Exploring transfer learning

Authors
Joakim Abdinur Iusuf - joakimai@kth.se
Johan Maseng - jmaseng@kth.se
Alexander Falk - alefalk@kth.se

Abstract

Transfer learning is a well established technique in deep learning which is especially useful when there is limited data avaliable. This study explores the use of transfer learning using the pre-trained model ResNet34 for computer vision task using the Oxford-IIIT Pet Dataset to address two objectives: binary classification to recognize dogs and cats and multi classification to identify 37 different breeds. For binary classification the last layer of ResNet34 was replaced and fine-tuned using the Pet dataset which resulted in a test accuracy of 99.319%. For the multi classification task, different fine-tuning strategies were applied such as modifying learning rates, fine-tuning more layers, instead of just the last layer, applying data augmentation and checking the effects of fine-tune or not the batch-norm parameters. The optimal configuration achieved a test accuracy of 94.429 %.

1 Introduction

Transfer learning is one of the most common use cases in deep learning. It opens up opportunities to solve new problems by adapting pre-trained models to new datasets that are often smaller in size. In this project, we will explore transfer learning in a computer vision task by using Convolutional Neural Networks (ConvNets). In particular, we will use the pre-trained model ResNet34. To do this, we will download ResNet34 and fine-tune it for two specific tasks, described below.

1.1 Approach and Goals

Our primary objective is to fine-tune ResNet34 to solve two problems using the Oxford-IIIT Pet Dataset [1]. The first problem we will try to solve is binary classification to distinguish between dog and cat images. The second problem is more complicated, and it is multi-class classification to identify the specific breed of cat or dog, which involves 37 different outputs.

To solve the binary classification task, we will replace the final layer of ResNet34 and fine-tune it using the Pet Dataset. The goal is to achieve a test accuracy of more than 99 % using the Adam optimizer.

For the multi-class classification problem, we will explore different strategies: The benefit of fine-tuning more layers than just the last layer. Using different learning rates for different layers. Applying data augmentation during training. Evaluating the effect of fine-tuning batch normalization parameters and updating the batch mean and standard deviations. The goal is to achieve a test accuracy around 95%, again using the Adam optimizer.

1.2 Model Architecture

The model we used for classification of cats and dogs as well as their breeds was ResNet-34, which is a 34-layer Convolutional Neural Network (CNN). The model is built up with 4 different residual blocks and 1 convolutional and final layer:

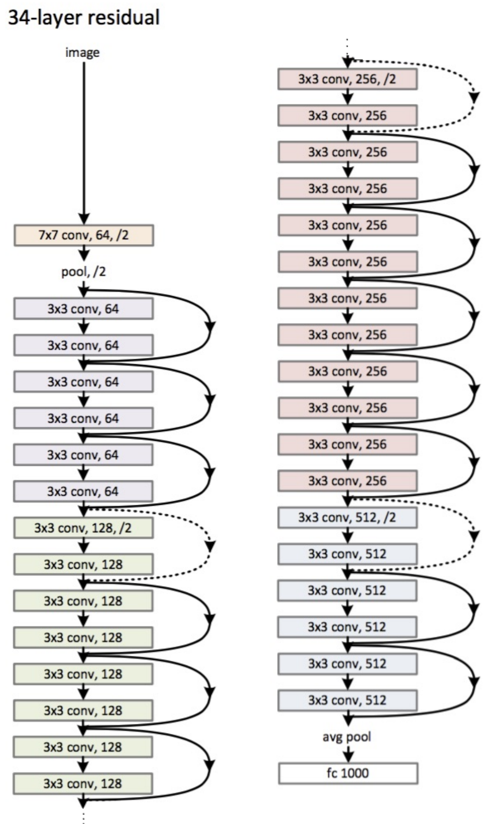

Figure 1: ResNet-34 architecture [2].

conv2: 3 blocks * 2 layers per block = 6 layers
conv3: 4 blocks * 2 layers per block = 8 layers
conv4: 6 blocks * 2 layers per block = 12 layers
conv5: 3 blocks * 2 layers per block = 6 layers

ResNet-34 was imported from the deep learning library PyTorch, which was based on the paper *Deep Residual Learning for Image Recognition* [3].

2 Methodology

2.1 Dataset and pre-trained model

We used the Oxford-IIIT Pet Dataset, which contains images of cats and dogs as well as a total of 37 breeds (25 cat breeds and 12 dog breeds and around 200 images of each breed). The data was split into 70 % for training, 20 % for validation, and 10 % for testing. We used the pre-trained ResNet34 model, which was trained on the ImageNet dataset, and modified it to suit our classification tasks. All explorations were done on Google Colab.

2.2 Binary classification

For the binary classification task, we replaced the final fully connected layer of ResNet34 with 2 output features, so that it would be suitable for binary classification. This layer was finetuned using the Adam optimizer with a learning rate of $1e$-4 for 3 epochs.

2.3 Multi-class classification

For multi-class classification, we modified the final layer to have 37 output neurons, each corresponding to a breed. We first only fine-tuned the final layer using a learning rate of $1e$-4 for 5 epochs. We then attempted to perform further optimization by fine-tuning more layers combined with different learning rates (2.3.1), augmenting the data (2.3.2) and batch-normalizing the parameters of the model (2.3.3).

2.3.1 Fine-tuning more layers and exploring different learning rates

We incrementally fine-tuned up to five layers. When unfreezing layers in ResNet34 to fine-tune them, we are actually unfreezing entire residual blocks, mentioned above. We incrementally fine-tuned more layers with the learning rates $1e$-4 and $1e$-3.

We then did a fine search in the range $5e$-5 and $1.5e$-4 to find the optimal learning rate when fine-tuning the final layer and second to last layer.

2.3.2 Data augmentation

To further improve the model, we applied data augmentation. This included random horizontal flips, random rotations up to 10 degrees, and random resized crops to 224 x 224 pixels.

2.3.3 Batch-norm parameters

Lastly, we explored the effect of fine-tuning or not the batch-norm parameters and updating the estimate of the batch mean and standard deviations.

3 Results

3.1 Binary classification

For the binary classification task, after replacing the final layer, we set the learning rate to $1e$-4 and trained our model for 3 epochs. The test accuracy after training was 99.593 %, the evolution of the validation accuracy across the epochs are visible in Table 1.

	Validation Accuracy
Epoch 1	98.775 %
Epoch 2	97.617 %
Epoch 3	99.319 %

Table 1: Validation accuracies for each epoch when training our model using a learning rate of $1e$-4

3.2 Multi-class classification, baseline

For the multi-class classification task, when only replacing the final layer, we set the learning rate to $1e$-4 and trained our model for 5 epochs. The test accuracy after training was 90.489 %.

3.2.1 Fine-tuning more layers and exploring different learning rates

After solving the problem when only replacing the final layer, we explored the benefit of fine-tuning more layers. The learning rate was again set to $1e$-4. The results of this are displayed in Table 2. Next, we changed the learning rate to $1e$-3 and solved the problem by only replacing the final layer as well as fine-tuning more layers to see if there was a benefit to doing this. Results are displayed in Table 3.

Fine-tuned layers	Test Accuracy
Final and second to last layer	91.576 %
Final, second to last, and third to last layer	91.033 %
Final, second to last, third to last, and fourth to last layer	90.625 %
Final, second to last, third to last, fourth to last, and fifth to last	86.141 %

Table 2: Test accuracies when fine-tuning more layers with a learning rate of $1e$-4

Fine-tuned layers	Test Accuracy
Final layer only	79.076 %
Final and second to last layer	87.228 %
Final, second to last, and third to last layer	80.707 %
Final, second to last, third to last, and fourth to last layer	65.897 %
Final, second to last, third to last, fourth to last, and fifth to last layer	70.924 %

Table 3: Test accuracies when fine-tuning more layers with a learning rate of $1e$-3

The results sjow that fine-tuning the final layer and second to last layer gives the best test accuracy. The fine search for 10 different learning rates when replacing only the final layer and second to last layer yielded the following test accuracies.

Learning Rate	Test Accuracy
4.999999873689376e-05	91.712 %
6.111110997153446e-05	92.799 %
7.222221756819636e-05	92.799 %
8.333333244081587e-05	91.848 %
9.444444731343538e-05	91.848 %
0.00010555556218605489	92.391 %
0.0001166666770586744	90.625 %
0.00012777777737937868	91.984 %
0.00013888889225199819	91.576 %
0.0001500000071246177	89.402 %

Table 4: Test accuracies for different learning rates when fine-tuning the final and second to last layers

The results indicate that, among the learning rates we have explored, a learning rate around $7.2e$-5 yields the best test accuracy.

3.2.2 Data augmentation

We explored the benefit of applying data augmentation. We used a random horizontal flip, which flips the image horizontally with a probability of 50%. We also randomly rotated images by up to 10 degrees. Moreover, we did a random resized crop, which means we resized images to a random size and then cropped it to the target size of 224 x 224 pixels. The learning rate was set to $7.2e$-5, we trained for 3 epochs, and we only fine-tuned the final layer and the second to last layer. This yielded a test accuracy of 94.429 %.

3.2.3 Fine-tuning or not the batch-norm parameters

When applying data augmentation and fine-tuning batch-norm parameters we get the test accuracy mentioned above, meaning 94.429 %. While unfreezing the layers and fine-tuning them, the running estimates of the mean and standard deviation are also updated. When we do not fine-tune batch-norm parameters we get a test accuracy of 93.207 %, and the running estimates of the mean and standard deviation remain as they were in the pre-trained model.

4 Discussion

We explored different aspects of fine-tuning ResNet34 for both binary and multi-class classification problems, and managed to achieve a test accuracy of more than 99 % for the binary classification problem and around 95 % for the multi-class classification problem.

One thing to note is that our coarse and fine searches were limited due to time and access to GPU's. A better learning rate than the one we found ($7.2e$-5) could be found by more extensive coarse and fine searches.

We also noted that our model started overfitting after about 4 epochs. Using a learning rate scheduler, rather than fixed epochs, could maybe prevent overfitting by varying the learning rate during training.

We observed an advantage when fine-tuning batch normalization parameters and updating the running mean and standard deviation (94.429 % versus 93.207 %). Our results show that using batch-normalization is advantageous for bigger models with regards to the test accuracy, this was a somewhat expected result with reference to the effects batch-normalization had in Assignment 3.

Furthermore, we saw that data augmentation significantly improved our test accuracy, likely due to its ability to improve generalization by providing diverse training samples. This indicates that augmentation techniques are important for improving model performance.

Interestingly, the best performance on the multi-class classification problem was achieved by fine-tuning only the final and second-to-last layers. This suggests that sometimes, less extensive fine-tuning can be more effective, maybe because it preserves the pre-trained features in the earlier layers.

In summary, we successfully improved the test accuracy on the multi-class classification problem from 90.489 % to 94.429 % through our explorations. This highlights the value of experimenting with different fine-tuning strategies, learning rates, and data augmentation. Moreover, achieving a test accuracy greater than 99 % on the binary classification problem shows the effectiveness of transfer learning in solving new problems with limited data.

References

[1] Parkhi, O. et al. (2012). The Oxford-IIIT Pet Dataset. https://www.robots.ox.ac.uk/ vgg/data/pets/

[2] Siddharth, M. (2021) *Building Resnet-34 model using Pytorch – A Guide for Beginners.*, https://www.analyticsvidhya.com/blog/2021/09/building-resnet-34-model-using-pytorch-a-guide-for-beginners/

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.48550/arXiv.1512.03385