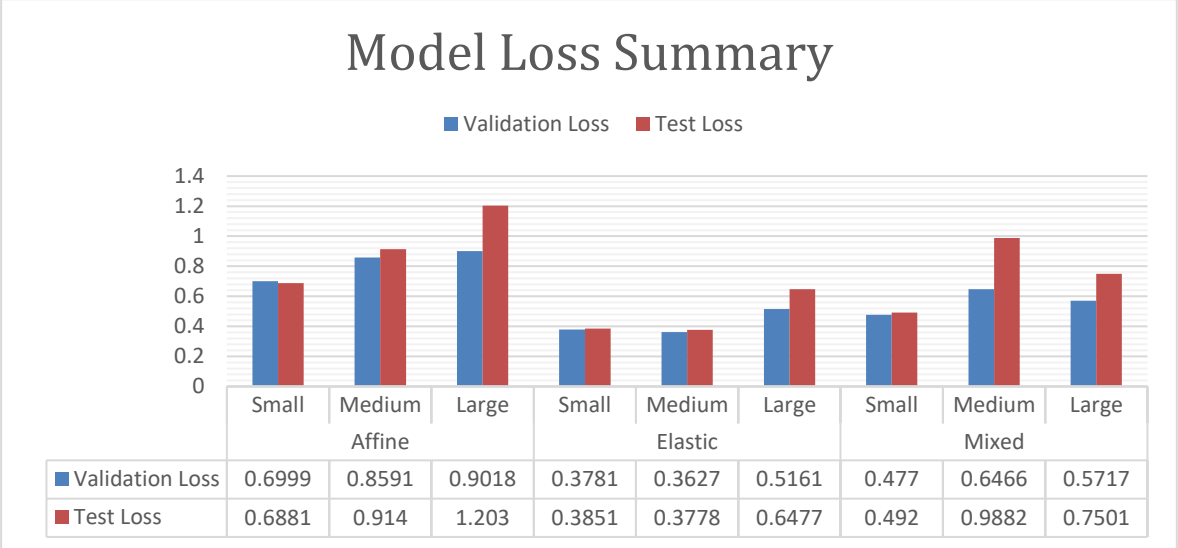
Dataset	Architecture	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	Computation time/epoch
Affine	Small	73.00%	0.6999	74.00%	0.6881	7 seconds
	Medium	79.00%	0.8591	76.80%	0.914	12 seconds
	Large	80.67%	0.9018	80.60%	1.203	19 seconds
Elastic	Small	86.22%	0.3781	88.50%	0.3851	7 seconds
	Medium	94.22%	0.3627	93.40%	0.3778	12 seconds
	Large	93.57%	0.5161	90.80%	0.6477	19 seconds
Mixed	Small	84.11%	0.477	81.10%	0.492	7 seconds
	Medium	86.78%	0.6466	82.40%	0.9882	12 seconds
	Large	85.89%	0.5717	84.50%	0.7501	19 seconds

Table 9 Experiment 2 Results [tbl. 9]



	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
	Affine			Elastic			Mixed		
Validation Accuracy	73.00%	79.00%	80.67%	86.22%	94.22%	93.57%	84.11%	86.78%	85.89%
Test Accuracy	74.00%	76.80%	80.60%	88.50%	93.40%	90.80%	81.10%	82.40%	84.50%

Figure 45 Model Accuracy Summary Graph for Experiment 2 [fig. 45]



	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
	Affine			Elastic			Mixed		
Validation Loss	0.6999	0.8591	0.9018	0.3781	0.3627	0.5161	0.477	0.6466	0.5717
Test Loss	0.6881	0.914	1.203	0.3851	0.3778	0.6477	0.492	0.9882	0.7501

Figure 46 Model Loss Summary Graph for Experiment 2 [fig. 46]

Looking at the model loss summary graph, it is evident that the Large architectures for all three of the datasets still resulted in a model that overfit the training data rather poorly – even when a validation dataset was introduced to mitigate this issue. It is also clear that overfitting has occurred in some of the Medium architectures, however the 'early stopping' technique that has been employed will revert to the latest epoch before overfitting began to occur.

The model accuracy summary graph also helps to visualise and compare the performance of the models for each of the datasets. It can easily be derived from this graph that the Elastic dataset is the best performing, closely followed by the Mixed dataset, and then finally the Affine dataset.

Once again, the best performing model from this second round of experiments is the Medium architecture that was trained on the Elastic dataset. This achieved a test accuracy of 94.40% and model loss value of 0.3778. The details of this model were saved externally so that it can later be incorporated into a Java application.

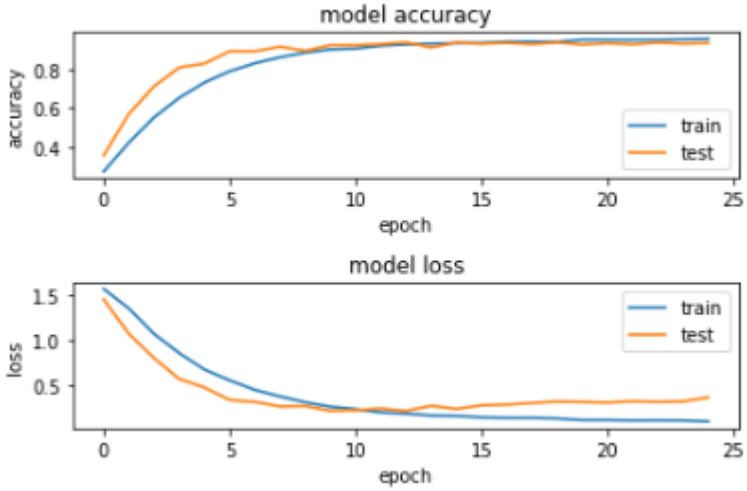


Figure 47 Model Accuracy/Loss graph for Elastic/Medium architecture [fig. 47]

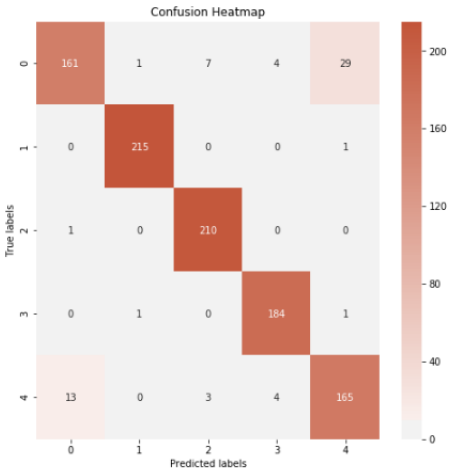


Figure 48 Confusion Matrix for Elastic/Medium architecture [fig. 48]

The Confusion Matrix displayed above is the one obtained for the best performing model. This provides us with some raw statistics on how well the model has been able to classify the data. The 'True Labels' represent the actual class label for each instance in the testing data and the 'Predicted Labels' represent the class that the model believes the instance belongs in. Out of the 1,000 test images, 935 of these images were classified correctly, with the remaining 65 being incorrectly classified. We can see that 42 of these misclassifications occur when the model believes the image is in Class 5 - Unclassifiable however it is actually in Class 1 – No Maintenance Required, or vice versa.

All of the code used to develop and train the models, the Model accuracy/loss graphs, and the Confusion Matrices for every individual test can be viewed in the appendix of this report.

9 Graphical User Interface Development

A Java application has been developed to act as the gateway between the end user and the optimal Neural Network Model that will classify their images. This application was developed in NetBeans and contains a number of attractive interfaces that guide the user through the process of automatically examining maintenance issues in railway track images.

When the user first loads the Java application, they will be greeted with the following menu screen.

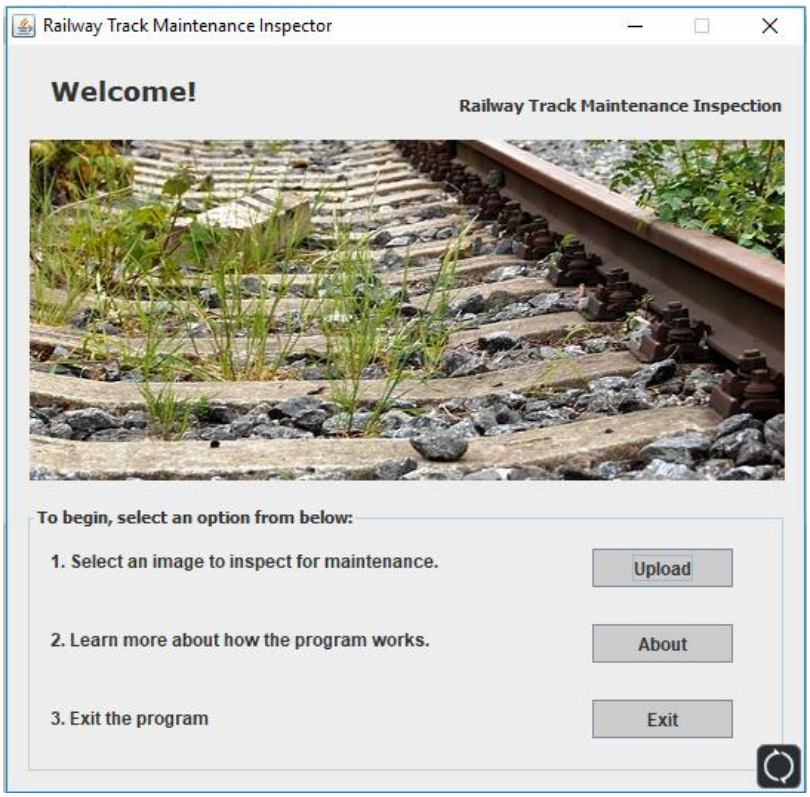


Figure 49 Java Application Menu Screen [fig. 49]

Here they have three options to choose from. They could either select an image that they would like to inspect by clicking on the ‘Upload’ button, or they could learn more about the application and the incorporated Neural Network architecture by clicking on the ‘About’ button, or they could exit the program by clicking on the ‘Exit’ button.

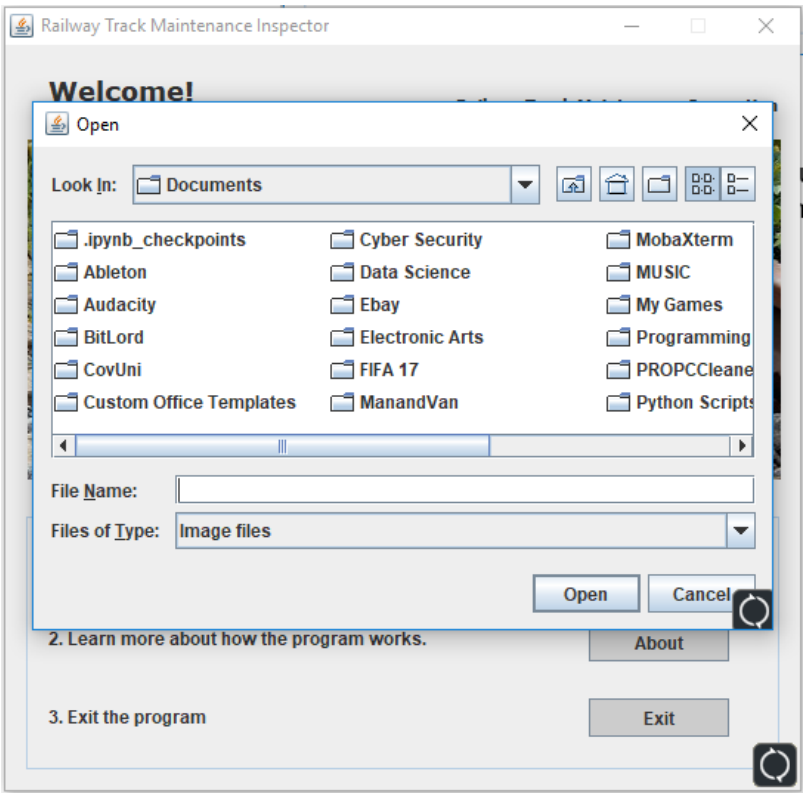


Figure 50 Java Application Image Upload Screen [fig. 50]

In the case where the 'Upload' button is clicked, the user will be prompted to select an image. The application has been designed so that only image files may be opened.

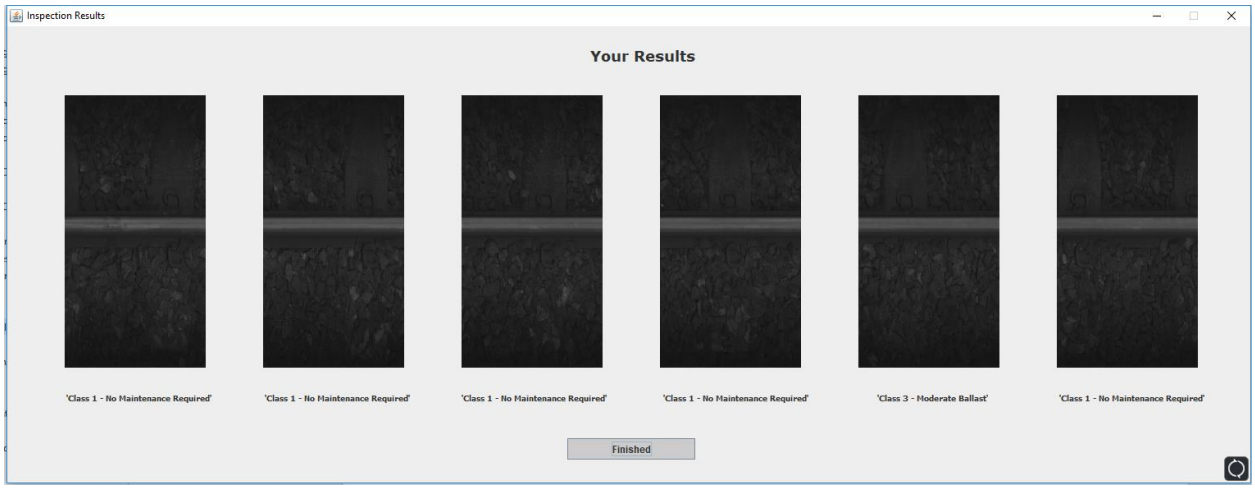


Figure 51 Java Application Prediction Result Screen [fig. 51]

When an image is selected, the Java program will execute Python scripts that will perform necessary data pre-processing tasks; such as slicing the image and down sampling and will feed each of these images into the Neural Network model which will carry out the classification. The predictions that have been made will be returned to the Java program and the above interface will present these predictions to the user along with the section of the track that corresponds to each prediction. The user can click on the 'Finished' button to return to the menu screen.

Figure 52 Java Application About Screen [fig. 52]

If the ‘About’ button is clicked, then a new pop-up interface will appear. This will give details of the technologies that have been used to develop the Neural Network model that is incorporated in the application. The user can go back to the menu screen by clicking the ‘Back to Menu’ button.

The program source code can be viewed in the appendix of this report.

10 Project Management

Project management refers to the organization and careful planning of each component in the project and is one of the most important skills that inherently can determine how successful the project becomes. A selection of project management techniques and their corresponding supporting documentation have been discussed below.

10.1 Project Report Versioning

This report was written in an incremental fashion where each new version added extra content and may have edited, re-formatted, or deleted, existing content. If this report was to be saved as a single document that was constantly updated with the latest version, then issues could arise if I wanted to revert to an older version for any reason as Microsoft Word generally only saves the latest version of each document. To mitigate this, each session where the document was edited or appended to concluded with it being saved with a new version number that was incremented each time.

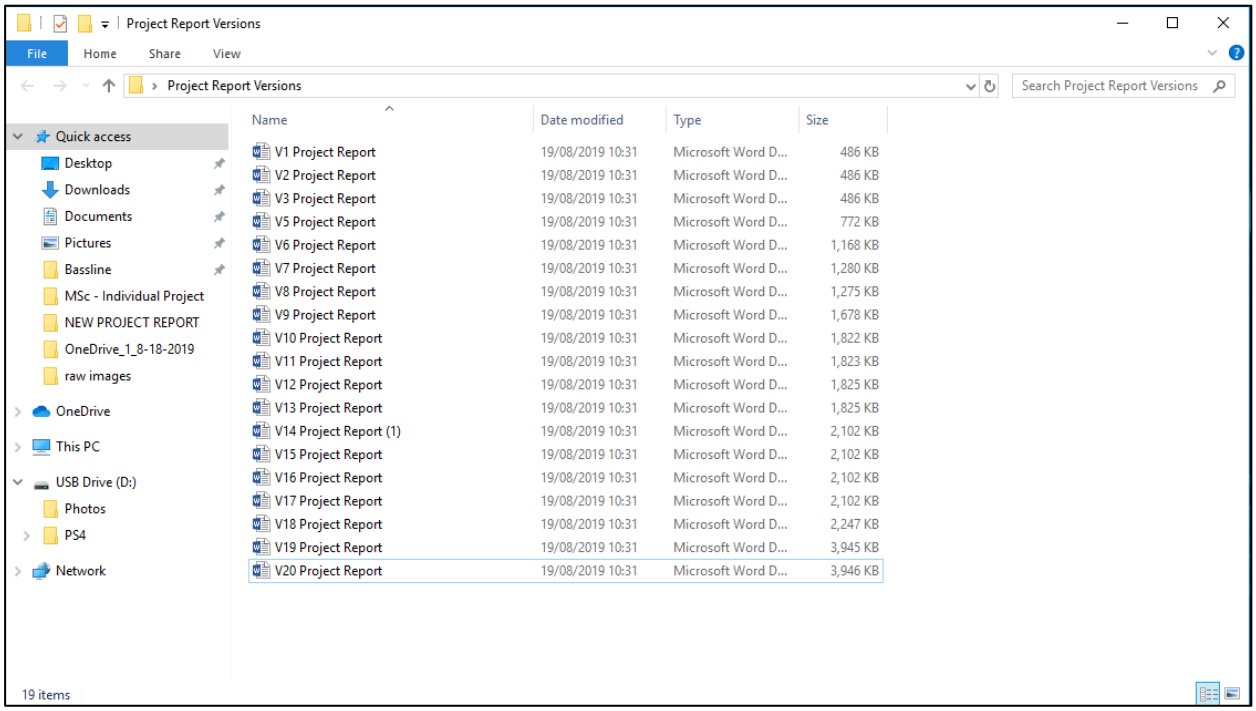


Figure 53 Project Report Versioning [fig. 53]

10.2 Project Report Backup

Extending on the points made in Subsection 9.1, as well as creating a new version for each report edit, measurements were also taken to prevent the accidental loss or deletion of the work that had been completed. Occasional back-ups of the project reports were saved to the University's One Drive and to a personal USB device. Saving the work to a cloud-based platform as well as a secondary physical device meant that if the primary copies did become lost, deleted, or corrupt, then there will always be relatively up-to-date versions that I could fall back on.

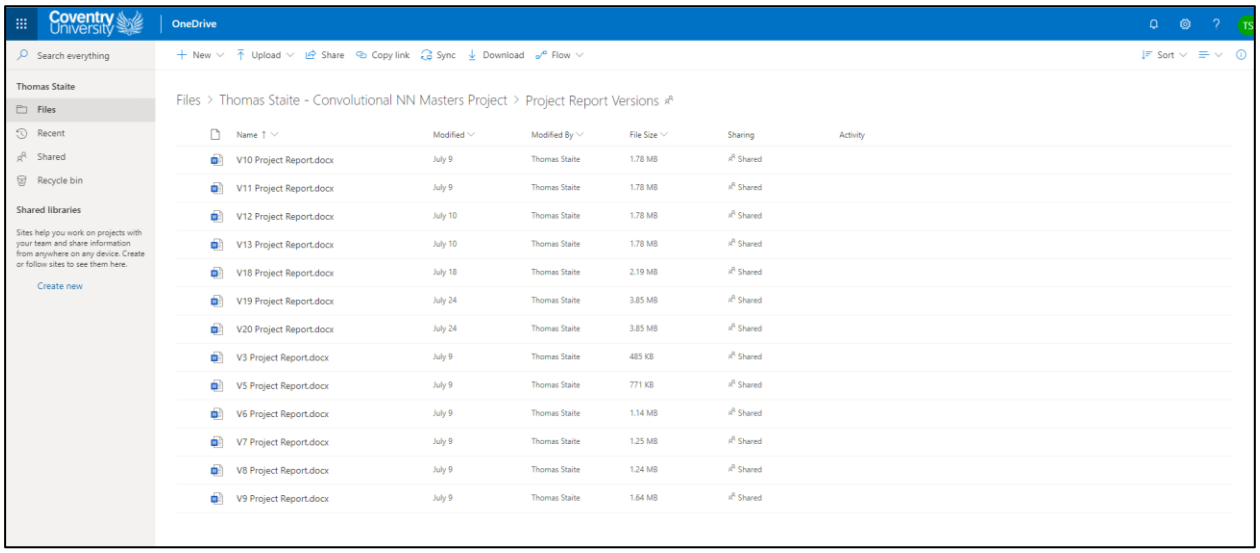


Figure 54 Project Report Cloud Backup [fig. 54]

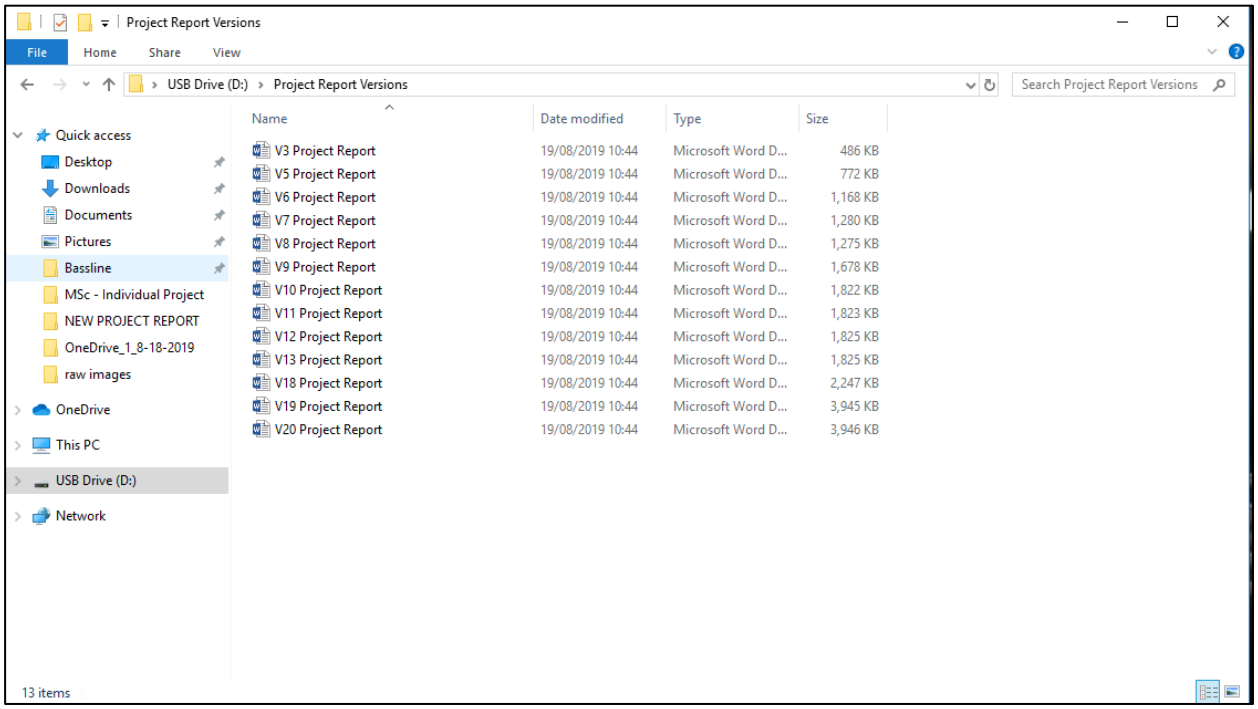


Figure 55 Project Report USB Backup [fig. 55]

10.3 Project Schedule

One difficult task relating to project management and preparation is the scheduling of each task and calculating the overall time required to complete these tasks while ensuring the completion of the whole project is done before the deadline. Various tools exist to help with scheduling and the one that has been used in the past and is a personal favourite is the classical project Gantt Chart. This includes an Excel document that tables all the tasks in the project, their start time, the expected time required to complete the task, the end time, and creates an attractive graph that helps you to visualise the progress of the project through time. This Gantt Chart has been referred to on several occasions and updated to better reflect how the project actually progressed.

Task Name	Task Detail	Start Date	End Date	Duration (Days)	Days Complete	Days Remaining	Percent Complete
Task One	Identify project idea and discuss with	29/04/2019	20/05/2019	21	21.00	0.00	100%
Task Two	Complete and submit proposal form	21/05/2019	24/05/2019	3	3.00	0.00	100%
Task Three	Complete and submit ethics form	27/05/2019	03/06/2019	7	7.00	0.00	100%
Task Four	Receive approved ethics forms / Edit and re-submit if required	03/06/2019	10/06/2019	7	7.00	0.00	100%
Task Five	Perform necessary Project Management techniques (project versioning, OneDrive setup, meeting minutes template, etc.)	03/06/2019	05/06/2019	2	2.00	0.00	100%
Task Six	Write Project Introduction	05/06/2019	07/06/2019	2	2.00	0.00	100%
Task Seven	Perform secondary research on CNN (including; how they work, use cases in image classification)	10/06/2019	13/06/2019	3	3.00	0.00	100%
Task Eight	Literature Review on related studies and previous attempts	14/06/2019	19/06/2019	5	5.00	0.00	100%
Task Nine	Document my approach, including the proposed solution to the problem, and how it can be achieved	19/06/2019	21/06/2019	2	2.00	0.00	100%
Task Ten	Receive feedback from supervisor on work completed and make necessary changes	20/06/2019	27/06/2019	7	7.00	0.00	100%
Task Eleven	Collect and analyse image data required for the project	27/06/2019	28/06/2019	1	1.00	0.00	100%
Task Twelve	Perform any necessary data preparation tasks	01/07/2019	06/07/2019	5	5.00	0.00	100%
Task Thirteen	Develop Convolutional Neural Network models	08/07/2019	15/07/2019	7	7.00	0.00	100%
Task Fourteen	Conduct experiment to compare NN architectures and dataset expansion methods	16/07/2019	22/07/2019	6	6.00	0.00	100%
Task Fifteen	Train the system with collected data, test with testing data, and analyse results	22/07/2019	26/07/2019	4	4.00	0.00	100%
Task Sixteen	Write up development and testing methods into the report and document results obtained	29/07/2019	30/07/2019	1	1.00	0.00	100%
Task Seventeen	Develop a Graphical User Interface (GUI) for the CNN	29/07/2019	02/08/2019	4	4.00	0.00	100%
Task Eighteen	Test GUI and implement any changes necessary	02/08/2019	03/08/2019	1	1.00	0.00	100%
Task Nineteen	Write up conclusions	05/08/2019	06/08/2019	1	1.00	0.00	100%
Task Twenty	Present project to supervisors	07/08/2019	12/08/2019	5	5.00	0.00	100%
Task Twenty-one	Final changes to project report and submit	12/08/2019	19/08/2019	7	7.00	0.00	100%
Task Twenty-two	Project independent marking by supervisor & 2nd Assessor - Project marking feedback.	26/08/2019	02/09/2019	7	7.00	0.00	100%

Figure 56 Gantt Chart Table [fig. 56]

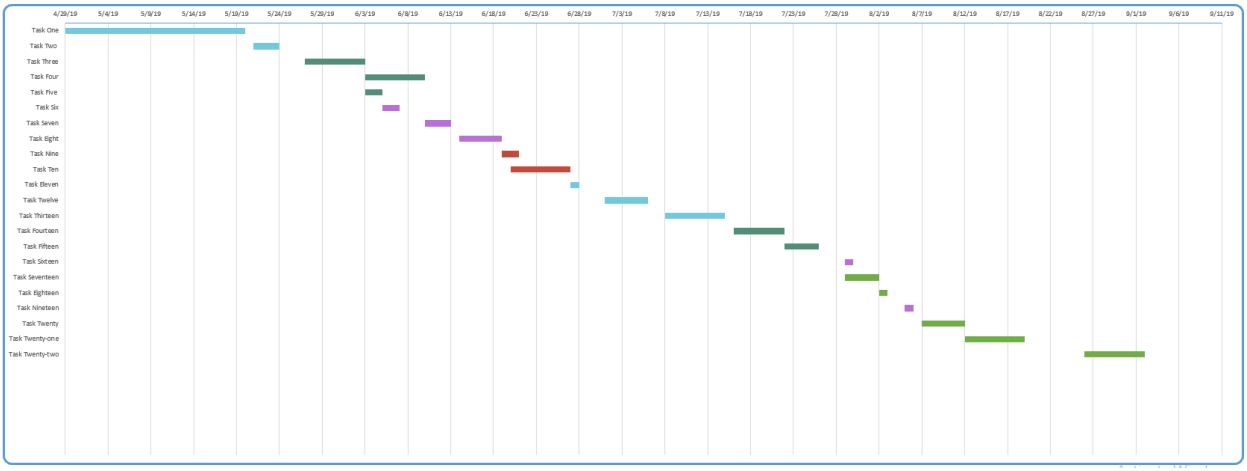


Figure 57 Gantt Chart Graph [fig. 57]

10.4 Supervisor Communication

Weekly meetings with my supervisor provided me with excellent opportunities to speak one-on-one and discuss the ideas that I had come up with, any queries about aspects of the project I was unsure of and enabled me to receive feedback and improvement suggestions that would help me to produce more qualitive work. I have found these meetings very helpful and have tried to always take on board the comments made by my supervisor and action on them where possible. The meeting records have been attached in the appendix.

In addition to the weekly meetings, there was also regular contact between myself and my supervisor via email. This was our primary method of communication and helped to organise and manage said meetings. I found this communication method very useful.

10.5 Quality Management

As previously mentioned in the 'Implementation' section, the incremental project development technique was employed to gradually progress the state of the project. By following this management technique along with the scheduling Gantt Chart, I would evaluate the project at the end of each section that was completed and only move on to the next section when I was completely satisfied with all the work that had been produced. As a result of doing this, not only could I maintain high level quality of work, but also, I could ensure that no content is accidentally left out altogether.

10.6 Social, Legal, Ethical and Professional Considerations

As the project draws to an end, it is important to reflect on and discuss the Social, Legal, and Ethical issues surrounding the nature of the research and the experiments that have been conducted. By highlighting these considerations and keeping them in mind during the process of completing the project, we can monitor the work to make sure it firstly complies with all UK law, but also ensure it is ethical and socially accepted. This is especially important if the work and the developed application is to be implemented into a real-life business scenario.

First and foremost, any project that requires the collection, storage, and processing of private data will need to plan how this data will be gathered, what methods or technologies will be used to securely store the data, what tools will be used during the processing of the data, and also how the data will be safely discarded at the end of the project (if applicable). The dataset used in this project was collected from my supervisor and restrictions were verbally agreed between myself and my supervisor – we agreed that the data will not be shared or distributed in any way and will also need to be stored securely on the university's OneDrive. By complying to this verbal agreement throughout the project, a level of trust was built between Mauro and myself.

If the Java application that has been developed is to be implemented into a real-life business scenario then it is likely that it will be used by a wide community of Rail Engineers whom may differ in age, gender, ethnicity, religious beliefs, disabilities, etc. With that being said, it is paramount that the GUI can be easily understood and navigated by all members of this community. One improvement which is a concern for future work is to allow the application to support language localisation, whereby the user is able to translate the text into a language that they can better understand.

It is mandatory as stated in the university regulations that all project work undertaken by Coventry University students require an ethics approval before research may commence. The Coventry University Ethics website states "*The University Applied Research Committee under the*

leadership of the Pro-Vice-Chancellor (Research) introduced a new Ethics Governance procedure during the 2008/09 academic year. A series of workshops were held to explain the new process to academics during the Summer of 2008. The process is underpinned by taking a risk-based approach – this is to protect the subjects of research, the researchers undertaking work on sensitive topics or objects of study and the University." (Coventry University Registry Research Unit, n.d.) The online ethics form was completed, submitted, reviewed, and re-submitted, and was approved by the university. This form can be found in the appendix of the project

11 Critical Appraisal

The aims and objectives of this project were:

- **Aim**: Design, development, and test a Convolutional Neural Network for an Image Processing classification task which will classify images of railway track into those that require maintenance and those that do not.
- **Objective**: A suitable data set of images will be gathered and pre-processing techniques will be applied to images in the dataset to help prepare it for training the models
- **Objective**: Convolutional Neural Network architectures will be designed, developed, trained, tested, and compared to identify the best performing model for this use case.
- **Objective**: A Graphical User Interface will be designed and developed in Java.
- **Objective**: The CNN and GUI will be consolidated into one product.

An original dataset was supplied by my supervisor and an initial exploration of the images contained in that dataset revealed some glaring issues that required attention. Part of my research was to look at ways in which I could overcome these issues and prepare the data so that it could be used to train and test the CNN models when they developed later on in the project. The process of expanding the dataset by slicing each image into a number of sub-images was fairly straightforward and I managed to achieve the desirable results quickly. However; as mentioned in the 'future work' section, the exact method of image slicing that I would have liked to implement (creating sub-images that perfectly depicted only one sleeper per image) required the separate development of a computer vision system to perform this task – unfortunately, I did not have enough time to research and develop this system and so I had to opt for the simpler method.

Labelling of each individual instance in the now-expanded dataset was a tedious but critical task. As this data was being prepared to train a Machine Learning model, I had no choice but to manually label each instance so that the models could use it to train – this process took up a relatively large amount of available project time but ultimately helped me to achieve the desired results at the end of the experiment.

After the data had been labelled it became instantly clear that a) there was a class imbalance issue, and b) the volume of data to train the models on was still rather small. This is when I began to conduct my literature review into dataset expansion methods and came across instance synthesizing methods like Affine Transformations and Elastic Distortions. I was able to make quick decisions about the research that I would like to conduct, how to go about finding the materials, and what I would like to do in order to advance the literature's current position.

Dimension Reduction concluded the data pre-processing section of the report and upon review of the work that had been completed in this section, I am very happy with how it has turned out – the original data went through a number of phases to finally produce the three datasets that were eventually used to train the model. I believe this work has been approached with a professional and considerate mind-set.

When it came to design the architecture of the CNN's, there was constant reference back to the literature review which was being used to inform the decisions that were made about model configurations and parameter settings. However, sometimes it was not completely clear to me what settings I felt should be used and so during the implementation stage, found myself loosely experimenting with various parameter values – this is bad practice. For future reference, I should try to understand how each parameter affects the behaviour of the models, as this would help me make confident decisions about what settings I believe will provide the best results in my models.

The initial implementation of the CNN models and conduction of the experiment was to be done using TensorFlow (an alternative to Keras which provides many more customizable features), however unsuccessful attempts at implementing these models halted the progress of the project. To ensure I did not fall too far behind schedule, I decided to have a separate attempt at implementation using the Keras library. This attempt provided me with fast results and so it was decided that I would terminate my work in TensorFlow.

The results that have been obtained from my experiments were recorded accurately and the comparison and evaluations have been thorough. The data pre-processing work, and the model design and implementation, enabled me to train model architectures that were able to perform its classification task with over 90% accuracy, which is very satisfying to see.

Finally, the Java application that was developed took a little longer than I had expected as I did not anticipate the level of complexity of the language. Resourcefulness and perseverance meant that even though I may have struggled at times, I was able to find ways to overcome problems. The final version of the application now works flawlessly.

12 Conclusions

12.1 Achievements

Undertaking this project and being guided throughout its process by my supervisor has enabled me to complete an individual piece of research that involved the conduction of professional scientific experiments, that can help contribute to advancing the progress of research in this area. The literature review that was written focused on data pre-processing tasks relevant to image processing and has helped me to identify gaps in previous research, opening up opportunities for me to help improve the quality of other peoples work while also improving my own knowledge in the subject area.

The CNN architectures that were designed and developed helped me to realise that it is not always the most complicated models that provide the best results, and that simpler models can sometimes outperform these complicated models while also being faster to train, as was the case for my implementations. By selecting three architecture sizes, I was also able to find the optimal model for this use case which could be incorporated into a Java application.

Comparing model performance across the three types of dataset expansion techniques helped to answer questions surrounding the best expansion technique to use and whether the introduction of a ‘mixed’ dataset which was not considered in the research paper that I looked at, was able to add value to the research by further improving model performance.

The Java application that was created enabled users to select their own images that they would like to inspect for maintenance. The program was tested on a selection of the original dataset that was provided by my supervisor and each time it was able to complete the classification task efficiently with only few prediction errors being made.

12.2 Future Work

Being restricted in the amount of time I had to complete this project meant that some of the ideas that I would have liked to implement were unable to come to fruition. There are many directions in which this project could be taken and so this ‘Future Work’ section discusses these ideas and how they could be implemented in the future if I chose to continue the work outside of university.

Implementing the neural network models and backpropagation algorithm myself without using the Keras library would help me to build a much stronger understanding of these technologies and how they are used to train the neural network models. This work, however, would require a vast amount of extra time for research into the design and development of these models and how to go about building them from scratch. I did not have enough time to perform this research and so to ensure the experiments could be completed on-time, I opted to use the Keras library to assist me in building these models.

When evaluating the experiment results, it became apparent from the Confusion Matrices that a few of the trained models had issues with separating instances in Class 3 – Moderate Ballast and Class 4 – Excess Ballast. After speaking to my supervisor, we suggested that future work could include combining these two classes into just one general ‘Excess Ballast’ class. The expectation from this research is an improvement in the model performance

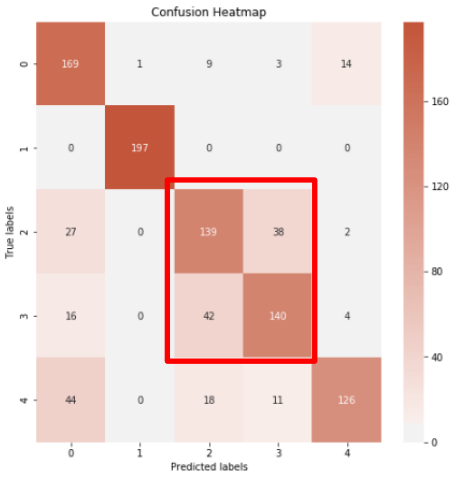


Figure 58 Mixed/Small Architecture Confusion Matrix [fig. 58]

Referring to the data pre-processing task of splitting the images, one major area of future work for this project could be to research ways in which each image could be sliced in a way so that every resulting sub-image contains only one whole sleeper. The current method that was used in this project was to just split the image using equal sized slices – this resulted in a quantity of images that were not suitable for training as they did not contain a whole sleeper, or they contained more than one sleeper. By altering this data pre-processing step so that every sub-image can be used, we will be able to make the fifth class (Class 5 – unclassifiable) redundant. This could also help to improve the accuracy of the developed models.

13 Student Reflections

Allocating some time to reflect on my personal approach to this project and the work that has been completed can help to identify what went well, what didn't quite go to plan, and how improvements to my approach can help me to develop my skills, so that future projects that I may undertake can become of even higher quality. Having the mindfulness to criticize yourself shows that you are willing to better yourself professionally and learn from any potential mistakes or bad practice that you have incorporated into your work.

Upon assignment of the project, it was a top priority of mine to quickly define the scope of the project and understand exactly what Mauro was expecting from me. We held an initial meeting where questions were asked about aspects of the project. This helped me to develop a greater understanding and 'fill any gaps' and ambiguities before project planning could commence. Thoroughly defining the project expectations early on meant that I was fully aware of the steps that needed to be taken in order to complete the work and achieve the aims and objectives.

Discussed further in the 'Project Management' section, a selection of management tools and practices were put to use for multiple reasons: they helped to maintain the quality of the work being produced, they enabled me to monitor the time available to me allowing for the scheduling of progression milestones, they kept documents and resources relating to the project organised, and they also generally helped with keeping track of the progress of the project. Efficient project management is a set of highly-transferable skills that are required in almost every job within the Data Science domain. By practicing my project management skills in this project, I am able to learn more about myself and what my personal preferences are when it comes to documenting and monitoring project progression.

Another positive note on my approach was remaining resourceful. Having previous programming experience in the Python language, I was already aware of a number of technologies and tools that could be utilised in this project. By being resourceful, I was able to find out more information about specific programming environments, programming modules and packages, and data pre-processing methods that all benefited the quality of the work produced. In addition to this, using sources of information that were highly-trusted and officially published by a trusted publisher has helped me to perform the necessary research and literature review, which again has greatly improved the content in this report.

Following on from the previous point on resourcefulness, my supervisor was very approachable and professional and provided me with assistance when requested. Having this frequent level of communication; both electronically through emails and verbally in our meetings, was very beneficial to me. From a high-level point of view, I was able to build a professional relationship with my supervisor, and in terms of the project itself it meant that I could receive honest feedback each week on any work that had completed - this provided me opportunities to refine and correct the work where needs be. I believe that my communication skills and confidence throughout the project has improved.

My organization during this project has also been one of the strengths of my approach. Concentrating on the project management materials and planning each week ahead of time gave the project direction and ensured it had a steady flow of progress. This in turn made sure that I could attend each weekly meeting with Mauro with adequate progress which was reported and then further discussed for continuous feedback. Specific details of weekly progress can be viewed in the meeting records, attached in the appendix of this report.

Developing the data pre-processing tasks, and Neural Network Architectures in Python and the Graphical User Interfaces in Java have enabled me to become aware of new modules and packages that can provide useful tools to improve the quality of my implementations. Some of these modules that have been used in my work are very popular in the wider Data Science community and so it is a great advantage that I have exposed myself to them. Experience in programming in these two languages will be two of the most sought skills when applying for a wide range of roles post-university, and so it is always a benefit to gain more practice using these technologies through project work.

Although it was always clear to me what was expected in terms of the overall project aims and objectives and the project deliverables, I found on a couple of occasions that I had misinterpreted the feedback received from my supervisor. For example, in one of our meetings, Mauro suggested that I begin to research into the data pre-processing tasks that may need to be applied to the dataset. Here he was looking for the **specific** tasks relating to image processing, however the research that I conducted was focused on a general list of data pre-processing tasks – this included many tasks that were completely irrelevant to image processing. With this being said, in the future I should bear in mind any ambiguities in these discussions and should try and clearly define what is required of me every week.

Extending the previous point made, some aspects of the work that did not completely satisfy my supervisor's feedback needed to either be completely re-done or have additional content added to it, this of course took up additional time and meant that at some stages of the project I was beginning to fall behind schedule. When this happened, I had to persevere by adjusting my project Gantt Chart to limit the number of hours I could spend on other tasks in the project, as well as sacrificing some hours at the weekend to catch-up with the work.

Another weakness that I learnt about my approach was the inability to remain persistent when being introduced to concepts that I was not able to grasp straight-away. For example, in the early weeks of the project Mauro wanted me to learn the mathematics behind the backpropagation algorithm. I found this difficult at first and struggled to remain focused on learning the content as it seemed a little overwhelming. Eventually after repeatedly exposing myself to the algorithm and practicing what I had learnt, I was able to understand it in detail and even give a lecture on it to Mauro. This critical analysis skill is very important in technical roles and it is one that I am aware needs a fair amount of improvement.

The report that was written to document the project was originally presented in a two-columned academic paper-styled format. Although this gives the report a professional appearance, it was not the accepted format that the University expects the paper to be written in. Therefore, I had to re-format all of the content into a single-page document with specific headings and style – this was a time-consuming and tedious process. To avoid unnecessary extra work like this in the future, I should be wary of how my work should be presented.

The project PowerPoint document that was created and presented provided me with an opportunity to present my work in a step-by-step manner to my supervisor and also to a 2nd supervisor whom had no prior knowledge on the project that was being completed by myself. A few issues were pointed out and feedback at the end of the presentation. First and foremost, the presentation had a time limit of 15-20 minutes – my presentation went on for almost 40 minutes. If this presentation was to be given in a conference or an interview, for example, then these time constraints can be quite strict and so I should practice the presentation beforehand and time it to see how long I expect it to last for. I can then remove or add slides to decrease/increase the length of the presentation. Moreover, code snippets; unless they are pivotal to the presentation, should not be included as the audience may not be able to read or understand it on first glance.

Finally, some of the slides contained an overwhelming amount of text. Mauro suggested that the audience should be able to read each slide in about 30 seconds and so I should reduce the text down into bullet points which I can expand on when explaining the content.

Bibliography and References

Adit Deshpande, 2016, *A Beginner's Guide to Understanding Convolutional Neural Networks* [online] Available from: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ [accessed on 06/07/2019]

Coventry University Registry Research Unit, n.d. *Ethics at Coventry University* [online] Available from: https://ethics.coventry.ac.uk/about/ethics-at-cu.aspx [accessed on 17/08/2019]
Keras Documentation Website, n.d., *Keras: The Python Deep Learning Library* [online] Available from: https://keras.io/ [accessed on 20/07/2019]

Madhi Varman, 2018, *How to calculate the number of parameters in the CNN?* [online] Available from: https://medium.com/@iamvarman/how-to-calculate-the-number-of-parameters-in-the-cnn-5bd55364d7ca [accessed on 08/07/2019]

Maria Molodova et al, 2014, *Automatic detection of squats in railway infrastructure* [online] Available from: https://www.researchgate.net/publication/265085881_Automatic_Detection_of_Squats_in_Railway_Infrastructure [accessed on 09/07/2019]

Office of Rail and Road, 2018, *Rail Safety Statistics 2017 – 2018 Annual Statistical Release* [online] Available from: https://dataportal.orr.gov.uk/media/1166/rail-safety-statistics-2017-18.pdf [accessed on 03/07/2019]

Office of Rail and Road, 2019, *Passenger and Freight Rail Performance 2018-2019 Q4 Statistical Release* [online] Available from: https://dataportal.orr.gov.uk/media/1240/passenger-freight-performance-2018-19-q4.pdf [accessed on 03/07/2019]

Patrice Y. Simard et al, 2003, *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis* [online] Available from: http://cognitivemedium.com/assets/rmnist/Simard.pdf [accessed on 25/06/2019]

S. Faghih-Roohi, 2016, *Deep convolutional neural networks for detection of rail surface defects* [online] Available from: http://www.dcsc.tudelft.nl/~bdeschutter/pub/rep/16_002.pdf [accessed on 09/07/2019]

Summit Saha, 2018, *A Comprehensive Guide to Convolutional Neural Networks* [online] Available from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [accessed on 06/07/2019]

Yann LeCun et al, *Backpropagation applied to handwritten zip code recognition*, *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989 [online] Available from: http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf [accessed on 05/07/2019]

List of Figures

List of Tables

Appendix A – Project Specification

Appendix A1: MASTER PROJECT BRIEF FORM 2018-19-20

1. Your details:

Full Name: **Thomas Staite**
Student ID: **6422510**
E-mail @coventry.ac.uk: **staitet@uni.coventry.ac.uk**
Module Code: **M08CDE**
Course of Study: **MSc Data Science & Computational Intelligence**
Project Supervisor: **Dr Mauro Innocente**

2. Project title (provisional) [Meaningful, relevant and concise]

Development of Convolutional Neural Network for the Detection of Railway Track Maintenance

3. Outline (synopsis) of your project. [What are the aim and objectives of the project?]

To perform secondary research on Convolutional Neural Networks (CNNs) for the application of image classification – with emphasis on developing an understanding of their inner-workings, why they are preferred over other network architectures for this use case, which data pre-processing techniques are necessary and why, and how such networks can be optimized to yield greater network performance.

To critically review existing literature and attempts in image classification using Neural Networks - identifying and discussing the benefits, drawbacks, limitations, and obtained results in previous studies and relating this back to my own project for constant improvement and refinements. Reviewing previous work will help influence decision making in the project.

To apply the researched data pre-processing techniques to the provided dataset to prepare it for training and testing of the CNN.

To fully develop, train, and test the researched CNN for the application of classifying images of railway tracks into those that require maintenance, and those that do not. To analyse the performance of the network using a range of metrics based on the resulting Confusion Matrix, and to implement parameter tuning and network optimisation techniques in attempt to improve performance.

To design and develop a suitable Graphical User Interface (GUI) to act as the gateway between the user and the developed network – allowing the user to view the locations of the railway track that require maintenance.

4. Intended user or group of users and their requirements. [a) Who is the intended user or group of users? b) Why you think there is need for this project? c) What are the needs of the intended user that your product should satisfy?]

The initial intended user for this project is my supervisor, Dr Mauro Innocente, who works with the Research Institute for Future Travel and Cities (FTC) at Coventry University and whom listed the requirements for the project on the Moodle website.

A potential use case for this project could be to help railway engineers efficiently locate areas of a railway track that require maintenance without having to manually check the whole track themselves – saving both time and energy. The GUI will provide a helpful interface between the end-user (who

COM PG Projects (M08CDE, 7011CEM) Page 1 of 20 Project Brief Form - MSc

may be computer-illiterate) and the developed network and will provide the necessary information needed to allow these users to fulfil their role.

5. Systems requirements and project deliverables. [a) What are the characteristics/properties that the final product should possess? b) What are the process stages and the corresponding deliverables that will enable you to create the final product?]

The Project Report will be written using Microsoft Word and will contain evidence of the research and literature review that was carried out before the development of the CNN. In addition to this, evidence of data pre-processing techniques, final program code, training/testing results, network optimisation, evidence of the developed GUI, project discussions (benefits, drawbacks, and future work), and project management materials will also be included in this report.

Development of the network architecture will be completed in Python programming language and documented in Jupyter Notebook - which enables the combination of Python code and Plain text with various formatting capabilities.

The network should be incorporated as a Python program with a suitable GUI that can be clearly understood and easily navigated by users of all levels of computing ability. User Acceptance Testing (UAT) may be implemented to gain an understanding of end-user's experience, and adaptations may be implemented based on the feedback received.

When the Python code has been extracted from the Jupyter Notebook to its own file, a GitHub repository should be set up to store the program code – allowing for adjustments to be submitted and the ability to rollback to previous versions if errors arise.

6. Research [a) How will you investigate/identify in detail the needs of the specified user in (3) b) How will you investigate the background of the project?]

Weekly meetings with Mauro will provide opportunities for me to ask questions and find out information about areas in the project that have a degree of ambiguity. Feedback from Mauro will help influence the decisions I make and will ensure that the project will run as smoothly as possible and that the final product meets (and hopefully surpasses) his expectations.

Further research into the technologies that I will be using throughout the project will involve examining external resources such as content on the internet and from the University Library. Full care must be made to ensure the sources of this secondary research is reliable, accurate, and up-to-date. In addition to this, when it comes to reviewing literature of previous attempts, I should be vigilant about when and where the study was conducted, whether it's up-to-date, and whether the results obtained are accurate and trustworthy.

When it comes to developing the CNN, functions from a selection of external Python libraries will be used to aid with the development. Official documentation for these libraries can be accessed online and should be the main source of information regarding these libraries.

7. Evaluation. [a) What makes a product successful? b) How will you demonstrate that your product fulfils the needs of the user in (3)? c) How will you evaluate the product?]

A conclusive piece of research into CNNs and their application in image recognition, along with a critical literature review of previous studies will demonstrate competence in project research. The work completed in this section will be presented to Mauro who may provide feedback.

A successful data preparation stage will involve overcoming the limitations of the supplied dataset (namely, the lack of classification labels, the limited number of images, and further issues with the images themselves) to produce a new dataset which will be fit to train/test the model. This new dataset should be correctly labelled, should have a much-increased quantity of images, and should be resolved of any issues with the image themselves.

For the CNN itself, it will be trained and tested using the resulting dataset from the Data Preparation stage. Performance testing metrics will be employed to provide details on how well the model has been able to train and generalise, and optimisation will hopefully further improve the model. A successful model should yield high predictability power on both the training and the testing set.

As previously mentioned, a User Acceptance Test (UAT) may be implemented to provide Mauro and any other potential end-users with an opportunity to try the product. Any feedback from this test will be used to implement changes; whether necessary, or preferable. This stage will also act as a demonstration of the work that has been completed.

The project will be written professionally using a Microsoft Word document and an accompanying template. Every stage of the project should be fully documented so that when it is submitted, my supervisor, and 2nd assessor has evidence that the work has been completed, that it is accurate, and that it can be reproduced in the future. In addition to this, my supervisor should have access to any program code or project management documentation that he requires.

8. Development skills. [a) What information and resources do you need to complete the project successfully? b) Which of these do you need to acquire yourself?]

During the literature review section, it is paramount to be able to dissect previous studies and critically review what their attempt included, where they were successful/unsuccessful, how they went about completing their project and how they achieved their results. Being able to relate this back to my project will provide me with a lot of useful information that can help influence the decisions I make. The BSc project that was completed last year began to develop our research and reviewing skills, and the MSc project should help us continue to develop these skills.

Being able to comfortably program in Python will be necessary for the Data Preparation phase as well as the actual development of the CNN and its GUI. Knowledge of the language and the external libraries that will be used will provide me with the technical skills needed to be able to complete the work efficiently.

Having the competence to conduct professional and credible scientific experiments – in this case testing various values for parameters of a CNN model (number of layers, initial weights and biases, learning rate, etc.) will be critical in ensuring the project yields accurate and repeatable results. Future work completed in this area of Data Science may involve using my work as part of someone else's literature review. They must have confidence that my work is rigorous otherwise they may not trust it.

Being able to effectively communicate the progress of the project to Mauro will ensure that he is kept up-to-date with the work that is being done. Any ambiguities in the project will need to be resolved and this can only be done if communication channels are in-place and frequently used. Weekly meetings will provide me with opportunities to discuss the project face-to-face with Mauro and provide great opportunities to improve soft skills at the same time.

9.Skill acquisition. [How do you intend to gain the skills, information and resources specified in (7)?]

During the literature review section, I should be considerate about other people's work and should be open-minded about 'why' and 'how' they went about completing their projects. I should try and find reasons 'for' and 'against' the methodologies and technologies other people used and should provide backup research to help support the opinions I form. If I can do this, then my literature review will be critical but fair.

Having already completed an undergraduate degree in Computer Science, my Python programming skills have been developing for the past 3 years. Knowledge of the external libraries relating to Data Science however is a relatively fresh challenge for me. When it comes to Data Preparation, and development of the CNN, I should read about popular libraries that assist in this work (NumPy, Pandas, SciPy, SciKit, etc.) to gain a better understanding of how they operate.

Researching the correct approach to conducting scientific experiments with a focus on what variables need to be recorded and how to document the experiment will be one way in which I will be able to prepare for the development phase of the project. In addition to this, Mauro may be able to provide feedback on my work and indicate what may need changing.

As mentioned in the previous section, arranging weekly meetings with my project supervisor and taking advantage of these meetings will help me develop my soft skills. In addition to this, discussing the ambiguates of the project will enable me to work on my problem-solving skills and will help improve my confidence when working around clients.

10. Estimate the number of hours you are planning to spend for each of the following tasks:

Background research and learning new skills	60
Literature review of previous attempts	40
Data gathering and Pre-processing	25
Product development	100
Product evaluation and adjustments	40
Final report preparation	25
Other – Project meetings, project management, experiment setup	30
Total number of hours	320

Important note: Ethical approval must be obtained for ALL research projects and BEFORE any data are collected.

All students MUST get ethics approval on projects. Any project which has not received approval will be failed.

You MUST complete the Ethics Online Procedure. You CANNOT start with data collection until your ethics application has been approved. Failure to comply will result in you not continuing with your project hence automatically fail your project

See CU Ethics About at < https://ethics.coventry.ac.uk/about/default.aspx>

The grade for the Project Brief may only be awarded if the Ethics Online Procedure has been completed for approval by supervisor.

This form must be submitted electronically on your Moodle project web ***before*** 18:00 on the due date.

I will complete my Ethics application within one (1) week from the Project Brief Deadline: YES ☑

Signature Tom Staite Date 28/05/2019

Appendix B – Meeting Records

Meeting Date: 23/05/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	23/05/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- A completed project brief printed and ready to be reviewed before submission online. Mauro and I discussed section 10 (estimated hours spent on the project) in detail to try and come up with plausible estimations.
- We identified key research areas that will need to be addressed during the project – this included transfer learning, network hyper-parameter experiments, and how to conduct a scientific experiment professionally and accurately.
- Mauro informed me about the meeting records that needed to be completed after each meeting and discussed the purpose of keeping them both for project management and as evidence of the meeting taking place.

Agreed Action Points:

- To consolidate previous academic work on neural networks and store it on a USB device ready for next week's meeting.
- To obtain and fill out this meeting record and subsequent records for future meetings.
- To review section 10 of the project brief, adjusting the estimated number of hours for each section of the project so that they reflect a more accurate project duration.
- To begin looking at the research areas discussed in the meeting in preparation for the commencement of the research and literature review sections of the project.

Date and Time of next meeting:	30/05/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Meeting Date: 30/05/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	30/05/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- I began by informing Mauro of the work that had been completed since our last meeting. This included the reviewed Project Brief that was submitted online, the first attempt of the Ethics approval, the completed meeting records, and a revision on the basic concepts of ANN and the Backpropagation algorithm.
- The previous academic work on Neural Networks that was requested by Mauro was brought in on a USB device and we went through the content. Mauro was able to see the lecture slides that were used last semester for the Artificial Neural Networks module and so has an idea of what I have been taught and where there may be gaps in my knowledge.
- We discussed the architecture and properties of ANN and spoke about how to carry out research in terms of the content that needs researching and how to split the available research time to ensure the project does not fall behind. We also discussed varies ways of experimenting with hyper-parameters of the network and how we can measure performance of a model for both the training and the testing stages.

Agreed Action Points:

- To await feedback on the online Ethics approval and to make any necessary changes that Mauro reports on.
- To revise and produce a 20 – 25-minute lecture on the basic concepts of Artificial Neural Networks, the Backpropagation algorithm (making sure to include the relevant mathematical equations and formulas) and techniques for network optimisation, ready to present this lecture to Mauro during the next meeting (06/06/2019)

Date and Time of next meeting:	06/06/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Meeting Date: 06/06/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	06/06/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- A PowerPoint presentation on an introduction into Artificial Neural Networks, Gradient Descent, and Backpropagation was created during the week and presented to Mauro at this meeting.
- Mauro asked me questions on the content of the PowerPoint to see whether I understood the concepts and mathematics behind how the basic Neural Network model works. We agreed that at this current moment in time I did not yet fully understand these topics enough to be able to implement my own network.
- Mauro suggested that I should go away and try to implement a simple Neural Network with just one hidden layer and a small number of neurons in each layer. I should then use the backpropagation algorithm to complete a forward pass; evaluating the performance of the network on a single training pattern, and a backwards pass; updating each individual weight and bias in the network in order to decrease the cost/increase the networks performance.

Agreed Action Points:

- To make necessary changes to the project Ethics form and to re-submit it before the deadline.
- To manually implement the neural network described above in Key Point no. 3, while in the process learning the concepts and mathematics behind gradient descent, and the backpropagation algorithm.
- To produce a lecture/presentation on the simple NN that I have implemented and to go through the backpropagation algorithm step-by-step using a real training example and randomly initialised weights and biases.

Date and Time of next meeting:	13/06/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Meeting Date: 13/06/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	13/06/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- The major focus of this meeting was to present Mauro with a lecture on the simple 3-layered neural network that I had implemented during the week. I took Mauro through a step-by-step example of the backpropagation algorithm, explaining how we first evaluate the performance of the network on a training example with the current weights and biases, and discussed how we can look to increase this performance by using gradient descent techniques to reduce the overall cost. A step-by-step walkthrough of the backwards pass was then explained, where we calculated the updates of the weights and biases at the output layer by using the chain rule, and then backpropagated this to the hidden layer to allow us to update the weights and biases in this layer.

Agreed Action Points:

- To make necessary changes to the project Ethics form and to re-submit it before the deadline.
- To implement the NN described in the lecture in Python, using matrix and vector representations of the inputs, weights, and biases to decrease the learning time needed to train the network.
- To look at data pre-processing techniques that may be applied to our training data in order to prepare it for training of the neural network.
- To research why weights and biases are randomly initialised between specific upper and lower limits and whether the choice of activation function influences these limits.
- To research and find out if weight and bias updates should also be constrained to have values between a specific upper and lower limit or whether they may take any value.

Date and Time of next meeting:	20/06/2019 at 09:00

Signatures of those present:

Supervisor:

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Student: Tom Staite

Meeting Date: 27/06/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	27/06/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- The OneDrive directory had been created and all the content relating to the project have been uploaded and stored in this directory. This directory has been shared with Mauro.
- I explained why the inputs and weights in a feed-forward neural networks are normalised to values between a small range to prevent the outcome values after applying an activation function to have gradients near 0, leading to network saturation where training will either stall or occur very slowly.
- Data preparation tasks relevant to image processing and more specifically the use case for this project were discussed. Mauro and I spoke about image segmentation, identifying classes and applying training labels to the data, how to solve the class imbalance problem, and how to represent the image as a matrix of pixels.
- The literature review that I had started was presented to Mauro. This included research into affine transformations and elastic deformations for synthetic production of new instances to help mitigate the class imbalance problem.
- I spoke about how a standard feed-forward neural network performs poorly on image data due to the unmanageable quantity of network parameters that require learning, and how a CNN is optimised for image processing by taking advantage of local connectivity and parameter sharing, resulting in a reduction in the number of parameters required to train the model.
- Finally, I discussed the types of layers in a CNN (convolutional, pooling, fully-connected) and what they consist of. Mauro suggested that I may not need to use padding in my architecture as the features of interest will be located at the centre of each image.

Agreed Action Points:

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

- To research ways in which we can automate the process of segmenting the images in the data set without losing any valuable information, and then save them individually, resulting in an increase to the size of the dataset.
- To manually review the images and decide on the class labels that will be used when training the network. I should check if there are more than one type of maintenance a section of the track could require and if so should consider this during the preparation phase.
- To continue the literature review – resuming work on techniques used for synthetic production of instances (to handle the class imbalance problem) and then reviewing literature on CNN architectures for image processing to identify what other academics have achieved and how this can relate to my work.

Date and Time of next meeting:	11/06/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Meeting Date: 11/07/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	11/07/2019	**Time:**	09:00

Key Points Brought to the Meeting:

- The literature review that I had continued to work on was presented and feedback on this was provided.
- Mauro and I spoke about ways in which the images in the provided dataset could be sliced into a number of sub-images in attempt to increase the size of the trainable dataset. We decided that I would experiment with several slice sizes to see which one resulted in the highest number of useable training sub-images
- Other suitable image preparation tasks were discussed including data synthesis, handling class imbalance, and manually labelling the data.
- The experiment to be conducted was outlined. I decided that I would like to test three datasets (affine, elastic, and mixed) on three network architectures (with increasing size), equalling a total of 9 tests – this would be done to see:
  a) if; similarly, to one of the studies I reviewed, the elastic dataset performed better than the affine dataset, and to see if the mixed dataset (containing both affine and elastic instances) was able to provide even better performance than the previous two.
  b) Which network architecture size provided the best performance on the three datasets and whether there was a large training time / model accuracy trade-off.

Agreed Action Points:

- To experiment slicing a small proportion of images into 6, 7, and 8 equal slices and manually calculate which slice quantity provides the highest percentage of useable data
- To apply this slice quantity to all images in the dataset, and then to manually examine each instance in the dataset and categorize it into one of the pre-determined classes.
- To create three new datasets from these classes while handling the class imbalance problem by applying affine transformations and/or elastic distortions to instances in the original dataset
- To perform downsampling on the instances in each of the three datasets

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

- To design the three network architectures that I will conduct the experiments on and to develop the three network architectures in Python on Google Colab.

- To carry out all 9 tests, training each network with each dataset, and to record the results from the experiment for evidence and so that they can be compared.

Date and Time of next meeting:	18/07/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Meeting Date: 25/07/2019

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

Face-To-Face Student/Supervisor Meeting Record

Project Title:	Development of Convolutional Neural Network for the Detection of Railway Track Maintenance		
Student Name:	Thomas Staite	**Student ID:**	6422510
Supervisor:	Mauro S. INNOCENTE	**Student UID:**	
Date Today:	25/07/2019	**Time:**	10:30

Key Points Brought to the Meeting:

- An experiment was conducted to split the dataset with 6, 7, and 8 equal size slices to see which one produced the highest percentage of sub-images that were useable. This slice quantity was then applied to every instance in the dataset to dramatically expand the number of instances.
- It was decided that 5 classes would be used in this project and the task of manually passing through each image in the expanded dataset to filter them into one of the 5 pre-determined classes was accomplished.
- Three new datasets were created from these 5 classes. One that had new synthesized images by applying affine transformations, one that had new synthesized images by applying elastic distortions, and one that had a mixture of affine transformations and elastic distortions applied to it.
- The images were down sampled from dimensions 2048 x 1072 to 256 x 134 so that the whole dataset could be imported into the program without any memory issues and allowing the network architectures to train on it effectively.
- The three CNN architectures were designed and the program code for these models were written in Python. Each of the three datasets were trained on each of the three models for a total of 9 tests. The results and evaluation metrics were recorded.
- Mauro and I spoke about how two of the classes were similar in nature and so another implementation could see the combining of these two classes. We also spoke about model overfitting which is apparent in some of the networks that were trained.

Agreed Action Points:

Coventry University MSc Project Meeting Record Faculty of Engineering & Computing

- To discuss the overfitting that is apparent in some of the models that were trained and to re-run the experiments using a validation set which will help combat overfitting.
- To develop the Graphical User Interface in either Python or Java and to incorporate the best CNN from the 9 tests. The user should be able to input a new image into this program and the program should be able to apply the necessary data preparation tasks before testing the image on the CNN model for classifications.
- To continue writing up the project report.

Date and Time of next meeting:	01/08/2019 at 09:00

Signatures of those present:

Supervisor:

Student: Tom Staite

Appendix C – Project Presentation

DEVELOPMENT OF A CONVOLUTIONAL NEURAL NETWORK FOR THE DETECTION OF RAILWAY TRACK MAINTENANCE

Project Presentation by Thomas Staite (6422510)

1

Presentation Outline

- - Introduction and Project Motives
- - Exposure to Track Inspection task
- - Automation of Track Inspection task
- - Data Gathering and Pre-Processing tasks
- - Research/literature review
- - Expansion of dataset (Affine, Elastic)
- - CNN Architecture Designs
- - CNN development
- - CNN training, testing, experiment results
- - Graphical User Interface – Java Application

2

Introduction and Project Motives

Source: https://www.railworks.com/services/inspection-maintenance

- Railway track inspection for maintenance needs can be a tedious, time-consuming, yet critical task – railway engineers have to manually navigate the track and inspect each section.
- Moreover, the need for fulfilment of this task can have a number of repercussions, including delays to the rail transport service as well as introducing the risk of work-related injuries.

- Some Statistics:
- Nationally, for Q4 (1st Oct – 31st Dec) of 2018, 13.7% of trains were late by more than 5 minutes to their destination, with 4.3% 'Cancelled or Significantly Late'.
- 6,661 workforce injuries reported in 2017, averaging **over 18 injuries each day**. 164 of these injuries classified as 'major injuries'.

(Both statistics published by 'The Office of Rail and Road')

- It can be suggested that some percentage of these statistics may be related to this important and repetitive task of railway track inspection.

3

Alternative approach to manual inspection of the Railway Track

- Build a system that can automate the task. The system will receive information about the railway track and will 'decide' whether it requires maintenance or not. This can be thought of as a Classification task.
- Automating the task could deliver multiple benefits:
- - Reduce the time required to inspect each section of track...
- ... which can lead to quicker identification and scheduling of maintenance needs...
- ... which could in turn prevent or reduce delays to the transport service.

- - Also reduce the number of workers required to complete the task...
- ... which could reduce the quantity of work-related injuries...
- ... while helping rail companies reduce costs associated with employment (hiring, wages, etc.)

4

The Inspection Task

5

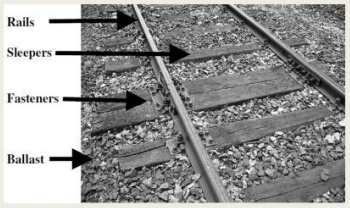
Basic Overview of track components

Rails: The lines which the train travels on

Sleepers: A support for the railway lines. The lines are fastened to the sleeper, and the sleeper maintains a fixed amount of space between the lines.

Fasteners: The component used to secure the railway line to the sleeper.

Ballast: the track bed upon which the Sleepers lay on. It is used to bear the load From the railroad, facilitate drainage of Water, and keep down vegetation that may Interfere with the track structure.



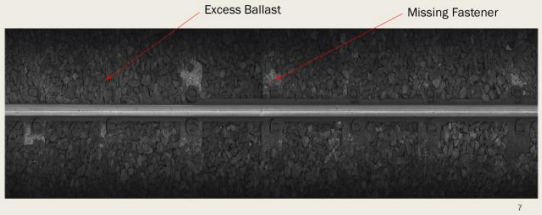
Source: https://knowledge4civil.wordpress.com/2016/12/17/components-of-permanent-way/

6

Specific Maintenance Tasks

1. **Missing Fastener:** The component used to secure the rail to the sleeper is broken or missing

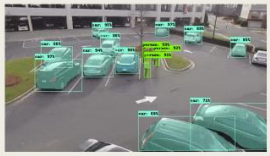
2. **Excess Ballast:** Too much ballast on the sleeper prevents us from checking the existence and condition of a fastener.



7

Task Automation with Machine Learning

- Analysing track components is a visual task – requires engineers to look and evaluate presence and condition of components.
- It is a type of visual or 'Image Processing' task.
- Machine Learning technologies; and more specifically, Convolutional Neural Networks (CNN) have proven to be effective at automating Image Processing tasks.
- Some implementations of CNN for Image Processing include: facial recognition, medical imagery analysis, computer vision for self-driving cars.



Source: https://towardsdatascience.com/how-to-do-everything-in-computer-vision-2ab1e42c4699?gi

8

Automation Process Steps

- - Acquisition of a relevant image dataset
- - Data Pre-processing tasks
- - CNN Architecture designs
- - CNN Architecture development and training
- - CNN Architecture testing
- - Graphical User Interface (GUI)

9

The Original Dataset

- The original raw-image dataset for this project was supplied by Mauro and consisted of 997 images (5.8GB).
- Each image in the dataset depicted a section of railway track – typically between 6 – 8 sleepers.
- Each image had dimensions: 6432pxl width x 2048pxl height .
- Each image had only one colour channel (grayscale)

Example Images From The Dataset

10

Initial Review of The Dataset

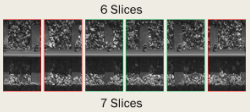
- Upon initial review of the gathered dataset, it was clear that there were a number of preparation issues that needed to be resolved, namely:
- - **The images were too large** – Training a CNN with small images is a computationally expensive task. Therefore attempting to train the network with images of dimensions 6432 x 2048 on a standard desktop PC would almost certainly result in 'out of memory' issues.
- - **low-volume dataset** – In Machine Learning, the greater the volume of training and testing data, the more reliable our models become. 997 images may simply not be enough to train a CNN to perform our classification task accurately.
- - **Unlabelled data** – Each instance in the acquired dataset had not yet been labelled, and so it became apparent that I would need to spend time manually labelling each image.

11

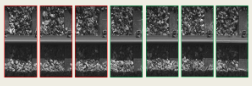
Resolving issues with the original dataset – Data Pre-processing part 1

- One way to expand the volume of the dataset while reducing the image dimensions is to split each image into a number of equal sized 'sub-images'.
- But how many slices should be used?
- Experiment was conducted to determine which quantity of slices resulted in the highest percentage of useable training images.
- 20 images from the dataset were split using 6, 7, and 8 slices, and the percentage of useable images were calculated.
- Results:
- **6 slices- 70.83%**
- 7 slices – 50.00%
- 8 slices – 48.75%

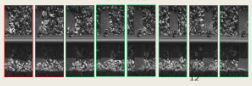
6 Slices



7 Slices



8 Slices



12

Python Code – Image Slicing

```
def importImages():

    X_data = []
    for i in range(1,21):
        image = cv2.imread(r'A:/Documents/Coventry University/MSc/MSc - Individual Project/raw images/RailwayImage (' +
        X_data.append (image)

    npImageArray = np.array(X_data)

    print('X_data shape:', np.array(X_data).shape)

    return npImageArray


imageArray = importImages()


#img = Image.fromarray(imageArray[17])
#img.show()
```

```
def cropImage(images, cropSize):
    fileNumber = 1
    boundary = int(6432/cropSize)

    for j in images:
        for k in range(0,cropSize):
            imageSlice = j[:, (k-1)*boundary:((k+1)*boundary), :]
            writeToFile = Image.fromarray(imageSlice)
            writeToFile.save(r'A:/Documents/Coventry University/MSc/MSc - Individual Project/CroppedImages/'+str(cropSi
            fileNumber = fileNumber + 1


# Slice images into 6, 7, 8 equal slices
cropImage(imageArray, 6)
cropImage(imageArray, 7)
cropImage(imageArray, 8)
```

13

Expanding the dataset by slicing each image into 6

- From the previous experiment it was found that splitting each image into 6 equal slices yielded the highest percentage of useable data.
- Therefore each image in the dataset was split into 6 equal slices to give a total of **5,982 images** – we can expect roughly 70% of this data to be useable.
- Each one of the new images had dimensions 1072 width x 2048 height.
- To help speed up network training time, and to ensure no memory issues occurred when training the CNN model, I decided to reduce the dimensions of each image…

14

Scaling Images Down

- The images in the current dataset had dimensions 1072 width x 2048 height.
- After applying scaling to downsample the images, each image now had dimensions 134 width x 256 height – An eighth of the original size.

Python Code

```
def Resize(img):
    height = 256
    width = 134
    dim = (width, height)

    resized = cv2.resize(img, dim, interpolation=cv2.INTER_LINEAR)
    return resized
```

```
i = 1
while i <= 2000:
    img  = np.array(Image.open(r'A:/Documents/Coventry University/MSc/MSc - Individual Project/Classes/Class 5/Climage
    newImg = Resize(img)
    imageToWrite = Image.fromarray(newImg)
    writeToFile = imageToWrite
    writeIcon = i + 48
    writeToFile.save(r"A:/Documents/Coventry University/MSc/MSc - Individual Project/Classes/AffineAndElastic/Class 1/C
    i=i+1
```

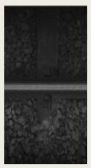
15

Labelling the Dataset

- - A lengthy process that involved manually labelling each image with one of five pre-defined class labels….

Class 1 – No Maintenance Required

Class 2 – Missing Fastener



Class 3 – Moderate Ballast

Class 4 – Excess Ballast

Class 5 – Unclassifiable

16

Class Imbalance and Literature Review – Part 1

- The dataset at this point consisted of 5,982 fully-labelled images. Upon completion of this labelling task, it was evident that class imbalance issues were present.
- In order to mitigate issues with class imbalance, it was decided that each class should contain an equal quantity of images that the network will train on – 2,000 images in each class (for a total of 10,000 images) was the chosen amount.
- The first part of the literature review for this project explored ways in which we could reach 2,000 images in each class by synthesizing new instances.
- One paper by Patrice Y. Simard et al from Microsoft Research involved a comparison of synthesis by applying Affine Transformations and Elastic Distortions.

17

Elastic Distortions

- The method outlined in the literature:
- Create random displacement fields for height and width, with values randomly sampled from unif(−1,1)unif(−1,1). A displacement field defines a direction and magnitude to move a pixel.
- Smooth the fields with a gaussian filter. Since μ=0μ=0 for unif(−1,1)unif(−1,1), most values will be close to 00 after the gaussian filter is applied. Thus most of the changes made by the fields will be small (assuming the gaussian filter's sigma value is large enough).
- Multiply the fields by a scaling factor to control intensity of the deformations.
- Use interpolation to apply the displacement fields to the image.

21

Affine Transformations

- A function between affine spaces which preserves points, straight lines and planes.
- Sets of parallel lines remain parallel after an affine transformations
- Does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line.
- Examples include:
- - Translation
- - Scaling
- - Reflection
- - Rotation
- - Sheer Mapping

Examples

Elastic Distortions - Examples

22

Affine Transformations – Python Code part 1.

```
def randomTransformationsChoice(image):
    choice = random.randrange(3, 6)
    if choice == 1:
        transformedImage = ApplyRotation(image)
    elif choice == 2:
        transformedImage = ApplyScaling(image)
    elif choice == 3:
        transformedImage = ApplyTranslation(image)
    else:
        transformedImage = ApplyShearing(image)
    return (transformedImage)
```

Rotation

```
# RANDOM ROTATION BETWEEN -0.1 AND 0.1
def ApplyRotation(newImg):
    choice = random.uniform(-0.1, 0.1)
    tform = AffineTransform(scale=(1, 1), rotation=choice, shear=0, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Scaling

```
# RANDOM SCALING BETWEEN 0.92 AND 1.02
def ApplyScaling(newImg):
    choice = random.uniform(0.92, 1.02)
    tform = AffineTransform(scale=(choice, choice), rotation=0, shear=0, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

19

Elastic Distortions – Python Code

```
def elastic_transform(image, alpha, sigma, random_state=None):

    '''Elastic deformation of images as described in [Simard2003]_.
    .. [Simard2003] Simard, Steinkraus and Platt, "Best Practices for
       Convolutional Neural Networks applied to Visual Document Analysis", in
       Proc. of the International Conference on Document Analysis and
       Recognition, 2003.
    '''
    if random_state is None:
        random_state = np.random.RandomState(None)

    # Random_state.rand - creates an array of the given shape and populates with random samples from a uniform dist. ov
    # Sigma - Elasticity coefficient
    # Alpha - Scaling factor to control intesity of deformation

    shape = image.shape
    dx = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma, mode="constant", cval=0) * alpha
    dy = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma, mode="constant", cval=0) * alpha

    # Return co-ordinate matrices from co-ordinate vectors
    x, y = np.meshgrid(np.arange(shape[1]), np.arange(shape[0]))
    indices = np.reshape(y+dy, (-1, 1)), np.reshape(x+dx, (-1, 1))

    distored_image = map_coordinates(image, indices, order=1, mode='reflect')
    return distored_image.reshape(image.shape)
```

23

Affine Transformations – Python Code part 2.

Translation

```
# RANDOM TRANSLATION BETWEEN -100 AND 100 (BOTH AXIS)
def ApplyTranslation(newImg):
    xchoice, ychoice = random.randrange(-100, 100), random.randrange(-100, 100)
    tform = AffineTransform(scale=(1, 1), rotation=0, shear=0, translation=(xchoice, ychoice))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

Shearing

```
# RANDOM SHEARING BETWEEN -0.1 AND 0.1
def ApplyShearing(newImg):
    choice = random.uniform(-0.1, 0.1)
    tform = AffineTransform(scale=(1, 1), rotation=0, shear=choice, translation=(0, 0))
    image = warp(newImg, tform, output_shape=(2048, 1071))

    coords = corner_peaks(corner_harris(image), min_distance=5)
    coords_subpix = corner_subpix(image, coords, window_size=13)

    return (image)
```

20

Literature Review – Part 1 (Cont.)

The results from the study by Patrice Y. Simard et al presented a decrease of 0.2% in the error when they used Elastic Distortions rather than Affine Transformations

Algorithm	Distortion	Error
2 layer MLP (CE)	None	1.6%
2 layer MLP (CE)	Affine	1.1%
2 layer MLP (CE)	Elastic	0.7%
Simple Conv (CE)	Affine	0.6%
Simple Conv (CE)	Elastic	0.4%

What they didn't do:-

A gap in their research is that they did not consider the possibility of preparing a dataset that included both instances that had undergone Affine Transformations mixed with instances that had undergone Elastic Distortions – could this result in an even smaller error value?

24

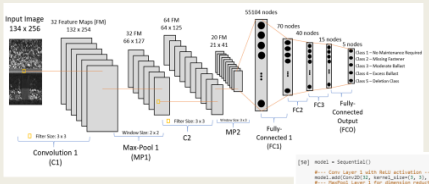
CNN Architecture Design and Literature Review – Part 2

- Second part of the review was concerned with architecture design
- A paper by S. Faghih-Roohi et al compared three network architectures with increasing scale for a similar use case of rail maintenance detection.
- A 'small' network, a 'medium' network with an extra Convolutional and Max-Pooling layer, and a 'large' network with an extra Fully-Connected layer.
- The number and size of feature maps, and the Max-pooling window size also varied amongst the three architectures

Deep convolutional neural network				
Type of layer		Small	Medium	Large
Convolutional layer 1	Feature maps Filter size	6 17 × 9	10 9 × 5	20 9 × 5
Max-pooling layer 1	Filter size	3 × 3	2 × 2	2 × 2
Convolutional layer 2	Feature maps Filter size	10 8 × 3	20 9 × 6	40 9 × 6
Max-pooling layer 2	Filter size	3 × 3	2 × 2	2 × 2
Convolutional layer 3	Feature maps Filter size	- -	30 4 × 2	60 4 × 2
Max-pooling layer 3	Filter size	-	2 × 2	2 × 2
Fully-connected layer 1	Number of nodes	70	120	320
Fully-connected layer 2	Number of nodes	18	30	80
Fully-connected layer 3	Number of nodes	-	-	8
Estimated number of parameters (weights)		22000	120600	644200

25

Large CNN Architecture



29

CNN Architecture Design

- Inspired by the study outlined on the previous slide, I decided that I would like to train and compare three varying network architectures.
- All three of my architectures had 2 Convolutional and Max-Pooling layers, however the number of Fully-Connected layers varied based on the architecture size
- Likewise, the number and size of the feature maps and Max-Pooling windows also varied.

Convolutional Neural Network Architecture				
Layer Type	Hyper Parameter	Small	Medium	Large
Convolutional Layer 1	Feature Maps Filter Size Stride Padding	6 5 x 5 1 valid	10 5 x 5 1 valid	32 3 x 3 1 valid
Max-Pooling Layer 1	Window Size	3 x 3	2 x 2	2 x 2
Convolutional Layer 2	Feature Maps Filter Size Stride Padding	10 5 x 5 1 valid	20 5 x 5 1 valid	64 3 x 3 1 valid
Max-Pooling Layer 2	Window Size	3 x 3	2 x 2	3 x 3
Fully Connected Layer 1	No. of neurons	40	40	70
Fully Connected Layer 2	No. of neurons	-	20	40
Fully Connected Layer 3	No. of neurons	-	-	15
Model Parameters		137,111	1,470,245	3,879,701

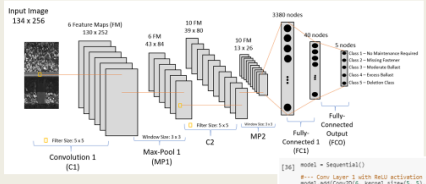
26

Training the Models

- Three datasets: Affine, Elastic, Mixed…
- … were trained on three architectures: Small, Medium, Large…
- … for a total of 9 experiments
- Data originally split into 80% training (8,000 instances) and 20% testing (2,000) instances.

30

Small CNN architecture

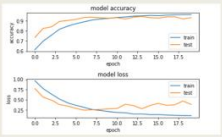


27

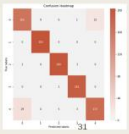
Initial Experiment Results

Dataset	Architecture	Train Accuracy	Train Loss	Test Accuracy	Test Loss	Computation time/epoch
Affine	Small	66.62%	0.6921	73.30%	0.6911	7 seconds
	Medium	82.58%	0.4015	80.30%	0.6129	12 seconds
	Large	91.81%	0.2333	82.10%	0.848	19 seconds
Elastic	Small	80.44%	0.5166	84.20%	0.4663	7 seconds
	Medium	95.98%	0.1084	93.10%	0.3867	12 seconds
	Large	93.58%	0.1469	92.90%	0.3762	19 seconds
Mixed	Small	78.87%	0.5103	78.30%	0.5293	7 seconds
	Medium	92.66%	0.1745	84.70%	0.5938	12 seconds
	Large	93.99%	0.1516	83.70%	0.8614	19 seconds

Model accuracy and loss

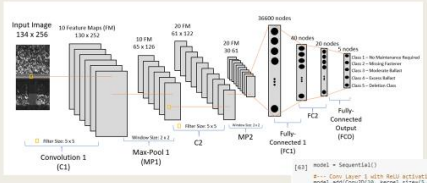


Model Confusion matrix



31

Medium CNN Architecture



28

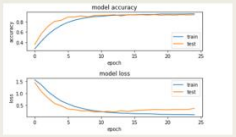
Issue with the Experiment

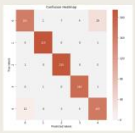
- The models that were trained tend to overfit the training data, i.e. it began to lose its generalization capabilities.
- To mitigate this, the experiment was repeated with the introduction of a Validation set as well as an implementation of 'Early Stopping'
- The dataset was now split into 70% training (7,000 instances), 20% testing (2,000 instances) and 10% validation (1,000) instances.

32

New Experiment Results

Dataset	Architecture	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	Computation time/epoch
Affine	Small	78.00%	0.6999	74.00%	0.688	7 seconds
	Medium	79.00%	0.8591	76.80%	0.914	12 seconds
	Large	80.67%	0.9018	80.60%	1.203	19 seconds
Elastic	Small	86.22%	0.3781	88.50%	0.3851	7 seconds
	Medium	94.22%	0.1827	93.40%	0.3778	12 seconds
	Large	93.57%	0.5161	90.80%	0.6477	19 seconds
Mixed	Small	84.11%	0.477	81.10%	0.492	7 seconds
	Medium	86.78%	0.6468	82.40%	0.9882	12 seconds
	Large	85.89%	0.5717	84.50%	0.7501	19 seconds

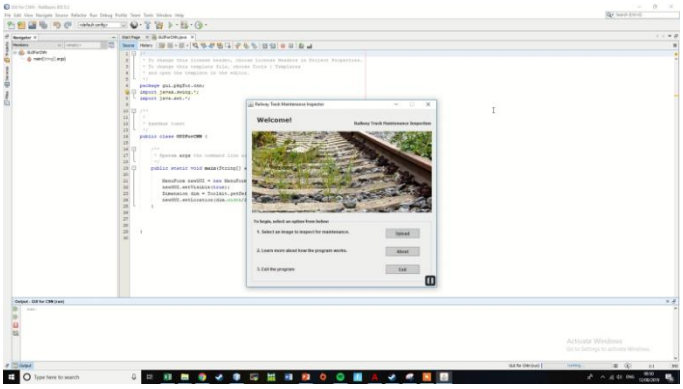




33

GUI Demonstration

34



Appendix D – Certificate of Ethics Approval

Coventry University

Certificate of Ethical Approval

Applicant:

Thomas Staite

Project Title:

Development of Convolutional Neural Network for the Detection of Railway Track Maintenance

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

19 June 2019

Project Reference Number:

P91330

Appendix E – CNN Experiment Code

Convolutional Neural Networks Experimentation

Mount Google Drive (for access to images)

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fw

Enter your authorization code:
··········
Mounted at /content/drive
```

Import Useful Libraries

```
[ ] # import statemnts
    import cv2
    import numpy as np
    import keras
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from keras.models import Sequential
    from keras.layers.convolutional import Conv2D
    from keras.layers import Dense
    from keras.layers.convolutional import MaxPooling2D
    from keras.layers import Flatten
    from keras.layers import Dropout
    from keras.callbacks import EarlyStopping
    from keras.callbacks import ModelCheckpoint
    from sklearn.metrics import confusion_matrix
```

Set Learning Hyper-Parameters

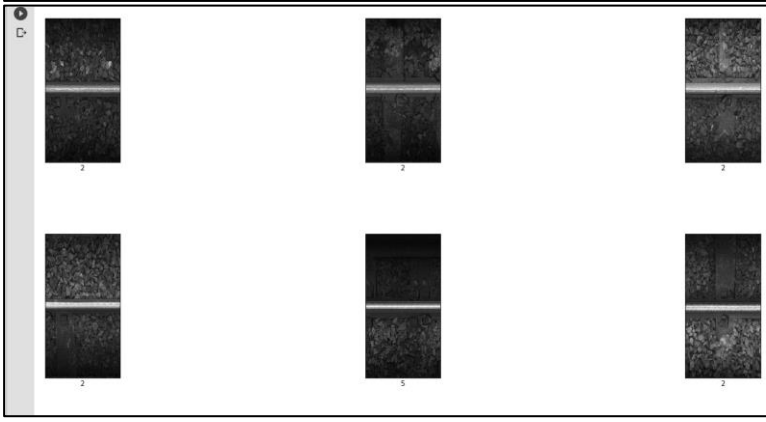
```
[ ] batch_size = 20
    num_epoch = 30
    n_classes = 5
    dropout = 0.75
```

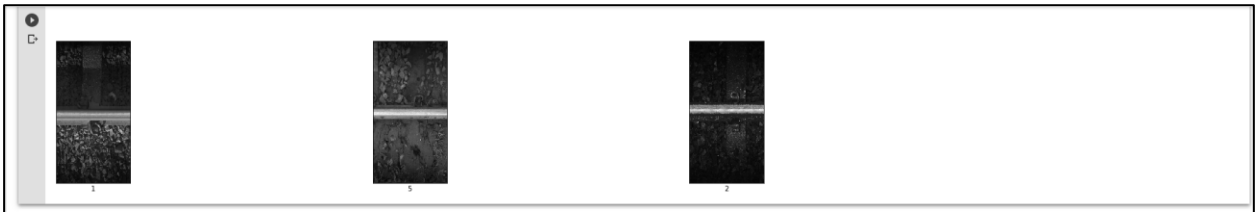
Import Dataset From Google Drive (Either Affine, Elastic, or Mixed)

```
[ ] data = []
    labels = []

    for i in range(1,6):
      for j in range(1,2001):
        img  = np.array(cv2.imread('/content/drive/My Drive/Thomas Staite - Convolutional Neural Network/NEW DATASETS/AffineAndElastic/Class '+str(i)+'/Class'+str(i)+'Image'+str(j)+'.png', 0))
        data.append(img.flatten())
        labels.append(i)
        if j % 100 == 0:
          print('class:', i, 'file step:', j)


    data = np.array(data)
    labels = np.array(labels)
    print(data[0])
    print(labels)
```

Display a random selection of the dataset

```
[ ] def display_images(batchSize, data, labels):
      idx = np.arange(0, len(data))
      # Shuffle the order of the data and select an amount to show
      np.random.shuffle(idx)
      idx = idx[: batchSize]
      data_shuffle = [data[i] for i in idx]
      labels_shuffle = [labels[i] for i in idx]
      x = data_shuffle
      y = labels_shuffle
      # Plot the images using MatPlotLib.pyPlot
      fig, axes = plt.subplots(3, 3)
      fig.set_size_inches(22.5, 15.5)
      fig.subplots_adjust(hspace = 0.5, wspace = 0.5)
      for i, ax in enumerate(axes.flat):
        ax.imshow(x[i].reshape(256,134), cmap='gray')
        ax.set_xlabel(int(y[i]))
        ax.set_xticks([])
        ax.set_yticks([])
      plt.show()

[ ] display_images(9, data, labels)
```





Function to split the dataset into training and testing

```
def splitData(Xvals, Yvals):

    idx = np.arange(0, len(Xvals))
    # Shuffle the order of the data to preperare it for splitting
    np.random.shuffle(idx)
    Xvals = [Xvals[i] for i in idx]
    Yvals = [Yvals[i] for i in idx]

    Xvals = np.array(Xvals)
    Yvals = np.array(Yvals)
    # Split the data into 90% training and 10% testing
    X_train, X_test, y_train, y_test = train_test_split(Xvals, Yvals, test_size=0.1, random_state=50)

    print('--- Training Data ---')
    print('No. instances X_train:', X_train.shape)
    print('No. instances y_train:', y_train.shape)
    print('')
    print('--- Testing Data ---')
    print('No. instances X_test:', X_test.shape)
    print('No. instances y_test:', y_test.shape)

    y_train = y_train -1
    y_test = y_test -1
    print(y_train)

    return(X_train, X_test, y_train, y_test)
```

```
# Call function to split the data
X_train, X_test, y_train, y_test = splitData(data, labels)


# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, n_classes)
y_test = keras.utils.to_categorical(y_test, n_classes)
```

```
--- Training Data ---
No. instances X_train: (9000, 34304)
No. instances y_train: (9000,)

--- Testing Data ---
No. instances X_test: (1000, 34304)
No. instances y_test: (1000,)
[1 3 1 ... 0 0 1]
```

Reshape the data into the format accepted by CNN

```
img_rows = 134
img_cols = 256


X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)

# Normalization will help to remove distortions caused by lights and shadows in an image.
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print('X_train shape:', X_train.shape)
```

```
X_train shape: (9000, 134, 256, 1)
```

CNN Architecture 1 - Small

```
model = Sequential()

#--- Conv Layer 1 with ReLU activation ---
model.add(Conv2D(6, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
#--- MaxPool Layer 1 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(3, 3)))
#--- Conv Layer 2 with ReLU activation ---
model.add(Conv2D(10, (5, 5), activation='relu'))
#--- MaxPool Layer 2 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(3, 3)))
#randomly turn neurons on and off to improve convergence
model.add(Dropout(0.25))
#flatten reshape and prepare data for FC layers
model.add(Flatten())
#--- Fully Connected Layer 1 with ReLU activation ---
model.add(Dense(40, activation='relu'))
#one more dropout for convergence' sake
model.add(Dropout(0.5))
#--- Fully Connected Output Layer with softmax to squash the matrix into output probabilities
model.add(Dense(n_classes, activation='softmax'))
```

Model Summary

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 130, 252, 6)       156
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 43, 84, 6)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 39, 80, 10)        1510
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 13, 26, 10)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 13, 26, 10)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 3380)              0
_________________________________________________________________
dense_11 (Dense)             (None, 40)                135240
_________________________________________________________________
dropout_8 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_12 (Dense)             (None, 5)                 205
=================================================================
Total params: 137,111
Trainable params: 137,111
Non-trainable params: 0
_________________________________________________________________
```

CNN Architecture 2 - Medium

```
model = Sequential()

#--- Conv Layer 1 with ReLU activation ---
model.add(Conv2D(10, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
#--- MaxPool Layer 1 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(2, 2)))
#--- Conv Layer 2 with ReLU activation ---
model.add(Conv2D(20, (5, 5), activation='relu'))
#--- MaxPool Layer 2 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(2, 2)))
#randomly turn neurons on and off to improve convergence
model.add(Dropout(0.25))
#flatten reshape and prepare data for FC layers
model.add(Flatten())
#--- Fully Connected Layer 1 with ReLU activation ---
model.add(Dense(40, activation='relu'))
#--- Fully Connected Layer 2 with ReLU activation ---
model.add(Dense(20, activation='relu'))
#one more dropout for convergence' sake
model.add(Dropout(0.5))
#--- Fully Connected Output Layer with softmax to squash the matrix into output probabilities
model.add(Dense(n_classes, activation='softmax'))
```

Model Summary

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 130, 252, 10)      260
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 65, 126, 10)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 61, 122, 20)       5020
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 30, 61, 20)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 30, 61, 20)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 36600)             0
_________________________________________________________________
dense_13 (Dense)             (None, 40)                1464040
_________________________________________________________________
dense_14 (Dense)             (None, 20)                820
_________________________________________________________________
dropout_10 (Dropout)         (None, 20)                0
_________________________________________________________________
dense_15 (Dense)             (None, 5)                 105
=================================================================
Total params: 1,470,245
Trainable params: 1,470,245
Non-trainable params: 0
_________________________________________________________________
```

CNN Architecture 3 - Large

```
model = Sequential()

#--- Conv Layer 1 with ReLU activation ---
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
#--- MaxPool Layer 1 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(2, 2)))
#--- Conv Layer 2 with ReLU activation ---
model.add(Conv2D(64, (3, 3), activation='relu'))
#--- MaxPool Layer 2 for dimension reduction and feature selection ---
model.add(MaxPooling2D(pool_size=(3, 3)))
#randomly turn neurons on and off to improve convergence
model.add(Dropout(0.25))
#flatten reshape and prepare data for FC layers
model.add(Flatten())
#--- Fully Connected Layer 1 with ReLU activation ---
model.add(Dense(70, activation='relu'))
#--- Fully Connected Layer 2 with ReLU activation ---
model.add(Dense(40, activation='relu'))
#--- Fully Connected Layer 3 with ReLU activation ---
model.add(Dense(15, activation='relu'))
#one more dropout for convergence' sake
model.add(Dropout(0.5))
#--- Fully Connected Output Layer with softmax to squash the matrix into output probabilities
model.add(Dense(n_classes, activation='softmax'))
```

Model Summary

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 132, 254, 32)      320
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 66, 127, 32)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 64, 125, 64)       18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 21, 41, 64)        0
_________________________________________________________________
dropout_11 (Dropout)         (None, 21, 41, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 55104)             0
_________________________________________________________________
dense_16 (Dense)             (None, 70)                3857350
_________________________________________________________________
dense_17 (Dense)             (None, 40)                2840
_________________________________________________________________
dense_18 (Dense)             (None, 15)                615
_________________________________________________________________
dropout_12 (Dropout)         (None, 15)                0
_________________________________________________________________
dense_19 (Dense)             (None, 5)                 80
=================================================================
Total params: 3,879,701
Trainable params: 3,879,701
Non-trainable params: 0
_________________________________________________________________
```

Compile Model, define loss function, optimizer, and performance metrics

```
[ ] #Adaptive learning rate (adaDelta) is a form of gradient descent
    #categorical since we have multiple classes (5)
    model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])

    # Implement Early Stopping to prevent overfitting
    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)

    # Implement Model Checkpoint to find the epoch that provided best model
    mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
```

Train the model on the dataset

```
[ ] #model training
    model_log = model.fit(X_train, y_train, batch_size=batch_size, epochs=num_epoch, verbose=1, validation_split=0.1, callbacks=[es, mc])
```

```
Epoch 00016: val_acc did not improve from 0.83333
Epoch 17/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.4038 - acc: 0.8190 - val_loss: 0.5703 - val_acc: 0.8422

Epoch 00017: val_acc improved from 0.83333 to 0.84222, saving model to best_model.h5
Epoch 18/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3861 - acc: 0.8248 - val_loss: 0.5810 - val_acc: 0.8400

Epoch 00018: val_acc did not improve from 0.84222
Epoch 19/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3750 - acc: 0.8337 - val_loss: 0.5758 - val_acc: 0.8322

Epoch 00019: val_acc did not improve from 0.84222
Epoch 20/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3513 - acc: 0.8437 - val_loss: 0.5973 - val_acc: 0.8444

Epoch 00020: val_acc improved from 0.84222 to 0.84444, saving model to best_model.h5
Epoch 21/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3429 - acc: 0.8542 - val_loss: 0.7064 - val_acc: 0.8200

Epoch 00021: val_acc did not improve from 0.84444
Epoch 22/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3288 - acc: 0.8584 - val_loss: 0.6871 - val_acc: 0.8167

Epoch 00022: val_acc did not improve from 0.84444
Epoch 23/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3105 - acc: 0.8693 - val_loss: 0.6981 - val_acc: 0.8267

Epoch 00023: val_acc did not improve from 0.84444
Epoch 24/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.3162 - acc: 0.8748 - val_loss: 0.6745 - val_acc: 0.8489

Epoch 00024: val_acc improved from 0.84444 to 0.84889, saving model to best_model.h5
Epoch 25/30
8100/8100 [==============================] - 19s 2ms/step - loss: 0.2965 - acc: 0.8825 - val_loss: 0.7841 - val_acc: 0.8389
```

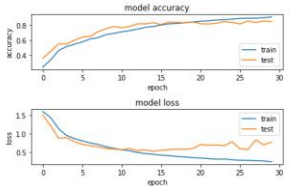
Print final test loss and test accuracy

```
[ ] score = model.evaluate(X_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
```

```
Test loss: 0.7500581729412079
Test accuracy: 0.845
```

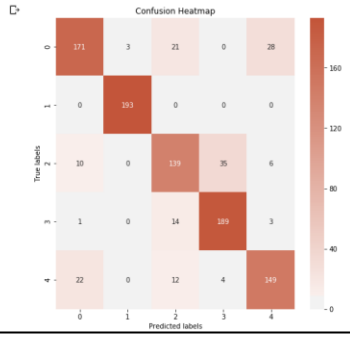
Plot model accuracy and model loss graphs

```
[ ] import os
    # plotting the metrics
    fig = plt.figure()
    plt.subplot(2,1,1)
    plt.plot(model_log.history['acc'])
    plt.plot(model_log.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='lower right')
    plt.subplot(2,1,2)
    plt.plot(model_log.history['loss'])
    plt.plot(model_log.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper right')
    plt.tight_layout()
    fig
```



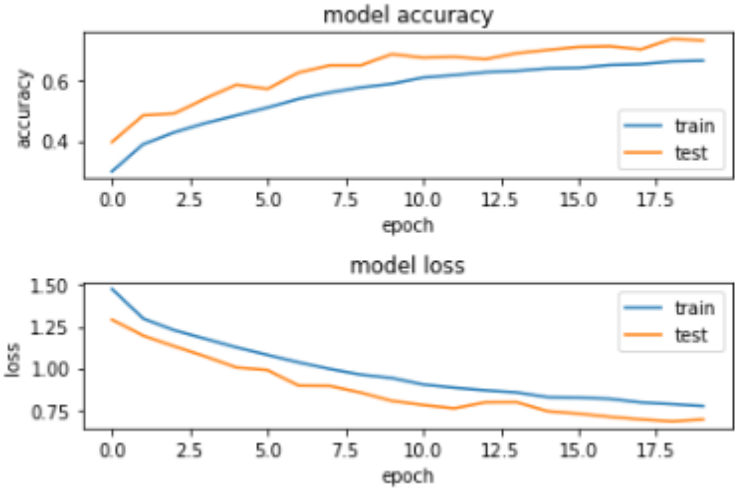
Plot Confusion Matrix Heatmap

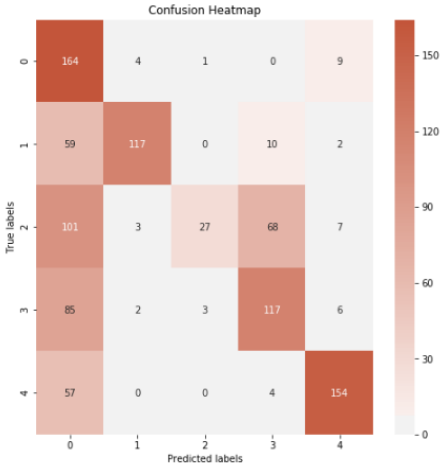
```
[ ] predictions = model.predict(X_test)
    y_pred = (predictions > 0.5)
    matrix = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))

    fig, ax = plt.subplots( figsize=(8,8))
    #ax= plt.subplot()
    sns.heatmap(matrix, annot=True, center=True,cmap=sns.diverging_palette(220,20,as_cmap=True), fmt='d', ax = ax);
    # labels, title and ticks
    ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
    ax.set_title('Confusion Heatmap');
```



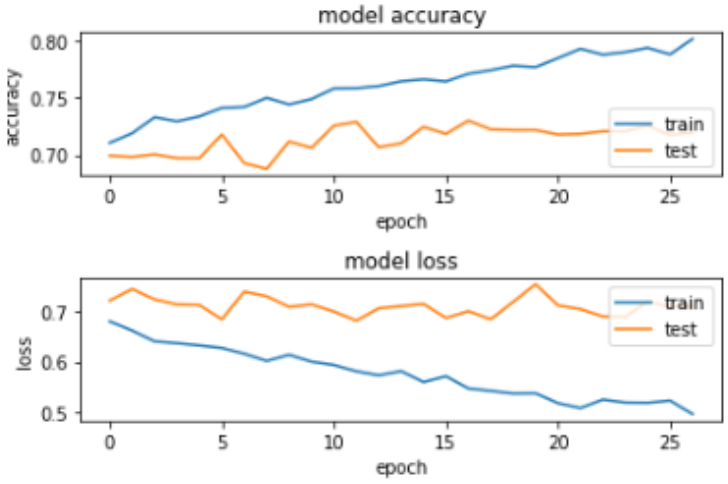
Appendix F – Experiment Results

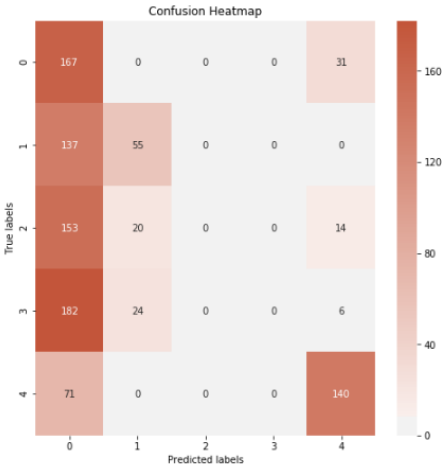
Affine Dataset – Small CNN architecture – Experiment 1



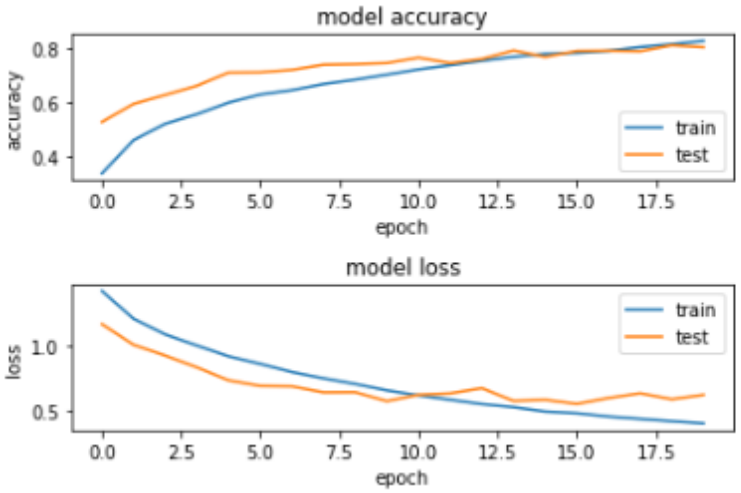


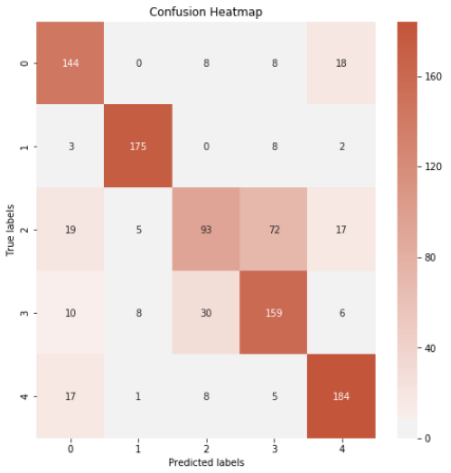
Affine Dataset – Small CNN architecture – Experiment 2



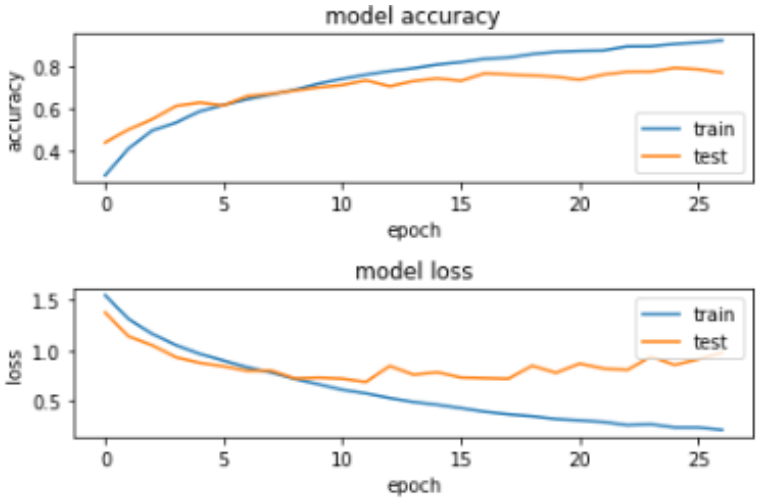


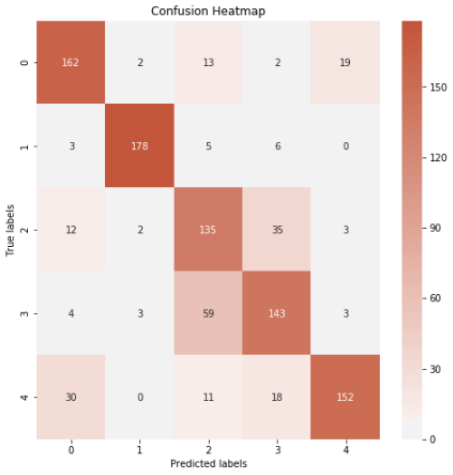
Affine Dataset – Medium CNN architecture – Experiment 1



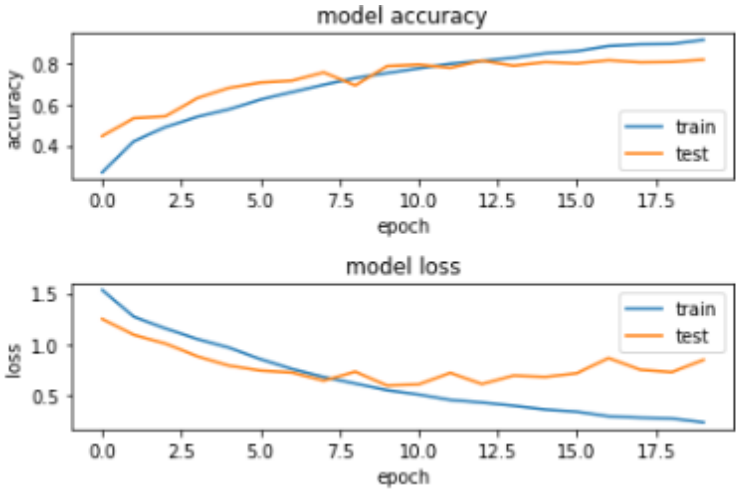


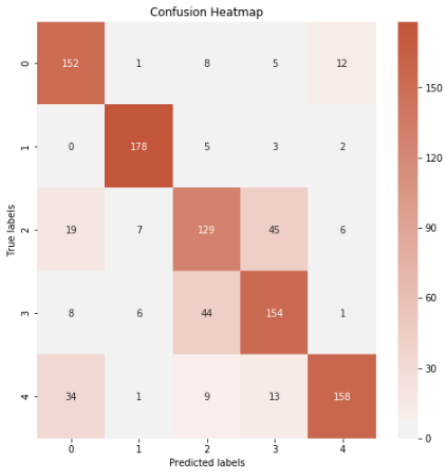
Affine Dataset – Medium CNN architecture – Experiment 2



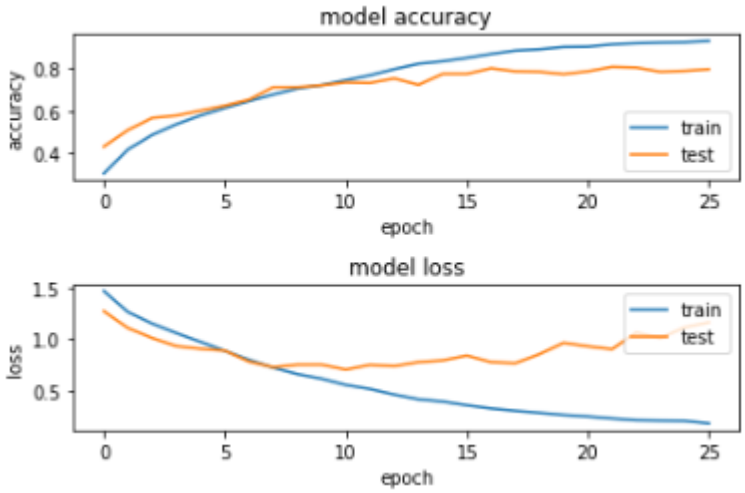


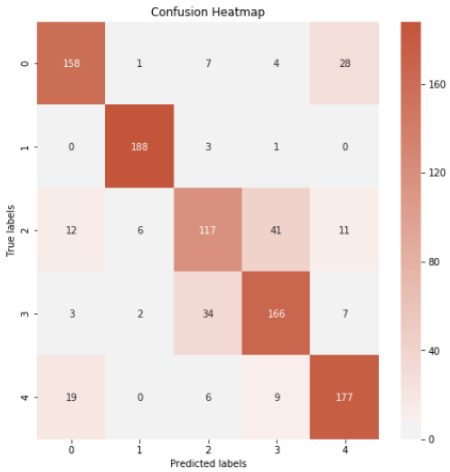
Affine Dataset – Large CNN architecture – Experiment 1



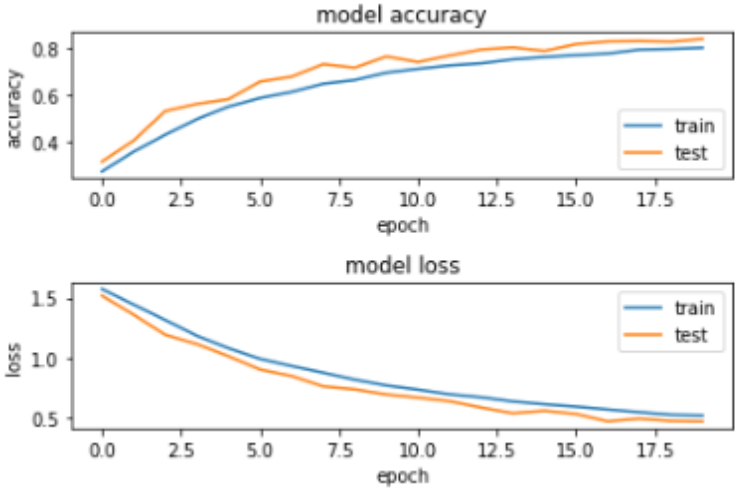


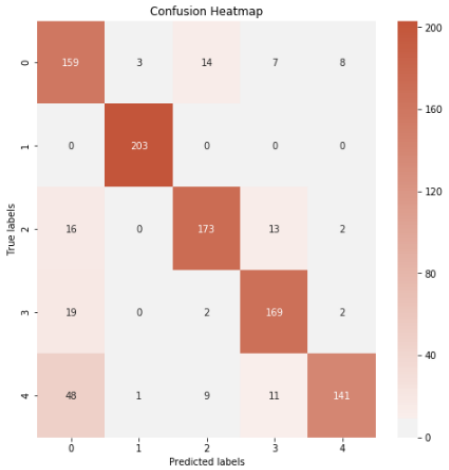
Affine Dataset – Large CNN architecture – Experiment 2



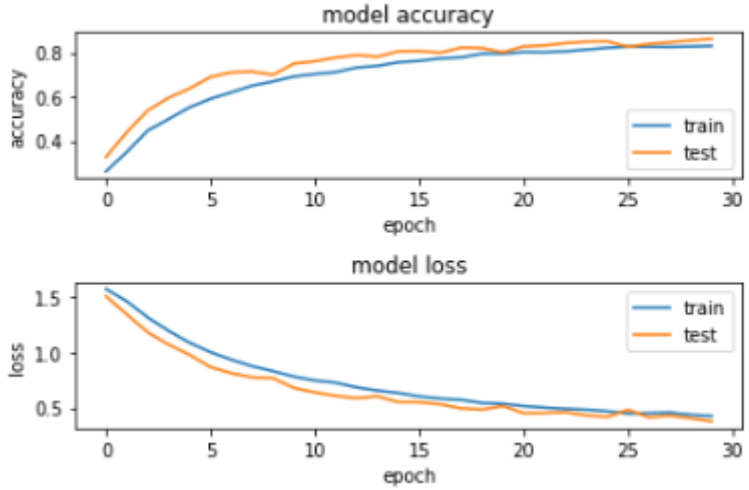


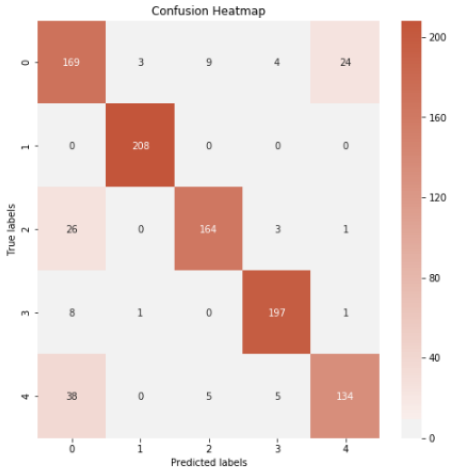
Elastic Dataset – Small CNN architecture – Experiment 1



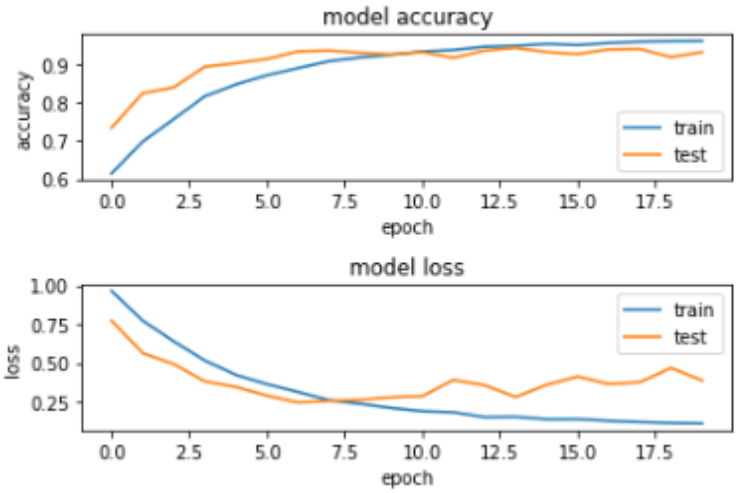


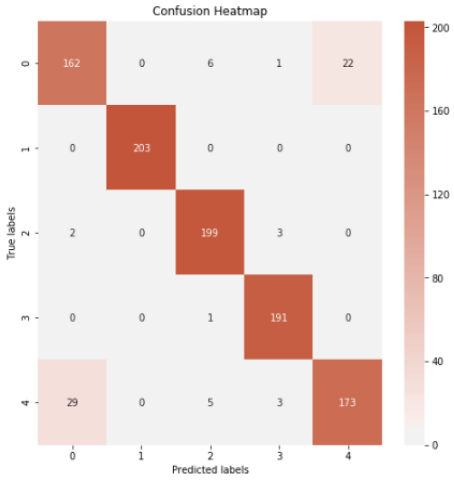
Elastic Dataset – Small CNN architecture – Experiment 2



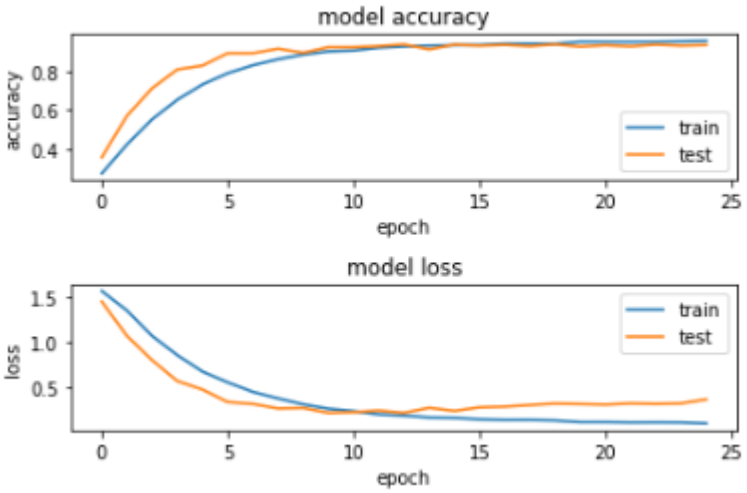


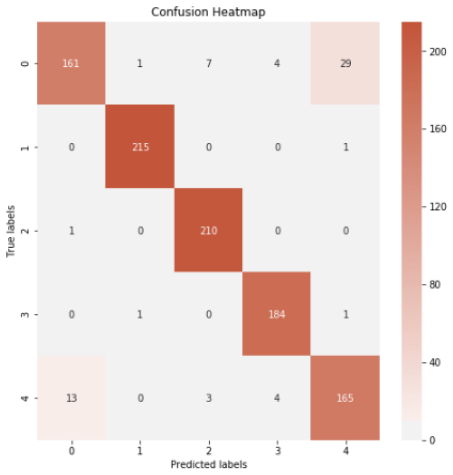
Elastic Dataset – Medium CNN architecture – Experiment 1



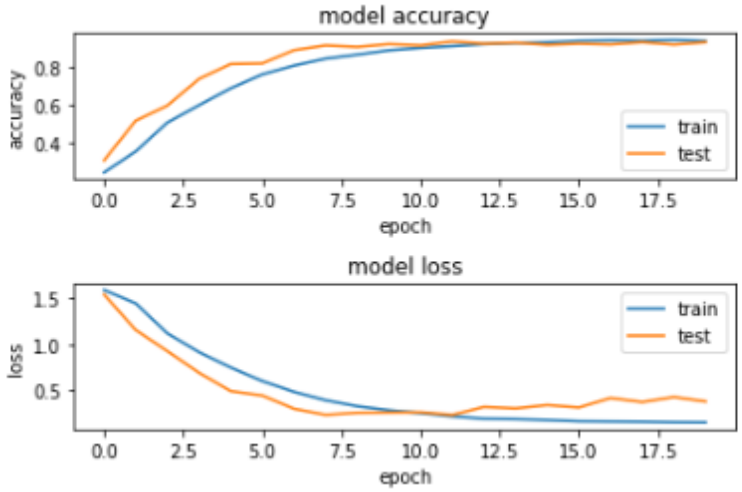


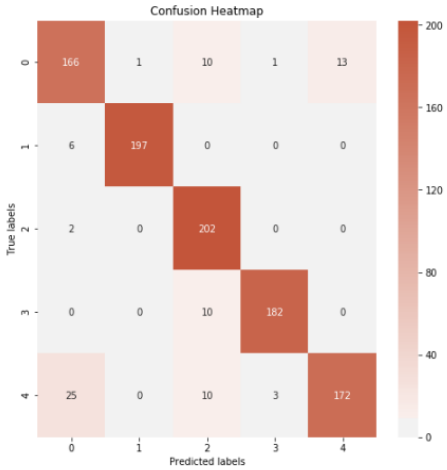
Elastic Dataset – Medium CNN architecture – Experiment 2



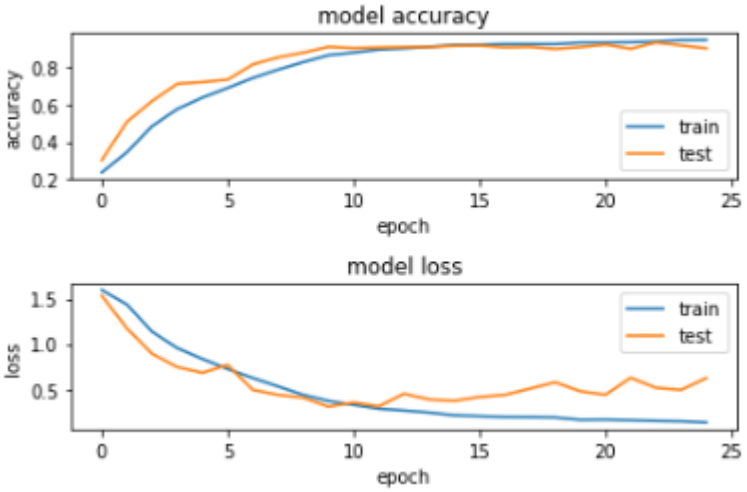


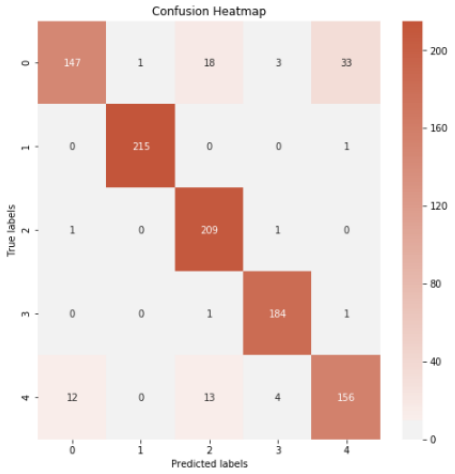
Elastic Dataset – Large CNN architecture – Experiment 1



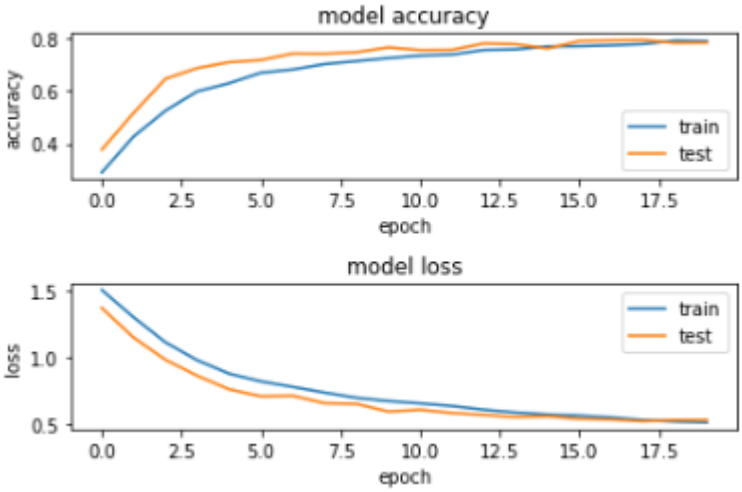


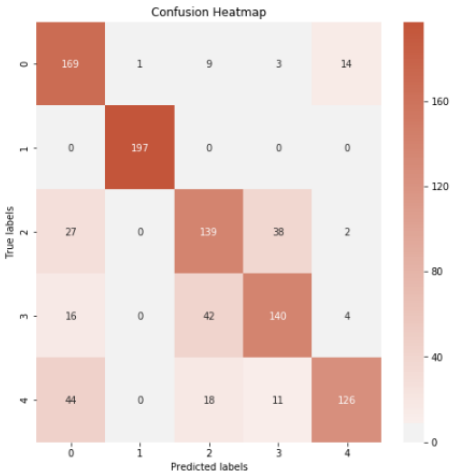
Elastic Dataset – Large CNN architecture – Experiment 2



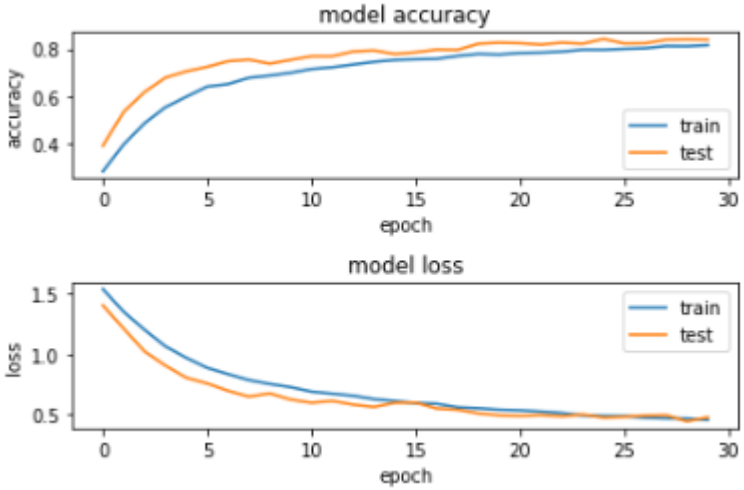


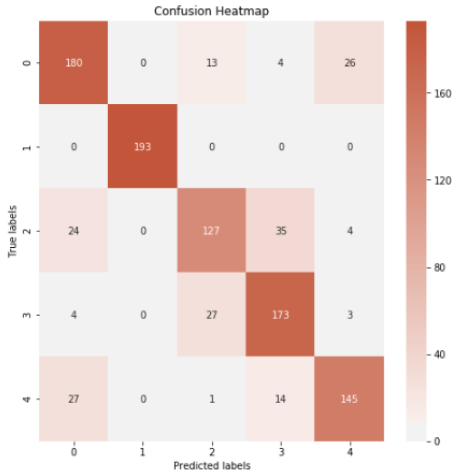
Mixed Dataset – Small CNN architecture – Experiment 1



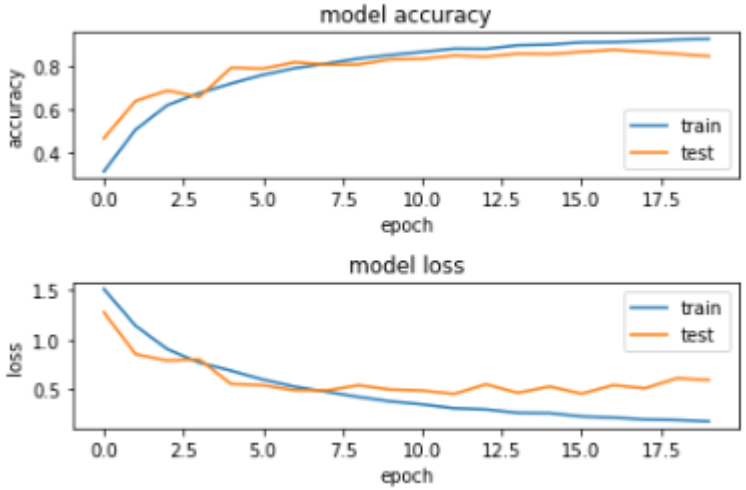


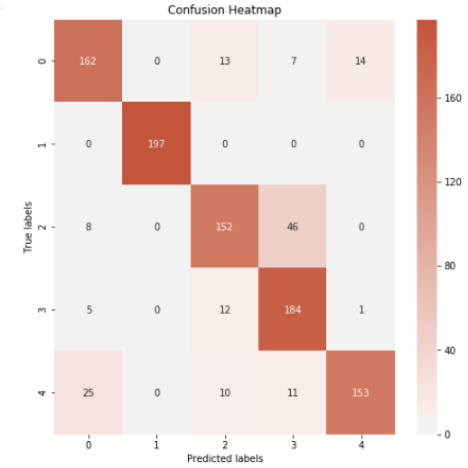
Mixed Dataset – Small CNN architecture – Experiment 2



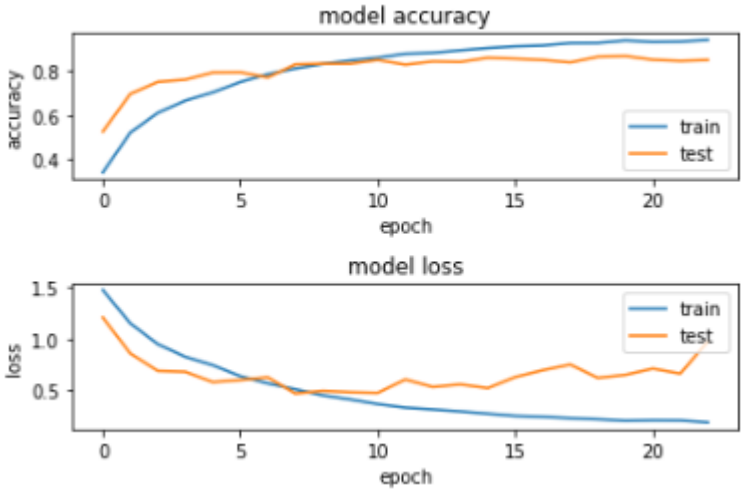


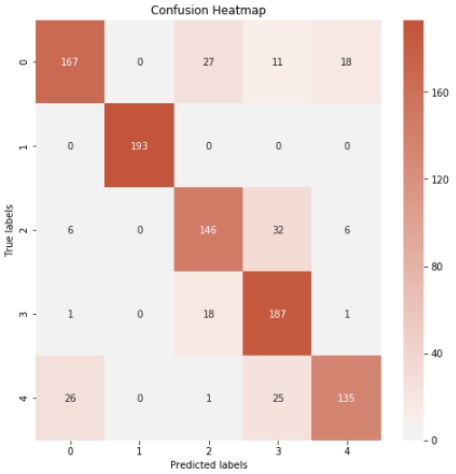
Mixed Dataset – Medium CNN architecture – Experiment 1



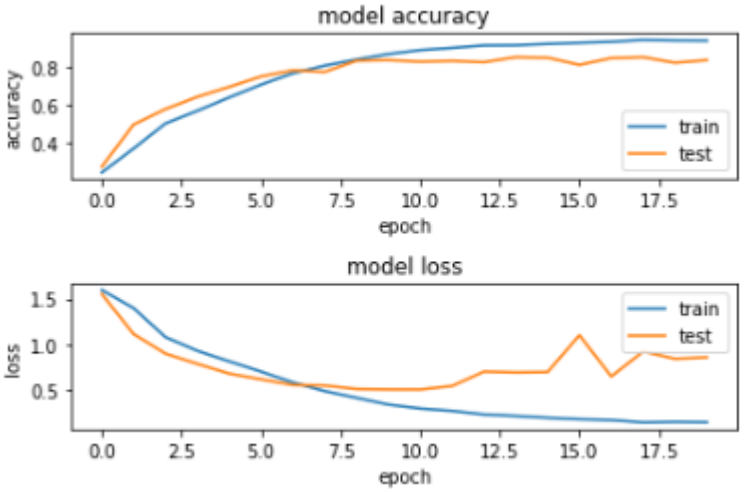


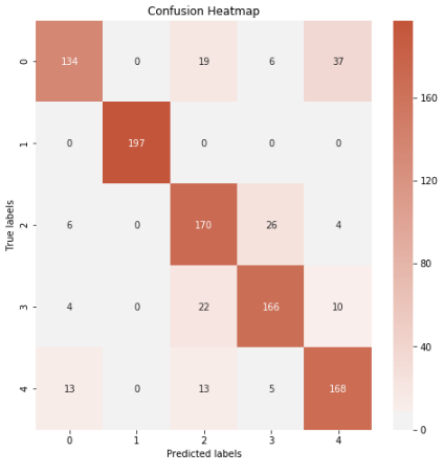
Mixed Dataset – Medium CNN architecture – Experiment 2



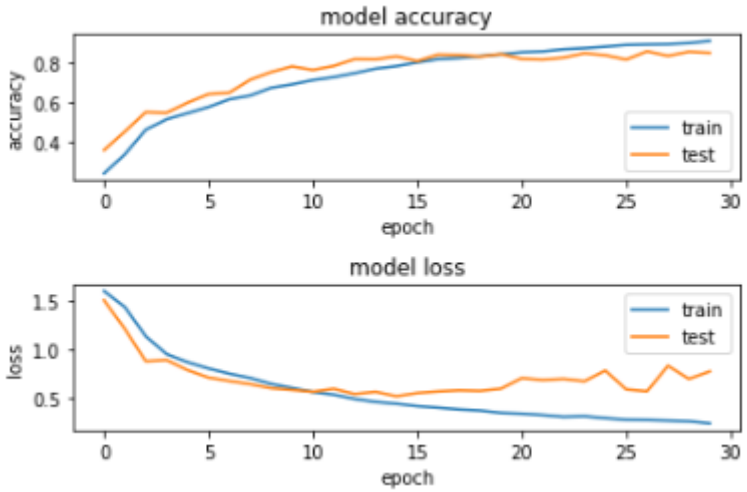


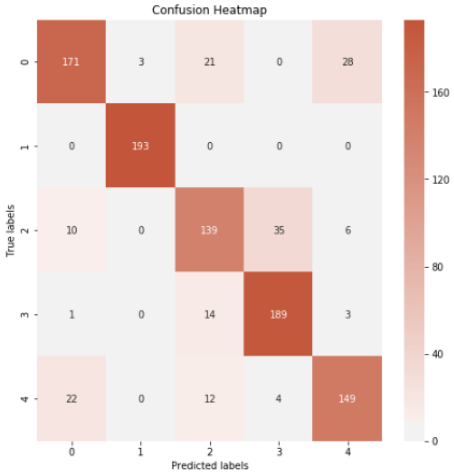
Mixed Dataset – Large CNN architecture – Experiment 1





Mixed Dataset – Large CNN architecture – Experiment 2





Appendix G – Java Application Code

MenuForm.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package gui.pkgfor.cnn;
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.FilenameFilter;

//import java.awt.event.*;
//import javax.swing.GroupLayout;
import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.filechooser.FileNameExtensionFilter;



/**
 *
 * @author tomst
 */
public class MenuForm extends javax.swing.JFrame {
    public File newFile;

    /**
     * Creates new form addPhotosForm
     */
    public MenuForm() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

```java
    private void exitMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        System.exit(0);
    }

    private void exitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void aboutMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        AboutForm newAboutForm = new AboutForm();
        newAboutForm.setVisible(true);
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        newAboutForm.setLocation(dim.width/2-newAboutForm.getSize().width/2, dim.height/2-newAboutForm.getSize().height/2);
    }

    public void setFile(File newFile) {
        this.newFile = this.newFile;
    }

     public File getFile() {
        return newFile;
    }
```

```java
    private void selectImageMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        JFileChooser fileChooser = new JFileChooser();


        FileNameExtensionFilter imageFilter = new FileNameExtensionFilter("Image files", ImageIO.getReaderFileSuffixes());


        fileChooser.addChoosableFileFilter(imageFilter);
        fileChooser.setAcceptAllFileFilterUsed(false);
        int returnValue = fileChooser.showOpenDialog(null);
        if (returnValue == JFileChooser.APPROVE_OPTION)
        {
            File selectedFile = fileChooser.getSelectedFile();
            setFile(selectedFile);
            resultGUI newResultGUI = new resultGUI();
            try {
                newResultGUI.setNewFile(selectedFile);
            } catch (IOException ex) {
                Logger.getLogger(MenuForm.class.getName()).log(Level.SEVERE, null, ex);
            }
            newResultGUI.setVisible(true);
            Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
            newResultGUI.setLocation(dim.width/2-newResultGUI.getSize().width/2, dim.height/2-newResultGUI.getSize().height/2);


        }
    }
```

```
259
260     /**
261      * @param args the command line arguments
262      */
263     public static void main(String args[]) {
264         /* Set the Nimbus look and feel */
265         Look and feel setting code (optional)
286         //</editor-fold>
287
288         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
291                 new MenuForm().setVisible(true);
292
293             }
294
295         });
296     }
297
298     // Variables declaration - do not modify
299     private javax.swing.JButton about;
300     private javax.swing.JButton exit;
301     private javax.swing.JLabel exitTheProgram;
302     private javax.swing.JLabel learnMoreAbout;
303     private javax.swing.JLabel menuImage;
304     private javax.swing.JPanel panelForButtons;
305     private javax.swing.JPanel panelForImage;
306     private javax.swing.JPanel panelForOptions;
307     private javax.swing.JLabel railwayTrackMainText;
308     private javax.swing.JButton selectImage;
309     private javax.swing.JLabel selectImageToInspect;
310     private javax.swing.JLabel welcomeText;
311     // End of variables declaration
312 }
```

AboutForm.java

```
6   package gui.pkgfor.cnn;
7
8   import java.awt.Dimension;
    import java.awt.Toolkit;
10
11  /**
12   *
13   * @author tomst
14   */
15  public class AboutForm extends javax.swing.JFrame {
16
17      /**
18       * Creates new form aboutForm
19       */
20      public AboutForm() {
21          initComponents();
22      }
23
24      /**
25       * This method is called from within the constructor to initialize the form.
26       * WARNING: Do NOT modify this code. The content of this method is always
27       * regenerated by the Form Editor.
28       */
29      @SuppressWarnings("unchecked")
30      Generated Code
106
107     private void formWindowClosed(java.awt.event.WindowEvent evt) {
108         // TODO add your handling code here:
109         setDefaultCloseOperation(AboutForm.DISPOSE_ON_CLOSE);
110     }
111
112     private void formWindowClosing(java.awt.event.WindowEvent evt) {
113         // TODO add your handling code here:
114         setDefaultCloseOperation(AboutForm.DISPOSE_ON_CLOSE);
115     }
116
117     private void closeAboutMouseClicked(java.awt.event.MouseEvent evt) {
118         // TODO add your handling code here:
119         super.dispose();
120     }
121
```

```
122     /**
123      * @param args the command line arguments
124      */
125     public static void main(String args[]) {
126         /* Set the Nimbus look and feel */
127         Look and feel setting code (optional)
148         //</editor-fold>
149
150         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
153                 new AboutForm().setVisible(true);
154
155             }
156         });
157     }
158
159     // Variables declaration - do not modify
160     private javax.swing.JLabel aboutTheSoftwareTitle;
161     private javax.swing.JButton closeAbout;
162     private javax.swing.JScrollPane jScrollPane1;
163     private javax.swing.JTextArea jTextArea1;
164     // End of variables declaration
165 }
166
```

GUIforCNN.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package gui.pkgfor.cnn;
7  import javax.swing.*;
8  import java.awt.*;
9
10 /**
11  *
12  * @author tomst
13  */
14 public class GUIForCNN {
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20
21         MenuForm newGUI = new MenuForm();
22         newGUI.setVisible(true);
23         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
24         newGUI.setLocation(dim.width/2-newGUI.getSize().width/2, dim.height/2-newGUI.getSize().height/2);
25     }
26
27
28
29 }
30
```

resultGUI.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package gui.pkgfor.cnn;
7
8  import java.awt.Graphics2D;
9  import java.awt.Image;
10 import java.awt.image.BufferedImage;
11 import java.io.BufferedReader;
12 import java.io.File;
13 import java.io.IOException;
14 import java.io.InputStreamReader;
15 import java.nio.file.Path;
16 import java.nio.file.Paths;
17 import java.util.Arrays;
18 import javax.imageio.ImageIO;
19 import javax.swing.ImageIcon;
20
21 /**
22  *
23  * @author tomst
24  */
25 public class resultGUI extends javax.swing.JFrame {
26
27     private File fileToInspect;
28     private String fileName;
29     private String predictions;
30     private String[] predictionArray;
31
32
```

```
32
33      private void runPythonModel() throws IOException{
34
35          try {
36          String s = null;
37          Process p = Runtime.getRuntime().exec("python -W ignore C:/Users/tomst/Desktop/dataPrepForFinalCNN.py \""+fileName+"\"");
38          //Process p = Runtime.getRuntime().exec("python C:/Users/tomst/Desktop/hello.py");
39
40          BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
41
42          while ((s = in.readLine()) != null){
43              //System.out.println(s);
44              predictions = s;
45              System.out.println(predictions);
46              splitPredictions();
47          }
48      }
49          catch(IOException ie){
50              ie.printStackTrace();
51          }
52
53      }
54
55
56      public static BufferedImage cropImage(BufferedImage bufferedImage, int x, int y, int width, int height){
57      BufferedImage croppedImage = bufferedImage.getSubimage(x, y, width, height);
58      return croppedImage;
59  }
60
61
```

```
62      public static BufferedImage toBufferedImage(Image img)
63  {
64      if (img instanceof BufferedImage)
65      {
66          return (BufferedImage) img;
67      }
68
69      // Create a buffered image with transparency
70      BufferedImage bimage = new BufferedImage(img.getWidth(null), img.getHeight(null), BufferedImage.TYPE_INT_ARGB);
71
72      // Draw the image on to the buffered image
73      Graphics2D bGr = bimage.createGraphics();
74      bGr.drawImage(img, 0, 0, null);
75      bGr.dispose();
76
77      // Return the buffered image
78      return bimage;
79  }
80
```

```
82      private void splitPredictions(){
83          System.out.println(predictions);
84          predictions = predictions.substring(1, predictions.length()-1);
85          predictionArray = predictions.split(", ");
86
87          image1Class.setText(predictionArray[0]);
88          image2Class.setText(predictionArray[1]);
89          image3Class.setText(predictionArray[2]);
90          image4Class.setText(predictionArray[3]);
91          image5Class.setText(predictionArray[4]);
92          image6Class.setText(predictionArray[5]);
93
94
95          ImageIcon iconLogo = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage0.png");
96          Image image = iconLogo.getImage(); // transform it
97          Image newimg = image.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
98          iconLogo = new ImageIcon(newimg);  // transform it back
99          imageSplit1.setIcon(iconLogo);
100
101
102         ImageIcon iconLogo2 = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage1.png");
103         Image image2 = iconLogo2.getImage(); // transform it
104         Image newimg2 = image2.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
105         iconLogo2 = new ImageIcon(newimg2);  // transform it back
106         imageSplit2.setIcon(iconLogo2);
107
108         ImageIcon iconLogo3 = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage2.png");
109         Image image3 = iconLogo3.getImage(); // transform it
110         Image newimg3 = image3.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
111         iconLogo3 = new ImageIcon(newimg3);  // transform it back
112         imageSplit3.setIcon(iconLogo3);
113
114         ImageIcon iconLogo4 = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage3.png");
115         Image image4 = iconLogo4.getImage(); // transform it
116         Image newimg4 = image4.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
117         iconLogo4 = new ImageIcon(newimg4);  // transform it back
118         imageSplit4.setIcon(iconLogo4);
119
120         ImageIcon iconLogo5 = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage4.png");
121         Image image5 = iconLogo5.getImage(); // transform it
122         Image newimg5 = image5.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
123         iconLogo5 = new ImageIcon(newimg5);  // transform it back
124         imageSplit5.setIcon(iconLogo5);
```

```
126         ImageIcon iconLogo6 = new ImageIcon("C:\\Users\\tomst\\OneDrive\\Documents\\NetBeansProjects\\GUI for CNN\\croppedImage5.png");
127         Image image6 = iconLogo6.getImage(); // transform it
128         Image newimg6 = image6.getScaledInstance(172, 332,  java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
129         iconLogo6 = new ImageIcon(newimg6);  // transform it back
130         imageSplit6.setIcon(iconLogo6);
131
132     }
133
134
135     public void setNewFile(File fileToInspect) throws IOException {
136         this.fileToInspect = this.fileToInspect;
137         this.fileName = fileToInspect.getPath();
138         System.out.println("Result GUI Image:");
139         System.out.println(fileToInspect.getPath());
140         runPythonModel();
141     }
142
143     /**
144      * Creates new form resultGUI
145      */
146     public resultGUI() {
147         initComponents();
148         //fileToInspect = getFile();
149     }
150
151     /**
152      * This method is called from within the constructor to initialize the form.
153      * WARNING: Do NOT modify this code. The content of this method is always
154      * regenerated by the Form Editor.
155      */
156     @SuppressWarnings("unchecked")
157     Generated Code
348
349     private void formWindowClosed(java.awt.event.WindowEvent evt) {
350         // TODO add your handling code here:
351         setDefaultCloseOperation(resultGUI.DISPOSE_ON_CLOSE);
352     }
353
354     private void formWindowClosing(java.awt.event.WindowEvent evt) {
355         // TODO add your handling code here:
356         setDefaultCloseOperation(resultGUI.DISPOSE_ON_CLOSE);
357     }
```

```
359     private void finishedButtonActionPerformed(java.awt.event.ActionEvent evt) {
360         // TODO add your handling code here:
361         super.dispose();
362     }
363
364     private void finishedButtonMouseClicked(java.awt.event.MouseEvent evt) {
365         // TODO add your handling code here:
366          super.dispose();
367     }
368
369     /**
370      * @param args the command line arguments
371      */
372     public static void main(String args[]) {
373         /* Set the Nimbus look and feel */
374         Look and feel setting code (optional)
399
400         /* Create and display the form */
401         java.awt.EventQueue.invokeLater(new Runnable() {
402             public void run() {
403                 new resultGUI().setVisible(true);
404             }
405         });
406
407
408     }
409
410     // Variables declaration - do not modify
411     private javax.swing.JButton finishedButton;
412     private javax.swing.JLabel image1Class;
413     private javax.swing.JLabel image2Class;
414     private javax.swing.JLabel image3Class;
415     private javax.swing.JLabel image4Class;
416     private javax.swing.JLabel image5Class;
417     private javax.swing.JLabel image6Class;
418     private javax.swing.JLabel imageSplit1;
419     private javax.swing.JLabel imageSplit2;
420     private javax.swing.JLabel imageSplit3;
421     private javax.swing.JLabel imageSplit4;
422     private javax.swing.JLabel imageSplit5;
423     private javax.swing.JLabel imageSplit6;
424     private javax.swing.JLayeredPane jLayeredPanel;
425     private javax.swing.JPanel jPanel1;
426     private javax.swing.JPanel jPanel2;
427     private javax.swing.JLabel yourResults;
428     // End of variables declaration
429 }
```