COMP47650 Deep Learning Project

Authors:
Zachary O'Connor (16305871)
Sabeer Bakir (16333886)
Kacper Twardowski (16401636)

Abstract

Neural Networks (NNs) have been steadily growing in popularity for many years now. Convolutional NNs (CNNs) have been proven to be particularly useful at image classification, among other tasks. One of the many fields NNs have had an impact on is the field of medicine, with . As the chest X-ray is one of the most commonly accessible radiological examinations for screening and diagnosing different lung diseases, it's an attractive source of data for projects involving NNs, as NNs require a large number of data samples in order for them to learn the underlying trends of the data. In this paper we present our project on training NNs to recognise pneumonia in chest x-rays.

1 Introduction

Machine learning and artificial intelligence have been making great strides in the field of medicine in the last few years. A key reason for this is that computers are much better than humans at learning from many Gigabytes of data in a very short time, whether this is finding patterns or trends in long lists of numbers or predicting tumour growth from looking at Computed Tomography scans. Given the right architecture and sufficient training, NNs have been known to surpass human performance at these tasks, as well as many more. The medical applications for CNNs include but not limited to: radiology, electronic signal analysis, disease diagnosis, and medical image analysis. In related work we discuss the papers we researched looking for previous projects undertaken where people used NNs while working with medical data or x-rays.. In Experimental Setup we discuss the preprocessing required before the different models used could be trained on the data set, as well as a description on each of the models used in the project. In Results we show the ROC AUC of each of the models when going through the training and validation data over 9 epochs.

2 Related Work

Our approach involves the task of object recognition within x-rays. In the following papers, we view related work involving NNs being used for medical problems or with x-rays.

Chest Pathology Detection. This discusses the ability of CNNs to learn mid-high level image representations. They use a basic feed-forward method to train their CNN where the intermediate layers receive the the features generated from the previous layer as input and send their output to the next layer as input. This paper describes a project very similar to ours: they're looking to train a NN to diagnose a disease (or lack of one) in patent chest x-rays. The only difference being the illnesses they're diagnosing are enlarged heart and/or enlarged mediastinum.

Using Deep Using Deep Convolutional Neural Network Architectures for Object Classification and Detection Within X-ray Baggage Security Imagery. One obstacle the authors faced was getting enough data to train the CNN they used. In the context of security x rays, there's a limited number of samples available to use for training a network, and a CNN requires a large number of samples to train it effectively. Their solution was to pre-train a CNN on general image classification

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

tasks for which there's more data available, then optimize at a later process towards the application of security x rays. For comparison, they also train a Support Vector Machine (SVM) classifier on the CNN features. This paper also used the AlexNet architecture when training the SVM which encouraged us to look at the AlexNet architecture as one of the models for our project. This is similar to our own project, although we'll be looking at medical x-rays, not security x-rays.

Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. A recommendation from our tutor, this article offers a solution to low precision and overfitting in NNs. The authors address the effect of excessive multiply-accumulative operations needed to compute the weighted sums of the neurons' inputs. The authors proposed a solution that compressed the network using Hash-Net, with a single function to randomly group connection weights and force them to share a single parameter value. The results of this change were improvements in power consumption and computation speed. This paper was useful to us because if we could make our network as accurate as a more complex network without the risk of overfitting or need for more computational power that would be very beneficial.

Optimal Brain Damage. Another recommendation from our tutor, this article addresses overfitting in a complex NN containing many weights. It describes a strategy where the least important weights are removed from a network to decrease the complexity of the network while maintaining its level of performance. Research has found that networks with too few weights are unable to represent the data trends accurately, while networks with too many weights don't generalise well and tend to overfit the data. A trade-off is needed to keep the weights showing the trends of the data and minimise the complexity of the network. The proposed solution is to delete parameters with the smallest saliency (the effect their deletion would have on the training error) and retrain the network. The end result is a network whose performance matches or exceeds the original but with half the weights, thus greatly reducing its complexity. The network captures the complex trends of the data without overfitting it. This article was an interesting read for us because similar to the Quantized Neural Networks paper, this suggested an approach reducing the number of weights in a network while maintaining a high performance level which could be very valuable as

ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localisation of Common Thorax Diseases. This paper used over 100,000 x-ray images from over 30,000 patients with 8 different disease image labels (a patient can have multiple diseases) to create and work with a database of x ray images to train a weakly supervised multi-label image classification and disease localisation framework. The authors of this article initialized a Deep CNN framework with 4 different pre-trained models: ResNet-50, AlexNet, GoogleNet and VGG, achieving the best results with the ResNet-50 model. They then experimented using ResNet-50 with different pooling strategies to initialise the Deep CNN framework. Between average looping, LSE pooling and max pooling they discovered LSE pooling achieved the overall best performance. This is very similar to our own project, the only differences being the models we'll use and the only disease we're looking at is Pneumonia.

3 Experimental Setup

There were several stages of preprocessing needed before we were able to train our models on our training data.

Firstly, our training data had less than 5000 samples, which is quite low when working with NNs. We needed to increase the number of samples we had in order to train the network over more data. We did this by including data augmentation in the preprocessing stage of the project, specifically flipping each image horizontally (giving a mirror like impression of the original image) which doubled the training data we had. Secondly, we needed to deal with an imbalance in the data set. We had about 1400 Normal images and about 4800 Pneumonia images in the training set so we needed some way getting approximately the same number of samples of each class (i.e closer to 1:1 than 3:1). We chose to use another feature of image augmentation to correct this imbalance.

For this project we were required to train at least three algorithms/models: 2 baselines and a proposed architecture. The models we chose were: fully trained AlexNet, ResNext with transfer learning, a Convolutional Neural Network (CNN), and a Multi-layer Perceptron (MLP).

AlexNet, designed by Alex Krizhevsky, is a very well known NN architecture. It competed in the ImageNet Large Scale Visual Recognition Challenge in Sep 2012 and achieved a top-5 error of 15.3 %, though it was later beaten by a deep CNN of 100 layers that belonged to Microsoft. It has quite a simple architecture, containing 8 layers: 5 convolutional and 3 fully connected. After the first 2

convolution layers there's a max pooling layer. An overlapping max pool layer is similar to a regular max pool layer which down samples the tensor width and height without altering the depth, but with overlapping, the adjacent windows over which the max is computed over lap each other. It uses the ReLu function instead of the TanH function as this allows the NN to train faster. It can also run on multiple GPUs.

The ResNext architecture is similar to that of ResNet, a popular architecture containing residual blocks in which intermediate layers of a block learn a residual function with reference to the block input. The difference between them is that instead of ResNet blocks it uses ResNext blocks which leverage a "split-transformation strategy". Instead of performing over the full input feature map, the block input is projected into lower dimensional representations, each one having a separate convolutional filter applied to it, before all the results are merged. The cardinality of a ResNext cell refers to the number of branches or groups in the cell. Research has shown that the performance of the model improves from increasing the cardinality of the cell rather than increasing the width or depth of the network.

A CNN is a special kind of multi-layer neural network designed to recognise different patterns from pixel images with minimum processing. They have countless applications in image/video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series, making them a natural choice for a project involving medical image classification. Their basic architecture consists of an input layer, output layer, and a number of hidden layers, usually convolutional layers that convolve with a multiplication or some other dot product. It usually uses the ReLu activation function.

An MLP is a typical kind of Artificial Neural Network (ANN). It contains a series of layers made up of neurons and their connections to other neurons. The neurons have the ability to calculate the weighted sum of their inputs and apply an activation function (often ReLu or TanH) to obtain a signal that's transmitted to the next neuron. The main difficulties of working with an MLP are training, and architecture optimisation. The architecture is very important as we saw in Optimal Brain Damage that if the network has too many weights there's too much complexity and it will over fit the data. If it has too few weights, it will not be complex enough to capture the underlying trends of the data. Training is also sometimes an issue as training an MLP on large datasets can be very time consuming, though our initial data set was relatively small so this wasn't as much of an issue.

The main hyper-parameters we looked at were batch size and the number of epochs we'd run the models for. We chose to have the same values for all the models because we noticed all the models maxed out in validation accuracy after 3-4 epochs. To avoid bias when comparing the results of the difference models we chose to have the same augmentation for each model. After experimenting a little with various batch sizes, we decided on a batch size of 16 for the models, as that was the highest we could go where the training could be done on our PCs in reasonable time.

4 Results

To measure the performance of the classification models we look at the Receiver Operating Characteristic (ROC) curve to visualize performance of these binary classifiers. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for the probabilities of the classifier predictions. The larger the area under the curve, the better the model is at distinguishing between the 2 classes. We primarily examined the training data results as this contained 1000s of samples, rather than the 10 or so that were in the validation data set. Might need to change that line..

In the Fig 4 we review the performance of CNN. We can see in the ROC graph for the validation data the models performance varies a little in the first few epochs, but shows improvement over time. Similarly in the graph measuring the loss, it's significantly higher in the first four epochs, but shows a downward trend from epoch 5 onwards. Although the curve for training data may improve slightly after the 10th epoch based on the upward trend observed in the graph, it's likely this improvement would be negligible, so we shall take 8 epochs as the optimal number for the CNN architecture and run that many for the test data. We note the extreme and seemingly random results in the loss graph for the validation data. We suspect this is due to the high variance of the model's prediction accuracy over such a small sample of x-rays.

In Fig 4, we review the performance of AlexNet. This model shows less improvement on the training data over all the epochs than the other modules, but it still has a very high ROC score overall. Similarly to the CNN model, it's results on the validation data seem quite volatile and random. Our explanation for this is the same as for the odd validation results in Figure 1.

In Fig 4 we observe the performance of MLP. This model saw a small but consistent improvement in both loss and ROC AUC over every epoch. There was also a considerable difference in both the loss graph and the AUC ROC graph in the performance of the training data and the validation data. Th most likely reason for this is due to the difference in the number of data samples in the training data and the validation data. With less samples in the model, there's more variance in the number of samples the model correctly classifies.

In Fig 4 we review the performance of ResNext. Though the performance of this model goes up and down several times over the first 10 epochs, there is a slight improvement in the AUC ROC overall. What's interesting is we see the validation curve mimicking the performance of the training curve, though in an exaggerated fashion, similar to the CNN ROC curves.. A slight increase in the training ROC curve often leads to a dramatic increase in the validation ROC curve.

Overall, we saw little further improvement in any of the models when trained past 10 epochs. When running the models over the test data, the ROC would often peaked at 5 or 6 epochs before declining from there, so we kept the max number of epochs to 10 as there was no need to have more than that. While most of the models show a consistent drop in loss between the first and 10th epoch, the CNN and MLP models seem to show the most significant improvement in ROC AUC by the 10th epoch, while AlexNet and ResNext tend to hover at the same level, though both still show excellent score in ROC AUC.
Fig 4 shows the final results attained by the trained models working with the test data.
It's interesting that the best model in performance was MLP, and the worst model was CNN. A possible explanation for this could be that MLP has the better immediate performance, but given time CNN responds better to training over many epochs. We can see this from Figure 3 showing the results of MLP against the training data. It has a very high initial ROC AUC score, but doesn't see much improvement after in the later epochs. On the other hand, CNN shows one of the lowest initial ROC AUC scores in the first epoch, but improves more than any model at the last epoch. The performance of AlexNet is rather surprising as it showed a high ROC AUC score from the first epoch in the training data and even showed a slight improvement in later epochs. If we'd had time to train the models over more than 10 epochs we may have seen different results.
The top 2 best performing models both showed relatively little variance compared to the others. What's interesting is that despite ResNet coming in second place behind MLP, it has a much lower variance than any other model observed in the project. Even the first place model MLP has more than twice the variance than ResNet does.
Obviously these results aren't the most accurate that have been seen. One data scientist managed to achieve an accuracy of 96% on the training data and 91.5% on the test data using a CNN with two convolutional layers, one hidden layer, a batch size of 32, running it for 25 epochs.

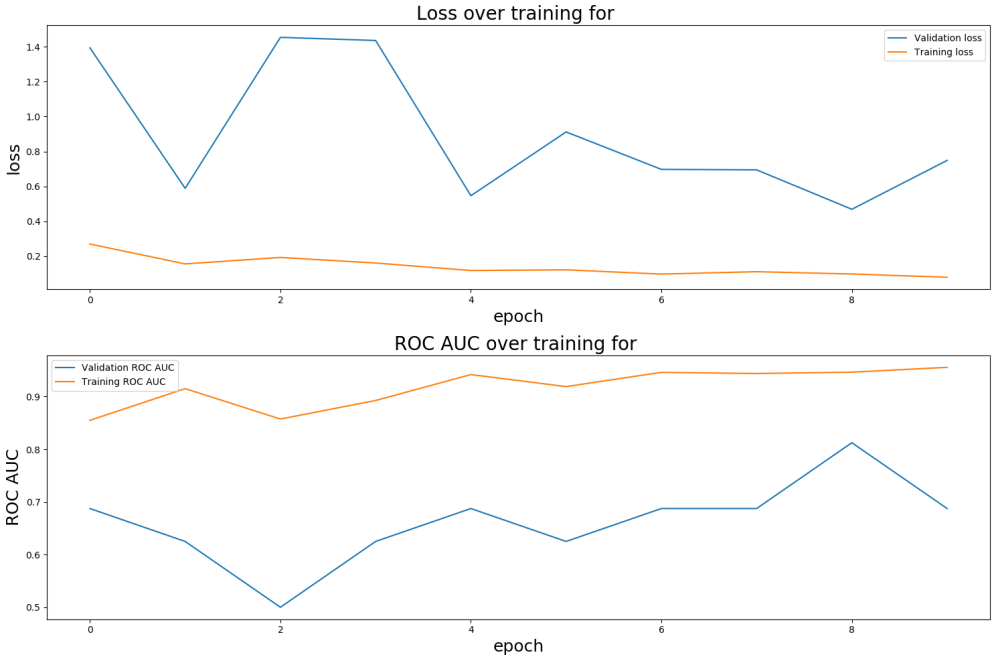

Figure 1: CNN results

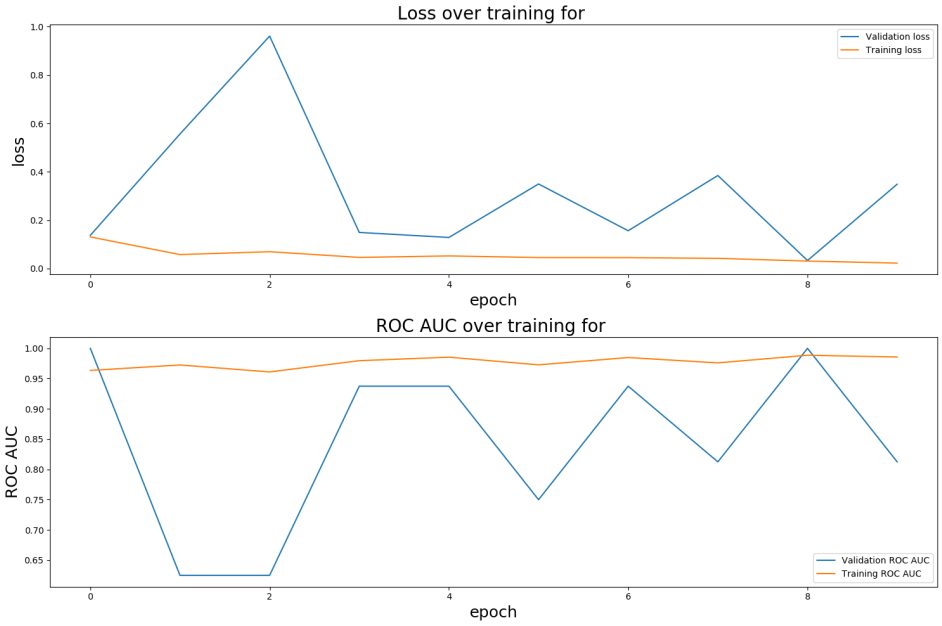

Figure 2: AlexNet results

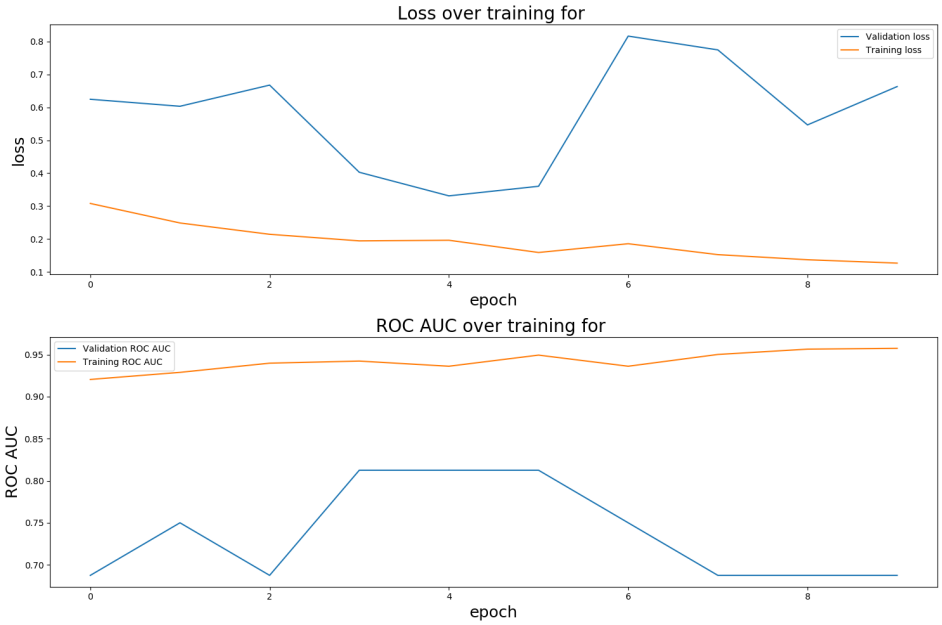

Figure 3: MLP results

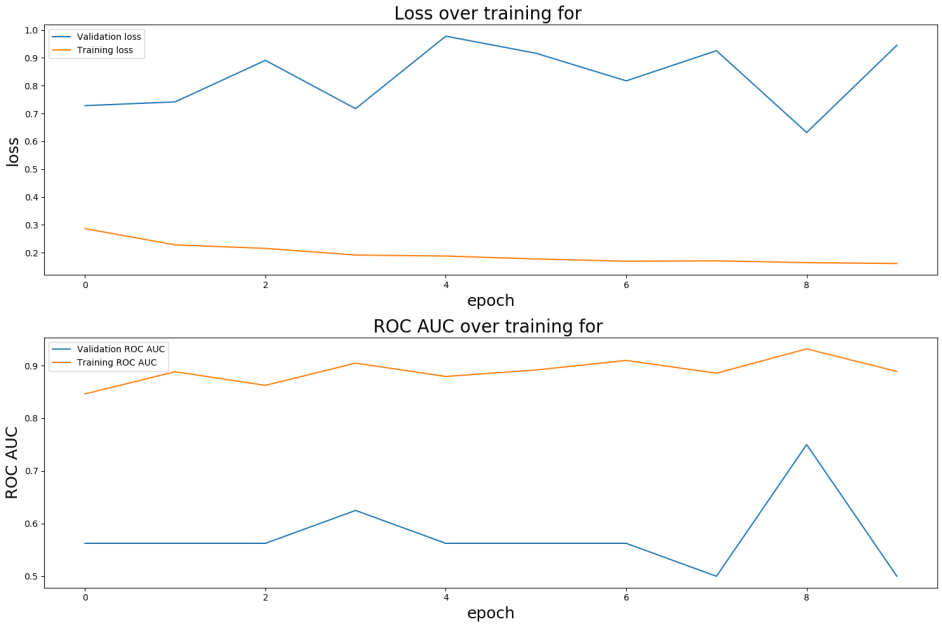

Figure 4: ResNext results

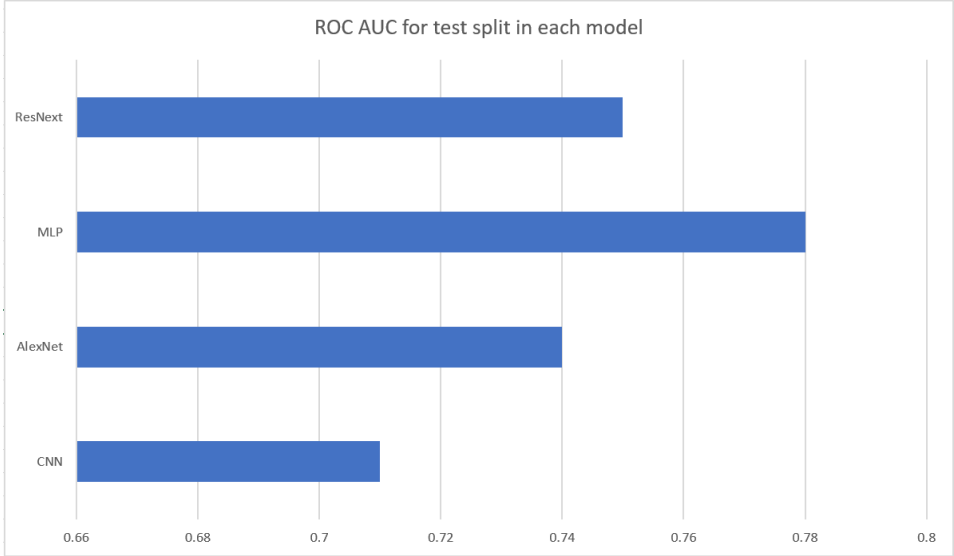

Figure 5: Test data model results

5 Conclusion and Future Work

Overall, our models performed quite admirably, despite being trained on what was originally a relatively small sample of data. We found that all 4 models achieved very high scores overall on the training data no despite running for just 10 epochs each, and when faced with the test data were able to correctly diagnose 3/4 of the samples. For future work it would be interesting to be able to train the models on the data for more epochs access to more computational power. In one of the articles we studied the author achieved a higher accuracy score than us in the testing data, having run his CNN model for 25 epochs. We learned from our research that too many layers and weights in a network leads to the model overfitting the data, and while it would be interesting to explore how much is too much for a given model it's impressive they were able to achieve an average accuracy score of 75% on the test data after running for just 10 epochs. Due to time constraints we couldn't do too much experimentation with different methods to deal with unbalanced data. We used horizontal flipping, center cropping and shift scale rotating, but one could also run tests with vertical flipping, scaling outward or inward, and different translations.

Acknowledgments

We thank our tutor Ellen Rushe for her guidance throughout the project and her recommendations for some of the papers in our research, lecturer Gianluca Pollastri for lecture notes dealing with overfitting, image classification, padding, and case studies involving the AlexNet

References

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M. and Summers, R.M., 2017. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2097-2106).

Bar, Y., Diamant, I., Wolf, L., Lieberman, S., Konen, E. and Greenspan, H., 2015, April. Chest pathology detection using deep learning with non-medical training. In 2015 IEEE 12th international symposium on biomedical imaging (ISBI) (pp. 294-297). IEEE.

Akcay, S., Kundegorski, M.E., Willcocks, C.G. and Breckon, T.P., 2018. Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery. IEEE transactions on information forensics and security, 13(9), pp.2203-2215.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. and Bengio, Y., 2017. Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research, 18(1), pp.6869-6898.

LeCun, Y., Denker, J.S. and Solla, S.A., 1990. Optimal brain damage. In Advances in neural information processing systems (pp. 598-605).

https://towardsdatascience.com/pneumonia-diagnosis-using-cnns-bfd71e3c05