1. Problem 1. Person Re-Identification

1.1 Data Pre-Processing

The dataset involves person re-identification, or the task of matching an image of a person (probe) to a gallery of previously seen people done using a portion of the Market-1501 dataset. Each image shows an entire person roughly centred in the image. The images are initially loaded at a size of 128 x 64. Pre-processing of the images involved resizing each image into a size of 64 x 32, as well as converting into grayscale, as the task is not concerned with any form of colour recognition. These two processes were primarily done due to limited computational resources especially pertaining to the deep learning method, as decreasing the size of the image inputs as well as decreasing the dimensions of the images, from 3 (RGB image) to 1 (grayscale image), results in a vast decrease in total model parameters and therefore model training time, which will be discussed further later on. No crops were done as the task is concerned with person re-identification and requires a full image of a person. The dataset was also vectorized for the purposes of PCA for the non-deep learning method.

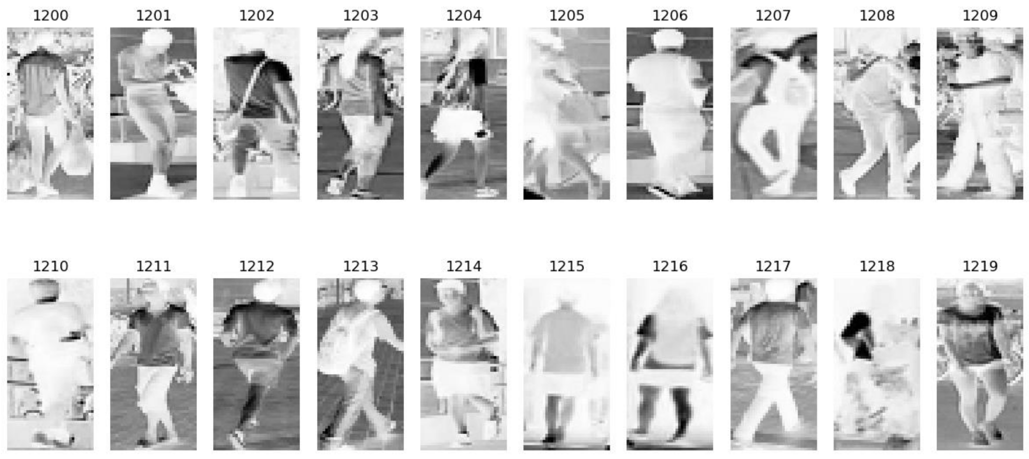


Figure 1: Samples after resizing and grayscale conversion. Captions for every sample denote the identity label.

1.2 Non-Deep Learning and Deep Learning Approaches and Details

PCA was utilised for the non-deep learning approach fitted on the training set resized and grayscale-converted images. After, both the training and testing data were transformed using the fitted PCA object. For the purposes of calculating Top-N accuracy, the training and testing dataset images were transformed to 90%, 95%, and 99% reconstruction using the cumulative sum of the explained variance in the dataset at each percentage point mentioned, for example finding the point at which the cumulative sum gets past 95% and storing it in a variable. The 90%, 95%, and 99% reconstructed or transformed dataset were used to calculate Top-10, Top-5, and Top-1 performance respectively. An example of the transformed data is as such:

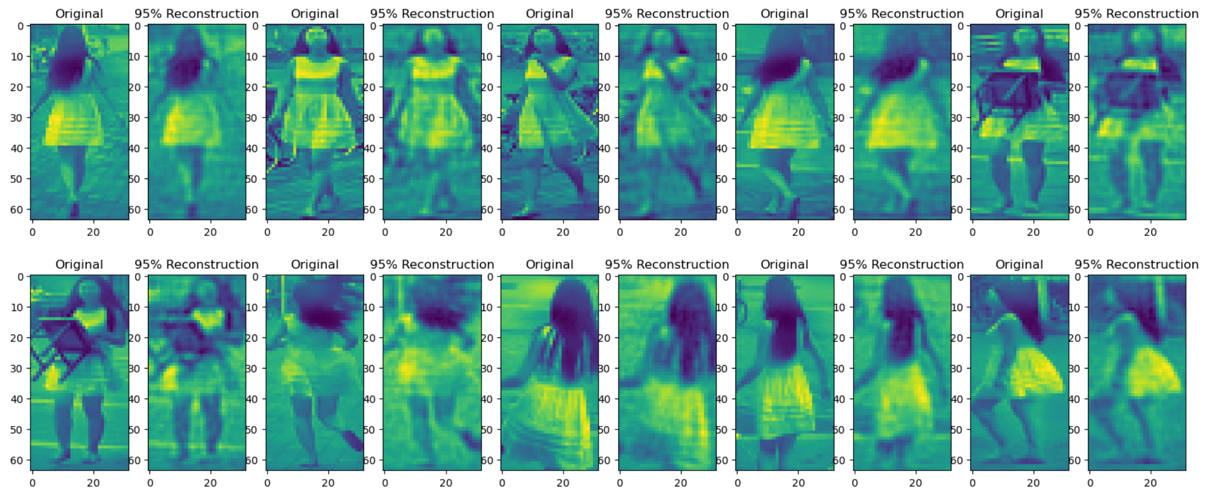


Figure 2: A 95% reconstruction of images within the training set using PCA and the cumulative sum of the 95% explained variance ratio.

For the deep learning method, we are using a metric learning method, in particular triplet loss. For this method, the training set was reshaped into (5933, 64, 32, 1) for the purposes of the deep learning model which will the discussed later as opposed to the vectorized (5933, 2048) shape previously used for PCA. The testing was reshaped into (301, 64, 32, 1) each for the gallery and probe sets. From these, we can apply triplet tests using the helper methods in the provided template.

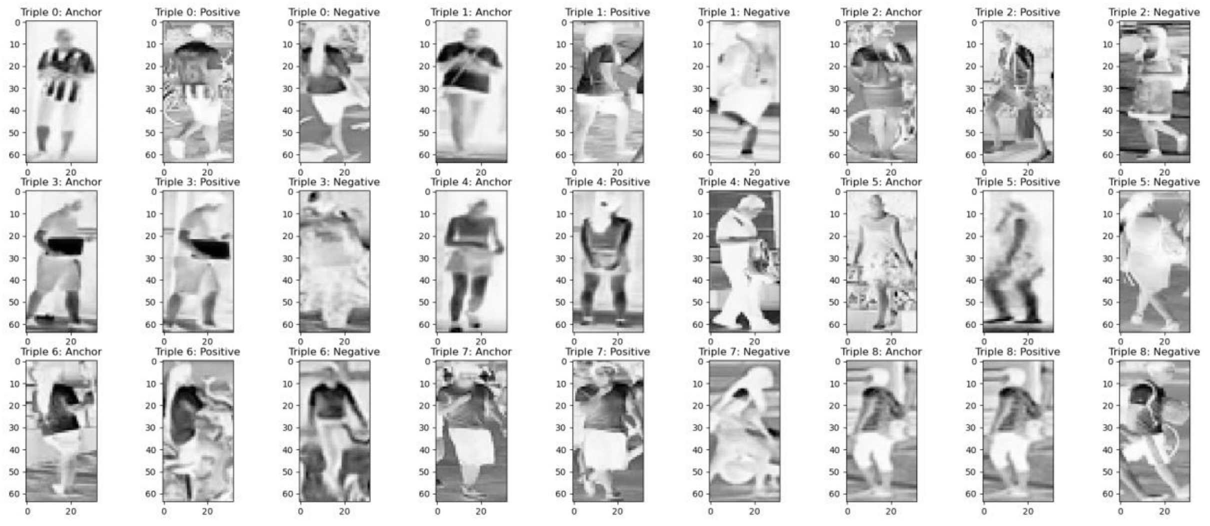
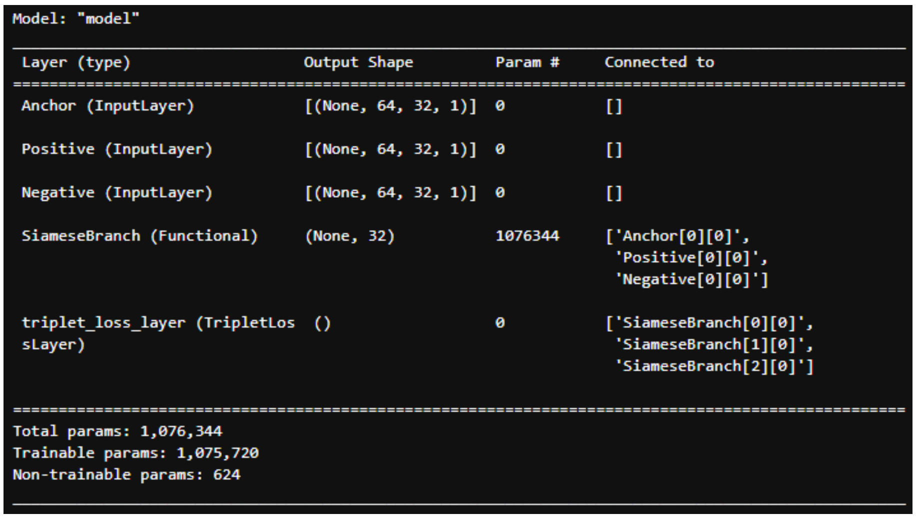


Figure 3: Sample triplet test from the training set. Anchor denotes the ground truth, positive denotes an image containing a person with the same identity label, and negative

denotes an image containing a person with a different identity label. Images were plotted to visually check.

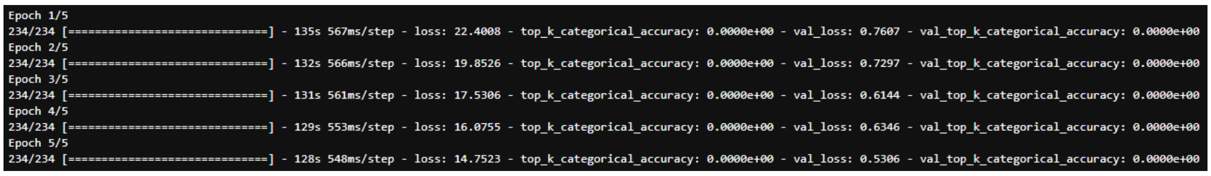
A VGG network was utilized from the triplet data to train the model for person re-identification. The network takes a dummy input of (64, 32, 1) or the size of each image in the dataset using an embedding size of 32 taking into account a network for this task complexity. The network utilizes three groups of convolution blocks at sizes of 8, 16, 32 filters, a dense layer at a size of 256, a small amount of spatial dropout (0.2), and no other dropouts. An embedding layer comes out of the network, and a triplet loss was used overall. The inputs take in the anchor, positive, and negative images as can be seen in the sample in Figure 3, and are passed through the entire network. Finally, it is passed through the triplet loss layer, which is also the output, with a margin of 1 as we are using the normalisation of the embeddings.



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 Anchor (InputLayer)            [(None, 64, 32, 1)]  0           []

 Positive (InputLayer)          [(None, 64, 32, 1)]  0           []

 Negative (InputLayer)          [(None, 64, 32, 1)]  0           []

 SiameseBranch (Functional)     (None, 32)           1076344     ['Anchor[0][0]',
                                                                  'Positive[0][0]',
                                                                  'Negative[0][0]']

 triplet_loss_layer (TripletLos  ()                  0           ['SiameseBranch[0][0]',
 sLayer)                                                          'SiameseBranch[1][0]',
                                                                  'SiameseBranch[2][0]']

==================================================================================================
Total params: 1,076,344
Trainable params: 1,075,720
Non-trainable params: 624
__________________________________________________________________________________________________
```

Figure 4: Summary of the VGG triplet network. The model has 1,076,344 parameters.

The network was compiled and trained on the testing set with the Root Mean Squared Propagation (RMSProp) optimiser, a batch size of 256 and 5 epochs, taking into account computational considerations and limited training time. It was also compiled using the Top K Categorical accuracy metric, with a k of 1, 5, and 10 for Top-1, Top-5, and Top-10 accuracy respectively.



```
Epoch 1/5
234/234 [==============================] - 135s 567ms/step - loss: 22.4008 - top_k_categorical_accuracy: 0.0000e+00 - val_loss: 0.7607 - val_top_k_categorical_accuracy: 0.0000e+00
Epoch 2/5
234/234 [==============================] - 132s 566ms/step - loss: 19.8526 - top_k_categorical_accuracy: 0.0000e+00 - val_loss: 0.7297 - val_top_k_categorical_accuracy: 0.0000e+00
Epoch 3/5
234/234 [==============================] - 131s 561ms/step - loss: 17.5306 - top_k_categorical_accuracy: 0.0000e+00 - val_loss: 0.6144 - val_top_k_categorical_accuracy: 0.0000e+00
Epoch 4/5
234/234 [==============================] - 129s 553ms/step - loss: 16.0755 - top_k_categorical_accuracy: 0.0000e+00 - val_loss: 0.6346 - val_top_k_categorical_accuracy: 0.0000e+00
Epoch 5/5
234/234 [==============================] - 128s 548ms/step - loss: 14.7523 - top_k_categorical_accuracy: 0.0000e+00 - val_loss: 0.5306 - val_top_k_categorical_accuracy: 0.0000e+00
```

Figure 5: Sample model training statistics. ‘k’ = 5 in this instance.

1.3 Approach Evaluations

To extract the Top-N accuracy for the non-deep learning methods, a simple K Neighbors Classifier was used, with 3 n_neighbors, 'distance' weight function, and -1 n_jobs to use all processors to run the neighbours search primarily for the sake of a quicker computational runtime. Each of the following KNNs were fitted on the n% reconstructed transformed gallery datasets, then evaluated using an 'eval_model' helper function which evaluates the fitted KNN on the training set (gallery sets against probe sets) and returns training set performance as well as confusion matrices for better visualisation.

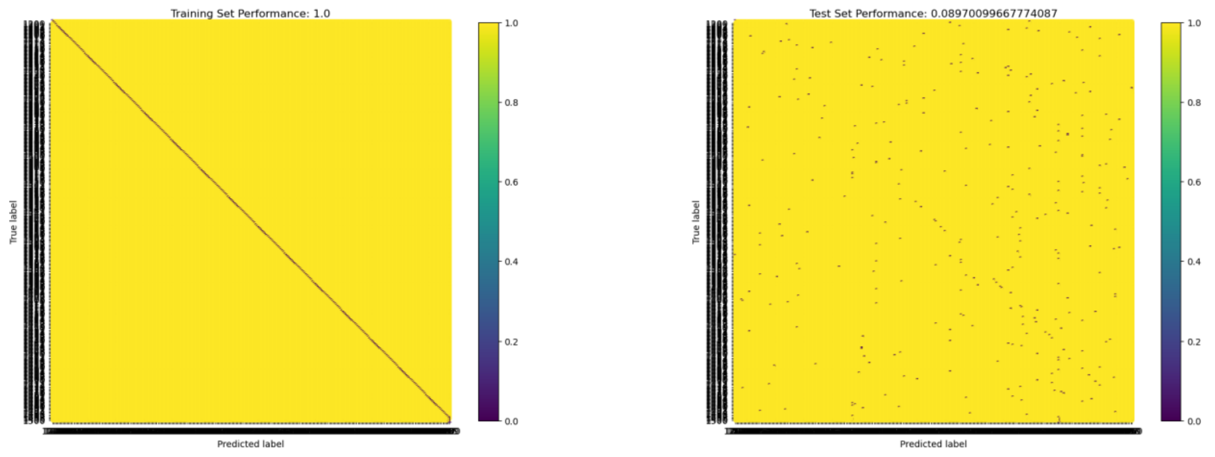


Figure 6: Testing set performance and confusion matrix for the 90% reconstructed training data set. The performance is 8.97% (Top-10 accuracy).

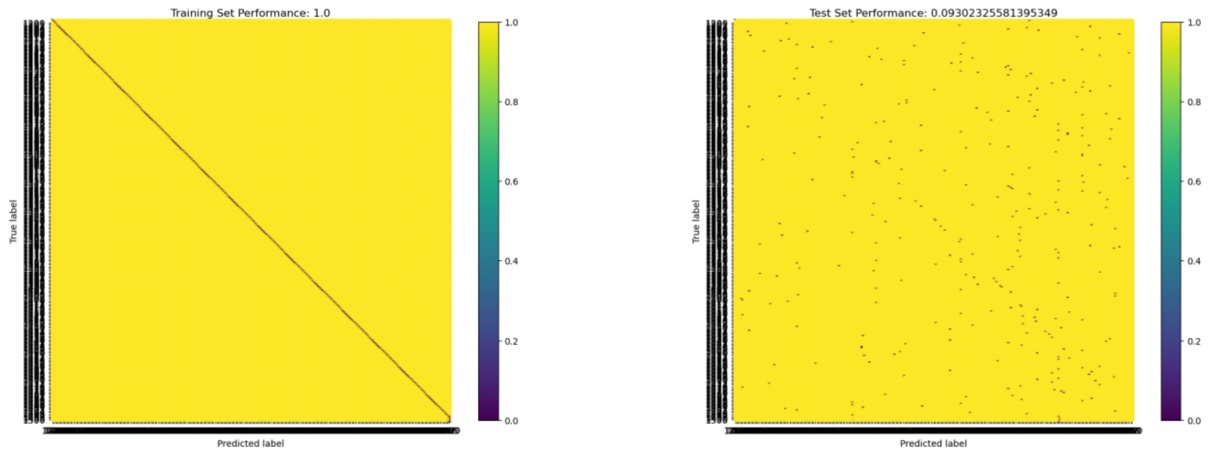


Figure 7: Testing set performance and confusion matrix for the 95% reconstructed training data set. The performance is 9.30% (Top-5 accuracy).

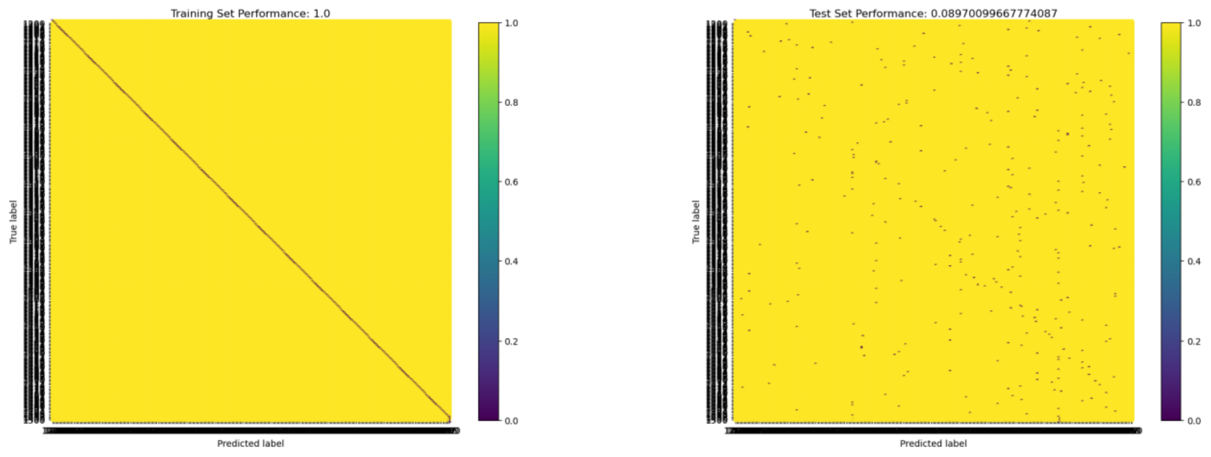


Figure 8: Testing set performance and confusion matrix for the 99% reconstructed training data set. The performance is 8.97% (Top-1 accuracy).

Overall, the performance is around the 9% mark using the non-deep learning, PCA method. This means that in 9% of cases, the closest person identified is the correct person.

For the deep learning method, the training dataset was passed into the trained network from which we can extract T-SNE embeddings for each 'k' in Top K Categorical Accuracy network for better visualisation.

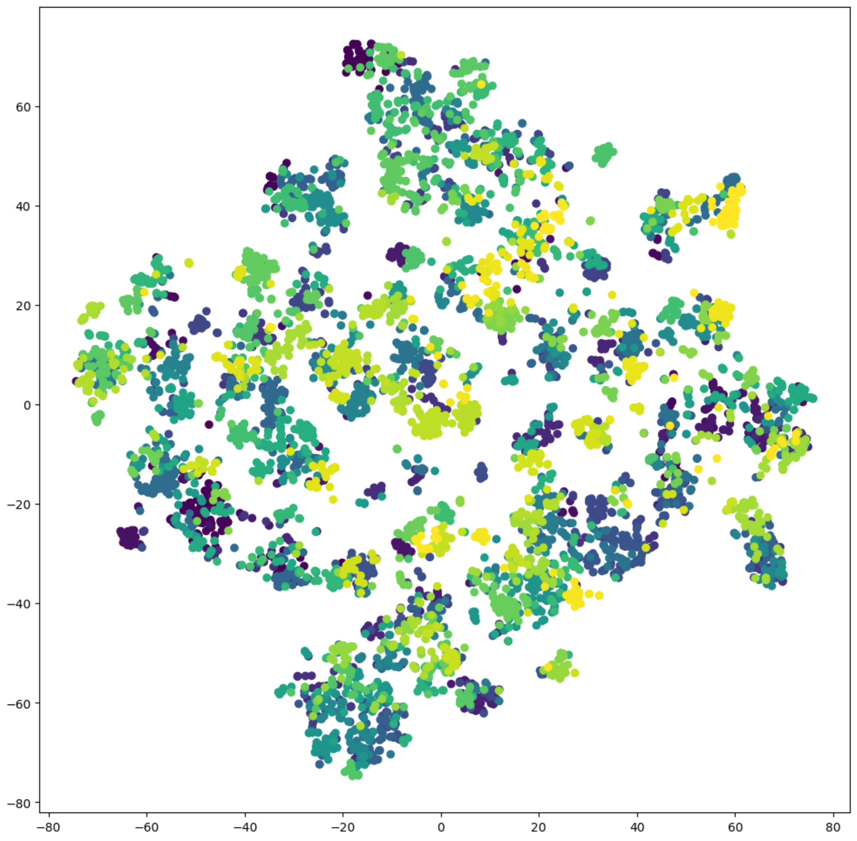


Figure 9: T-SNE embeddings from the training set with a ‘k’ of 1.

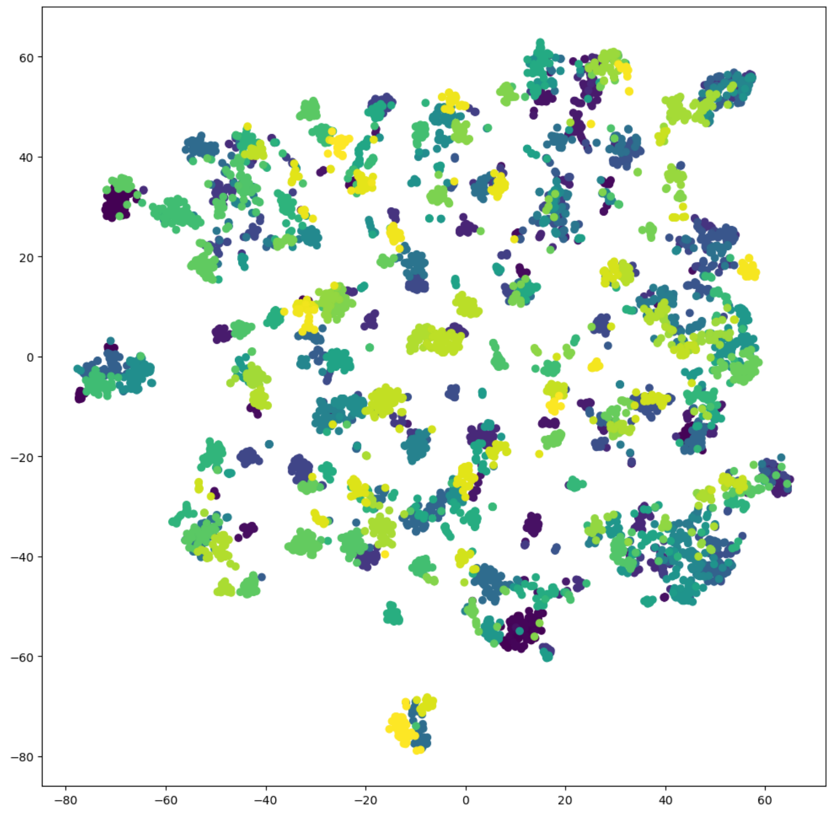


Figure 10: T-SNE embeddings from the training set with a ‘k’ of 5

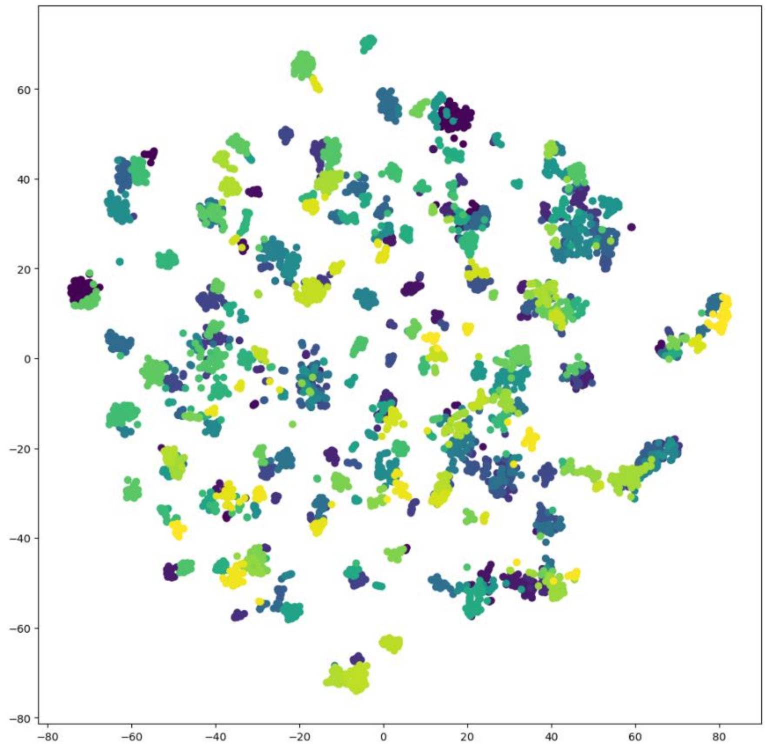


Figure 11: T-SNE embeddings from the training set with a ‘k’ of 10

We can see from the T-SNE embeddings that the training set has really good identity separation. Each identity (colour) is visually very near to one another, meaning the model can accurately predict individuals or images with the same identity from the training dataset. Further, we can pass this into our previous KNN to obtain a numerical Top-N accuracies of the training dataset using network predictions of the gallery and probe datasets.

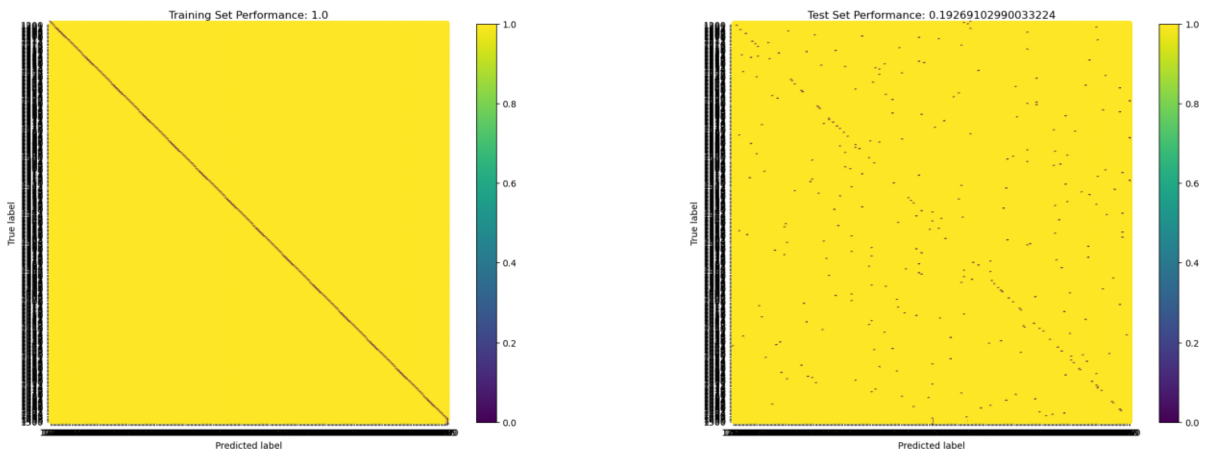


Figure 12: Top-1 testing set accuracy is 19.3%.

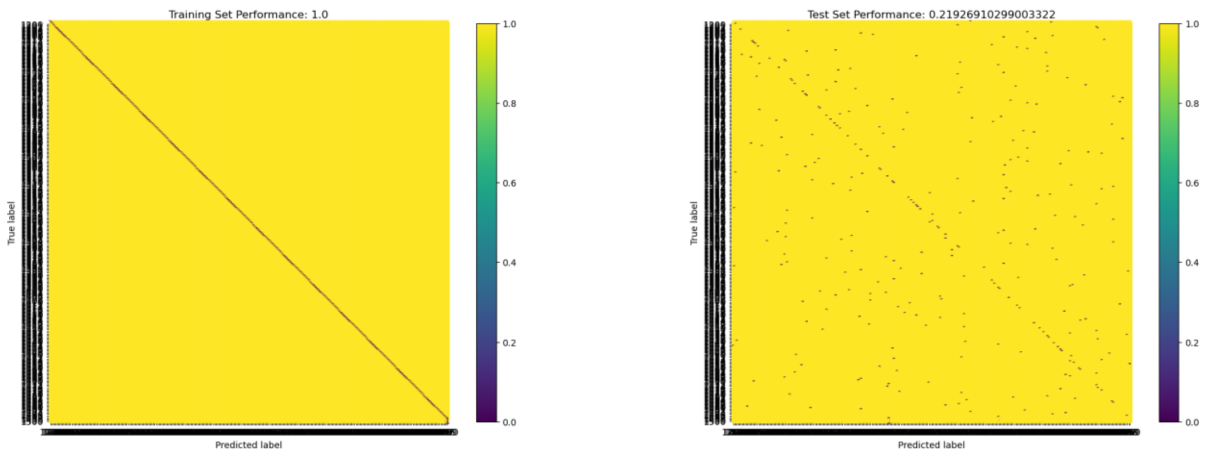


Figure 13: Top-5 testing set accuracy is 21.9%

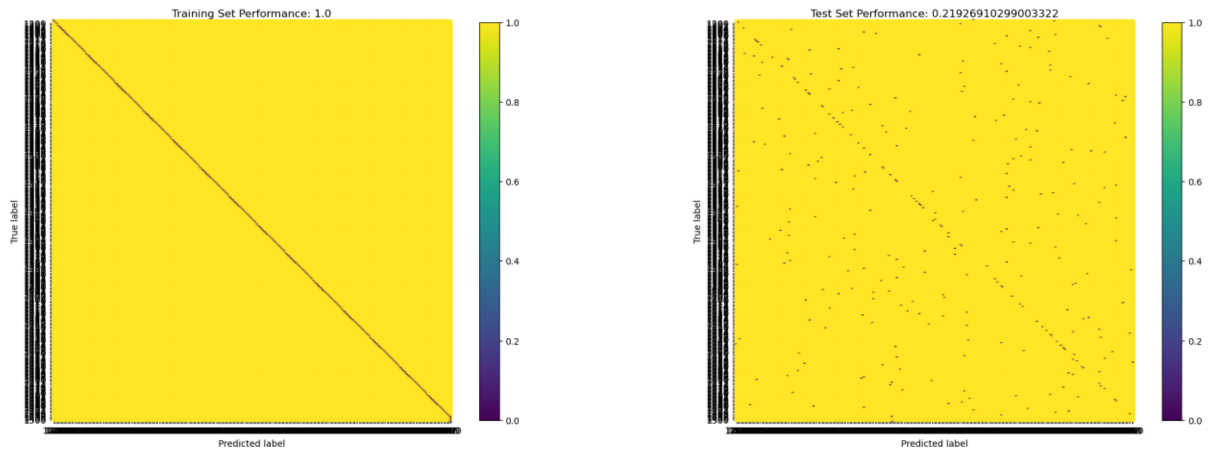


Figure 14: Top-10 testing set accuracy is 21.9%

The deep-learning method has more than twice the level of accuracy in person re-identification in comparison to the non-deep learning PCA method (19.3% vs 8.97% for Top-1 accuracy, 21.9% vs 9.3% for Top-5 accuracy, and 21.9% vs 8.97% for Top-10 accuracy). This is expected when considering the difference in computational efficiency and runtime between the PCA and deep learning method. The deep learning method far takes longer to train at the benefit of increased accuracy when identifying images of people in the dataset. Considering the low numbers of accuracy for both methods, the increase in training time and complexity for the deep learning method is well worth the increase in accuracy.

	Non-Deep Learning Method	Deep Learning Method
Top-1 Accuracy	19.3%	8.97%
Top-5 Accuracy	21.9%	9.3%
Top-10 Accuracy	21.9%	8.97%

Table 1: Top-N accuracy comparisons between the non-deep learning and deep learning methods.

1.4 Ethical Considerations of Person Re-identification

There are a variety of ethical considerations in the space of machine learning and deep learning pertaining to tasks involving person re-identification. There have been many instances throughout the past whereby people unknowingly have had their identities, whether that be images of their face or their entire body, used without their knowledge for a variety of purposes by a variety of groups. In some cases, these purposes have been used to recognise some rather unsavoury identities, such as those of suspected criminals, illegal aliens, and the like. Seemingly ever-present, many groups have utilised the internet to engage for their own individual purposes, and due to the nature of the web it is not unnatural that they may collect data from users without their consent or knowledge, at the massive expense of their breach in digital privacy and security. Data sources include social media pages and public videos and recordings. A majority of these datasets and models are used in security and surveillance research by both public and private organisations, such as the Iarpa Janus Benchmark-C dataset used by a U.S. government body for intelligence research, which includes a total of upwards of 20,000 images of people's faces without any of their knowledge or consent. Some limitations of these models include the possibility that some of these identities only have partial visibility or may have lighting inconsistencies, vastly different conditions to the way the models have been initially trained with high-definition and front-facing images. This is one of the reasons why spontaneous and candid images are sought after by these organisations, as they better simulate real-life such that the model can be further trained on a variety of identities and conditions to improve their accuracy (Madhumita Murgia & Harlow, 2019).

Problem 2. Multi-Task Learning and Fine Tuning

2.1 Data Pre-Processing

Images were obtained from the Oxford-IIIT Pets Dataset using the helper methods provided in the template. Images were loaded at a size of 256 x 256 to capture as much information about the cats and dogs images as possible without sacrificing network complexity in the future as well as using an acceptable size when considering computational restrictions. A batch size of 128 was used to capture a significant number of images for properly training the network or model acceptably in the future. No further resizing or conversion to grayscale was done to maintain as much information in the images as possible. With the use of the helper functions, segmentation masked images were also gathered. Originally, these masks were 1-index but were converted to 0-index and 32-bit float for the purposes of working with tensorflow. The final size of the images in the dataset is (128, 256, 256, 3) for the normal images and (128, 256, 256, 1) for the segmentation masked images. These images were used for both the from-scratch and fine-tuned network.

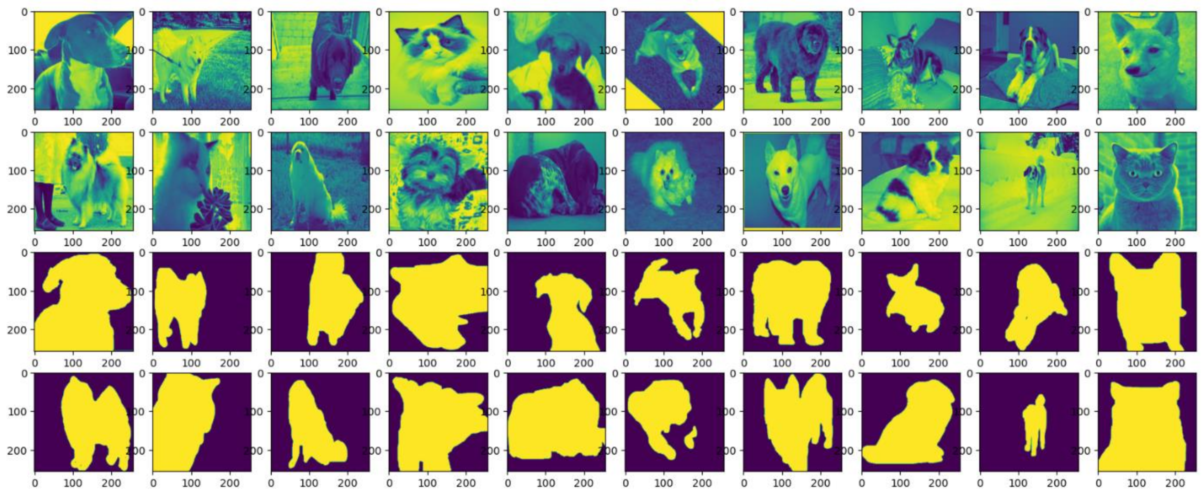
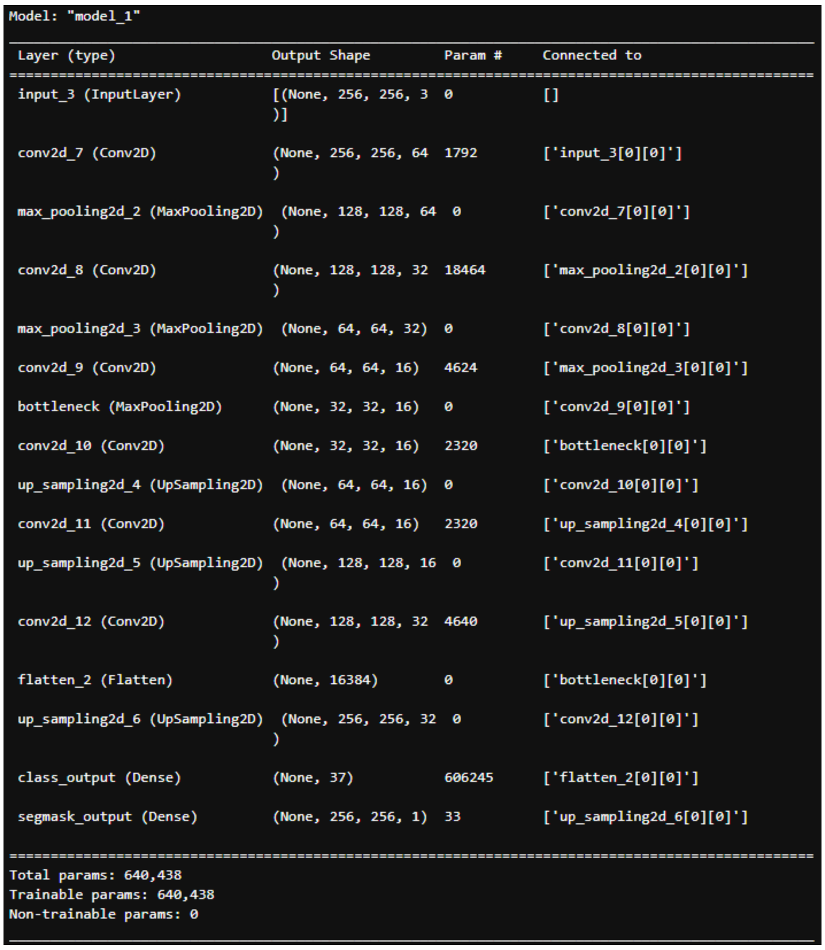


Figure 15: Sample of the images from the training dataset.

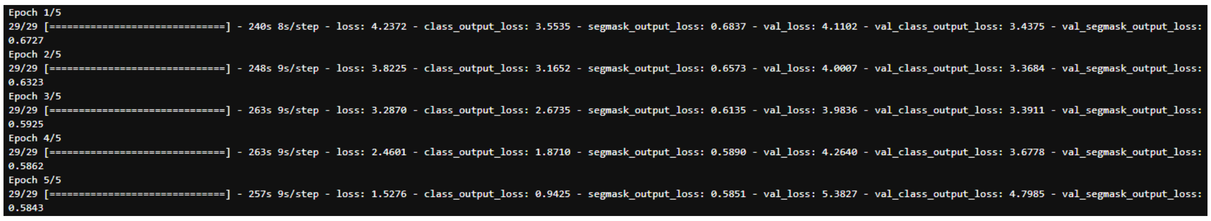
2.2 From-Scratch and Fine-Tuned Method Details

The from-scratch network utilises an autoencoder, multitask learning approach. It takes an input image of size (256, 256, 3), or the size of our images. It then has three 3 x 3 Conv2D and 2 x 2 MaxPooling2D layers for the first output (image classification task to identify cat or dog breed). Each of the three Conv2D layers have ReLU activation functions and 'same' padding, with 64, 32, and 16 filters respectively. Then we have a dense output layer with 37 classes (breeds of either cats or dogs) with a softmax activation. For the second output (semantic segmentation) we have three 3 x 3 Conv2D and 2 x 2 UpSampling2D layers. Each of the three Conv2D layers also have ReLU activation functions and 'same' padding, but have 16, 16, and 32 filters respectively. The dense output layer for the segmentation mask uses a sigmoid activation for the binary task of identifying whether a masked image is either a cat or a dog. There is thus two outputs of this model, one to classify breeds from a cat or dog image, and one to identify whether a masked image is a cat or dog. This network was compiled using the Adam optimiser and a sparse categorical cross entropy loss function for the image classification task and a binary cross entropy loss function for the semantic segmentation task, then fitted against the testing dataset with a batch size of 128 and 5 epochs taking into account model training time, task complexities, as well as the number of classes (breeds) to classify.



```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 256, 256, 3  0           []
                                )]

 conv2d_7 (Conv2D)              (None, 256, 256, 64  1792        ['input_3[0][0]']
                                )

 max_pooling2d_2 (MaxPooling2D)  (None, 128, 128, 64  0          ['conv2d_7[0][0]']
                                )

 conv2d_8 (Conv2D)              (None, 128, 128, 32  18464       ['max_pooling2d_2[0][0]']
                                )

 max_pooling2d_3 (MaxPooling2D)  (None, 64, 64, 32)  0           ['conv2d_8[0][0]']

 conv2d_9 (Conv2D)              (None, 64, 64, 16)   4624        ['max_pooling2d_3[0][0]']

 bottleneck (MaxPooling2D)      (None, 32, 32, 16)   0           ['conv2d_9[0][0]']

 conv2d_10 (Conv2D)             (None, 32, 32, 16)   2320        ['bottleneck[0][0]']

 up_sampling2d_4 (UpSampling2D)  (None, 64, 64, 16)  0           ['conv2d_10[0][0]']

 conv2d_11 (Conv2D)             (None, 64, 64, 16)   2320        ['up_sampling2d_4[0][0]']

 up_sampling2d_5 (UpSampling2D)  (None, 128, 128, 16  0          ['conv2d_11[0][0]']
                                )

 conv2d_12 (Conv2D)             (None, 128, 128, 32  4640        ['up_sampling2d_5[0][0]']
                                )

 flatten_2 (Flatten)            (None, 16384)        0           ['bottleneck[0][0]']

 up_sampling2d_6 (UpSampling2D)  (None, 256, 256, 32  0          ['conv2d_12[0][0]']
                                )

 class_output (Dense)           (None, 37)           606245      ['flatten_2[0][0]']

 segmask_output (Dense)         (None, 256, 256, 1)  33          ['up_sampling2d_6[0][0]']

==================================================================================================
Total params: 640,438
Trainable params: 640,438
Non-trainable params: 0
__________________________________________________________________________________________________
```

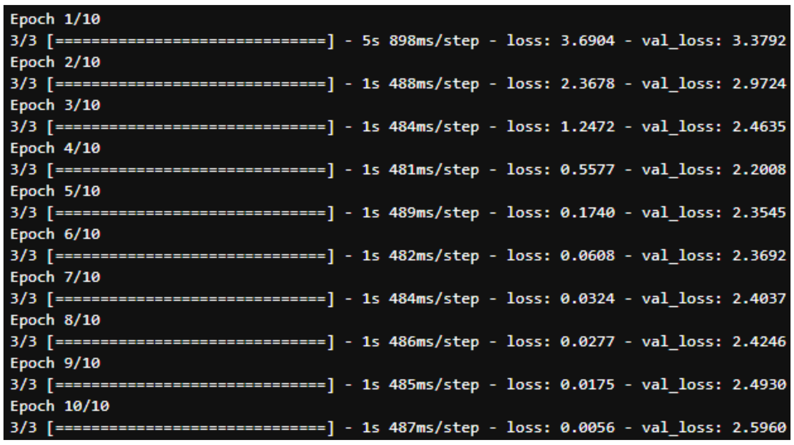
Figure 16: Summary of the from-scratch model. There is a total of 640,438 parameters.



```
Epoch 1/5
29/29 [==============================] - 240s 8s/step - loss: 4.2372 - class_output_loss: 3.5535 - segmask_output_loss: 0.6837 - val_loss: 4.1102 - val_class_output_loss: 3.4375 - val_segmask_output_loss:
0.6727
Epoch 2/5
29/29 [==============================] - 248s 9s/step - loss: 3.8225 - class_output_loss: 3.1652 - segmask_output_loss: 0.6573 - val_loss: 4.0007 - val_class_output_loss: 3.3684 - val_segmask_output_loss:
0.6323
Epoch 3/5
29/29 [==============================] - 263s 9s/step - loss: 3.2870 - class_output_loss: 2.6735 - segmask_output_loss: 0.6135 - val_loss: 3.9836 - val_class_output_loss: 3.3911 - val_segmask_output_loss:
0.5925
Epoch 4/5
29/29 [==============================] - 263s 9s/step - loss: 2.4601 - class_output_loss: 1.8710 - segmask_output_loss: 0.5890 - val_loss: 4.2640 - val_class_output_loss: 3.6778 - val_segmask_output_loss:
0.5862
Epoch 5/5
29/29 [==============================] - 257s 9s/step - loss: 1.5276 - class_output_loss: 0.9425 - segmask_output_loss: 0.5851 - val_loss: 5.3827 - val_class_output_loss: 4.7985 - val_segmask_output_loss:
0.5843
```

Figure 17: From-scratch model training statistics.

For the fine-tuned approach using the pre-trained MobileNetV3Small model, we apply a GlobalAveragePooling2D layer to the original model. Then, we add four dense layers with ReLU activation to classify results better and learn more complex functions. The first two dense layers have 1024 filters, the third has 512, and the final layer has 37 layers for the 37 breeds of cats and dogs in the dataset with softmax activation (Sharky, 2019). The model was also compiled using the Adam optimiser with categorical cross entropy and binary cross entropy for the two tasks. It was then trained against the testing set with a batch size of 32 and 10 epochs taking into account model training time and computational restrictions.



```
Epoch 1/10
3/3 [==============================] - 5s 898ms/step - loss: 3.6904 - val_loss: 3.3792
Epoch 2/10
3/3 [==============================] - 1s 488ms/step - loss: 2.3678 - val_loss: 2.9724
Epoch 3/10
3/3 [==============================] - 1s 484ms/step - loss: 1.2472 - val_loss: 2.4635
Epoch 4/10
3/3 [==============================] - 1s 481ms/step - loss: 0.5577 - val_loss: 2.2008
Epoch 5/10
3/3 [==============================] - 1s 489ms/step - loss: 0.1740 - val_loss: 2.3545
Epoch 6/10
3/3 [==============================] - 1s 482ms/step - loss: 0.0608 - val_loss: 2.3692
Epoch 7/10
3/3 [==============================] - 1s 484ms/step - loss: 0.0324 - val_loss: 2.4037
Epoch 8/10
3/3 [==============================] - 1s 486ms/step - loss: 0.0277 - val_loss: 2.4246
Epoch 9/10
3/3 [==============================] - 1s 485ms/step - loss: 0.0175 - val_loss: 2.4930
Epoch 10/10
3/3 [==============================] - 1s 487ms/step - loss: 0.0056 - val_loss: 2.5960
```

Figure 18: Fine-tuned model training statistics.

2.3 Method Evaluations

The model was predicted on the testing set, by which we can then create confusion matrices, embeddings, and potential KNNs to evaluate.

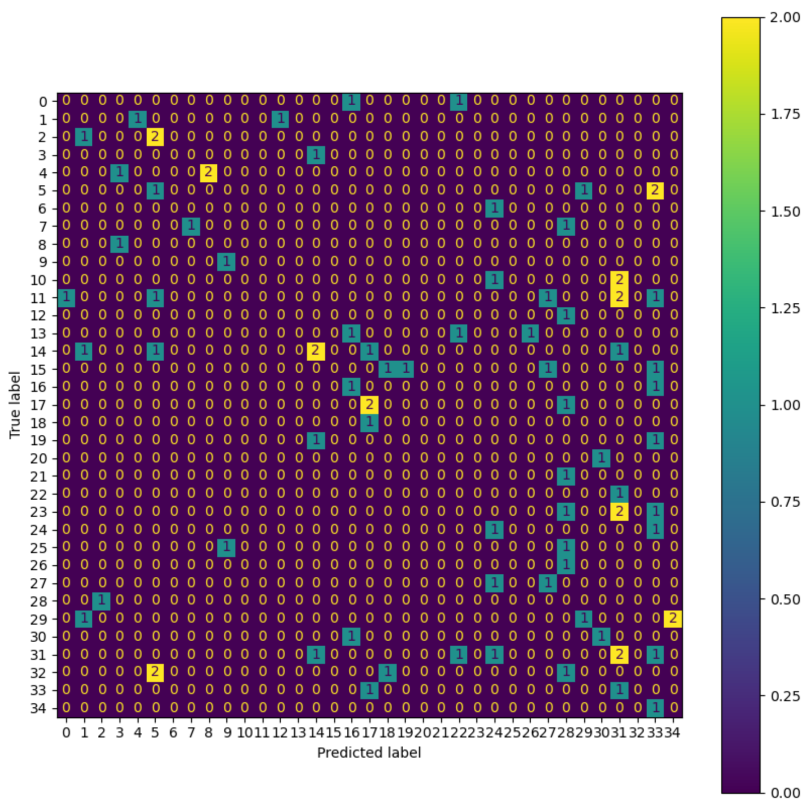


Figure 19: Confusion matrix for from-scratch image classification.

Visually, our confusion matrix does not look very good. This means that the model is continuously predicting the incorrect breeds for many images of cats and dogs passed through the model from the dataset. We can pull out data from the 'bottleneck' layer, or the final MaxPooling2D layer for the first output of our model and then extract embeddings and use t-SNE.

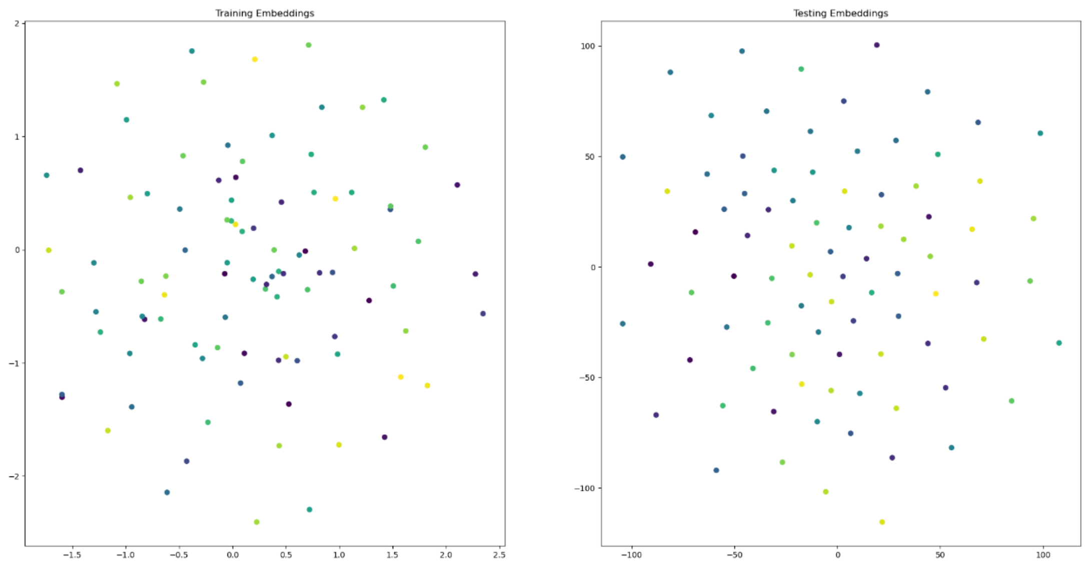


Figure 20: t-SNE embeddings for image classification using the from-scratch network.

This also does not look too great visually. Some identical breed classifications are somewhat near to each other and separated from other breeds, but not enough to conclude that our model does the classification task reasonably. From these embeddings, we can train a K Neighbors Classifier using 10 n_neighbors, 'distance' weights function, and -1 n_jobs to numerically output the performance of our model on the testing set.

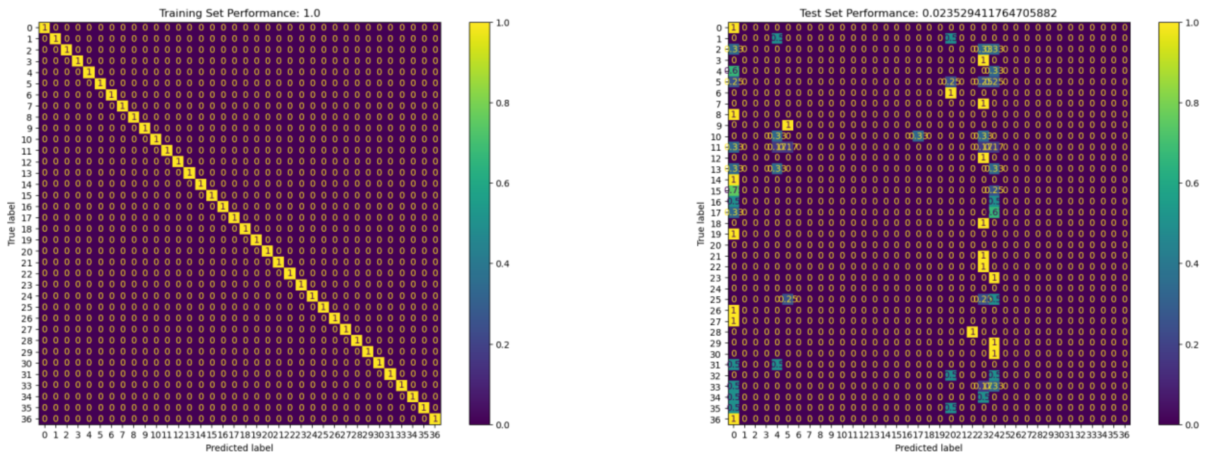


Figure 21: KNN for image classification using the from-scratch network. The performance is 2.35%

For the fine-tuned approach, we can follow a similar workflow when evaluating performance.

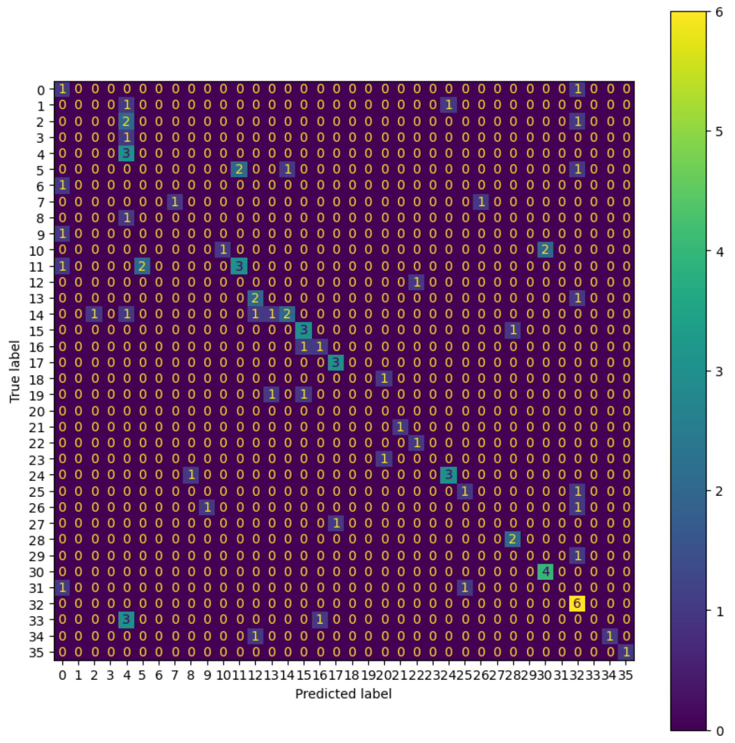


Figure 22: Confusion matrix for fine-tuned image classification.

Visually, our confusion matrix for the fine-tuned network when classifying breeds from cats and dogs images seems to show an improvement over our from-scratch network. There is a somewhat clearer diagonal on the confusion matrix, signifying the network has improved on correctly identifying breeds from the images in the testing set, or certainly an improvement over the from-scratch network.

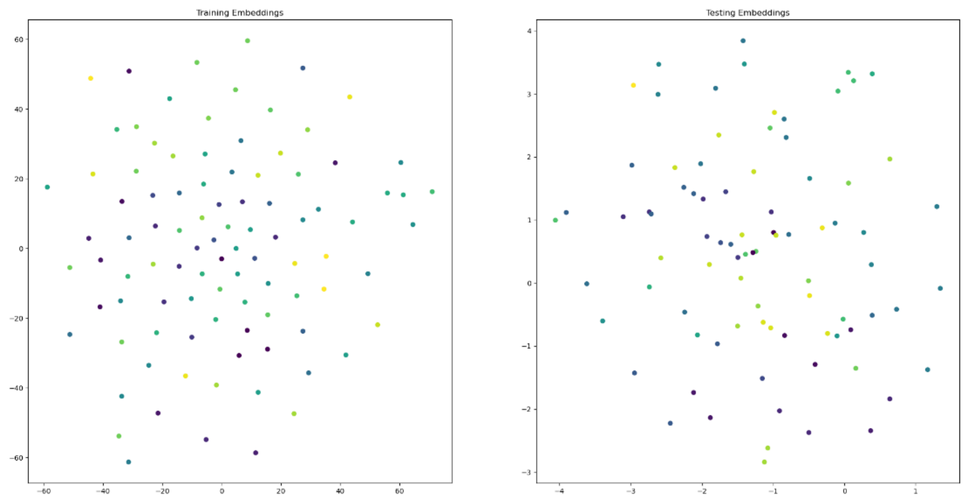


Figure 23: t-SNE embeddings for image classification using the fine-tuned network.

We see a similar visual performance on our t-SNE embeddings. Some points are somewhat near to each other from the same breeds, though it is difficult due to the sparse nature of the points.

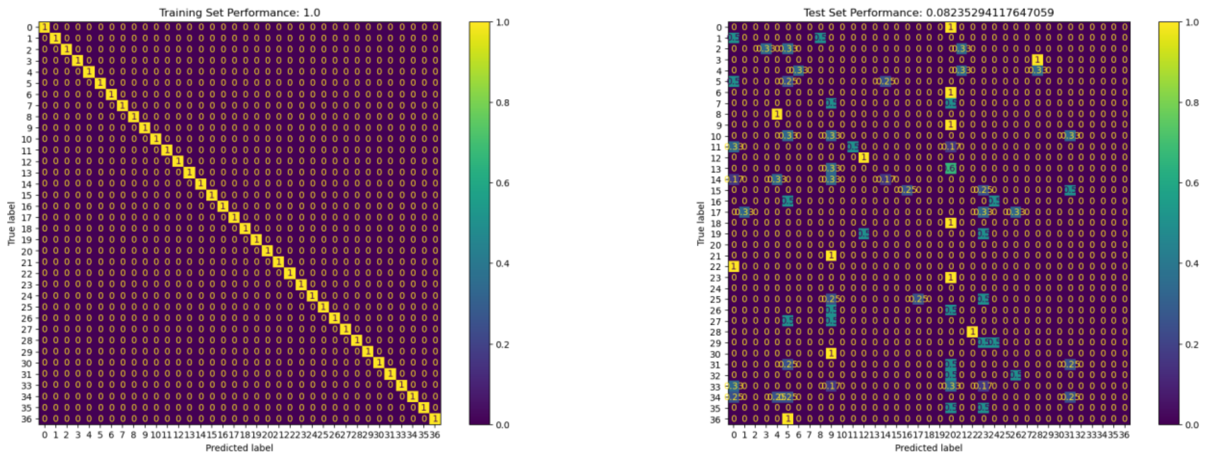


Figure 24: KNN for image classification using the from-scratch network. The performance is 8.24%

This numerically proves the assertion that the fine-tuned network improves on the performance of the from-scratch network. The fine-tuned approach improves accuracy by nearly four times, 8.24% vs 2.35%, for the image classification task. It is important to note that results may change if the number of epochs used to train either approach were changed. If more epochs were used to train the fine-tuned approach in the future, there could be an even more significant improvement on accuracy.

2.4 Discussion of Performance and Issue Mitigations

A variety of issues were found when implementing both the from-scratch and fine-tuned models and solutions. The from-scratch model was largely adapted from the week 10 lecture and tutorial content of CAB420 Machine Learning, and though the tasks by which they were developed from were similar, it required some mitigations and changes to suit the image recognition and semantic segmentation tasks at hand here. The filters used for the Conv2D blocks were one such instance, where different numbers of filters were used, attempted, and experimented on to achieve an acceptable level of accuracy without reaching the limits of computational restraints. In some cases, too large number of filters were used such that the computer these models were trained on fully crashed to due to the overwhelming computational load and required a hard restart. Different loss functions for both models were also used, such as initially using categorical cross entropy for image recognition instead of sparse categorical cross entropy, but it was found that the latter performed better as well as saving time in computation and memory as it uses a single integer for a class instead of a whole vector (Skadaver, 2019). Sparse categorical cross entropy also better suited the dataset as each class in the dataset are mutually exclusive (Master M, 2018), or in other words each cat or dog in the dataset belongs to one breed and one breed only. For both fits of the from-scratch and fine-tuned models, different batch size and epoch values were experimented taking into account training times and accuracies achieved and the computational restrictions at hand. Smaller batch sizes and larger epochs resulted in an increase in training times, and it was found that the resulting increase in accuracy while testing was inequitable to the time increase. In the future, it may be beneficial to further test these values and find the best middle ground between accuracy and training times.

References

1. Madhumita Murgia, & Harlow, M. (2019, September 18). *Who's using your face? The ugly truth about facial recognition*. Financial Times. https://www.ft.com/content/cf19b956-60a2-11e9-b285-3acd5d43599e
2. Sharky. (2019, May 30). *How to add a few layers before the model in transfer learning with tensorflow*. Stack Overflow. https://stackoverflow.com/questions/56369228/how-to-add-a-few-layers-before-the-model-in-transfer-learning-with-tensorflow
3. Skadaver. (2019, August 5). *Cross Entropy vs. Sparse Cross Entropy: When to use one over the other*. Cross Validated Stack Exchange. https://stats.stackexchange.com/questions/326065/cross-entropy-vs-sparse-cross-entropy-when-to-use-one-over-the-other
4. Master M. (2018, December 1). *Sparse_categorical_crossentropy vs categorical_crossentropy (keras, accuracy)*. Data Science Stack Exchange. https://datascience.stackexchange.com/questions/41921/sparse-categorical-crossentropy-vs-categorical-crossentropy-keras-accuracy