AI and Machine Learning Coursework

Sebastian Lynch

March 13, 2025

1 Task 1 - Image Classification

1.1 Introduction

1.1.1 Machine Learning for Image Classification

Machine learning is the scientific study of algorithms and statistical models used by computer systems to perform a specific task without being explicitly programmed. In this task, we are working on an image classification problem where each image belongs one of 10 classes, each representing objects like parachutes, dogs, and trucks. The goal is to develop a machine learning model that can accurately classify test images into their respective categories. All images are labelled meaning this is a supervised learning task. Image classification has a broad range of real-world applications, including vehicle autonomy and medical diagnosis [1].

1.1.2 Approaches

Our first approach uses a support vector machine (SVM) model. Since the late 1990s, SVMs have been used for image classification, such as classifying handwritten digits in the MNIST dataset [2], face detection [3] and 3D object classification [4]. It works by mapping the training data into a higher-dimensional space and constructing hyperplanes that best separate the data.

In recent years, traditional machine learning techniques like SVMs have been overshadowed by the rise of deep learning models, particularly Convolutional Neural Networks (CNNs). Deep learning is a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.

A CNN is an artificial neural network consisting of several convolutional layers, each preceded by an activation function and a pooling layer. Convolutional layers apply filters to extract important features like edges and textures. The activation function introduces non-linearity, allowing a model to learn complex patterns and make better predictions by transforming the output of each neuron. Pooling layers reduce the dimensions of feature maps by downsampling, helping to retain essential information while reducing computation and overfitting.

CNNs are particularly powerful due to their ability to automatically learn hierarchical features from raw data, without the need for manual feature extraction. Studies have shown that these models have better capabilities than traditional machine learning methods when working with large-scale data [5]. The availability of large labelled datasets and more powerful hardware has increased rapidly throughout the 21st century, leading to CNNs becoming the dominant force in classification models.

We will test the performance of three CNN models on image classification: LeNet, VGG16, and ResNet50. We will then further improve the best performing model by applying data-augmentation techniques and exploring model architecture changes.

1.1.3 Evaluation Protocol

First, we analyse the test accuracy score, which is the proportion of correctly classified test images. While useful, accuracy alone is insufficient for a comprehensive evaluation; therefore, we examine confusion matrices alongside precision, recall, and F1-score for each class. Precision measures correct positive predictions out of all predicted positives, recall measures correct positives out of all actual positives, and the F1-score is their harmonic mean.

Additionally, the training time and model size (number of parameters) are also considered. The training time is the time taken for the model to train using the T4 GPU in Google Colab.

1.2 Support Vector Machine

An SVM classifier is heavily dependent on feature extraction. Feature extraction transforms raw image data into a feature vector. We use the histogram of gradients (HOG) feature descriptor, commonly used in object recognition tasks. HOG is a method for representing and capturing the structure and shape of objects, developed in 2005, originally for human detection [6].

This model has three hyperparameters: C, the regularisation parameter, controls the complexity of the hyperplanes; choice of kernel, a function that transforms data into a higher-dimensional space (linear or RBF); and Gamma, the kernel coefficient, which controls the smoothness of the hyperplanes (irrelevant if the kernel is linear). In order to tune the hyperparameters, we employ a grid search. This is a hyperparameter tuning technique in machine learning that systematically tests multiple combinations of hyperparameters to find the best-performing model. The combinations come from the following hyperparameters: C values: 0.1, 1 or 10; Kernel: either linear or the Radial Basis Function (RBF); Gamma: 'scale', 'auto', 0.01, 0.1 or 1.

The search finds that parameters of C=10, RBF kernel, and gamma = 'auto' produce the best model (by cross-validation accuracy). This model produces a test accuracy score of 54.50% and the confusion matrix is shown in Figure 1. Precision, recall, and F1-scores per class are shown in Figure 2.

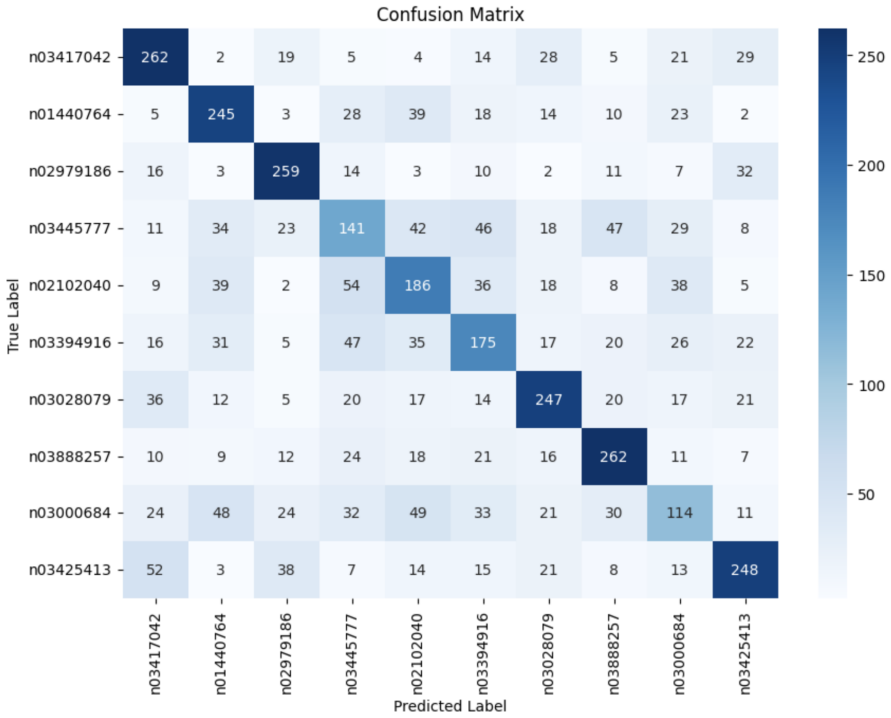

Figure 1: Confusion matrix for the best performing SVM model.

Classification Report:

	precision	recall	f1-score	support
0	0.59	0.67	0.63	389
1	0.58	0.63	0.60	387
2	0.66	0.73	0.69	357
3	0.38	0.35	0.37	399
4	0.46	0.47	0.46	395
5	0.46	0.44	0.45	394
6	0.61	0.60	0.61	409
7	0.62	0.67	0.65	390
8	0.38	0.30	0.33	386
9	0.64	0.59	0.62	419
accuracy			0.54	3925
macro avg	0.54	0.55	0.54	3925
weighted avg	0.54	0.54	0.54	3925

Figure 2: Precision, recall and F1-score for each class of the best performing SVM model.

The confusion matrix and F1-score reveal significant variability in the model's performance across different classes. The highest F1-score is 0.69, while the lowest is 0.33. The confusion matrix shows that no class is being significantly over-predicted or under-predicted. The grid search took 35 minutes and a single training session takes 5-6 minutes.

1.3 LeNet

One of the earliest applications of CNNs to image recognition was the LeNet model. First introduced by Yann LeCun and his colleagues in 1998 [2], LeNet was originally designed to classify handwritten digits from the MNIST dataset. It was designed to work with 28x28 greyscale images, meaning that the image dataset must be preprocessed appropriately. Notably, the resizing and recolouring will certainly lead to a loss of information.

The structure of the LeNet model used is created using TensorFlow and shown in Figure 3.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(1, 28, 28, 6)	156
max_pooling2d (MaxPooling2D)	?	0
conv2d_1 (Conv2D)	(1, 10, 10, 16)	2,416
max_pooling2d_1 (MaxPooling2D)	?	0
conv2d_2 (Conv2D)	(1, 1, 1, 120)	48,120
flatten (Flatten)	(1, 120)	0
dense_8 (Dense)	(1, 84)	10,164
dense_9 (Dense)	(1, 10)	850

Total params: 61,706 (241.04 KB)
Trainable params: 61,706 (241.04 KB)

Figure 3: The structure of the Lenet model.

There are 3 convolutional layers, each increasing in size, progressively learning more complex features, and 2 pooling layers. All 3 of these layers use 5x5 kernels to capture spatial patterns. It ends with 2 fully connected layers to make the final classification decision.

The model's architecture includes several hyperparameters, such as the number of filters, kernel size, and activation function. However, we will keep these unchanged to maintain the integrity of the LeNet model; focusing on the training hyperparameters.

The training hyperparameters are: optimiser (default is Adam), learning rate (implicitly set by Adam), loss function (default is `SparseCategoricalCrossentropy`), batch size, and epochs. The only hyperparameters we will tune are the optimiser (Adam or Stochastic Gradient Descent) and the batch size (16 or 32). This means that we have 4 possible combinations. The best performing model is using the Adam optimiser and a batch size of 32, as evidenced by Figure 4, with a test accuracy of 50.47% after 10 epochs. The confusion matrix for the model is shown in Figure 5.

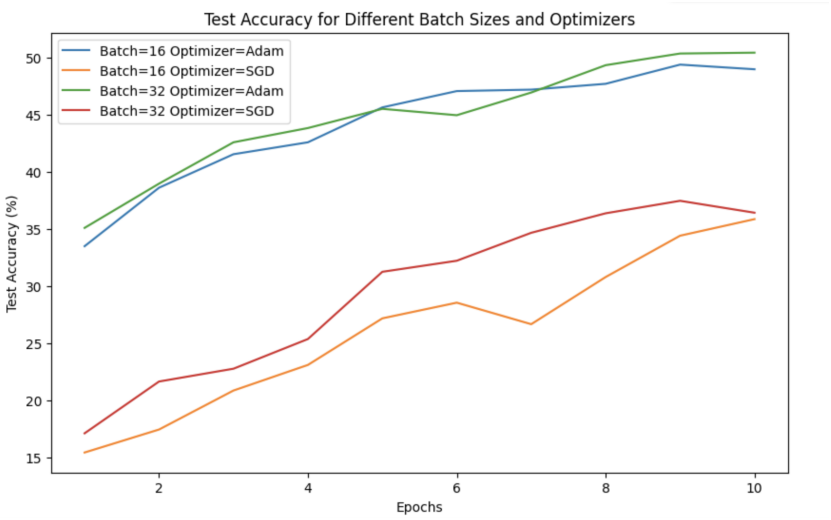

Figure 4: A line plot of test accuracy against epochs for each of the 4 LeNet models that were tested.

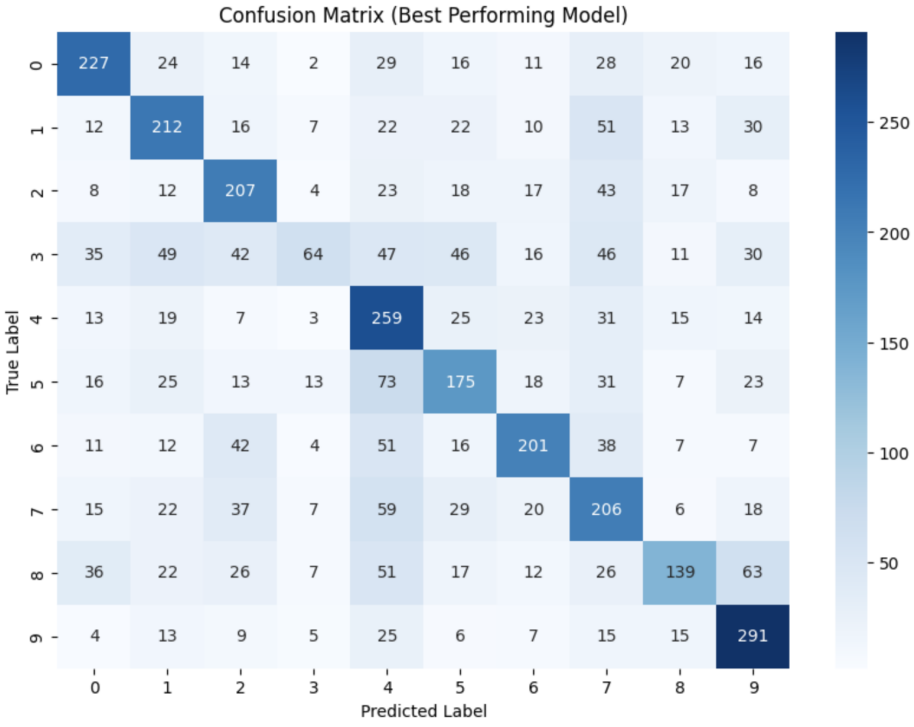

Figure 5: Confusion matrix for the best performing LeNet model.

Precision, recall and F1-scores per class are displayed in Figure 6.

Classification Report for Best Model:

	precision	recall	f1-score	support
0	0.60	0.59	0.59	387
1	0.52	0.54	0.53	395
2	0.50	0.58	0.54	357
3	0.55	0.17	0.25	386
4	0.41	0.63	0.49	409
5	0.47	0.44	0.46	394
6	0.60	0.52	0.56	389
7	0.40	0.49	0.44	419
8	0.56	0.35	0.43	399
9	0.58	0.75	0.65	390
accuracy			0.50	3925
macro avg	0.52	0.50	0.49	3925
weighted avg	0.52	0.50	0.49	3925

Figure 6: Precision, recall and F1-scores of the Lenet model.

The test accuracy for the best performing model on the 10th epoch is worse than the SVM model's. The confusion matrix demonstrates a large imbalance in the distribution of predicted classes; the one class is only predicted 116 times, whereas another is predicted 639 times. The number of true instances is close to 400 for all classes. One LeNet model takes 10 minutes to train.

1.4 VGG16

To overcome the constraints which caused the LeNet model's shortcomings, we need a model that supports RGB images and higher resolutions. In 2014, the Visual Geometry Group (VGG) from the University of Oxford published a paper [7] introducing VGG16; a 'Very Deep Convolutional Network'. The 16 refers to the number of weight layers (13 convolutional and 3 fully connected). The model uses 3x3 convolution filters, whereas prior models such as LeNet had tended to use 5x5 or 7x7 filters. The model expects a 224x224 image with RGB channels.

VGG16 is pretrained (on ImageNet data) and available in TensorFlow Keras. As it is pretrained, we use transfer learning. In transfer learning, only the fully connected layers at the end are trained, while the convolutional base stays frozen. The general structure is shown in Figure 7.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 128)	65,664
dense_1 (Dense)	(None, 10)	1,290

Total params: 14,781,642 (56.39 MB)
Trainable params: 66,954 (261.54 KB)
Non-trainable params: 14,714,688 (56.13 MB)

Figure 7: Structure of our VGG model.

We use the same hyperparameters used in the LeNet model: Adam optimiser, 10 epochs, and a batch size of 32. This model achieves a test accuracy of 94.88% after 10 epochs. The confusion matrix and F1-scores per class further evidence the strong performance, shown in Figures 8 and 9 respectively.

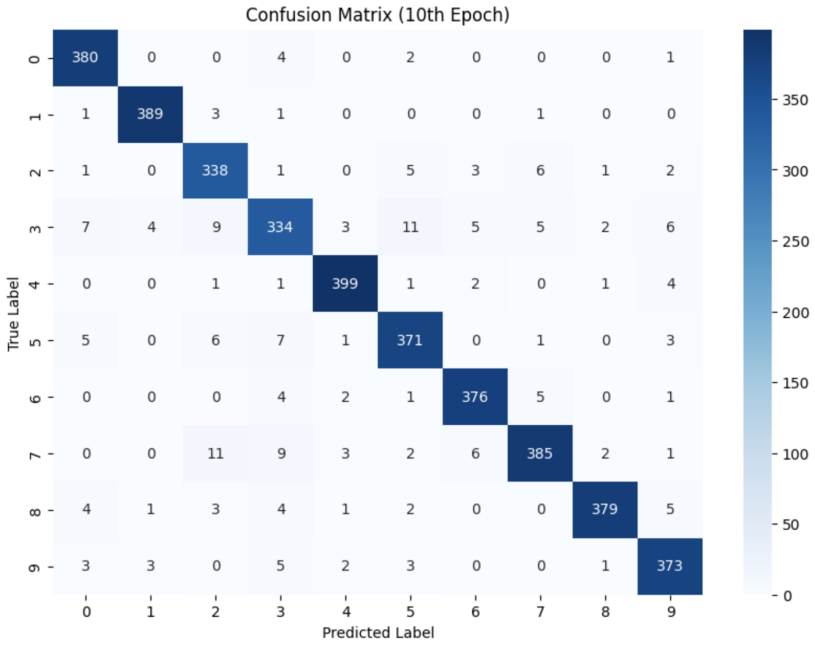

Figure 8: Confusion matrix of the VGG model.

	precision	recall	f1-score	support
0	0.95	0.98	0.96	387
1	0.98	0.98	0.98	395
2	0.91	0.95	0.93	357
3	0.90	0.87	0.88	386
4	0.97	0.98	0.97	409
5	0.93	0.94	0.94	394
6	0.96	0.97	0.96	389
7	0.96	0.92	0.94	419
8	0.98	0.95	0.97	399
9	0.94	0.96	0.95	390
accuracy			0.95	3925
macro avg	0.95	0.95	0.95	3925
weighted avg	0.95	0.95	0.95	3925

Figure 9: Precision, recall, and F1-scores of the VGG model.

A line plot of test accuracy over epochs is shown in Figure 10. The training took 15 minutes.

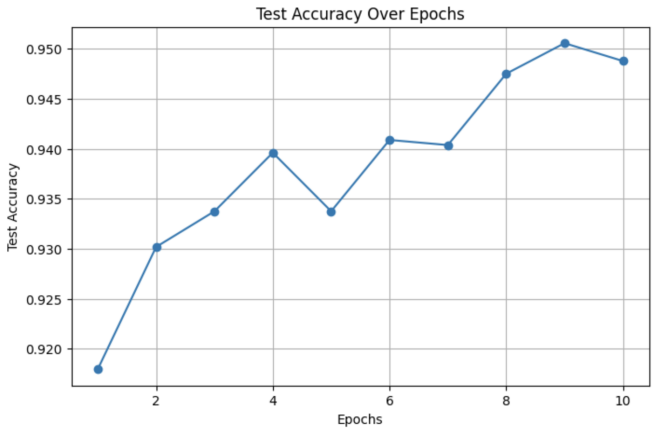

Figure 10: Test accuracy of the VGG model after each epoch.

1.5 ResNet50

Another model to use for transfer learning is ResNet50, first introduced in 2015, in a paper [8] published by researchers at Microsoft. It has 50 convolutional layers, making it much deeper than VGG16. This model introduces skip connections, allowing inputs to bypass intermediate layers and be added to later layers.

ResNet50 expects pixel values to be normalised using the mean and standard deviation of ImageNet; this is done using a predefined TensorFlow Keras function. The general structure is shown in Figure 11.

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 4, 4, 2048)	23,587,712
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None, 2048)	0
dense_12 (Dense)	(None, 128)	262,272
dense_13 (Dense)	(None, 10)	1,290

Total params: 23,851,274 (90.99 MB)
Trainable params: 263,562 (1.01 MB)
Non-trainable params: 23,587,712 (89.98 MB)

Figure 11: Structure of the ResNet model.

The performance of the ResNet model is remarkably strong. After 10 epochs, the test accuracy is 99.39%. The confusion matrix and F1-scores, shown in Figures 12 and 13 further evidence the strong performance.

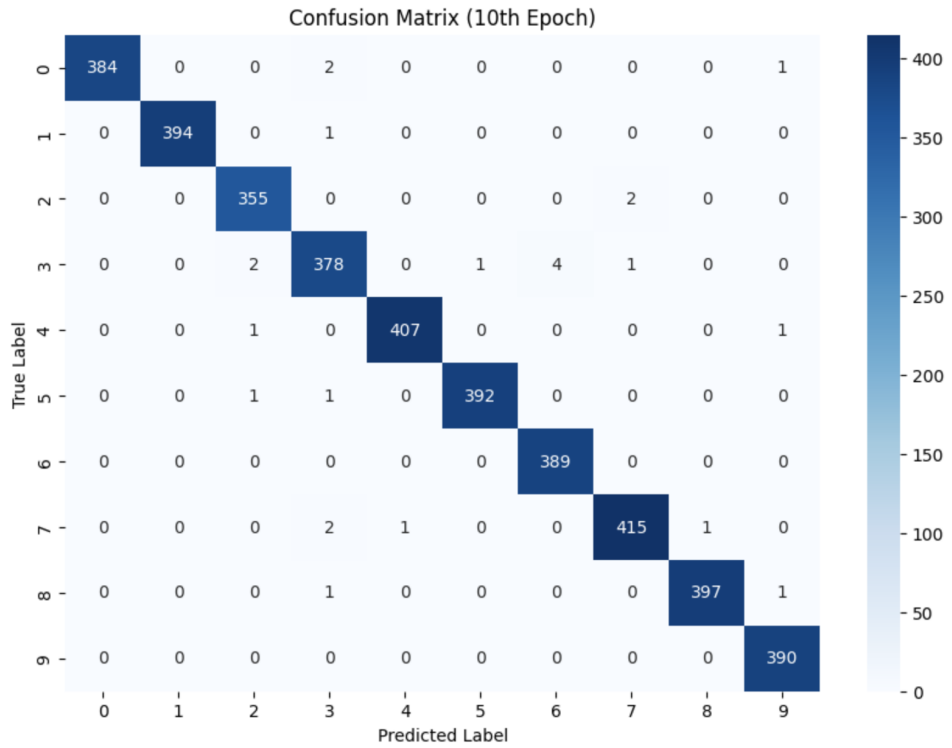

Figure 12: Confusion matrix of the ResNet model.

	precision	recall	f1-score	support
0	1.00	0.99	1.00	387
1	1.00	1.00	1.00	395
2	0.99	0.99	0.99	357
3	0.98	0.98	0.98	386
4	1.00	1.00	1.00	409
5	1.00	0.99	1.00	394
6	0.99	1.00	0.99	389
7	0.99	0.99	0.99	419
8	1.00	0.99	1.00	399
9	0.99	1.00	1.00	390
accuracy			0.99	3925
macro avg	0.99	0.99	0.99	3925
weighted avg	0.99	0.99	0.99	3925

Figure 13: Precision, recall, and F1-scores of the ResNet model.

A line graph of test accuracy after each epoch is shown in Figure 14. The model took 15 minutes to train.

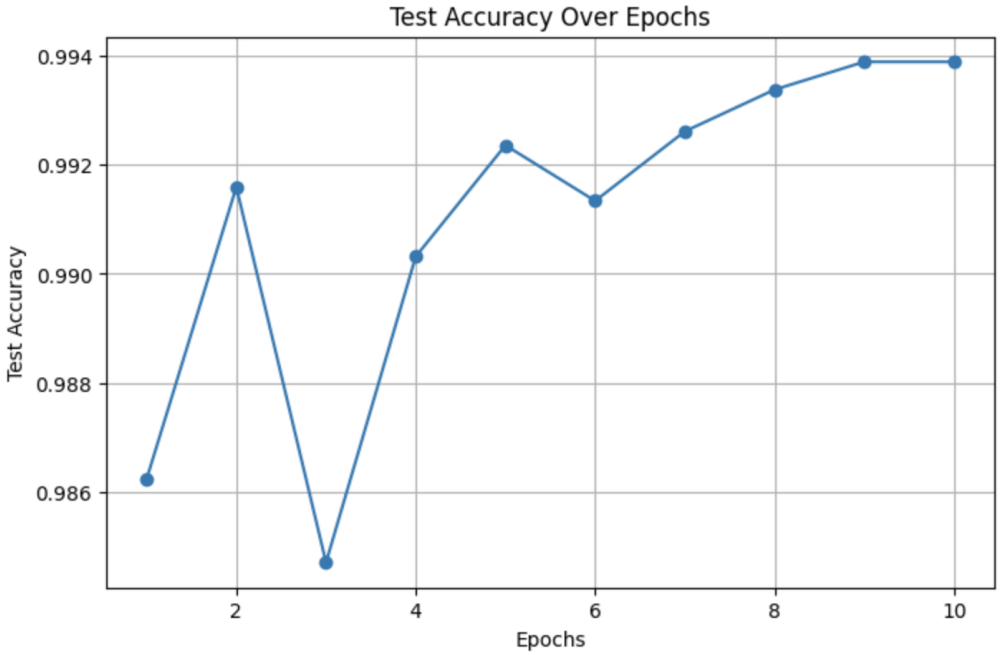

Figure 14: Test accuracy of the ResNet model after each epoch.

1.6 Model Selection for Further Adjustments

Of the 4 models that we have tested so far, our choice of model for data augmentation and model architecture changes is between VGG16 and ResNet50, as these models massively outperformed SVM and LeNet by all metrics. Both models took a similar amount of time to train, with ResNet recording a higher test accuracy of 99.39%, versus 94.88% from VGG. The ResNet model's accuracy is so high that the effect of data augmentation and model architecture changes may not be clear. Since the VGG model has more room for improvement, we will use this model.

1.7 Data Augmentation

Data augmentation is a technique which artificially expands the size and diversity of a training dataset. It involves creating modified versions of the original data by applying transformations such as rotation, flipping (horizontal/vertical), and zooming.

Applying rotation, horizontal flipping, zooming, and brightness adjustments to the training data and retraining the model, we find that the test accuracy score decreased from 94.88% to 93.91%, showing that data augmentation has not had a positive impact. The confusion matrix and F1-scores are shown in Figures 15 and 16.

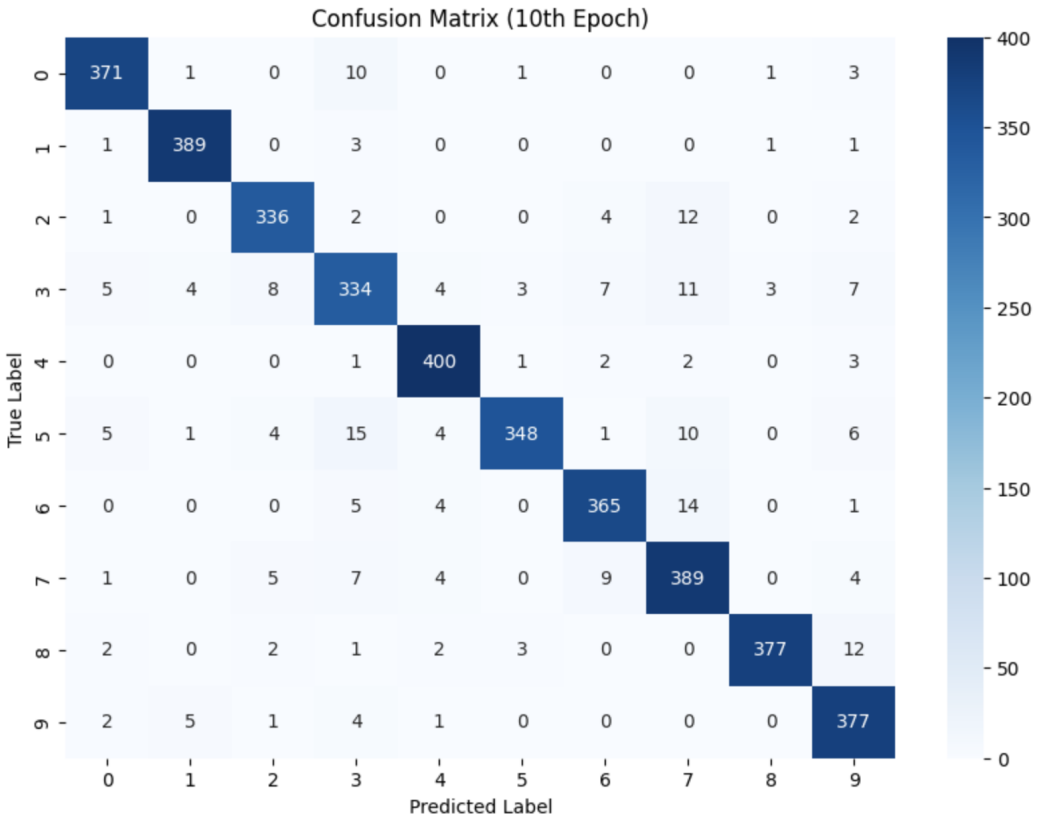

Figure 15: Confusion matrix of the VGG model with data augmentation.

	precision	recall	f1-score	support
0	0.96	0.96	0.96	387
1	0.97	0.98	0.98	395
2	0.94	0.94	0.94	357
3	0.87	0.87	0.87	386
4	0.95	0.98	0.97	409
5	0.98	0.88	0.93	394
6	0.94	0.94	0.94	389
7	0.89	0.93	0.91	419
8	0.99	0.94	0.97	399
9	0.91	0.97	0.94	390
accuracy			0.94	3925
macro avg	0.94	0.94	0.94	3925
weighted avg	0.94	0.94	0.94	3925

Figure 16: Precision, recall, and F1-scores of the VGG model with data augmentation.

1.8 Model Architecture Changes

Firstly, we can adjust the structure of the fully connected dense layers. The final layer should remain as 10 neurons to reflect the number of classes. In our original VGG model, we used 128 neurons in the penultimate layer. We will now test a new structure for the fully connected dense layers of 512, 256, and 10 neurons, as shown in Figure 17.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 512)	0
dense_5 (Dense)	(None, 512)	262,656
dense_6 (Dense)	(None, 256)	131,328
dense_7 (Dense)	(None, 10)	2,570

Figure 17: Structure of the new VGG model.

The performance of this model is again weaker than our original VGG model, with a test accuracy of 93.78%. The confusion matrix and F1-scores are shown in Figures 18 and 19.

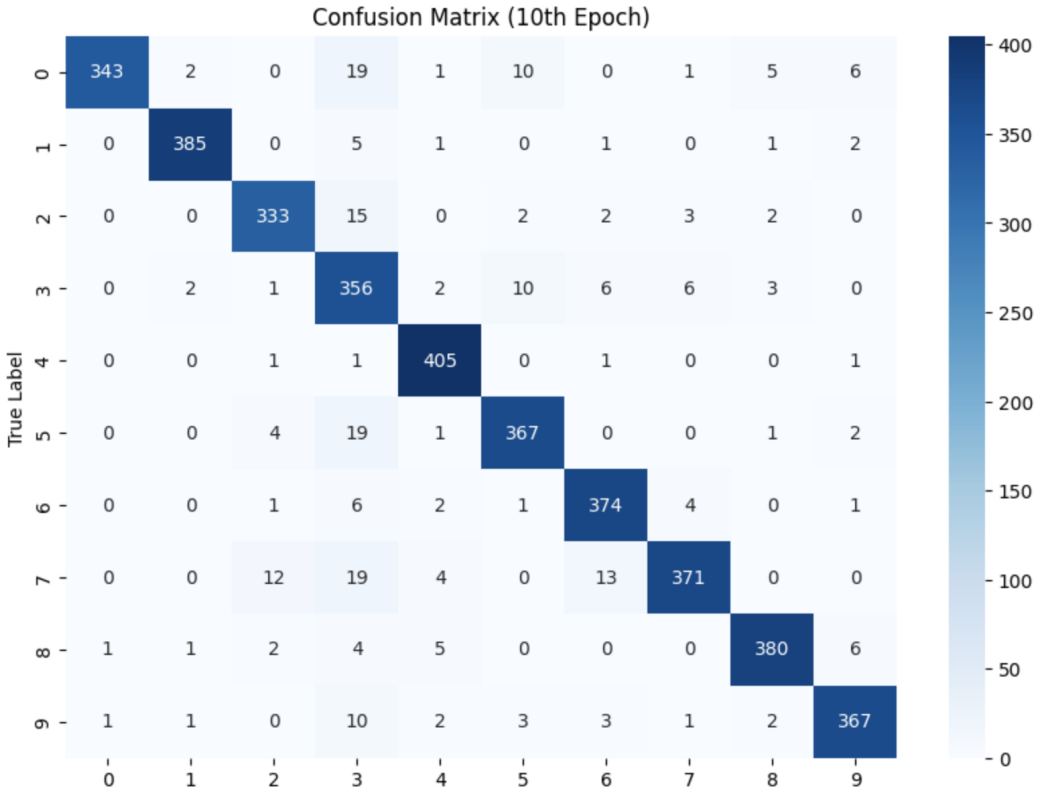

Figure 18: Confusion matrix of the VGG model with the new structure.

	precision	recall	f1-score	support
0	0.99	0.89	0.94	387
1	0.98	0.97	0.98	395
2	0.94	0.93	0.94	357
3	0.78	0.92	0.85	386
4	0.96	0.99	0.97	409
5	0.93	0.93	0.93	394
6	0.94	0.96	0.95	389
7	0.96	0.89	0.92	419
8	0.96	0.95	0.96	399
9	0.95	0.94	0.95	390
accuracy			0.94	3925
macro avg	0.94	0.94	0.94	3925
weighted avg	0.94	0.94	0.94	3925

Figure 19: Precision, recall, and F1-scores of the VGG model with the new structure.

Another change we can try is to retrain some of the layers inside of the pretrained VGG model. In our previous models, these layers have remained frozen. VGG16 consists of 13 convolutional, 5 pooling, and 1 input layers. We keep the first 10 layers frozen but allow the other layers to be trained.

The performance of this model is marginally stronger than our original VGG model, with a test accuracy of 95.13%, showing that this structural change has had a small positive impact. The confusion matrix and F1-scores are shown in Figures 20 and 21.

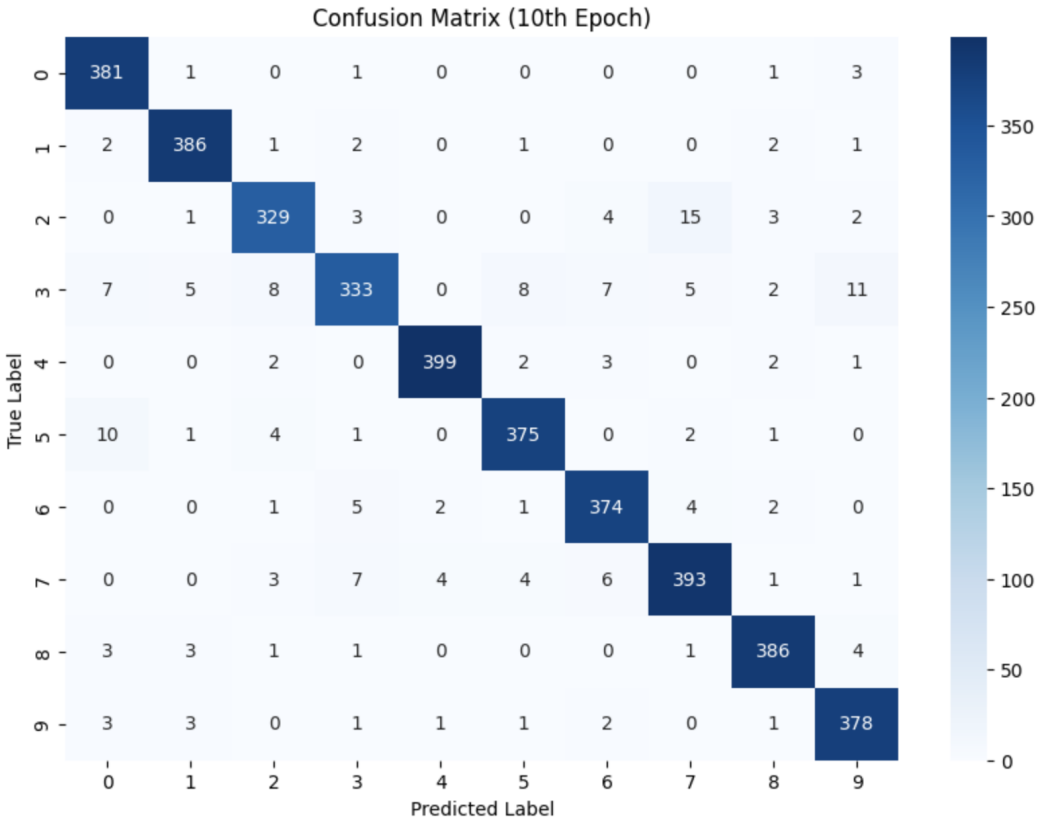

Figure 20: Confusion matrix of the VGG model with some layers no longer frozen.

	precision	recall	f1-score	support
0	0.94	0.98	0.96	387
1	0.96	0.98	0.97	395
2	0.94	0.92	0.93	357
3	0.94	0.86	0.90	386
4	0.98	0.98	0.98	409
5	0.96	0.95	0.95	394
6	0.94	0.96	0.95	389
7	0.94	0.94	0.94	419
8	0.96	0.97	0.96	399
9	0.94	0.97	0.96	390
accuracy			0.95	3925
macro avg	0.95	0.95	0.95	3925
weighted avg	0.95	0.95	0.95	3925

Figure 21: Precision, recall, and F1-scores of the VGG model with some layers no longer frozen.

1.9 Conclusions

For SVM, while a test accuracy of 54.5% may not be exceptional, it suggests that the model is learning and capable of making predictions. The LeNet model was inherently limited due to its relatively low number of parameters and its reliance on 28x28 greyscale inputs, restricting its capacity to learn complex features and leading to a test accuracy of only 49%.

The ResNet50 model used with transfer learning produced highly impressive results, classifying 3901/3925 testing images correctly. The data augmentation and model architecture changes to the VGG model had no or very little positive impact, perhaps because the VGG model already generalises very well and is highly tuned for feature extraction. Data augmentation and architecture changes might have been more useful when training a model from scratch, such as our LeNet model.

1.10 Future Work

With more time, it would have been interesting to apply further tuning of hyperparameters, data augmentation, and model architecture changes to the ResNet50 model, which has also been shown to have stronger performance than VGG16 in other studies [9]. It also would have been interesting to tune hyperparameters for the VGG model using grid search or random search, but the code would have had a very long runtime.

References

[1] Hiroshi Fujita. "AI-based computer-aided diagnosis (AI-CAD): the latest review to read first". In: *Radiological physics and technology* 13.1 (2020), pp. 6–19.

[2] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[3] Edgar Osuna, Robert Freund, and Federico Girosit. "Training support vector machines: an application to face detection". In: *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE. 1997, pp. 130–136.

[4] Massimiliano Pontil and Alessandro Verri. "Support vector machines for 3D object recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 20.6 (1998), pp. 637–646.

[5] Pin Wang, En Fan, and Peng Wang. "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning". In: *Pattern recognition letters* 141 (2021), pp. 61–67.

[6] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[7] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[8] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Sheldon Mascarenhas and Mukul Agarwal. "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification". In: *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*. Vol. 1. 2021, pp. 96–99. DOI: 10.1109/CENTCON52345.2021.9687944.