COMP47650 Deep Learning Project: Pneumonia Chest X-Ray Classification

Robert McCarthy
Center of Biomedical Engineering
University College Dublin
Dublin, Ireland
robert.mccarthy@ucdconnect.ie

Abstract

Deep learning has achieved remarkable performances in image classification recent years but has generally relied on large, healthy training datasets. In medical imaging tasks available data is often very limited, especially when the pertinent disease is rare. In this paper, several convolutional neural network architectures are successfully trained to detect pneumonia from x-rays. This achieved using a dataset with only 5,216 training images and a 3:1 class imbalance. Techniques such as data augmentation, dropout, and oversampling are employed to tackle these dataset issues. It is found that favouring convolutional layers and minimising fully-connected layers can improve the efficiency and final performance of the model. Following this strategy, a model with only roughly 400,000 parameters is trained from scratch to achieve an AUC of 0.94 on the test data. Additionally, the use of transfer learning is found to be very effective in this data-scarce task.

1 Introduction

Advances in deep learning in the past 10 years have led to dramatic improvements in automated image classification, with human-level performance recently being surpassed [3]. Convolutional neural networks (CNNs) have been the essential tool in this achievement due to their superior efficiency over fully-connected neural networks (FCNs). This efficiency has allowed CNNs to tackle and essentially solve the complex image classification challenge offered by ImageNet [1]. AlexNet in 2012 [2], represented the first major deep-learning breakthrough in the ImageNet challenge, utilising a 8 layer CNN. Since then various innovations have allowed for deeper and deeper networks, serving to consistently improve performance [3,4,5]. In 2015, the massive 152 layer ResNet [3] surpassed human performance in the ImageNet challenge.

The diagnosis of many medical diseases relies on expert human interpretations of medical images. For example, radiologists use chest x-rays as the primary method to diagnose pneumonia. This process can be problematic for several reasons. Firstly, it requires the availability of an expert radiologist. Secondly, even expert human interpretations are not perfect and misdiagnoses are inevitable. Deep-learning has the potential to tackle these issues. A deep-learning model effective in diagnosing diseases from medical images could forego the requirement of an expert radiologist. More-over, if the models abilities were to exceed those of expert radiologists, patients could be offered improved healthcare. Indeed, several studies including CheXNet [6] have already presented deep-learning models capable of outperforming expert radiologists. However, like [2,3,4,5,6], ChexNet relied on a large training dataset - in this case 112,120 individual x-ray images. Often it can be difficult to obtain such a large labelled medical dataset to train with, especially if the disease is rare.

In this paper various CNN models are trained on a limited dataset to perform pneumonia diagnosis on chest x-ray images. The dataset used [7] presents difficulties due to its small size and significant class

Preprint. Under review.

imbalance. Several techniques are employed to overcome these issues. Data augmentation is used to enlarge the training data, helping to avoid overfitting and improve generalisability. The minority class (in this case non-pneumonia images) is oversampled to tackle class imbalance, discouraging the model from defaulting to predicting the majority class. In the main experiment of the paper, four different model architectures are trained and evaluated; two self-designed 6-layer CNNs trained from scratch, and two slightly different pretrained ResNet models used in a transfer learning approach. To aid model interpretability, the 'Class Activation Mapping' (CAM) [8] and 'gradient-weighted Class Activation Mapping' (Grad-CAM) [9] techniques are used to visualise model attention.

Results of the paper find that by favouring convolutional layers over fully connected layers, and minimising the number of fully connected parameters, performance can be improved and model efficiency drastically increased. Transfer learning is found to be a very effective approach in the pneumonia detection task. Better performing models are found to produce better CAM and Grad-CAM visualisation heataps - inline with expectations.

The remainder of the paper will proceed as follows: Section 2 reviews past approaches and the state-of-the-art in more detail; Section 3 details the methodology of the performed experiments; Section 4 presents the primary results of these experiments; and finally Section 5 discusses and draws conclusions from these results. An appendix is also included with further results and analysis.

2 Related Work

ImageNet [1] is an image database which takes advantage of the massive amounts of freely available images on the internet. It consists of millions of human-labeled high-resolution images, split into thousands of categories. Due its impressive scale, accuracy, diversity and hierarchical structure, the dataset has become a benchmark for computer vision tasks. In 2010 an annual competition using a subset of ImageNet was initiated called called the ImageNet Large-Scale Visual Recognition Challenge (ISVRC) [10]. This landmark competition has nurtured the dramatic improvements seen in image classification in recent years.

2.1 AlexNet

AlexNet [2] was the first major breakthrough in deep-learning image classification, achieving a winning top-5 error rate of 16.4% - a massive decrease from 25.8% in the previous year. This work took advantage of CNNs as easier to train models than standard FCNs due to their fewer connections and parameters. To overcome training time bottleneck issues with very large CNNs, optimized GPU implementations were developed to parallelize and speed up 2D convolution operations. This made feasible the training of the 60 million parameter AlexNet model.

The AlexNet model contained five convolutional (conv) layers and three fully-connected (fc) layers. Overlapping max-pooling layers layers were used after several of the layers in the network. The pooling layers serve to (i) increase robustness to noise, (ii) introduce a level of translation invariance to predictions, and (iii) progressively reduce the spatial size of feature maps to reduce the amount of parameters and computation in the network. Normalisation layers were also employed but these were later found to provide no benefit [4]. The ReLU non-linearity activation function was applied after every hidden layer of the net. ReLU was chosen as it is very computationally efficient and avoids saturation issues seen in other activation functions. The paper demonstrates that models can train much quicker with ReLU versus tanh. The benefits ReLU offers have seen it become a very popular activation since.

Despite the massive dataset being used, the large size of AlexNet made overfitting a problem. Two primary techniques were employed to combat this. Firstly, data augmentation was used. This serves to artificially enlarge the the dataset being used, thus making overfitting more difficult. Augmentation was achieved via random image translations and horizontal reflections, and random changes to the intensity of image RGB channels. The second technique used to combat overfitting was dropout [11], applied to the first two fully-connected layers of the net with a probability of 0.5. Without dropout the network was found to substantially overfit. However, it was also found that dropout doubled the number of iterations required for convergence.

Input images to AlexNet were 224x224 RGB. To update parameters, stochastic gradient descent was employed with a batch size of 128, momentum of 0.9 and weight decay of 0.0005. Along

with regularizing weight sizes, a small amount of weight decay was found to reduce training error. The learning rate was initialised at 0.01 and then divided by 10 when the validation error stopped improving.

It was found that depth of the network was very important to results - network performance degraded if any single conv layer was removed. However, at the time network depth was constrained by GPU memory limitations.

2.2 Convolutional neural networks post AlexNet – deeper is better

Since AlexNet, improvements have been made year-on-year in ISVRC using deeper and deeper CNNs. Training large CNNs is difficult for various reasons such as computational burdens and diminishing/exploding gradients. As such, each improvement in ISVRC has generally come with a proposed technique to allow extra depth be achieved in a feasible manner.

VGGNet VGGNet in 2015 [4] further reduced the top 5% error rate down to 7.3% by increasing their network to a size of 16-19 layers. The net was quite similar to AlexNet, but the increase in depth was made feasible using very small 3x3 conv filters at every conv layer in the network. This is unlike in AlexNet where larger filter sizes are used in the earlier layers of the net. By decreasing the filter size, more layers and fewer parameters could be used to achieve the same receptive field. This provides two benefits: (1) The extra layers introduced each come with a ReLU activation, thus non-linearity is increased and the decision function becomes more discriminative. (2) The reduction in parameters decreases memory burdens, allowing the network go even deeper.

In deep nets, bad weight initialisation can stall learning due to the instability of gradients. Thus, training of the larger VGGNets was performed in stages. First a network shallow enough to avoid issues with with random initialisation was trained. Then the early conv layers and fc layers of deeper networks were initialised with the layers of shallow net, allowing these deeper nets be trained successfully.

In VGGNet, the convolution stride of the 3x3 filters is fixed to 1 pixel, and the padding is also set to 1. This means spatial resolution is preserved after convolution. To reduce resolution, spatial pooling is carried out by five max-pooling layers. Maxpooling is performed over a 2×2 pixel window with a stride of 2, meaning the height and width of feature maps are halved. To prevent too much information being lost via this spatial compression, features map depth/the number of kernels is doubled after max-pooling. The 19 layer VGGNet had a total of 144 million parameters.

ResNet This work dropped the ISVRC top 5% error rate down to 3.57%, surpassing human performance in the task. The residual networks (ResNets) used to achieve this result have depths of up to 152 layers - x8 times deeper than VGGNets. Previously, performance was seen to degrade when adding further layers to a VGGNet. Here residual connections are used to avoid this issue, allowing drastic increases in depth prompting significant improvements in performance. These residual connections consist of identity shortcut connections which skip through the network. While very deep networks can struggle to simply learn an identity function, residual connections eliminate this issue and also help vanishing gradient issues. Thus, these deep ResNets are much easier to optimize than standard deep networks. Impressively, a 34 layer ResNet has lower complexity and fewer filters than a 19 layer VGGNet.

The ResNet approach adopts batch normalisation [12] after convolution layers and before activation. Batch normalisation serves to limit issues with exploding/diminishing gradient enabling networks with tens of layer to converge. Weights are initialised using Kaiming-He initialisaton [REF], improving learning over random initialisation. The techniques used here allow the extremely deep ResNets be trained from scratch, rather than in stages as in VGGNet.

Another notable architecture is that of GoogLeNet which made use of novel inception layers to increase computational efficiency by 28 times compared to VGGNet, allowing a network depth of 22 layers.

2.3 Applications in Medical Imaging

Deep learning models, like the ones discussed above, can be trained to detect pathologies in medical images such as x-rays. Recently, these techniques have been shown to surpassed expert human performance in diagnosing illnesses from medical images, for example in the case of CheXNet [6].

In CheXNet a deep CNN is trained to automatically detect pneumonia from chest x-rays. Here the network is a 121-layer Dense Convolutional Network (DenseNet) [14], trained on over 100,000 frontal-view X-ray images. To deal with class imbalance, the minority class (pneumonia images) was oversampled. Images were downscaled to 224x224 and normalised based on the mean and std of images in the training set. Training data was augmented via random horizontal flipping. To aid model interpretation, CAM [8] was employed to visualise areas of the image most indicative of the disease. CAM was made possible by the global average pooling that occurs after the final conv layer of the DenseNet.

3 Experimental Setup

3.1 Dataset

The 'Chest X-Ray Images (Pneumonia)' dataset from Kaggle [7] was used here. In its original state, this dataset provides 5,216 training images, 16 validation images, and 624 test images. Images are either labelled as 'normal' or 'pneumonia', where 'pnuemonia' indicates the presence of pneumonia in the x-ray. In the training data a significant class imbalance is found, with 74.3% pneumonia images present versus 25.7% normal images. Table 1 displays full information on the provided data split. The small size of the dataset and the class imbalance in the training data are notable issues which must be overcome.

Table 1: Provided data split

	Normal	Pneumonia	Total
Training	1341	3875	5216
Validation	8	8	16
Test	234	390	624

Note also that this dataset provides a problematic training/validation/test split. The validation set is incredibly small (about 0.3% the size of the training data) and is likely to introduce large variability in validation results, leading to sub-optimal assessment of the models ability to generalise during training. Upon further investigation of the validation data it is seen that only 'bacteria' pneumonia images are present, with a complete absence of 'virus' images. To obtain an improved validation set, a split could be performed on the training data. However, this introduces a data leak risk - in the training data most patients appear to have multiple x-ray images. Thus, in spite of its potential issues, the original data split is maintained for the main experiments of this paper.

A secondary set of experiments were also performed, in which the training data was split into a new 90/10 training and validation split, providing 537 validation images - a much healthier number. It was investigated whether this split could improve assessment of the models generalisability or negatively effect this assessment due to the aforementioned data-leakage.

3.2 Model Architectures

Experiments were performed with several model architectures. Overall, three CNNs were designed, implemented and trained from scratch. These architectures were largely inspired by AlexNet and VGGNet. Making use of transfer learning, experiments were also performed with a 50-layer ResNet (ResNet50) pretrained on the ImageNet dataset. This ResNet50 was used as a feature extractor to train two different fc layer configurations. These model architectures will now be described in more detail.

3.2.1 Models trained from scratch

When choosing sizes for the three self-designed architectures two key factors were accounted for. Firstly, the dataset being used is very small. This means that the ultra-deep state-of-the-art architectures seen in [3,4,5] would likely easily overfit the training data. Secondly, deeper models are more difficult and time-consuming to train. Taking these factors into account, it was decided to generally limit model depths to sizes similar to AlexNet. This model had enough learning capacity to perform excellently in the extensive dataset provided by ImageNet. Thus, it was reasoned that a similar sized model would have enough capacity to learn the much narrower task of pneumonia detection in x-rays.

Many of the design decisions made in these 3 models are inspired by VGGNet [4]. Following its findings, small 3x3 conv filters with the stride and padding of 1 were used for all conv layers. Max-pooling was applied after the activation of selected conv layers to reduce the feature map spatial dimensions. This was done with a 2x2 pooling window with a stride of 2, as in [4]. The number of kernels used is doubled after pooling to prevent excessive information loss. The ReLU activation was applied after each layer, as in [2,3,4,5,6], due to its computational efficieny and faster convergence. As overfitting is a significant concern in this small dataset, dropout was applied to the hidden fully connected layers of the models.

Techniques such as batch normalisation, residual connections, and inception layers were developed to allow very deep networks be trained. As such, these were deemed unnecessary for the relatively shallow networks used here.

Three separate networks were designed. These are referred to as SmallCNN, LargeCNN, and CamCNN. Table 5 in section 6.2 of the appendix shows full details of these architectures. Table 2 gives a summary of these architectures.

SmallCNN This model consists of 4 conv and 2 fc layers and is designed to be slightly smaller than AlexNet. The smaller depth here is chosen for the following reasons: (i) to prevent overfitting to the small dataset, (ii) to enable easier training, (iii) pneumonia detection likely requires a much narrower set of features to be learnt than in the ISVRC challenge. Thus, a shallower model may be adequate. Maxpooling is applied after all conv layers to suitably compress the size of feature maps to limit fc parameter count. Due to the smaller size of fc layers used here compared to [2,3], dropout of probability 0.25 is applied - reasoning that 0.5 may hinder learning too much. This model has roughly 4 million parameters, the vast majority of which are in the fc layers.

LargeCNN This model doubles up on SmallCNN. It consists of 8 conv and 2 fc layers. Its design is similar to the smallest model of [4], but slightly toned down with less kernels in the early conv layers and just two smaller fc layers. This toning down was to minimise the parameter count. The reasons for this larger, deeper model vs SmallCNN are as follows: (i) Results in literature have consistently shown increased depth to improve results. (ii) The features required to detect pneumonia may be particularly difficult to learn and so may require more conv layers than in SmallCNN. Due the escalated risk of overfitting in LargeCNN, dropout was increased to 0.5 in the fc layers. This model contains 72 million parameters in total.

CamCNN While the previous two architectures are relatively shallow, they still contain a very large number of parameters. The majority of these parameters are in the fc layers (see table 2). CamCNN is motivated as follows; conv layers have been shown to be much more efficient than fc layers in image classification tasks. Indeed, more recent architectures such as [3,14] have used minimal fc parameters by employing global-average-pooling after the last conv layer and mapping these pooled features directly to predictions. The CamCNN architecture follows this strategy. Compared to SmallCNN it has an exta conv layer but just one small fc layer. This serves to significantly reduce the number of parameters in the model. Despite the overall reduced parameter count, the extra conv layer in CamCNN could be very beneficial considering; (i) the findings in AlexNet that each conv layer improved performance, and (ii) the total receptive field of SmallCNN is quite limited. While improving efficiency, this reduced parameter count may also prevent overfitting. The model is termed CamCNN as it allows CAM visualisations.

Table 2: Summaries of model layers and parameters

	SmallCNN	CamCNN	LargeCNN	ResNet3fc	ResNet1fc
Conv Layers	4	5	8	49	49
FC Layers	2	1	2	3	1
Total Layers	6	6	10	52	50
Trainable					
Conv Params	97,440	392,608	4,666,048	0	0
FC Params	33,557,506	514	67,115,010	8,396,802	4,098
Total Params	33,654,946	393,122	71,781,058	8,396,802	4,098

3.2.2 ResNet50 - Transfer Learning

It has been shown that the features learnt by deep models on large datasets (e.g. ImageNet) tend to transfer well to new tasks [15]. Transfer learning takes a model pretrained in one task and fine-tunes it or uses it as a feature extractor in another task. This is particularly useful when available data is scarce in the new task. Here in an efficient transfer learning approach, a ResNet50 pretrained on ImageNet was obtained and used as a feature extractor i.e., its conv layer weights were frozen and only the new fc layers were trained. This approach assumes that the features learnt from ImageNet are applicable to the pneumonia classification task. Two different fc layer variations were tested:

Three fc layers (ResNet3fc) This configuration aims to compensate for the lack of trainable conv parameters by employing 3 fc layers, giving a total of roughly 8 million trainable parameters. To avoid overfitting, dropout of 0.25 was used. ReLU was employed as activation after the hidden layers.

One fc layer (ResNet1fc) This configuration seeks to maximise transfer learning efficiency by mapping features directly to probabilities. This gives a single fc layer with just 4,098 parameters.

3.3 Training Procedure

The final fc layer of all networks outputs two probabilities: one the probability that the image is 'normal', the other that it is 'pneumonia'. To train the models, mini batch gradient descent was used with a batch size of 16, as in [6]. This size is small enough to allow for quick gradient steps yet still allowing a good estimate of the gradient. Cross-entropy loss was used, as is standard. To gain access to a GPU, models were trained on Google Colaboratory.

To avoid overfitting to the small dataset, data augmentation was performed on the training images. Small random rotations and translations were applied, as well as random horizontal flips and small random changes in image brightness, intensity, contrast and hue. This encourages the model to focus on the features which are 'truly' indicative of the disease when making predictions - rather than unique features of individual images. Horizontal flips are a commonly used and effective augmentation technique [2,3,4,6]. Although the body is not horizontally symmetric, horizontal flips were applied in [6], a paper informed by expert radiologists. As such it was assumed safe to apply random horizontal flips in these experiments. Image pixel values were scaled between 0 and 1 and then normalized by a mean of 0.5 and standard deviation of 0.5 in each channel. Normalising the data ensures it is well distributed and in a similar range to initialised weights, helping stabilise and speed-up learning. Images were resized to 256x256 pixels before being input to the model. This size, similar to [2,4,6] but slightly larger, compresses the images to reduce the number of parameters required while still maintaining enough resolution to allow detection of fine features.

The order in which training data was sampled was randomised each epoch. This was to improve overall gradient estimation in the mini-batch gradient descent process. To tackle the class imbalance, the minority class (non-pneumonia images in this case) were oversampled as in [16], ensuring a roughly equal number of images from each class were sampled each epoch. This serves to weight each class equally during training, discouraging the model from biasing predictions towards the majority class.

To optimize the model Adam [16] was used with standard parameters (β_1=0.9 and β_2=0.999) and a learning rate of 0.001. Although SGD with momentum was experimented with, Adam tended to learn slightly quicker so was maintained for the main experiments. A weight decay of 0.0005 was applied, as in [2,3,4], helping regularise weight sizes to keep learning stable and avoid overfitting. The training, validation and test sets were used following standard supervised-learning procedure. Model weights were saved throughout training to allow test-set evaluation of the model which performed best on the validation-set.

3.4 Visualisations - Model Interpretability

Both the CAM [8] and Grad-CAM technique [9] were used to visualise model predictions. Grad-Cam was used for SmallCNN, LargeCNN, and the ResNet models. CAM was used for CamCNN, made possible by the global-average-pooling and single fc layer of the model. Note CAM could also be used by ResNet1fc but this is left for future work. Please refer to the referenced papers for further details of these visualisation techniques.

4 Results

4.1 Evaluation Techniques

The main measure used here to evaluate model performance on the test set is the Area Under the Receiver Operating Characteristic curve (AUC). This is a commonly used binary classification evaluation measure, often seen in works related to medical diagnosis. AUC indicates how well the model performs across all classification thresholds thus, is an excellent evaluation measure for this binary classification task.

In addition to AUC, some secondary evaluation measures are employed to allow a more in-depth analysis of model performance on the test-set. These are the accuracy, precision, recall, and F1 score of the model when predicting with the default threshold of 0.5.

4.2 Performance on Test Data

Now the results of the main experiment (using the original dataset split) are presented. The test results for each architecture are displayed in table 3. Note that LargeCNN has been excluded here as its training was very unreliable (refer to section 6.5 of the appendix for more details). Training curve etc. can be found in section 61. of the appendix.

Table 3: Test results

	SmallCNN	CamCNN	ResNet3fc	ResNet1fc
Accuracy	0.814	0.853	0.870	**0.878**
Precision	0.551	0.650	**0.816**	0.778
Recall	0.921	**0.938**	0.834	0.883
F1	0.690	0.768	0.825	**0.827**
AUC	0.924	**0.940**	0.936	0.937

CamCNN outperforms SmallCNN in every evaluation measure, including AUC. This is despite SmallCNN having roughly 85 times more parameters. This results demonstrates that conv layers are far more effective and efficient than fc layers when it comes to image classification.

SmallCNN achieved a training loss of 0.103 (97% accuracy) versus a test loss of 0.589 (81% accuracy). This is evidence of significant overfitting to the training data. On the test data its precision is very poor (0.551), and model predictions are heavily biased towards pneumonia (the majority class). It is likely the model learnt poor features for the following reasons; (i) it was limited to 4 conv layers only; (ii) The presence of a large number of fc parameters allowed overfitting to the training data, bypassing the need to learn better features. Visualisation results also demonstate this model to learnt poor features (see section 6.3 of appendix).

Compared to SmallCNN, CamCNN performs slightly worse on the training data but much better on the test data. This is likely because; (i) it has an extra conv layer, (ii) The combination of global pooling after the last conv layer (which greatly compresses feature information), with minimal fc parameters makes overfitting more difficult, forcing better features to be learnt. Impressively, CamCNN obtains the best AUC of all models (0.94). However, at the default threshold of 0.5 it still favours the majority class with a precision of just 0.650 versus a recall of 0.938.

The transfer learning ResNets perform impressively, demonstrating that features learnt from the ImageNet dataset are relevant to this task. Although these ResNets obtain a slightly worse AUC than CamCNN, their predictions are much more balanced across the two classes (see F1 scores). Interestingly, the ResNets perform worse in training than the shallow CNNs (see table 4 in appendix 6.1) but (mostly) perform better in testing. This indicates that the frozen conv layers provide very general features, preventing excessive overfitting and improving generalisability to the unseen test data.

The similarity in results between ResNet3fc and ResNet1fc demonstrate that the majority of the work is being done by the pretrained conv layers. In fact, the ResNet1fc performs slightly better, receiving the best accuracy and F1 score of all models - despite having the smallest number of trainable parameters by a large margin. This exhibits the power and efficiency of transfer learning, particularly useful in this case where models trained from scratch struggle due to a lack of data. The better performance of all models on in training versus the testing shows overfitting due to the lack of training data - despite measures employed such as dropout and data augmentation.

Despite oversampling, all models tend to bias the majority class in their predictions, especially the CNNs trained from scratch. This may be explained as follows; there are roughly 3 times less 'normal' images than 'pneumonia', and so the model sees these normal images 3 times more due to oversampling. Since there are very few of these 'normal' images (just 1,241), the model can overfit to these normal images quite easily. Rather than learning good features to identify pneumonia in images, the model can simply learn to predict 'pneumonia' if the presented image is not one of the 1241 'normal' images it is familiar with. As such, when presented with images the model is completely unfamiliar with in testing, predictions are greatly biased toward 'pneumonia'.

5 Conclusion and Future Work

In this paper, it is demonstrated that deep learning models can achieve high performance in x-ray pneumonia classification, even with limited training data. Techniques such as data augmentation, dropout, and oversampling helped avoid issues with the small dataset, but could not completely prevent overfitting. It was found that convolutional layers are far more effective than fully-connected layers in this task; a model with 5 conv layers, one fc layer, and 400,000 parameters performed significantly better than one with 4 conv layers, two fc layers, and 44 million parameters. Training a 10-layer model from scratch was found to be very difficult as no specific techniques were employed to account for its extra depth.

Transfer learning was found to be very effective. This allowed the high-quality features from a very deep model to be obtained without the difficulties of training the model from scratch. This is particularly useful when dealing with a small dataset, as is the case here. Using this transfer learning approach, an AUC of 0.937 was achieved with just 4,098 trainable parameters.

There are a number of avenues which could explored in future work. To improve performance, a deeper version of CamCNN with extra conv layers could be trained. This would be done employing techniques such as residual connections [3] and batch normalisation [12] to compensate for the extra depth. An interesting transfer learning experiment would be to fine-tune the pretrained ResNet50, rather than use it as a feature extractor. A model trained on a large database of x-rays rather than ImageNet could also be experimented with for transfer learning.

From the CAM and Grad-CAM visualisations, it appears that all trained models are often paying attention to irrelevant parts of the x-ray (i.e., areas away from the lungs) to some extent when making predictions. However, attention generally appears improved as model performance on the test set improves. A more thorough analysis of these model visualisation results is required.

References

[1] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.

[3] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[5] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[6] Rajpurkar, Pranav, et al. "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning." arXiv preprint arXiv:1711.05225 (2017).

[7] Chest X-Ray Images (Pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

[8] Zhou, Bolei, et al. "Learning deep features for discriminative localization." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9] Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE international conference on computer vision. 2017.

[10] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.

[11] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[12] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.

[13] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[14] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[15] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." arXiv preprint arXiv:1411.1792 (2014).

[16] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

6 Appendix

6.1 Main experiment results

Figures 1-4 contain the training and testing result plots for all four models from the main experiment (original data split).

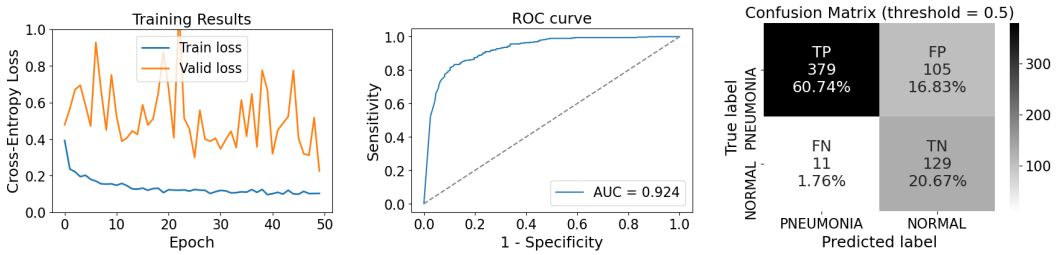

Figure 1: SmallCNN

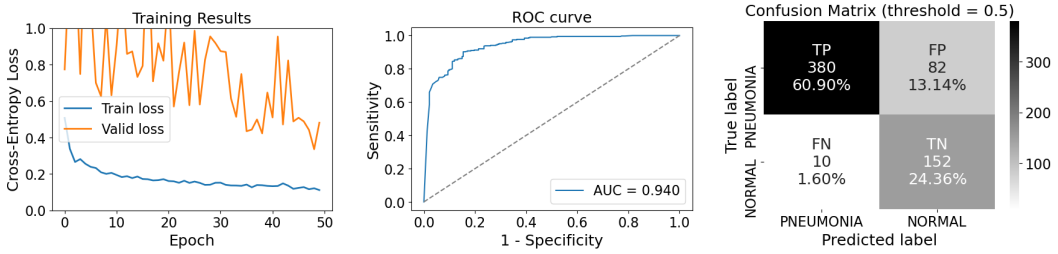

Figure 2: CamCNN

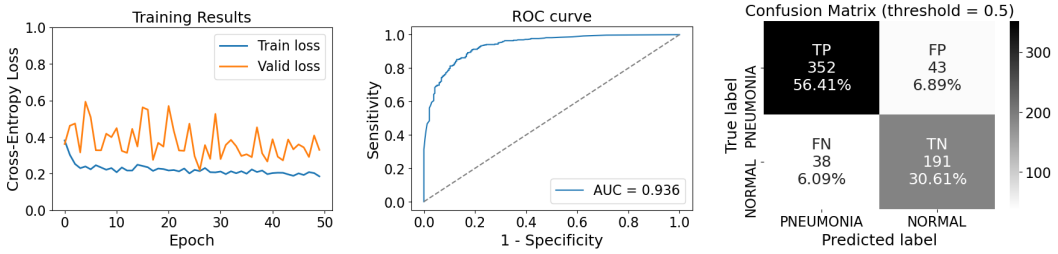

Figure 3: ResNet3fc

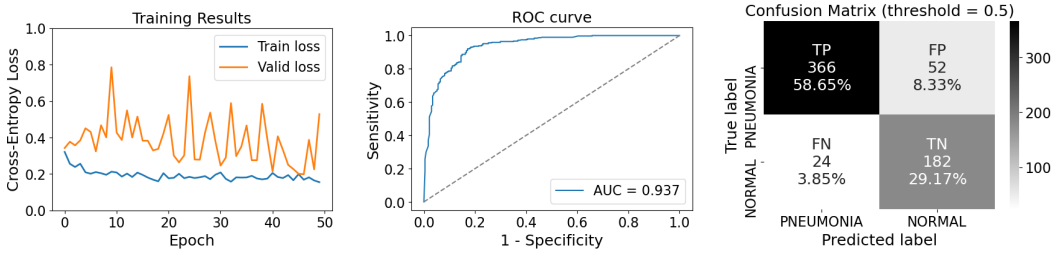

Figure 4: ResNet1fc

Table 4 contains losses and accuracy on the training, validation, and test data. Note this is for the best performing model for each architecture i.e. the model evaluated in the main experiment. Note also that the training results here are obtained from augmented training images.

Table 4: Model performance across all data

	SmallCNN	CamCNN	ResNet3fc	ResNet1fc
Loss				
Training	**0.103**	0.121	0.214	0.168
Validation	0.225	0.336	0.220	**0.198**
Test	0.589	0.446	0.419	**0.332**
Accuracy				
Training	**0.966**	0.965	0.948	0.958
Validation	0.875	0.750	0.875	**0.938**
Test	0.814	0.853	0.870	**0.878**

6.2 Model Architectures

Table 5 displays further details about each models architecture. Components are specified as follows; Convolutional layers: "conv<layer number>-<number of channels>". Fully-connected layers: "fc<layer number>-<nodes>". Dropout: "drop-<dropout probability>".

Table 5: Model Architectures

SmallCNN	CamCNN	LargeCNN	ResNet3fc	ResNet1fc
conv1-16	conv1-16	conv1-32	Pretrained	Pretrained
maxpool	*maxpool*	*maxpool*	ResNet50	ResNet50
conv2-32	conv2-32	conv2-64	minus	minus
maxpool	*maxpool*	*maxpool*	old fc	old fc
conv3-64	conv3-64	conv3-128	———	———
maxpool	*maxpool*	conv4-128	fc1-2048	fc1-2
conv4-128	conv4-128	*maxpool*	*drop-0.25*	
maxpool	*maxpool*	conv5-256	fc2-2048	
drop-0.25	conv5-256	conv6-256	*drop-0.25*	
fc5-1024	*maxpool**	*maxpool*	fc3-2	
drop-0.25	glob-pool	conv7-512		
fc6-2	*drop-0.25*	conv8-512		
	fc6-2	*maxpool*		
		drop-0.5		
		fc9-2048		
		drop-0.5		
		fc10-2		

6.3 Visualisations

Figure 5 display a example visualisation for each model. Generally, it was found that the better a model performed in testing, the better its heatmaps where. 'Better' heatmaps are more focused on patches of the lungs when predicting pneumonia. For the worst model, SmallCNN, the focus generally seems to be on the ribs and spine (see figure 5 (a)). ResNetfc1, the model which performed best in testing, appears to have the heatmaps which are most focused on patches of the lungs when making pneumonia predictions. However, even this model often pays attention to irrelevant areas. For example, in figure 5 (d) the upper left arm played a significant role in the pneumonia prediction.

6.4 Alternative data split

Figure 6 presents the training plots of a SmallCNN and ResNet3fc with the alternative data split. As a reminder, this split created by performing a 90/10 split on the original training data, in order to obtain more validation data. As can be seen in figure 6, the validation loss closely tracks the training loss. This is unlike when the original split was used, where validation loss was significantly higher than the training loss. This suggests that this alternative split has introduced data leakage. This is

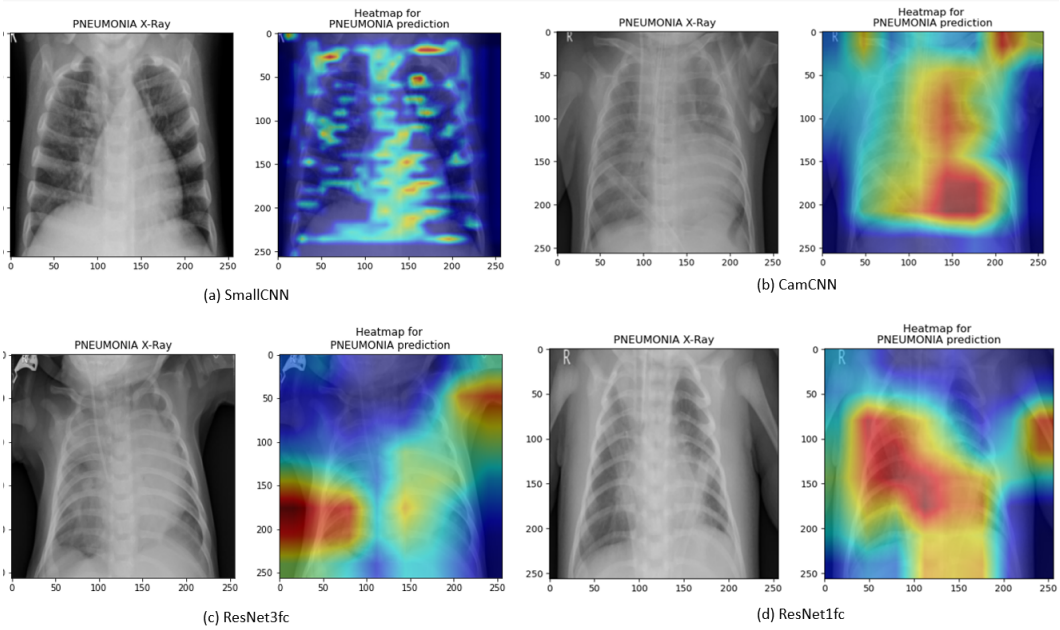

Figure 5: Sample heatmaps from each model

likely due to the presence of multiple images from each patient in the original training dataset. Thus, in the new validation set most patients also have images in the new training set. This means the new validation data is very similar to the training, explaining why performance on the two is similar.

In this alternative split the original test data was maintained. Interestingly, the best models from this alternative-split experiments performed slightly worse on the test-set than the best models from the original-split experiments. Table 6 compares testing performance (on the same test data) when model was trained with the original-split versus the alternative-split. The fact that the original-split models perform better on the test data displays that the original validation-set gives a better measure of generalisability than the new validation-set - this makes sense as there is data leakage in the new validation set so measurement of generalisability is poor. However, it is also possible that the slight drop in performance is due to the fact that the alternative split has 10% less training data

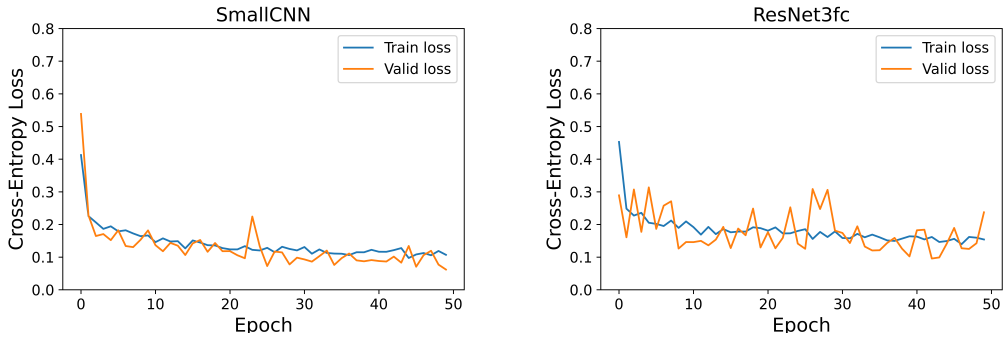

Figure 6: Training results with the alternative data split

6.5 LargeCNN

LargeCNN proved quite tricky to train. Figure 7 displays training results over 3 separate runs. Here the only difference between runs was the optimizer used, as specified in the legend. In one run it learns straight away, in another does not start learning till the 25th epoch, and in the third run in does not learn at all. These issues are likely due to the depth of the model. Indeed, LargeCNN did not

Table 6: Test Performance: Original vs Alternative Split

	SmallCNN	ResNet3fc
Accuracy		
Original	**0.814**	**0.870**
Alternative	0.808	0.854
AUC		
Original	**0.924**	**0.936**
Alternative	0.911	0.929

incorporate any techniques such as batch normalisation or He initialisation to account for its depth. Further investigation into exactly why its training was so inconsistent is required.

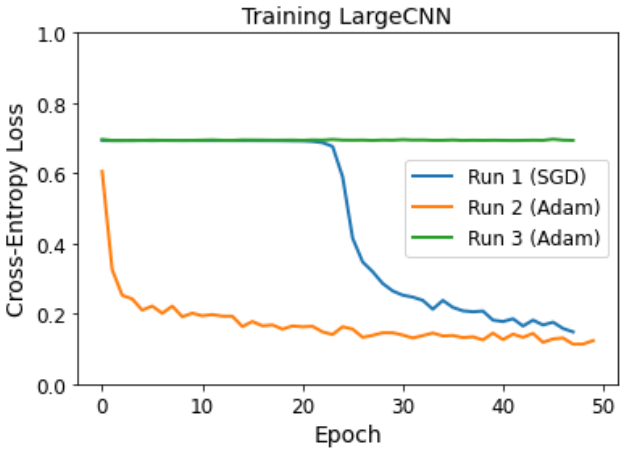

Figure 7: Training loss of LargeCNN in three seperate runs