

# **ECS 795P – Deep Learning and Computer Vision**

## **Deeper Networks for Image Classification**

### **1. Introduction**

Deep convolutional neural networks are commonly used for image classification problems as they have been found to produce high accuracies [1] [2] [3]. Many studies have compared models and found that network depth is an important factor for learning features [4] [5], and therefore improving classification results. However, the question of whether adding more layers will lead to improved models, had led to further problems, such as overfitting, the problem of vanishing gradients [6] [7], and the fact that deeper networks require more parameters, and therefore have more memory and computational requirements, in particular during training. As a result, it is important to consider the trade-off between accuracy and computation.

In this assignment, some of the most popular image classification networks are considered and compared on common databases.

### **2. Related Work**

The ImageNet dataset [8], containing 1.2 million images belonging to 1000 different classes, is commonly used to compare image classification networks. In 2012, Krizhevsky et al. [1] proposed the AlexNet architecture, which was one of the first deep convolutional neural networks to significantly improve accuracy on the ImageNet dataset. This network used a ReLU activation function which helped overcome the vanishing gradient problem, and used drop-out layers to address the problem of over-fitting.

After AlexNet, studies generally looked at trying to improve upon the architecture by restructuring processing units and designing new blocks [2] [3] [9]. Zeiler and Fergus looked at improving feature extraction by using low spatial resolution. This was also used by the VGGNet architecture proposed by

Simonyan et al. [4], which looked at increasing depth by replacing the large kernel-sized filters used in AlexNet (11x11, 5x5) with multiple fixed size kernels of size 3x3. This allowed the depth of the network to be increased so that more complex features could be learned, but less parameters were used so the cost was reduced. Still, VGG has a large computational requirement due to the large width of convolutional layers, and this also makes it slow to train.

A new trend in deep CNN architecture was the shift to modular uniform layer design to make it easier for CNNs to be built for different tasks easily. The GoogLeNet architecture [5] uses an inception module that approximates a sparse CNN with a normal dense construction. This is based on the knowledge that, as most of the activations in a deep network either have a value of zero, or redundant due to correlations between them, not all of the output channels will have a connection with the input channels. Therefore, not all neurons are needed, and so the inception module helps to reduce the kernel size and therefore limit computational requirements. The Inception module also uses wider kernels, implementing multiple kernels of different sizes within the same layer to recognise features of different sizes.

The development of generic blocks that can be used in CNN architectures has been investigated to improve network representations at any stage [10], and have been used for applications such as assigning attention to spatial and channel information [11] [12] [13] [14]. The idea of channel boosting was also explored, which uses transfer learning to boost channels [15].

Several studies have looked at developing new connections to improve the convergence of deep CNN architecture, such as information gating mechanism across multiple layers, skip connections, and cross-layer channel

connectivity [16] [17] [18]. Kaimang et al. proposed ResNet [19] which addresses the vanishing gradient problem by using short-cut connections. ResNet uses Residual blocks, and learns the identity mappings between these, and this allows networks to be training deeper without exploding gradients. This also addresses the degradation problem in which accuracy gets saturated and then begins to degrade.

One of the biggest challenges for image classification with deep CNN architectures is their applications to real world tasks, in which timing and computational constraints become an issue. Some attempt to address this include knowledge distillation, small network training and squeezing of pretrained networks [20] [21] [22] [23]. GoogLeNet also address this issue by replacing conventional convolution in the inception module to reduce kernel size, and ShuffleNet look at a point-wise group convolution to reduce the computation [24].

### 3. Method

#### 3.1. *Model Architectures*

For this assignment, three common image classification networks were trained and evaluated with several different datasets. The models were trained for each different dataset in the same way, with a learning rate of 0.0002, a batch size of 100, and no data augmentation.

The number of epochs during training varied for different datasets. The models were trained using Google Colab GPUs.

Three models were trained: VGG-16, ResNet-50 and Inception-V3. The full model architectures can be seen in Appendix C. All of the models were loaded as a Pytorch model, and an extra fully connected layer was added with the number of outputs correlating to the number of classes for each dataset.

#### a) **VGG-16**

VGG16 is a 16-layer variant of the VGGNet proposed by Simonyan and Zisserman in their paper ‘Very Deep Convolutional Neural Networks for Large-Scale Image Recognition’ [4]. The VGG-16 model uses increased depth to learn more complex features in order to improve classification accuracy.

During training, it was found to have large weights, and therefore it was quite slow to train.

#### b) **Inception-V3**

Inception-V3 is a variant of the GoogLeNet Inception module architecture, proposed by Szegedy et al. in their paper ‘Going deeper with Convolutions’ [5]. The aim of the network is to act as a multi-level feature extractor due to variable kernel sizes within the same module. The Inception-V3 architecture comes from the later paper ‘Rethinking the Inception Architecture for Computer Vision’ [25].

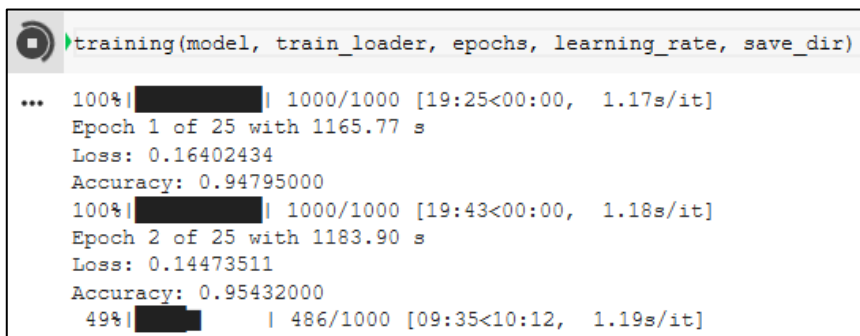
This model was found to have the smallest weights, and trained quickly.

#### c) **ResNet-50**

ResNet-50 is a 50 layer variant of ResNet , proposed by He et al. in their paper ‘Deep Residual Learning for Image Recognition’ [19]. The ResNet-50 architecture uses residual modules to overcome the vanishing gradient problem and create an extremely deep network.

During training this model was generally found to train the quickest, and the model weights were found to be much smaller than VGG-16.

**Figure 1:** Example Output from Model Code Training



```

training(model, train_loader, epochs, learning_rate, save_dir)
... 100%|██████████| 1000/1000 [19:25<00:00, 1.17s/it]
Epoch 1 of 25 with 1165.77 s
Loss: 0.16402434
Accuracy: 0.94795000
100%|██████████| 1000/1000 [19:43<00:00, 1.18s/it]
Epoch 2 of 25 with 1183.90 s
Loss: 0.14473511
Accuracy: 0.95432000
49%|██████████| 486/1000 [09:35<10:12, 1.19s/it]

```

### 3.2. *Datasets*

The models were trained and tested on three different datasets: MNIST, CIFAR-10 and Tiny-ImageNet.

#### a) **MNIST**

The MNIST (Modified National Institute of Standards and Technology) dataset is a collection of handwritten digits [26]. The data is all monochrome, normalised and centred in a 28x28 size image.

For this dataset, the models were trained for 6 epochs.

#### b) **CIFAR-10**

The CIFAR-10 (Canadian Institute for Advanced Research, 10 classes) dataset [27]

contains 60,000 colour images of size 32x32. The images are labelled with one of 10 mutually exclusive classes:

- |               |          |
|---------------|----------|
| 0. aeroplane  | 5. dog   |
| 1. automobile | 6. frog  |
| 2. bird       | 7. horse |
| 3. cat        | 8. ship  |
| 4. deer       | 9. truck |

For this dataset, the models were trained for 25 epochs

#### c) **Tiny ImageNet**

The Tiny-ImageNet dataset [28] is a subset of the ImageNet dataset, containing 100,000 images with 200 classes (500 images per class). Each image is coloured and of size 64x64.

For this dataset, the models were trained for 30 epochs.

### 3.3. *Data Preparation*

For input into the models, the data was pre-processed using the Pytorch 'torchvision.transforms' module. The input images were resized to the required model input size (224 for VGG and ResNet, 299 for Inception), and in the case of the greyscale MNIST data, the number of channels were expanded to 3. The data was then converted to a Pytorch tensor type, and normalised.

## 4. **Experimental Results**

### 4.1. *Evaluation Methods*

The model accuracies were scored by the 'top-1 accuracy', and also by the 'top-5 accuracy'. The 'top-1 accuracy' was defined as the number of correctly classified data items divided by the total number of items. The 'top-5 accuracy' was defined in the same way,

**Table 1:** Validation Set Results for different model architectures on multiple datasets

			Model		
			Vgg16	InceptionV3	ResNet50
Dataset Accuracy	MNIST	Top-1 Accuracy	0.9916	0.9923	0.9923
		Top-5 Accuracy	1.0000	0.9999	0.9999
	CIFAR	Top-1 Accuracy	0.7878	0.8141	0.8081
		Top-5 Accuracy	0.9810	0.9881	0.9899
	Tiny-ImageNet	Top-1 Accuracy	0.2458	0.3669	0.2906
		Top-5 Accuracy	0.4829	0.6201	0.5557

except the data item is said to be correctly classified is the correct class is within the top five predicted classes.

#### 4.2. Validation Set Results

The validation sets of each dataset were evaluated on the trained models and their top-1 and top-5 accuracies were recorded in Table 1. It should be noted that due to GPU limitations, the models training on the Tiny-ImageNet datasets were not able to be trained fully, and so generally achieved poor results, although this is also partly due to the more difficult classification task.

From these results it can be seen that generally the Inception-V3 model produced the highest accuracy values, closely followed by ResNet 50. VGG-16 scored lowest.

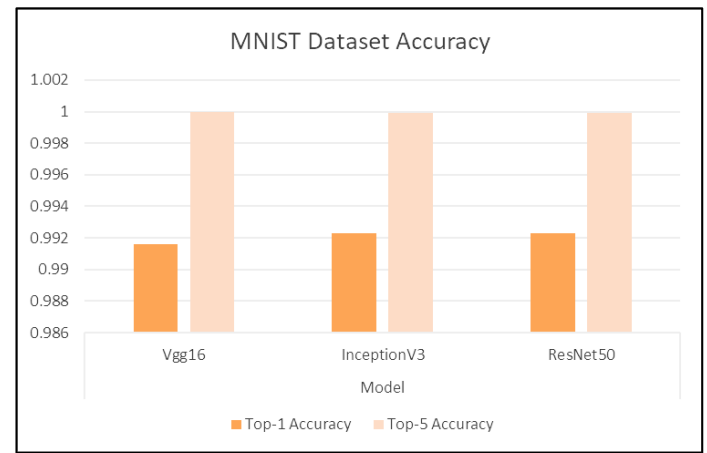
##### 4.2.1. MNIST Results

The MNIST dataset validation set results are plotted in Figure 2 for easier visualisation. It can be observed that the true class was in the top-5 predicted classes nearly 100% of the time, and that all models achieved a very high accuracy, with Inception and ResNet scoring equally highest.

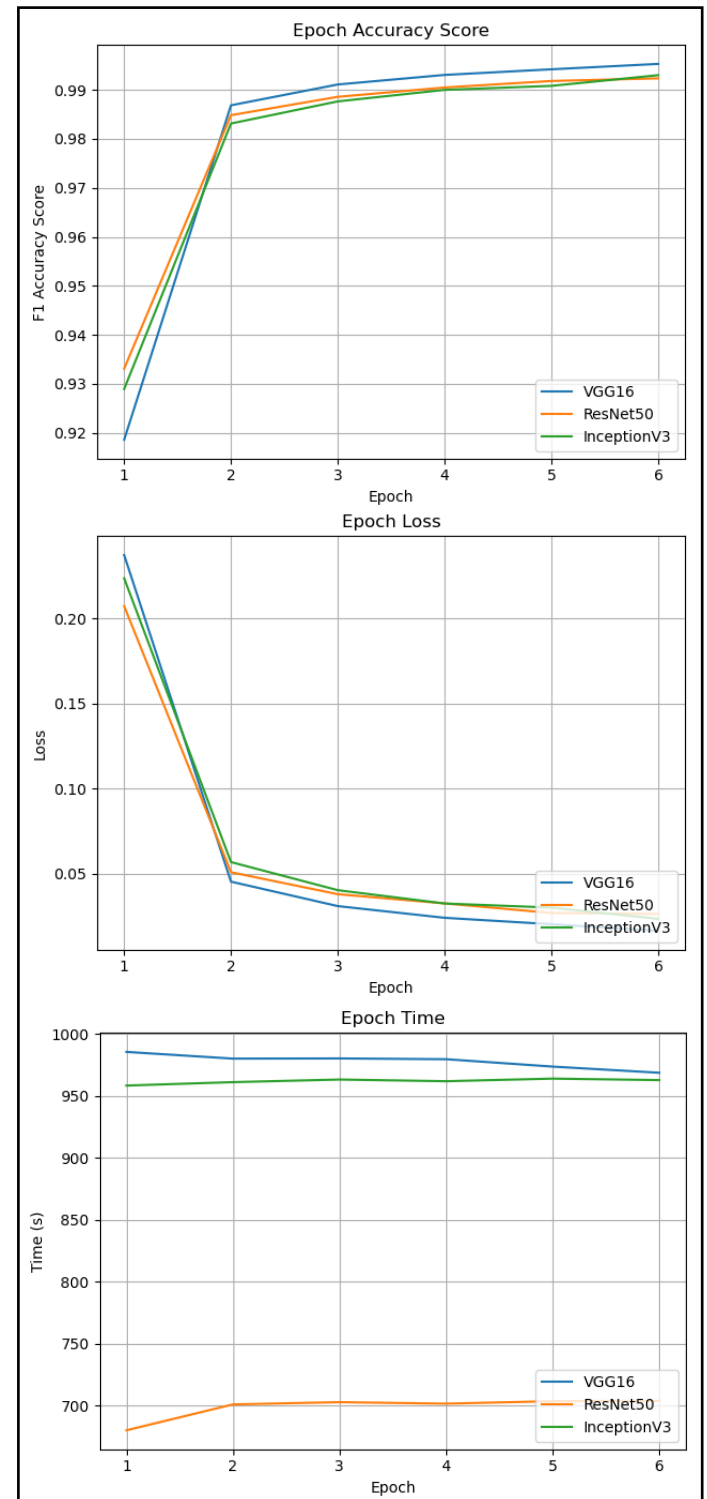
The accuracy, loss and epoch time during training was recorded and can be seen in Figure 3. From these graphs it can be observed that VGG achieved the highest training accuracy and lowest training loss, followed by ResNet and then Inception. ResNet was much faster at training than the other two models. VGG was slowest.

Looking at the confusion matrixes produced for the MNIST dataset results for each model (Appendix Aa), it can be seen that all the models performed very well on this dataset and there wasn't really any noticeable class confusion for any of the models.

In Appendix Ba, some of the misclassified images are shown with their incorrect prediction. From these results, it can be seen that the misclassified results generally occur where the input image is quite difficult even for a human to identify as the numbers are written poorly. Therefore, it seems likely that a human would be unable to classify this dataset much better than any of the models.



**Figure 2:** MNIST validation set results for models



**Figure 3:** CIFAR-10 training accuracy, loss and epoch time

#### 4.2.2. *CIFAR-10 Results*

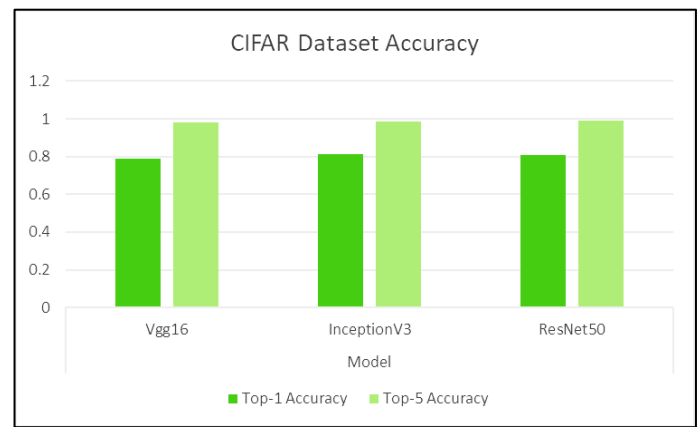
The CIFAR-10 dataset validation set results were plotted in Figure 4. It can be seen that Inception achieved the highest top-1 accuracy, and that VGG achieved the worst. VGG also achieved the worse top-5 accuracy, but ResNet achieved the highest top-5 accuracy.

From the training accuracy, loss and epoch time plots in Figure 5, it can be seen that again the VGG model achieved the highest training accuracy and lowest training loss, followed by ResNet and then Inception. Again, ResNet was the fastest model to train, and VGG was the slowest.

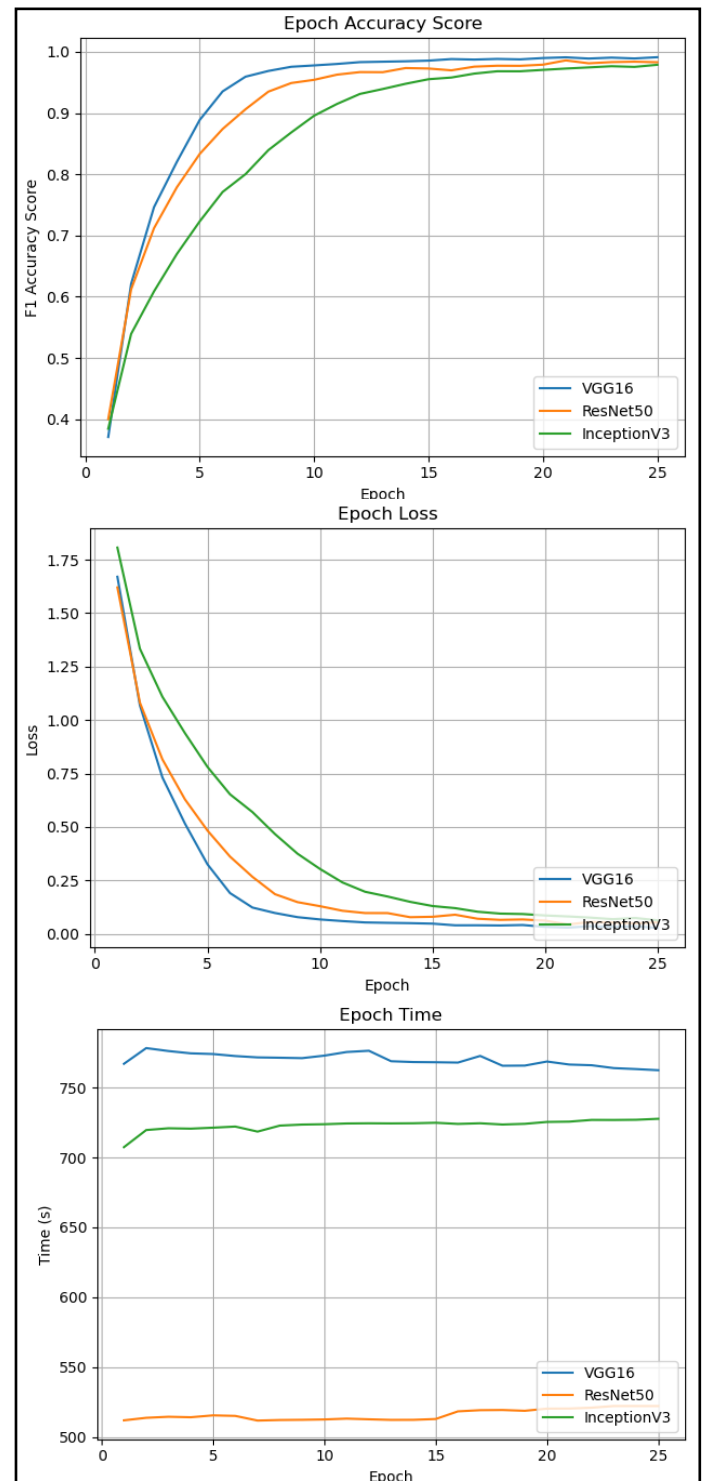
Looking at the confusion matrixes produced for the CIFAR-10 dataset results for each model (Appendix Ab), it can be seen that all the models performed fairly well on this dataset but there was more class confusion here than there was for the MNIST dataset. The VGG-16 and Inception-V3 models behaved very similarly, with only a small amount of class confusion. These models only noticeably seemed to struggle with identifying between classes 3 (cat) and 5 (dog) which seems understandable as they both may contain similar features, especially for the dataset's low resolution images. In the case of the VGG-16 model this may also be that more training is required as class 3 (cat) only achieved a 53% accuracy which is much lower than the other classes.

The ResNet-50 model however behaved quite differently. For most classes there was no confusion at all, but where there was difficulty in identifying between classes, the confusion was greater. In general confusion was not between classes, but certain classes experienced more confusion than others. Classes that were commonly misclassified were 3 (cat), five (dog), and six (frog). Class 1 (aeroplane) was often misclassified as 9 (truck), class 3 (cat) was commonly misclassified as 5 (dog), and class 6 (frog) was commonly misclassified as 4 (deer). Classes 1 (aeroplane) and 3 (cat) also only achieved a 50% accuracy so maybe more training would help.

In general, from these results from the ResNet-50 confusion matrix, it can be seen that there is often a relationship between class confusion,



**Figure 4:** CIFAR-10 validation set results for models



**Figure 5:** CIFAR-10 training accuracy, loss and epoch time

e.g. aeroplane and trucks are both vehicles, and some animals. Where there is absolutely no confusion (i.e. between the majority of classes) it can be seen that the ResNet residual network architecture has clearly learned the unique features of these classes.

These results are also reflected in Appendix Bb, where some examples of misclassified images are shown. It can be seen here again that most of the class confusion is between vehicles, and animals which make sense as these classes are more likely to share similar features.

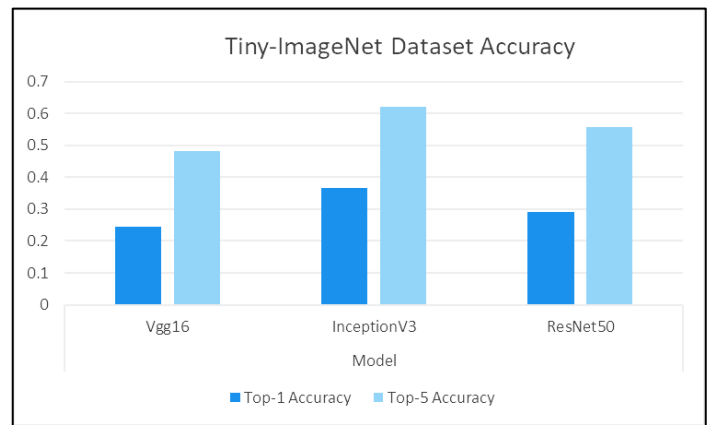
#### 4.3. *Tiny-ImageNet Results*

The Tiny-ImageNet dataset validation set results were plotted in Figure 6. It can be seen that again Inception achieved the highest top-1 accuracy, and that VGG achieved the worst. This was also true of the top-5 accuracy results.

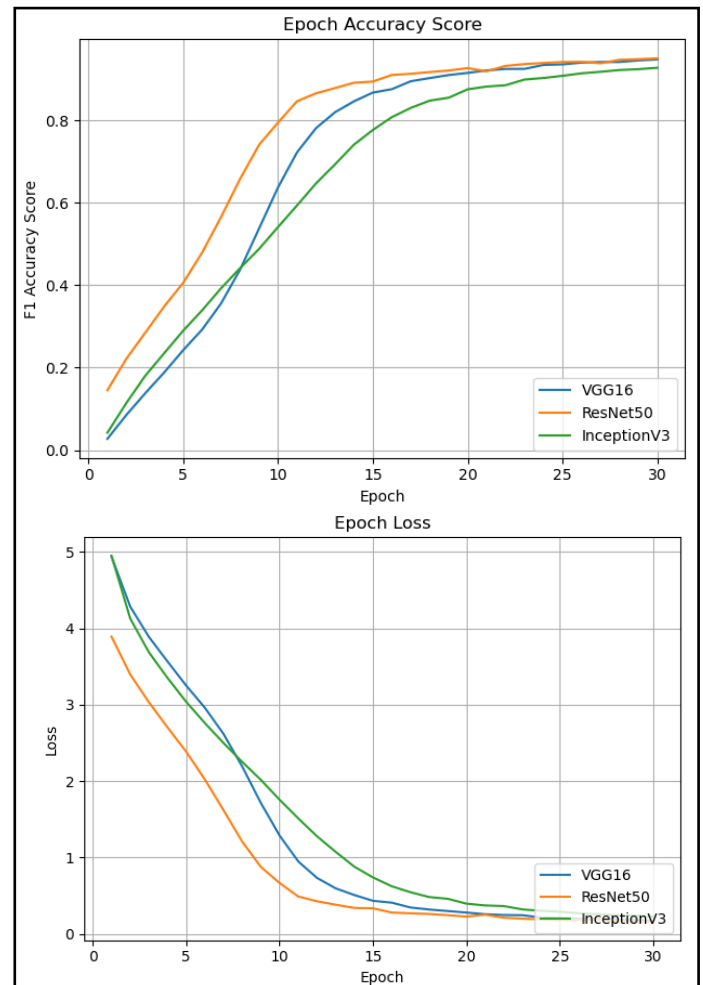
From the training accuracy and loss plots in Figure 5, it can be seen that ResNet reached the highest accuracy first, and was later joined by VGG. This same pattern was observed in reverse for the epoch loss.

Due to the larger number of classes for this dataset, confusion matrixes were not produced, but some examples of misclassified images were plotted in Appendix Bc. From these images, in most cases it is clear to see why an image has been misclassified, for example a Yorkshire retriever is misclassified as a different breed of dog by the VGG and ResNet models, and a remote control is classified as a sandal by the Inception model which is likely due to the shape.

Looking at the misclassified classes from each model the Inception model predictions are clearest to understand and seem to relate mostly to the shape of the image which makes sense as the inception model focuses on different sized feature extraction. The VGG and ResNet mispredictions are more difficult to interpret, for example swimming trunks are predicted as a boa constrictor by the VGG model. If the models were able to be trained further perhaps more trends would have emerged.



**Figure 6:** *Tiny-ImageNet validation set results for models*



**Figure 7:** *Tiny-ImageNet training accuracy and loss*

## 5. Conclusions

Overall, the VGG-16 model tended to score the highest training accuracy, but the lowest validation set accuracy. This suggests that perhaps this model overfits more than the other two. This model was also the slowest to train, which therefore makes it less practical than the other two.

The Inception-V3 and ResNet-50 models both achieved similar accuracies, although ResNet-50 appeared to suffer from a bit more class confusion. However, ResNet was significantly faster than the other models. Considering potential real life applications where computational requirements are an important consideration, ResNet therefore is probably the best of the evaluated models to use for image classification.

## 6. References

- [1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS*, pp. 1106-1114, 2012.
- [2] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *ECCV*, 2014.
- [3] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *ICLR*, 2014.
- [4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015.
- [6] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166, 1994.
- [7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (PMLR)*, vol. 9, pp. 249-256, 2010.
- [8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, 2015.
- [9] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *Proc. ICLR*, 2014.
- [10] A. Khan, A. Sohail, U. Zahoor and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, pp. 5455-5516, 2020.
- [11] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," *IEEE/CVF conference on computer vision and pattern recognition*, pp. 7132-7141, 2018.
- [12] F. Wang, M. Jiang and C. Q. e. al., "Residual attention network for image classification," *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 6450-6458, 2017.
- [13] A. G. Roy, N. Navab and C. Wachinger, "Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks," *Lecture Notes in Computer Science (LNCS)*, vol. 11070, pp. 421-429, 2018.
- [14] S. Woo, J. Park, J. Y. Lee and I. S. Kweon, "CBAM: Convolutional block attention module," *Lecture Notes in Computer Science (LNCS)*, vol. 11211, pp. 3-19, 2018.



- [15] A. Khan, A. Sohail and A. Ali, "A New channel boosted convolutional neural network using transfer learning," *arXiv:1804.08528*, 2018.
- [16] R. K. Srivastava, K. Greff and J. Schmidhuber, "Highway Networks," in *ICML*, 2015, arXiv:1505.00387.
- [17] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *Multimedia Tools and Applications*, vol. 77, pp. 10437-10453, 2015.
- [18] G. Huang, L. V. D. Maaten and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings of 30th IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2261-2269, 2017.
- [19] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.
- [20] W. Chen, J. T. Wilson and S. T. e. al., "Compressing neural networks with the hashing trick," *32nd International Conference on Machine Learning (ICML)*, 2015.
- [21] S. Han, H. Mao and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," *4th International Conference on Learning Representations (ICLR)*, 2016.
- [22] J. Wu, C. Leng, Y. Wang and e. al., "Quantized convolutional neural networks for mobile devices," *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, 2016.
- [23] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *CEUR Workshop Proceedings*, 2018.
- [24] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18-23, 2018.
- [25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Computer Vision and Pattern Recognition*, 2016.
- [26] Y. LeCun, "The mnist database of handwritten digits," <http://yann.lecun.com/exdb/mnist/>, 1998.
- [27] A. Krizhevsky, G. Hinton and e. al., "Learning multiple layers of features from tiny images," 2009.
- [28] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," Stanford University, 2015.