University of Southern Queensland

Faculty of Health, Engineering & Sciences

Asset Classification with Deep Learning from Computer Added Design Parts

A dissertation submitted by

Joshua Garry

in the fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Engineering Mechanical

Submitted: October, 2022

Abstract

In this project an industrial client focused method for asset classification is proposed; where the output of a machine learning algorithm can identify the CAD model corresponding to the subject of the image input. This method aims to use a per part identification system based on the client's own database as opposed to the norm of general categorical identification. Part of this study also aims to assess the feasibility of the image rendering process used to procure the proposed client machine learning dataset. This dataset would then be used to train a lightweight convolutional neural network model in such a way that could be replicated by an end client. The application of using transfer learning from previously trained models to reduce the training time is also explored. This involved training multiple iterations of classification models and using them as well as existing models from the Google Keras and Tensorflow framework for machine learning. The main test model was made with the base model attached to a head consisting of dense neural network layers. This produced a best result of 13.75 percent accuracy and 18.75 with fine tuning enabled with the Keras ResNet50 model as the transfer learning base. This was also tested on a real world sample set of images to identify if the model could return a relevant result within the top three returned results. This yielded a best result of 55.55 percent accuracy with the model trained with the ResNet50 base, however only 33.33 percent with the same model that had further fine tuning. This in combination with a visual assessment of retuned results, gives rise to the main findings that the most influential factor for success in implementing this technique, is the selection of the pretrained model base; as well as the assertion that finetuning may increase the accuracy on the rendered results yet decrease the context transferability that is incorporated into a pretrained model.

University of Southern Queensland

Faculty of Health, Engineering and Sciences

ENG4111/ENG4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

University of Southern Queensland

Faculty of Health, Engineering and Sciences

ENG4111/ENG4112 Research Project

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses, and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

J. Garry

0061107131

Acknowledgements

A thank you to Tobias Low for being willing to supervise this project. Thank you to Sue Garry for taking the time to proofread and support me this far.

Contents

Nomenclature

Term	**Abbreviation**
Application Programming Interface	API
Artificial Intelligence	AI
Central Processing Unit	CPU
Convolutional Neural Network	CNN
Graphical Processing Unit	GPU
Graphical User Interface	GUI
Index of Refraction	IOR
Mechanical Components Benchmark	MCB
Neural Network	NN
Random Access Memory	RAM
Tensor Processing Unit	TPU

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Aim of the study

This study aims to explore the possibilities of identifying assets, mainly computer aided design (CAD) modelled parts using elements of machine learning that will be capable of outputting identifiable information about the part being filmed or photographed based on a local client database information feed into the model. This is aimed at being feasible to implement for industrial end consumers who possess a CAD database and average computing resources. This differs from other classifiers in scope as the goal of this study is to target local CAD databases owned by a client and to identify the part numbers that are associated with the model input.

The hypothesis is that the output 'impression' of a pre-trained model, can be used as an input to a smaller NN that will interpret the impressions from any angle of the part and link it to a one-hot categorical output that will represent the part numbers of the local database. By making the second or 'head' model considerably smaller and less sophisticated, the hope is that it can be easily retrained when the client has a new addition to the database, as the main and deeper classifier remains unchanged.

1.2 Project Scope

The scope of this project will be to define and understand the fundamental operating principles of deep image classification models. This will include well known and standardised approaches to machine learning problems. This studly will limit the scope to exclude implementing uncommon approaches to building lower-level software modules, due to the arbitrary number of possible approaches that could be used in combination.

The scope will also be limited by reviewing and assessing the existing methods of achieving similarly defined problems in the literature and in available resources for the given application of identifying mechanical style parts. This will also be limited to non-proprietary methods and the journals that are available through the University of Southern Queensland.

To conduct initial background research, deep learning models emphasise architectures and implementation strategies used to identify objects as well as possible methods to facilitate generalised identification. This would potentially allow the client to supply only CAD files while being capable of recognising the real world. This is not a project requirement, but the upper limit of functionality expected from this project.

The scope will reproduce existing methods or architectures in code that can output expected results given known inputs. This validates that the methods described in the literature are implemented correctly and that any result(s) described in this study are authentically attributed to the features described. This can be done by using scandalised datasets that can be used as a performance benchmarks.

Evaluation of the performance of the implemented code will occur with identification of areas of potential improvement for the given task. This is planned to be done in a series of stages that act as rudimentary guides for the project. However due to the exploratory nature of this project, the methodology of later iterations of development will depend on the previous result(s).

An optional part included in the scoping of this project, is to explore the nature of identification of parts that are partially obscured and or parts that are worn or damaged. Independently, there is also the optional possibility of implementing a real time camera-based identification proof of concept.

1.3 The Problem

Some businesses may be able to identify labour time spent attempting to finding parts already on the premises that are misplaced or mislabelled. Lost revenue could also be attributed to manufacturing or partway manufacturing a part that is, for whatever reason, left in a transitory state indefinitely. Another aspect is that not every member of the workforce is equipped with the knowledge or information technology to diagnose every part that the business uses on sight.

By providing an open source method to equip the workforce with a tool that may be able to reduce the number of parts that are unidentified, reduction of the number of parts wasted occurs, as well as saving floor, yard or shelf space. In addition, lost productivity could be regained both on the part of the individual potentially querying a part and the person who's time would be used addressing that query, as well as the disruption of workflow that occurs.

1.4 Background

Generally, the standard for machine learning classifiers has been set by large companies such as Google, Microsoft, Apple and Nvidia. The networks, amongst other factors, have been the most successful at bringing image classifiers to consumers with usable results, as they are not trained by the client but instead on the providers end, with large datasets and with better hardware. This provides obvious advantages as it reduces the computing power required for the client to use the classifiers as well as the amount of space that is needed on the device. This approach can be used to make classifiers that have intuition on general categories of targets but may lack the context of the clients' own local environment. For example, a network trained to categorise mechanical components from a service provider, may be able to accurately distinguish between the types or component by category such as 'bearing' or 'bracket' but may not be able to tell if the client has two similar appearing but unique parts that are used in their database such as part number *A* or part number *B*. For a comparison like this, in conceptual terms, some information must be stored locally. An example of one such network would be Figure 1 where there are an equal number of output neurons as there are categories to identify - in this example 0 through 9. If this network were used to identify a part from 10 part numbers it would have

to be trained locally on a dataset specific to the parts that it would be identifying. This means that a network like this would need to be reconfigured and retrained if the client made an addition to their database and now had 11 possible part numbers.

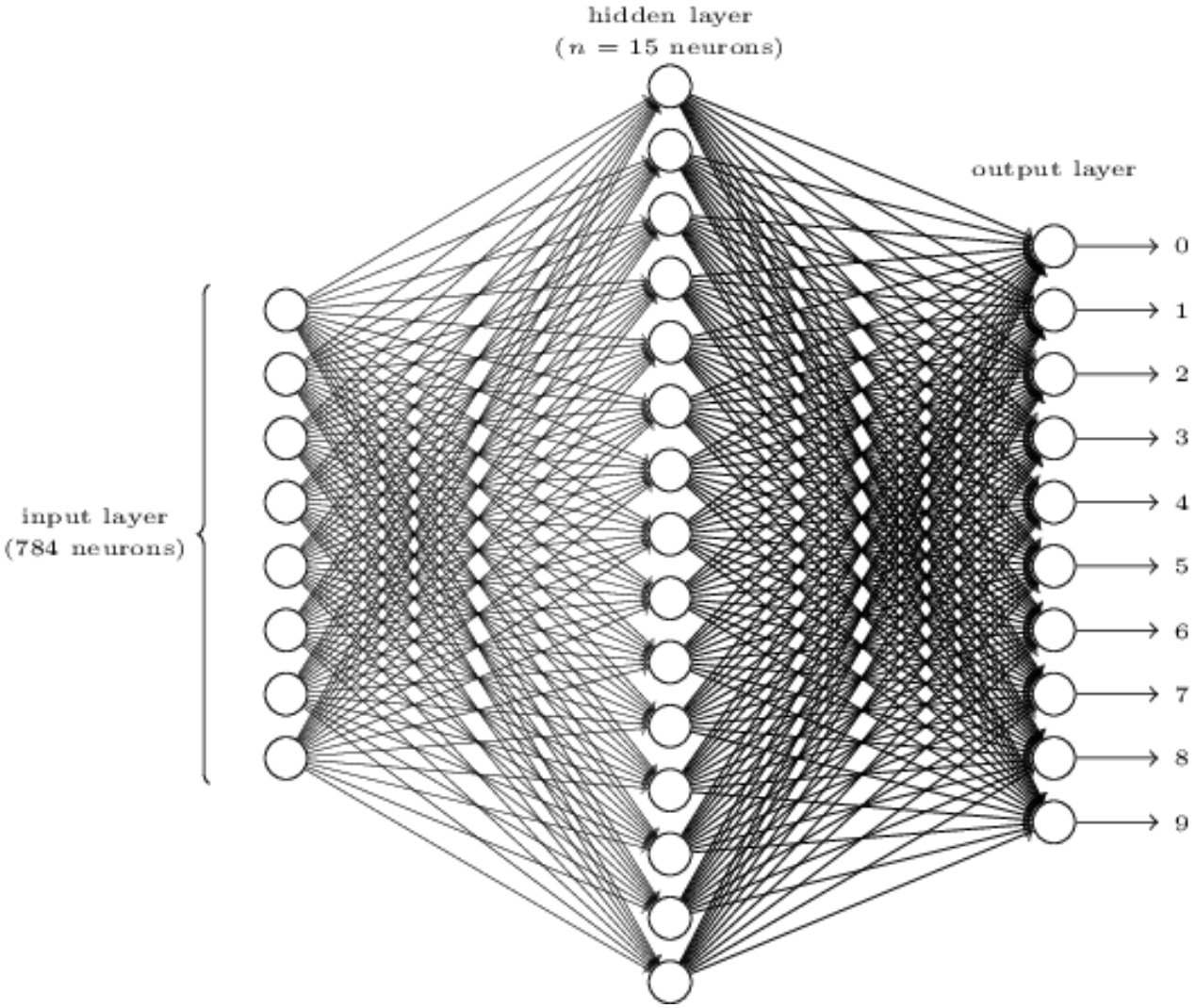


Figure 1 An example of a dense one-hot output neural network (Michael, 2015)

Instead of a one NN classifier with a one-hot output, a possible alternative would be to use a pre-trained model and store the raw categorical output vector in a lookup table that would mirror the database of the client. To use this classifier, the image would be fed into the network and the output vector could be ranked against other vectors in a lookup table and return the closest fit. This method would overcome the NN scaling issue discussed above and would also run very efficiently on low end hardware. One major issue though is the angle of the part being perceived. As a pre-trained classifier would interpret the same part from a different angle as an entirely different part, the output of the categorical indication vector may also be very different. For future reference, categorical, one-hot output vectors will be referred to as ‘impressions’ for practicality.

To utilise the advantages from both scenarios set out above, the proposed network architecture that will be primarily under investigation will be a combination of the two.

Chapter 2 – Literature Review

2.1 Introduction

Typically, the algorithms commonly used to address problems of a similar nature are neural network classifiers. This is where an image is input into the trained network and a result is given in the format of a selection of hard coded categories. These are a type of machine learning algorithm that have become a prevalent approach since a fast-learning algorithm was proposed by (Hinton, Osindero and Teh, 2006) that played an influential role in the development of the modern "deep learning" neural networks (Gulli, Kapoor and Pal, 2019). Hinton's (et, al) work was also responsible for the term 'deep' when referring to neural networks as he introduced utilising 3 to 5 layers. Since that time it is not uncommon to use upwards of 200 layers for deep learning applications (Gulli, Kapoor and Pal, 2019).

(Iyer *et al.*, 2005) reviewed existing methods that could be used for three-dimensional (3D) shape searching based on 3D similarity. This was justified with the corporate setting in mid where staff members, particularly new ones would not know the correct keywords to search for the part or may lack knowledge of the design context that justifies that naming conventions used. (Iyer *et al.*, 2005) uses the shape representation definition of 'a formal scheme for describing shape or some aspects of shape together with rules that specify how the scheme is applied to any particular shape.' One of these shape representations is a shape vector (or feature vector) which uses a series of numeric values to quantify a part. The method for defining such a convention is described as having the capacity for scope, uniqueness, stability, sensitivity, efficiency, multi scale support and capable of computing locally or without the use of cloud computing. Methods such as Global feature-based-techniques, Moments, Spherical harmonics, Geometric parameters, Manufacturing feature recognition-based techniques, Graph-based techniques, Histogram-based techniques, Product information-based techniques (this includes 2D neural network image searches), Group technology (GT), 3D object recognition-based techniques and Aspect graph method were evaluated. "As an example, if the application requires classification of shapes, global object features could be used. If the application requires detection of local dissimilarities among relatively similar shapes, techniques that allow local support such as topological graph-based or skeletal graph-based techniques can be used." (Iyer *et al.*, 2005).

Table 1
Qualitative comparison of 3D shape searching techniques

Technique	Shape representation	Paper Refs.	Refs. for cost	Computational cost (*)	Comparison cost (*)	Multi-scale support	Local support	Advantages	Limitations
Global feature based	Moments	[87,88,93]	+	$O(N^3)$ where N is the number of voxels along each axis	$O(N)$ where N is the number of extracted moments	No	No	a. Fast computation and comparison; b. General Shape Classification	a. Low stability for some classes of shapes
	Spherical harmonics	[96,97,99, 100]	+	$O(N^3K)$ where N is the number of voxels along each axis and K is the number of spherical functions	$O(N)$ where N is the number of extracted harmonics	No	No		
Mfg. FR based	Feature relationship graph	[102–105, 107,108]	+	N/A	N/A	No	No	a. Uses manufacturing domain knowledge	a. Same shape may have different recognized features
Graph based	B-Rep graph	[110,111]	[111]	None	$O(N_0N_j \times L_0L_j)$ where N_0 and N_j are number of nodes less the number of toroidal surfaces and L_0 and L_j are number of links in graphs G_0 and G_j respectively	No	Yes	a. Graphs contain detailed shape information; b. Partial matching possible	a. Large graph sizes affect efficiency; b. Similar models need not have similar graphs
	Graph spectra	[112, 114–116]	[115]	$O(V^3)$ where V is number of B-Rep graph vertices	$O(N)$ where N is the size of spectra considered	No	Yes		
	Reeb graph	[67,117, 118]	[67]	$O(V \log V)$ where V is number of mesh vertices	$O(M(M+N))$ where M and N are number of nodes in the two graphs with $M \pounds N$	Yes	No	a. Simpler representation than B-Rep graph; b. Fast partial matching including local attributes possible	a. Not applicable to all classes of shapes; b. Choice of Reeb function affects results significantly
	Skeletal graph	[119–121]	[121]	$O(N^3)$ where N is the number of voxels along each axis	NP-complete in the worst case	Yes	Yes		
Histogram based	Shape histograms	[74,124]	+	$O(N^3B)$ where N is the number of voxels along each axis and B is the number of histogram bins	$O(B^2)$ where B is the number of histogram bins	No	No	a. Quick comparisons possible in large databases; b. Good method for general shape classification	a. Large number of bins required to accurately describe a shape
	Shape distributions	[75,126, 127]	[75]	$O(S \log N)$ where S is number of samples and N is number of triangles	$O(B)$ where B is the number of histogram buckets	No	No		a. Different shapes can have similar distributions
Product information based	GT code	[131–134]	+	N/A	N/A	No	No	a. Uses enterprise specific domain knowledge	a. Difficult to automate because of high dependence on human intervention
	Section image	[40,42,132]	+	N/A	N/A	No	No	a. Works well for legacy drawings in 2D; b. Useful for searching 3D models from 2D drawings	a. Large amount of data and time required for training neural network for complex shapes
3D object recognition based	Aspect graph	[80]	+	N/A	N/A	No	No	a. Useful for image based recognition of components	a. Very high storage space requirements
	Extended Gaussian image	[81,136]	+	N/A	N/A	No	No	a. Unique representation for convex objects	a. An infinite number of non-convex objects can possess the same EGI.
	Geometric hashing	[85, 137–139]	[138]	$O(MN^{C+1})$ where M is total number of models, each model having N features and C is number of features needed to form a basis	$O(HS^{C+1})$ where H is complexity of processing hash table bin, S is number of features in model and C is number of features needed to form a basis	No	No	a. Useful for exact matching between shapes	a. Very high storage space requirements

N/A, not available; none, representation does not have to be computed; (*), cost values provided are generic and may vary according to algorithm implementation; +, no reference provide any cost calculations.

Table 1 Comparison table of shape recognition techniques

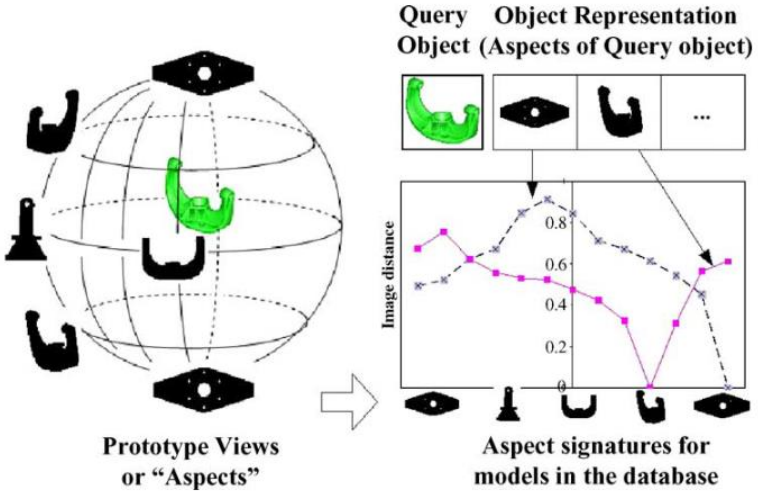


Figure 2 An example of aspect graphs for matching 3D objects (Iyer et al., 2005)

2.2 Successful Approaches

(Rucco *et al.*, 2019) describes a method of part classification where parts are first classified by shape characteristics only, then this is refined with assembly context that the part is used in. The aim is not to be able to find parts but to classify parts into categories such as axis, bearing, cube-like, nut and so on. This way the part can be identified by how it is used and not only by the shape. Since the parts with very similar shapes can be used for different possess the second stage is justified. The datasets were gathered from online 3D file repositories such as grabcad.com and traceparts.com. This is implemented with an artificial neural network (ANN) in a one-versus-rest output, otherwise known as one-hot output. The feedforward backpropagation algorithm was used as the learner algorithm and the logistic loss function with L-BFGS optimization. This was done using the Python SciKit-MPL classifier library, on standard hardware. Out of the variations of ANN trained the most successful results were made with a feature selection and mutual information algorithm inputting into the first layer of the NN which yielded a 95.72% successful classification rate.

Another approach that is similar in aim is by Qin (*et al.*, 2014) at State Key Lab of CAD & CG, Zhejiang University, where parametric features are extracted from models and then used to form "high dimensional input vectors". This strategy was used in conjunction with other three-dimensional strategies to achieve better results. This yielded a classification accuracy of 98.64 percent and used a dataset of 7464 models with 28 categories. Out of these, 5990 samples were used for training and the remaining 737 for validation.

This project proposed here differs from both of the above as the goal is not to classify parts into categories but to classify the model by its locally ascribed part number. This means that, if the above approach were implemented, the network would need to be retrained every time a change was made to the local database in the form of a new part being added or a part name is changed. This is not ideal, as the trading prosses can take a long time to complete and can use computing resources that are not available to an entity such as an

engineering department, such hardware like tensor processing units (TPUs). This means that aspects of the above approach would not be appropriate. For instance, the ability to classify a component to a universal category, even if those categories were company specific.

This also differs from the studies where the aim is to only seek to use existing CAD models as inputs and retrieve similar models based on semantic categorisation. Whereas this study seeks to move towards using images from the real world to identify specific parts. This has the advantage of being able to be used for like-to-like CAD lookup as well as images captured from the real world. This means that the images used in this study must be rendered with the focus on realism so that this cross compatibility exist in inputs. This is far more ambitious in scope than these papers proposed algorithms.

Another point of difference is that these models have access to some key information that is not available in other contexts. Obstructed information retrieval is much easier with three isometric style views, as used by (Qin *et al.*, 2014), than to infer information about a sample with only what is available in the random viewpoint of the captured image.

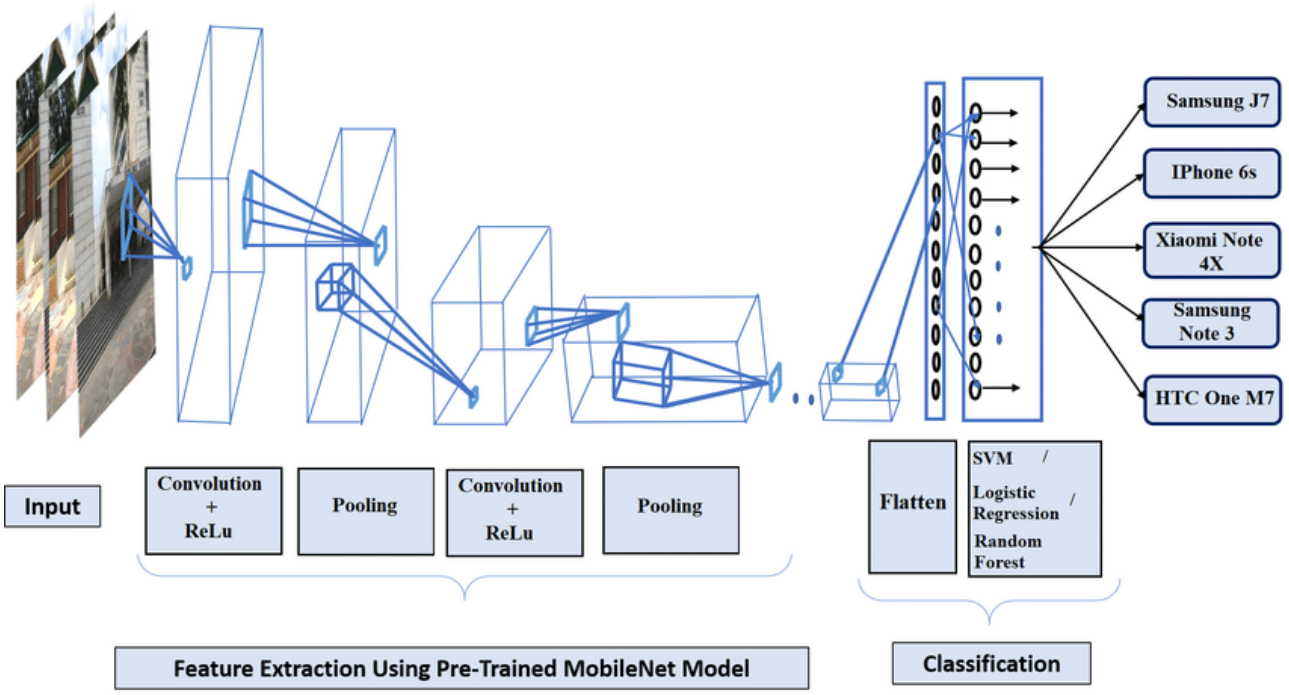


Figure 3 Example of repurposing previously trained layers in transfer learning (Haider et al.*, 2019)*

The success of Haider (*et al.*, 2019) at classifying images taken by three camera models with a deep Convolutional Neural Network (CNN) and transfer learning approach show the potential feasibility of transfer learning. The explanation of transfer learning is "Taking a pre-trained model which is already trained for another purpose and removing the final fully connected layer from the model so that rest of the model will work as a feature extractor for new dataset. After getting the extracted features from this model, they are used to train different types of machine learning classifier. The weights of the all layer are kept fixed" (Haider *et al.*, 2019). An example of this is exhibited in in Figure 3 above where the layers form the MobileNet model

are reused. Given that the method boasts a classification accuracy of 93 percent with a training dataset used has only 3900 images, and should be considered as relatively small by standard machine learning trends (Gulli, Kapoor and Pal, 2019), the claimed susses despite the small dataset is attributed to the effect of transfer learning from a previously trained CNN.

Gulli, Kapoor and Pal (2019) also outline the main advantages of using transfer learning in Deep Learning with Tensorflow 2 an Keras. They state that transfer learning "has multiple advantages. First, we can rely on publicly available large-scale training and transfer this learning to novel domains. Second, we can save time for expensive large training. Third, we can provide reasonable solutions even when we don't have a large number of training examples for our domain. We also get a good starting network shape for the task at hand, instead of guessing it."

2.3 Deep Learning Theory

Although this study will not cover entry level details in depth, any of the fundaments mentioned in subsequent text can adequately be described by the following. An overview of the fundamentals of machine learning and neural networks can be gained from Rumelhart et al (Hinton and Williams 1986; Werbos, 1990; Leven, 1996; Hinton, Osindero and Teh, 2006; Herculano-Houzel, 2009; Schmidhuber, 2015) where the fundamentals are covered. A detailed explanation of how the activation functions and back propagations work can be found in (Leven, 1996) as well as in (Werbos, 1990), this is used extensively as the backbone of most learning algorithms (Michael, 2015). In (Gulli, Kapoor and Pal, 2019) these methods are used and expanded upon to make standard libraries that can be easily deployed, such as in the case of Tensorflow (Abadi *et al.*, 2015, 2016). Schmidhuber (2015) gives an overview of the common neural network (NN) as well as other methods such as regression.

Chapter 3 – Research Design and Methodology

3.1 Challenges of client database model

As outlined in the background section, there are significant challenges that arise from including real world data in the scope of this project. More specifically client-side training. One of which is how to procure images from a local database that can be used to train the 'head' section of the network. For this to translate into the real world the database must be rendered into a realistic set of images. The assumption will be made that the client is able to provide a mass export of three dimensional standardised files such as '.STEP' or Wavefront '.obj'.

Another problem that arises from using these computer generated images is the lack of similarity to viewing them in real life. As rendered images can all be done in the same environment with the same background, lighting settings, floating elevation, material properties and camera settings, there is a tendency for these images to conform to standards that do not represent real world viewing. One way to combat this is by adding large amounts of randomness to the rendered images so that the CNN is not trained with an inability to accept generalised inputs that would come from a camera in the real world. This can be done by adjusting the settings for the material properties randomly. These include randomly changing the camera angle and distance as well as the light source angle, distance and intensity. Another aspect that helps generalise the learning is by limiting the images to a grayscale range and thus reduces any colour complexity that is added in the real world.

3.2 Transfer learning for context generalisation

As mentioned above, the CNN must generalise inputs between the real-world samples and the computer generated images from model geometry. It is the intention of this study to experiment with the use of transfer learning to enhance the transferability of learning from generated images to real world photo domain. The hypothesis is that because networks that are trained on real world data already have less of a bias towards the context of an image rather than a network that is only trained on computer generated images and then used in the real world.

In practicality transfer learning is the practice of taking a pretrained neural network and replacing some of the last layers with, or joining new layers onto the output of the network to reuse for a different purpose. This often saves taring time as the base network already has a good understanding of the inputs and only the last added layers need to be retrained. This has the obvious advantage of needing less training but also has the added benefit of retaining knowledge that is not part of the later training data (Abadi *et al.*, 2016; Gulli, Kapoor and Pal, 2019).

It is often the case that in the beginning of the training session the base model layers are set to untrainable and only the added layers are trained. When the output has converged on a result is common for all the layers to become trainable to achieve a higher validation accuracy(Abadi *et al.*, 2016) (Haider *et al.*, 2019).

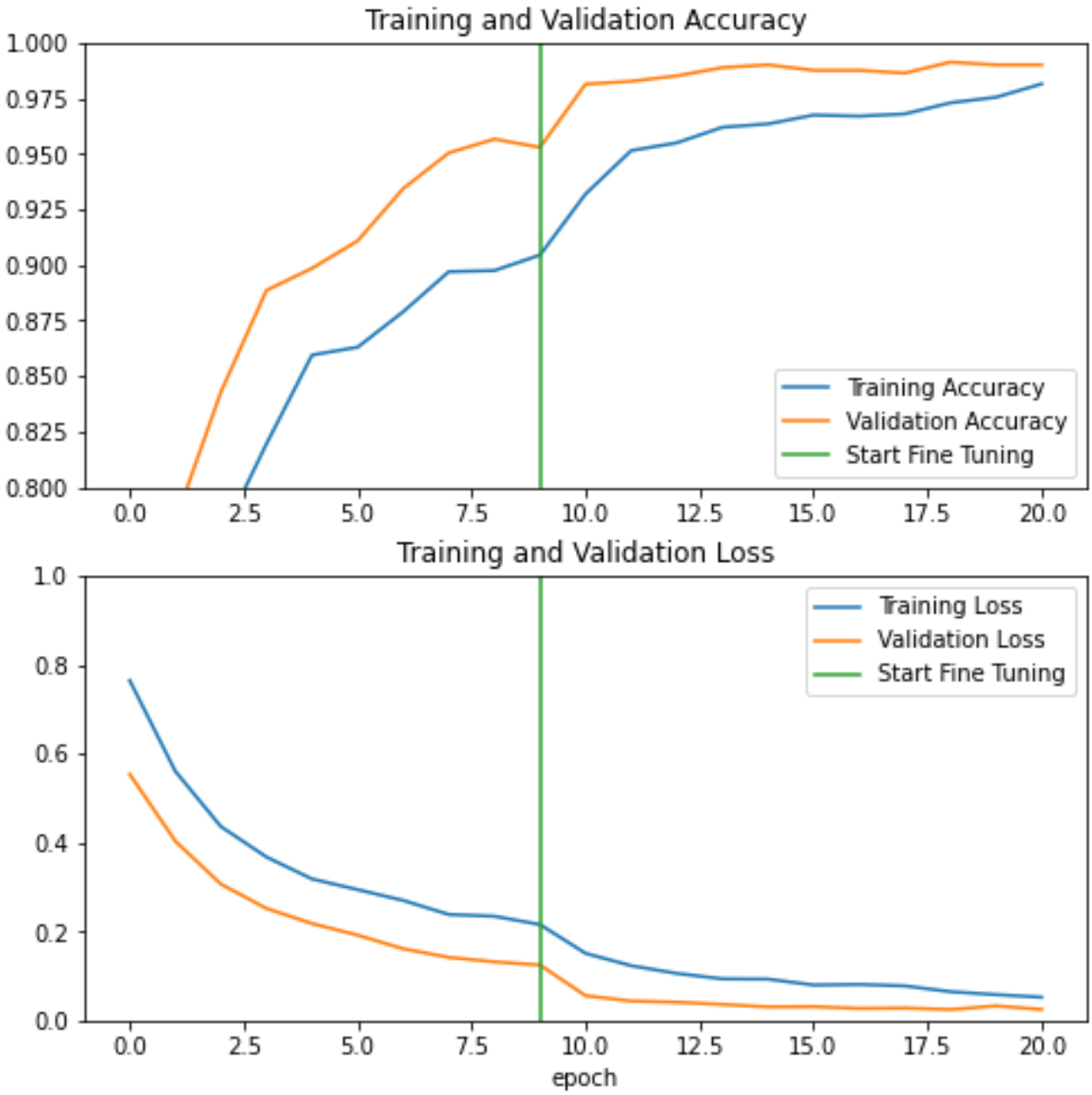


Figure 4 An example of fine tuning in transfer learning (Abadi et al.*, 2016)*

3.3 Preposed Network Architectures

The aim of setting network architectures gaols here, is to give the project some initial scope and planning goals that serve to demonstrate that parts of the machine learning process are working by using easier validation targets to aid development.

The first, a convolutional neural network (CNN) style is to display a working proof of the concept that shows that is it is possible for the deep learning process to take in some information in the form of images and output one-hot classifications of the input images. This is intended to validate that the back propagation algorithm

portion of the convolutional neural network classifier, is working a basic capacity to memorise parts that it has encountered before but from a slightly unknown angle. This will use a small and pre rendered database to limit the number of variable aspects.

The second benchmark architecture to make, is to be able to classify parts into categories, replicating the above studies covered in the literature review section. This would use the shape benchmark repository that is available and use renderings of the mode files that are randomly generated.

The third architecture that can be used to benchmark progress, is to use a classifying deep network to form 'impressions'; a column vector of one-hot outputs in floating point decimal form of the images of a locally defined part, that then uses an impression head to link the impressions of a part from all angles into a second one-hot output, corresponding to the local part code category. This is intended to be a means of avoiding training the more intelligent parts of the image classifier, each time a part is added to the local database; but only to collect all impressions of that part from several angels and to train the much smaller impressions head portion which would requires less depth in learning - resulting in faster training times. This could be implemented by training the categorical classifier or using open-source pre-trained networks such as the open CV project or Keras models.

A final measure that has been left as optional is to try and use partly hidden, obstructed or removed portions of the input images of the parts, as a means of demonstrating asset identification that is not subject to ware or damage. This has been played out in the scoping as a purely optional activity if time permits.

3.2 Datasets

3.2.1 Shape Benchmark

(Jayanti *et al.*, 2006) has made a benchmark dataset of engineering parts that can be used as a standardised benchmark for engineering model classification. These are divided up into three main categories, being Flat-Thin Wall Components, Rectangular-Cubic Prism and Solid of Revolution. The major advantage of this dataset is that it is relatively small and has accompanying rendered images that are consistent in appearance.

Category	**No of parts**
Back Doors	7
Bracket like Parts	18
Clips	4
Contact Switches	8
Curved Housings	9
Miscellaneous	12
Rectangular Housings	14

Slender Thin Plates	12
Thin Plates	23
Bearing Blocks	7
Contoured Surfaces	5
Handles	18
L Blocks	7
Long Machine Elements	15
Machined Blocks	9
Machined Plates	49
Miscellaneous	12
Motor Bodies	7
Prismatic Stock	36
Rocker Arms	10
Slender Links	13
Small Machined Blocks	12
T shaped parts	15
Thick Plates	12
Thick Slotted plates	20
U shaped parts	25
90-degree elbows	41
Bearing Like Parts	20
Bolt Like Parts	53
Container Like Parts	10
Cylindrical Parts	43
Discs	51
Flange Like Parts	14
Gear like Parts	36
Intersecting Pipes	9
Long Pins	58
Miscellaneous	12
Non 90-degree elbows	8
Nuts	19
Oil Pans	8
Posts	11
Pulley Like Parts	12

Round Change At End	21
Simple Pipes	16
Spoked Wheels	15

3.2.2 Mechanical Components Benchmark

Another dataset that can be used in a similar way is the Mechanical Components Benchmark (MCB) dataset, made an open source by (Kim *et al.*, 2020). This dataset contains 18,038 mechanical components split into 68 classes. This dataset is structured as a "hierarchical semantic taxonomy" as shown below. These files are stored as Wavefront '.obj' files and need to be encoded in some form to be used as an input to a neural network. In this case, this means that the files must be rendered as images in to be used as inputs. This comes with some challenges as the polygon count for some of these files are relatively low and make round components appear jagged. This is addressed in more detail in the results section.

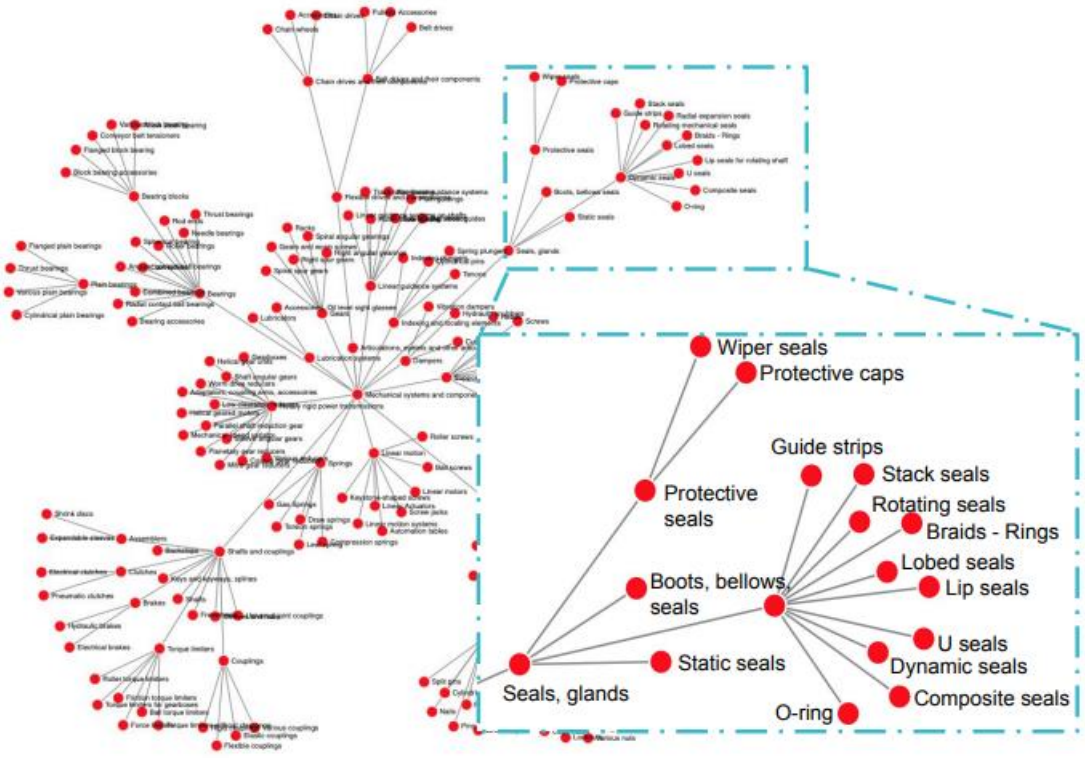


Figure 5 The hierarchy taxonomy of mechanical components based on the International Classification for Standards from (Kim et al., 2020)

3.2 Dataset Augmentation

After splitting a dataset into training and testing lists, it is common to use some method of stretching of the training dataset that helps give more training samples to the neural network (Michael, 2015). This is intended

to prevent overfitting of the training dataset. These methods include, scaling, shifting, rotating, and flipping in random combinations to maximise the training information. This has commonly been observed to increase the generalisation of learning and reduce overfitting by increasing the dataset size as well as increase the variety of input styles that the network can handle (Abadi *et al.*, 2016) (Gulli, Kapoor and Pal, 2019).

3.3 Validation and Training

Throughout this prosses the validation processing in integrated into the machine learning workflow. Particularly in cased of supervised learning where the data has corresponding labels for all samples of data it. It is standard practice to incorporate a dataset split where roughly 70 to 90 percent of the dataset is used to train the network in repetitions known as epochs, where one epoch represents a loop of all the training samples. Since the remaining portion of the dataset is not used to train with this can be used to test the performance of the network on samples that it is not familiar with. In this evaluation phase backpropagation is not done resulting in no learning taking place. This has the practical effect of the network training the test dataset with complete amnesia encountering them for the first time every epoch. This means that the test, or otherwise known as validation dataset, can be treated as a reliable source of accuracy.

A distinction should be made however to the credibility of validation accuracies that are derived from data sources that are fabricated in the same fashion as the testing data. This is a potential problem when the intention is to repropose this for the real-world domain. Because of this the readeer should be mindful of the two types of validation testing context.

This is important to any scientific study as ethical claims made about the content of any report must be backed up by validation data or else disclaimed against. For these reasons validation testing, once per epoch of every training loop will be implemented as well as an assessment of the transferability to the real world domain context.

3.4 Softmax Output as an Indicator of Probability

"The output from the Softmax layer can be thought of as a probability distribution" (Michael, 2015). In this case it will be implemented to refine the selection power of the CNN output. This is because the sum of a Softmax output is one and thus can be used to represent a relative sureness of the selection of the part. This takes the form of

$$a_i = \frac{e_j}{\sum e_j} \qquad 1$$

Where 'a' is the output of the next layer and 'e' is the values of each of the previous layers.

Chapter 4 – Results and Discussion

The following section outlines the main activities undertaken to implement the desired experiments described in chapter 3. This is presented with the prepossessing sections first for the convenience of the readers understanding and then the chronological order of the development of the tested architectures. As an arbitrary naming convention for the test networks the abbreviation ‘TestNetx’ is used in code and in the report.

4.1 Dataset Prepossessing and Image Rendering

As the mechanical component dataset consists of only Wavefront object files, these need to be rendered in order to be used as inputs into the CNN. One problem that arises from this, is the low polygon count of the models which have rounds that are rough looking and not true to life representations of the components that they represent. Using the open-source 3D graphics software Blender to load and render a bearing from the MCB dataset, the problem is made clear.

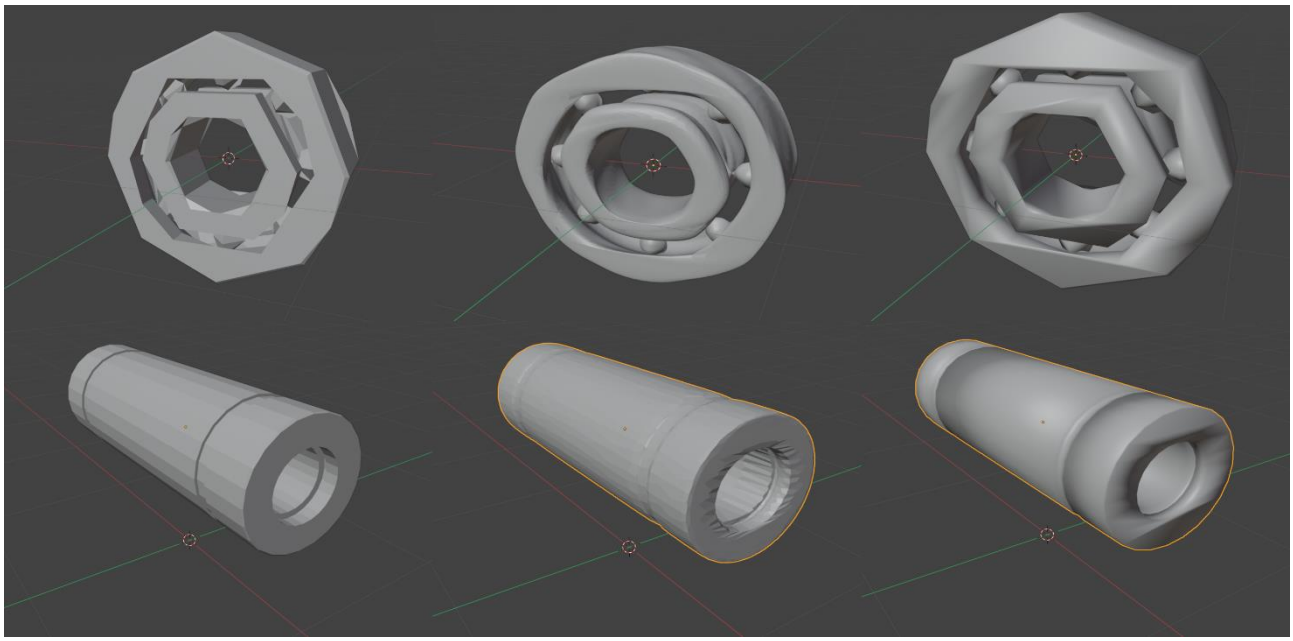


Figure 6 From the top left bearings ‘00040187.obj’ and below ‘00000043.obj’ unedited, surface smoothed with Catmull-Clark algorithm and Blender’s Render Smooth option

By setting values for material properties that were reasonable by visual inspection, the flowing result was achieved. However due to some parts such as pins rounding edges that are not meant to be rounded, this method was not used in the final batch rendering. This improved the render times as demonstrated in finger.

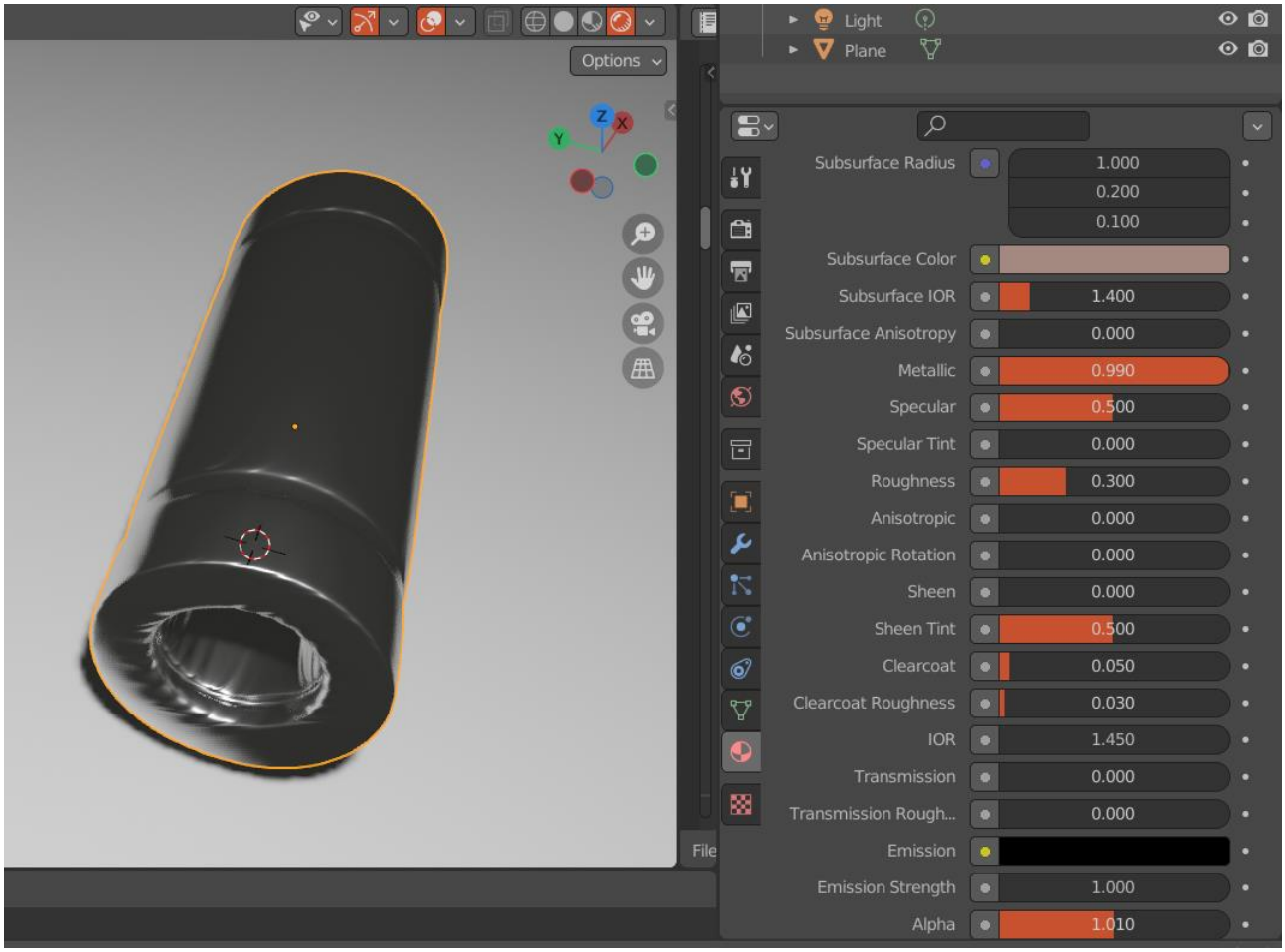


Figure 7 Blenders material settings used on '00000043.obj' from MCB dataset

To automate the rendering of the large MCB dataset a python script was written to utilise the Blender application programming interface (API). These settings are tabulated below.

Table 2 Randomised changes to the Blender material settings

Property	**Value Range**
Surface subdivision no	0
Plane scale	20
Object height	[0.8, 1.6]
Object rotation	([0, 2pi], [0, 2pi], [0, 2pi])
Surface colour	(0.369414, 0.252091, 0.216505, 1)
Surface index of refraction	[1.000277, 1.000293] * for air at 1 atmosphere
Surface Anisotropy	0

Metallic	[0, 1]
Specular	0.5
Specular Tint	0
Roughness	[0.1, 0.5]
Anisotropic	0
Anisotropic Rotation	0
Sheen	0
Sheen Tint	0.5
Clear coat	[0, 0.05]
Clear coat Roughness	[0, 0.03]
Index of Refraction	1.450
Transmission	0
Transmission Rough	0
Emission	(0, 0, 0, 1)
Emission Strength	1
Alpha	1
World Colour	(0.242276, 0.240045, 0.218748, 1)
World Colour Strength	0.1
Light Position	([-5, 5], [-5, 5], [0, 10])
Light Power	[300, 3000]
Camera Position	(0, [2, 5], [2, 5])

* Citation Optics by Zajac and Hecht (2003)

By using the default python random model, each of the values in square brackets could be initialised with a non-integer random value for each model rendered. The algorithm for this goes as follows, the old imported object is removed, the settings in the above table ware initialised with random values in the ranges, the next object is imported, the surface is smoothed with the subdivision tool, then the material properties and scene properties are updated with the values made from the above table. The location of the imported object is changed to increase the height, so that it is not penetrating the surface. The camera position is then moved along the y and z coordinates, the light is then set to a random point in 3D space, the camera direction is then adjusted to point in the location of the imported object using the vector made by subtracting the position of the camera from the position of the object and then converting this to a Euler rotation. The world background colour and intensity is updated, the render path is set and finally the render operation is stated.

Figure 8 A sample of typical renderings. From the left column a wheel, castor wheel, flange fitting and report

Above is a sample of the typical renderings with five parts across the columns as well as the random scene generations displayed down the rows. The angle of the flange may be difficult for a classifier to identify as most of the distinguishable features are only visible from the front and not the sides or top. This would be a challenge for most humans to identify.

Table 3 Categorical Distribution of MCB Dataset Samples

Feature Type	**Rendered Images**	**Percentage of Dataset**
Bearing	2625	3.48
Bushing	2961	3.93
Castors and wheels	1815	2.41
Clamp	785	1.04
Disc	545	0.72
Fitting	8780	11.65
Flange	1990	2.64
Fork joint	235	0.31
Gear	2575	3.42
Handles	8755	11.61
Hinge	305	0.4

Hook	610	0.81
Motor	3730	4.95
Nut	4569	6.06
Pin	11164	14.81
Plate	1537	2.04
Pulley	1309	1.74
Ring	2314	3.07
Rivet	214	0.28
Rotor	1973	2.62
Screws and Bolts	13450	17.84
Spring	623	0.83
Stud	632	0.84
Switch	317	0.42
Washer	1580	2.1
Total	**75393**	

The performance of this approach was very poor, as the rendering time nominally exceeded 2 seconds. This is a practical problem as the MCB has 18,038 potentially resulting in possessing times of 32 hours. Another problem with running this script from the Blender graphical user interface (GUI), resulted in occasional crashes. Some of these could be attributed to memory overflow and failure to read the object files as well as unknown errors.

To increase the stability as well as the performance, the scripts can be executed by running Blender from the command line and supplying the headless flag as well as the python script as an input argument. This resulted in better performance and higher stability, yet still had a lot of crashes.

To reduce this problem further, the command line call was run from inside a python while loop so that crashes could be caught, and the imported file could be flagged and the process repeated. A file dump was introduced to the Blender python API script to output the current model path that is in use as well as the settings applied to the scene. This means that the imported object can be flagged more easily.

```
Writing: C:\Users\J\AppData\Local\Temp\blender_scripting.crash.txt
Error   : EXCEPTION_ACCESS_VIOLATION
Address : 0x00007FFC4FB21F21
Module  : VCRUNTIME140.dll
Thread  : 00002b2c
```

Figure 9 Crash report from Blender when scripting render process

4.1.1 Blender Efficiency Study

Making an input pipeline was attempted within the Blender python API script with the intention that while the GPU is rendering the scene, the CUP can be utilised by preparing the next scene. This was done using the python module threading form the standard library. Due to the way that the script and the Blender program is structured, this approach was found to be not possible.

Blender Processing Sequence CPU and GPU Load Over Time with and without Smoothing

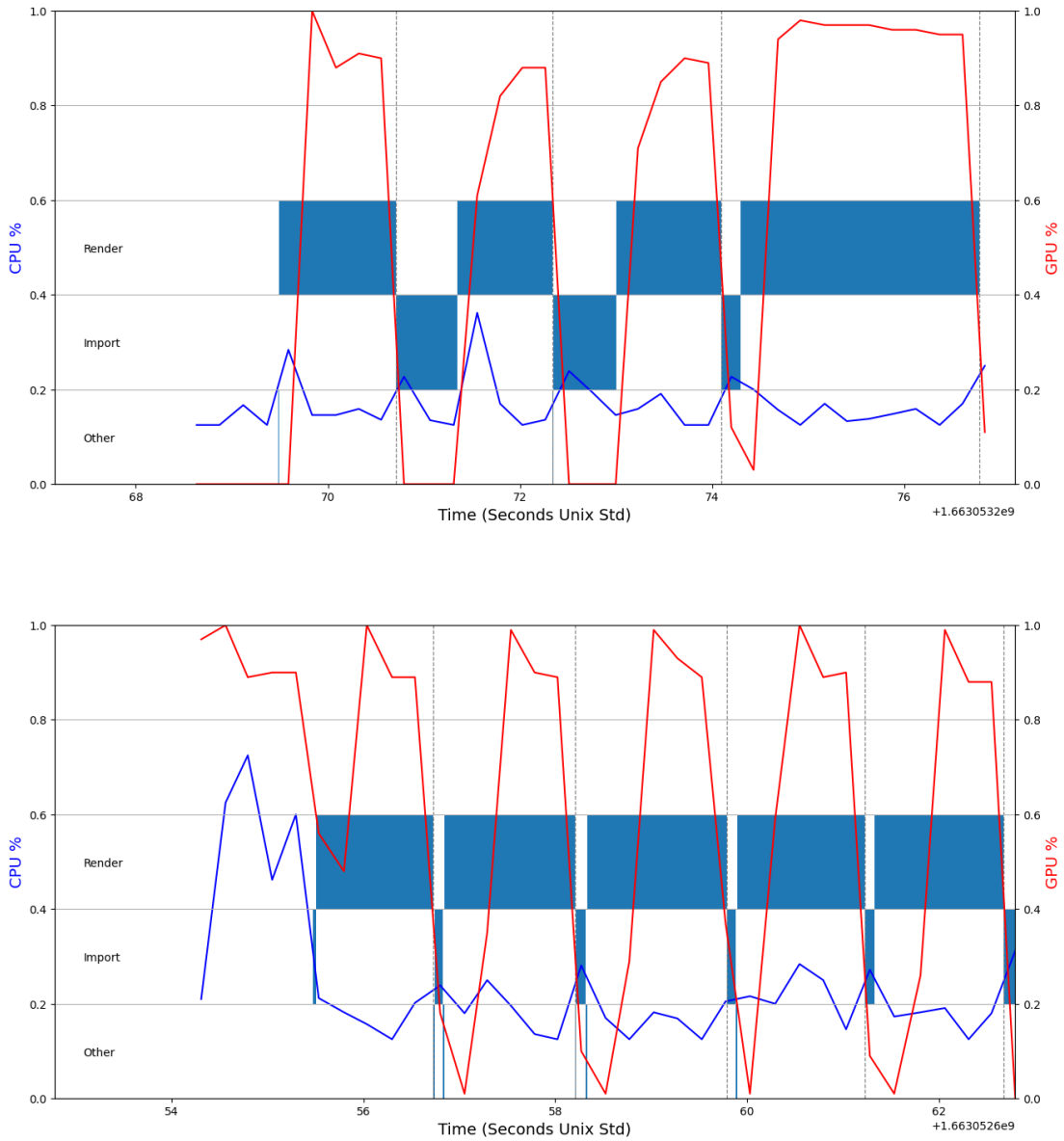


Figure 10 Top: Sequence of process without surface smoothing. Bottom: Without smoothing. Vertical grey lines indicate a new part. Note the x axis time scale is not consistent across both plots.

The above graphs indicate the computation time to render an image in Blender. The time per object varied widely as it is heavily dependent on the number of faces in the object file. Most of the processing time is spent by the graphical processing unit (GPU) rendering the image, as indicated by the above graph. The second most computationally expensive process is done by the central processing unit (CPU) importing the model from an object file into the current Blender scene. The remaining computations described earlier are done in under 10 milli seconds and these are displayed as one group on the above graph. This indicates that there are possible improvements that could be utilised in the rendering process, by incorporating an input pipeline in the rendering prosses so that the rendering could be done while the CPU is preparing work for it to do. Under optimum conditions, this may reduce the time down to only that of the largest process time displayed above.

It is also worth noting that this prosses can be optimised more when the models are not smoothed as opposed to when they are, as indicated by the top and bottom plot.

4.1.2 Experimentation with Blender Instance Count Efficiency

As an experiment, a comparison was made on a set of 50 object files to benchmark whether running two independent instances would produce a render rate that was faster than only one. While two instances were running, the GPU load was always at full capacity. While when it was only one, the GPU load was intermittent as seen in Figure 10. The two-instance run was configured such that the files were executed in alphabetical order for the first and in reverse for the second. So that they did not waste time rendering the same part. The results indicate that it is more computationally efficient to only run one instance. One possible reason for this behaviour, was that the time taken to manage the threads of the CPU and GPU was greater than the efficiency gained from keeping the GPU occupied the whole time.

Table 4 Render Program Instance Count Performance

Rendering Method	**Time to render set of 50 objects** (Seconds)
Single Blender Instance	83.55
Two Blender Instances	103.12

4.1.3 Corrupted File Filtering

It was noticed in the experimentation phase of ‘TestNet4’ that crashes were being caused by corrupted files and would halt the training process. This was fixed by developing a rudimentary search algorithm that could identify corrupted files be walking through the MCB rendered dataset and parsing the images into the *imread* function of the Python MatPlotLib library. This meant that these images could be removed easaly. A total of 3 of these images were removed.

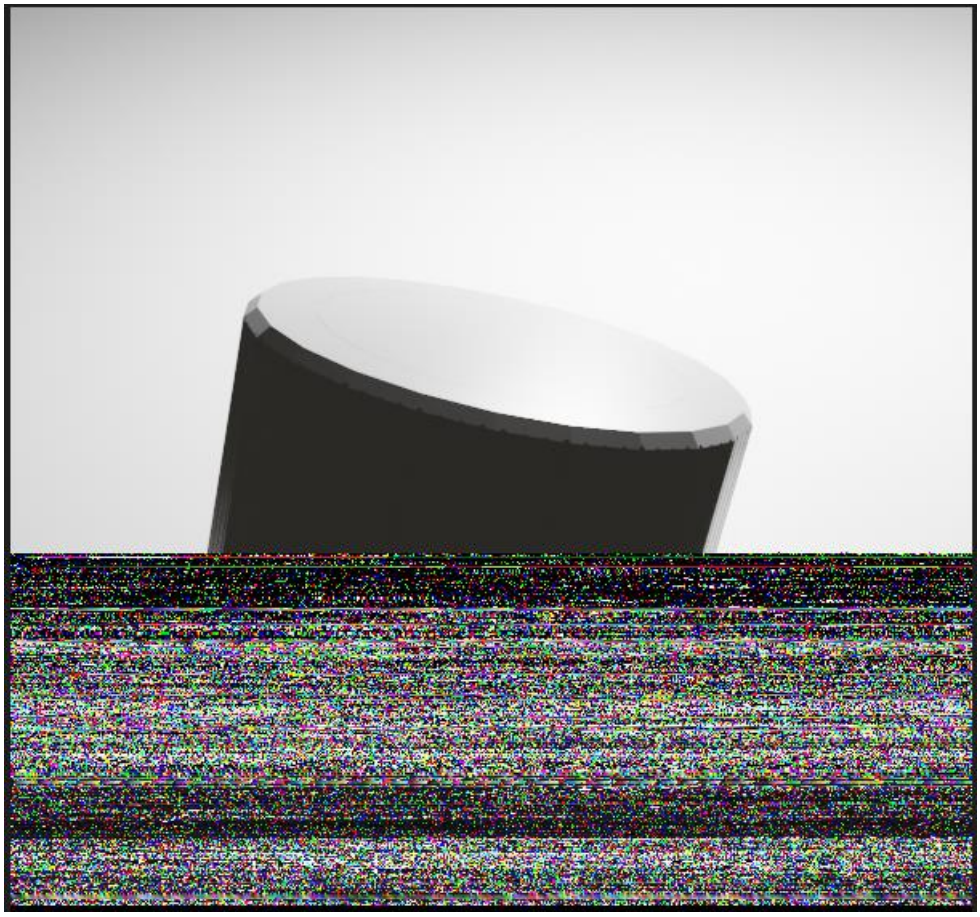


Figure 11 Example of corrupted file from Blender rendering

4.2 Prepossessing for Local Part Number Training

Due to the way that the MCB dataset is structured, with subfolders grouping categories of parts together, for 'TestNet5' and onwards, a directory restructure is required where the category of parts is the part number itself. In other words, when the dataset is retrieved from the directory, the sample is an image the target, or answer, is the directory name. In earlier models, architectures where categorical part type learning was the goal, the default structure is required. This setup is a result of the way the Tensorflow inputs data from the method *tf.keras.preprocessing.image_dataset_from_directory()* as the target for any given sample is the directory name that the image is located in.

This process was done twice, the first one to produce a slice of random 100 objects from the MCB dataset for use in 'TestNet5' and 'TestNet7' and the second with 250 objects for use in the 'TestNet6' experiment. The justification for this is made in the sections for each of the architectures. This was done by collecting all the paths of the object files recursively in the MCB test folder and then selecting from these 'N' samples to copy to the new selection 'slice' folder and moving all images with it. Because the main MCB dataset had only 4 images per object an additional 12 were rented on each of these selection slices.

For future reference the two dataset configurations will be referred to as ‘categorical structure’ and ‘local structure’. Where categorical structure uses the category of part as the target as opposed to local-structure has the part numbers as the targets.

4.3 ‘TestNet1’

The prototype network 1, henceforth referred to as TestNet1, is constructed with the following architecture. This is implemented in Python 3.10 with the Tensorflow 2.0 and Keras library. The layers with an architecture consisting of three convolutional layers starting with an input layer of 60 by 60 by 3 pixels with a 3 by 3 convolution mask applied over the layer, then a max pooling layer, sequential NN with Adam optimisation. The input shape is justified by the 1;10 downscaling of the images from 600x600 plus to square 60x60 pixel images.

Table 5 TestNest1 layer structure and shape

Layer no	**Layer type**	**Dimension**
1	Convolution	60x60x3
2	Max pooling	2x2
3	Convolution	120, 3x3
4	Max pooling	2x2
5	Convolution	120, 3x3
6	Dense	120
7	Dense	45

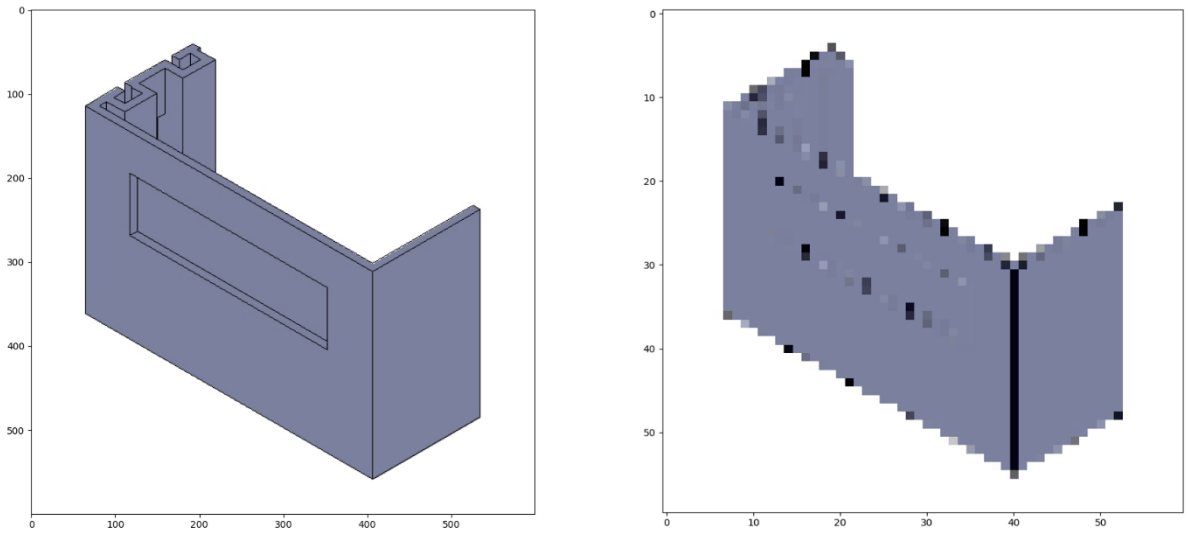


Figure 12 From the left 600 x 600 then 60 x 60 pixel image

This was trained on the categorically structured engineering shape benchmark repository with 80 to 20 percent training to testing dataset split. The training was done by loading the entire set of processed images into the training function without breaking into batches. After each epoch (being a loop of the full set of the training data) had completed, the test was run with only the test dataset. After 60 epochs, this produced an output accuracy of 72.832 percent.

This output was interpreted as suspiciously high for such an initial attempt. As such a small dataset was used with no known standard improvements, such as dataset augmentation, had been used. The culprit reason for this was hypothesised as being instances of at least two near identical photos of the same part existing in the training and testing dataset split, displayed below. These samples of the dataset contaminated the testing process as the network did not learn generalised information but rather over learnt the training dataset, thus not gaining a deep intuition of the context of the images but memorising the training samples. This is described by (Michael, 2015) as being a possible occurrence for networks that are evaluated in the training stage, particularly when the epoch count is high.

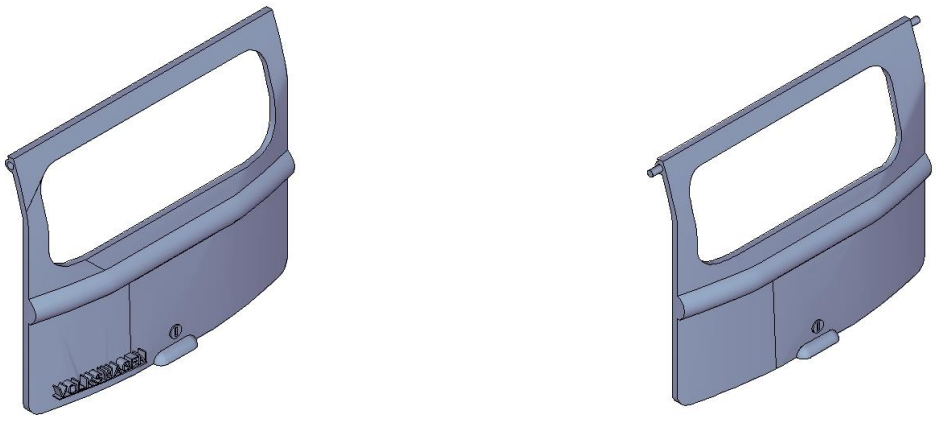


Figure 13 From the left ‘backdoor.jpg’ and ‘backdoor1.jpg’ are independent in the 3D benchmark repository

4.4 ‘TestNet2’

This prompted the addition of an algorithm capable of splitting and shuffling the dataset into the training and testing with like-datapoints of objects being exclusively randomly ordered in only one list. This requires that the items get grouped together fist before splitting. Split the unique groups apart in the training and testing split, and then shuffle all the datapoints around within the training and testing. The unique groups were identified by name similarity. For example, a part of *part.jpg* and *part1.jpg* would be put into the same group.

Table 6 Best Setup for 'TestNet2'

Variable	**Value**
Split	0.8
Rotations	0 to 360 degrees in increments 30
Translations	30
Seed	1
Batch size training	8
Batch size testing	5

The best result that was achieved from this setup was 40.0 percent accuracy on the validation data using the above global parameters. The data was trained on a Nvidia DGX station and ran for over 15 hours. This only made it 84.3 percent through the 866 samples, 693 of which were used for training and for each of those images there were 50 augmented photos produced, giving a total of 34,650 images per training loop, otherwise referred to as an epoch.

Because the code for retrieving, separating and augmenting was done by writing a function in python manually, this was not efficiently done. One of the reasons for this slow processing speed is that the augmented images were produced before the start of each batch of learning as opposed to being stored in memory which would have been considerably faster. Another reason for this large inefficiency was that this method was not optimised with a pipeline. This is where the CPU prepares work for the GPU, while it is busy, so that the downtime is minimised between GPU workloads. In the context of machine learning, this usually means that the images are retrieved and loaded into random access memory (RAM) in batches and sent to the GPU when it has finished processing the last batch. However, due to the function utilising a sequential workflow, the slowdown was apparent.

Another reason that this method was unviable to use, was the degree of inefficiency in the way that the image augmentations were done. In this attempt, augmentation was done on the CPU using the SciPy modules *ndimage* function to zoom and rotate, then Matplotlib's *imread* function loaded into the RAM and NumPy data arrays to perform image translations. The officially recommended method to include augmentation in a Tensorflow project, is by adding random transformative layers into the model so that the augmentation occurs on the GPU while training (Abadi *et al.*, 2015) (Abadi *et al.*, 2016).

4.5 'TestNet3'

The main goal of the 'TestNet3' was to try and utilise the more efficient and faster methods from Tensorflow and Keras with *image dataset from directory()* function as well as the mirrored distributed training strategy form the Tensorflow module in combination with using the dataset objects with the *prefetch()* method so that an input pipeline could be used. This would make training considerably faster than the previous implementations if successful.

This was trained on the Shape Benchmark dataset in a categorical structure where the output layer of the model corresponded to the 3 main group categories of the dataset. This was used to see if the network could identify the category of model from the relatively small dataset of 874 images without any image augmentation. This lightweight system was easy to debug and explore possibilities quickly, without huge amounts of time invested in writing code.

Table 7 Best Setup for 'TestNet3'

Variable	**Value**
Split	0.8
Seed	1
Batch Size	100
Batch Size per Replica	25
Epochs	50
Steps per Epoch	5
Image Downscaling	6
Best Validation Accuracy Achieved	99.58%

This gathered a 99.58 percent accuracy, which is a good but misleading result, due to the low number of output categories - in this case 3. This did however demonstrate that the input pipelines were considerably faster in training the model, than the previous 'TestNet2' experiment that was trained with the same dataset. It also achieved considerably higher accuracy. Although for the same reasons, this is discredited, as on the 'TestNet1' as legitimate deep learning. The purpose of this experiment was not to show that it is possible to classify images but that they can be done faster and more easily with the *Tensorflow.Data.Dataset* objects as well as aid in development of the next development iteration as well as benchmark the classification accuracy.

Using the layer shape of three convolutional layers and 600 by 800 by 120 by 3 densely connected layers, should yield a deeper intelligence than the shallower network in 'TestNet2' as indicated by the success of the deeper networks used by Goodfellow (*et al.*, 2014) to identify complicated information.

Table 8 Layer Structure of 'TestNet3'

Layer no	**Layer type**	**Dimension**
0	Input	100x100x3
1	Convolution	3, 1x1
2	Max pooling	2x2
3	Convolution	120, 3x3
4	Max pooling	2x2
5	Convolution	120, 3x3
6	Dense	600
7	Dense	800
7	Dense	120
8	Dense	3

One of the major hurdles that this method ran into that is unique to the distributed training strategy, was the tendency for Tensorflow models to not release GPU memory when the task was complete or when the process was interrupted. This was a known issue reported in the Tensorflow GitHub repository bug submission. The user ‘zheng-xq’, marked as a contributor to the project, posted the explanation “currently the Allocator in the *GPUDevice* belongs to the *ProcessState*, which is essentially a global singleton. The first session using GPU initializes it and frees itself when the process shuts down. Even if a second session chooses a different *GPUOption*, it would not take effect. (Abadi *et al.*, 2015)” A temporary solution was submitted on the development forum stack overflow, where the user ‘Oliver Wilken’ (2017) suggests putting the original code in a function and then starting a new prosses and then joining it to the main thread when it has terminated using the python inbuild multiprocessing library. By joining to the main thread again, the Tensorflow GPU memory could be cleared. In order to utilise this to the fullest exent, an external kill function was implemented by adding a ‘kill’ key pare to a saved text file, so that when a running script needed to be stopped, the text file could be edited and the script would terminate gracefully at the end of the current training epoch. This method yielded no further crashes.

Another thing that was explored, was the standardisation of output metrics by incorporating model saves when a new best-classification result was made. This meant that the experiment’s performances could be tracked easily. This worked by saving the setup parameters that could be adjusted to a text file along with the model information such as the weights and biases and layer structure.

4.6 ‘TestNet4’

Test network four aims to identify parts by categories and was the first network that was used on the MCB database on the rendered images that are discussed in the data prepossessing section. This was using the same input pipeline and generator techniques as used in ‘TestNet3’. A variety of layer structures, depths and augmentation parameters were experimented with.

The image augmentation was done by adding random rotation, translation, contrast and zoom layers to the model. Augmentation by mirroring was not implemented as the distinction between left- and right-hand versions of parts needs to remain. An example of this is parts that have thread, since thread is mostly right hand, except in select uses. This may be relevant information that a classifier can use to differentiate parts.

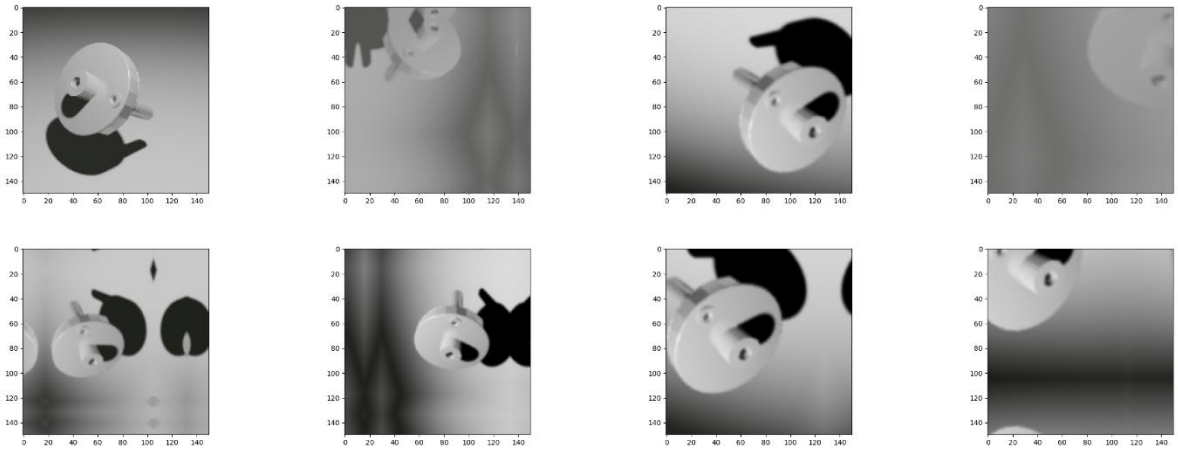


Figure 14 Example of Image Augmentation

In the above figure, the top left is the original image belonging to the handle category and the remaining are generated from this, using a random contrast and brightness of 60 percent and translation and zoom of 40 percent. These values are higher than the ranges that were tested for demonstration purposes.

Various implementations were tried with adjustments to the layers structure, convolution mask, augmentation parameters of this setup, used to identify optimum working results. Below in Table 9 is the layer structure of that yielded the best results.

For of these iterations are noteworthy. The first used ten percent augmentation for each of the four augmentation layers described above as well as a layer structure of 4 convolutional layers with 4 dense layers of 600, 800, 120 and 25. This did not converge on the training dataset very well.

After experimenting with the amount of augmentation the next iteration used six percent and used roughly the same layer structure as the VGG16 network with some simplifications to work in a sequential style. The justification for choosing this layer structure is based on the success that the VGG16 network had particularly that this network stands out as one with relatively few layers compared to the rivalling structures such as ResNet50 that is 50 layers deep (Gulli, Kapoor and Pal, 2019) (Simonyan and Zisserman, 2014). The layer

structure is the same as that of Table 9 except that every convolutional layer after layer 2 had half the depth. This showed much more promise than the shallower network at converging on higher validation accuracy.

Experimenting further, the next iteration of 'TestNet4' used double depth of the previous attempt. This output the highest classification accuracy of 30.316 percent classification on the standardised validation dataset of the MCB dataset.

Another test was done to see if doubling the layer depth again would result in higher accuracy, but this did not yield any decent results to date. This implies that it is redundant to add more depth to the convolutional layers. Whether this is the case while changing the number of layers, layer types or activation functions it can not be definitively stated.

Table 9 Layer Structure of best 'TestNet4'

Layer no	**Layer type**	**Dimension**
0	Input	224x224x3
-	Random Contrast	6%
-	Random Rotation	6%
-	Random Translation	6%
-	Random Zoom	6%
1	Convolution	3, 1x1
2	Convolution	128, 1x1
3	Max pooling	2x2
4	Convolution	256, 1x1
5	Max pooling	2x2
6	Convolution	512, 1x1
7	Max pooling	2x2
8	Convolution	1024, 1x1
9	Max pooling	2x2
10	Convolution	1024, 1x1
11	Dense	25

The classification rate of 30.316 percent is amicable as a lot of the components would not be easily identifiable. This is also compounded with the problem explained in section 3.1 where some views of a part are feature rich and others are not. This classification accuracy should not be considered in the same context as other machine learning project that have a biased set of images that show the subject in question clearly. This is also a good result given the relatively small dataset of 75,393 images to train from, the common benchmark ImageNet dataset for classification has 14,197,122 images in it (Yang *et al.*, 2020). Although this dataset has more

categories and has a more diverse set of images it is worth putting into perspective the size of datasets that are common in machine learning applications.

4.7 ‘TestNet5’

Unlike the above architecture, this is the first to utilise the local per part category structured employed. This was done using a pretrained model with transfer learning utilised by connecting the last layer of the pre trained model to the trainable payers added. This was configured in such a way that the only layers that are trainable are the added layers. This was done with the intention of utilising the generalised image recognition of the base model. Four pre trained models were used as a base for this experiment, these were the ‘TestNet3’, ‘TestNet4', VGG16 and ResNet50 classifiers. These were all run using the same setup parameters and training data.

Table 10 layer structure of 'TestNet5'

Layer no	**Layer type**	**Dimension**
0	Input	224x224x3
-	Pretrained Model	
-3	Dense (Relu)	100
-2	Dense (Relu)	100
-1	Softmax	

As to not contaminate the dataset pool, the 100 sample slice discussed in section 4.2, was taken from the validation dataset of the MCB standardised testing folder. This is further split into the testing and training datasets that are used to train each of the networks.

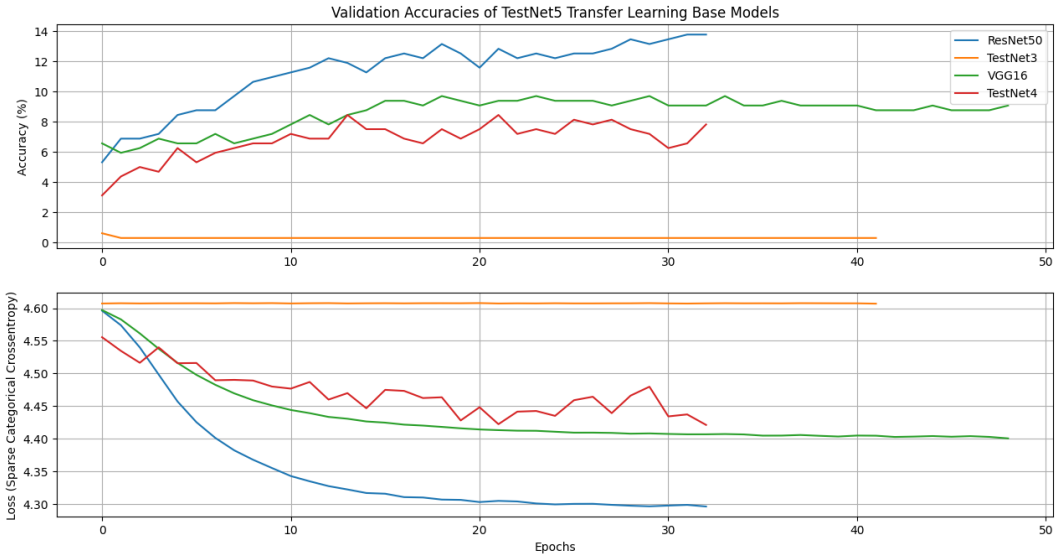


Figure 15 TestNet5 comparison of best loss values recorded

The above figure was produced by recording the loss and accuracy of the model on the test sample over the course of the training. When a new lowest total loss value was found the model weights and biases were saved as well as the settings and training recording up to that point. This means that it is not included in the above as this is only comprised of the best recorded results.

This plot indicates that the base model used in transfer learning has a huge effect on the performance of the model. It is surprising the extent that the base model makes in the output classification rate, even though ‘TestNet4' was trained on this style of input images the ResNet50 surpassed this despite being trained in a completely different context. This supports the hypothesis discussed in section 3.2 as the pretrained models seem to have the ability to classify parts with less emphasis on the lighting conditions or surface finish of the parts to a greater extent. This being said, the results presented here are far from optimal and the indications are therefore preliminary and should be treated with low credibility.

4.8 ‘TestNet6’

This test network is identical to the ‘TestNet5’ with the ResNet50 base model with the one difference being the size of the sample slice. In ‘TestNet5’ the sample slice was 100 parts, in this case that is increased to 250 parts. All other aspects are left the same as a control measure.

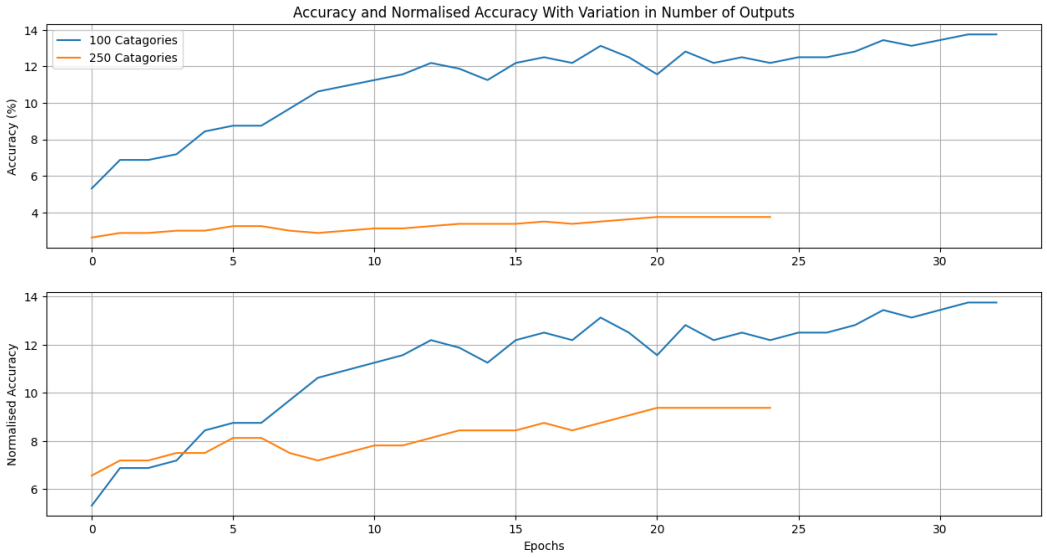


Figure 16 A direct comparison of the effect of local database size on classification accuracy

The above plot indicates that this approach dose not scale with size well. In fact, this show that not only is this the case but even when the two are compared relative to the random selection hit rate the ‘TestNet6’ variant scores significantly worse. A random hit rate refers to the assumption that a random selection in a 100 sample set will produce a one percent hit rate and 250 would yield 0.4 percent likewise. When applied to the accuracy values in the upper section of Figure 16 the accuracy value expressed as percentage is divided by the percentage expected hit rate giving the normalised accuracy.

Based on nothing more than the results displayed above it may be the case that this method is less effective on large datasets in a non-linier proportion. This may be the case that the reason for this is that the dataset is arranged in too sparse of arrangement where each sample only had 15 derived images from each part and thus there was proportionately less training per part in comparison to the dataset size.

4.9 ‘TestNet7’

‘TestNet7’ is continuation from the best variant of ‘TestNet5’ being the ResNet50 base where the training is continued where training the base model is enabled. This was run on the MCB 100 part slice with the same settings as in ‘TestNet5’ to emphasise the effect of fine tuning.

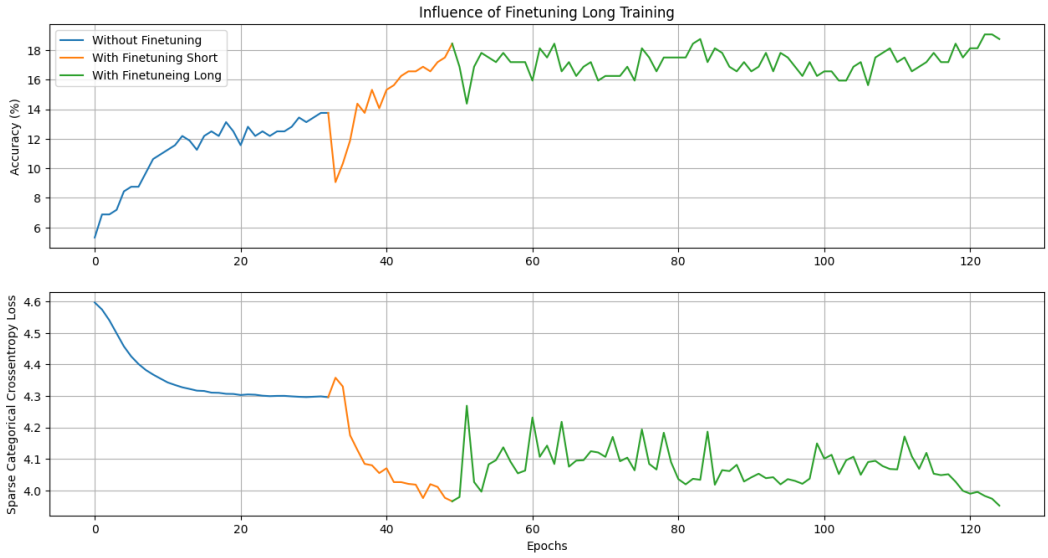


Figure 17 continued training of best TestNet5 with base model training enabled

In the above plot the blue line indicates the performance of TestNet5 where the only layers that can be trained are the output layers added to the ResNet50 base, the orange line is the best result where all layers are trained and green where the training as allowed to keep going to 200 epochs with a lower learning rate of 0.00001. Since only the best results from each are saved both do not show the regions where overfitting occurs. This indicates that much higher accuracy can be achieved by fine tuning. Yet the extent of this is limited, the best validation accuracy being 0.1875 percent from the initial 13.75 percent and intermediate 18.437 percent. The effects of this are further discussed in section 4.10.2.

The third stage, displayed in green trained for 200 epochs and did not yield a significant increase in accuracy on the validation dataset. It can be inferred that long training over the epochs is not necessary in achieving much higher results and only serves to increase overfitting even when a very low learning rate is used.

Although it was not directly measured it was evident that the training time increased dramatically when the fine truing was enabled. This would be of significant detriment to a client based product as the computation times would be drastically increased.

4.10 Testing and comparison of Test Networks

As a means of evaluating the performance of each of the test networks in the context of real-world use. A variety of images was collected that were known to have vary similar or matches in the 100 and 250 MCB sample slices. These are displayed in Figure 18 below. In this set of images an assortment of bolts are included as well as items that vary in identification difficulty. One such item that is intended to test the flexibility of the system is the colter pin, also often referred to as a split pin, of row two, column three. In the dataset this is already bent as it would be in use whereas the selected image shows it in its straightened configuration. Each of the images selected have already have a white background to limit the complexity of the task. Ideally the networks would not need this care taken in the style of input, however due to the classification accuracy of only approximately 18 percent this test had to be limited in scope.

Figure 18 Hand collected samples that also have like objects in the 100 part MCB slice

The compete findings of this experiment are presented in appendix 2.1 and 2.2 where the raw data output as well as images returned are displayed. These images in Appendix 2.2 should be interpreted as; input images

in columns one and four with the three returned results to the right of each in the order of best match, second best match and third best match.

4.10.1 Relevance of Softmax Output as a Relibable Indicator

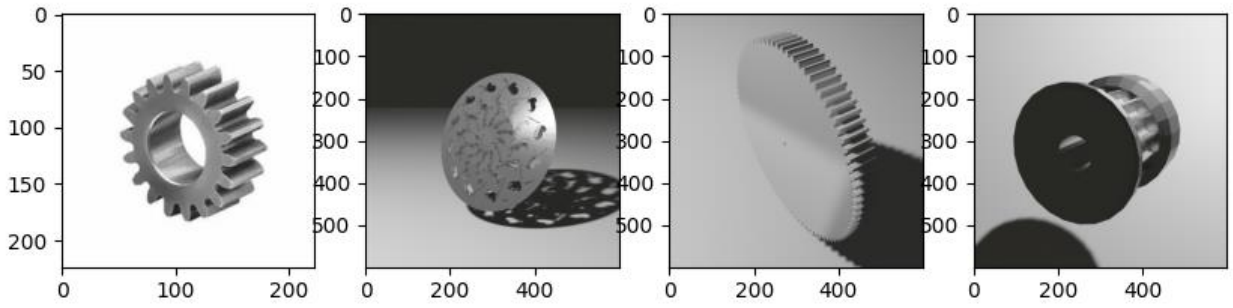


Figure 19 A sample of results returned from TestNet5:ResNet50

Above is a typical example of the top three results returned from 'TestNet5:ResNet50' when given the input image on the left of 'Gear_0'. From the left, the input image that is fed into the trained network, then the results with the highest Softmax output values from the left to the right. In this case, left to right in Figure 19 the Softmax output values are 0.1280, 0.067 and 0.0469, indicating that the network is operating with low certainty of the results that are output. In this case, there is no gear that directly corresponds to the shape of the input image. The closest match is legitimately the second result returned above. Some credit must also be given to the association of teeth that is made, as the sprocket on the right is also returned. The decorative plate may be the closest match, as its geometry appears round as well as jagged radially. Although not optimum, the result can be thought of as understandable and not random in nature. This means that the 'TestNet5:ResNet50' has clearly learned some geometric attributes of the parts supplied and is not random in the way that it operates.

Another point that is made evident by the test results of 'TestNet5:ResNet50', is that Softmax output correlates to accuracy. The average Softmax output value for an incorrect result was 0.1032 and for a relevant result it was 0.1962. An example of this Softmax output is in Appendix 2.1 'Flange_0' result one, where the returned results, as seen in Appendix 2.2, returns a flange that is nearly identical to the photo supplied. Thishis returns a value of 0.9833, as the network can answer the prompt with certainty. This means that the method as a whole has the of ranking outputs. As discussed above, this Softmax value is not particularly strong in its sorting ability when the outputs are low.

The visual presentation of Appendix 2.2 is done with the aim of highlighting the human-like identification style of each of these images and how close the identification is in a semantic sense. For instance, it is possible to empathise with the network as to why it often returns a helical spring when a bolt is input. Since both have

right had screw-like appearances the confusion between them is understandable. This semantic association between parts is one way to gauge the progress of the network as opposed to purely numerical means.

4.10.2 Disparity of Validation Accuracy to Tested Accuracy Across Testing Networks

Simply be visual inspection of the calibre of results returned in Appendix 2.2 it is evident that only one of the test networks stands out positively in meaningful results, being 'TestNet5:ResNet50'. Despite having a far lower validation accuracy of 13.75 precent in comparison to 'TestNet7's 18.437 percent, this sores significantly higher on the test with scores of 55.55 and 33.33 percent classification accuracy of including a relevant result in the top three results. Given that these two networks are identical in every way expect for the addition of fine tuning in 'TestNet7' the result here strongly indicates that finetuning increases the dependence on image context as opposed to the object in the image.

The reasoning for this may be that there is a large emphasis when training these types of CNNs that the subject of an image is the important aspect and not the rest of the image, the brightness of the image or any other cosmetic appearance that is not dependant on the object in question. This is usually trained out of the model with the use of data augmentation which seeks to randomly change the cosmetic appearance of the image so that the association of the image style or format is minimised. This is more prevalent on the instances where the images are constructed from renderings like they are in this case as there is far less naturally random background variation despite the effort that was taken and described in section 4.1 as well as 4.6. This is on display in this instance where the difference between validation accuracy of the test set of data, as distinguished in section 3.3, strongly disagrees with the accuracy on real world samples. Given that we know the image dataset that the ResNet50 was trained on, being the ImageNet competition dataset (Yang *et al.*, 2020) includes only real world data this hypothesis can be made with some certainty.

This effect may be combatted by increasing the photorealism of the rendered images and or the randomness o rendering parameters and background that the images. This may help break the association between the style of the produced images from the object in the image.

Given that one of the major aims of this project is to make progress on a client-based model that can use real world data inputs regardless of the style of the image input. This lack of transferability is of paramount concern considering the potential lack to the feasibility of this method being used in production. Further discussion of possible recommendations for further research to address this use will be made in section 5. Although the trained head will still yield more reliable results, it is unfortunate that the potential increase of accuracy that is often attributed to fine tuning is not applicable in this case.

Chapter 5 – Conclusions

Though this study may smaller lines of experimentation were attempted. Initial networks one through four tried to make a classifier that could be used for general categorical classification. These were used later in subsequent test networks as a benchmark for base model transfer learning. This architecture was the primary point of fucus and may experiments were derived to test the influence of singular factors at a time in controlled tests. To conclude, a had selected set of images were used to assess the validity of returned results based on a subjective measure of relevance.

Factors of interest in this study were the rendering prosses and performance of use, the influence of training data and quality of implementation, classification quality when built from the data available, comparison of base model influence on the output accuracy of the network, the effect of scale on the method performance, the effect of fine tuning as well as the transferability of context domain. These factors were mostly explored individually with the control of changing nothing but the effect of the factor of interest. This means that the results have some credibility even with a small sample size. Below are the findings in order of significance.

5.1 Comparison of Base Model Influence on The Output Accuracy

The largest unanticipated factor in this project was the effect that the base model would have in the accuracy of the part identification. It was expected that ‘TestNet3’ may have produced no good results, and ‘TestNet4’ - the best results - as it is trained, not only in the context of industrial component intuition but also on the same dataset style that the ResNet50 and VGG16 have not encountered before. The ResNet50 would be a close second. Yet the extent that ‘TestNet4’ performed worse than that of VGG16 as well as ResNet50 was surprising. Only two layers were added to the base model, as a trainable head meant that only a very small portion of the networks were being trained. The extent that generalised object understanding an intuition the pre trained models may be quite substantial. This gives credibility to the sentiment that client-side trainable lightweight models are feasible.

5.2 Effects of Fine Tuning on Transferability of Context Domain

The most definitive observation of this project was the disparity between increased validation accuracy and accuracy on a real sample set when an identical model used fine tuning of the base model and the other did not. This holds the implication that the fine tuning is not applicable to this style of model where synthetic images are created. As discussed in section 4.10.2. the decrease is proposed to be from the overwriting of the trained style dissociation that occurs in the training of the ResNet50 model that causes a drastic decrease in the transferability of the CNN.

5.3 Rendering Prosses, Model Realism and Performance

Since the strong indication given by the test in section 4.10 showed that the CNNs all did not perform well in terms of context transferability. The implication is that the rendering prosses used was not real enough in ether background setting or object presentation. This caused the networks to score relatively poorly in testing but particularly bad when the prior learning was overwritten in fine tuning of 'TestNet7'. This aspect of the project could be improved.

This may not be entirely the fault of the rendering not having enough photorealism but the low polygon count of the models lacking realism. No amount of rendering could present some of the objects particularly some of the bearings like in Figure 6. This could be easy to implement on an existing database that an industrial private company may be able to share as most CAD software has an export capability where the fineness can be adjusted.

The rendering process could be refined significantly. Given that this process took a considerably amount of time to set up for this project a potential client model would need to address this before it would be feasible as a product. This was demonstrated by showing the CPU and GPU over the course of a single iteration of an image rendering. Another thing that could improve the performance of this prosses could be to reduce the Blender program down to bare bones of only what was necessary for the rendering process and reduce the memory required.

5.4 The Effect of Scale on The Method Performance

Another area where this method seems to do poorly is when the number or parts is increased. This was considerably worse than the proportion that the output classification was scaled up by. This cannot be stated definitively though as the accuracy of the first model, 'Testnet5' was only 13.75 precent. This implies that further refinement of the process would need to be done before making this assessment.

5.5 Classification Quality of Constructed Models with Limited Data

Given that the dataset size of 75,393 images is smaller than that used to train the competing ResNet50 and VGG16, a humble result of 30.316 classification accuracy across 25 classes was achieved. As discussed in section 3.1 this challenge in unique in how difficult the sample data is to interpret. This demonstrates that the proof of concept is valid that the dataset and methodology can produce a model that can converge on less loss error. This is important as in the case that the method that this project set out to explore could not produce any promising results the point of failure could be blamed on the dataset or the model layer letup or training parameters. This is less relevant the described method produces some very valid results.

5.6 Influence of Training Data and Quality of Implementation

The development of 'TestNet2' showed that the contamination of data from the testing dataset to the training dataset can have huge implications on the reporting of the progress of the network. This is not an issue for the later network architectures as there are not categorical classifiers. Yet the importance of having a clean dataset is highlighted.

5.7 Part Obstruction and Ware Study

Given that the results of section 4.10 are not in the 90 percent classification rate. It would be premature to assess the change in accuracy from obscuring parts when the output results are partially questionable as is in some cases. This section should be addressed after further development and refinement of the overall method.

5.8 Further Research

To fully explore the topic of the proposed method. Some factors should be studied in more detail. Some suggestions are mentioned below. The field of machine learning not only relatively young but is notoriously indeterminate in the amount of variability that is present in construction of NNs and CNNs (Gulli, Kapoor and Pal, 2019). Given this some factors that may have merit researching further include;

- Implementing a rendering pipeline in Blender.
- Blender program reduction of features for performance.
- Increasing the randomness of appearance settings in rendering.
- Exploring more realistic rendering techniques to improve domain transferability.
- Exploring the effect of coloured images instead f grayscale for increased transferability.
- The pretrained model being of an industrial part classification nature unlike the ResNet50 and VGG16 models that are primarily oriented around classifying people animals and ordinary objects.
- Retrying fine tuning when the dataset is more photorealistic or more random in appearance.
- The effect of using transfer learning when more layers are removed from the top of the base model.
- The effect that further optimising the number of layer and layer shapes may have
- The effect to of changing the activation functions, optimiser function loss function.
- The effect that a larger database would have on a classifier model like that of 'TestNet4'.

References

Abadi, M. *et al.* (2015) *TensorFlow, Large-scale machine learning on heterogeneous systems*. Available at: https://doi.org/10.5281/zenodo.4724125.

Abadi, M. *et al.* (2016) 'TensorFlow: A System for Large-Scale Machine Learning', in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, pp. 265–283. Available at: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

Goodfellow, I. *et al.* (2014) 'Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks', *Street View and reCAPTCHA Teams, Google Inc.* [Preprint].

Gulli, A., Kapoor, A. and Pal, S. (2019) *Deep Learning with TensorFlow 2 and Keras*. 2nd edn. Birmingham, UK: Packt.

Haider, M.A. *et al.* (2019) 'Camera Model Identification using Deep CNN and Transfer Learning Approach', in, pp. 626–630. Available at: https://doi.org/10.1109/ICREST.2019.8644194.

Herculano-Houzel, S. (2009) 'The human brain in numbers: A linearly scaled-up primate brain', *Frontiers in human neuroscience*, 3, pp. 31–31.

Hinton, G.E., Osindero, S. and Teh, Y.-W. (2006) 'A Fast Learning Algorithm for Deep Belief Nets', *Neural computation*, 18(7), pp. 1527–1554.

Iyer, N. *et al.* (2005) 'Three-dimensional shape searching: state-of-the-art review and future trends', *Computer aided design*, 37(5), pp. 509–530.

Jayanti, S. *et al.* (2006) 'Developing an engineering shape benchmark for CAD models', *Computer aided design*, 38(9), pp. 939–953.

Kim, S. *et al.* (2020) 'A Large-scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks', in *Proceedings of 16th European Conference on Computer Vision (ECCV)*.

Leven, S. (1996) 'The roots of backpropagation: From ordered derivatives to neural networks and political forecasting', *Neural networks*, 9(3), pp. 543–544.

Michael, N. (2015) *Neural Networks and Deep Learning*. Determination Press.

Oliver, W. (2017) *Clearing Tensorflow GPU memory after model execution, Stack Overflow*. Available at: https://stackoverflow.com/a/44842044 (Accessed: 8 October 2022).

Qin, F. *et al.* (2014) 'A deep learning approach to the classification of 3D CAD models', *Journal of Zhejiang University. C Science*, 15(2), pp. 91–106.

Rucco, M. *et al.* (2019) 'A methodology for part classification with supervised machine learning', *Artificial intelligence for engineering design, analysis and manufacturing*, 33(1), pp. 100–113.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) 'Learning representations by back-propagating errors', *Nature (London)*, 323(6088), pp. 533–536.

Schmidhuber, J. (2015) 'Deep learning in neural networks: An overview', *Neural networks*, 61, pp. 85–117.

Simonyan, K. and Zisserman, A. (2014) ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’. arXiv. Available at: https://doi.org/10.48550/ARXIV.1409.1556.

USQ Safety Risk Management System (2022).

Werbos, P.J. (1990) ‘Backpropagation through time: what it does and how to do it’, *Proceedings of the IEEE*, 78(10), pp. 1550–1560.

Yang, K. *et al.* (2020) ‘Towards Fairer Datasets: Filtering and Balancing the Distribution of the People Subtree in the ImageNet Hierarchy’, in *Conference on Fairness, Accountability, and Transparency*. Available at: https://doi.org/10.1145/3351095.3375709.

Zajac, A. and Hecht, E. (2003) *Optics*. 4th edn. Pearson Higher Education.

Appendices

A1 Project Specification

ENG4111/4112 Research Project

Project Specification

For: Joshua Garry

Title: Asset classification with deep learning from computer added design parts

Major: Mechanical Engineering

Supervisors: Dr Tobias Low

Enrollment: ENG4111 – EXT S1, 2021

ENG4112 – EXT S2, 2021

Project Aim: To explore the possibilities of identifying assets, mainly CAD model parts using elements of machine learning that will be capable of outputting identifiable information about the part being filmed or photographed based on database information feed into the model. This is aimed at being feasible to implement for industrial end consumers.

Programme: Version 2, 17th March 2021

1. Define and understand the fundamental operating principles of deep image classification models.
2. Review and assess the existing methods of achieving similarly defined problems in the literature and in available resources.
3. To conduct initial background research deep learning models emphasizing architectures and implementation strategies used to identify objects.
4. Reproduce existing methods or architectures in code that can output expected results given known inputs as validation.
5. Evaluate the performance of the implemented code and identify areas of potential improvement for the given task.
6. Prototype the first intermediate configuration and train parts on the one-hot output format.
7. Prototype the second configuration using a generalized output format more suitable or real-world implementation.
8. Further validate the numerical models using data from other sources of CAD model such as from online databases.
9. Write and draft thesis paper.

If time and resource permit:

1. Code a model capable of identifying parts that are partially obstructed.
2. Code a model capable of identifying parts that are worn or damaged.
3. Implement a real time camera application to demonstrate proof of concept.

A2 Experiment Results

A2.1 Output Performance of Test Networks on Hand Selected Samples Numeric Data

Table 11 Test Network outputs on real collected images

	TestNet5:ResNet50		TestNet5:TestNet3		TestNet5:VGG16		TestNet5:TestNet4		TestNet6		TestNet7		TestNet7.1	
	Certainty	Correct	Certainty	Correct	Certainty	Correct	Certainty	Correct	Certainty	Correct	Certainty	Correct	Certainty	Correct
Bearing_0	0.01	0	0.011327	0	0.862873	0	0.010287	0	0.065059	1	0.023557	0	0.01	0
	0.01	0	0.010503	0	0.003019	0	0.009997	0	0.013905	0	0.009863	0	0.01	0
	0.01	0	0.010351	0	0.001368	0	0.009997	0	0.003924	0	0.009863	0	0.01	0
Bearing_1	0.063682	0	0.011607	0	0.011325	0	0.01	0	0.011603	0	0.01	0	0.062417	0
	0.039156	0	0.010682	0	0.009987	0	0.01	0	0.006408	1	0.01	0	0.062295	0
	0.029417	0	0.010546	0	0.009987	0	0.01	0	0.00396	0	0.01	0	0.058052	0
Bearing_3	0.395746	0	0.011607	0	0.010461	0	0.01	0	0.05559	0	0.01	0	0.023252	0
	0.133545	1	0.010682	0	0.009995	0	0.01	0	0.022869	0	0.01	0	0.02251	0
	0.092513	0	0.010546	0	0.009995	0	0.01	0	0.018369	0	0.01	0	0.017445	0
Bolt_0	0.228902	1	0.011607	0	0.081062	0	0.01	0	0.010814	0	0.865021	0	0.542013	0
	0.176344	0	0.010682	0	0.009282	0	0.01	1	0.004561	0	0.032084	0	0.082592	0
	0.110354	0	0.010546	0	0.009282	1	0.01	0	0.004116	0	0.023068	0	0.048904	1
Bolt_1	0.296432	0	0.011607	0	0.035473	1	0.018257	0	0.006953	0	0.010105	1	0.087784	0
	0.104031	0	0.010682	0	0.022658	0	0.010056	0	0.005871	1	0.009999	0	0.056161	0
	0.080105	1	0.010546	0	0.022431	1	0.009915	0	0.003981	0	0.009999	1	0.020429	0
Bolt_2	0.203692	0	0.011607	0	0.135246	1	0.01	0	0.013486	1	0.010722	0	0.165714	1
	0.059095	0	0.010682	0	0.03151	1	0.01	1	0.006132	1	0.009993	0	0.163926	0
	0.031963	1	0.010546	0	0.020382	0	0.01	0	0.004186	0	0.009993	1	0.102467	0
Bolt_3	0.232403	1	0.011607	0	0.083205	0	0.014897	0	0.0109	0	0.295243	0	0.245346	1
	0.178251	0	0.010682	0	0.009261	0	0.009951	0	0.004627	0	0.217536	1	0.178716	0
	0.110652	0	0.010546	0	0.009261	1	0.009951	1	0.00416	0	0.088536	0	0.116003	0
Bush_0	0.089239	1	0.011606	0	0.01	0	0.02798	1	0.005662	0	0.01	0	0.621829	0
	0.058514	0	0.010682	0	0.01	0	0.0227	1	0.003993	0	0.01	0	0.356196	0
	0.020549	0	0.010544	0	0.01	0	0.02265	1	0.003993	0	0.01	0	0.000263	0
Cotter_Pin_0	0.110447	0	0.011607	0	0.01	0	0.069813	0	0.007316	0	0.010342	0	0.01	0
	0.104363	0	0.010682	0	0.01	0	0.052275	0	0.005416	0	0.009997	0	0.01	0
	0.041691	0	0.010546	0	0.01	0	0.012299	1	0.003981	0	0.009997	0	0.01	0
Flange_0	0.983375	1	0.011591	0	0.32187	0	0.01	0	0.077727	0	0.594985	0	0.753628	0
	0.012284	0	0.010675	0	0.060951	0	0.01	0	0.003704	0	0.403544	0	0.24618	0
	9.23E-05	0	0.010514	0	0.00765	0	0.01	0	0.003704	0	0.000117	0	2.57E-05	0
Gear_0	0.128008	0	0.011607	0	0.01	0	0.01	0	0.004	0	0.05889	1	0.046249	0
	0.066897	0	0.010682	0	0.01	0	0.01	0	0.004	0	0.018431	0	0.017179	1
	0.04687	1	0.010546	0	0.01	0	0.01	0	0.004	1	0.009415	0	0.014927	1
Handle_0	0.075898	0	0.011593	0	0.041323	0	0.070487	0	0.019734	0	0.010204	0	0.015174	0
	0.029256	0	0.010685	0	0.019441	0	0.031097	0	0.009639	0	0.009998	0	0.010672	0
	0.015892	0	0.0104	0	0.009584	0	0.009168	0	0.008277	1	0.009998	0	0.00994	0
Nut_0	0.225933	0	0.011607	0	0.010293	0	0.022028	0	0.010899	0	0.13144	0	0.011698	0
	0.172455	0	0.010682	0	0.009997	0	0.009879	0	0.004562	0	0.072147	0	0.009983	0
	0.109654	0	0.010546	0	0.009997	0	0.009879	0	0.004092	0	0.026154	0	0.009983	0
Plumbing_Elbow_0	0.147167	0	0.011545	0	0.952253	1	0.01	0	0.040828	0	0.01	0	0.016207	0
	0.050756	0	0.01067	1	0.001134	0	0.01	0	0.007558	1	0.01	0	0.011022	0
	0.033729	0	0.010282	0	0.000476	0	0.01	0	0.003837	0	0.01	0	0.009926	0
T_Slot_0	0.023921	0	0.011607	0	0.241601	0	0.145304	0	0.004	0	0.01	0	0.478045	0
	0.019455	0	0.010682	0	0.068502	0	0.010948	0	0.004	0	0.01	0	0.075014	0
	0.018774	0	0.010546	0	0.061846	0	0.00861	0	0.004	0	0.01	0	0.017769	0
Washer_0	0.490808	0	0.011607	1	0.01	1	0.01	1	0.574935	0	0.50142	1	0.649495	1
	0.269258	0	0.010682	0	0.01	0	0.01	0	0.03484	0	0.481907	1	0.106655	1
	0.076494	1	0.010546	0	0.01	1	0.01	1	0.002005	1	0.000799	0	0.009902	1
Washer_1	0.154339	1	0.011484	1	0.01	1	0.027925	0	0.018755	1	0.581517	1	0.674185	1
	0.101201	1	0.010673	0	0.01	0	0.010183	0	0.012488	1	0.385429	1	0.321242	1
	0.01516	0	0.009978	0	0.01	1	0.009815	1	0.01003	0	0.002966	0	0.001181	0
Includes a Relevant Result		0.5555		0.16666		0.3888		0.3888		0.5		0.3333		0.3333
Fist Result Relevancy		0.277778		0.111111		0.277778		0.111111		0.166667		0.222222		0.222222

This table was made by inputting each of the test samples into each of the test networks and ranking the outputs to get the top three. These top results were compared manually to the input image and scored on the basis of relevantly similar or not. This means that the test is a closer assessment of its real world capability for the use that is outlined in the scope of this document in section 1.2. By assessing the relevancy of the top three returned results as opposed to just the one output a more complete picture of the networks can be appreciated.

A2.2 Images Corresponding to outputs of Test Networks Hand Selected Samples

This section presents the same experiment as Table 11 but in a more visual format. These images are arranged with the first, and fifth column being the input image and the three immediately to the of each being the top results be measure of Softmax output.

Figure 20 Input to output comparisons of TestNet5:ResNet50

Figure 21 Input to output comparisons of TestNet5:VGG16

Figure 22 Input to output comparisons of TestNet5:TestNet4

Figure 23 Input to output comparisons of TestNet5:TestNet3

Figure 24 Input to output comparisons of TestNet6

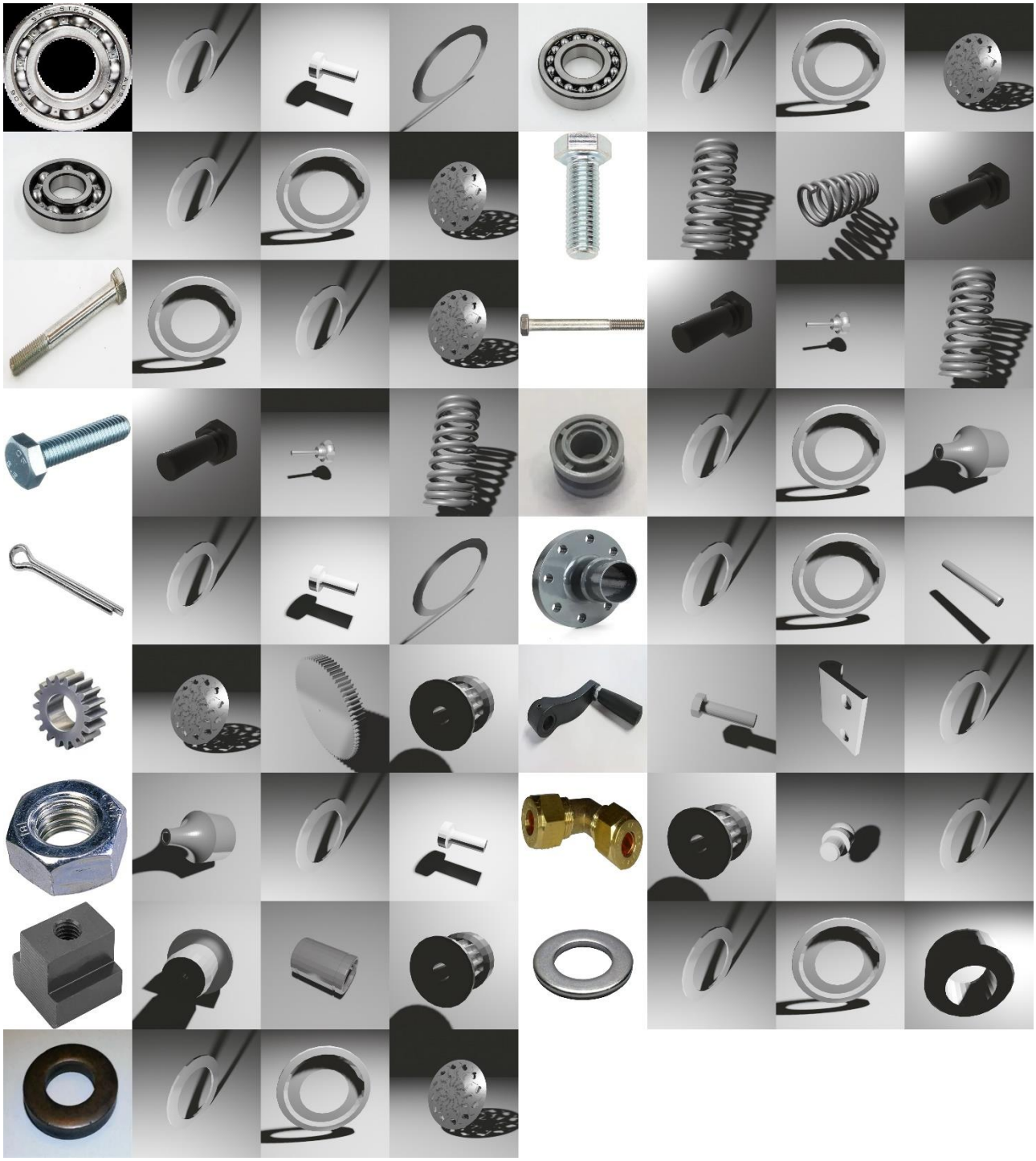


Figure 25 Input to output comparisons of TestNet7 low epochs

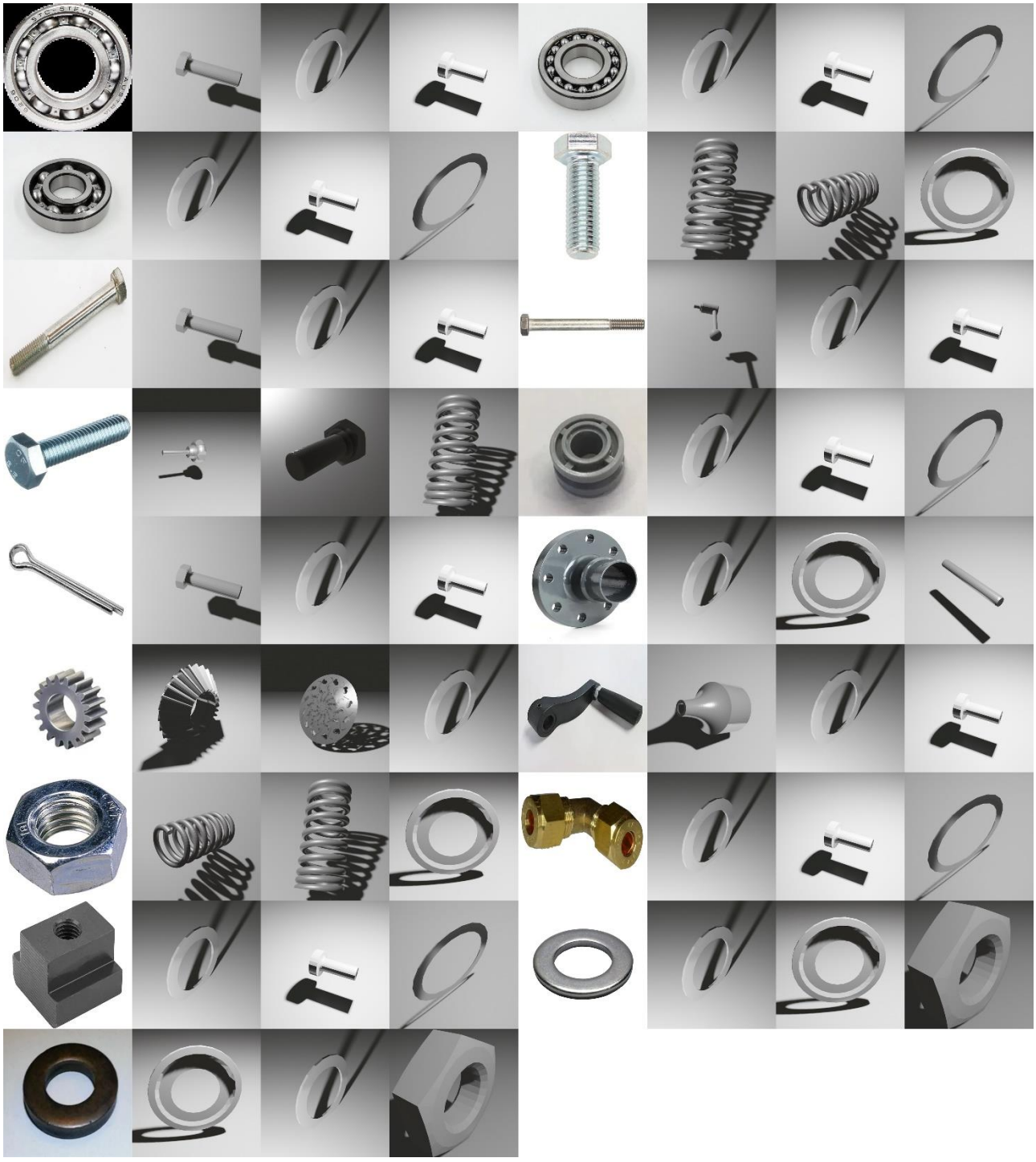
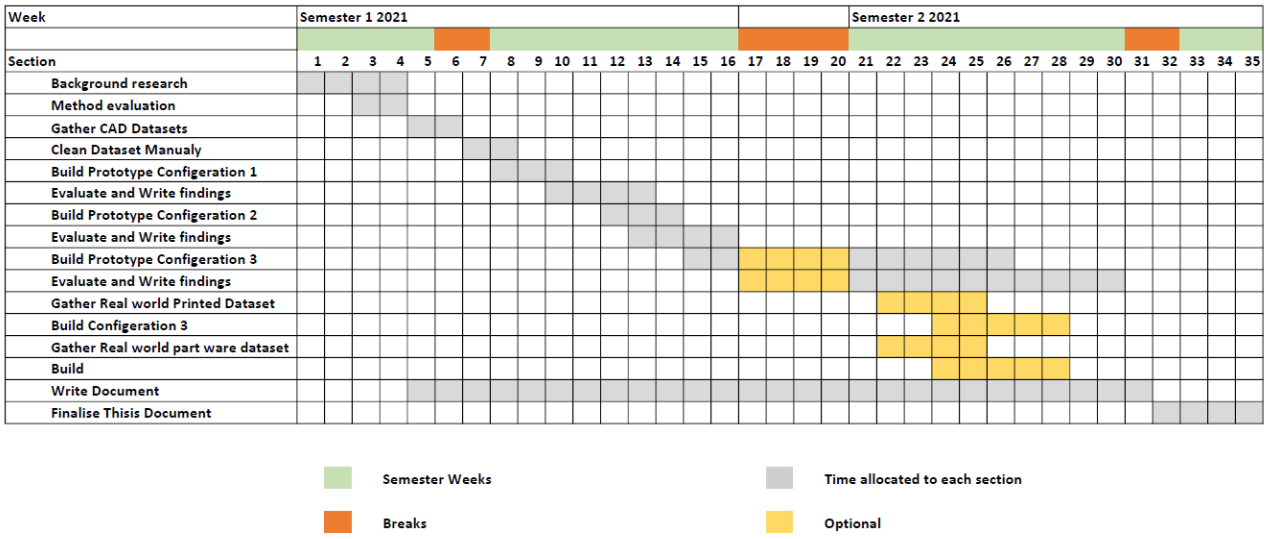


Figure 26 Input to output comparisons of TestNet7 high epochs

A3 Timeline

Table 12 Research Proposal Specification Shedual



A3 Resources

Table 13 Resource Anticipation Cost

Component	Cost	Quantity	Estimated Cost
Access to USQ's NVIDA server	No Addition Cost		$0

A5 Risk Assessment

Risk Matrix					
	Consequence				
Probability	**Insignificant** No Injury 0-$5K	**Minor** First Aid $5K-$50K	**Moderate** Med Treatment $50K-$100K	**Major** Serious Injury $100K-$250K	**Catastrophic** Death More than $250K
Almost Certain 1 in 2	M	H	E	E	E
Likely 1 in 100	M	H	H	E	E
Possible 1 in 1,000	L	M	H	H	H
Unlikely 1 in 10,000	L	L	M	M	M
Rare 1 in 1,000,000	L	L	L	L	L
Recommended Action Guide					
Extreme:	**E**= Extreme Risk – Task ***MUST NOT*** proceed				
High:	**H** = High Risk – Special Procedures Required (Contact USQSafe) Approval by VC only				
Medium:	**M**= Medium Risk - A Risk Management Plan/Safe Work Method Statement is required				
Low:	**L**= Low Risk - Manage by routine procedures.				

Figure 27 Risk assessment weight matrix (USQ Safety Risk Management System, 2022)

Table 14 Risk Assessment Table

Hazzard	**The Risk**	**Consequence**	**Probability**	**Existing Controls**	**Risk Level**	**Additional Controls**	**Amended Consequence**	**Amended Probability**	**Risk Level**
Damaging USQ IT Resources	Crashing Machine	Minor	Unlikely	USQ Staff Oversight	Low	Care will be taken not to write careless code	Minor	Unlikely	Low

A6 Ethics

As part of ongoing practice as an engineer engaging in continuing professional development, ethics are and continue to be of major importance. Below is a sighed copy of the code of conduct required in ENG4903 Professional practice 2.

USQ Engineering Professional Practice Conference
Code of Conduct: PROFESSIONAL AND PERSONAL ATTRIBUTES [1]

The following shall be the agreed standard of behaviour at all times, and it shall be observed by all conference attendees.

Ethical conduct and professional accountability during conference attendance as demonstrated by:

- **Not engaging in fraudulent, corrupt or criminal conduct, but engaging in the program** in accordance with USQ statutory requirements and with professional engineering commonly accepted standards.
- **Understanding and application of** 'due-diligence' by careful analysis and weighted evaluation in Professional Practice course processes.
- **Awareness of** the fundamental principles of intellectual property rights and protection and obligations. This includes referencing of relevant presented or published works and data in presentations.

Use of effective oral and written communication during the conference as demonstrated by being as proficient as possible in listening, speaking, reading and writing, including:

- Respectfully comprehending critically and with fairness the viewpoints of others;
- Engaging in discussion, expressing views / information effectively and succinctly,
- Presenting arguments and justification, debating and negotiating contents of technical presentations;
- Using appropriate textual, diagrammatic, pictorial and graphical media best suited to the context;
- Appreciating and limiting the impact of body language, personal behaviour and other non-verbal communication processes, as well as observing the fundamentals of decent and respectful human social behaviour, especially in regard to any cross-cultural differences.

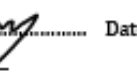
Being creative and innovative with a pro-active demeanour as demonstrated at the conference by:

- **Utilising the opportunity for learning of** new developments in the engineering discipline and specialisations.
- **Further developing your awareness of** broader fields of science, engineering, technology and commerce from which new ideas and interfaces may be drawn; and readily engaging with multiple engineering disciplines to exchange ideas.

Engaging in Orderly management of oneself and professional conduct at the conference by demonstration of:

- **Critical self and peer review and performance evaluation** against appropriate criteria as a primary means of tracking personal development needs and achievements at the conference.
- **Being a participant** in the professional and intellectual community at this conference.
- **Managing** time and processes effectively, **while prioritising** competing demands to achieve personal, and future career goals and objectives as part of this conference.
- **Presenting** a professional image in all circumstances, including relations with fellow conference participants regardless of culture of origin or engineering discipline.
- **Be considerate, respectful, and collaborative**, always mindful of your surroundings and of fellow participants, paying attention during the presentations and refraining from surfing the net, playing games, dozing off etc..
- **Attendance of all and completion of each of the sessions**, not departing the room without scanning out, always and only using your own conference ID barcode at every instance.
- **Participation as an** effective conference member in the diversity of engineering attendees, including those with multi-level, multi-disciplinary and multi-cultural dimensions.
- **Timely attendance and completion of all conference tasks**.
- **Recognising** the value of alternative and diverse viewpoints, diverse cultural perspectives, respectful scholarly debate and the importance of professional networking.
- **Present a professional image (i.e. this includes appropriate engineering business-like dress code)**, in all circumstances with fellow stakeholders across a wide range of engineering disciplines.

I Joshua John Garry ***understand that failing to comply with this code of conduct may be treated as academic misconduct and/or may result in fail grade in this course. As an enrolled conference member, I am aware that failing to successfully complete this course might delay my graduation by one year.***

Signature:  Dated: 05/ 03 /2021

[1] Extracted and paraphrased from Engineer's Australia, *STAGE 1 COMPETENCY STANDARD FOR PROFESSIONAL ENGINEER, Section 3. PROFESSIONAL AND PERSONSL ATTRIBUTES*, 2012 © Engineers Australia

A7 Quality Assurance

As a responsible and ethical engineer, it is a responsibility to predict possible harm that may be caused by their actions. One of which is the quality assurance of the work that is published as being complicit in misleading or falsifying results that may be incorporated in other people's work. The damage this includes can be in the form of financial harm, the safety of persons or the loss of time.