# Transfer Learning, from ImageNet to X-rays

**Edvard Aldor**
Department of Computer Science
KTH Royal Institute of Technology
ealdor@kth.se

**Jesper Petersson**
Department of Computer Science
KTH Royal Institute of Technology
jepeters@kth.se

**Quintus Roos**
Department of Computer Science
KTH Royal Institute of Technology
quintusr@kth.se

## Abstract

This paper inquires how well transfer learning performs on X-ray images, specifically identifying normal lungs and lungs infected with pneumonia. Transfer learning is defined as using a model that already is trained on a certain task, and is then reused to solve another comparable task. The used dataset was downloaded from the website Kaggle, emanating from another paper that is related to this experiment. When working with datasets, a reasonable concern is that the model will overfit the data. To avoid this, noise was added in the pre-processing step. The pre-trained models used in this experiment were AlexNet, GoogleNet and ResNet18. The last fully connected layer of the pre-trained models were replaced with two nodes and its weights re-trained to suit the new classification problem. After the models had been retrained they were also fine-tuned by unfreezing all the layers and retraining them again but with a lower learning rate. The best performing networks were then used to create an ensemble model. The conclusion from the experiment is that the fine-tuned networks performed worse than the pre-tuned models, most likely due to overfitting. As a result, an ensemble model was created using the pre-fine-tuned models, which gave a minor performance increase.

Preprint. Under review.

## 1 Introduction

Deep learning requires vast amounts of data, and consumes a considerable amount of time when training a model from scratch. Transfer learning is a method that can avoid the need of using extensive amounts of data, and with the added benefit, it has a quicker training time. This can be a beneficial method to employ when the datasets are scarce with data (Tan et al. 2018). This experiment tests transfer learning by comparing the performance of the pre-trained models Resnet18, Googlenet and Alexnet on chest X-ray images of pneumonia. The dataset of X-ray images comes from Kaggle and has been used before in similar experiments. The pre-trained versions of these models are trained on ImageNet. The experiment continued with fine-tuning the models to see if this enhanced the performance. Lastly a ensemble model was created by the best performing networks. The results showed that the fine-tuning process made each network perform worse than its counterpart, probably because of overfitting the data. The ensemble model performed best out of all the other models.

## 2 Related Work

There exists plenty of tests done around transfer learning, both generally and on X-ray images. The experiment this project seeks to inquire has been done before in a very similar manner by (Chouhan et al. 2020). The authors performed a thorough experiment, testing five pre-trained models with cross validation, the models they used were GoogleNet, Inception V3, AlexNet, DenseNet121 and ResNet18. These models were then used to create an ensemble model, which achieved the highest test accuracy compared to the other models. ResNet18 ran the most epochs (200) and reached the highest test accuracy of 94.23% compared to the individual models. The ensemble model performed better than the individual models and reached a test accuracy of 96.39%.

## 3 Data

The dataset used in this assignment is called *Chest X-ray Images (Pneumonia)* and was downloaded from Kaggle. The dataset itself originates from a paper by (Kermany et al. 2018). This dataset contains 5863 X-ray images of patient divided into two distinct categories: “normal” and “pneumonia”. The original split between training, validating and testing was 5216, 16 and 624 respectively. In order to increase the amount of validation images, 100 images of healthy patients and 200 images of sick patients were moved from the training set and added to the validation set resulting in 4916 pictures for training, 316 for validation and 624 for testing. All images were pre-processed including resizing, cropping, flipping and normalizing them before training. The reason why pre-processing was done was to add noise to the images, as it has been shown that it can improve the models generalization capabilities (Neelakantan et al. 2015). The pre-processing was done by:

1. Resizing the images to 224 x 224 x 3 pixels (ensure images are the same size)
2. Random Resized Cropping (increase connections between pixels)
3. Random Horizontal Flipping (increase symptom detection for both sides of the chest)
4. Normalizing (ensure that the features have values from similar ranges)

# 4 Methods

For solving the problems discussed in the introduction section, the following process was done three times, once for each of the pre-trained models in the experiment. The process is described below.

First, the data was pre-processed as described in section 3. After all data had been processed, one of the pre-trained networks was loaded. Since transfer learning was used, the weights of the pre-trained network were not updated (i.e. frozen). As the last fully connected layer was trained on 1000 classes from the ImageNet, Those classes needed to be replaced at the last layer with nodes corresponding to the number of classes in the dataset, which in this case is two, one for normal and one for pneumonia. These new weights then needed to be retrained so they corresponded to the used dataset. Only the weights of the last layer were updated during training as the other weights were frozen.

The training used cyclic training with a max learning rate of $0.1$, default learning rate of $0.01$ and a momentum of $0.9$. Both cyclic and momentum were used to help with converging and reduce the need to update the hyper-parameters. The training was done for 25 epochs and with a batch size of 32. The reasoning for choosing the number of epochs to be small was to reduce the risk of overfitting because of the small amount of available data. When the network had been trained for 25 epochs, it was then also fine-tuned. The fine-tuning process was done by unfreezing all layers in the network and then retraining all layers again, but this time for 10 epochs and with a learning rate of 1e-5. The fine-tuned networks were then tested by using the test set to evaluate its performance.

In order to analyze the experiments, each network was tested before and after the fine-tuning to examine if it improved the networks performance. A comparison was made between each of the models in order to observe which had the highest accuracy before and after fine-tuning. Lastly, after all networks had been trained and fine-tuned, an ensemble model was implemented to detect any potential improvements in the accuracy on the test data. Ensembling was done by each of the networks giving their prediction for an image, and then a majority vote was made to see which prediction was a majority.

## 4.1 Pre-trained networks

The chosen pre-trained networks are ResNet18, GoogleNet and AlexNet. There exists plenty of pre-trained networks to choose from that are trained on ImageNet. Resnet18, GoogleNet and AlexNet were chosen because of their size, and qualified for the scope of this project. With more time and computational power, it would be reasonable to add more networks that might create a better ensemble model.

## 4.2 Different Approaches

There are multiple ways of approaching this experiment using transfer learning. Firstly, there are other pre-trained networks that can be used. There is also a possibility of changing the hyper-parameters, increasing or decreasing the learning rate, changing the amount of epochs ran during training, changing the momentum or changing from cyclic learning rate to decaying learning rate or other. The fine-tuning process can also be changed, using a different amount of epochs or a change of only unfreezing some of the layers rather than unfreezing all of them. With the low amount of data in the dataset, there is a considerable risk that the networks will overfit. To counteract this problem, noise can be added in the pre-processing step. Varying the amount of noise is a reasonable approach if overfitting has become a problem. Lastly, without using transfer learning, it is still possible to train from scratch but another dataset is required to avoid overfitting.

# 5 Experiments

## 5.1 Training

The different networks' training phase was not stable. The values of the loss and accuracy fluctuate a lot during training but would eventually have been improved from their initial values. On all the networks the training ran over 25 epochs with momentum and a cyclic learning rate. The figures below illustrate the volatile accuracies and losses.

### 5.1.1 AlexNet

AlexNet achieved a accuracy of 85.26% on the test data after training. Figure 1 and 2 shows the accuracy and loss respectively over the 25 epochs of training.

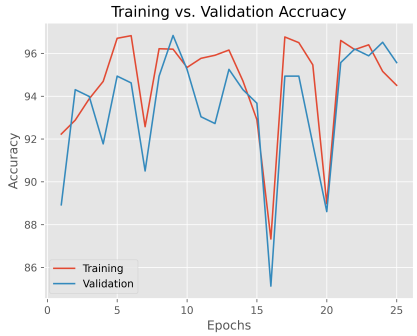


Figure 1: Accuracy over number of epochs during training of AlexNet

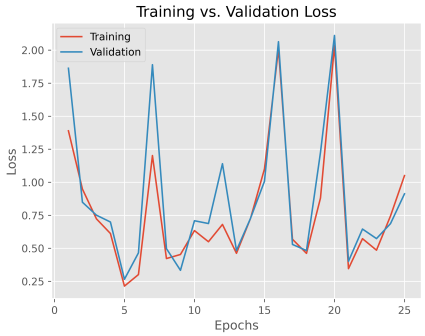


Figure 2: Loss over number of epochs during training of AlexNet

### 5.1.2 GoogleNet

GoogleNet achieved a accuracy of 86.22% on the test data after training. Figure 3 and 4 shows the accuracy and loss respectively over the 25 epochs of training.

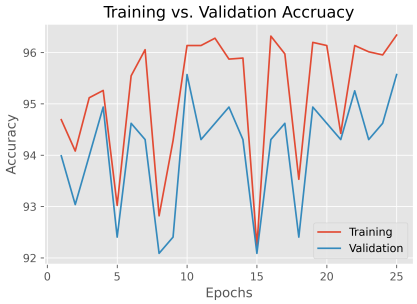


Figure 3: Accuracy over number of epochs during training of GoogleNet

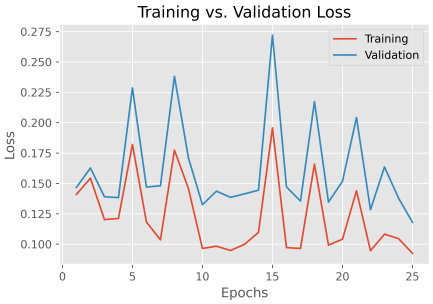


Figure 4: Loss over number of epochs during training of GoogleNet

### 5.1.3 ResNet18

At last ResNet18 achieved a accuracy of 86.54% on the test data after training. Figure 5 and 6 shows the accuracy and loss respectively over the 25 epochs of training.

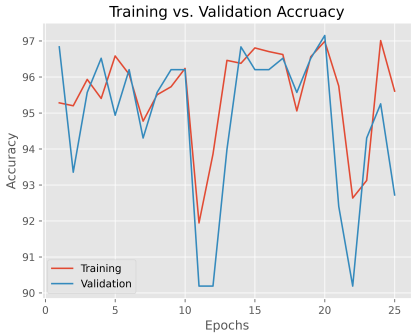


Figure 5: Accuracy over number of epochs during training of ResNet18

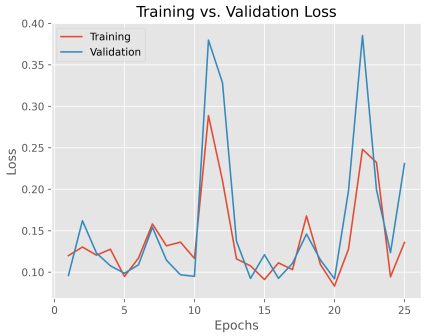


Figure 6: Loss over number of epochs during training of ResNet18

### 5.2 Fine-tuning

During fine-tuning all layers were unfrozen and retrained on the training data but this time with a learning rate of 1e-5. A low learning rate was used since during fine-tuning there should not be any large changes to the weights since they are already calibrated. Instead fine-tuning only adjusts the weights to be more inline to our specific problem. After fine-tuning had been applied there was a decline in the accuracies of all three networks, the results are shown in table 1.

| **Model** | **Before Fine-tuning** | **After Fine-tuning** |
|---|---|---|
| AlexNet | 85.26 % | 82.69 % |
| GoogleNet | 86.22 % | 84.13 % |
| ResNet18 | 86.54 % | 83.97 % |

Table 1: Accuracy on the test data before and after fine-tuning

### 5.3 Ensembling

The ensembling combined the predictions from the trained versions of AlexNet, GoogleNet and ResNet18 since they achieved better accuracy than their fine-tuned versions. The ensembling was done with a majority vote where the resulting label was the one with most predictions from the networks. With ensembling there was an improvement in the accuracy, achieving $87.34\%$ on the test data.

### 5.4 Discussion

Each pre-trained network before fine-tuning performed fairly similarly with ResNet18 having the best performance with an accuracy of 86.54%. This performance was very good considering the amount of data available. However, the training itself was unstable for all three models, having both accuracies and losses fluctuate during all of the 25 epochs. Although the resulting accuracies and losses improved after training, no convergence occurred. Training with more epochs might have made the networks converge but with the time and computational power available it was not feasible.

The fine-tuning of each network did not enhance the performance but instead decreased it. The performance in all likelihood decreased because of overfitting the training data or that the network had not converged during training. The most likely of the two is overfitting which was due to the small amount of data in the dataset. The decrease in performance might also have been due to the networks had not converged during training, since fine-tuning on a network that has not converged might disrupt the weights. With a larger dataset and more epochs of training we would probably have seen an increase in accuracy after fine-tuning. There is also a possibility that by adding more noise to the data there could have been improvements during fine-tuning, since noise might help avoid overfitting according to (Neelakantan et al. 2015). Another approach might be to fine-tune only some layers instead of fine-tuning all layers in the networks, since the earlier stages of layers focus more on abstract details like curves and edges while the layers at the later stages have more focus on the

details. Thus fine-tuning might perform better if only applied to the layers that are at the later stages. This would also allow for fewer epochs of training which decreases the risk of overfitting.

As all the fine-tuning networks performed worse than their regularly trained versions, no fine-tuning network was used in the ensembled model. The ensembled model performed best out of all networks with an accuracy of 87.34% which is not a major improvement but an improvement nonetheless. Thus, without using more data it is possible to increase test accuracy by using ensembling methods. The downside with this method is the requirement of more computational power.

# 6 Conclusion

Transfer learning can be a powerful tool when the amount of data available is limited. However, larger datasets present more possibilities for fine-tuning and more extensive training of the networks, as the risk for overfitting is reduced. The benefits of using fine-tuning is that it can greatly increase the performance as is seen in related work but if only a small amount of data is available chances are that it will do more harm to the networks due to overfitting. What can be done when only a small amount of data is available is to implement ensembling, which may improve the network. The only downside is that it requires a lot of computational resources since multiple networks need to be trained.

## 6.1 Future Work

As seen in related work, more networks, more data and possibly adding more noise makes it possible to train these networks over more epochs or better fine-tuning without overfitting to the data. While adding noise may also help, adding more epochs or using more extensive fine-tuning becomes more valuable and can greatly increase the performance as is seen in related work. Also, using more networks in ensemble learning could potentially result in more favorable results. Another suggestion would be to unfreeze a specific number of layers, instead of the whole network. As such, the different layer's capabilities can be assessed and compared.

## References


Chouhan, Vikash et al. (2020). "A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-ray Images". In: *Applied Sciences* 10.2. ISSN: 2076-3417. DOI: 10.3390/app10020559. URL: https://www.mdpi.com/2076-3417/10/2/559.

Kermany, Daniel S et al. (2018). "Identifying medical diagnoses and treatable diseases by image-based deep learning". In: *Cell* 172.5, pp. 1122–1131.

Neelakantan, Arvind et al. (2015). "Adding gradient noise improves learning for very deep networks". In: *arXiv preprint arXiv:1511.06807*.

Tan, Chuanqi et al. (2018). "A survey on deep transfer learning". In: *International conference on artificial neural networks*. Springer, pp. 270–279.