Question 1: Person Re-Identification

1.1 Data Pre-Processing

Images show the entire person at the center. With retrospect to memory-heavy deep-learning network chosen, images have been resized from 128 x 64 → 64 x 32 and gray scaled. Given task of person re-identification, these pre-processing decisions have not impacted image recognisability, a concern if images had been cropped rather than resized.

Figure 1: Gallery Samples: Top row shows sample images before resize and gray-scaling, while bottom images show vice versa. Captions above each column denote image ID.

Additionally, dataset has been normalised through standardisation to assist the chosen non-deep-learning method as it generally ensures equal treatment of features and improves variance explanation and interpretability. This can greatly benefit some methods (PCA) or because of the process of computing scatter matrices (LDA), it will have no effect on the results.

1.2 Network Design and Training

The task specifies a non-deep learning image reduction and deep learning-based re-identification network be trained and evaluated for person-reidentification. For a non-DL approach, the options are generally limited to LDA (Linear Discriminant Analysis) or PCA (Principal Component Analysis). The main difference between the two is that LDA is supervised (meaning it takes class labels into account) and PCA is not. With person-reidentification as this project's objective, classification is not necessarily important. Robustness to variation and matching accuracy are.

In this case, both models have been already trained through sklearn automatically, though only one will have a further analysis in this report. Figures 2 & 3 indicate the cumulative variance within the dataset and each model's ability to capture it within a number of components. With them, it can be seen that:

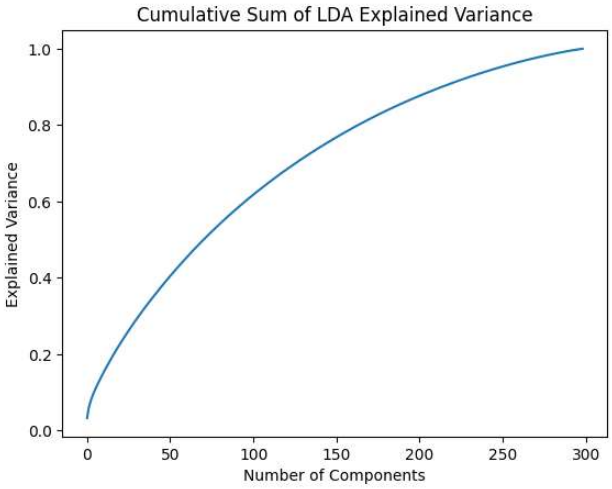

Figure 2: A 'cumsum' (cumulative sum) of the explained variance ratio within the LDA model. X indicates the number of features required to achieve high discriminative power, while Y shows the variance explained.

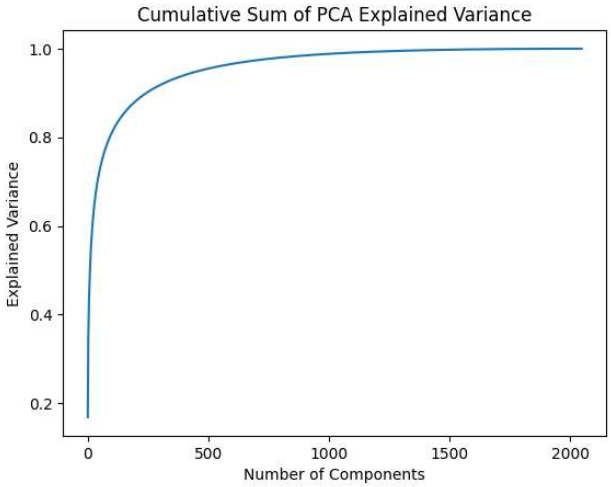

Figure 3: A 'cumsum' (cumulative sum) of the explained variance ratio within the PCA model. X indicates the number of features required to achieve high discriminative power, while Y shows the variance explained.

- The rapid increase indicates that the principal components capture most of the variance in the original data in a few components. Suggesting that they are effectively summarising the data, which is desirable, though it does not necessarily preserve discriminative information.
- By contrast, the slower increase towards 1 indicates that LDA is working properly, that it is maxmising the separability between classes while projecting the data onto a lower-dimesional space.

Because this task desires dimensionality reduction and variance explanation over classification / class separability, PCA was selected over LDA. Its notable robustness to outliers and noise are ideal for re-identification.

As for the chosen DL-method, there are realistically only three choices, though out of all of them, a Siamese Triplet network was preferred. In the case of standard / contrastive loss in a

Siamese network, standard loss is just a typical binary cross-entropy loss (aka MSE loss), whereas contrastive loss computes the distance between pairs of samples in the embedding space to optimize the model to minimize the distance between similar pairs / vice versa. Triplet loss in this circumstance goes a step further, incorporating a third 'negative' sample in addition to the standard anchor and positive samples, aiming to ensure that the anchor-positive distance is smaller than the anchor-negative distance. Because of this, the Triplet network tends to learn a more robust representation, and why the Siamese Triplet network will be utlised in the future report.

For its design, for the most part it's an identical replication of the VGG_net Triplet network used in CAB420_Metric_Learning_Example_3_Triplet_Loss.ipynb. It only truly differs with the network size used, from 28, 28, 1 → 64, 32, 1. This existing architecture was preferred for the network as it simplifies handling the three inputs (anchor, positive and negative). In this setup, the loss function is a layer within the network and the other layers in the network are effectively hidden. See *Appendix A* Figures 16 & 17 for a full depiction of the network.

1.3 Network Evaluation

Cumulative Matching Characteristic (CMC) curves have been generated for the testing 'probe' and 'gallery' sets by utilizing a set of features along with their associated IDs. This was achieved through the utilization of functions such as 'get_ranked_histogram_l1_distance', 'ranked_hist_to_CMC', and 'plot_cmc'. Figure 4 displays the resultant curves from these functions, depicting a noticeable contrast between the 'logarithmic' progression of the Triplet network towards a 'Count' / match-rate of 1 and the nearly linear progression of PCA. It suggests that the Siamese network achieves higher identification accuracy rapidly with each additional match. While both methods achieve a peak at (or just below) 1, the triplet network may exhibit greater efficacy across various ranks, especially for top-ranked matches. This implies that the triplet network could potentially outperform PCA in scenarios where swift and accurate identification is necessary, such as surveillance.

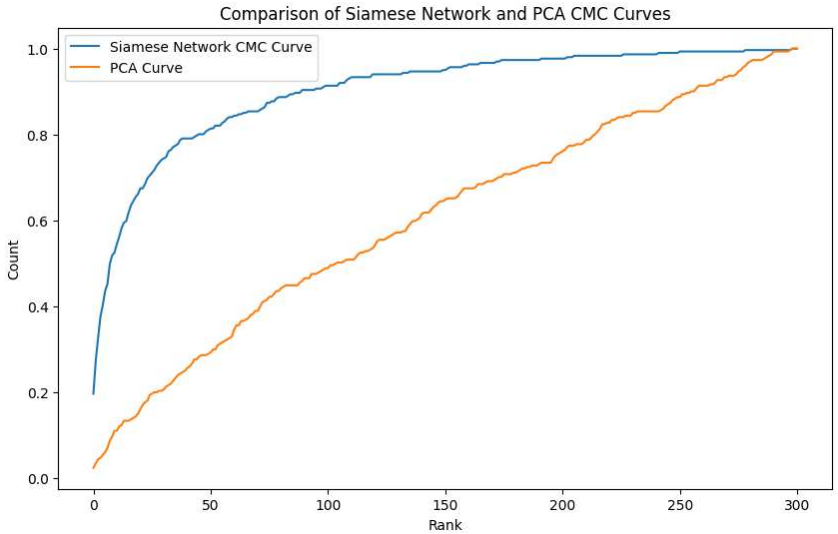

Figure 4: CMC curve for PCA & Siamese Triplet Network, going across identification rank/count on the y-axis and feature count or 'rank' across the x-axis.

This implication is furthered by reviewing Top-N performance between both networks. Table 1 shows corresponding outputs for Top-N performance from 1 to 10. The difference in accuracy

between the two networks widens as the rank increases and the triplet network outperforms PCA more significantly as matches are considered, in accordance with the trend identified in Figure 4.

Network	Top-N Accuracy									
	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
Triplet Siamese	0.196013	0.272425	0.325581	0.375415	0.401993	0.435216	0.451827	0.498339	0.518272	0.524917
PCA	0.023256	0.033223	0.043189	0.046512	0.053156	0.059801	0.069767	0.086379	0.093635	0.109635

Table 1: Top-N Accuracy Values obtained going from Top-1 to Top-10 between both networks.

These results are promising. Given how small the network is at Top-1, a 0.196 score for triplet is very high, as is 0.4 for Top-5. However, the network certainly has flaws. The network is very simple, an embedding size 32 is quite small, and it hasn't been trained for very long, all decisions to conserve computational resources. Consequences of this direction are visible in the network embedding in Figure 5.

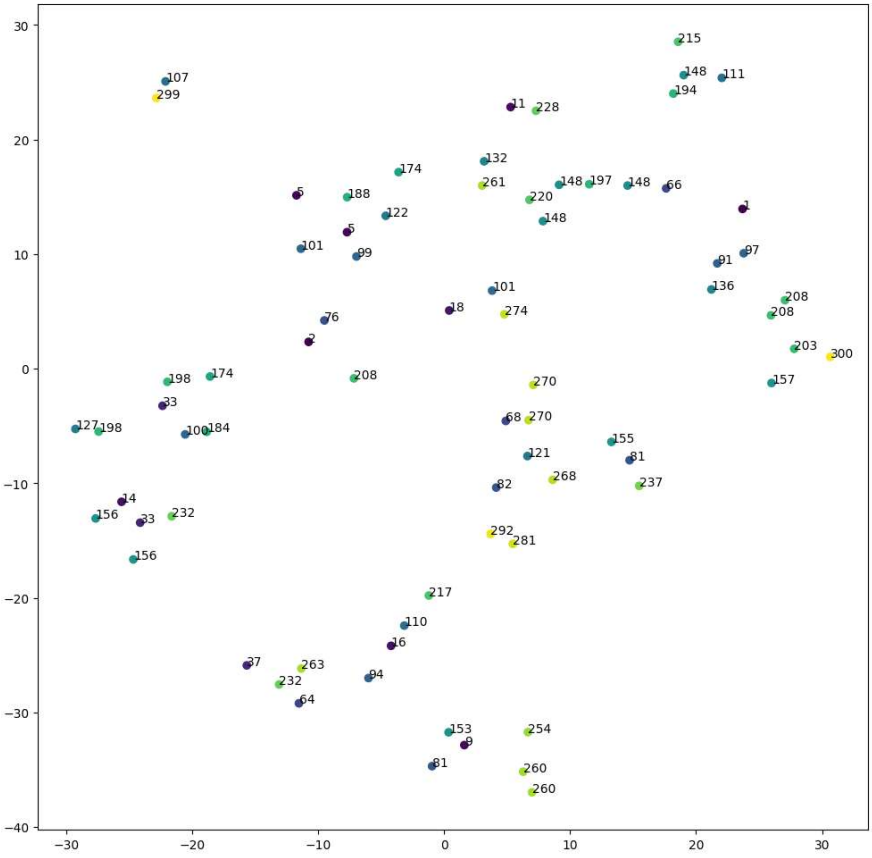

Figure 5: Embedding for Siamese Triplet Network using 75 outputs. Numbers are used to indicate pairing ID for each dot.

Pairs are generally together, but there is a significant amount of overlap. This suggests that there are clusters of data points which share similar characteristics or features. This can affect identification accuracy, as if the embeddings overlap too much there will be cases of misidentification. It also indicates that the current embedding method is failing to capture important variations between different individuals.

This confusion is carried over when reviewing Triplet distance results against the positive and negative embeddings. Seen in Figure 6, results are generally favourable, but in the case of anchor 7 the positive distance is greater than the negative distance, and it cannot be explained by image similarity.

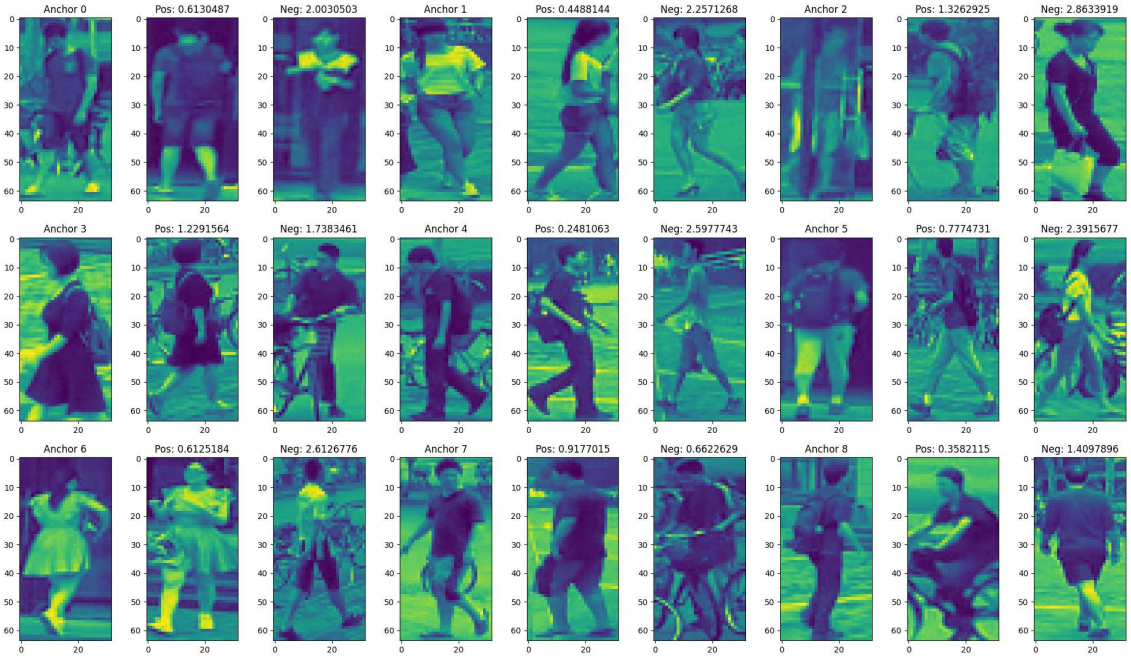

Figure 6: Embedding distances between anchor, positive and negative images, displayed above the positive and negative image.

It is almost certain that these flaws are the result of network design. The VGG network carried over verbatim is clearly not appropriate for the task, regardless of the necessary effect it has on model performance. It doesn't compare with PCA, in-spite of poor re-identification accuracy, manages to finish its operation ~45 times faster across several tests, with an average time of ~5 seconds (round up) over Triplet's ~228 seconds (round up), which is a result of non-dl's inherent simplicity, at least within 'from scratch' applications.

1.4 Ethical Considerations

The implication for person re-identification lies in its name. There is a risk inherent to the individual and their right to privacy and civil liberties when authority is given to a machine. Never in history have organisations been given the tools for widespread, automated surveillance that determines ones' own likeness by mathematical similarities.

This capability is only facilitated by vast quantities of likenesses to adequately train the networks for the real world. These can be harvested anywhere, and very little, if any, indication is given to its owner. An article by the financial times includes an investigation of an American Activist finding her face present in a government database used to recognize faces of interest. She states "The first images were from 2008, all the way through to 2015. They were taken in closed meetings, certainly private in the sense that it was me goofing around with friends rather than me on stage"

(Madhumita, 2019). She went on to explain another half-dozen had been clipped from YouTube videos of her speaking at events, on topics including freedom of expression and digital privacy.

It implies that organisations have the right to a person's face without their knowledge, let alone consent, to be used in training networks for and including Re-IDing systems. Widespread implementation of such as the proposed Social Credit system in China, detailed in a report from the university of Hong Kong (Shen, 2019) to "maintain political and social stability" impedes various human rights. It goes beyond mere Re-IDing and applies a draconic level of surveillance to its civilian population and their everyday tasks, purchases and activities. Especially as technology cannot achieve a moral obligation to zero false predictions, caused by lack of database variety, network biases or other failings.

Question 2: Multi-Tasking

2.1 Data Pre-Processing

The Oxford-IIIT Pet Dataset used displays images of cats and dogs with labels corresponding to their breed (for a total of 37 classes, 12 cats and 25 dogs) and semantic segmentation information labelling the foreground and background. Given the task calls for image classification and semantic segmentation simultaneously for two distinct networks, processing resources are limited compared to task requirements. Images have been adjusted from their native random dimensions (anywhere from 300 x 300 down to ~100 x ~100) to 128 x 128, both to simplify implementation and decrease processing time. An augment has also been applied which performs a left-right flip on 50% of the data to reduce expected overfitting. No further changes are made, and images remain in colour for recognising pet breed.

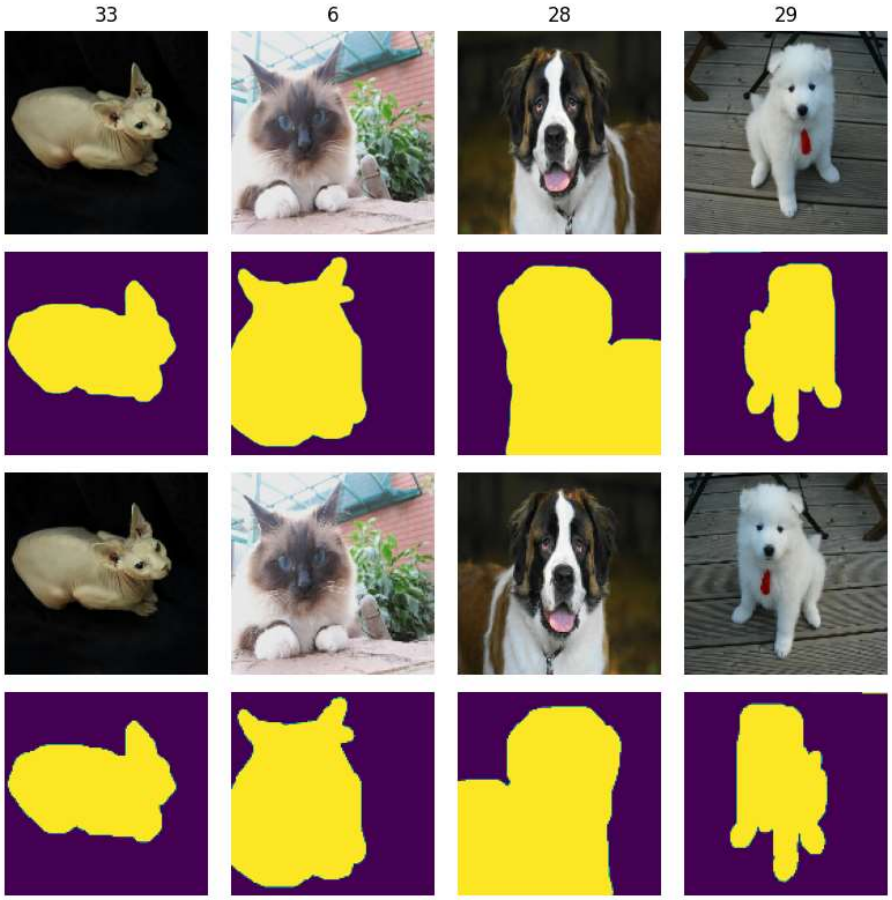

Figure 7: Data Samples: Top set of images & segmatic masks represent original data, while bottom set of image and segmatic mask represent dataset after pre-processing. Above each is class ID.

Both networks will use the pre-processed data for training and testing, with augmentation being detracted for testing. While the option for shuffling data is available given the data-loading method used, it is not for this project as it negatively affects training.

2.2 Network Implementation and Training

Specified in the project requirements, the objective is to implement two methods for semantic segmentation and classification and compare their performance. One must be designed from

scratch, the second 'fine-tuned' approach must use an existing model, namely the MobleNetV3Small from Keras. It can be assumed that both networks should utilise generally similar classification and semantic segmentation heads, with separate backbone layers. Considering expected performance problems, both output heads will need to balance effectiveness with simplicity.

After extracting features from the designated backbone network, both the semantic segmentation and classification heads are added to either classify the input images into different classes, or to generate segmentation masks. The classification head consists of three layers, starting with a Flatten layer to 'flatten' the feature maps from the backbone network into a 1D vector. This is passed through a ReLU activation function in the Dense layer to introduce non-linearity and learn complex decision boundaries. The final Dense layer uses a softmax activation output for class probabilities, which is suitable for multi-class classification tasks. A code snippet is available in Appendix B. As for segmentation, it is extensively similar to the decoder part of an autoencoder found in CAB420_Encoders_and_Decoders_Additional_Example_Semantic_Segmentation.ipynb. It primarily consists of a series of convolution layers followed by upsampling layers. The final convolutional layer uses a sigmoid activation to output binary segmentation masks. A code snippet is available in Appendix B. (Could be mentioned in the final part of the report instead of here.)

Only the backbone component differs between both networks. For the "from-scratch" network, it follows a design commonly used in CNN architectures for feature extraction. It consists primarily of convolutional layers -filtering upwards (32, 64, 128)- followed by max-pooling layers. ReLU activation helps introduce non-linearity to the model, enabling it to learn more complex patterns in the data, and Padding='same' ensures the spatial dimensions of the feature maps remain the same after convolution, preserving spatial information. A code snippet is available in Appendix B. For the "Fine-Tuned" MobileNetV3Small network, the backbone can be imported through keras. MobileNetV3Small is an in-built model, so the only changes necessary are to remove the top and pre-processing layers, which is as simple as changing a setting to False. Model was given a batch size of 450. The general design of the whole model is available as a code snippet in Appendix B.

A batch size of 32 was used and the network and was trained for 20 epochs on an 'adam' optimiser, though a callback was added to stop training it exceed 15 minutes. For simplicity, a validation set was not partitioned from the training set, instead the testing set is used for that purpose. Both newtorks operate on 'sparse categorical crosssentropy' and 'binary crossentropy' loss functions. The model summary can be seen in Appendix C under Figure 18.

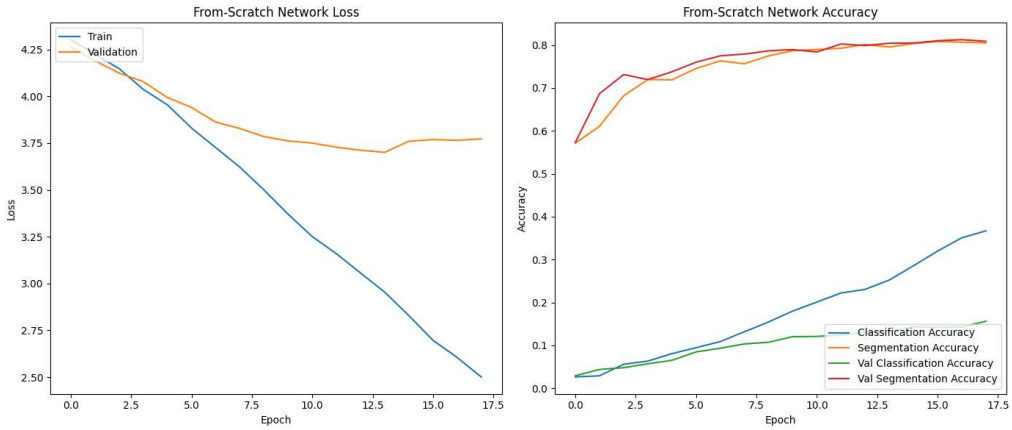

Figure 8: "From-Scratch" model training performance showing loss (left) and accuracy (right). Accuracy shows values for all outputs, including validation. Model loss functions' lack of convergence indicates it has not finished training after 17 epochs.

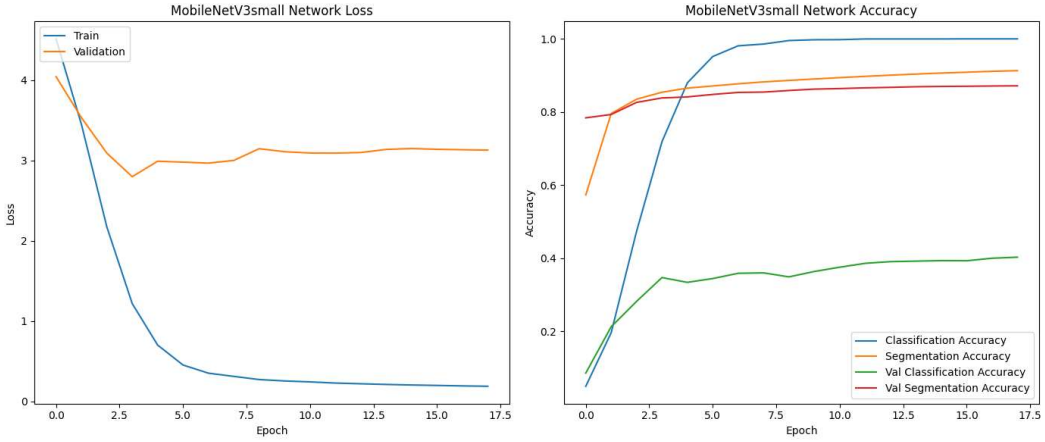

Figure 9: "Fine-Tuned" MobileNetV3small model training performance showing loss (left) and accuracy (right). Accuracy shows values for all outputs, including validation. Overfitting is observed, though it appears to have converged after 17 epochs.

Model performance for Figure 8 indicates the model has not finished training after 17 epochs and is overfitting. Whereas the "Fine-Tuned" model appears to have converged after 17 epochs with similar minor overfitting.

2.3 Network Evaluation

The "From-Scratch" network classification results can be seen in Figure 10 through a confusion chart. Weighted F1 scores of ~0.361 and ~0.152 are obtained for training and testing sets. These low scores, especially for the training set, indicate the model has not truly finished training.

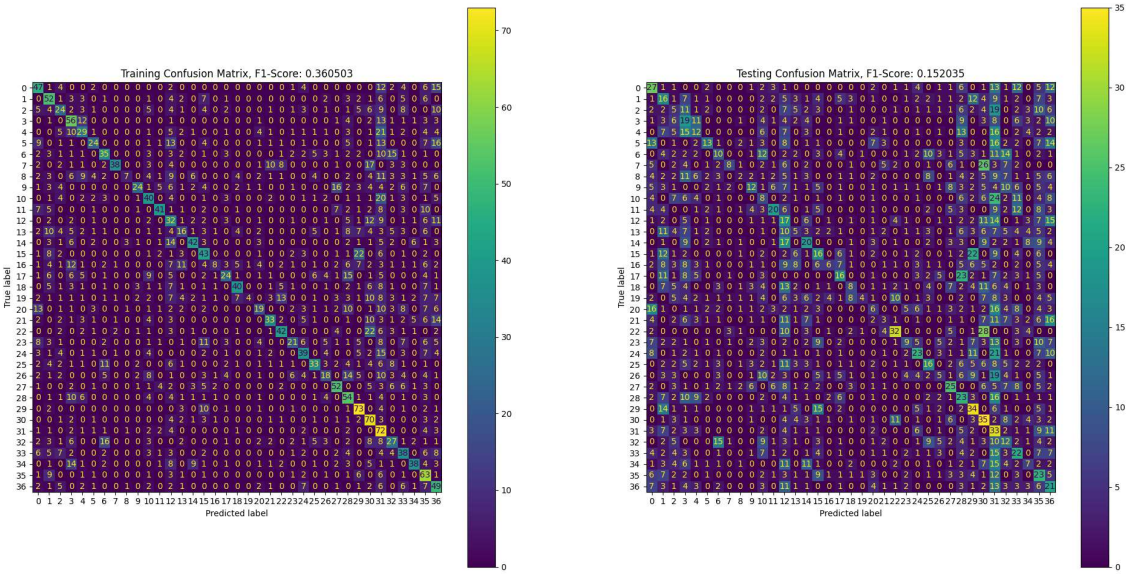

Figure 10: Confusion charts for training (left) and testing (right) sets. Unfinished training is clearly visible in the testing set from consistent misclassification and low F1-Scores.

Figure 11 explores general segmentation mapping results for the “From Scratch” network results. Through them, it can be observed that:

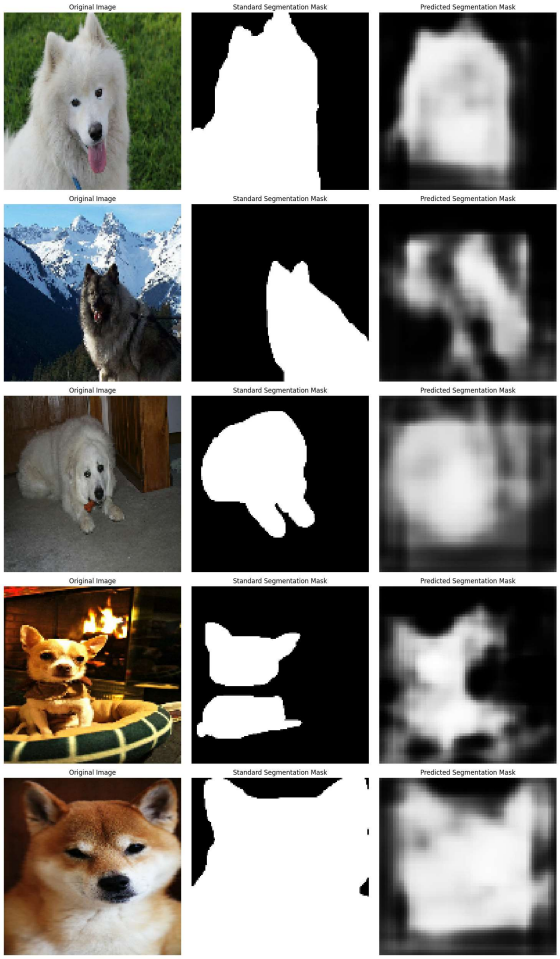

Figure 11: Comparison of original images (left), true segmentation masks (middle) and predicted results from the "From Scratch" model (right).

- Finer details and more complex shapes are often missed within the prediction results. This may be a result of the output shape seen in Appendix C Figure 18, as a single channel may not capture enough information or vice versa. It is also possible that the model is struggling to recover spatial details due to upscaling or just as easily be the result of a lack of training.
- Complex backgrounds (angular shapes and multiple colours and varied lighting) have a heavy impact on the segmentation prediction. It suggests that the model is struggling to differentiate between foreground objects and background clutter, which may be a result of its design or lack of training or both.

Overall performance of the "From Scratch" model will be reviewed in tandem with the "Fine Tuned" model. The "Fine-Tuned" network classification results can be seen in Figure 12 through a confusion chart. Weighted F1 scores of ~0.622 and ~0.393 are obtained for training and testing sets, which is acceptable. An article by the University of Oxford evaluating this dataset produced a classification testing set accuracy of only ~0.5 (Parkhi, Vedaldi, Zisserman, & Jawahar, 2021), so ~40% is considered acceptable. Overfitting observed in Figure 9 is apparent, with testing set performance significantly below the training set. Overall performance is still better than the "From-Scratch" model, suggesting a failure in training time, backbone design or both.

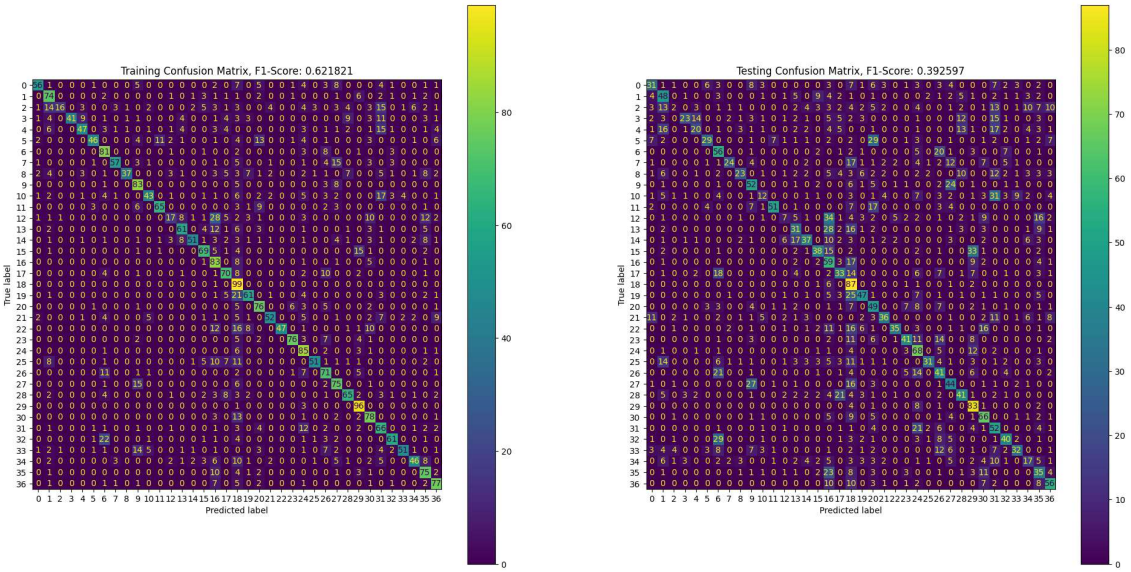

Figure 12: Confusion charts for training (left) and testing (right) sets on the "Fine Tuned" model. Overfitting can be observed within the testing set.

Figure 13 explores general segmentation mapping results for the "Fine Tuned" network results. Through them, it can be observed that:

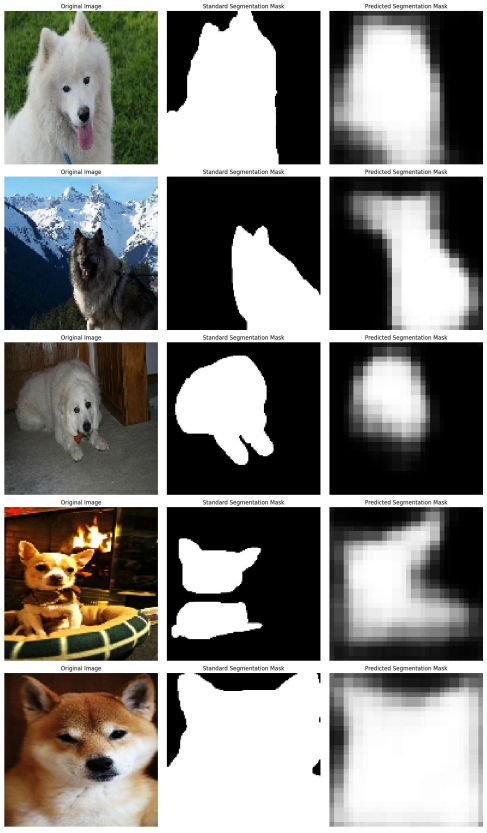

Figure 13: Comparison of original images (left), true segmentation masks (middle) and predicted results from the "Fine Tuned" model (right).

- The more 'pixelated' results over Figure 11 are almost certainly a consequence of the more drastic up-scaling used for two of the three upscaling layers present in the segmentation head used for the "Fine Tuned" model. This was necessary, as it produces the necessary output size of (None, 128, 128, …), but it does impact performance.
- The "Fine Tuned" model appears drastically more competent when capturing the finer details and shapes present in the images compared to Figure 11. This suggests that either the "Fine Tuned" model backbone is better for performance, or that a lack of training was indeed the cause for the "From Scratch" models' performance. Though it should be stated that this model is still far from perfect, experiencing similar issues identified in Figure 11.

Overall, it is likely that the poor performance identified for the "From Scratch" model is a result of a lack of finalised training and/or the design of its segmentation head and backbone layer. The "Fine Tuned" model shares several of the same issues, as its segmentation results perform poorly against smaller details.

2.4 Improvements

Poor performance on semantic segmentation outputs across both models can be credited to design focus on computational performance rather than model performance. In order to improve model performance, both models will be trained for the full 20 epochs rather than 15 minutes. This should reduce the undertraining identified for the "From Scratch" network without overfitting the "Fine Tuned" network. Further epochs have not been considered for this reason and instead batch size has been increased from 450 to 600. By exposing the model to more examples and allowing it to learn from the data over a longer period, it can capture more complex patterns and relationships in the data, which is what it appears to be missing.

The backbone layer of the "From Scratch" model has been identified as a significant factor contributing to suboptimal performance for both classification and semantic segmentation tasks. Its current design lacks the depth necessary to capture complex patterns in the data effectively. One solution is to enhance the model's depth by incorporating additional convolutional layers. Pairing existing convolutional layers can facilitate the extraction of more intricate features from the input data, thereby improving the model's ability to discern meaningful patterns.

Subpar performance on semantic segmentation, particularly in capturing finer details can be attributed to the segmentation head's design. While upscaling is necessary for preserving spatial context, it often compromises finer details. An alternative approach would be to use Conv2DTranspose layers instead of convolution and upscaling layers. Conv2DTranspose layers perform learnable upsampling by learning interpolation during training. This approach offers greater flexibility and adaptability in capturing dependencies and features during upsampling, albeit at a higher computational cost. Allowing the model to train for as long as possible for all epochs will not necessarily remedy this, but it will reduce the risk of undertraining both models. A model summary can be seen in Appendix C Figure 20.

Confusion charts for the training and testing sets on the updated "From Scratch" model can be seen in Figure 14. Weighted F1 scores of ~0.636 and ~0.157 indicate the model has adequately trained but is overfitting, as seen with the significantly lower testing set score. Because the "Fine Tuned" model does not display any relevant change in performance from Figure 12 -as expected- the

logical explanation is that the “From Scratch” model’s ability to generalise to unseen data has still not been improved to any meaningful degree from the applied changes.

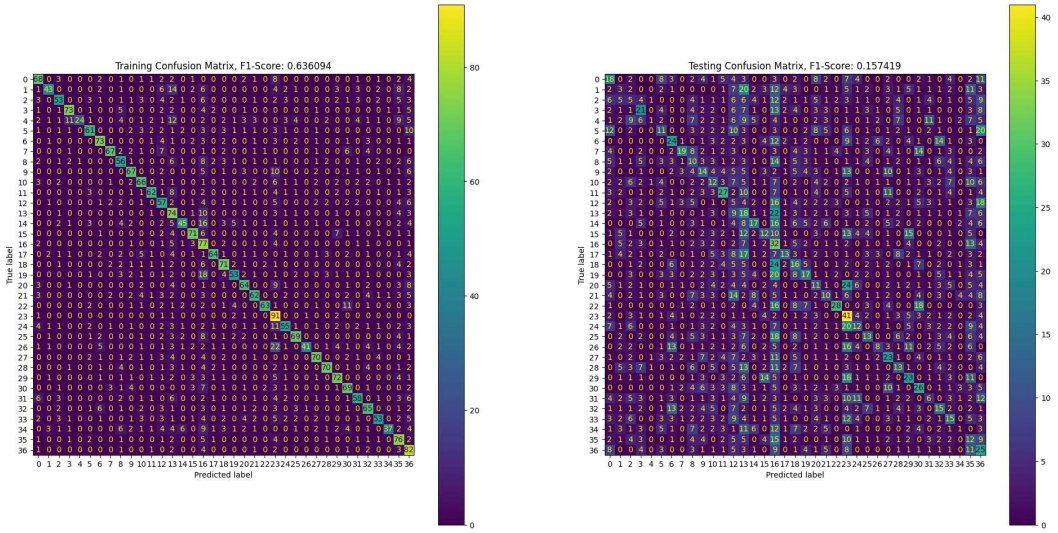

Figure 14: Confusion charts for training (left) and testing (right) sets on the updated “From Scratch” model. Overfitting can be observed within the testing set.

The results of the updated segmentation head in Figure 15 reveal disparate results and varied performance. In the “From Scratch” model, predictions appear disordered and significantly impacted by background details. While the contour of the animal in each image remains discernible, suggesting that the Conv2DTranspose layers are effectively capturing these features, they struggle to accurately differentiate between foreground and background elements. Conversely, the “Fine Tuned” model exhibits noticeable improvement, with the same issue appearing less pronounced and arguably looking better compared to Figure 13. They still appear pixelated but are more coherent.

Figure 15: Comparison of original images (left), true segmentation masks (middle) and predicted results from the "Fine Tuned" model (right) for both "From Scratch" (left) and "Fine Tuned" (right) models utilising the updated segmentation head.

Overall, lackluster performance from the "From Scratch" model can be credited to design and the complexity of the dataset used. MobileNetV3Small is a relatively lightweight CNN architecture. However, despite its compact size, it still contains many parameters and has a relatively deep architecture, especially compared to the "From Scratch" model. Its larger capacity allows it to capture more complex patterns and relationships in the data, and training such a model from scratch would require a larger and more varied dataset in addition to more computational resources than are available.

Appendix A

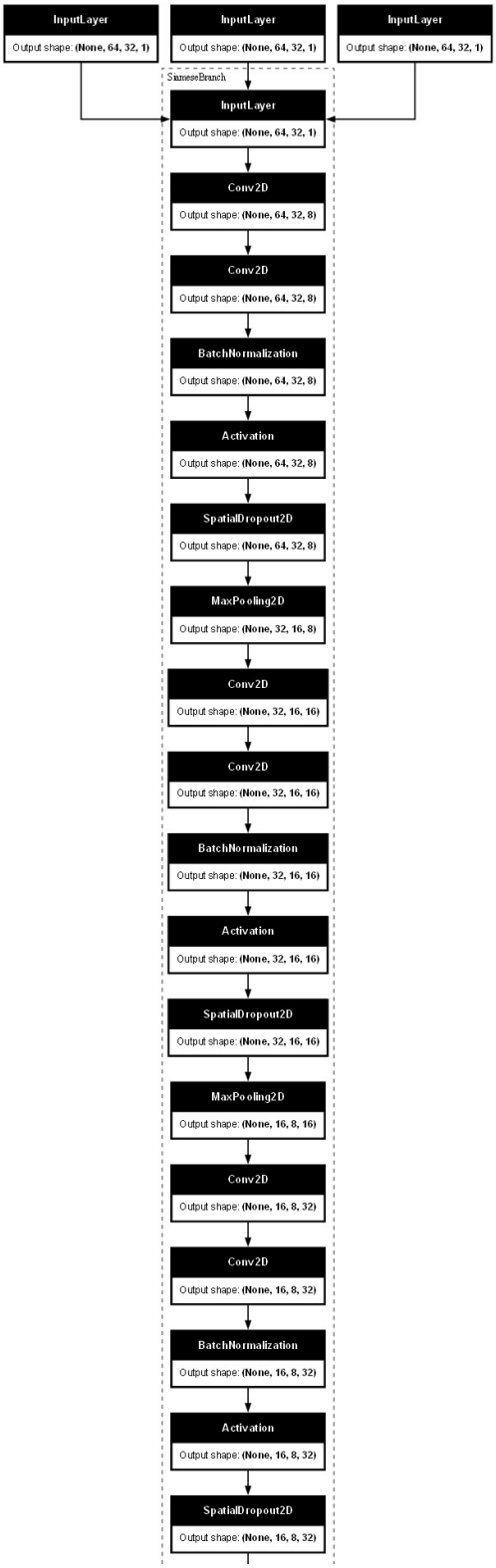

Figure 16: Keras Triplet Network output showing expanded hidden layers. Partitioned due to size and continued on next page.

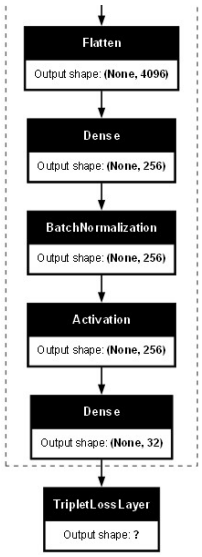

Figure 17: Continued Keras Triplet Network w/ expanded nest.

Appendix B

```
def classification_head(backbone_output, num_classes):
    x = Flatten()(backbone_output)
    x = Dense(64, activation='relu')(x)
    class_output = Dense(num_classes, activation='softmax',
name='class_output')(x)
    return class_output
```

Code Snippet 1: classification head

```
def segmentation_head(backbone_output):
    x = Conv2D(128, (3, 3), activation='relu',
padding='same')(backbone_output)
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    x = UpSampling2D((2, 2))(x)
    segmentation_output = Conv2D(1, (1, 1), activation='sigmoid',
name='segmentation_output')(x)
    return segmentation_output
```

Code Snippet 2: segmentation head

```
def backbone(input_shape):
    inputs = Input(shape=input_shape)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
    x = MaxPooling2D((2, 2))(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2, 2))(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2, 2))(x)
    return Model(inputs, x)
```

Code Snippet 3: “From-Scratch” backbone head

```
def my_awesome_model(input_shape, num_classes):
    # Define input
    inputs = Input(shape=input_shape)
    # Backbone
    backbone_output = mobile_base(input_shape)(inputs)
    # Classification head
    class_output = classification_head(backbone_output, num_classes)
    # Segmentation head
    segmentation_output = segmentation_head(backbone_output)
    # Combine classification and segmentation heads
    return Model(inputs=inputs, outputs=[class_output, segmentation_output])
```

Code Snippet 4: Overall model design (from “Fine Tuned” model)

Appendix C

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 128, 128, 3)	0	-
functional_1 (Functional)	(None, 16, 16, 128)	93,248	input_layer_1[0][0]
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147,584	functional_1[0][0]
up_sampling2d (UpSampling2D)	(None, 32, 32, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 32, 32, 64)	73,792	up_sampling2d[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 64)	0	conv2d_4[0][0]
flatten (Flatten)	(None, 32768)	0	functional_1[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 32)	18,464	up_sampling2d_1[0][…
dense (Dense)	(None, 64)	2,097,…	flatten[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 128, 128, 32)	0	conv2d_5[0][0]
class_output (Dense)	(None, 37)	2,405	dense[0][0]
segmentation_output (Conv2D)	(None, 128, 128, 1)	33	up_sampling2d_2[0][…

Figure 18: “From-Scratch” network summary

Layer (type)	Output Shape	Param #	Connected to
input_layer_16 (InputLayer)	(None, 128, 128, 3)	0	-
MobilenetV3small (Functional)	(None, 4, 4, 576)	939,120	input_layer_16[0][0]
conv2d_24 (Conv2D)	(None, 4, 4, 128)	663,680	MobilenetV3small[0]…
up_sampling2d_15 (UpSampling2D)	(None, 8, 8, 128)	0	conv2d_24[0][0]
conv2d_25 (Conv2D)	(None, 8, 8, 64)	73,792	up_sampling2d_15[0]…
up_sampling2d_16 (UpSampling2D)	(None, 32, 32, 64)	0	conv2d_25[0][0]
flatten_5 (Flatten)	(None, 9216)	0	MobilenetV3small[0]…
conv2d_26 (Conv2D)	(None, 32, 32, 32)	18,464	up_sampling2d_16[0]…
dense_5 (Dense)	(None, 64)	589,888	flatten_5[0][0]
up_sampling2d_17 (UpSampling2D)	(None, 128, 128, 32)	0	conv2d_26[0][0]
class_output (Dense)	(None, 37)	2,405	dense_5[0][0]
segmentation_output (Conv2D)	(None, 128, 128, 1)	33	up_sampling2d_17[0]…

Figure 19: “Fine Tuned” Network Summary

Layer (type)	Output Shape	Param #	Connected to
input_layer_42 (InputLayer)	(None, 128, 128, 3)	0	-
functional_57 (Functional)	(None, 16, 16, 128)	287,008	input_layer_42[0][0]
conv2d_transpose_45 (Conv2DTranspose)	(None, 32, 32, 128)	147,584	functional_57[0][0]
flatten_16 (Flatten)	(None, 32768)	0	functional_57[0][0]
conv2d_transpose_46 (Conv2DTranspose)	(None, 64, 64, 64)	73,792	conv2d_transpose_45…
dense_22 (Dense)	(None, 64)	2,097,…	flatten_16[0][0]
conv2d_transpose_47 (Conv2DTranspose)	(None, 128, 128, 32)	18,464	conv2d_transpose_46…
class_output (Dense)	(None, 37)	2,405	dense_22[0][0]
segmentation_output (Conv2D)	(None, 128, 128, 1)	33	conv2d_transpose_47…

Figure 20: Updated “From Scratch” Network Summary

References

Madhumita, M. (2019, September 19). *Who's using your face? The ugly truth about facial recognition*. Retrieved from Financial Times: https://www.ft.com/content/cf19b956-60a2-11e9-b285-3acd5d43599e

Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2021). *Cats and Dogs*. University of Oxford, Department of Engineering Science. Oxford: University of Oxford. Retrieved from https://www.robots.ox.ac.uk/~vgg/publications/2012/parkhi12a/parkhi12a.pdf

Shen, C. F. (2019). *Social credit system in China*. Hong Kong: University of Hong Kong.