Detecting Facemasks with Deep Learning

Kenton Lam* Matthew Low† Nicholas Mann‡ Oliver Sutton§

April 10, 2021

Abstract

A method for detecting facemasks with deep learning for the purposes of health & safety was investigated. Various neural network architectures, such as multilayer perceptrons, convolutional neural networks and transfer learning were used, as well as image transformation techniques such as greyscaling and Canny edge-detection. A custom convolutional neural network architecture achieved 99.04% test accuracy on a dataset of masked and unmasked faces from the MaskedFace-Net and Flickr-Faces-HQ datasets. *We give consent for this to be used as a teaching resource.*

1 Problem

The global COVID-19 pandemic continues to ravage countries around the world with over 40 million people infected and over 1 million deaths [4]. As recommended by the World Health Organisation, and numerous state and local governments, face masks, when worn correctly, present an effective solution to limiting the spread and reducing the risk of the virus. This is because COVID-19 can be transmitted to people who are more than 2 metres away, meaning the use of face masks greatly decreases the rate of transmission [25]. Most notably, it has been estimated that the rate of infections decreased by 3% per day in New York City after face masks were required to be worn, totalling to 60,000 reduced infections from 17 April to May 9. Meanwhile, there was an estimated reduction of 78,000 infections in Italy from April 6 to May 9 [28]. Yet, facemasks are still commonly being misworn by many members of the community, and enforcement of facemasks is a tough process. As seen in Figure 1, there are multiple ways that a mask can be misworn:

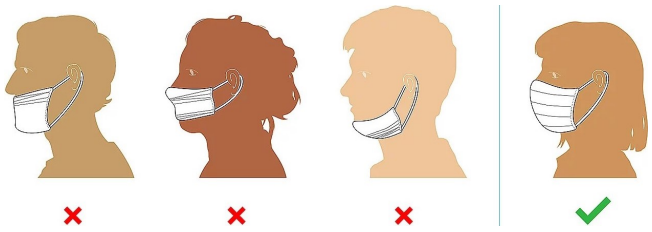

Figure 1: Examples of incorrect mask wearing [20].

To solve this problem of incorrect face mask wearing, we seek to develop a machine learning model using varying neural network architectures to not only detect whether an individual is wearing a face mask, but if its being worn correctly or not. This is a multi-class classification problem where the classes are based on how, and if, a mask is worn correctly.

Several existing models are able to detect face masks [17, 7], however not much work has been done on detecting whether face masks have been worn *correctly*. We aim to address this gap as the detection of correctly-worn face masks is arguably just as important to the prevention of disease spread since incorrectly worn masks do not necessarily reduce the spread of the virus.

*The University of Queensland, Student Number `<REDACTED>`.
†—, Student Number `<REDACTED>`.
‡—, Student Number `<REDACTED>`.
§—, Student Number `<REDACTED>`.

The classes are neatly described in Figure 2.

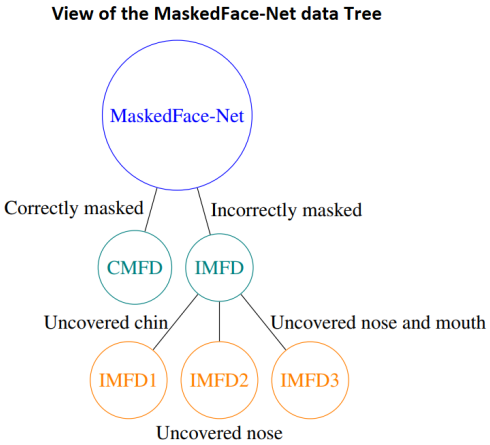

Figure 2: The different classes that our model will predict.

We seek to produce a detection model that:

- isn't overfitted to the training data;
- accurately predicts how people are wearing masks; and,
- predicts on images of us wearing masks to see how the model performs on actual images, such as the below images:

Figure 3: The model will predict the classes of similar images later on. (Nick left, Matthew right)

The dataset that will be used for investigation and training the neural network models is the *MaskedFace-Net* dataset [3]. The dataset was created by computationally Photoshopping masks onto images of people. It contains images of correctly and incorrectly worn images. This will be combined with the *Flickr-Faces-HQ* dataset [9] to allow the model to learn mask-less images.

2 Significance

The project is significant on three main levels:

- As previously evident, correctly worn masks are vital for public health. Once a model is built, and performs to a sufficiently high accuracy, it would be combined with camera software and deployed to high pedestrian traffic areas to detect how face masks are worn. This could then notify police or business owners that people aren't wearing masks, resulting in further consequences taken up.
- Aside from the COVID-19 significance, the project demonstrates how to handle large amounts of image data.
- The solution framework can be applied to any problems involving the use of PPE, such as construction sites or prior to entering operating theatres.

This project's relevance is constantly increasing as more countries are facing prolonged lockdown stages, with little COVID-19 cases decreasing. Notably, France and Germany have gone back into lockdown as virus cases soar in Europe [26]. By combining this model with suitable camera technology, people who aren't wearing masks correctly could be dealt with according to local face mask laws.

3 Solution

To determine the most optimal solution to this problem, the approach was broken up into 4 main sub areas:

1. Find a suitable training environment for training the neural networks and storing the data;
2. Implement suitable image size reduction techniques to appropriately handle the large dataset;
3. Determine a suitable loss function and optimiser for the classification problem; and,
4. Experiment on different types of neural networks, such as Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), and transfer learning networks that can be suitably applied to this problem space.

These are the main aspects of the problem and finding the best way to solve each sub area would help find the best solution. Each one of these dot points was then assessed on the following criterion:

- Computational efficiency: how long did it take to train the model / upload data / perform image reduction techniques?
- Amount of storage it took up: how much storage do the models / images take up?
- Cost: are there any associated costs?
- Consistency: how consistent are the results / networks / image reduction techniques / training environments?

3.1 Creating a Suitable Training Environment

The dataset was roughly 120 gigabytes in size which indicated a few initial issues. Firstly, if training was to be done on such a large dataset, it would have to be stored somewhere large enough and be efficiently accessed by the Python kernel as the images are loaded into the dataloaders. Secondly, with the potential for a large amount of training that would need to be performed, there was a high potential for it to take a very long time.

To solve these issues, some research was undertaken to investigate different training environments. Both free and paid options were investigated. Listed below are some of the findings, and a brief analysis on their suitability, with the chosen platforms in **bold**:

- **Google Colab + 15GB of Google Drive space**
  - Cost: free
  - Findings: this was a reliable training environment that was very familiar to us from previous STAT4402 assignments. Empirical analysis found similar performance with lower-resolution images (more to come in the later sections).
  - GPU: A selection from K80s, T4s, P4s and P100s. Note that a specific GPU cannot be specified in Google Colab, meaning that training times may vary from run to run.
  - Can be run continuously for 12 hours, although was not consistent.
- **Google Colab Pro + 15GB of Google Drive space**
  - Cost: $15 a month.

  - GPU: Similar to regular Colab, a selection from K80s, T4s, P4s and P100s. Just like Colab, a specific GPU cannot be specified, meaning that training times may vary from run to run.
  - Findings: we chose to also go with this option as it gave better reliability on GPU-intensive tasks.
- Google Colab Pro + 100GB of Google Drive space
  - Cost: $2.49 + $15 a month.
  - Findings: we found the additional space unnecessary for our purposes as we were using only the smaller images for our network.
- **Paperspace 200GB**
  - Cost: ≈$8 a month USD + utilisation cost.
  - Findings: empirical comparisons with Google Colab free instances showed little difference in training performance. There were also reliability issues with the Paperspace service, such as terminating or hung instances.
- AWS SageMaker with GPU-accelerated notebook instance
  - Cost: ≈$5 an hour.
  - GPU: GPUs vary depending on the type of instance chosen.
  - Findings: this is quite expensive considering the GPU acceleration is always on regardless of utilisation.
- AWS SageMaker with separate training instances
  - Cost: ≈$3 an hour
  - GPU: GPUs vary depending on the type of instance chosen.
  - Findings: although this would be inexpensive compared to the GPU-accelerated notebook instance, this requires the usage of a suite of AWS-based tools which we were not familiar with, so we chose not to pursue this option.

From our analysis, Google Colab, its Pro Version and Paperspace were used as the training environments. The performance on each system was relatively consistent, although some issues with usage limits on the free version of Colab made running experiments for a long period of time quite difficult. This was remedied by ensuring that Colab tabs stayed open and were monitored for the duration of any experiments, as idle timeouts would cancel code runs. AWS SageMaker was too expensive for the training phase.

3.2 Image Size Reduction Techniques

As previously indicated, the dataset was approximately 210,000 images each with 1024 by 1024 pixels dimensions amounting to roughly 120 gigabytes of data. It was also found from prior analysis that each mask was uniform in colour and texture across all of the images. This isn't a big surprise since the MaskedFace-Net dataset was generated by computationally Photoshopping masks onto the Flickr-Faces-HQ dataset. This issue indicated that the model could potentially overfit and not learn the mask's general shape, and rather be extremely dependent on the mask's colour and texture. As a result, this would prevent the model from generalising to other types of masks such as black masks or white masks. Below, are four images: the first two images are example photos and the final two images are different kinds of masks that could be worn out in the public. Note, there are many other types of masks that could be seen.

Figure 4: Left two images: images from dataset. Right two images: alternative masks

It was impractical to train on the complete dataset, as the training environments could not contain the dataset in a cost effective manner. To overcome these issues, the images were initially resized to 128×128 pixel images, reducing the dataset to 3.1 gigabytes. An additional dataset was created by applying greyscale transformations to the images to see how neural networks would perform images where the mask was less noticeable and uniform. A third dataset was created by applying Canny edge-detection to the images to identify the main features of the mask such as edges and folds. By performing the greyscale and Canny edge-detection, the datasets were further reduced to 1.7 and 1.5 gigabytes respectively - a very successful reduction in size! The initial hypothesis was: if the model can perform well on Canny edge-detection images, then it should be able to generalise to other kinds of masks as it is learning the shape of the mask. This will be explored later on. To find the best model, training was then performed on each of the three datasets: the original RGB images, the greyscale images, and the Canny edge-detection images.

Figure 5: Normal, Greyscaled, and Edge-Detected images

Another issue faced by the dataset was the imbalance in the incorrect mask sub-classes as indicated by the two piecharts below:

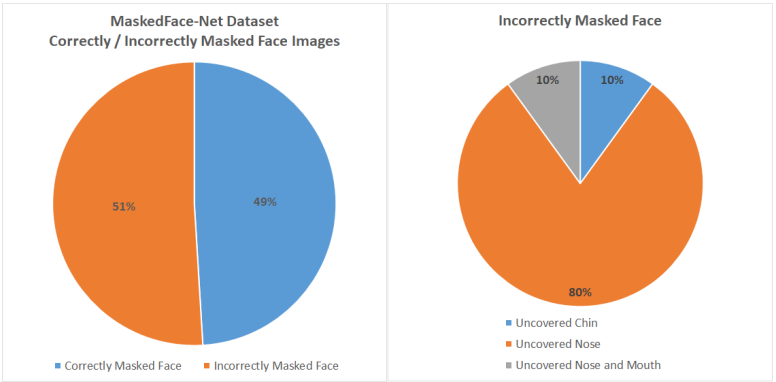

Figure 6: Pie chart showing the make-up of the MaskedFace-Net dataset. The number of Flickr-Faces-HQ images is roughly equal to the number of correctly masked images.

The concern about this is that the model might not have a large enough dataset to learn the features of the less represented class, and it might not be able to differentiate between images that have similar

features. This issue could have been solved by reducing the size of the other subclasses so that there is an even spread between all the subclasses, however this would mean we would have less samples for the model to learn on. One possible area of interest is to see how the model performs on these lesser represented subclasses when the number of other samples is decreased or the number of samples in the lesser represented subclasses is increased.

In terms of the classification problem, the training set, $\{\tau_i = (\boldsymbol{x}_i, y_i)\}$, will consist of feature vectors, $\boldsymbol{x}_i \in \mathbb{R}^N$, and target vectors, $y_i \in \mathbb{N}$. The length of the feature vectors, N, will depend on the network that the image is trained on. In the case of the MLP, the vector will have to be one dimensional and will have size $N = 49152$ (for RGB images) and size $N = 16384$ (for greyscale images). This is because each row in each channel of the image is connected end-to-end and treated as the input vector. For the CNN and transfer learning models, the feature vectors will consist of $\mathbb{R}^N \times \mathbb{R}^N$ vectors. The RGB images will have 3 channels of $\mathbb{R}^N \times \mathbb{R}^N$ while the grey scale images will have 1 channel of $\mathbb{R}^N \times \mathbb{R}^N$. In both cases, $N = 16384$ representing the pixels of the 128×128 images.

The target vectors will be the class number they belong to. For example, an image that is correctly wearing a mask is classified as $y = 1$. For future reference, the classes will be denoted by:

- No mask: 0
- Correct mask: 1
- Mask covering nose and mouth: 2
- Mask covering mouth and chin: 3
- Mask covering chin only: 4

3.3 Optimiser and Loss Function Experimentation

Unfortunately, there is no easy way of determining the set of hyper parameters, loss function or optimiser that gives the best results for a given problem. Of course, it is possible to be close to the optimal selection, but this requires a lot of computation, time, and exploration of different combinations. Ideally, optimal combination should achieve the best accuracy and is also time and computationally efficient. To solve this issue, empirical tests were undertaken as they provide a sufficient way to find suitable hyper parameters, loss functions and optimisers.

Before describing the results from the tests, it is important to mention that whenever free deep learning training software is used, GPU availability might fluctuate and the performance might not be consistent giving different results. Regardless, the following empirical tests provide a solid baseline into what set-up works well.

To begin the empirical testing, AlexNet [11] was chosen as the testing architecture and the CIFAR-10 data set [10] was chosen as the training data set. The choice of AlexNet was because it is a large neural network and it should be able to indicate how the optimisers would perform on large neural networks if we decided to build and test one in our solution. Additionally, the number of images in the CIFAR-10 data set, being 60,000 32×32 images, is some what comparable to the face mask data set. Although there are many more classes in the CIFAR-10 data set and it does not have images of people, it will still provide a good indication as to how the optimisers and loss functions perform on a similar problem.

A description of the available PyTorch library optimisation algorithms by [24] describes the best generic optimisers. This formed our decision on which optimisers to use for the empirical testing. Although RMSProp performed well for [24], it did not significantly reduce the error rate after 15 epochs of our test scenario. This was possibly due to a bug in the RMSProp optimiser or misconfigured hyper parameters. As a result, it was not tested any further. The following optimisers will be tested:

- SGD: Stochastic Gradient Descent with a learning rate of 0.001 and momentum of 0.9. This has performed very well for previous STAT4402 assignment questions.

- Adam: Adam Optimiser with default parameters. This optimiser is slowly becoming one of the most used and is fairly robust for any hyper parameter selection. [5]
- AdaGrad: AdaGrad optimiser with default parameters.
- AdaDelta: AdaDelta optimiser with default parameters. This was derived from the AdaGrad method but improves on some drawbacks of the method. [27]
- AdamWF: The AdamW optimiser is a variant of the Adam optimiser that utilises decoupled weight decay regularisation [18]. Empirical evidence by [18] has shown that weight decay improves Adam's generalisation performance.
- AdamWT: The AdamW optimiser but with improved convergence parameters by using the AMSGrad variant. This uses "long-term memory" of the past gradients [23].
- AdamAK: Adam Optimiser with a smaller learning rate of 0.0003. A tweet by Andrej Karpathy [8] joked about this being the best learning rate. Despite this, the experiments indicate that this works well.

To test which loss function performs the best, it was decided to test the Cross-Entropy (CE) Loss function and Mean Square Error (MSE) Loss function. These two loss functions have been used in prior STAT4402 studies, but also there has been a large amount of success with these loss functions in multiple areas of neural networks. Other loss functions do exist, however, they will not be tested. From research, the CE loss function was the most common type of loss function amongst classification problems [2].

Another parameter that can be tested is the batch size. It is possible that some optimisers work better with varying batch sizes as it is possible that larger batch sizes allow for faster convergence to the true solution, or perhaps they work worse as there are too many parameter updates required at once. Due to limitations in the amount of RAM available on many of the free training environments, the largest batch size that will be tested is 1000. When larger batch sizes, such as 2000 were used, there were persistent crashes.

An optional parameter called `num_workers` in the `DataLoader` object will be investigated. When the parameter is zero, the data is loaded inside of the main process meaning that the training phase is sequential inside of the main process, and once a batch has been trained on, another batch is read in from the disk. But, when this parameter is non-zero, it creates sub-processes that load the data, thus making use of the multiple cores that are on offer in numerous training platforms. In other words, after a batch has been used to train on, the next batch is already queued in memory and ready to go by the time it is needed. This ensures that the data loader isn't affecting the computation time, and rather the computation time is only influenced by the forward and backward computations.

Due to limited testing time, `num_workers` was set to 2 as [22] found that more than 2 did not result in major improvements. Realistically, `num_workers` should be set to the smallest number such that the time required to do the forward and backward computations isn't higher than the time it takes to read in the next batch as this will not result in the batches being processed quicker. It was also found from our research that setting the parameter to 2 was the most optimal as increasing the `num_workers` parameter to 4 or 8 resulted in little speedup of training time: at most, the 4 workers achieved 9 seconds faster training times compared to 2 workers, whereas 8 workers achieved less than half a second faster than 4.

For the next graphs, the following legend will be used:

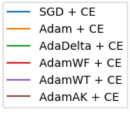

Figure 7: Legend for the Empirical Test Graphs

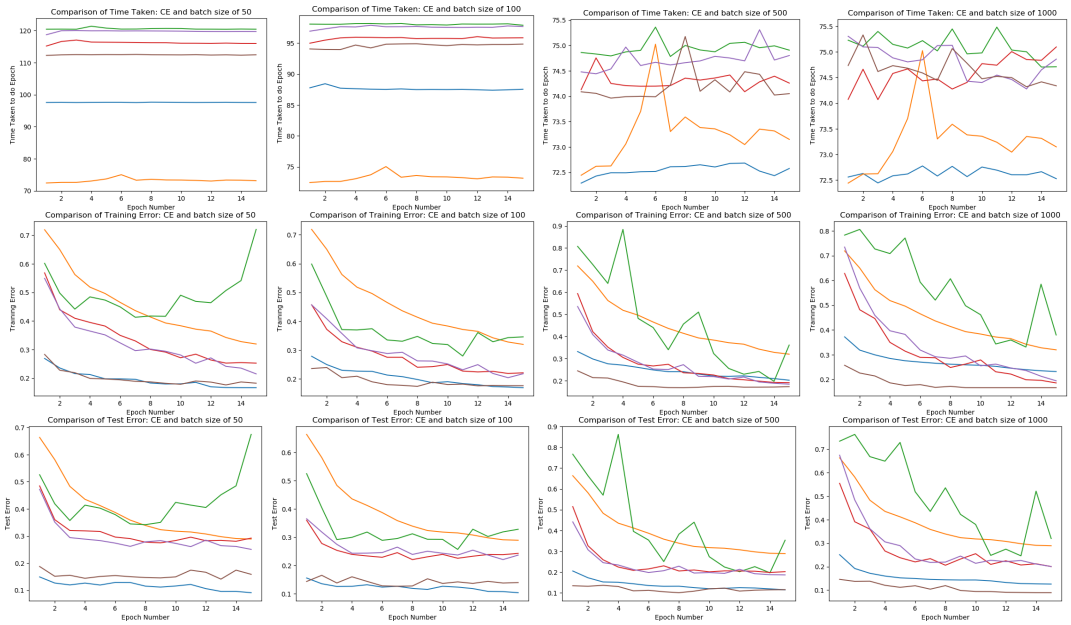

Figure 8: Cross-entropy loss, from top to bottom: time per epoch, training accuracy during epochs and test accuracy during epochs for each of the different batch sizes (50, 100, 500, 1000 from left to right).

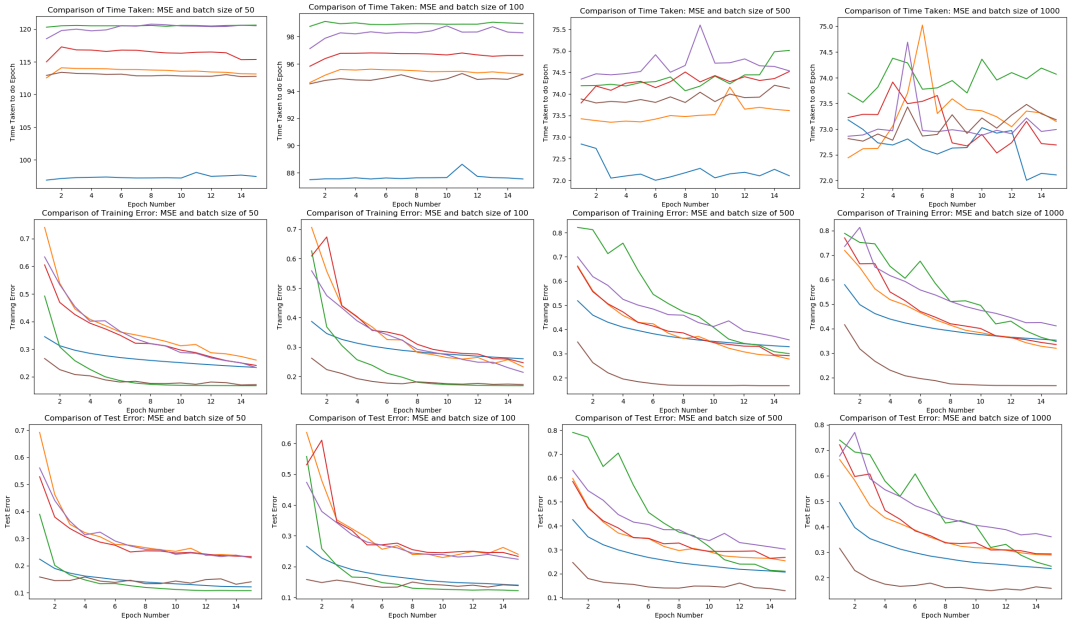

Figure 9: MSE loss, from top to bottom: time per epoch, training accuracy during epochs and test accuracy during epochs for each of the different batch sizes (50, 100, 500, 1000 from left to right).

From the results, it appears that the CE loss function was quicker than the MSE loss function for the smaller batch sizes, while the time difference for larger batch sizes was not that noticeable. The AdamAK and SGD optimisers performed the best in terms of error rates across most combinations of the optimiser, batchsize and loss function. The graphs also indicated that a larger batch size lead to a lower error. The larger batch sizes had a shorter time per epoch, while the smaller batch sizes had a larger time per epoch. From these tests, we can conclude that the SGD or AdamAK optimiser performed the best, with the best performing loss function being the Cross-Entropy loss function.

3.4 Optimiser Experiments on OliNet

It is important to note that although the results on the AlexNet/CIFAR combination might be useful in determining the best optimiser for that scenario, it does not mean that the performance necessarily transfers to our problem domain and our network. We therefore decided to also do some optimiser experimentation on our dataset and our OliNet model (defined later), albeit at a smaller scale. The CE loss function was also chosen due to its prominence in classification problems.

We experimented with the same optimisers as before (SGD, Adam, AdaGrad, Adadelta, AdamWF, AdamWT/AMSGrad and AdamAK) on the OliNet introduced in Section 4.2 to get a better idea of how the optimisers performed on our specific dataset. The OliNet model was trained for 5 epochs with each of the optimisers trained in the previous section, and the best two optimisers were chosen to run for 20 epochs.

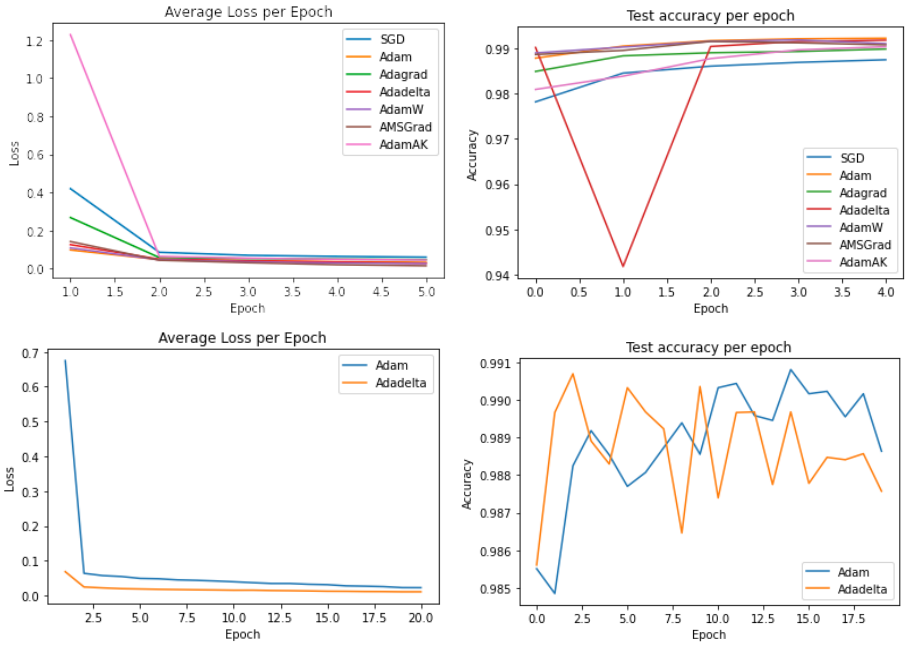

Figure 10: Head-to-head of 7 different optimisers with 5 epochs on RGB, and head-to-head of Adam, Adadelta with 20 epochs on RGB.

From these tests, it was concluded that Adam was the best-performing optimiser on our given dataset. We chose the default parameters of $\alpha = 0.001$, $\boldsymbol{\beta} = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, and used this optimiser for the rest of our experimentation.

3.5 Architecture Experimentation

To find the best neural network, Multilayer Perceptrons (MLP), convolutional neural networks (CNN), and transfer learning will be investigated. MLP architectures and CNN architectures are very common but transfer learning is an increasingly popular concept in machine learning which enables the knowledge of an existing model to be "transferred" to another domain [21]. Briefly, there is a source domain and learning task, $\mathcal{D}_S$ and $\mathcal{T}_S$, as well as a target domain and task, $\mathcal{D}_T$ and $\mathcal{T}_T$, and seek to improve learning the target function f_T. Transfer learning arises in the case that $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

The following diagram aptly describes how transfer learning works. Network A, in the top row, is trained on some classification problem and the parameters are transferred to Network B for another classification problem.

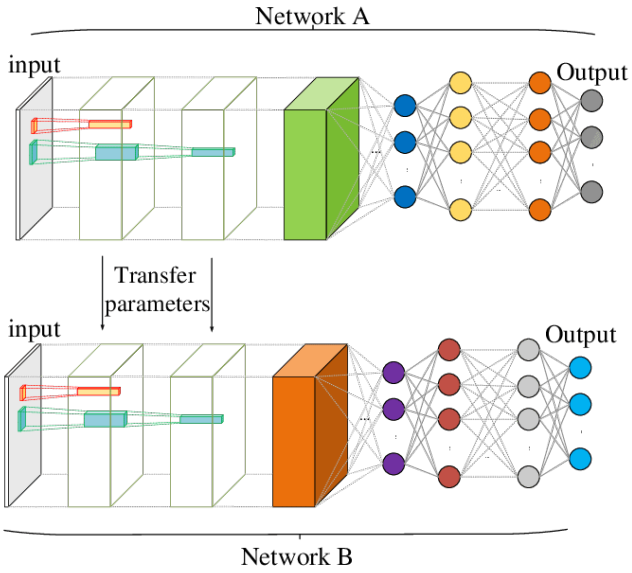

Figure 11: Transfer learning diagram [14].

In our case, the source domain was ImageNet and the target domain is face mask images. These may be similar image dimensions but differ quite significantly in distribution. Also, the target labels (mask correctness) and source labels (image category) differ.

As a result, the target task is different from the source task of AlexNet, so we are attempting *inductive transfer learning*. Specifically, we will use parameter sharing similar to [16]. This approach assumes that models for the two tasks might have similar parameters due to the related task. As a result, we take a pretrained AlexNet model and augment the final fully connected layers with different layers for our classification task.

4 Main findings

In order to assess the performance of each model, the following criteria was systematically used:

- Test Accuracy: the model's performance on the test set is important since it indicates how the model generalises the learnt features from the training set to the test set.
- Train Accuracy: the model's performance on the training set is important because it shows how the model's weight values change as the model learns the dataset. Learning curves also indicate the pace that it happened.
- Test F-Score: the F-score is the harmonic mean of the precision and recall for classification problems and is a very common measure used. These terms are defined as:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- Epoch time: if the time it takes to train an epoch is very high, then the model might be too complex and unnecessary/difficult to implement. This isn't the most important metric.
- Parameters: the number of parameters is an important metric because it indicates the complexity and amount of storage of the model. If there were two models with the same level of accuracy, but one had a lower number of parameters, then the one with the lower number of parameters would be preferred as it will take up less space (if it was used in an embedded system) and it would be less prone to over fitting.

4.1 MLP Model

Due to the prior success on using Figure 12 to classify the MNIST digits [13], testing was initially performed on this simple MLP model. After training the model on that dataset, it achieved a classification rate of above 95% using the CE loss function and optimiser SGD with a learning rate of 0.001. Using a newly initialised model, the model was trained using the face mask dataset.

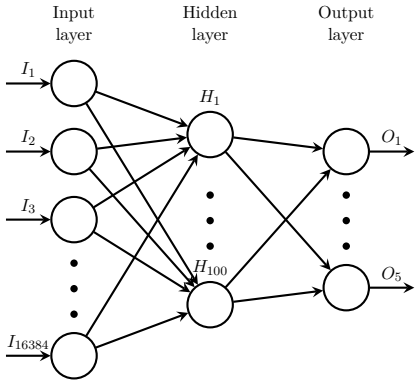

Figure 12: The initial MLP that was used for testing. For RGB, the input layer consisted of 16384 neurons.

This network was chosen as a baseline to test future neural networks due to its simplicity. However, its simplicity did not have the capacity to learn enough features: the performance was very bad with the accuracy reaching a plateau of roughly 30% accuracy.

To improve this, a new MLP was constructed, as illustrated by Figure 13. This featured 2 hidden layers, with 1000 and 100 neurons each. The input layer was $3 \times 128 \times 128$ for RGB images and 128×128 for greyscale/edge-detected. The output layer had 5 neurons. After the hidden and output layers, ReLU activation was used.

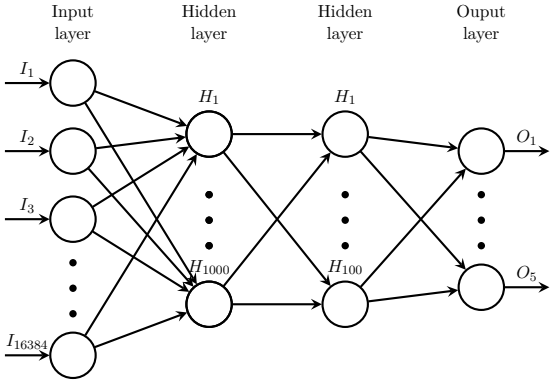

Figure 13: The next MLP that was tested. For RGB, the input layer consisted of 16384 neurons.

From the figures, the training accuracy and the test accuracy of the MLP model quickly reaches a plateau of roughly 95% accuracy on the training and test datasets after three epochs of training.

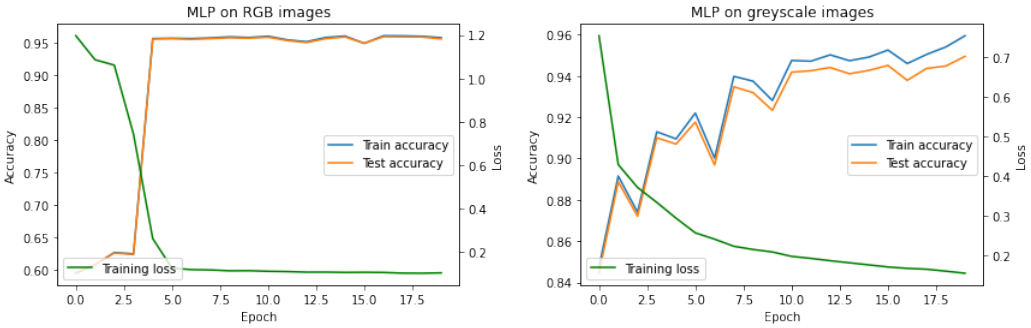

Figure 14: Performance of MLP models.

The RGB and greyscale models were then tested on images of ourselves,

Figure 15: Performance of the MLP network on Kenton's and Nick's face respectively.

On these RGB images, it was quite surprising to see the model performing reasonably well on the images of ourselves. Only two out of Kenton's four images were predicted correctly while only and four out of Nick's five images were predicted correctly.

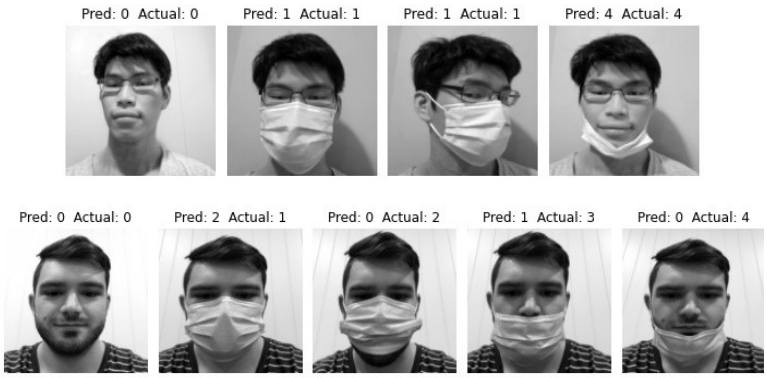

Figure 16: Performance of the greyscale MLP on Kenton's and Nick's faces.

Similarly on the greyscale images, It correctly predicted all of Kenton's images, while only predicting one out of Nick's five images.

Although it would be ideal to see closer to 100% accuracy on the test images, the MLP model performed reasonably well on the RGB images. More complex MLP models could have be made by increasing the number of layers, increasing the number of neurons, adding in dropout layers, and experimenting with different activation functions. However, due to the high performance of this baseline model, it was reasoned that no more MLP models should be tested.

4.2 OliNet Model

Following on from the success of the MLP, different CNN architectures were experimented on. It was initially hypothesised that a CNN would achieve a higher test set accuracy with fewer weights as the architecture has been been extremely successful in the image classification domain. The chosen custom architecture called OliNet was inspired by AlexNet, which rose to prominence in 2012 after winning the ImageNet LSVRC-2012 competition by a large margin. It achieved this using convolutional and max-pooling layers as well as the ReLU non-linear activation functions instead of the a tanh or sigmoid [12]. Seen in Figure 17, OliNet consists of two convolutional (3×3 filter, stride of 1) and max-pooling (3×3 filter, stride of 3) layers, followed by three fully connected layers. ReLU activation functions were placed after the last max-pooling layer, as well as between the fully connected layers.

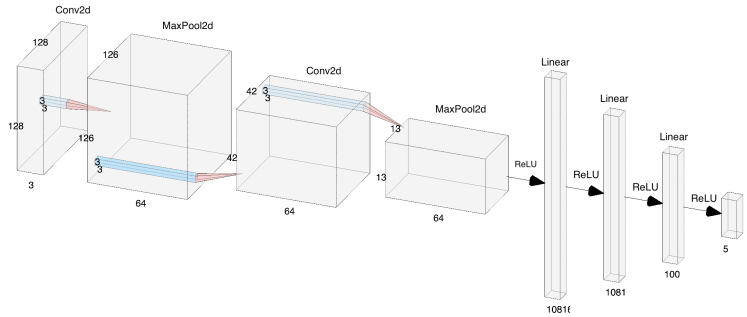

Figure 17: The OliNet architecture: image generated using [15].

After training the OliNet for twenty epochs, the model's test set accuracy plateaus at 99 and 98 percent on the RGB and grey scale datasets respectively whilst its train set accuracy continues to increase.

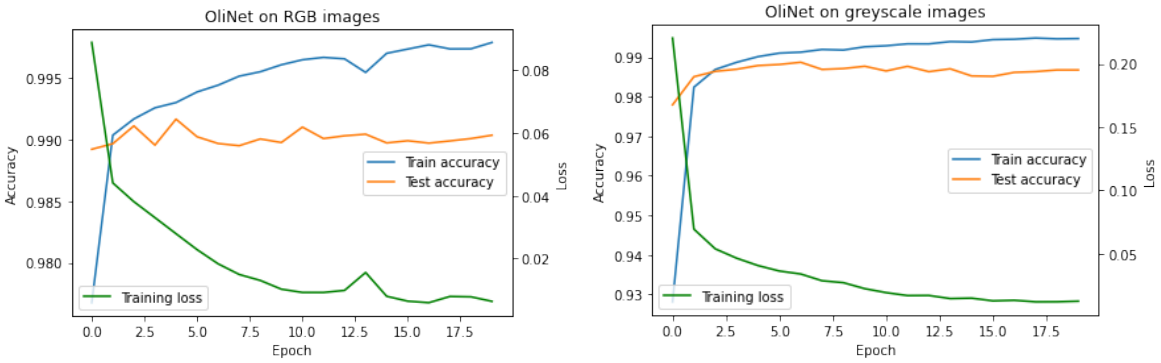

Figure 18: Performance of OliNet models.

After testing the models on pictures of ourselves we experienced mixed results, with OliNet performing better on the RGB image set. It classified three out of four of Kenton's and one out of five of Nick's images.

Figure 19: Performance of the RGB OliNet on Kenton's and Nick's faces.

But it was worse on the greyscale dataset, only predicting two out of four of Kenton's images and one out of five Nick's images.

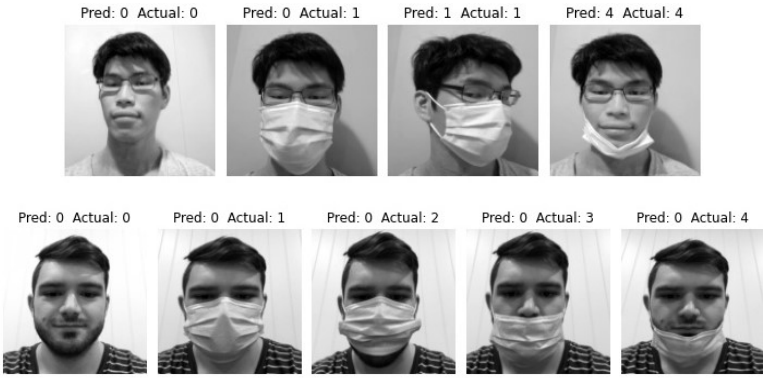

Figure 20: Performance of the greyscale OliNet on Kenton's and Nick's faces.

4.3 ResNet Model

This model was obtained from the transfer learning approach explained earlier. Specifically, we chose to apply inductive transfer learning to AlexNet and ResNet-18 via parameter sharing. These were chosen because of their excellent performance on the ImageNet dataset, so it was reasoned that their feature layers would be well suited to extracting features from images. AlexNet is a more traditional convolutional neural network architecture, whereas ResNet-18 uses residual blocks to avoid the vanishing gradient problem. The ResNet-18 was chosen to speed up training with its smaller network. It was hypothesised that a larger network would not be necessary for good performance due to the reasonable performance of the earlier MLP and CNN models. Using a smaller network with fewer parameters could also prevent overfitting to the training data.

For both, pretrained models were downloaded from the PyTorch Hub and the final fully connected layer was replaced with a fully connected linear layer of dimension 5. Other layers were unchanged.

After this, the models were trained for 20 epochs on our masked face dataset to adapt to our domain. This was done for both the greyscale and RGB images. Note that because the models expect 3 channel input, the single-channel of the greyscale images was duplicated to three identical channels. This is the same image visually, just represented in 3 channels.

For completeness, the following two figures indicate AlexNet and ResNet-18.

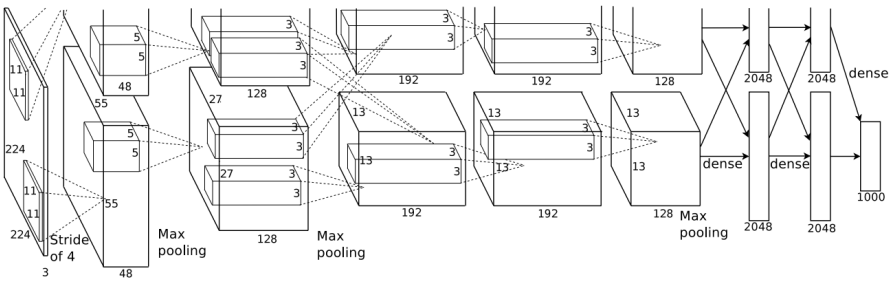

Figure 21: Architecture diagram for AlexNet [11]

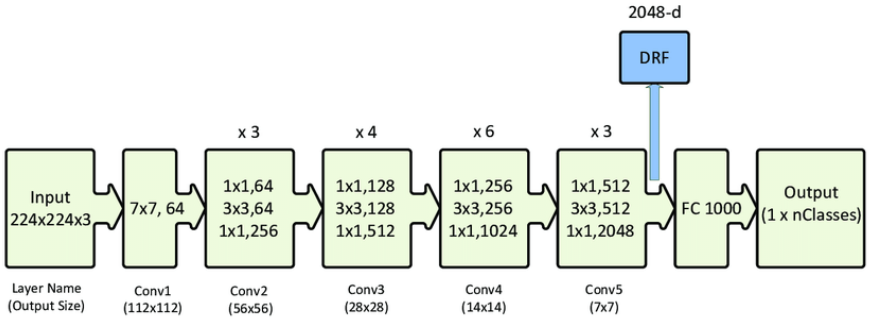

Figure 22: Architecture diagram for ResNet-50[19]. ResNet-18 is a variant with fewer hidden layers. "DRF" indicates "Deep Residual Features", the input to the fully connected layers.

The following graphs show the training and test accuracy during training, as well as the loss function. It can be seen that towards the later epochs, training accuracy was increasing but test accuracy was plateauing or decreasing slightly. This may have indicated overfitting.

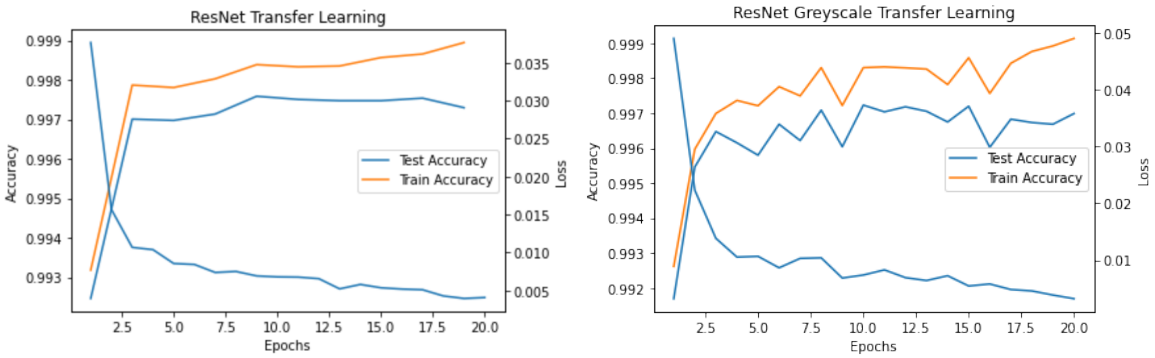

Figure 23: Performance of ResNet models.

From the graphs, it is evident that ResNet achieved the highest training and test set accuracy amongst all of the models. Most importantly, it was able to do so without significant increases in the training time, due to the more advanced architecture of the residual network. Due to this, ResNet was chosen as the best transfer learning model over AlexNet.

Below is an example of testing the model on images of ourselves. Of the images tested, it predicted class 0 (no mask) on all the images of Kenton and Nick. As a result, it achieved 22% accuracy on these images. This is obviously lower than expected, given the training and test accuracy, and will be discussed later. A possible cause is overfitting to the texture and colour of the dataset's masks.

Figure 24: Performance of ResNet on Kenton's and Nick's faces on the RGB dataset.

The ResNet model trained on the greyscale dataset performed no better, predicting class 0 for all images.

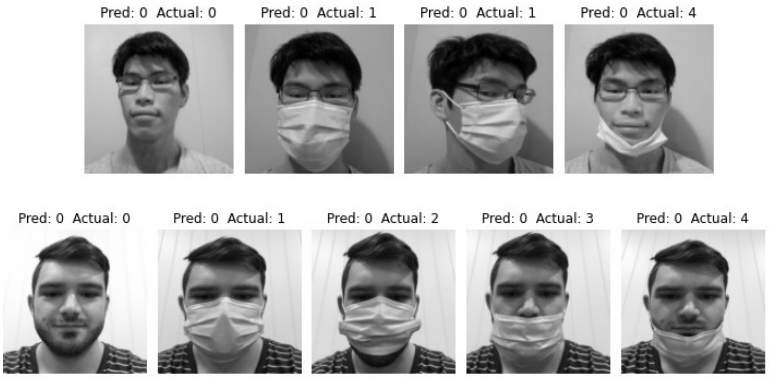

Figure 25: Performance of ResNet on Kenton's and Nick's faces on the greyscale dataset.

4.4 Best Three Models

From our findings, we summarise the results in the following table. Note that edge-detection models are omitted due to their poor performance, both on test set as well as our own test images.

Performance Metrics					
Model	Test Acc.	Train Acc.	Test F-Score	Epoch Time	# Parameters
MLP (RGB)	95.66	95.85	0.96	118	49 253 605
MLP (Grey)	94.95	95.95	0.95	95[1]	16 485 605
OliNet (RGB)	99.04	99.79	0.99	122	11 849 298
OliNet (Grey)	98.60	99.47	0.99	116	11 848 146
ResNet (RGB)	99.73	99.89	0.99	131	11 179 077
ResNet (Grey)	99.70	99.91	0.99	131	11 179 077

After training the models for 20 epochs, we found that:

- MLP performed reasonably well, achieving a test accuracy of approximately 95% and having the lowest training time.
- OliNet performed much better, achieving 98% and 99% accuracy on the greyscale and RGB images respectively.
- The transfer learning approach performed very well, with ResNet achieving 99.7% accuracy. We can see that ResNet was the most accurate model and had a comparable training time to the

[1]Trained with SGD due to optimiser issues.

other two.

Results for edge-detection were not as favourable as hoped; although we had 98.2% test accuracy with OliNet, for example, the images that were output by the Canny edge-detection algorithm were not interpretable, there were not any clear features that the model was detecting, and the model performed very poorly on our own test images. There are many reasons why this could have been the case, including improperly chosen threshold values and the Canny edge-detection highlighting unwanted features (such as eyebrows and hair). This will be discussed further in the Limitations section.

4.5 Confusion Matrices

Once the training was performed, confusion matrices were produced for each model. Each model performed significantly well across the mask classes, with no obvious weakness for both the RGB and greyscale datasets. The models performed well considering the class imbalance. Due to the poor performance of the edge-detection images, they have been omitted. These matrices are below:

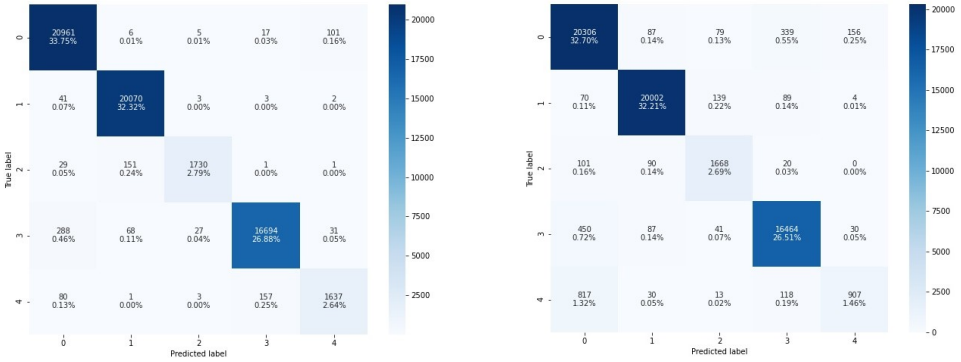

Figure 26: Left to right: Confusion matrices for the MLP model using the RGB and greyscale datasets.

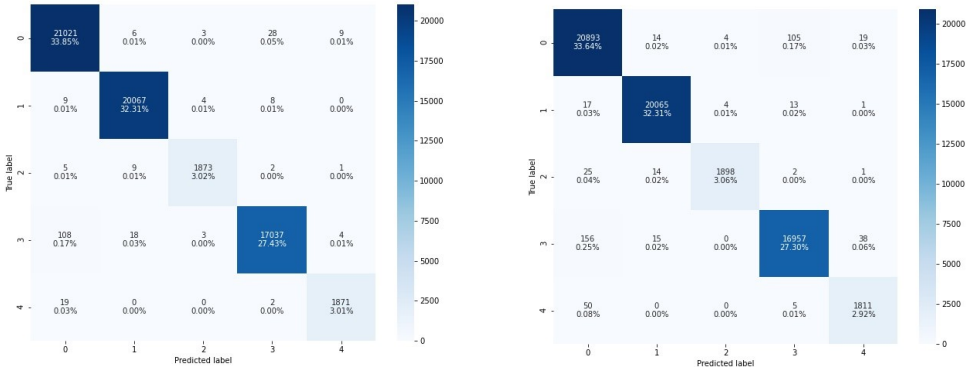

Figure 27: Left to right: Confusion matrices for the OliNet model using the RGB and greyscale datasets.

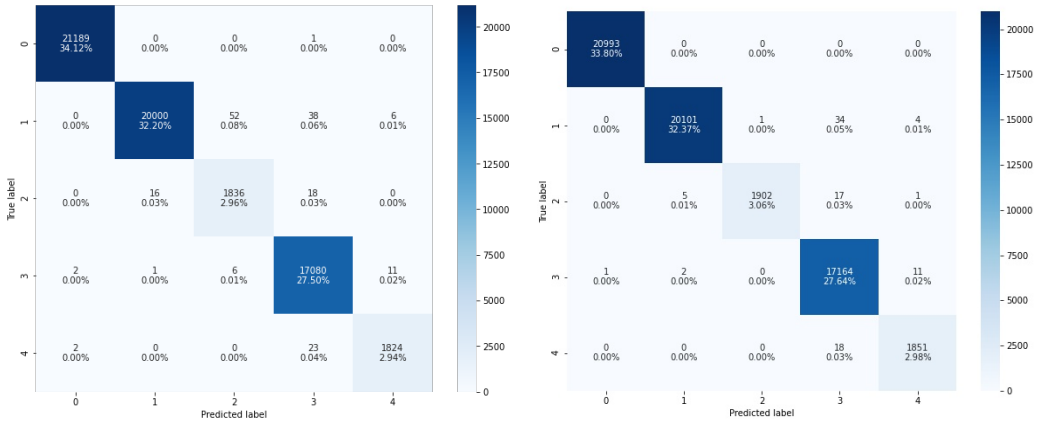

Figure 28: Left to right: Confusion matrices for the ResNet model using the RGB and greyscale datasets.

The model performed extremely well on misrepresented classes, the accuracy in class 2 and class 4 (mask covering nose and mouth and mask covering chin only respectively) was significantly high with very few misclassified images, as indicated by the off diagonals.

4.6 Final Model

Based off our previous findings, we decided to choose the RGB **OliNet** as our final model. This was because it struck a good balance between test set performance (achieving 99.04% test accuracy on RGB and 98.6% test acc on greyscale) and real-world performance on our own images. It also has reasonable computational efficiency, as indicated by the epoch time, and moderate memory usage, as indicated by number of parameters. The F-Scores did not give a good indication of how good one model was over another. The runner-up model is the RGB MLP, which proved to be remarkably accurate on the real-life images despite its architectural simplicity.

5 Limitations

Although machine learning and deep learning problems have "blackbox" solutions, and some issues are hard to diagnose, the five limitations listed below would have most likely affected the most optimal model being found and the results.

5.1 Dataset Limitations

There were several major limitations with the dataset:

- The lack of diversity in mask colour and texture resulted in a biased model that was less accurate on people with facial hair or skin tone similar to the mask's blue colour. Additionally, the accuracy of young children wearing masks was also limited, perhaps due to the small representation in the dataset or due to their smaller face shapes.
- As a consequence, the image transformation techniques weren't effective in improving model generalisability because the greyscaled images still had a very similar shade and the edge-deteced images a distinctive texture.
- Because the dataset was computer-generated, there were a number of misclassified samples that negatively impacted our models training performance.
- A more even number of images per sub class of incorrectly worn masks would ensure the model was accurate for all classes. This could be fixed by creating more augmented images for the respective class. However, the dataset authors realistically should have created a uniform class representation.

5.2 Model Training Phase

During the training phase, there were some difficulties experienced with the large size of the dataset. The dataset had to be downloaded initially, taking roughly an hour, and then preprocessing and image transformations, taking roughly 20 minutes, had to be completed prior to training. The large size of the dataset presented difficulties in the image transformation and preprocessing. This was mitigated by downscaling, but still complicated the file transfer process. It was not practical to complete the transformations and then upload the transformed data to a training environment as uploading all the transformed files would have taken a significant amount of time. Instead, the original reduced images files were uploaded as `.zip` files and then transformed directly on the notebook instance, reducing upload times three-fold.

When using Google Colab, once the files were uploaded to Colab, the typical approach of mounting the Google Drive into the Colab instance and loading directly resulted in very poor performance. As a result, a workaround was used which copied the `.zip` file from Google Drive onto the local Colab instance, and then unzipped it in the local environment. This means that Colab was no longer requesting the files from the Google Drive at every iteration and instead could access it directly from the local environment.

Despite the careful selection of platforms, they were still not perfect. Namely, Paperspace and Colab both had time limits on their free tiers, which limited the complexity of the models we could train.

5.3 Errors in the Dataset

One thing that was unexpected was the dataset containing multiple mislabelled images and incorrectly drawn masks. Below are some examples in which the mask drawing algorithm failed to do so appropriately. All masks here were erroneously generated, with the first and last labelled as "correctly worn", but are clearly missing masks. The middle three images, while also being mislabelled, have their masks placed in incorrect positions.

Figure 29: Errors in the MaskedFace-Net dataset and ResNet predictions.

It is hard to know exactly how this would have affected the training. It would have definitely contributed to decreasing the model's predictive capacity due to the cross-over between classes. If time permitted, these images would have been removed from the dataset and training would have been performed on the new dataset.

5.4 Generalisation to Other Mask Types

As mentioned above, once the models were trained using the MaskedFace-Net dataset, the models were tested on images of ourselves. This was to see how the models generalised to images that were not apart of the dataset. Although the final models were considered successful due to their high training and test accuracy, it was seen that they generalised poorly to images of us. Additionally, it was found that the models generalised poorly to images, as indicated in the figure below, the transfer learning network, ResNet performed poorly on classifying how a black mask is worn. It achieved an accuracy of 1 out of 5. This indicates very poor generalisation to other mask types.

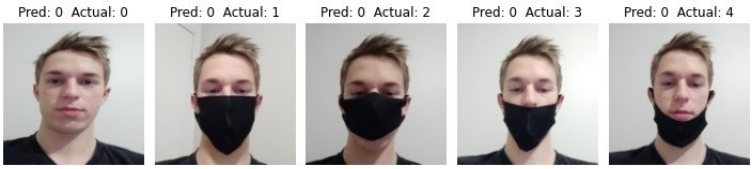

Figure 30: ResNet predictions of Black Masks

This poor generalisation can be attributed to two reasons. The first is that the dataset did not contain images of people wearing different masks and hence the model could not learn the features of other masks. The second reason is that the models were significantly overfitting and started detecting unwanted features. It is often hard to say which reason contributed to the poor generalisation due to the "black box" nature of many deep learning solutions.

This issue can be mitigated by training on a dataset that contains masks with different colours and textures. The images could perhaps be labelled by the type of mask that is present and how the mask is worn; in other words, instead of the labels consisting of "incorrect", it could be "incorrect - black mask". In doing so, the model could learn to differentiate between different masks. This would result in a dataset that would be significantly bigger in size but arguably is worth it considering the poor generalisation to other masks, and the requirement for high accuracy considering the problem. The larger dataset would be easy to create: instead of Photoshopping images of blue masks, other masks could be used instead.

Following from this poor generalisation performance, we wanted to investigate which images in the training dataset where similar to the images of ourselves. This was done by taking the input to the output layer in each neural network and finding the nearest ℓ_2-norm neighbours. This would give a good indication as to which images in the training set are the closest in terms of the learnt feature space. In terms of the set-up, suppose that the model consists of multiple layers and the input to the output, in each case, is a linear vector $\boldsymbol{x}_i \in \mathbb{R}^N$ where N is the number of neurons in that layer. Then, for the k-th image in the training set, we take the input to the output layer, $\boldsymbol{x}_i^{(k)}$, and find the top 5 which are closest to a particular image of ourselves.

From implementing this, for the MLP we see that:

Figure 31: MLP Nearest Neighbours for two Images.

From implementing this, for the OliNet we see that:

Figure 32: OliNet Nearest Neighbours for two Images.

From implementing this, for the ResNet we see that:

Figure 33: ResNet Nearest Neighbours for two Images.

Due to the poor performance, we have only decided to include one image as the trends were consistently seen across multiple images. Using the same image for the three models allows us to compare how each model "interprets" the image. The ℓ_2-norm nearest neighbours highlights the poor performance of the models and their inability to generalise the images of ourselves.

- The MLP predicted found multiple incorrectly labelled images for Kenton's class 1 image, indicating that the incorrectly labelled images in the dataset affected the performance of the model.
- OliNet had mixed results across both of Kenton's images indicating that images of class 0 were close to images of class 1.
- ResNet is predicting that an image of class 0 is similar to an image of class 1 as it has similar nearest neighbours. Additionally, it appears to be picking up on the glasses in the image. This could indicate the models have been overfitted.

This indicates that each model has limitations in different aspects and further work would need to be done to address these.

5.5 Poor Performance of Edge Detection Image Transformations

Various techniques were attempted to offset the dataset's uniformity (described earlier). One technique was Canny edge-detection would improve generalisation performance by ignoring the colours and focusing only on the mask's texture. Initially, this was done with the `cv2.Canny` function of OpenCV with an upper parameter of 255 and a lower parameter of $\lfloor 255/3 \rfloor$.

Below are some examples of edge-detected images from the dataset. It can be seen that the algorithm was successful in extracting edges from the images and the masked face edges appear differently from unmasked faces.

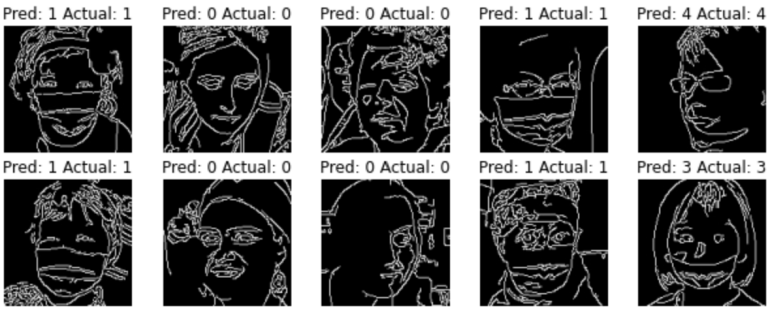

Figure 34: ResNet predictions of edge-detected images.

The models trained on edge-detected images achieved good training and test accuracy on the dataset (similar to RGB and greyscale) so we were hopeful of their performance on images of ourselves. This would be the most important test of the models' generalisation.

However, the edge-detected models consistently underperformed on our images.

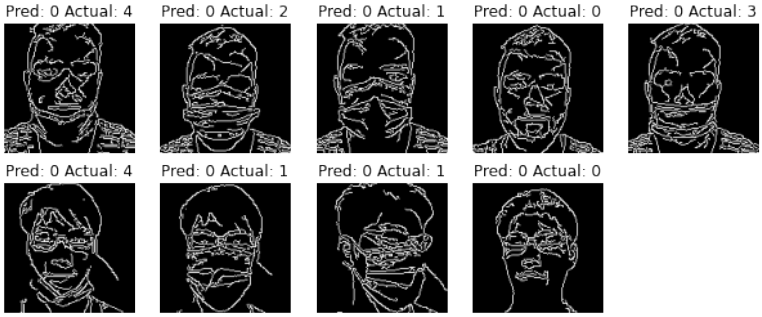

Figure 35: ResNet predictions of Nick's and Kenton's edge-detected images.

It was expected that edge-detection would have good generalisation performance, so this was quite surprising. We attempted to tweak the Canny edge-detection parameters in an attempt to improve performance. The model was trained again on edge-detected images with reduced upper and lower thresholds of 150 and 50. This was to improve the sensitivity of the algorithm and better capture the mask boundary.

This achieved similar performance on the dataset but unfortunately, this only worsened the situation with our images. The adjusted thresholds led to too much noise on our images so the model frequently predicted 3 and 4, often incorrectly.

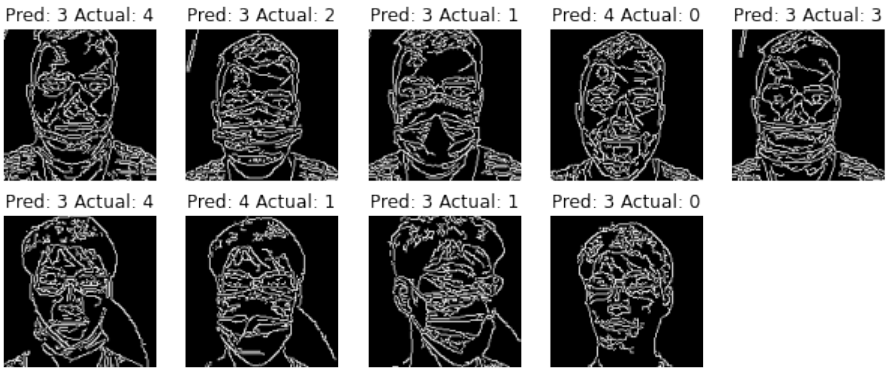

Figure 36: ResNet predictions of Nick's and Kenton's new edge-detected images.

There are several reasons this could be happening. Due to the automated generation of face masks,

all the masks in the dataset had clearly defined and consistent folds regardless of lighting and photo angle. This was not the case for our images, due to different lighting and textures in the real-life masks. Below, it can be seen that the dataset's masks have consistent edges across the mask with obvious boundaries at the top and side of the mask. This is not the case for our real-life images, which most likely leads to the model performing poorly.

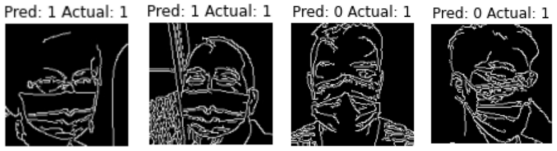

Figure 37: ResNet predictions of edge-detected images in the dataset (first two) and our own images (last two).

This would have led to poor performance on any mask not similar to the dataset's mask texture, significantly impacting generalisation.

6 Conclusions

By designing, experimenting, and finally testing the three different neural network approaches on different image reduction techniques, mixed results were achieved. ResNet achieved the best train and test accuracy while also being the smallest network in terms of parameters. This was followed by OliNet and finally the MLP network. The MLP network was marginally quicker than the OliNet while ResNet took roughly 10 seconds more. Yet, when looking at each model's performance on the images of ourselves, we see that OliNet performed the best overall. This resulted in us choosing OliNet as the final solution as it aligned the closest to the problem goals.

In this report, we have demonstrated the significance of undertaking this project. With slight changes and additional tests, the model is on track to be implemented with camera technology to detect how people are wearing face masks. The project also demonstrates how to handle large amounts of image data and which training environment is suitable. Finally, this solution approach can be applied to similar problems of detecting if people are wearing PPE correctly.

A number of limitations of the dataset were observed during testing, particularly the uniform colour of the photoshopped face masks and often inaccurate face mask generation. In an attempt to mitigate these issues, image transformation techniques such as greyscaling and canny-edge detection were applied but ultimately proved ineffective as the masks still had a distinctive shade and texture in the transformed images.

Despite this, the work done is still very worthwhile. Much of the background and preliminary research will remain relevant in future. In particular, the analysis of limitations will assist similar projects. This project provides a framework for detecting face mask correctness. It is expected that subsequent projects, with access to a more varied dataset of real face masks, could be quite successful with these approaches.

7 Further Work

There are a few areas for further work. The first is using a dataset containing real images of face masks as opposed to computationally generated ones such as the dataset by [6]. This dataset is significantly smaller in size, roughly 10000 images but excellent results have been obtained by [1]. The images in this dataset contain more of an observable background, have varying poses and objects in the foreground such as sunglasses. Using this dataset could allow us to test some of the following:

- How do our trained models generalise to those images and vice versa?

- How do models trained on the other dataset perform on the images of ourselves?
- Are there better models with different datasets that achieve similar results?
- What features are learnt between the different datasets?

If time permitted, testing either one of these would be very interesting and could potentially diagnose the overfitting issue we experienced as the other dataset contained more mask types.

Finally, more computational power would allow for experimentation with more sophisticated models and higher-resolution images, possibly increasing the accuracy of the neural network model. Similarly, more in-depth empirical tests could be performed by testing more hyperparameters, loss functions, and optimisers, or simply testing for more epochs to give a better indication of how the optimisers converge to the optimal weights. This would have been time consuming, and possibly out of the project's scope.

References

[1] ayushimishra2809. *Face mask detection*. June 2020. URL: https://www.kaggle.com/ayushimishra2809/face-mask-detection.

[2] Jason Brownlee. *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Aug. 2020. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[3] Adnane Cabani et al. *MaskedFace-Net – A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19*. 2020. arXiv: 2008.08016 [cs.CV].

[4] *Coronavirus Cases:* URL: https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1?.

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

[6] Wobot Intelligence. *Face Mask Detection Dataset*. June 2020. URL: https://www.kaggle.com/wobotintelligence/face-mask-detection-dataset.

[7] Mingjie Jiang, Xinqi Fan, and Hong Yan. *RetinaMask: A Face Mask detector*. 2020. arXiv: 2005.03950 [cs.CV].

[8] Andrej Karpathy. *3e-4 is the best learning rate for Adam, hands down*. Nov. 2016. URL: https://twitter.com/karpathy/status/801621764144971776?lang=en.

[9] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. arXiv: 1812.04948 [cs.NE].

[10] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Apr. 2009.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[12] Vihar Kurama. *A Review of Popular Deep Learning Architectures: AlexNet, VGG16, and GoogleNet*. June 2020. URL: https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/.

[13] Yann Lecun. *THE MNIST DATABASE*. URL: http://yann.lecun.com/exdb/mnist.

[14] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. "Transfer Learning of Temporal Information for Driver Action Classification". In: Apr. 2017.

[15] Alex Lenail. *NN SVG*. 2020. URL: https://alexlenail.me/NN-SVG/.

[16] Xinhao Li and Denis Fourches. "Inductive transfer learning for molecular activity prediction: Next-Gen QSAR Models with MolPMoFiT". In: *Journal of Cheminformatics* 12.1 (Apr. 2020). DOI: 10.1186/s13321-020-00430-x. URL: https://doi.org/10.1186/s13321-020-00430-x.

[17] Mohamed Loey et al. "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic". In: *Measurement* 167 (Jan. 2021), p. 108288. DOI: 10.1016/j.measurement.2020.108288. URL: https://doi.org/10.1016/j.measurement.2020.108288.

[18] Ilya Loshchilov and Frank Hutter. "Fixing Weight Decay Regularization in Adam". In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[19] Ammar Mahmood et al. "Automatic Hierarchical Classification of Kelps Using Deep Residual Features". In: *Sensors* 20.2 (Jan. 2020), p. 447. DOI: 10.3390/s20020447. URL: https://doi.org/10.3390/s20020447.

[20] Julie Mazziotta. *Here's How to Correctly and Safely Wear a Face Mask, Plus How to Not Fog up Your Glasses*. 2020. URL: https://people.com/health/heres-how-to-correctly-and-safely-wear-a-face-mask-plus-how-not-to-fog-up-your-glasses/.

[21] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. DOI: 10.1109/tkde.2009.191. URL: https://doi.org/10.1109/tkde.2009.191.

[22] *PyTorch DataLoader num_workers - Deep Learning Speed Limit Increase*. Sept. 2019. URL: https://deeplizard.com/learn/video/kWVgvsejXsE.

[23] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ.

[24] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. Mar. 2020. URL: https://ruder.io/optimizing-gradient-descent/.

[25] Leonardo Setti et al. *Airborne Transmission Route of COVID-19: Why 2 Meters/6 Feet of Inter-Personal Distance Could Not Be Enough*. Apr. 2020. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7215485/.

[26] Bevan Shields. *'Overwhelmed': Germany, France return to lockdown as virus cases soar in Europe*. Oct. 2020. URL: https://www.smh.com.au/world/europe/overwhelmed-germany-france-return-to-lockdown-as-virus-cases-soar-in-europe-20201029-p569jz.html.

[27] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012). arXiv: 1212.5701. URL: http://arxiv.org/abs/1212.5701.

[28] Renyi Zhang et al. "Identifying airborne transmission as the dominant route for the spread of COVID-19". In: *Proceedings of the National Academy of Sciences* 117.26 (2020), pp. 14857–14863. ISSN: 0027-8424. DOI: 10.1073/pnas.2009637117. eprint: https://www.pnas.org/content/117/26/14857.pdf. URL: https://www.pnas.org/content/117/26/14857.