

and 270 degrees improves the model's ability to recognize and classify the rotated shape of the image. Third, add random noise. By adding random noise to the image, the model becomes less sensitive to noise and can respond to variability. Finally, gamma adjustment. Adjusting the gamma value of the image to 1 and 1.5 adjusts the contrast and helps the model produce consistent results under various lighting conditions.

Strengths	<ul style="list-style-type: none">• Random Forest is an ensemble model that combines multiple decision trees for predictions. This allows it to learn various features and provides robustness against data diversity and noise, enhancing the model's generalization ability.• Random Forest can estimate the importance of each feature. This helps determine the usefulness of HOG features and identify which features have a significant impact on classification decisions.• Random Forest tends to prevent overfitting through bootstrapping and random attribute selection. This enables the construction of a more generalized model.• The decision trees in Random Forest operate independently, allowing multiple trees to be trained and predicted in parallel. This enables efficient processing, particularly with large datasets.
Weaknesses	<ul style="list-style-type: none">• Random Forest involves using multiple decision trees, which can result in higher computational costs. Particularly with large datasets or numerous features, training and prediction may take considerable time.• Random Forest is an ensemble technique that combines multiple decision trees. As a result, interpreting model predictions or understanding the individual feature influences may be challenging.• HOG features are primarily used in image processing and rely heavily on edges and gradient orientations. Therefore, HOG features are most suitable when edges and gradient directions are crucial in providing effective information within the image. For other types of data, considering different features may be more appropriate.• HOG features represent global characteristics of an image, potentially overlooking local variations or detailed information. Consequently, HOG features may not be the optimal choice for certain problems.

Overall, by using random forest with HOG can estimate the importance of each feature which allows learn various features and provides robustness. The 256_ObjectCategories dataset includes a total of 30607 samples, which has reduced the sample to 2500 because it requires too much memory to use random forest with HOG.

4.4 Method 4: Visual Transformer

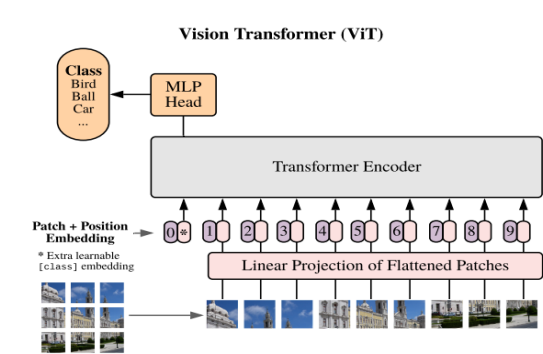


Figure of ViT Architecture

Visual Transformer from google uses transformer architecture that is based on text-based tasks. Overall approach of this model starts with taking original image and dividing it into patches. These patches can be 6x6 or 16x16 and flatten out into sequences and pass these patches into transformer layer. This is similar to transformers using position embedding where these embeddings indicate the location of the left or right order of text sequences. In the case of ViT, embeddings allow transformer layers to indicate where each patch were sourced from the original image. Finally, these embeddings are added to the original patches and passed into transformer encoder/decoder where it can be

any kind of architecture that is compatible with patch implemented input representation.

Hyperparameter of ViT

Layer	Variable	Value
Inputs	Input_shape	(128,128,3)
inception	Image_size	128
	Patch_size	10
	Num_patches	(Image_size//patch_size) ** 2)
Transformer	Projection_dim	64
	Num_heads	4
	Transformer_units	[Projection_dim * 2, projection_dim]
	Transformer_layers	4
Multi-layer perceptron	Mlp_head_units	[2048, 1024]

Table of VIT Hyper-Parameters

Visual Transformer Is a complex deep layer model, and hyperparameter of the transformer layer significantly affects the end results thus, it is essential to talk about the hyper-parameters used in this model. Learning rate/weight-decay (0.001, 0.0001) is used in *Adam-optimizer*. Patch size of 6 is going to be size of the patches where original 128x128 image is transformed into grid size of 6x6 crop. Then next parameter is to store *number of patches* in a variable where transformer layer is going to be using when defining the position embedding. Size of hidden dimension feature vectors (*projection dim =64*) is to project these patches into these 64 feature vectors then concatenate together as input into the first transformer encoder layer. Next hyper parameter is number of attention heads (*num_heads = 4*). In multi-head self-attention, this demonstrates having four different parameterizations of the query key and value matrices. Four separate transformations then aggregate the outputs of these four separate self-attention parameterizations from the previous layer. *Transformer unit lists* allows to override the multi-layer perceptron layer (MLP layer) to add skip connection that is like Resnet. In transformer block, it has skip connection from the output of self-attention then it goes through a forward layer that is going to compress the dimensionality then it skips ahead with previous output from the previous dense layer. Finally, transformer blocks (*transformer_layers = 5*) which demonstrates stacking top of the transformer encoder block that repeats it five times then there is hidden dimension of multi-layer perceptron layers (*mlp_head_units*) = [2048, 1024]).

Data Augmentation

Type	Value
layers.Normalization()	
layers.Resizing	(128,128)
Layers.RandomFlip()	(horizontal)
layers.RandomRotation	(factor = 0.02)
Layers.RandomZoom()	(height_factor=0.2, width_factor=0.2)

Table of Data augmentation

Data augmentation is applied to prevent overfitting with the Caltech-256 data set. There is normalization layer that is going to normalize the pixel value in the data set images around given mean and standard deviation and then calculate this mean and standard deviation from the data set by using this *data_augmentation.layers[0].adapt(x_train)* then resize the images to 128x128 then add horizontal flipping, random rotation and random zooming.

Implementation of multilayer perceptron (MLP)

When over-writing the dense layer to add skip connection, rather than using *model.adddense*, custom *MLP* function is used to overwrite with Keras functional Api.

Compiling Model

Model is compiled from above hyperparameters with tensorflow-addons Adam optimizer. Loss function is *SparseCategoricalCrossentropy* with the metrics of Accuracy and top 5 accuracy. In the case of image classification of Caltech-256 dataset, it is common to have top 5-k categorical accuracy where this is similar to information retrieval.

Since dataset have massive set of potential class labels (256), The top-5 accuracy metric provides an approximation of how well the model is performing, even when it makes incorrect predictions. It assesses whether the true label is among the top 5 predictions, indicating that the model has some level of understanding and is able to make reasonable guesses even if it does not predict the exact class label. For example, if there is a fighter-jet image and model is classifying as an airplane class, at least the fighter-jet is still in the top-5 prediction. This approach allows for a more comprehensive evaluation of the model's performance in scenarios where there are many possible class labels, and it is desirable to know how well the model is able to narrow down the correct class within the top 5 predictions.

Strengths	<ul style="list-style-type: none">Scalability: ViT can efficiently handle large-scale datasets, including Caltech 256, which contains a significant number of images (256 object categories with varying image sizes).Global Context: ViT captures global image information by dividing the input image into patches and treating them as sequence data. This enables the model to learn context and relationships among different parts of the image.Generalization: ViT has demonstrated strong generalization capabilities, achieving state-of-the-art performance on various image classification benchmarks. This indicates its ability to learn abstract representations that can be applied to different datasets, including Caltech 256.
Weaknesses	<ul style="list-style-type: none">Large Memory Requirements: ViT's reliance on self-attention mechanisms introduces scalability challenges. As the input image size increases, the memory requirements of the model also increase significantly. This can make training and inference computationally expensive, particularly for high-resolution images.Spatial Sensitivity: ViT processes images by dividing them into fixed-size patches, disregarding the spatial information within each patch. As a result, ViT may struggle with tasks that require precise localization or fine-grained spatial understanding.Limited Robustness to Input Variations: ViT's performance may degrade when applied to images with diverse lighting conditions, occlusions, or other types of variations that were not present in the training data. This limitation arises due to the model's reliance on large-scale datasets that may not capture all possible variations in real-world images.

Strengths of the Vision Transformer (ViT) model include its scalability for large datasets, global context understanding through attention mechanisms, and strong generalization performance. However, limitations of ViT include its lack of spatial sensitivity within patches, large memory requirements, and decreased robustness to input variations not seen in training data.

5.0 Individual Results

5.1 Method 1: Fine-Tuned Model

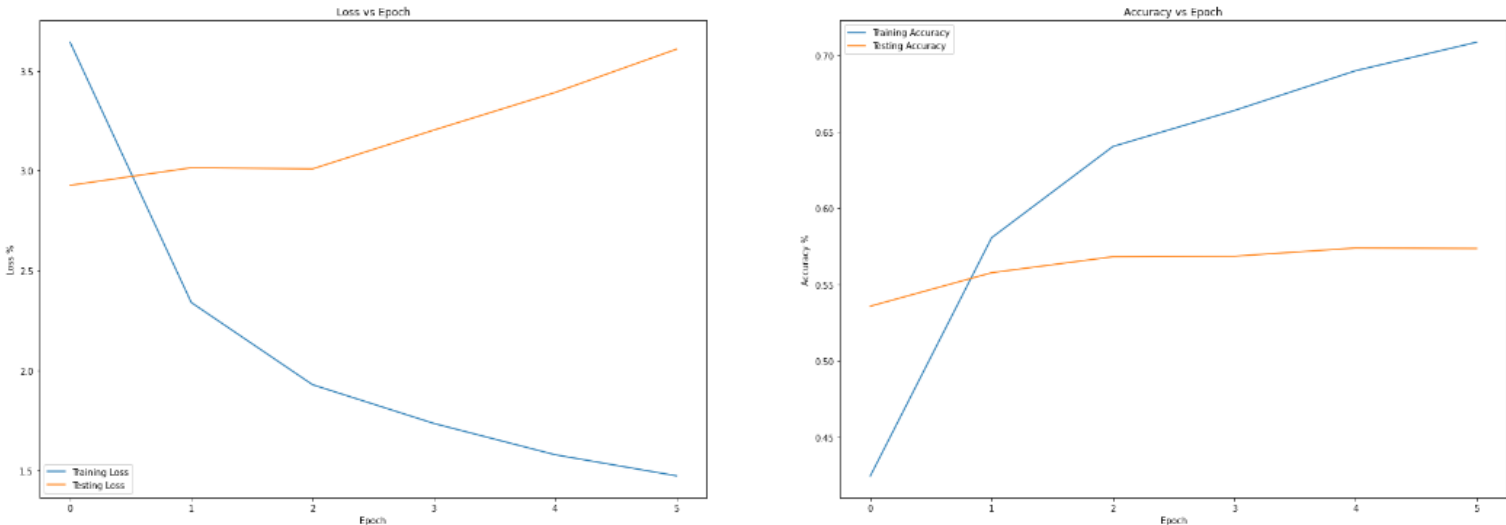


Figure 3 Method 1's Initial Training

Found in Figure 3 is the accuracy and loss performance of initial training. It's observed that the model was able to achieve 57.35% accuracy on the validation data, and 70.86% accuracy on the training data. The training for the initial

model took 8 minutes It's observed in the Loss vs Epoch plot of the graph which demonstrates that the model is overfitting at around 2 epochs due to the loss rising in the validation despite the training loss still reducing. From the initial training it's clear that the fine-tuned model can extract relevant features into embeddings and are able to use these features to successfully identify the provided classifications. However, 57.35% accuracy is still not an impressive metric to possess for the model. This leads to the fine-tuning results.

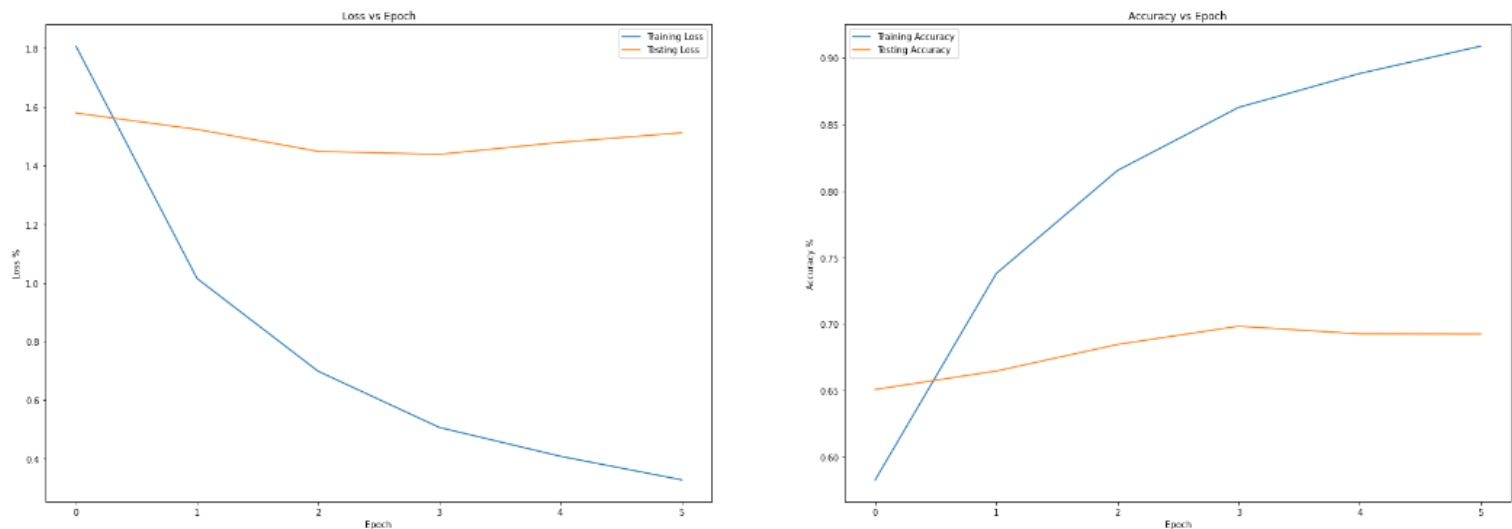


Figure 4 Method 1's Fine-tuning Training

By fine-tuning the performance of the model, the model achieved improved results. This can be seen in Figure 4 where the validation accuracy was able to reach 69.22%, improving from the previous results by 11.87%. Signs of overfitting can be seen at epoch 5, as the validation data in the accuracy flattens out, while the loss begins to rise in the validation dataset. The fine-tuned models training took 30.5 minutes to train. The improved results show that the inception model has been trained to specifically identify the provided classifications by modifying the existing image net. This has resulted in the improved performance demonstrated by the model. These results generate Table 5.

	Training Accuracy	Testing Accuracy	Inference Time (Seconds)	Training Time (Minutes)
Initial Training	70.86%	57.35%	0.162	8
Fine-Tuned	90.72%	69.22%	0.295	30.5
Total				38.5

Table 5 Method 1's Final Results

The models final results can be evaluated further by observing Figure 9 found in the appendix. It's clear that many of the classifications actually perform extremely poor, such as class 3, 15, 60, 67, 70 and so on. this is found to be caused by a mixture of poor data, such as having two types of classifications in a single image, having a small size dataset for a single classification, or due possibly too large variance with a medium size dataset. However, for many of the classifications in the dataset its seen to perform fairly well, ultimately shown in the averages as 70% for the f-1 score.

5.2 Method 2: Classification using Triplet Network Embeddings

To examine the quality and discriminative power of the triplet network created in this method two t-SNE plots were generated. The first shows the embeddings generated by an untrained base network DCNN and the second showing the embeddings generated by the DCNN after the triplet network has been trained (Figure 5). It is evident that the network has effectively clustered some of the classes after the triplet network has been trained. Overall, however the triplet network hasn't done the greatest job at class separation. This is expected as the database hold too much inter-class variation to allow for perfect results. The losses for both the triplet network and the base network can be seen in figure 6, 7, and 8.

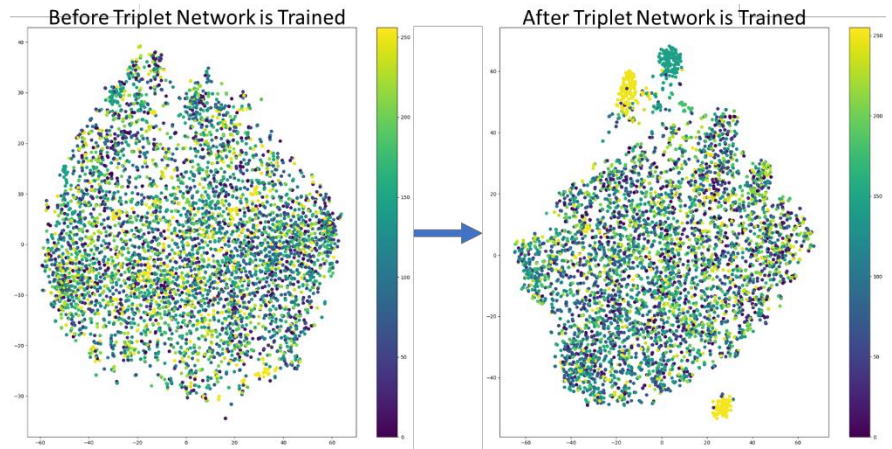


Figure 55: Plot of the base network embeddings pre and post trained.

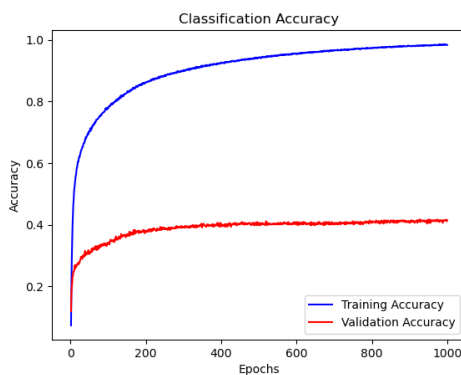


Figure 6 Model 2's Classification Accuracy

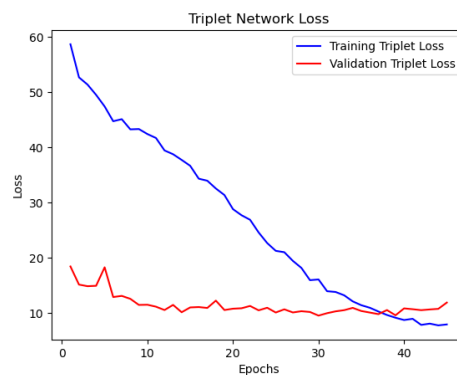


Figure 7 Model 2's Triplet Network Loss

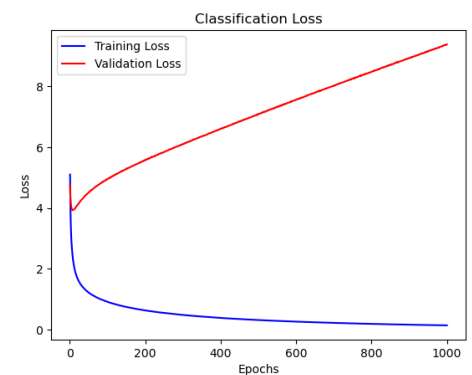


Figure 8 Model 2's Classification Loss

As seen in figure 6, validation loss of the triplet network can be seen to start increasing which is why earlier stopping was implemented (with a patience of 15) during training. This resulted in weights at about the 30th epoch being saved to the base network (Point at which the validation loss is the smallest). It is also evident that the classification model overfits at around the 10th epoch (Figure 7). This was expected however due to the nature of the embeddings it is working with. Overall it is evident that the approach had an accuracy of 99% on the training embeddings about 38% on the validation embeddings (Figure 8).

The approach was measured to take a total of 27.5 minutes to train (both the triplet network and the classifier included). It also has an inference time of 0.19 seconds.

One suggestion to improve this approach is to use transfer learning and use a pretrained model as the base network DCNN. This would result in a higher accuracy for class separation in embeddings calculations. Additionally, a different classifier could be used. Instead of a single Dense layer another DCNN could be used (This would be at the expense of training and inference time however). Furthermore, different loss functions could be experimented with such as contrastive loss or centre loss.

5.3 Method 3: Random Forest with HOG

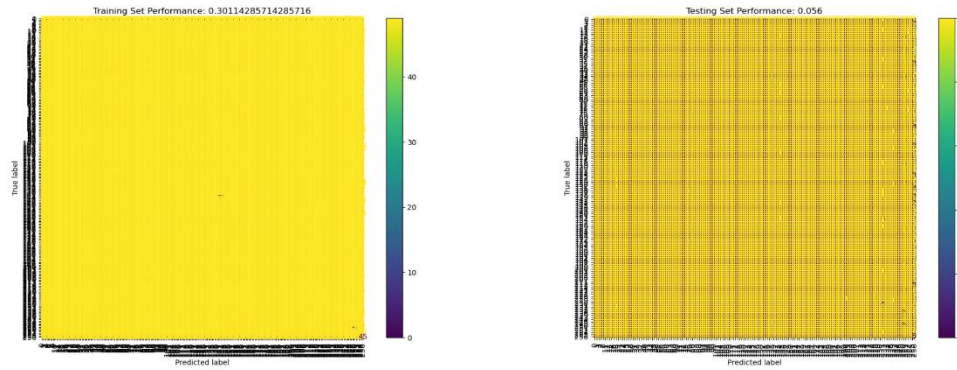


Figure Coarse HOG

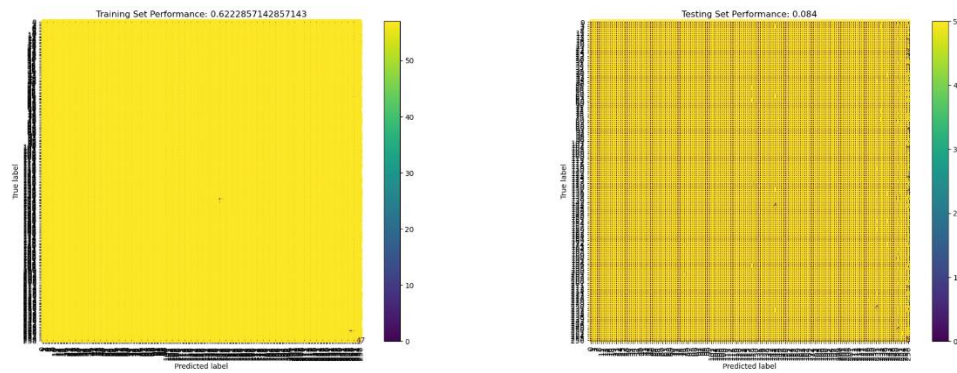
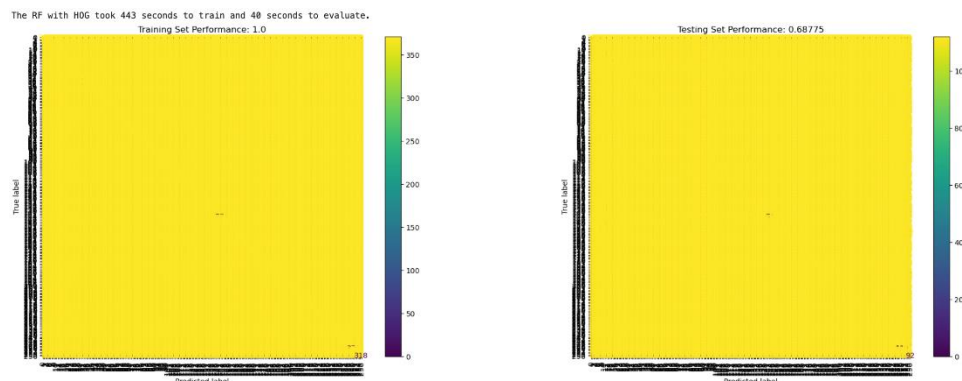


Figure Fine-grained HOG

Two approaches, coarse HOG and fine-grained HOG, were evaluated. The coarse HOG approach used parameters $\text{orientations}=4$, $\text{pixels_per_cell}=(32, 32)$, and $\text{cells_per_block}=(1, 1)$, resulting in $(1750, 64)$ and $(250, 64)$ shape for $x_{\text{train_hog}}$ and $x_{\text{test_hog}}$ respectively. It took approximately 4 seconds for HOG transformation, 0 seconds for SVM training, and achieved performance of 0.3011 on the training set and 0.056 on the testing set in approximately 15 seconds. The fine-grained HOG approach employed $\text{orientations}=16$, $\text{pixels_per_cell}=(8, 8)$, and $\text{cells_per_block}=(2, 2)$, yielding $(1750, 14400)$ and $(250, 14400)$ shape for $x_{\text{train_hog}}$ and $x_{\text{test_hog}}$ respectively. It took approximately 13 seconds for HOG transformation, 10 minutes (597 seconds) for SVM training, and achieved performance of 0.6222 on the training set and 0.084 on the testing set in approximately 15 minutes (937 seconds).

Overall, the fine-grained HOG approach outperformed the coarse HOG approach, capturing more detailed features and achieving higher accuracy. However, it required significantly more time and computational resources due to the increased complexity of feature extraction and training.



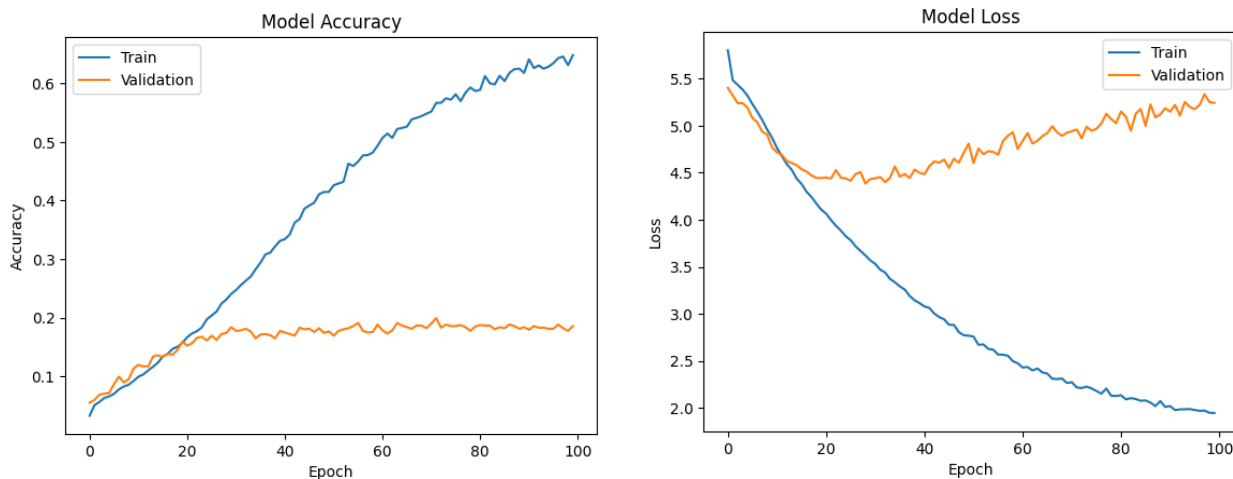
From here on, we discuss the results of random forest with data augmentation and PCA after transforming HOG. First, in the data preprocessing stage, the HOG transformation was performed in a more detailed manner. This allowed for the extraction of more refined image features. Compared to the previous coarse HOG method, the dimensionality of the feature vectors obtained through HOG transformation significantly increased. Previously, a 64-dimensional feature vector was used, but now a 14,400-dimensional feature vector was used.

After data augmentation and dimensionality reduction through PCA, the Random Forest classifier was used to train the model. However, we performed a random forest by cross-validating in advance to find the optimal parameters ($n_estimators=500$, $max_depth=7$). The RandomForest classifier utilizes ensemble learning techniques by combining multiple decision trees for prediction. The training process took approximately 443 seconds.

The performance on the training set achieved 100% accuracy, indicating that the model may have overfit the training data. Overfitting refers to a situation where the model becomes overly tuned to the training data, leading to a decrease in performance on unseen data. The performance on the test set was approximately 0.68775. This represents an improvement compared to the previous fine-grained HOG method alone (test set performance of 0.084). However, one of the reasons for the significant difference from the initial stage is that the expected PCA can result in data loss due to dimensional reduction. Nevertheless, the diversification of features through data augmentation and dimensionality reduction through PCA may have enhanced the generalization performance of the model.

In summary, by adopting the fine-grained HOG method and applying data augmentation and PCA, the performance on the test set has improved compared to previous results whereas there is still a possibility of overfitting,

5.4 Method 4: Visual Transformer

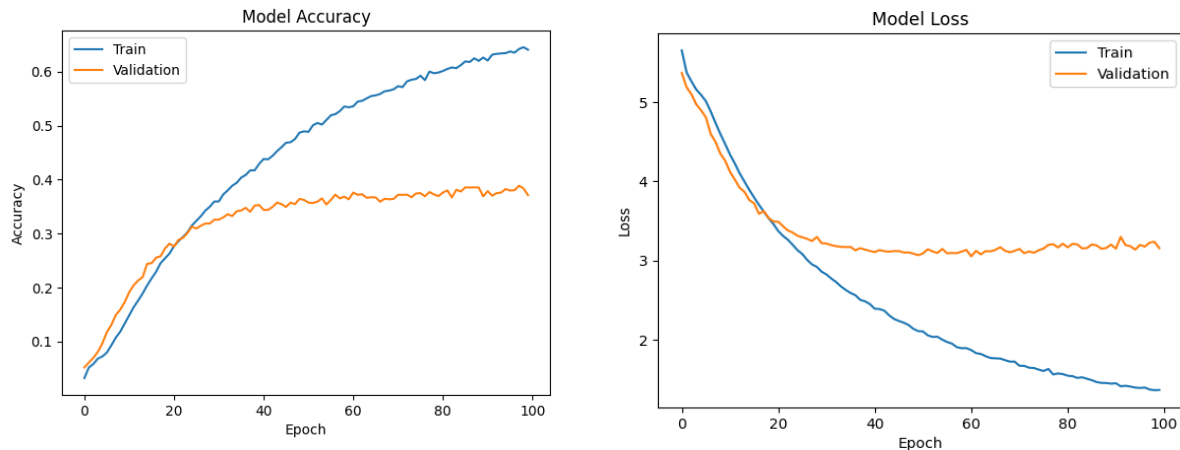


The two plots provided depict the model's accuracy and loss over epochs. Initially, the training loss steadily decreases, indicating that the model is learning and improving during training. The training accuracy and top-5 accuracy show a gradual increase as well. Similarly, the validation loss and accuracy exhibit similar trends, although the validation accuracy is slightly lower than the training accuracy.

However, after approximately 30 epochs, it becomes evident that although the training is progressing, the model's performance is not improving significantly. The accuracy and top-5 accuracy remain relatively low, and the validation accuracy does not show significant improvement. Moreover, the validation loss gradually increases while the training loss continues to decrease.

At the end of the training process, the model achieved a training accuracy of 64% and a top-5 training accuracy of 89%. However, the validation accuracy was only 18%, with a top-5 validation accuracy of 31.19%. These results strongly indicate that overfitting occurred during the training process.

The initial model performance was severely hindered by the limited dataset size of only training 7000 samples. However, upon providing a significantly larger dataset, the model's performance improved dramatically. The expanded dataset allowed the model to learn from a wider range of examples and patterns, enhancing its ability to make accurate predictions. The increased data size enabled the model to capture more complex relationships and generalize better to unseen data, resulting in a significant boost in performance. This could not be done at initial phase due to the lack of computing power. Memory was allocated with M1 pro Device with limited 16 GB memory which was too small for 30000 sample sizes. However, with the computing power from Google Colab A100, this allowed to utilize all 30000 samples from dataset while using total 31GB of system RAM and 17.5GB of GPU RAM.



Throughout the training process, there was a gradual improvement in the model's performance. The training accuracy started at a low value of 4.09% and increased to 34.95% by the end of training. The top-5 accuracy also showed improvement, starting at 7.90% and reaching 54.68% at the end of training. Similarly, the validation accuracy started at 6.76% and increased to 35.65% by the end of training. The top-5 accuracy on the validation set started at 13.10% and reached 54.74% at the end of training. Overfitting is observed by looking at the difference train/test accuracy and loss.

This model is not a competitive result on the Caltech-256 dataset compared to other models as a Fine-tuned inception classification model where it achieved 75% accuracy. According to Keras documentation, ViT achieved better performance when it is pre-trained on jft-300 million data set, then fine-tuning it on the target dataset. This indicates that vision transformers tend to benefit more from a large-scale of data in pre-training phase. As a mitigation strategy in the absence of pre-training, one can simulate its effects through data augmentation techniques to improve performance also by adjusting other hyperparameters of the model, as mentioned in the methodology.

6.0 Evaluation and Discussion

Model	Training Accuracy	Testing Accuracy	Inference Time (Seconds)	Training Time (Minutes)
Fine-Tuned Model	90.72%	69.22%	0.295	38.5
Triplet Network	99.82%	34.52%	0.19	27.5
HOG Random Forest	42.48%	26%	40	7.38
Visual Transformer	61.01%	35.64%	0.57	11.5

Fine-tuned Model

The fine-tuned model exhibits a high training accuracy of 90.72%, indicating that it performs well on the training data. However, the testing accuracy of 69.22% suggests that it struggles to generalize to unseen data. This drop in accuracy may indicate overfitting, where the model has memorized the training examples without effectively capturing the underlying patterns. It could be beneficial to consider applying regularization techniques, such as dropout or weight decay, to mitigate overfitting and improve generalization. On the positive side, the model has a relatively fast inference time of 0.295 seconds, which suggests that it can make predictions quickly. However, the training time of 38.5 minutes is relatively long, indicating a significant computational requirement for training the model.

Triplet Network

The Triplet Network demonstrates an impressively high training accuracy of 99.82%, indicating its ability to learn the training data well. However, the testing accuracy is considerably lower at 34.52%, suggesting poor generalization. This discrepancy implies that the model is overfitting the training data. The Triplet Network is known for its complexity, and such highly complex models are more prone to overfitting. It may be beneficial to explore techniques like early stopping or reducing model complexity to improve generalization. The inference time of 0.19 seconds is relatively fast, indicating efficient predictions. The training time of 27.5 minutes, while substantial, is shorter compared to the fine-tuned model.

HOG Random Forest

The HOG Random Forest model displays relatively low training and testing accuracies of 42.48% and 26%, respectively. These results suggest that the model struggles to capture the underlying patterns and features in the training data. The HOG (Histogram of Oriented Gradients) feature descriptor combined with a Random Forest classifier may not be sufficient for the complexity of the task at hand. Additionally, the relatively slow inference time of 40 seconds indicates that making predictions with this model may be time-consuming. On the positive side, the training time of 7.38 minutes is relatively short, indicating faster training compared to other models.

Visual Transformer

The Visual Transformer model achieves a moderate training accuracy of 61.01% and a testing accuracy of 35.64%. While the model performs better than the HOG Random Forest, there is still room for improvement in terms of generalization. The moderate training accuracy suggests that the model captures some patterns in the training data but fails to generalize effectively. Further exploration of the model architecture or training techniques may be necessary. The inference time of 0.57 seconds is reasonable, although slower compared to some other models. The training time of 11.5 minutes falls between the fine-tuned model and the Triplet Network.

Overall, based on the provided information, none of the models stand out as exceptionally strong performers in terms of testing accuracy. The fine-tuned model achieves the highest testing accuracy among the models mentioned, although it shows a significant drop from its training accuracy. The Triplet Network exhibits the highest training accuracy, but its testing accuracy is considerably lower. The HOG Random Forest model performs poorly in terms of both training and testing accuracies. The Visual Transformer model shows a moderate performance overall.

7.0 Conclusions and Future Works

Overall, the results of the different models on the Caltech-256 dataset show that the Fine-Tuned DCNN Model achieved the most competitive performance compared to other models. Even with the initial training of Method 1, the CNN architecture showed a performance of 56.35% accuracy, which outperformed the other models. The Siamese triplet loss approach, which aims to capture similarity or distance relationships between sample images, did not perform well on the Caltech-256 dataset. This is primarily due to the fact that the 30,000 objects in the dataset are not centered in the images and exhibit higher intra-class variance. The objects in the dataset have significant variations in scale, pose, and viewpoint, which makes it challenging for the triplet network to effectively capture the similarities between samples and perform accurate image classification. The RandomForest + HOG approach demonstrated the worst performance in the image classification task. This could be attributed to limited computing power and memory, which impacted the training process. Additionally, using only 2,500 training samples in this approach had a negative impact on model training, leading to overfitting, high variance, and limited diversity. Moreover, the HOG model is not sufficient to capture the fine-grained details and complex patterns present in the Caltech-256 dataset. The VIT model, while showing relatively poor testing accuracy compared to the DCNN architecture, still performed better than the HOG Random Forest and similar trend with triplet network models. The lower performance of the VIT model can be attributed to the absence of a pre-training process. It is important to note that VIT models typically benefit from a large-scale pre-training phase, such as pre-training on millions of images, which was not performed in this case.

One strong mitigation strategy for future work is to employ extensive data augmentation techniques to mitigate the weaknesses of the dataset. Analysing and adjusting the hyperparameters of each model is also crucial to improve their performance. Increasing the number of training samples for the HOG model would mitigate the impact of limited training data and improve testing accuracy. Additionally, for the VIT model, pre-training it on a larger dataset (at least 9 million images) as recommended in the VIT documentation would enhance its performance.

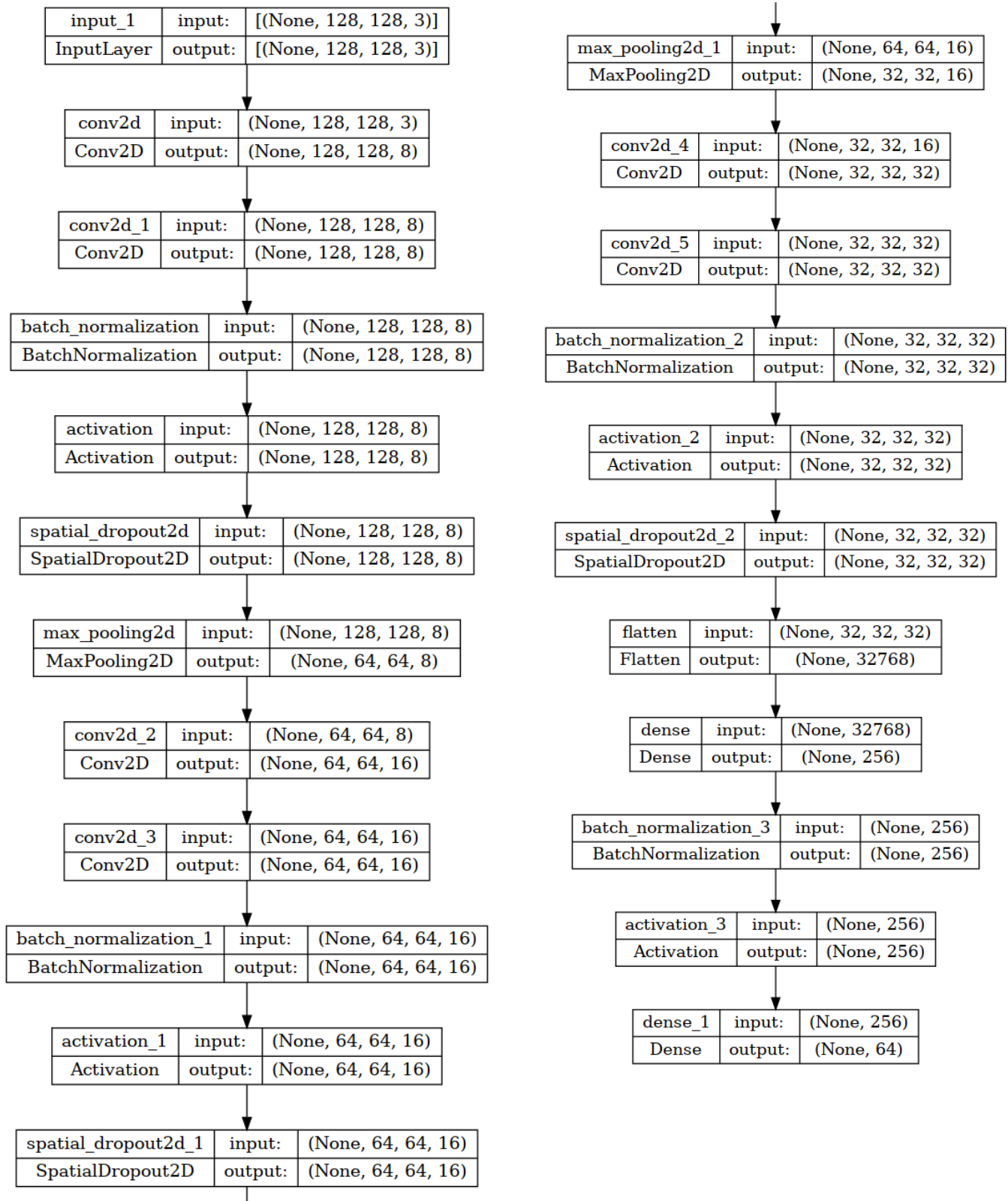
In conclusion, based on the testing accuracy of each model, it is evident that the fine-tuned DCNN model (Method 1) successfully performs image classification on the Caltech-256 dataset, while the other methods fail to achieve satisfactory results.

8.0 References

- Akkaya, B., & Çolakoğlu, N. (2019). *Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases*.
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Omran Al-Shamma, J. S., . . . Farhan, L. (2021). *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. Springer.
- Breiman, L. (2001). *Random Forests*. Berkeley: Springer.
- Chanti, D. A., & Caplier, A. (2018). *Improving Bag-of-Visual-Words Towards Effective Facial Expressive*. Grenoble: Improving Bag-of-Visual-Words Towards Effective Facial Expressive Image Classification.
- datagen. (n.d.). *Understanding VGG16: Concepts, Architecture, and Performance* . Retrieved from datagen: <https://datagen.tech/guides/computer-vision/vgg16/>
- Frossard, D. (n.d.). VGG. Retrieved from Papers With Code: <https://paperswithcode.com/method/vgg>
- Fukushima, K. (1980). *Neocognitron: A Self-organizing Neural Network Model*. Setagaya: NHK Broadcasting Science Research Laboratories. Retrieved from <https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf>
- Huilgol, P. (2023). *Top 4 Pre-Trained Models for Image Classification with Python Code*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/>
- IBM. (2022). *What is computer vision?* Retrieved from IBM: <https://www.ibm.com/topics/computer-vision#:~:text=By%202000%2C%20the%20focus%20of,annotated%20emerged%20through%20the%202000s>.
- Islam, K. T., Wijewickrema, S., Raj, R. G., & O'Leary, S. (2019). *Street Sign Recognition Using Histogram of Oriented*. Victoria : Journal of Imaging.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 2278-2324.
- Monien, K., & Decker, R. (2003). Strengths and Weaknesses of Support Vector. In D. Baier, & K.-D. Wernecke, *Innovations in Classification, Data Science, and Information Systems* (pp. 335-362). Brandenburg: Springer.
- Nakata, K., Ng, Y., Miyashita, D., Maki, A., Lin, u.-C., & Deguchi, J. (2022). *Revisiting a kNN-based Image Classification System with High-capacity Storage*. Kawasaki: Kioxia Corporation.
- Sarker, I. H. (2021). *Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions*. Singapore: Springer.
- Shaha, M., & Pawar, M. (2018). Transfer Learning for Image Classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 656-660). Coimbatore: IEEE.
- Simonyan, K., & Zisserman, A. (2015). VERY DEEP CONVOLUTIONAL NETWORKS. *ICLR*. Oxford: University Of Oxford.
- Sun, Y., Li, L., Zheng, L., Hu, J., Jiang, Y., & Yan, C. (2019). *Image Classification base on PCA of Multi-view Deep Representation*. Beijing.
- Zisserman, A. (2017) University of Oxford, Image Classification using Random Forests and Ferns. Available at: <https://www.robots.ox.ac.uk/~vgg/publications/2007/Bosch07a/bosch07a.pdf> (Accessed: 09 June 2023).

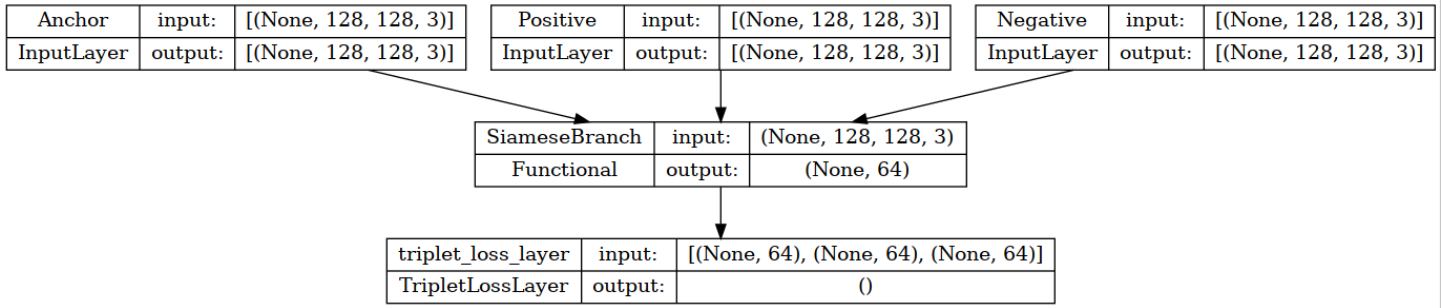
9.0 Appendices

9.1 Method 2 Base Network Topography



9.2 Method 4 Multilayer perceptron implementation

9.2 Method 2 Triplet Network Topography



9.3 Method 1's Classification report

Class	precision	recall	f1-score	support
0	0.83	0.75	0.79	20
1	0.89	0.73	0.80	22
2	0.77	0.74	0.75	27
3	0.38	0.64	0.48	25
4	0.63	0.77	0.69	22
5	0.57	0.70	0.63	23
6	0.80	0.44	0.57	18
7	0.64	0.73	0.68	44
8	0.80	0.63	0.71	19
9	0.60	0.75	0.67	20
10	0.78	0.82	0.80	51
11	0.93	0.74	0.82	38
12	0.52	0.82	0.64	17
13	0.93	0.64	0.76	22
14	0.95	0.88	0.91	24
15	0.44	0.79	0.56	14
16	0.74	0.58	0.65	24
17	0.72	0.78	0.75	23
18	0.79	0.56	0.66	34
19	0.82	0.78	0.80	18
20	0.91	0.78	0.84	41
21	0.86	0.82	0.84	22
22	0.64	0.74	0.68	19
23	0.86	0.75	0.80	24
24	0.82	0.64	0.72	22
25	0.27	0.50	0.35	18
26	0.82	0.75	0.78	24
27	0.59	0.52	0.55	25
28	0.48	0.48	0.48	21
29	0.23	0.24	0.23	21
30	0.76	0.84	0.80	19
31	0.75	0.79	0.77	19
32	0.54	0.56	0.55	25
33	0.61	0.78	0.68	18
34	0.62	0.72	0.67	18
35	0.63	0.75	0.69	16
36	0.94	0.70	0.80	23
37	0.47	0.78	0.58	18
38	0.47	0.53	0.50	15
39	0.73	0.83	0.78	23
40	0.69	0.91	0.78	22
41	0.67	0.63	0.65	19
42	0.79	0.76	0.78	25
43	0.83	0.89	0.86	27
44	0.70	0.78	0.74	18
45	0.69	0.71	0.70	31
46	0.88	0.33	0.48	21
47	0.61	0.55	0.58	20
48	0.75	0.67	0.71	18
49	0.35	0.54	0.42	13
50	0.89	0.55	0.68	29
51	1.00	0.58	0.74	24
52	0.65	0.88	0.75	17
53	0.85	0.77	0.81	22
54	0.79	0.48	0.59	23
55	0.67	0.38	0.48	21
56	0.82	0.74	0.78	19
57	0.67	0.35	0.46	23
58	0.40	0.23	0.29	26
59	0.80	0.36	0.50	11
60	0.47	0.74	0.57	19
61	0.56	0.83	0.67	12
62	0.59	0.81	0.68	21
63	0.64	0.62	0.63	26
64	0.67	0.80	0.73	15
65	0.87	0.93	0.90	14
66	0.46	0.81	0.59	16
67	0.39	0.80	0.52	20
68	0.65	0.75	0.70	20
69	0.69	0.45	0.55	20
70	0.48	0.64	0.55	22
71	0.74	0.76	0.75	38
72	0.74	0.83	0.78	24
73	0.62	0.25	0.36	20
74	0.71	0.63	0.67	19
75	0.82	0.93	0.87	15
76	0.90	1.00	0.95	18
77	1.00	0.50	0.67	18
78	0.73	0.42	0.53	19
79	0.50	0.38	0.43	21
80	0.73	0.80	0.76	20
81	1.00	0.76	0.86	21
82	0.36	0.69	0.47	13
83	0.80	0.89	0.84	9
84	0.83	0.40	0.54	25
85	0.60	0.82	0.69	11
86	0.73	0.73	0.73	15
87	0.69	0.55	0.61	20
88	0.81	0.62	0.70	21
89	0.76	0.62	0.68	45
90	0.29	0.91	0.44	11
91	0.91	0.76	0.83	38
92	0.67	0.77	0.71	26
93	0.59	1.00	0.74	13
94	0.91	0.56	0.69	18
95	0.60	0.75	0.67	56
96	0.80	0.29	0.42	14
97	0.68	0.79	0.73	19
98	1.00	0.15	0.26	20
99	0.90	0.90	0.90	21

class	precision	recall	f1-score	support
100	0.82	0.61	0.70	23
101	0.71	0.71	0.71	21
102	0.86	0.86	0.86	21
103	0.61	0.79	0.69	14
104	0.55	0.56	0.56	57
105	0.55	0.55	0.55	22
106	0.43	0.92	0.59	13
107	0.64	0.44	0.52	16
108	0.81	0.69	0.75	32
109	0.67	0.59	0.62	17
110	1.00	0.70	0.82	20
111	1.00	0.53	0.70	15
112	0.84	0.70	0.76	23
113	0.64	0.90	0.75	20
114	0.71	0.36	0.48	14
115	0.89	0.50	0.64	16
116	0.68	0.58	0.62	26
117	0.74	0.88	0.80	16
118	0.41	0.44	0.42	16
119	0.53	0.68	0.60	25
120	1.00	0.61	0.76	18
121	0.62	0.31	0.41	26
122	0.88	0.94	0.91	16
123	0.86	0.86	0.86	14
124	0.29	0.15	0.20	13
125	0.34	0.58	0.43	55
126	0.81	0.89	0.85	28
127	0.60	0.18	0.27	17
128	1.00	1.00	1.00	31
129	0.88	1.00	0.93	21
130	0.89	0.57	0.70	14
131	0.83	0.76	0.79	38
132	0.82	0.90	0.86	30
133	0.69	0.72	0.71	25
134	0.33	0.50	0.40	22
135	0.74	0.70	0.72	20
136	1.00	1.00	1.00	24
137	0.87	0.73	0.79	37
138	0.64	0.70	0.67	20
139	0.86	0.80	0.83	15
140	0.65	0.80	0.71	25
141	0.71	0.81	0.76	21
142	0.87	0.69	0.77	29
143	0.56	0.50	0.53	10
144	0.99	0.99	0.99	160
145	0.64	0.47	0.55	19
146	0.57	0.53	0.55	32
147	0.69	0.67	0.68	27
148	0.70	0.44	0.54	16
149	0.76	0.65	0.70	20
150	0.80	0.89	0.84	27
151	0.62	0.78	0.69	23
152	0.50	0.43	0.46	14
153	0.91	0.71	0.80	28
154	0.29	0.25	0.27	24
155	0.62	0.65	0.64	23
156	0.86	0.92	0.89	13
157	0.58	0.70	0.64	20
158	0.38	0.43	0.40	44
159	0.83	0.67	0.74	15
160	0.90	0.69	0.78	13
161	0.88	0.30	0.45	23
162	0.30	0.22	0.26	27
163	0.68	0.89	0.77	19
164	0.53	0.45	0.49	22
165	0.34	0.62	0.44	16
166	0.83	0.79	0.81	19
167	1.00	0.79	0.88	24
168	0.68	0.71	0.70	21
169	0.85	0.89	0.87	19
170	0.86	0.35	0.50	17
171	1.00	0.91	0.95	22
172	0.18	0.31	0.23	16
173	0.77	0.50	0.61	20
174	0.83	0.86	0.84	22
175	0.56	0.53	0.54	19
176	0.93	0.93	0.93	15
177	0.72	0.92	0.81	25
178	1.00	0.75	0.86	12
179	0.29	0.30	0.29	20
180	0.65	0.71	0.68	21
181	0.93	0.96	0.95	27
182	0.83	0.56	0.67	18
183	0.80	0.95	0.87	21
184	0.40	0.35	0.38	17
185	0.94	1.00	0.97	16
186	0.70	0.78	0.74	27
187	0.62	0.65	0.63	20
188	0.64	0.76	0.70	21
189	0.82	0.56	0.67	25
190	0.59	0.54	0.57	24
191	0.68	0.77	0.72	22
192	0.74	0.70	0.72	33
193	0.74	0.61	0.67	28
194	0.15	0.17	0.16	12
195	0.53	0.81	0.64	21
196	0.74	0.74	0.74	23
197	0.50	0.64	0.56	22
198	0.38	0.35	0.36	17
199	0.88	0.74	0.80	19
200	0.95	0.90	0.93	21
201	0.65	0.69	0.67	16
202	0.67	0.35	0.46	17
203	0.95	1.00	0.98	20

class	precision	recall	f1-score	support	class	precision	recall	f1-score	support
204	0.73	0.53	0.62	15	230	0.61	0.76	0.68	25
205	0.52	0.74	0.61	23	231	0.67	0.87	0.76	78
206	0.57	0.80	0.67	15	232	0.52	0.68	0.59	25
207	0.83	0.91	0.87	22	233	0.81	0.77	0.79	22
208	0.69	0.44	0.54	25	234	0.72	0.85	0.78	27
209	0.56	0.60	0.58	25	235	0.38	0.50	0.43	16
210	0.57	0.60	0.59	20	236	0.86	0.55	0.67	11
211	0.86	0.71	0.78	35	237	0.78	0.72	0.75	25
212	0.88	0.75	0.81	20	238	0.48	0.63	0.55	19
213	0.79	0.90	0.84	29	239	0.93	0.90	0.92	42
214	0.68	0.79	0.73	19	240	0.94	0.67	0.78	24
215	0.94	0.70	0.80	23	241	0.68	0.68	0.68	22
216	0.56	0.58	0.57	24	242	0.60	0.60	0.60	15
217	0.81	0.68	0.74	19	243	0.43	0.43	0.43	14
218	0.70	0.73	0.71	22	244	0.55	0.64	0.59	25
219	0.67	0.67	0.67	12	245	0.89	0.76	0.82	21
220	0.74	0.82	0.78	17	246	0.56	0.70	0.62	20
221	0.58	0.69	0.63	16	247	0.62	0.72	0.67	18
222	0.68	0.83	0.75	18	248	0.24	0.47	0.31	17
223	0.54	0.81	0.65	16	249	0.82	0.93	0.87	15
224	0.89	0.77	0.83	22	250	1.00	0.97	0.99	137
225	0.54	0.39	0.45	18	251	0.96	1.00	0.98	27
226	0.89	0.61	0.72	28	252	0.93	1.00	0.96	92
227	0.77	0.48	0.59	21	253	0.62	0.53	0.57	19
228	0.61	0.48	0.54	23	254	0.80	0.55	0.65	22
229	0.83	0.95	0.88	20	255	0.70	0.84	0.76	19
					256	0.82	0.65	0.73	170
					precision	recall	f1-score	support	
accuracy							0.70	6112	
macro avg					0.70	0.68	0.67	6112	
weighted avg					0.73	0.70	0.70	6112	

Figure 9 Method 1's Classification Report

9.4 Method 3's Classification report

	precision	recall	f1-score	support
0	1.00	0.17	0.29	18
1	1.00	0.15	0.27	13
2	0.67	0.32	0.43	19
3	0.88	0.37	0.52	19
4	1.00	0.30	0.47	23
5	0.00	0.00	0.00	8
6	0.00	0.00	0.00	10
7	1.00	0.11	0.20	37
8	0.00	0.00	0.00	13
9	1.00	0.20	0.33	10
10	1.00	0.05	0.09	43
11	0.34	0.42	0.38	31
12	1.00	0.06	0.11	18
13	0.00	0.00	0.00	12
14	0.00	0.00	0.00	14
15	1.00	0.56	0.71	9
16	1.00	0.24	0.38	21
17	1.00	0.07	0.12	15
18	1.00	0.36	0.53	22
19	0.25	0.57	0.35	14
20	1.00	0.44	0.61	16
21	1.00	0.20	0.33	20
22	0.00	0.00	0.00	16
23	1.00	0.30	0.46	10
24	1.00	0.13	0.24	15
25	1.00	0.12	0.22	8
26	1.00	0.31	0.47	13
27	0.00	0.00	0.00	21
28	0.00	0.00	0.00	17
29	1.00	0.07	0.13	14
30	1.00	0.20	0.33	10
31	1.00	0.75	0.86	4
32	1.00	0.75	0.86	12
33	0.00	0.00	0.00	11
34	0.00	0.00	0.00	4
35	0.00	0.00	0.00	11
36	1.00	0.46	0.63	13
37	0.00	0.00	0.00	12
38	0.50	0.32	0.39	19
39	1.00	0.17	0.29	12
40	1.00	0.20	0.33	15

41	1.00	0.29	0.44	7
42	1.00	0.50	0.67	10
43	1.00	0.06	0.12	16
44	1.00	0.12	0.22	8
45	1.00	0.33	0.50	18
46	1.00	0.27	0.42	15
47	0.00	0.00	0.00	4
48	1.00	0.06	0.11	17
49	1.00	0.29	0.44	7
50	1.00	0.07	0.13	14
51	1.00	0.12	0.22	8
52	0.00	0.00	0.00	1
53	0.67	0.18	0.29	11
54	1.00	0.12	0.22	8
55	0.00	0.00	0.00	21
56	1.00	0.20	0.33	5
57	1.00	0.27	0.43	11
58	1.00	0.12	0.22	8
59	0.00	0.00	0.00	5
60	1.00	0.46	0.63	13
61	0.00	0.00	0.00	15
62	1.00	0.23	0.38	13
63	1.00	0.06	0.11	18
64	0.00	0.00	0.00	5
65	1.00	0.17	0.29	6
66	1.00	0.12	0.22	8
67	0.00	0.00	0.00	11
68	0.00	0.00	0.00	15
69	1.00	0.22	0.36	9
70	0.00	0.00	0.00	19
71	1.00	0.15	0.27	13
72	0.00	0.00	0.00	11
73	1.00	0.38	0.55	8
74	1.00	0.56	0.71	9
75	1.00	0.18	0.31	11
76	1.00	0.26	0.42	19
77	1.00	0.12	0.22	8
78	0.78	0.88	0.82	16
79	0.00	0.00	0.00	21
80	1.00	0.40	0.57	10
81	1.00	0.33	0.50	6
82	1.00	0.15	0.26	20
83	0.00	0.00	0.00	3
84	0.00	0.00	0.00	18
85	0.00	0.00	0.00	6
86	0.00	0.00	0.00	6
87	1.00	0.40	0.57	15
88	0.00	0.00	0.00	9
89	0.20	0.12	0.15	34
90	1.00	0.16	0.27	19
91	0.00	0.00	0.00	18
92	0.00	0.00	0.00	17
93	1.00	0.27	0.43	11
94	0.00	0.00	0.00	4
95	0.67	0.09	0.15	23
96	1.00	0.21	0.35	14
97	1.00	0.11	0.20	9
98	0.00	0.00	0.00	4
99	1.00	0.07	0.12	15
100	1.00	0.30	0.46	10
101	0.00	0.00	0.00	8
102	1.00	0.05	0.10	19
103	1.00	0.08	0.15	12
104	1.00	0.10	0.19	39
105	0.00	0.00	0.00	19
106	1.00	0.21	0.35	14
107	0.00	0.00	0.00	7
108	1.00	0.07	0.12	30

109	1.00	0.31	0.47	13
110	1.00	0.20	0.33	5
111	1.00	0.07	0.13	14
112	0.00	0.00	0.00	14
113	0.00	0.00	0.00	17
114	1.00	0.07	0.13	14
115	1.00	0.11	0.20	9
116	1.00	0.35	0.52	20
117	0.00	0.00	0.00	8
118	0.00	0.00	0.00	10
119	1.00	0.19	0.32	16
120	1.00	0.11	0.20	9
121	0.00	0.00	0.00	14
122	1.00	0.09	0.17	11
123	0.00	0.00	0.00	25
124	0.00	0.00	0.00	7
125	1.00	0.14	0.24	29
126	1.00	0.17	0.29	12
127	0.00	0.00	0.00	21
128	0.37	0.67	0.48	33
129	1.00	0.26	0.42	19
130	1.00	0.29	0.44	7
131	1.00	0.08	0.14	26
132	1.00	0.16	0.28	31
133	1.00	0.06	0.11	17
134	0.00	0.00	0.00	6
135	1.00	0.13	0.23	31
136	0.71	0.45	0.56	22
137	0.89	0.36	0.52	22
138	1.00	0.39	0.56	31
139	1.00	0.35	0.52	20
140	0.00	0.00	0.00	8
141	1.00	0.60	0.75	10
142	1.00	0.04	0.08	25
143	0.00	0.00	0.00	19
144	0.53	0.85	0.66	95
145	1.00	0.14	0.25	14
146	1.00	0.04	0.07	26
147	0.71	0.23	0.34	22
148	0.67	0.67	0.67	15
149	1.00	0.10	0.18	10
150	1.00	0.17	0.29	18
151	1.00	0.09	0.17	22
152	1.00	0.19	0.32	16
153	0.00	0.00	0.00	6
154	1.00	0.35	0.52	20
155	1.00	0.22	0.36	9
156	0.00	0.00	0.00	6
157	1.00	0.07	0.14	40
158	1.00	0.04	0.08	23
159	1.00	0.08	0.14	13
160	1.00	0.11	0.20	18
161	1.00	0.18	0.31	11
162	1.00	0.17	0.29	18
163	0.00	0.00	0.00	10
164	1.00	0.06	0.12	16
165	0.00	0.00	0.00	8
166	0.00	0.00	0.00	7
167	1.00	0.06	0.11	17
168	0.00	0.00	0.00	3
169	1.00	0.21	0.35	14
170	1.00	0.33	0.50	18
171	0.00	0.00	0.00	4
172	1.00	0.08	0.15	24
173	1.00	0.27	0.43	11
174	1.00	0.71	0.83	7
175	0.00	0.00	0.00	12
176	1.00	0.45	0.62	11

177	0.00	0.00	0.00	8
178	0.00	0.00	0.00	12
179	0.00	0.00	0.00	6
180	1.00	0.14	0.25	7
181	1.00	0.12	0.21	17
182	0.00	0.00	0.00	8
183	1.00	0.16	0.27	19
184	0.80	0.33	0.47	12
185	1.00	0.06	0.12	16
186	1.00	0.05	0.10	20
187	0.00	0.00	0.00	13
188	0.00	0.00	0.00	12
189	1.00	0.14	0.25	14
190	1.00	0.36	0.53	11
191	1.00	0.10	0.17	21
192	0.58	0.35	0.44	20
193	0.00	0.00	0.00	8
194	1.00	0.10	0.18	10
195	1.00	0.47	0.64	19
196	0.00	0.00	0.00	7
197	1.00	0.06	0.11	18
198	0.00	0.00	0.00	15
199	1.00	0.21	0.35	14
200	1.00	0.29	0.44	14
201	1.00	0.57	0.73	7
202	1.00	0.43	0.60	7

203	1.00	0.57	0.73	7
204	1.00	0.14	0.25	14
205	0.00	0.00	0.00	13
206	0.00	0.00	0.00	14
207	1.00	0.28	0.43	18
208	0.00	0.00	0.00	1
209	1.00	0.22	0.36	18
210	1.00	0.40	0.57	5
211	1.00	0.24	0.38	17
212	1.00	0.24	0.38	17
213	1.00	0.08	0.15	12
214	0.00	0.00	0.00	13
215	1.00	0.50	0.67	6
216	1.00	0.05	0.09	21
217	0.00	0.00	0.00	8
218	1.00	0.15	0.27	13
219	1.00	0.20	0.33	5
220	1.00	0.50	0.67	6
221	1.00	0.14	0.25	14
222	1.00	0.07	0.13	14
223	1.00	0.22	0.36	18
224	0.00	0.00	0.00	22
225	1.00	0.18	0.30	17
226	0.53	0.57	0.55	35
227	1.00	0.12	0.21	17
228	1.00	0.10	0.18	20
229	0.86	0.86	0.86	7
230	1.00	0.50	0.67	8
231	0.31	0.75	0.43	55
232	1.00	0.50	0.67	10
233	0.48	0.52	0.50	29
234	1.00	0.29	0.44	14
235	0.00	0.00	0.00	15
236	1.00	0.08	0.14	13
237	0.53	0.38	0.44	21
238	1.00	0.57	0.73	7
239	0.24	0.70	0.35	30
240	0.00	0.00	0.00	7
241	0.00	0.00	0.00	2
242	1.00	0.30	0.46	10

243	1.00	0.10	0.18	10
244	0.00	0.00	0.00	6
245	1.00	0.15	0.27	13
246	1.00	0.13	0.24	15
247	1.00	0.42	0.59	12
248	0.83	0.28	0.42	18
249	0.00	0.00	0.00	13
250	0.11	0.78	0.20	117
251	0.82	0.38	0.51	24
252	0.30	0.94	0.45	49
253	0.00	0.00	0.00	6
254	1.00	0.18	0.31	11
255	0.00	0.00	0.00	18
256	0.05	0.89	0.09	98

accuracy		0.25	4000	
macro avg	0.65	0.19	0.25	4000
weighted avg	0.66	0.25	0.26	4000

9.5 Strengths and Weakness of other models

One of the earliest approaches of image classification approaches was proposed in 1979 and developed in 1980, the Neocognitron which is shown in Figure 10. This approach was developed for recognizing handwriting, using “S-layers” to perform local feature extraction, composing of “S-cells” (Fukushima, 1980). These features would then use G-cells to merge and subsample the S-cells extracted features. In more modern terms this representative of a modern neural convolutional network. However, the largest downfall of this model was the lack of backpropagation in the model. This led to poor efficiency and having to tune many parameters by hand as no labelled data could not be used in the model making the Neocognitron a unsupervised model.

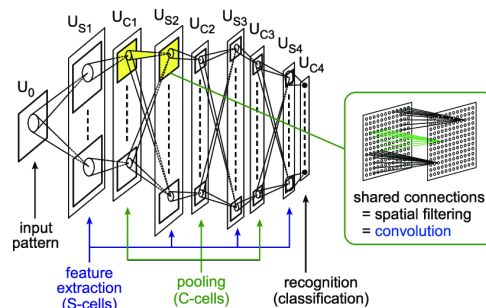


Figure 10 Neocognitron's Proposed Model

So, in 1998 the LeNet model, shown in Figure 11 was proposed which built on the Neocognitron by utilizing backpropagation which developed one of the first supervised models in a convolutional network (LeCun, Bottou, Bengio, & Haffner, 1998). This leveraged labelled data by the network to make informed decisions on if what it's doing is correct. A downfall with this model, like another issue with the Neocognitron, is the limited capacity and scalability of the convolutional network, as well as the high focus on identifying written characters. This is because the model has only around 60,000 trainable. The model also only had 3 convolution layers, two average pooling layers as well as used a SoftMax classifier.

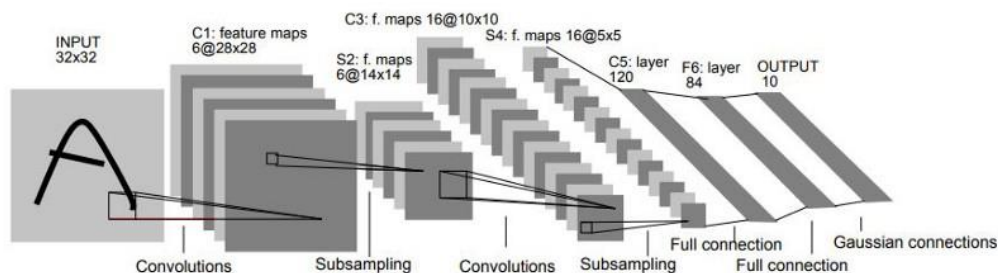


Figure 11 LeNet's Proposed Model

The LeNet was not expanded upon until 2012, in which introduced the AlexNet, shown in Figure 12, was introduced. The AlexNet was a convolutional network, that uses normalization layers which normalized all of the values within a channel. The model also introduced the idea of a rectified linear unit (ReLU) as an activation function (Krizhevsky, Sutskever, & Hinton, 2012). This model contains 60 million parameters, in comparison to LeNet's 60000 parameters, improving the potential adaptability and scalability of the network. The model also features 8 layers which significantly increase in depth in comparison to the LeNet, with 4 of the layers as convolutional layers.

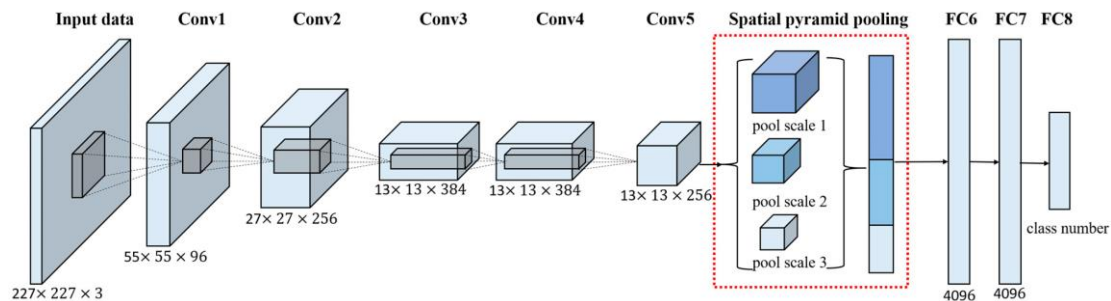


Figure 12 Alex Net's Proposed Model

The Alex Net popularised the use of convolutional networks which led to an explosion of different CNN models being developed. One of the popularised CNN's was the VGG network released in 2014, see Figure 13. The VGG differed from the Alex Net by utilizing a deeper architecture which improved the depth and complexity of the model, allowing for more complex features to be captured. The VGG network also used smaller and more uniform convolutional layers and pooling that allowed the model to interpret more patterns of an image (Simonyan & Zisserman, 2015) (Shaha & Pawar, 2018). This model performs particularly well within the Caltech 256 dataset, able to achieve a 6.8% top-5 validation error. (Simonyan & Zisserman, 2015). The VGG model however is a computationally expensive model (datagen, n.d.).

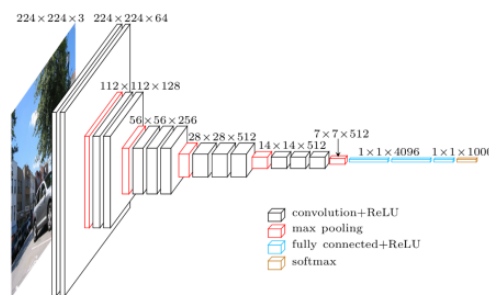


Figure 13 VGG Network's Proposed Model

Note. From "VGG" by Frossard, Davi, n.d, (<https://paperswithcode.com/method/vgg>)

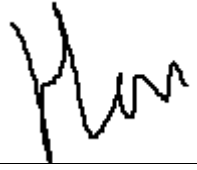
Mobile Net

With a focus on computational efficiency the next model considered was the mobile net V3 developed in 2019. This model reduced the number of necessary layers to achieve good performance. However, the performance of the model was hampered by its limited extraction of features by reducing the necessary layers for the model in comparison to the VGG. This leads to improved training time, but worse in terms of accuracy for the computational efficiency.

Inception Model

This led to the inception model that has a deeper model then the mobile net, but a shallower model in comparison to the VGG network. This network allows a combination of improved accuracy and improved computational time, ultimately being a balanced model between the two.

10.0 Contributions

Student	Description of Contribution	Percentage	Signature
Jake Overs	Presentation: Introduction, motivation, database description, Method 3 Description. Report: Introduction, Related Work, Database Description, Method 3 description and method 3 results.	25%	J.Overs
Ryan Hansen	Presentation: Method 1's sections, Results Report: Method 1's Methodology, Method 1's individual Results, Evaluation and Discussion, Conclusion and final works	25%	Ryan Hansen
Jeonghan Kim	Presentation: Method 3's sections Report: Method 3's Methodology/conclusion	25%	
Hajun Song	Presentation: Method 4's sections Report: Method4's Methodology, Conclusion/future work	25%	Hajun.S