Assignment 1B –

Person Re-Identification & Multi-Task Learning

CAB420 – Machine Learning

Gregory Mandall - n10757163

Submitted: May 2024

* Code adapted from CAB420 lectures and practical solutions

PERSON RE-IDENTIFICATION

1. Preprocessing

The Market-1501 dataset contains several issues including viewpoint variations, class imbalance, background clutter and obstructions, and potential lighting issues. This may lead to performance fluctuations due to variations in appearance and pose, disparate frequency of images per identity and the presence of complex backgrounds, making accurate identification increasingly challenging. Image complications are documented and explored further in the analysis section of this report. Figure 1 depicts the distribution of images across each class present in the dataset, visualising the class imbalance present in the dataset.

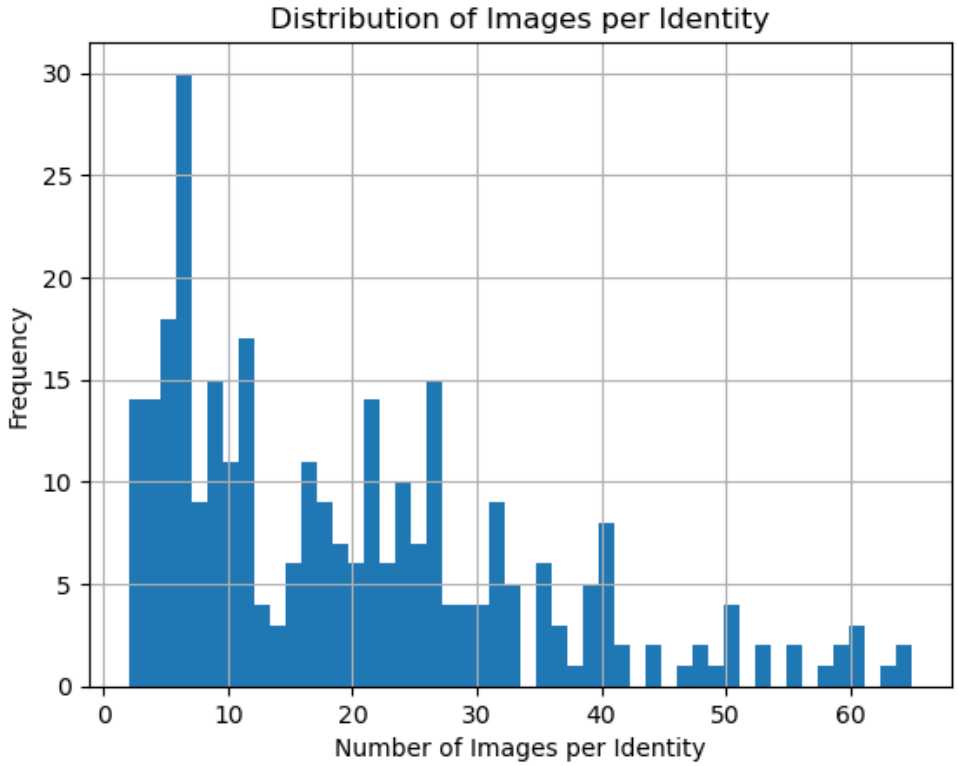

Figure 1: Distribution of Images per Identity to Visualise Class Imbalance

Data augmentation was implemented to address these issues and enhance the performance of the deep-learning and non-deep-learning methods. The data augmentation techniques applied included rotation up to 10 degrees, width and height shifts up to 10%, zoom up to 10%, horizontal flipping, and brightness modifications varying from 80% to 120% of the original brightness level. Implementing these methods introduced realistic variations that may potentially occur in real-world scenarios. Augmentation artificially expanded and increased the diversity of the training dataset without the need for additional data, attempting to promote increased generalisation and performance. Data augmentation examples can be seen in Appendix A, Figure 1.

Images were resized to 64x32 to increase computational efficiency while maintaining adequate performance levels. Images were kept in colour to retain critical distinguishing features crucial for recognising individuals based on clothing or accessory colours, which are significant identifiers in person re-identification tasks.

2. Model Details

Principal Component Analysis (PCA) was selected over Linear Discriminant Analysis (LDA) for the non-deep-learning method due to the dataset's structure. Several instances have a limited number of images representing the identity. This may limit the effectiveness of LDA, as LDA requires multiple samples per class to compute between-class and within-class variance effectively. Therefore, PCA's approach of reducing dimensionality while retaining a significant portion of variance was deemed suitable for handling the variations in the Market-1501 dataset. A PCA subspace was generated using augmented, vectorised training data and projected onto the gallery and probe sets. The cumulative sum of explained variance was used to determine the optimal number of features to retain that explain 95% of the variance, leading to the retention of 107 components. This threshold was selected to maintain a balance between dimensionality reduction and information retention by disregarding less informative features. Visualising the cumulative variance, shown in Figure 2, validated that 107 components were sufficient to capture essential data characteristics, facilitating a comparable representation of the dataset without the computational overhead of handling higher-dimensional data.

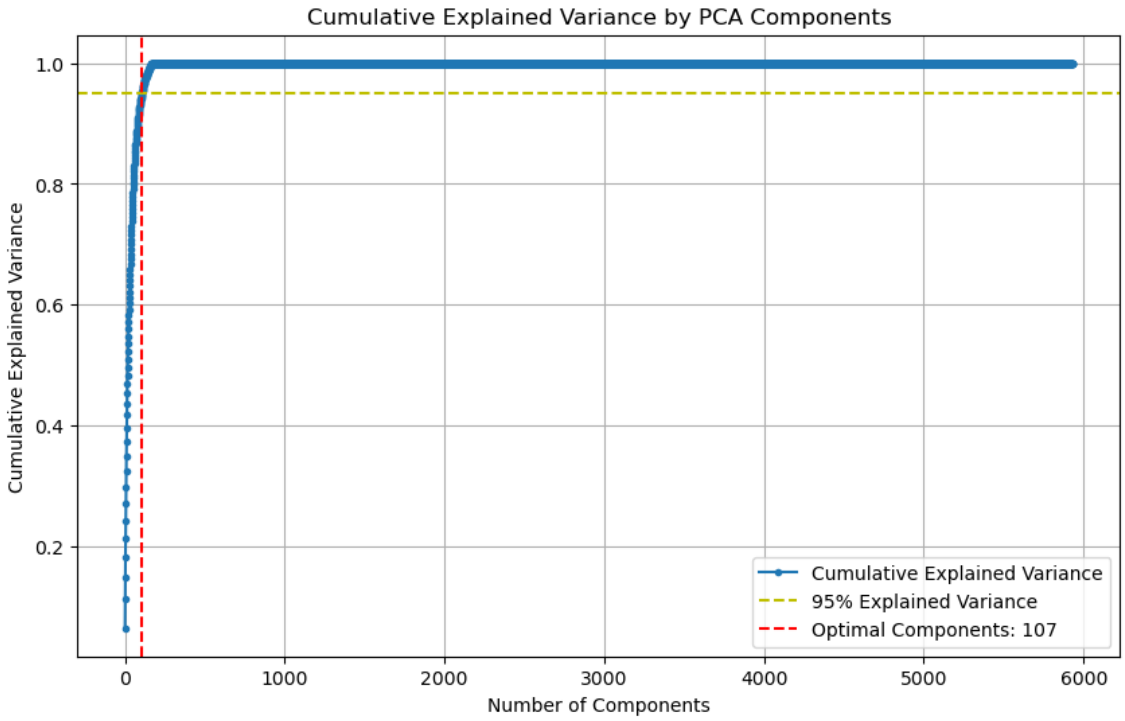

Figure 2: Cumulative Variance versus Number of Components to Visualise Optimal Feature Selection and Data Dimensionality.

The deep-learning method utilised a triplet loss network with a deep convolutional neural network (DCNN) backbone to optimise feature learning. This approach employed the strengths of DCNNs, which excel in processing pixel data from images. Triplet loss was implemented due to its increasing effectiveness in learning fine-grained differences between complex classes relative to contrastive loss. The neural network utilises a tailored VGG-like architecture. This design was selected as dense layers lead to slightly superior performance and enhanced computational efficiency compared to a ResNet in scenarios with lower filter counts involved in model development. The network consists of three main convolutional blocks. The first block includes two Conv2D layers with 16 filters of 7x7 kernel size. The larger kernel size in the initial block was chosen to capture broad image features early in the network. Each convolutional layer is followed by batch normalisation to reduce computation time. The block ends with a MaxPooling2D layer to reduce feature map dimensionality. The size of the filters decreases to 5x5 and then to 3x3 while increasing the number of filters to 32 and then to 64 as the network progresses. This allows the network to capture increasingly detailed features necessary for person re-identification. The convolution blocks are flattened, and the vector is passed to a dense layer of 300 units.

Embeddings are normalised to ensure that the magnitude of each embedding vector is consistent, facilitating stable distance calculations between vectors. Normalisation is critical for the triplet loss function to effectively measure the relative distances necessary for distinguishing between disparate identities in the Market-1501 dataset. Swish and ReLU activation functions were compared during experimentation. Ultimately, Swish activation was selected for the final network due to its superior performance metrics. A complete model breakdown is shown in Table 1, Appendix A. The triplet loss margin was set to 1.0 to define the distance to distinguish between dissimilar items in the feature space. A batch size of 350 was selected to ensure diverse class representation within each batch, optimising learning and enhancing the accuracy of gradient estimates during updates. The training extends over 40 epochs using Adam to balance between adequate convergence of the loss function and training time. Figure 3 showcases the training and validation loss curves, providing insight into the model's learning dynamics. The plot indicates that stability and convergence is achieved at approximately 35 epochs.

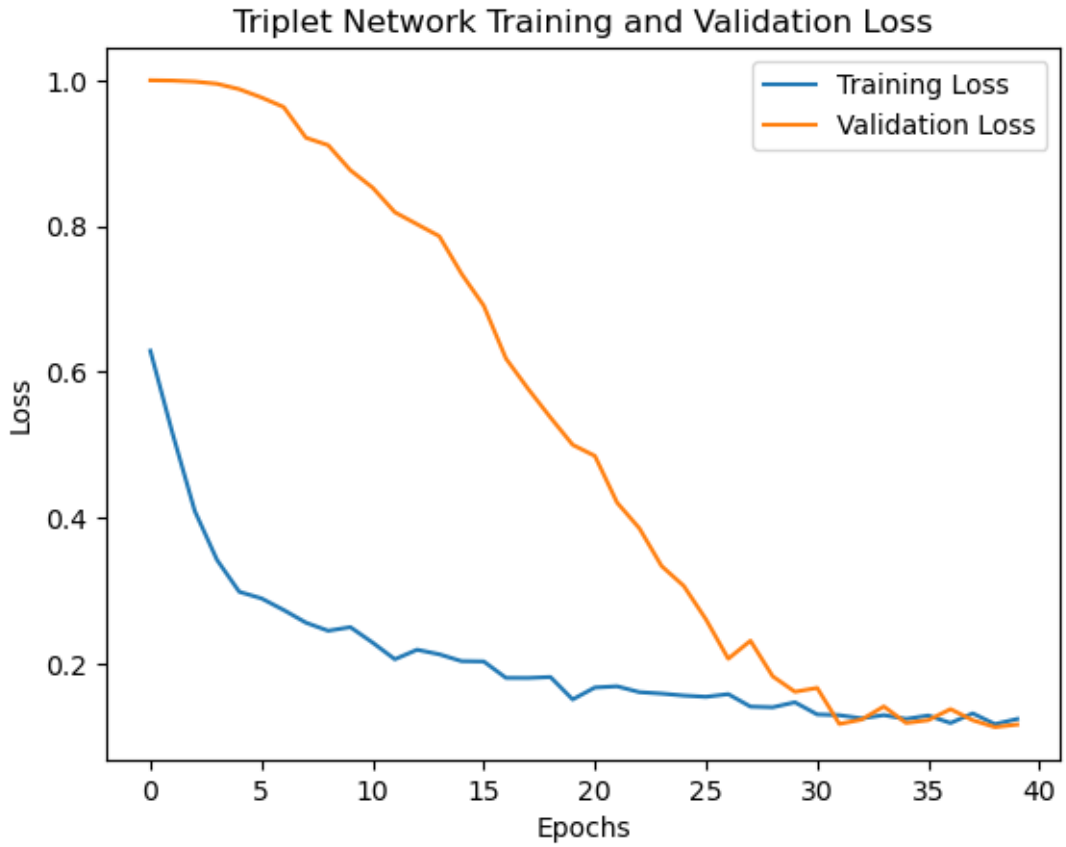

Figure 3: Loss versus Epochs to Determine Successful Convergence of the Triplet Network.

3. Evaluation and Analysis

The comparison between the PCA and Triplet Loss methods reveals significant disparities in performance across various scenarios. The Triplet Loss model demonstrates superior performance across all accuracy measures. This exhibits a clear advantage for the deep learning approach, potentially due to the method's ability to handle increasingly complex image variations. The Top-1, Top-5, and Top-10 accuracies are shown in Table 1 below.

Table 1: Top-1, Top-5, and Top-10 Accuracy for the PCA and Triplet Loss Method

Method	**Top-1 Accuracy**	**Top-5 Accuracy**	**Top-10 Accuracy**
PCA	0.14	0.23	0.35
Triplet Loss	0.25	0.48	0.62

Instances illustrating the superior performance of the deep-learning method can be observed in a significant quantity of images present in the gallery and probe datasets.

Images such as ID #1423 demonstrate the model's proficiency in adjusting to lighting changes that alter clothing hues. Image ID #1259 showcases the model's ability to maintain high accuracy measures despite significant distortions between the gallery and probe images. Figure 4 illustrates the robustness of the deep learning model, in which the graph delineates a consistent trend where deep learning outperforms PCA. The plot displays higher accuracy rates for individual IDs across a range of scenarios. Specific examples of instances where the triplet network outperforms PCA are shown in Figure 5. This suggests that the triplet loss network's superiority is particularly evident in scenarios involving increasingly convoluted images, facial or image distortions, and minor lighting variations that seem to affect clothing colours. However, PCA outperforms the triplet network across several instances where images from the probe and gallery sets exhibit uniformity in lighting, minimal distortion, and maintain consistent image quality. This showcases the model's simplicity and efficiency in processing less complex and highly uniform scenarios. Figure 6 depicts instances where PCA demonstrates superior performance under these conditions.

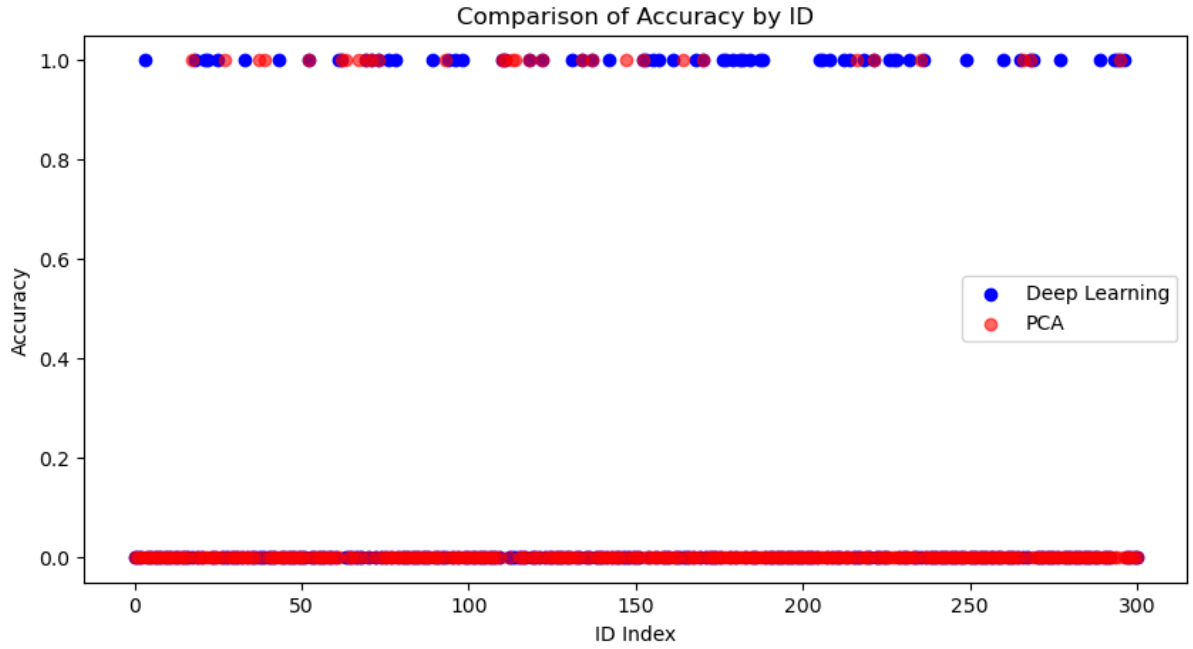

Figure 4: Comparison of Accuracy by ID to Visualise the Performance of Deep-Learning and Non-Deep-Learning Methods.

Figure 5: Instances where the Triplet Loss Network Significantly Outperforms PCA

Figure 6: Instances where PCA Outperforms the Triplet Loss Network.

The CMC curves for the Triplet Loss network and Principal Component Analysis reflect the disparities observed in the accuracy metrics and provide further insight into the comparative performance of the two methods. The curve for the Triplet Loss method exhibits a steeper rise, showcasing this method's ability to differentiate between similar and dissimilar samples with increasing accuracy relative to PCA, a critical capability for pattern recognition tasks. Additionally, the shallower slope of the PCA's curve may suggest that this method faces challenges in distinguishing between closely related classes. However, the plots suggest that each method converges to near 100% accuracy at approximately the same point. This convergence underscores the importance of both methodologies, highlighting that each may be strategically employed depending on specific use case requirements. Figure 7 visualises the CMC curves for the deep-learning and non-deep-learning methods.

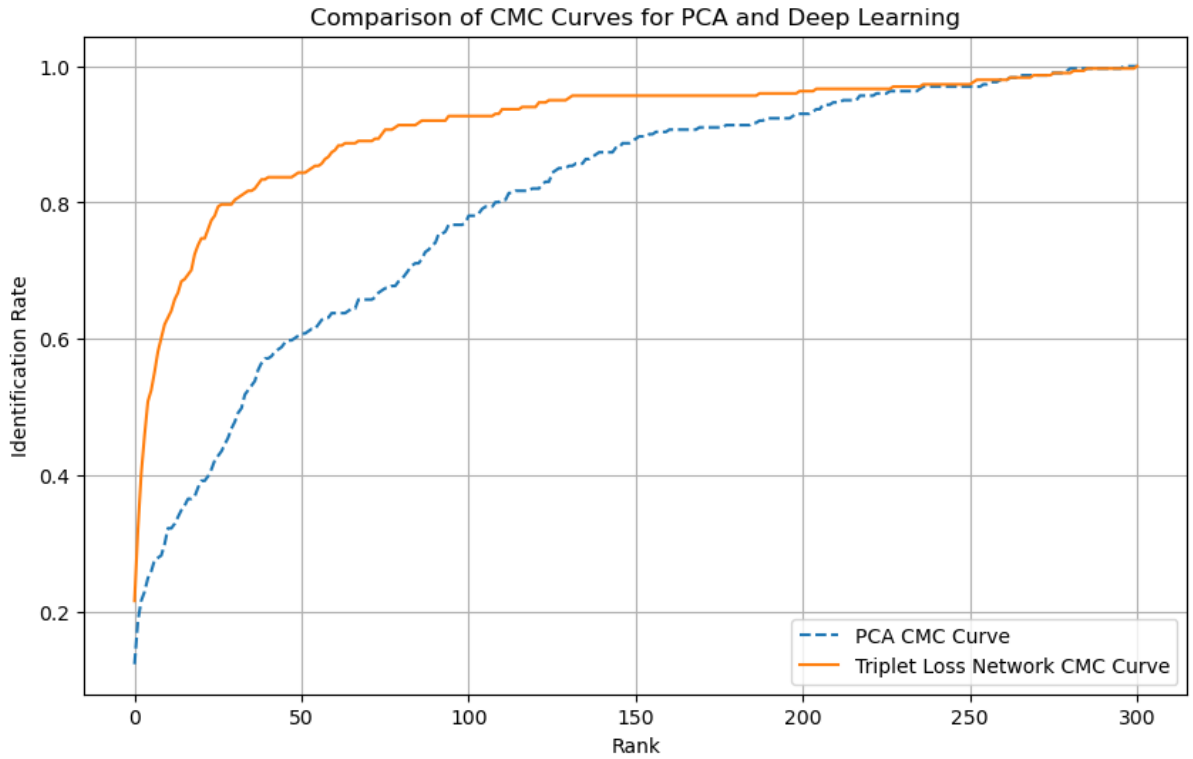

Figure 7: CMC Curve for PCA and the Triplet Loss Network to Provide a Graphical Representation of Each Method's Performance.

PCA demonstrates significantly lower training and inference times relative to the Triplet Loss method. This demonstrates the computational efficiency of the method, making it increasingly suitable for use case scenarios where computational resources are limited, despite its lower overall accuracy. However, person re-identification tasks require high accuracy. This necessitates the consideration of deep learning methods despite longer training and inference times, due to their ability to handle complex image data and achieve higher accuracy compared to traditional machine learning methods. Table 2 displays the training and inference times for the Triplet Loss Network and Principal Component Analysis.

Table 2: Training Time and Inference Time for PCA and Triple Loss Methods.

Method	Training Time, seconds (s)	Inference Time, seconds (s)
PCA	166.27	0.01
Triplet Loss	627.33	5.03

4. Ethical Considerations

Research into person re-identification is crucial in advancing facial recognition technology, which companies like Microsoft and Apple utilise for device security. However, ethical considerations associated with person re-identification must be considered due to factors such as privacy concerns, potential misuse, model limitations and applications, and inadequate data handling practices.

The creation and distribution of datasets such as 'Duke MTMC' exemplify significant ethical breaches in data collection, prioritising surveillance technology at the expense of civil, human, and privacy rights. To create the Duke MTMC dataset, images of citizens were compiled without the explicit consent of the individuals involved, resulting in inadvertently enrolling individuals into a global surveillance network used by foreign defence entities [1]. These individuals are now permanently part of a data pool that supports the expansion of biometric surveillance by governments and corporations. This unauthorised collection and use of personal data highlight the ethical concerns associated with person re-identification tasks due to the severe implications that may occur when legal or military entities access this data, potentially significantly impacting individuals' freedom and rights [2].

Deploying person re-identification technologies such as device authentication and healthcare management systems can enhance digital security and personal well-being, providing robust security measures for devices and ensuring effective patient monitoring in medical settings [3]. However, utilising this technology without consent in various applications such as retail to monitor customer behaviour, transportation hubs to track passenger flow, financial services to verify identity during ATM or branch transactions, and their application in public surveillance may erode public anonymity and privacy. Additionally, surveillance technologies utilised by law enforcement or government agencies without stringent oversight may lead to compromised civil liberties due to model limitations [4]. Model limitations, such as inaccuracies in facial recognition, can result in misidentification with severe repercussions. This may disproportionately affect certain demographic groups,

amplifying issues such as discrimination due to model bias [5]. Addressing these limitations is crucial to developing fair, unbiased, and reliable technologies that enhance quality of life and improve universal security measures.

REFERENCES

1. Mozur, Paul. (2019). "*One Month, 500,000 Face Scans: How China Is Using A.I. to Profile a Minority*". https://www.nytimes.com/2019/04/14/technology/china-surveillance-artificial-intelligence-racial-profiling.html.

2. Harvey, Adam. Laplace, Jules. (2021). "*Duke Multi-Target Multi-Camera Tracking Dataset*". https://exposing.ai.

3. Suleski T, Ahmed M, Yang W, Wang E. (2023). A review of multi-factor authentication in the Internet of Healthcare Things. *Digit Health*. doi: 10.1177/20552076231177144.

4. Slobogin C, Brayne S. (2022*).* "Surveillance Technologies and Constitutional Law". *Annu Rev Criminol*. 6:219-240. doi: 10.1146/annurev-criminol-030421-035102.

5. Almeida D, Shmarko K, Lomas E. (2022). The ethics of facial recognition technologies, surveillance, and accountability in an age of artificial intelligence: a comparative analysis of US, EU, and UK regulatory frameworks*. AI Ethics*. 2022;2(3):377-387. doi: 10.1007/s43681-021-00077-w.

MULTI-TASK LEARNING

1. Preprocessing

The images in the dataset demonstrate significant deformability due to the variation in pose, size, and fur pattern that characterises each breed. This variability can be pronounced within a single species, posing a significant challenge for automated recognition systems. To address these challenges, various preprocessing techniques were employed to accommodate the inherent challenges presented by the dataset. Data augmentation was utilised to simulate the diverse conditions and poses in which cats and dogs may be photographed. This included width and height shifts by up to 10%, zoom up to 10%, horizontal flipping to account for the asymmetrical poses pets may exhibit, and brightness modifications ranging from 80% to 120% of the original levels to adjust to different lighting conditions encountered in typical environments. These augmentation techniques were chosen to expand the diversity of the dataset, ultimately improving generalisation and enhancing model performance. Figure 8 visualises an example of the augmented images input into the network.

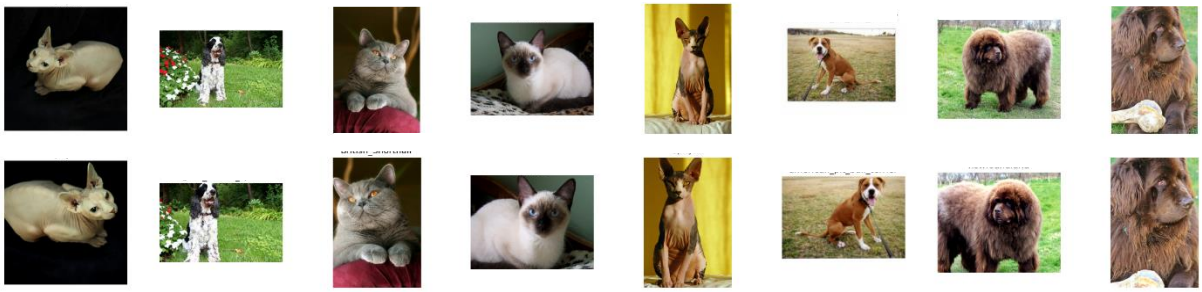

Figure 8: Comparison of Original and Augmented Images: The top row displays eight random original images, while the bottom row shows the same images after applying various data augmentation techniques including shifts, zoom, horizontal flipping, and brightness modifications.

Colour information was preserved as specific breeds may display unique colour patterns and markings crucial in identifying particular breeds. This ensures the models can utilise colour as a distinguishing feature. All images were resized to 128x128, facilitating computational efficiency while attempting to maintaining sufficient detail for effective model training.

2. Methods

The custom-designed multi-task DCNN utilises architecture optimised for the dual output of classification and segmentation. The model's backbone comprises of 3 convolutional layers paired with max pooling to encode the input image effectively, a design choice influenced by conventional VGG-like networks which are known for their simplicity and effectiveness in similar image-based tasks. The initial convolutional block consists of a Conv2D layer with 32 filters of size (3x3) and 'relu' activation applied with 'same' padding to maintain the spatial dimensions. This is proceeded by a MaxPooling2D layer with a pool size of (2,2), reducing the spatial dimensions by half. This block is crucial for capturing the initial features from the raw input images. The size of the filters increases to 64 and then to 128 as the network progresses, leading to the bottleneck of the model. The network then splits into two distinct heads. A classification head flattens the output from the backbone and passes it through a dense layer of 37 units equipped with a softmax activation function, corresponding to the 37 pet breeds. A segmentation head utilises a series of Conv2DTranspose layers to upscale the feature maps back to the original dimensions of the input image. Starting with 128 filters and sequentially decreasing to 64 and then 32 filters, these layers reconstruct the spatial dimensions necessary for detailed segmentation. The segmentation head ends with a final Conv2D layer with one filter and 'sigmoid' activation, producing the binary segmentation mask. A detailed overview of the Custom Model can be seen in Table 1, Appendix B.

The network uses the Adam optimiser with a learning rate of 0.001, optimising two loss functions concurrently: 'sparse_categorical_crossentropy' for classification and 'binary_crossentropy' for segmentation. Training was conducted over a maximum of 40 epochs, with a batch size of 64. An early stopping mechanism and custom callback were implemented to stop training after 15 minutes or on model convergence. This ensured that training remained computationally feasible and prevented excessive overfitting. Patience was set to 3 epochs to allow for minor fluctuations in overall loss improvement. Convergence was attained after 10 epochs. The model's training and validation loss were plotted to visualise convergence trends, providing visual insight into the model’s learning dynamics. Figure 9 depicts the loss trends of the custom model.

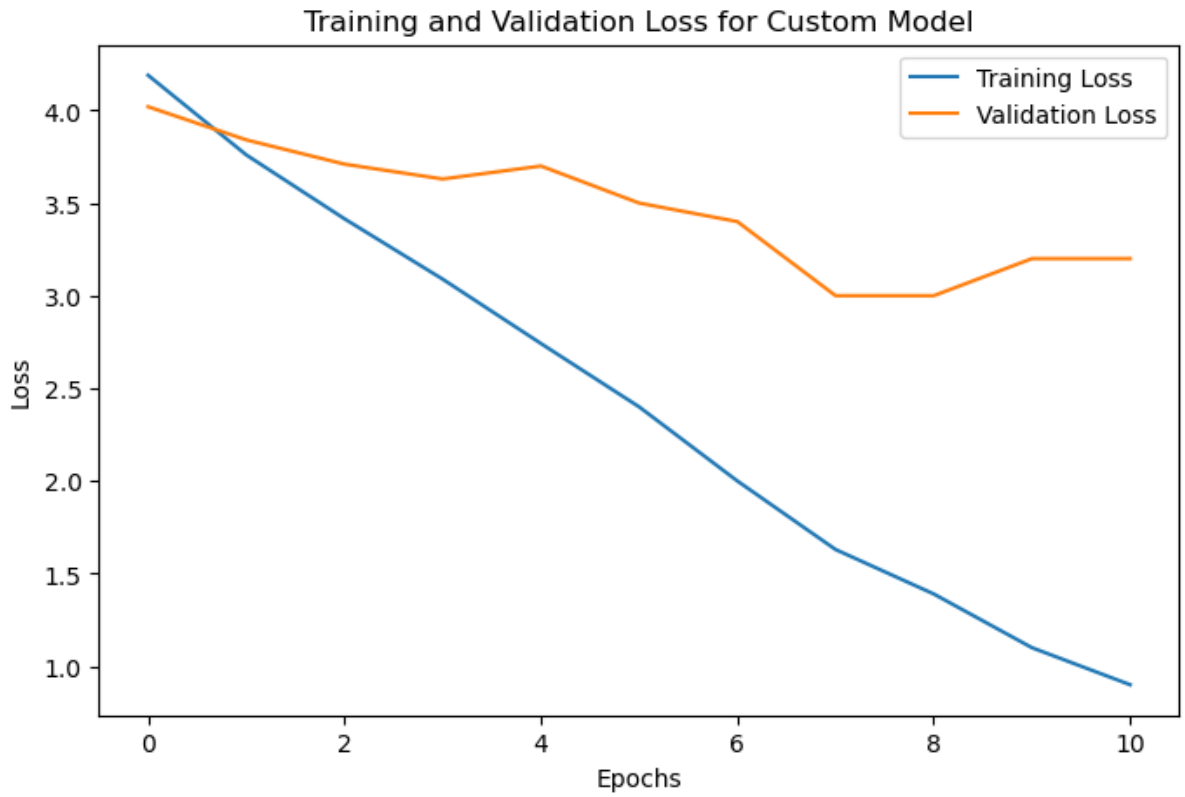

Figure 9: Loss versus Epochs to Determine Successful Convergence of the Custom Model.

The fine-tuned model implements the MobileNetV3Small architecture as its backbone. However, the top classification layer is removed. The new classification head replicates the head utilised in the custom-built model. The segmentation decoder builds upon the extracted features using a series of Conv2DTranspose layers. These layers incrementally increase the spatial resolution through filters of reducing sizes of 128, 64, 32, 16, and 8. The output is a binary segmentation mask generated by a final Conv2D layer with a 'sigmoid' activation, delineating the foreground from the background. A detailed overview of the fine-tuned model can be seen in Table 2, Appendix B.

All model layers were trained simultaneously by implementing an identical approach to the custom-built model. Convergence was attained after 10 epochs. The development of each model adheres to established design principles covered in the lecture material and implements known methods and architectures suited for similar image-processing tasks. This approach ensures the models are suitable for the task and remain computationally viable. Figure 10 depicts the loss trends of the fine-tuned model.

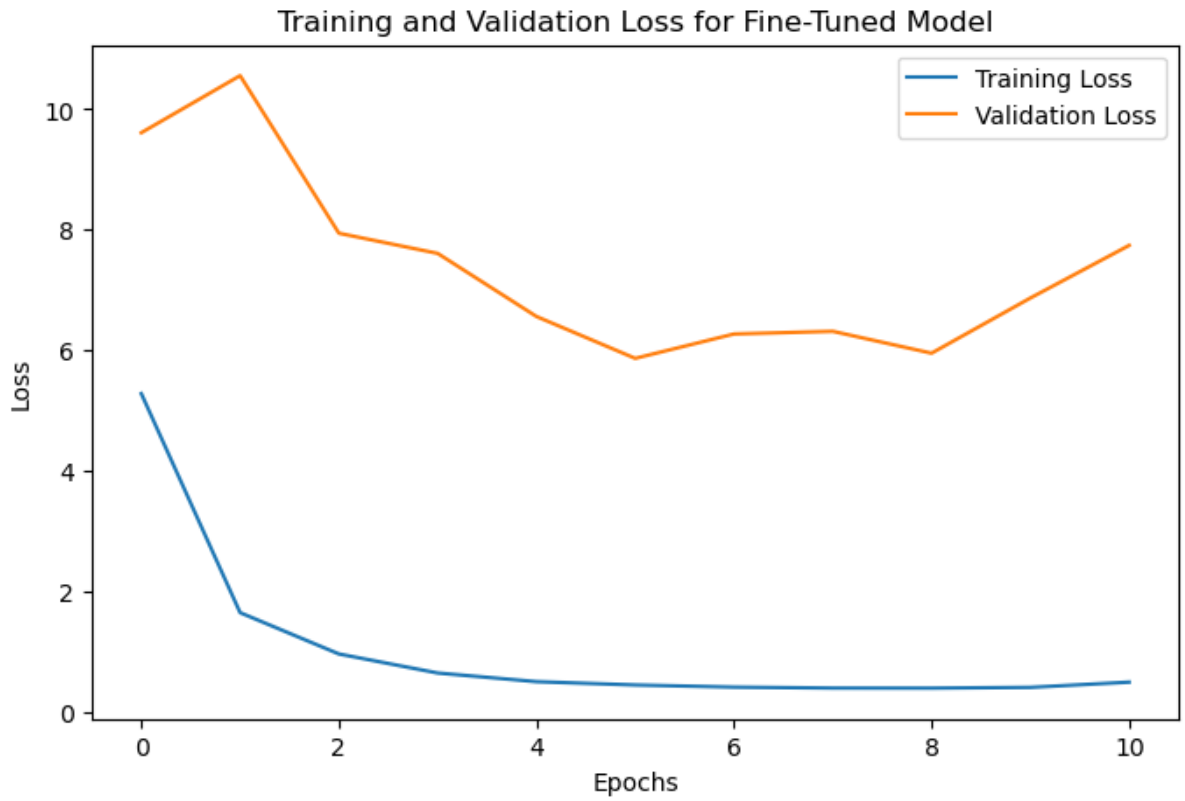

Figure 10: Loss versus Epochs to Determine Successful Convergence of the Fine-Tuned Model.

3. Evaluation

The custom model demonstrates poor performance in the classification task. However, the model shows significant improvements in the segmentation task relative to its classification scores, shown in Figure 11. The fine-tuned MobileNet performs better than the custom model in both tasks, achieving an F1 score of 0.37 in classification and 0.84 in segmentation on the test data sets, as shown in Figure 12. The discrepancies observed in performance may be attributed to architectural advantages such as depthwise separable convolutions and the initial weights provided by the MobileNet backbone. This allows the model to utilise pre-learned features learned from a diverse image dataset, inherently optimising parameter efficiency and increasing performance. The custom model fails to achieve enhanced performance levels without the depth and optimisations present in the fine-tuned model. Its simpler architecture may not capture complex patterns as effectively, resulting in lower overall performance.

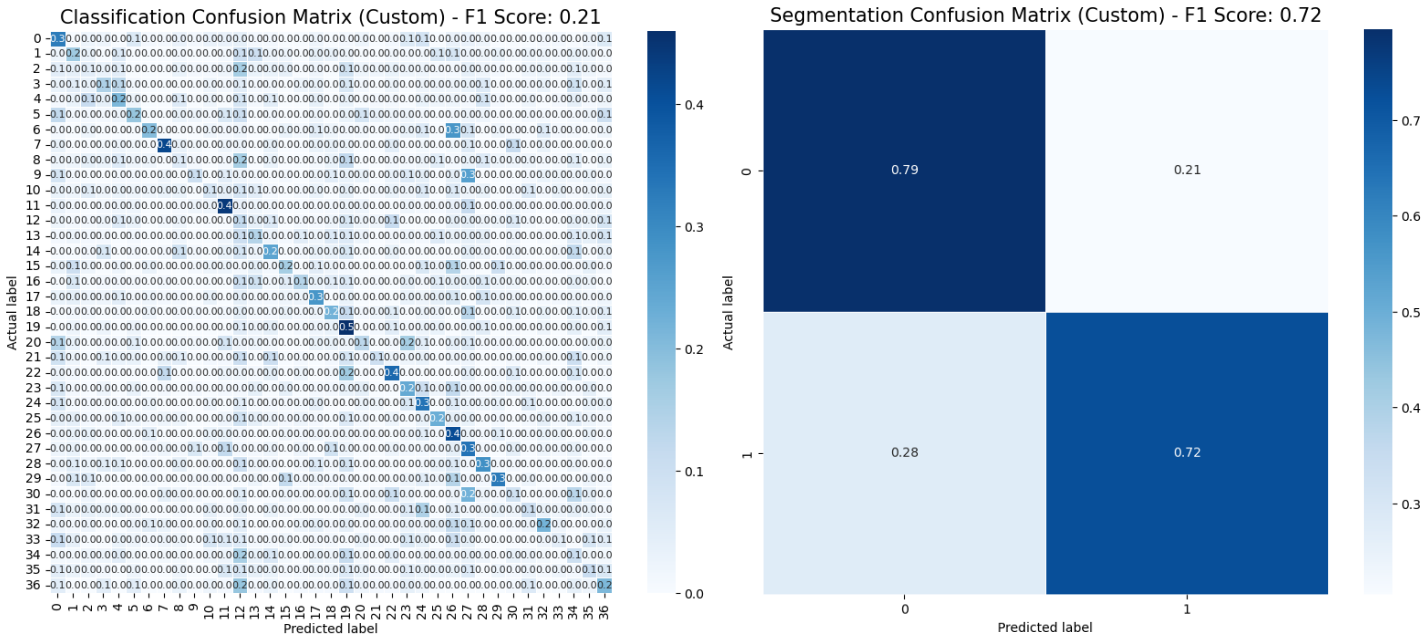

Figure 11: Classification and Segmentation Performance Analysis of Custom Model.

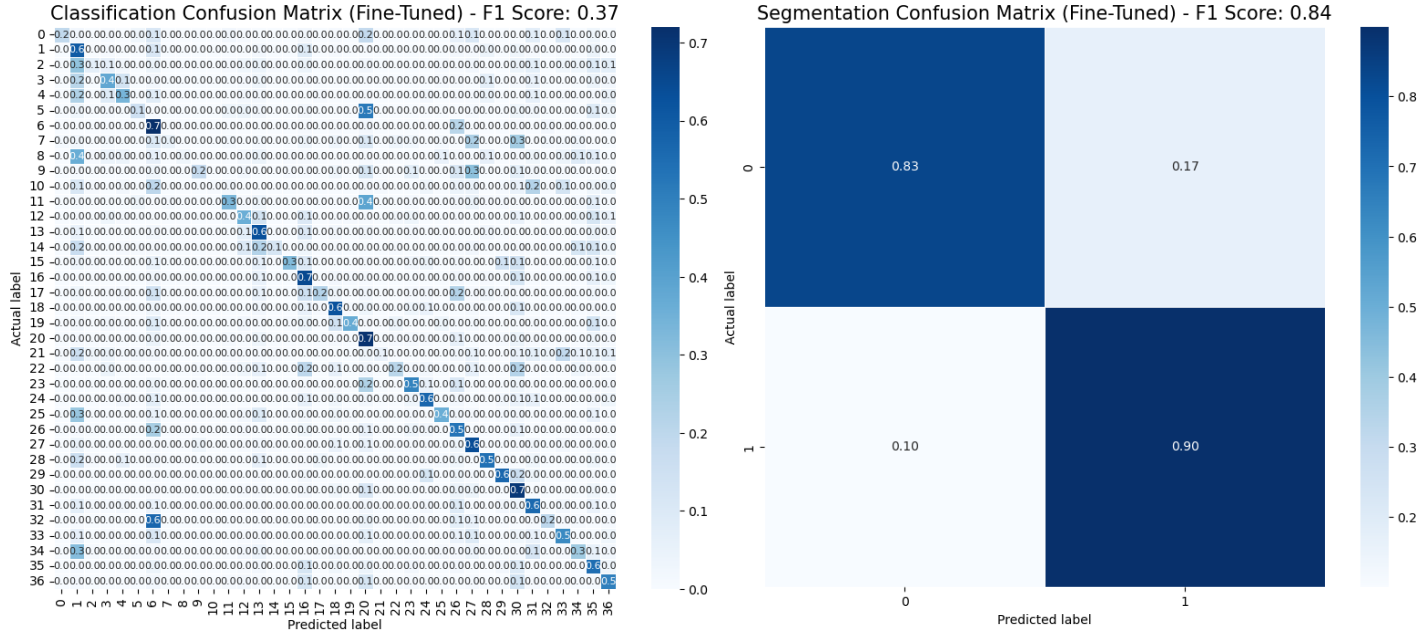

Figure 12: Classification and Segmentation Performance Analysis of Custom Model.

Both models achieve relatively low performance on the classification task. Analysis of the classification report reveals that the custom model and the fine-tuned network show similar difficulties in accurately classifying specific breeds. This is depicted by poor performance metrics on Class 2 and Class 12, where both models exhibit extremely low F1 scores. The consistency across models suggests inherent challenges with these classes due to the fine-grained nature of breed distinctions which require increasingly sophisticated feature learning to accurately classify. These challenges may stem from high intra-class variability which seems to be pronounced in these classes, shown in Figure 13. The classification reports of the custom and fine-tuned models are displayed in Table 3 and Table 4 of Appendix B.

Figure 13: Intra-class Variability Observed in Class 2 and Class 12 of the Oxford-IIIT Pet Dataset.

Each model performs better in segmentation than in classification. This may be due to the presence of semantic segmentation information in the dataset and the binary nature of segmentation, which is inherently less complex than the multi-class problem of breed classification. However, discrepancies can be observed between the actual and predicted masks for both the custom and fine-tuned models. The custom model shows significant data loss, with some regions of the predicted masks appearing as false negatives where there should be true positives. This performance decrease with the custom model may be due to its simpler architecture or the implementation of sub-optimal hyperparameter tuning due to computational constraints. This may lead the model to fail to capture increasingly complex patterns necessary for accurate segmentation and classification. Figure 14 visualises segmentation results for the custom and fine-tuned models.

Figure 14: Comparison of Segmentation Results: Fine-Tuned Model (Left) vs. Custom Model (Right). Columns from left to right: Original Image, True Mask, Predicted Mask.

The inference times reflect the efficiency of the models, with the fine-tuned MobileNet slightly outperforming the custom model despite its architectural complexity. This efficiency may be attributed to increasingly optimised initial weights, which reduce the need for extensive forward pass computations. Inference times are documented in Table 3, below.

Table 3: Inference Time for the Custom-Built and Fine-Tuned Models.

Model	**Inference Time, Seconds (s)**
Custom	12.15
Fine-Tuned	13.53

4. Limitations and Mitigation Strategies

A primary limitation encountered in the development of the multi-task learning models included high intra-class variability with various breeds. This variability seems to have led to a significant number of misclassifications, potentially due to an underrepresentation of each distinct-looking breed within several breed types. To mitigate this issue, various data augmentation techniques were employed in an iterative development cycle to track the improvements made with each implemented transformation. This approach improved the performance of each model while ensuring training times remained viable. F1 scores before and after applying augmentation are documented in Table 2 below. Increasingly advanced augmentation techniques such as elastic transformations were explored. However, due to the computational constraints of the development environment, training times exceeded the 15-minute threshold before reaching convergence when this was applied on top of the already implemented shifts, zooms, and flips. This technique should be considered in future investigations that have access to greater computational resources to explore the impact elastic transformations have on model performance. Subsequent experimentation may consider implementing generative adversarial networks (GANs) to generate new training samples that may provide increasingly balanced and diverse datasets.

Table 2: Classification Scores for the Custom and Fine-Tuned Models Before and After Applying Various Data Augmentation Techniques.

Model	**Classification F1 score Before Augmentation**	**Classification F1 score After Augmentation**
Custom	0.15	0.21
Fine-Tuned	0.32	0.37

The models' complexity was constrained by the available computational resources. This limited the exploration of increasingly advanced architectures and hyper-parameter tuning that could potentially enhance performance metrics. To mitigate this limitation, images were resized to a computationally viable size, and a simpler network architecture was implemented to ensure that training times remained feasible. Image sizes were initially resized to 64x64, then to 128x128 and 224x224. Performance metrics improved as the image resolution increased, demonstrating the positive impact of higher pixel counts on model accuracy. However, training models with images larger than 128x128 pixels significantly increased training time, making it an impractical solution given the resource constraints. Additionally, as training times with augmented data increased, extensive hyper-

parameter tuning became infeasible within the given resource bounds, further limiting potential performance improvements. This was mitigated by implementing sensible hyper-parameter values to initialise the models. Early stopping mechanisms were implemented to minimise excessive training times and the computational resource constraints imposed by the resources available, ensuring efficient model training while preventing overfitting. Exploring and developing increasingly efficient and complex architectures should be considered in future investigations with access to more computational resources. This will allow for a deeper investigation into advanced augmentation techniques, architectural complexity, and increasingly comprehensive hyper-parameter tuning, potentially leading to significant improvements in model performance.

APPENDIX A

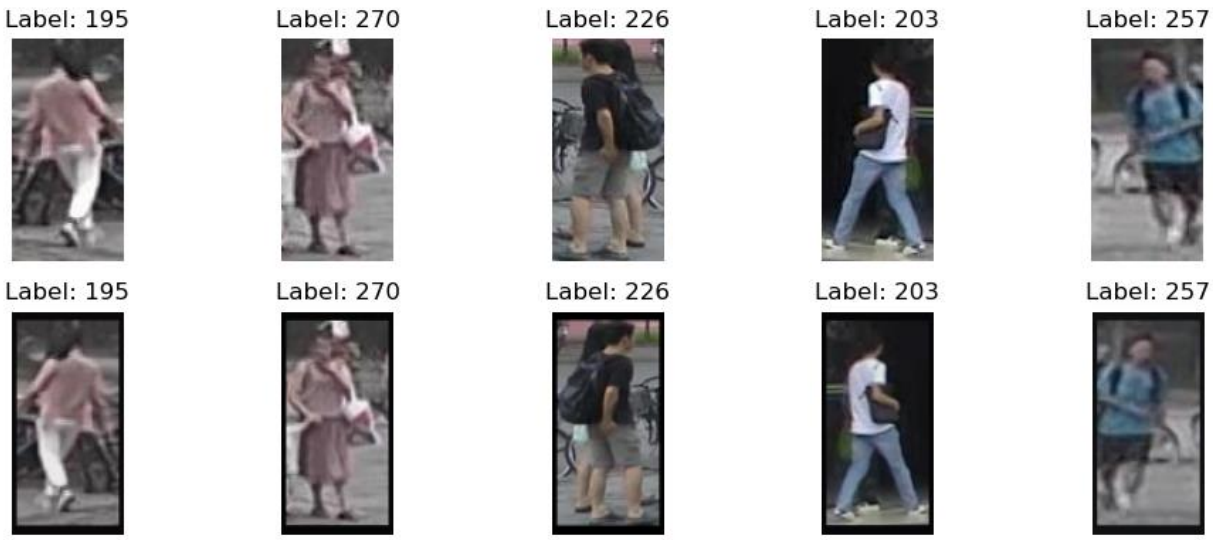

Figure 1: Comparison of Original and Augmented Images: The top row displays five random original images, while the bottom row shows the same images after applying various data augmentation techniques including shifts, zoom, horizontal flipping, and brightness modifications.

Table 1: Model Architecture and Detailed Parameter Values of the Network Developed for Person Re-Identification.

Layer (type)	Output Shape	Param #
img (InputLayer)	(None, 64, 32, 3)	0

conv2d (Conv2D)	(None, 64, 32, 16)	2,368
batch_normalization (BatchNormalization)	(None, 64, 32, 16)	64
conv2d_1 (Conv2D)	(None, 64, 32, 16)	12,560
batch_normalization_1 (BatchNormalization)	(None, 64, 32, 16)	64
max_pooling2d (MaxPooling2D)	(None, 32, 16, 16)	0
conv2d_2 (Conv2D)	(None, 32, 16, 32)	12,832
batch_normalization_2 (BatchNormalization)	(None, 32, 16, 32)	128
conv2d_3 (Conv2D)	(None, 32, 16, 32)	25,632
batch_normalization_3 (BatchNormalization)	(None, 32, 16, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 16, 8, 32)	0
conv2d_4 (Conv2D)	(None, 16, 8, 64)	18,496
batch_normalization_4 (BatchNormalization)	(None, 16, 8, 64)	256
conv2d_5 (Conv2D)	(None, 16, 8, 64)	36,928
batch_normalization_5 (BatchNormalization)	(None, 16, 8, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 5, 2, 64)	0
flatten (Flatten)	(None, 640)	0
dense (Dense)	(None, 301)	192,941

lambda (Lambda)	(None, 301)	0

Total params: 302,653 (1.15 MB)

Trainable params: 302,205 (1.15 MB)

Non-trainable params: 448 (1.75 KB)

APPENDIX B

Table 1: Model Architecture and Detailed Parameter Values of the Custom-Built Model Designed For Classification and Segmentation.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 128, 128, 3)	0	-
conv2d (Conv2D)	(None, 128, 128, 32)	896	input_layer[0][0]
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18,496	max_pooling2d[0]…
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 32, 32, 128)	73,856	max_pooling2d_1[…
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 128)	0	conv2d_2[0][0]
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 128)	65,664	max_pooling2d_2[…
conv2d_transpose_1	(None, 64, 64,	32,832	conv2d_transpose…

(Conv2DTranspose)	64)		
flatten (Flatten)	(None, 32768)	0	max_pooling2d_2[…
conv2d_transpose_2 (Conv2DTranspose)	(None, 128, 128, 32)	8,224	conv2d_transpose…
classification (Dense)	(None, 37)	1,212,453	flatten[0][0]
segmentation (Conv2D)	(None, 128, 128, 1)	33	conv2d_transpose…

Total params: 1,412,454 (5.39 MB)

Trainable params: 1,412,454 (5.39 MB)

Non-trainable params: 0 (0.00 B)

Table 2: Model Architecture and Detailed Parameter Values of the Fine-Tuned Model Designed For Classification and Segmentation.

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 128, 128, 3)	0	-
conv (Conv2D)	(None, 64, 64, 16)	432	input_layer_1[0]…
conv_bn (BatchNormalizatio…	(None, 64, 64, 16)	64	conv[0][0]
activation (Activation)	(None, 64, 64, 16)	0	conv_bn[0][0]
expanded_conv_dept… (ZeroPadding2D)	(None, 65, 65, 16)	0	activation[0][0]
expanded_conv_dept… (DepthwiseConv2D)	(None, 32, 32, 16)	144	expanded_conv_de…

expanded_conv_dept… (BatchNormalizatio…	(None, 32, 32, 16)	64	expanded_conv_de…
re_lu (ReLU)	(None, 32, 32, 16)	0	expanded_conv_de…
expanded_conv_sque… (GlobalAveragePool…	(None, 1, 1, 16)	0	re_lu[0][0]
expanded_conv_sque… (Conv2D)	(None, 1, 1, 8)	136	expanded_conv_sq…
expanded_conv_sque… (ReLU)	(None, 1, 1, 8)	0	expanded_conv_sq…
expanded_conv_sque… (Conv2D)	(None, 1, 1, 16)	144	expanded_conv_sq…
add (Add)	(None, 1, 1, 16)	0	expanded_conv_sq…
re_lu_1 (ReLU)	(None, 1, 1, 16)	0	add[0][0]
multiply (Multiply)	(None, 1, 1, 16)	0	re_lu_1[0][0]
expanded_conv_sque… (Multiply)	(None, 32, 32, 16)	0	re_lu[0][0], multiply[0][0]
expanded_conv_proj… (Conv2D)	(None, 32, 32, 16)	256	expanded_conv_sq…
expanded_conv_proj… (BatchNormalizatio…	(None, 32, 32, 16)	64	expanded_conv_pr…
expanded_conv_1_ex… (Conv2D)	(None, 32, 32, 72)	1,152	expanded_conv_pr…
expanded_conv_1_ex… (BatchNormalizatio…	(None, 32, 32, 72)	288	expanded_conv_1_…

re_lu_2 (ReLU)	(None, 32, 32, 72)	0	expanded_conv_1_…
expanded_conv_1_de… (ZeroPadding2D)	(None, 33, 33, 72)	0	re_lu_2[0][0]
expanded_conv_1_de… (DepthwiseConv2D)	(None, 16, 16, 72)	648	expanded_conv_1_…
expanded_conv_1_de… (BatchNormalizatio…	(None, 16, 16, 72)	288	expanded_conv_1_…
re_lu_3 (ReLU)	(None, 16, 16, 72)	0	expanded_conv_1_…
expanded_conv_1_pr… (Conv2D)	(None, 16, 16, 24)	1,728	re_lu_3[0][0]
expanded_conv_1_pr… (BatchNormalizatio…	(None, 16, 16, 24)	96	expanded_conv_1_…
expanded_conv_2_ex… (Conv2D)	(None, 16, 16, 88)	2,112	expanded_conv_1_…
expanded_conv_2_ex… (BatchNormalizatio…	(None, 16, 16, 88)	352	expanded_conv_2_…
re_lu_4 (ReLU)	(None, 16, 16, 88)	0	expanded_conv_2_…
expanded_conv_2_de… (DepthwiseConv2D)	(None, 16, 16, 88)	792	re_lu_4[0][0]
expanded_conv_2_de… (BatchNormalizatio…	(None, 16, 16, 88)	352	expanded_conv_2_…
re_lu_5 (ReLU)	(None, 16, 16, 88)	0	expanded_conv_2_…

expanded_conv_2_pr… (Conv2D)	(None, 16, 16, 24)	2,112	re_lu_5[0][0]
expanded_conv_2_pr… (BatchNormalizatio…	(None, 16, 16, 24)	96	expanded_conv_2_…
expanded_conv_2_add (Add)	(None, 16, 16, 24)	0	expanded_conv_1_… expanded_conv_2_…
expanded_conv_3_ex… (Conv2D)	(None, 16, 16, 96)	2,304	expanded_conv_2_…
expanded_conv_3_ex… (BatchNormalizatio…	(None, 16, 16, 96)	384	expanded_conv_3_…
activation_1 (Activation)	(None, 16, 16, 96)	0	expanded_conv_3_…
expanded_conv_3_de… (ZeroPadding2D)	(None, 19, 19, 96)	0	activation_1[0][…
expanded_conv_3_de… (DepthwiseConv2D)	(None, 8, 8, 96)	2,400	expanded_conv_3_…
expanded_conv_3_de… (BatchNormalizatio…	(None, 8, 8, 96)	384	expanded_conv_3_…
activation_2 (Activation)	(None, 8, 8, 96)	0	expanded_conv_3_…
expanded_conv_3_sq… (GlobalAveragePool…	(None, 1, 1, 96)	0	activation_2[0][…
expanded_conv_3_sq… (Conv2D)	(None, 1, 1, 24)	2,328	expanded_conv_3_…
expanded_conv_3_sq… (ReLU)	(None, 1, 1, 24)	0	expanded_conv_3_…

expanded_conv_3_sq… (Conv2D)	(None, 1, 1, 96)	2,400	expanded_conv_3_…
add_1 (Add)	(None, 1, 1, 96)	0	expanded_conv_3_…
re_lu_6 (ReLU)	(None, 1, 1, 96)	0	add_1[0][0]
multiply_1 (Multiply)	(None, 1, 1, 96)	0	re_lu_6[0][0]
expanded_conv_3_sq… (Multiply)	(None, 8, 8, 96)	0	activation_2[0][… multiply_1[0][0]
expanded_conv_3_pr… (Conv2D)	(None, 8, 8, 40)	3,840	expanded_conv_3_…
expanded_conv_3_pr… (BatchNormalizatio…	(None, 8, 8, 40)	160	expanded_conv_3_…
expanded_conv_4_ex… (Conv2D)	(None, 8, 8, 240)	9,600	expanded_conv_3_…
expanded_conv_4_ex… (BatchNormalizatio…	(None, 8, 8, 240)	960	expanded_conv_4_…
activation_3 (Activation)	(None, 8, 8, 240)	0	expanded_conv_4_…
expanded_conv_4_de… (DepthwiseConv2D)	(None, 8, 8, 240)	6,000	activation_3[0][…
expanded_conv_4_de… (BatchNormalizatio…	(None, 8, 8, 240)	960	expanded_conv_4_…
activation_4 (Activation)	(None, 8, 8, 240)	0	expanded_conv_4_…
expanded_conv_4_sq…	(None, 1, 1, 240)	0	activation_4[0][…

(GlobalAveragePool…			
expanded_conv_4_sq… (Conv2D)	(None, 1, 1, 64)	15,424	expanded_conv_4_…
expanded_conv_4_sq… (ReLU)	(None, 1, 1, 64)	0	expanded_conv_4_…
expanded_conv_4_sq… (Conv2D)	(None, 1, 1, 240)	15,600	expanded_conv_4_…
add_2 (Add)	(None, 1, 1, 240)	0	expanded_conv_4_…
re_lu_7 (ReLU)	(None, 1, 1, 240)	0	add_2[0][0]
multiply_2 (Multiply)	(None, 1, 1, 240)	0	re_lu_7[0][0]
expanded_conv_4_sq… (Multiply)	(None, 8, 8, 240)	0	activation_4[0][… multiply_2[0][0]
expanded_conv_4_pr… (Conv2D)	(None, 8, 8, 40)	9,600	expanded_conv_4_…
expanded_conv_4_pr… (BatchNormalizatio…	(None, 8, 8, 40)	160	expanded_conv_4_…
expanded_conv_4_add (Add)	(None, 8, 8, 40)	0	expanded_conv_3_… expanded_conv_4_…
expanded_conv_5_ex… (Conv2D)	(None, 8, 8, 240)	9,600	expanded_conv_4_…
expanded_conv_5_ex… (BatchNormalizatio…	(None, 8, 8, 240)	960	expanded_conv_5_…
activation_5 (Activation)	(None, 8, 8, 240)	0	expanded_conv_5_…

expanded_conv_5_de… (DepthwiseConv2D)	(None, 8, 8, 240)	6,000	activation_5[0][…
expanded_conv_5_de… (BatchNormalizatio…	(None, 8, 8, 240)	960	expanded_conv_5_…
activation_6 (Activation)	(None, 8, 8, 240)	0	expanded_conv_5_…
expanded_conv_5_sq… (GlobalAveragePool…	(None, 1, 1, 240)	0	activation_6[0][…
expanded_conv_5_sq… (Conv2D)	(None, 1, 1, 64)	15,424	expanded_conv_5_…
expanded_conv_5_sq… (ReLU)	(None, 1, 1, 64)	0	expanded_conv_5_…
expanded_conv_5_sq… (Conv2D)	(None, 1, 1, 240)	15,600	expanded_conv_5_…
add_3 (Add)	(None, 1, 1, 240)	0	expanded_conv_5_…
re_lu_8 (ReLU)	(None, 1, 1, 240)	0	add_3[0][0]
multiply_3 (Multiply)	(None, 1, 1, 240)	0	re_lu_8[0][0]
expanded_conv_5_sq… (Multiply)	(None, 8, 8, 240)	0	activation_6[0][… multiply_3[0][0]
expanded_conv_5_pr… (Conv2D)	(None, 8, 8, 40)	9,600	expanded_conv_5_…
expanded_conv_5_pr… (BatchNormalizatio…	(None, 8, 8, 40)	160	expanded_conv_5_…
expanded_conv_5_add (Add)	(None, 8, 8, 40)	0	expanded_conv_4_… expanded_conv_5_…

expanded_conv_6_ex… (Conv2D)	(None, 8, 8, 120)	4,800	expanded_conv_5_…
expanded_conv_6_ex… (BatchNormalizatio…	(None, 8, 8, 120)	480	expanded_conv_6_…
activation_7 (Activation)	(None, 8, 8, 120)	0	expanded_conv_6_…
expanded_conv_6_de… (DepthwiseConv2D)	(None, 8, 8, 120)	3,000	activation_7[0][…
expanded_conv_6_de… (BatchNormalizatio…	(None, 8, 8, 120)	480	expanded_conv_6_…
activation_8 (Activation)	(None, 8, 8, 120)	0	expanded_conv_6_…
expanded_conv_6_sq… (GlobalAveragePool…	(None, 1, 1, 120)	0	activation_8[0][…
expanded_conv_6_sq… (Conv2D)	(None, 1, 1, 32)	3,872	expanded_conv_6_…
expanded_conv_6_sq… (ReLU)	(None, 1, 1, 32)	0	expanded_conv_6_…
expanded_conv_6_sq… (Conv2D)	(None, 1, 1, 120)	3,960	expanded_conv_6_…
add_4 (Add)	(None, 1, 1, 120)	0	expanded_conv_6_…
re_lu_9 (ReLU)	(None, 1, 1, 120)	0	add_4[0][0]
multiply_4 (Multiply)	(None, 1, 1, 120)	0	re_lu_9[0][0]
expanded_conv_6_sq…	(None, 8, 8, 120)	0	activation_8[0][…

(Multiply)			multiply_4[0][0]
expanded_conv_6_pr… (Conv2D)	(None, 8, 8, 48)	5,760	expanded_conv_6_…
expanded_conv_6_pr… (BatchNormalizatio…	(None, 8, 8, 48)	192	expanded_conv_6_…
expanded_conv_7_ex… (Conv2D)	(None, 8, 8, 144)	6,912	expanded_conv_6_…
expanded_conv_7_ex… (BatchNormalizatio…	(None, 8, 8, 144)	576	expanded_conv_7_…
activation_9 (Activation)	(None, 8, 8, 144)	0	expanded_conv_7_…
expanded_conv_7_de… (DepthwiseConv2D)	(None, 8, 8, 144)	3,600	activation_9[0][…
expanded_conv_7_de… (BatchNormalizatio…	(None, 8, 8, 144)	576	expanded_conv_7_…
activation_10 (Activation)	(None, 8, 8, 144)	0	expanded_conv_7_…
expanded_conv_7_sq… (GlobalAveragePool…	(None, 1, 1, 144)	0	activation_10[0]…
expanded_conv_7_sq… (Conv2D)	(None, 1, 1, 40)	5,800	expanded_conv_7_…
expanded_conv_7_sq… (ReLU)	(None, 1, 1, 40)	0	expanded_conv_7_…
expanded_conv_7_sq… (Conv2D)	(None, 1, 1, 144)	5,904	expanded_conv_7_…
add_5 (Add)	(None, 1, 1, 144)	0	expanded_conv_7_…

re_lu_10 (ReLU)	(None, 1, 1, 144)	0	add_5[0][0]
multiply_5 (Multiply)	(None, 1, 1, 144)	0	re_lu_10[0][0], activation_10[0]…
expanded_conv_7_sq… (Multiply)	(None, 8, 8, 144)	0	multiply_5[0][0]
expanded_conv_7_pr… (Conv2D)	(None, 8, 8, 48)	6,912	expanded_conv_7_…
expanded_conv_7_pr… (BatchNormalizatio…	(None, 8, 8, 48)	192	expanded_conv_7_…
expanded_conv_7_add (Add)	(None, 8, 8, 48)	0	expanded_conv_6_…, expanded_conv_7_…
expanded_conv_8_ex… (Conv2D)	(None, 8, 8, 288)	13,824	expanded_conv_7_…
expanded_conv_8_ex… (BatchNormalizatio…	(None, 8, 8, 288)	1,152	expanded_conv_8_…
activation_11 (Activation)	(None, 8, 8, 288)	0	expanded_conv_8_…
expanded_conv_8_de… (ZeroPadding2D)	(None, 11, 11, 288)	0	activation_11[0]…
expanded_conv_8_de… (DepthwiseConv2D)	(None, 4, 4, 288)	7,200	expanded_conv_8_…
expanded_conv_8_de… (BatchNormalizatio…	(None, 4, 4, 288)	1,152	expanded_conv_8_…
activation_12 (Activation)	(None, 4, 4, 288)	0	expanded_conv_8_…

expanded_conv_8_sq… (GlobalAveragePool…	(None, 1, 1, 288)	0	activation_12[0]…
expanded_conv_8_sq… (Conv2D)	(None, 1, 1, 72)	20,808	expanded_conv_8_…
expanded_conv_8_sq… (ReLU)	(None, 1, 1, 72)	0	expanded_conv_8_…
expanded_conv_8_sq… (Conv2D)	(None, 1, 1, 288)	21,024	expanded_conv_8_…
add_6 (Add)	(None, 1, 1, 288)	0	expanded_conv_8_…
re_lu_11 (ReLU)	(None, 1, 1, 288)	0	add_6[0][0]
multiply_6 (Multiply)	(None, 1, 1, 288)	0	re_lu_11[0][0]
expanded_conv_8_sq… (Multiply)	(None, 4, 4, 288)	0	activation_12[0]… multiply_6[0][0]
expanded_conv_8_pr… (Conv2D)	(None, 4, 4, 96)	27,648	expanded_conv_8_…
expanded_conv_8_pr… (BatchNormalizatio…	(None, 4, 4, 96)	384	expanded_conv_8_…
expanded_conv_9_ex… (Conv2D)	(None, 4, 4, 576)	55,296	expanded_conv_8_…
expanded_conv_9_ex… (BatchNormalizatio…	(None, 4, 4, 576)	2,304	expanded_conv_9_…
activation_13 (Activation)	(None, 4, 4, 576)	0	expanded_conv_9_…
expanded_conv_9_de… (DepthwiseConv2D)	(None, 4, 4, 576)	14,400	activation_13[0]…

expanded_conv_9_de… (BatchNormalizatio…	(None, 4, 4, 576)	2,304	expanded_conv_9_…
activation_14 (Activation)	(None, 4, 4, 576)	0	expanded_conv_9_…
expanded_conv_9_sq… (GlobalAveragePool…	(None, 1, 1, 576)	0	activation_14[0]…
expanded_conv_9_sq… (Conv2D)	(None, 1, 1, 144)	83,088	expanded_conv_9_…
expanded_conv_9_sq… (ReLU)	(None, 1, 1, 144)	0	expanded_conv_9_…
expanded_conv_9_sq… (Conv2D)	(None, 1, 1, 576)	83,520	expanded_conv_9_…
add_7 (Add)	(None, 1, 1, 576)	0	expanded_conv_9_…
re_lu_12 (ReLU)	(None, 1, 1, 576)	0	add_7[0][0]
multiply_7 (Multiply)	(None, 1, 1, 576)	0	re_lu_12[0][0]
expanded_conv_9_sq… (Multiply)	(None, 4, 4, 576)	0	activation_14[0]… multiply_7[0][0]
expanded_conv_9_pr… (Conv2D)	(None, 4, 4, 96)	55,296	expanded_conv_9_…
expanded_conv_9_pr… (BatchNormalizatio…	(None, 4, 4, 96)	384	expanded_conv_9_…
expanded_conv_9_add (Add)	(None, 4, 4, 96)	0	expanded_conv_8_… expanded_conv_9_…
expanded_conv_10_e…	(None, 4, 4, 576)	55,296	expanded_conv_9_…

(Conv2D)			
expanded_conv_10_e… (BatchNormalizatio…	(None, 4, 4, 576)	2,304	expanded_conv_10…
activation_15 (Activation)	(None, 4, 4, 576)	0	expanded_conv_10…
expanded_conv_10_d… (DepthwiseConv2D)	(None, 4, 4, 576)	14,400	activation_15[0]…
expanded_conv_10_d… (BatchNormalizatio…	(None, 4, 4, 576)	2,304	expanded_conv_10…
activation_16 (Activation)	(None, 4, 4, 576)	0	expanded_conv_10…
expanded_conv_10_s… (GlobalAveragePool…	(None, 1, 1, 576)	0	activation_16[0]…
expanded_conv_10_s… (Conv2D)	(None, 1, 1, 144)	83,088	expanded_conv_10…
expanded_conv_10_s… (ReLU)	(None, 1, 1, 144)	0	expanded_conv_10…
expanded_conv_10_s… (Conv2D)	(None, 1, 1, 576)	83,520	expanded_conv_10…
add_8 (Add)	(None, 1, 1, 576)	0	expanded_conv_10…
re_lu_13 (ReLU)	(None, 1, 1, 576)	0	add_8[0][0]
multiply_8 (Multiply)	(None, 1, 1, 576)	0	re_lu_13[0][0]
expanded_conv_10_s… (Multiply)	(None, 4, 4, 576)	0	activation_16[0]… multiply_8[0][0]

expanded_conv_10_p… (Conv2D)	(None, 4, 4, 96)	55,296	expanded_conv_10…
expanded_conv_10_p… (BatchNormalizatio…	(None, 4, 4, 96)	384	expanded_conv_10…
expanded_conv_10_a… (Add)	(None, 4, 4, 96)	0	expanded_conv_9_… expanded_conv_10…
conv_1 (Conv2D)	(None, 4, 4, 576)	55,296	expanded_conv_10…
conv2d_transpose_3 (Conv2DTranspose)	(None, 8, 8, 128)	295,040	conv_1[0][0]
conv2d_transpose_4 (Conv2DTranspose)	(None, 16, 16, 64)	32,832	conv2d_transpose…
conv2d_transpose_5 (Conv2DTranspose)	(None, 32, 32, 32)	8,224	conv2d_transpose…
conv2d_transpose_6 (Conv2DTranspose)	(None, 64, 64, 16)	2,064	conv2d_transpose…
global_average_poo… (GlobalAveragePool…	(None, 576)	0	conv_1[0][0]
conv2d_transpose_7 (Conv2DTranspose)	(None, 128, 128, 8)	520	conv2d_transpose…
classification (Dense)	(None, 37)	21,349	global_average_p…
segmentation (Conv2D)	(None, 128, 128, 1)	9	conv2d_transpose…

Total params: 1,296,854 (4.95 MB)

Trainable params: 1,285,894 (4.91 MB)

Non-trainable params: 10,960 (42.81 KB)

Table 3: Classification Report of the Custom Model.

	precision	recall	f1-score	support
0	0.23	0.14	0.18	98
1	0.18	0.28	0.22	100
2	0.04	0.02	0.03	100
3	0.18	0.12	0.14	100
4	0.18	0.21	0.20	100
5	0.17	0.33	0.22	100
6	0.16	0.19	0.18	100
7	0.34	0.39	0.36	88
8	0.07	0.02	0.03	99
9	0.22	0.25	0.23	100
10	0.15	0.05	0.08	100
11	0.36	0.30	0.33	97
12	0.07	0.05	0.06	100
13	0.07	0.14	0.10	100
14	0.23	0.28	0.25	100
15	0.20	0.22	0.21	100
16	0.26	0.13	0.17	100
17	0.22	0.19	0.20	100
18	0.23	0.27	0.25	99
19	0.27	0.12	0.17	100
20	0.28	0.23	0.25	100
21	0.11	0.13	0.12	100
22	0.42	0.51	0.46	100
23	0.26	0.31	0.29	100
24	0.32	0.24	0.28	100
25	0.25	0.26	0.25	100
26	0.25	0.38	0.30	100
27	0.17	0.17	0.17	100
28	0.27	0.26	0.26	100
29	0.44	0.19	0.27	100
30	0.16	0.28	0.21	99
31	0.18	0.20	0.19	100
32	0.24	0.21	0.23	100
33	0.14	0.22	0.17	100
34	0.11	0.07	0.08	89
35	0.15	0.11	0.13	100
36	0.20	0.18	0.19	100

Table 4: Classification Report of the Fine-Tuned Model.

	precision	recall	f1-score	support
0	0.52	0.13	0.21	98
1	0.45	0.10	0.16	100
2	0.00	0.00	0.00	100
3	0.63	0.19	0.29	100
4	0.33	0.18	0.23	100
5	0.89	0.08	0.15	100
6	0.30	0.64	0.41	100
7	0.78	0.08	0.14	88

8	0.76	0.16	0.27	99
9	0.63	0.27	0.38	100
10	0.67	0.02	0.04	100
11	0.62	0.52	0.56	97
12	0.00	0.00	0.00	100
13	0.23	0.07	0.11	100
14	1.00	0.06	0.11	100
15	0.33	0.07	0.12	100
16	0.07	0.99	0.13	100
17	0.67	0.22	0.33	100
18	0.21	0.68	0.33	99
19	0.47	0.23	0.31	100
20	0.39	0.24	0.30	100
21	1.00	0.02	0.04	100
22	0.32	0.08	0.13	100
23	0.55	0.37	0.44	100
24	0.46	0.39	0.42	100
25	0.63	0.37	0.47	100
26	0.25	0.03	0.05	100
27	0.49	0.33	0.40	100
28	0.54	0.38	0.44	100
29	0.46	0.32	0.38	100
30	0.30	0.75	0.43	99
31	0.57	0.25	0.35	100
32	0.43	0.51	0.47	100
33	0.80	0.04	0.08	100
34	0.27	0.28	0.27	89
35	0.14	0.35	0.20	100
36	0.30	0.03	0.05	100