Using Transfer Learning to Classify the Oxford-IIIT Pet Dataset

Andreas Wrife
Royal Institute of Technology
awrife@kth.se

Leo Noharet
Royal Institute of Technology
lnoharet@kth.se

Cajsa Pierrou
Royal Institute of Technology
cajsap@kth.se

Abstract

Training a full CNN model from scratch can be time-consuming, expensive and complex, which makes transfer learning a convenient alternative. This work explores fine-tuning a ResNet network pretrained on the ImageNet data set, adapting it to the OXFORD-IIIT Pet data set and aiming to solve the binary classification problem of recognising cat and dog species as well as the multi-class problem of recognising cat and dog breeds. By our experimental results and a study in state-of-the-art methods, we find that the network can achieve high accuracies in these classification tasks when fine-tuned. In particular, we show improvements in performance when fine-tuning multiple layers, using data augmentations and various means of normalisation and regularisation, as well as using deeper networks. Several of these improvements does however come at the cost of parameter complexity and longer training.

1 Introduction

Image classification has a great number of important applications, but requires good accuracy to be reliable. A well trained Convolutional Neural Network (CNN) can achieve this thanks to its deep architecture. However, fully training a CNN from scratch is no simple task. It requires a large amount of labelled data appropriate for the purpose of the classification. Moreover, great computational power and memory capacity are central in training a CNN from scratch to complete training within a feasible time-frame. This can become very expensive. Finally, training from scratch can be tedious and complex, demanding expertise to achieve good enough accuracy [TSG+16]. An alternative to this approach is to fine-tune CNN trained on a broad and general data set. This report explores this training approach, by fine-tuning a ResNet model trained on ImageNet [HZRS15], to solve the image classification problem on the pet data set OXFORD-IIIT, which includes labelled pictures of cats and dogs of different breeds. When the dataset was first published, researchers reached an accuracy of 59%[PVZJ12]. The aim of this project was to reach accuracies in the upper 90%'s, both when classifying the species and the different breeds.

The work found that fine-tuning deeper networks and fine-tuning more layers than solely the last yields better performance, that excluding batch normalization layer from fine-tuning yielded both faster training and better accuracy at test time. Also, regularization by means of data augmentations work well, however this entails longer training.

2 Related Work

Kornblith et. al. describe their findings from exploring the effect of transfer learning from ImageNet and the difference in accuracy between fine-tuning a CNN and training from scratch CNNs on different data sets, including the OXFORD-IIIT Pet data set [KSL19]. They found that when training a CNN from scratch, accuracies of approximately 75% for ResNet architectures were achieved, while fine-tuning yielded accuracies around 94 % for ResNet-152 and ResNet-101, and 92% for ResNet-50, showing the benefit of transfer learning from ImageNet. For larger data sets, the accuracy benefits of fine-tuning compared to training from scratch decrease rapidly [KSL19].

3 Data

For this project, the OXFORD-IIIT Pet data set was used, which includes 7349 pictures of cats and dogs. The data set is labelled by both specie, i.e. cat or dog, and by breeds within each specie. There are 37 breeds of animal, among which 25 are dogs and 17 are cats. For each breed there is about 200 images. The data set is split into two sections: 3669 images for testing and 3680 images for training and validation. We used 80% of the 3680 images for training and the remaining 20% for validation.

The images were resized to the smallest acceptable size of our chosen network (ResNet), 224×224 pixels. As recommended in Inkawhich's tutorial [Ink17] and in the 'Project Type' document, this was done by transforming the images by a scale factor of 224, such that the shortest side of each image is re-scaled to 224, and then center cropping the image.

The data set also had to be normalized, for which we used the standard values for mean and standard deviation proposed in Inkawhich's tutorial [Ink17]. These values resulted in some very dark images, so we tried computing the mean and standard deviation of the OXFORD-IIIT Pet data set and normalizing using these values instead. The results were much brighter images, though no significant difference in either loss or accuracy. The lack of improvement in performance may be due to the standard values being derived from ImageNet, on which ResNet is trained, and which contains very similar images to what is in the OXFORD-IIIT Pet data set. We note that our computed means and standard deviations deviated from the standard values by less than 0.8. Manually computing these values for normalization may be more important if the domain differs more from the domain on which the network is trained.

4 Methods

In a tutorial on transfer learning with the Python PyTorch module, Inkawhich describes the basic principles and workflow to adapt a pre-trained network to a new data set [Ink17]. The main steps from this tutorial make up the framework upon which this project's code was built.

To approach the classification problems, we fine-tuned parameters one by one and observed their effect on the network and its performance, in what could be viewed as an informal ablation study. For each experiment, a baseline was set to which the experiments outcome was compared to. These baselines consisted of test accuracies together with plots of loss and accuracy on training and validation data during training.

4.1 Binary classification problem

For the binary classification problem of categorizing cats and dogs, we only reshaped and fine-tuned the final (fully connected) layer of our model. As Adam optimizer was used to compute the gradients, the main focus in fine-tuning was finding a good learning rate. Hence, a coarse search was implemented. The search began on a uniform grid search basis, with 10 values ranging from 0.001 to 0.00001 were used and test, validation and training accuracy was logged together with training and validation loss. At first, test accuracy was the primary indicator of whether a given learning rate performed well or not. Iteratively, the value range from which the coarse grid grew smaller. After some iterations, when differences in test accuracy between different learning rates grew smaller, the validation and training losses plotted against epochs became as important a metric to evaluate a learning rates performance as the test accuracy. The loss and accuracy plots indicate whether the current model overfitted or underfitted the model to the training data, as well as if the learning rate was higher of lower than the optimal. When a small enough interval of learning rate values was found, a finer random search was executed, sampling random values in the interval as randomly chosen trials of hyper-parameter values has been shown to be more effective than trials on a grid [BB12]. However, as the Adam optimizer does not have many hyper-parameters to optimize, and as solely the learning rate was optimized, grid searches and random searches proved to be approximately as efficient in learning rate searching.

4.2 Multiclass classification problem

To solve the multi-class classification problem of cat and dog breeds, exploring other fine-tuning approaches was required. In addition to the learning rate searches described in the binary classification problem, the general approach in this case to experiment with fine-tuning different hyper-parameters, and examining the effect that it has on the performance of the network. Identically to the binary classification fine-tuning approach, the accuracy on the test data was used to quantify the performance, whereas the loss and accuracy plots of training and validation data were used as a qualitative measurement of the performance.

4.2.1 Weight decay and Adam

For both binary and multi-class fine tuning tasks, experiments fine-tuning the weight decay hyper-parameter were conducted. At first, the weight decay values proposed in Lecture 5, 1e-5, 1e-4, 0 [Sul22] were tested. Out of those, 0 weight decay yielded best results. For that reason, an attempt with weight decay set to very small 1e-10 was conducted, however in vain, as it did not yield better results than with weight decay 0. After a short literature study, it became apparent that Loshchilov et. al. had showed that the implementation of the weight decay in the Adam optimizer of all software packages is wrong. [SG18] They propose a new optimizer AdamW that fixes this. While this was discovered in late 2017, as PyTorch contains both Adam and AdamW optimizers, it is our understanding that the weight decay implementation of Adam has not been changed, but the alternative AdamW has been added to the package. [LH17] Therefore, an alternative would have been to use the AdamW optimizer, and fine-tune the weight decay parameter. Moreover, another alternative approach is to fine-tune more hyper-parameters, such as momentum, or the parameters eps and $betas$ of the PyTorch Adam optimizer function.

4.2.2 Fine-tuning More Layers

An important experiment was to incrementally fine-tune more layers. At first, only the final layer, i.e. the fully connected layer, was fine-tuned by finding a good learning rate. Then, the parameter of layer groups Conv5_x to Conv1_x of ResNet18 were added gradually added to the group of fine-tuned parameters. The general hypothesis was that the more layers that were fine-tuned, the better test accuracy the model would produce based on [SLF22] . Also, our intuition was that the layers closer to the input should have smaller learning rate values and layers closer to the output of the network should have increasingly higher learning rate. This is because layers close to the input learn more general features of the data, which the ImageNet data set already incorporates well. Layers closer to the output also learn features more specific to the new dataset.

4.3 Alternative Approaches and Improvements to Method

An alternative approach, or rather an extension to the taken approach, could have been to log which test data cases the model classified wrongly, and attempt to identify from those test images why the model failed and what changes to fine-tuning could be made. Furthermore, confusion matrices could have been utilised to get an overview of the errors in classifying the 37 breeds. Also, as the training and validation data is the result of a random 80-20 split, it could have been helpful to split the data in a manner that ensures that the two sets contain all classes in approximately the same amount.

Our general approach was, though we tried to structure it, somewhat ad-hoc. Perhaps we would have benefited from an even more strictly structured ablation study of parameters and documentation of results.

5 Experiments

To ground our experiments we set up clear baselines for comparisons. These are detailed in sections 5.1 and 5.2 for the case of binary and multi-class classification, respectively, as well as in Section 5.3 where we use our best results from 5.2 as a baseline for our extensions for higher grades. Parameter settings for, and test accuracies achieved at, each baseline are listed in Table 7.2.

5.1 Binary Classification

Baseline B1: Pre-trained ResNet-18 without fine-tuning. (Test accuracy = 50.01%)
For this baseline we had to reshape the fully connected layer (to output size 2), but we did not re-train the model. Thus, performance of the model at this baseline corresponds to that of a random guesser.

Baseline B2: Pre-trained ResNet-18 with fine-tuning the last (fully connected) layer using Adam optimizer with all default settings. (Test accuracy = 98.39%)

The performance of the resulting model at baseline B2 yielded an accuracy not far from our goal of 99% test accuracy. Starting from this baseline, we initiated experiments with the batch size and number of training epochs. Increasing the batch size decreases the difference in loss and accuracy, respectively, between training and validation, thus working as normalisation. With a batch size of 16 and training for 15 epochs, our model achieved test accuracy 98.91%.

To see if we could improve the test accuracy further, we performed a search for a good learning rate. Our best found learning rate was 0.00115, which was very similar to the default learning rate of 0.001. The change in learning rate did not yield any significant improvement, so the default learning rate seemed essentially optimal. After experimenting with different approaches using ResNet18, we switched model to ResNet34, still using our best found learning rate. This yielded a test accuracy of 99.21%.

5.2 Multi-class Classification

Baseline M1: Pre-trained ResNet18 without fine-tuning. (Test accuracy = 2.59%)
For this baseline we had to reshape the fully connected layer (to output size 37), but we did not re-train the model. Thus, performance of the model at this baseline should correspond to that of a random guesser.

Baseline M2: Pre-trained ResNet-18 with fine-tuning the last (fully connected) layer using Adam optimizer with all default settings. (Test accuracy = 85.93%)

We started our experiments by only fine-tuning the last (fully connected) layer. It was evident from the plots of the models performance at baseline M2 that the default learning rate was too high, and that the model overfitted to the training data. We therefore performed a search for a better learning rate, and found our best one to be 0.0001. When running baseline M2 with a learning rate of 0.0001, our model achieved test accuracy 87.79% - a slight improvement in performance. The results of this improvement is represented as baseline M3. We proceeded with tuning the batch size and number of epochs. Provided a larger batch size (32) and longer training (45 epochs), our model achieved test accuracy 88.80%.

Baseline M3: Pre-trained ResNet-18 with fine-tuning the last (fully connected) layer using Adam optimizer and fine-tuned learning rate. (Test accuracy = 87.79%)

To increase the number of training data samples, we duplicated our training data set and performed data augmentations on the duplicated images. The augmentations we tested included flipping, small rotations (± 20), cropping and small resize scaling. Flipping was done horizontally, as vertical flipping did not yield any increase in accuracy. By duplicating the data set once and performing random horizontal flipping as augmentation, our model achieved test accuracy 88.40% - an improvement of about 1% from baseline M3. Duplicating the data set two times provided further slight improvements ($+\sim 0.4\%$) in test accuracy, though at the cost of longer training. In duplicating and augmenting the data set, the augmentations should be ensured to create different-looking images from each source image. If the augmentations are too subtle, we end up with (essentially) duplicates of all images, resulting in overfitting when training.

To evaluate the advantage of fine-tuning more layers, we iteratively incremented the set of fine-tuned layers. For computational efficiency, we fine-tuned layer-groups instead of individual layers when fine-tuning more than just the final fully-connected layer. In total, we fine-tuned layer groups 1-4. For each layer we searched for a good learning rate and tested the resulting test accuracy. It became apparent that including more layer-groups to the set of fine-tuned parameters improved performance, as long as the learning rate was appropriately small for each layer group. After some parameter searches, we ended up with the learning rates described in baseline M4 in table 7.2. By training all of these layers, an increase in test accuracy of 2% is gained compared to baseline M3.

Baseline M4: Pre-trained ResNet18 with fine-tuning all layers with appropriately small constant learning rates, no data augmentations and without fine-tuning the batch norm. (Test accuracy = 89.79%)

Besides using a constant learning rate for each layer we fine-tune, we may use a learning rate scheduler. We tried both exponentially decreasing and one cycle learning rate schedulers, though neither seemed to yield significant improvements in accuracy or evolution of loss during training. According to Leslie Smith [ST19], a one cycle learning rate scheduler with a high maximum learning rate has the ability to reach super convergence since high learning rates help to regularize the training, and demands therefore less of other regularization methods [ST19]. Since weight decay was not used in fine-tuning, a one cycle scheduler had the potential to solve the regularization problem. We believe that further experiments with the OneCycle learning rate scheduler is needed.

When fine-tuning more layers of our model, we also have the option to fine-tune the parameters for batch normalization. The accuracy and loss during training, with and without fine-tuning the batch norm, are shown in Figure 1 and 2. From these results we can deduct that not fine-tuning the batch norm layers allows us to equalize the performance of our model between the training and validation data sets. By excluding the batch normalization layers from the set of fine-tuned layers, a test accuracy of 90.24% is achieved, a slight improvement compared to baseline M4.

By combining the improvements described above, and a test accuracy of $\sim$90.3% is achieved when training the ResNet-18 model for 40 epochs. This model is defined in baseline M5.

5.3 Extensions for D/C grade

Baseline M5: Pre-trained ResNet-18 with fine-tuning all layers with constant learning rates for each layer except the batch normalization layers, and training on double the amount of training data achieved by augmenting using random horizontal flip. (Test accuracy = 90.3%)
This baseline corresponds to the result of our best performing model from Section 5.2.

For our extensions, we experimented with more sophisticated data-augmentations than in the previous section. We explored photometric augmentation in terms of Gaussian Blurs and Color Jitters, as well as affine transforms in terms of perspective.

The effect of Gaussian blur can help train a model in recognising more robust features [CK19]. Thus, it might have helped in recognising species, since cats and dogs usually have somewhat different silhouettes. However, too much blur may cause this difference to be negligible. Even more so could be said about the different breeds. With the same silhouette and blurred details, breeds could be almost indistinguishable under noise. The addition of Gaussian Blur as an augmentation yielded an accuracy of 88.34% - dipping below baseline M4. The validation accuracy function became a bit less smooth. This was obtained using a random sigma (variance of the amount of blur) in the interval (0.1,5). When setting a higher minimum standard deviation; sigma=(2,5) the accuracy decreased down to 87.59%. On the opposite side, setting sigma to the default parameters of sigma=(0.1,0.2) made the blur too unnoticeable, resulting in the same accuracy as that of baseline M4.

Transforming the perspective of inputs could have an important difference since the animals are photographed from a variety of perspectives. The one image could therefore be reused to simulate it being taken from a different view. However, with a too large perspective change the animal may become unrecognisable. The same goes with the amount of padding required on the side to compensate for the transformation. Training on a filler colour is not preferable (filler colour referring to the background colour appearing behind the image to fill out the canvas after perspective change or rotation). After adding perspective transformations to our data set and training with otherwise default settings, our model achieved an accuracy of 88.3%, thus not improving accuracy compared to baseline M4.

After experimenting with the photometric and affine data augmentations, we found that the best performing data augmentation was the combination of random horizontal flip and ColorJitter. However, when only one duplication of the training data was performed, the increase in accuracy remains small ($\sim$ 0.5%). The differences in training and validation accuracy and loss between baseline M4 and the model with one augmentation can be seen in figures 3 and 4 respectively. This model was our best performing model to date, yielding test accuracies of $\sim$90-91% depending on the number of epochs run.

With this model as a baseline, including one data augmentation, we experimented with deeper networks. After trying the pretrained versions of ResNet34, ResNet50 and ResNet152 without fine-tuning the hyper-parameters especially for these new deeper networks, significant increases in testing accuracy was achieved. Test accuracies of $\sim$92.9% and $\sim$93.9% were achieved using ResNet-50 and ResNet-152, respectively. As redoing all the work we did for finding good parameter settings for ResNet18 is not feasible to do within the given time-frame, we can only assume that further performance gains could be reached if the same amount of fine-tuning was done for the deeper networks. This result confirms what was found in [KSL19]: a test accuracy of $\sim$ 92% can be achieved when fine-tuning ResNet50 to the OXFORD-IIIT Pet data set, and a test accuracy of $\sim$ 94-95% can be achieved with ResNet152. The loss and accuracy plots of the training with ResNet50 and ResNet152 are shown in figures 5 indicate that hyper-parameters such as learning rate are not optimal for the network, which leads to believe that further performance gains can be achieved if profound parameter searches are conducted.

At the end of these experiments we found ourselves with a ResNet-18 that achieved very high training and validation accuracy, almost 0 loss but an almost 10% drop in test accuracy. These results indicated that we may have a problem generalization of our model. We started implementing dropout to attempt to combat this, though we did not finish.

6 Conclusion

In conclusion, transfer learning allowed us to reach high test accuracies when applying a pretrained neural network to a new set of data. The process was time consuming, especially when fine-tuning deeper networks. The deeper architectures did however yield advantages in higher possible accuracy. Other tools leading to improvements in performance were normalization and regularization. In particular, data augmentations were a good way to regularize and increase the amount of available training data. Excluding the batch-norm layers from the fine-tuning process also proved beneficial to the performance of the network.

Through this project we have learned a lot, both about the theory behind fine-tuning a deep neural network, but also about the process of doing so. Our top insights include using several metrics to evaluate the performance of a model - only looking at, for example, test accuracy is not sufficient to find out what is going on. Plotting and visualizing the evolution of metrics such as loss and accuracy proved greatly beneficial here. Furthermore, performing an ablation study on aspects and parameters of a network greatly aids the understanding of these, which in turn facilitates a methodical fine-tuning process.

7 Appendix

7.1 Figures

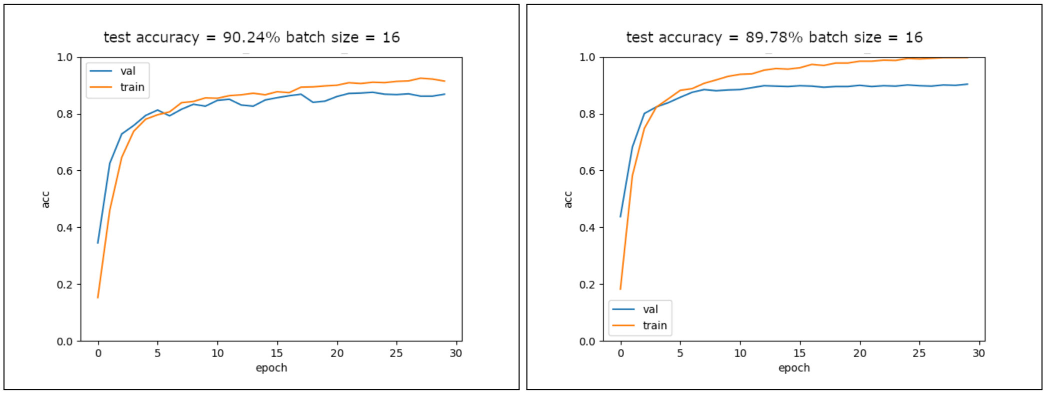

Figure 1: Comparison of training and validation accuracy, without (left) and with (right) fine-tuning of the batch normalization parameters for all layers of ResNet18.

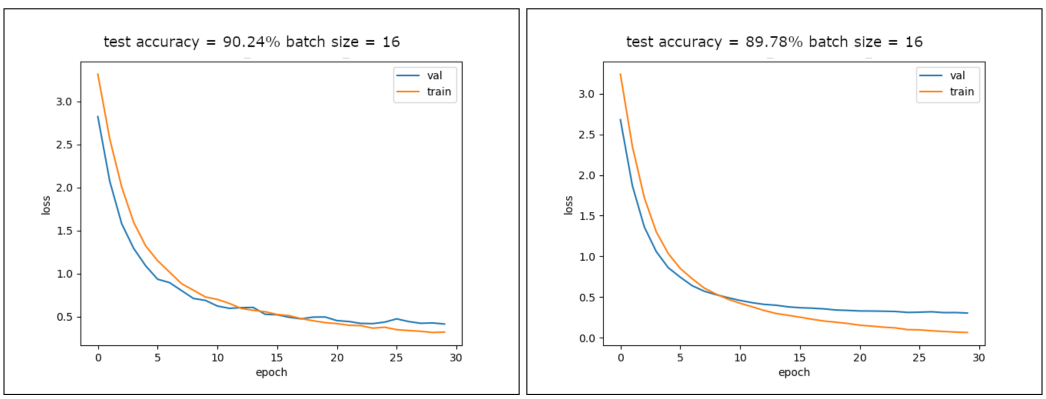

Figure 2: Comparison of training and validation loss, without (left) and with (right) fine-tuning of the batch normalization parameters for all layers of ResNet18.

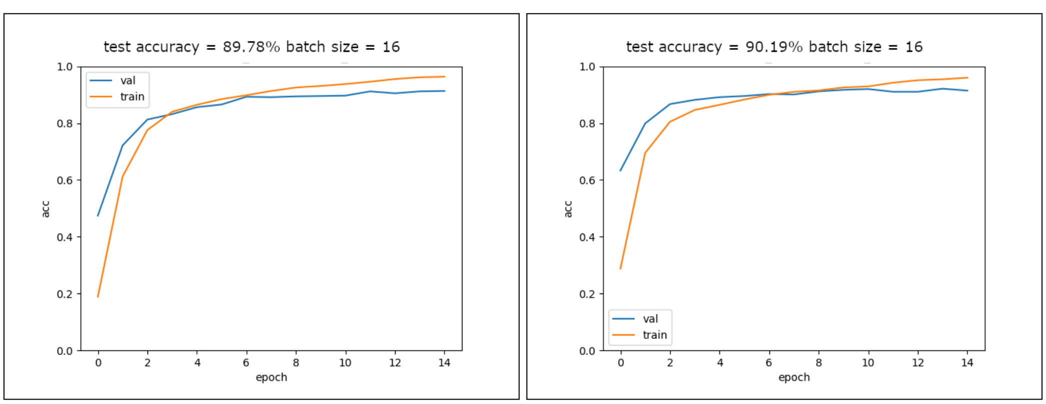

Figure 3: Comparison of training and validation accuracy, without (left) vs with (right) duplicating the data set and augmenting with ColorJitter and random horizontal flip

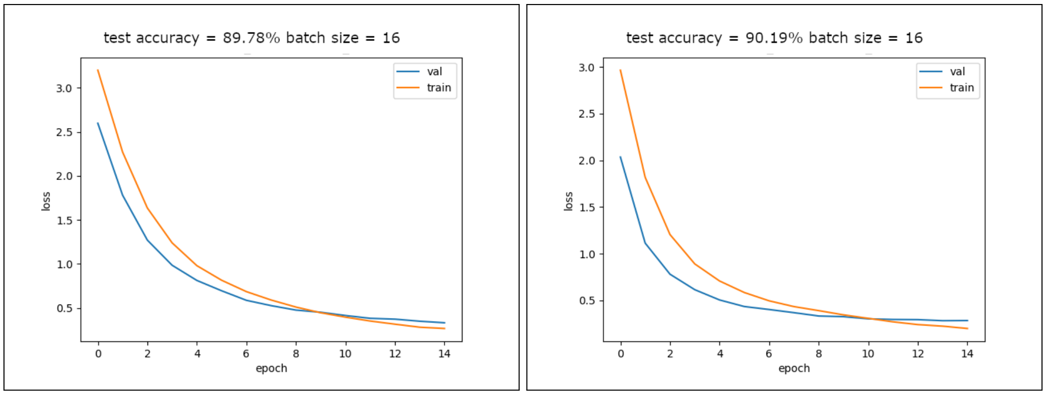

Figure 4: Comparison of training and validation loss, without (left) vs with (right) duplicating the data set and augmenting with ColorJitter and random horizontal flip

7.2 Tables

Table 1: Baseline Parameter Settings and Test Accuracies for ResNet-18 Model

			Learning Rate by Layer					
Baseline	Batch Size	Epochs	fc	4	3	2	1	Test Accuracy
B1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	50.01%
B2	8	15	0.001	-	-	-	-	98.39%
M1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	02.59%
M2	8	15	0.001	-	-	-	-	85.93%
M3	8	15	0.0001	-	-	-	-	87.79%
M4	16	15	0.0001	3e-6	1e-6	1e-7	1e-8	89.78%
M5	16	40	0.0001	3e-6	1e-6	1e-7	1e-8	90.30%

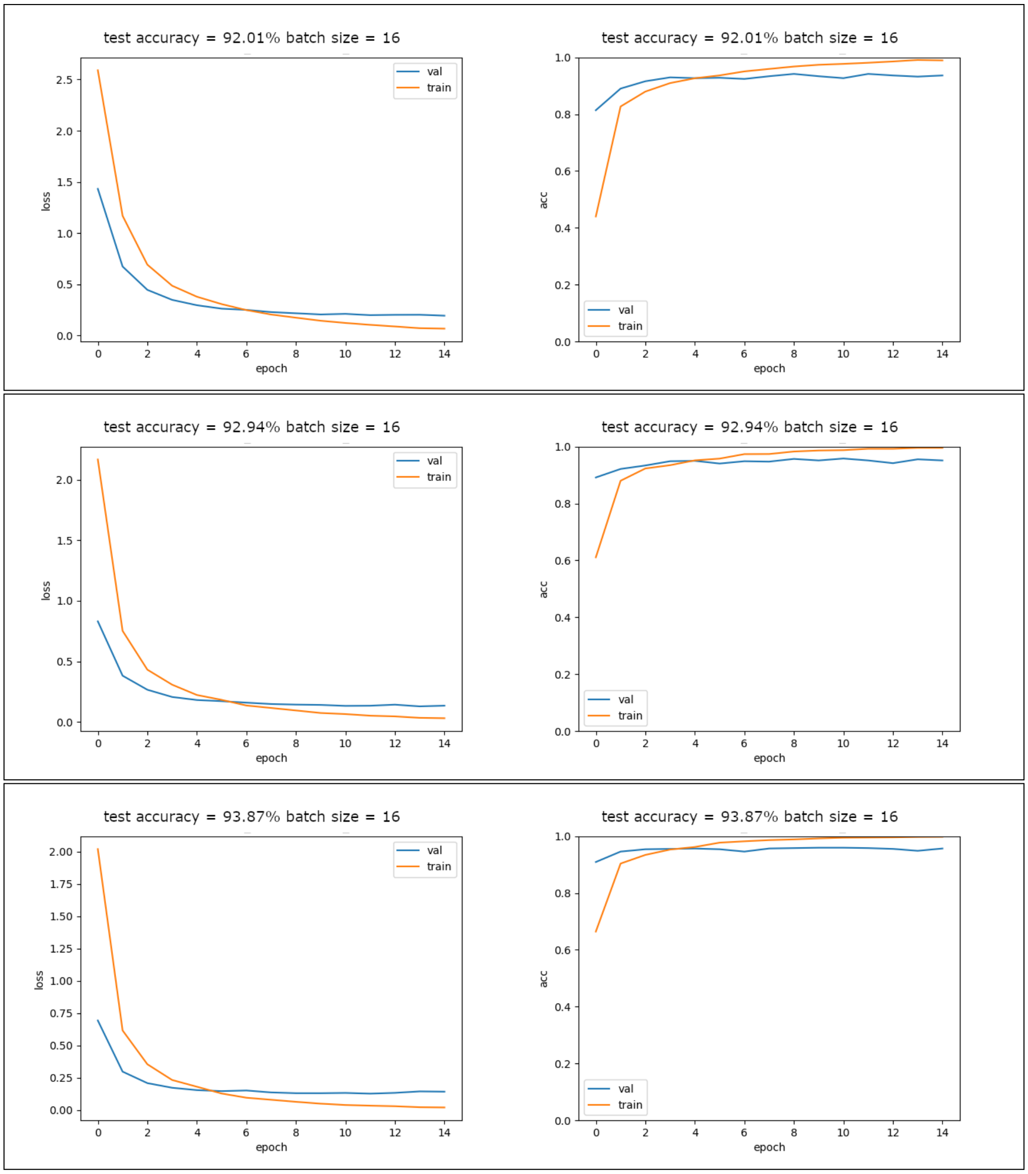

Figure 5: Comparison of performance between ResNet-34 (upper), 50 (middle) and 152 (lower). The corresponding plots for the same parameter setting for ResNet-18 is shown on the right side of figures 3 and 4

References

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[CK19] Shorten Connor and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal Of Big Data*, 6(1), 2019.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Ink17] Nathan Inkawhich. Finetuning torchvision models, 2017.

[KSL19] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

[LH17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[PVZJ12] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[SG18] Jeremy Howard Sylvain Gugger. Adamw and super-convergence is now the fastest way to train neural nets, 2018.

[SLF22] Noha Sarhan, Mikko Lauri, and Simone Frintrop. Multi-phase fine-tuning: A new fine-tuning approach for sign language recognition. *KI-Künstliche Intelligenz*, pages 1–8, 2022.

[ST19] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[Sul22] Josephine Sullivan. Lecture 5 - training & regularizing neural networks. Lecture Slides, March 2022.

[TSG+16] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016.