Data Analytics : X-Ray Pathology Identification using Neural Networks

Joseph Dawtry
ID: 180749506
j.d.dawtry@se18.qmul.ac.uk

Jonathan Hind
ID:110238414
j.hind@se11.qmul.ac.uk

Sandeep Walia
ID:180864669
s.s.walia@se18.qmul.ac.uk

Matthew Lee
ID:180789269
m.d.lee@se18.qmul.ac.uk

April 17, 2019

Student	Effort	Student	Effort
Joseph Dawtry	25%	Jonathan Hind	25%
Matthew Lee	25%	Sandeep Walia	25%

1 Introduction

Computer vision has potential medical applications, especially in image classification and object detection. The applications in Radiology are clear as the field requires highly skilled analysis of various images to diagnose a plethora of pathologies. Within this field, X-rays are one of the most common diagnostic images used. Combining expert knowledge with machine learning methodologies [1] paves the way for automated diagnosis software that could yield far-reaching rewards in health-care. The aim of this project is to produce a neural network that is able to identify pathologies in X-ray imagery. Section 2 discusses the literature of Machine learning in Radiology while Section 3 presents the Data management. Section 4 explains Neural Networks. Section 5 discusses the Methodology and Section 6 contains analyses and evaluates of results . Finally, Section 7 presents our conclusions and Section 8 gives the opportunities for Future Work.

2 Literature Review

The availability of large, labelled high-resolution image datasets, such as ImageNet [2], have supported recent breakthroughs in image classification. Prior to

this, there were only a handful of small labelled datasets available, consisting of many common problems such as noise, low resolution and a small number of categories. ImageNet [3] and datasets alike, have allowed more robust and complex models to be created, and have inspired further research across all image related domains.

Convolution Neural Networks (CNN) are the state-of-the-art for image classification & object recognition tasks and are known for their ability to produce strong results on highly challenging image datasets. CNNs are often used in other fields such as Natural Language Processing when semantically modelling sentences [4], and for sentence level classification [5].

The first use of neural networks for medical image analysis, can be traced back to the 1990's, where they were used in digital mammography to classify normal tissue from tissue which has been affected by calcifications [6]. CNNs require large amounts of labelled data in order to provide state-of-the-art results, and their recent popularity is helped by the shift to using graphical processing units (GPUs) which allow faster computation. CNNs are commonly applied to medical images due to their ability to extract features and this has led to many medical developments ranging from detection of Diabetic Retinopathy [7], to Lymph node metastasis [8].

As neural networks have many parameters, typically millions for image classification, and with limited data available for most medical analysis [9], it is possible that the network may not be able to learn any discriminative features. Because of this, it is common to use a CNN which has been previously trained on a different dataset. Shie et al [9]. investigated transfer learning as a potential solution to this problem. The approach used in the study consists of four steps:

1. An unsupervised construction of a 'codebook' from ImageNet, which contains learned feature maps using a variant of AlexNet [3].

2. Encoding the medical images using the codebook.

3. Using the learned weights to perform supervised learning on the labelled medical images using Support Vector Machines.

4. Adding heuristic domain-expert engineered features to the features learned during transfer learning to increase accuracy.

Shie et al. concluded it is possible to learn underlying structures from an unrelated image dataset and transfer the knowledge, to be used in other problems. The results obtained from applying the transfer learning features outperformed the human-created features, indicating that the features learned from transfer learning may be more robust and discriminative.

Kim et al. [10] also identified that the lack of sufficiently sized labelled datasets as an issue and transfer learning can offer assistance during training. However, it was found that the features extracted from training on a large, unrelated, dataset are biased and may not perform well on medical data. The proposed solution was to train the bias weights on a bridge dataset consisting of

radiographs in order to reduce the amount of bias. It was concluded that using a related bridge dataset could boost performance. However, when the bridge dataset was of a different type to the target dataset (e.g. X-ray to MRI, or MRI to CT), performance was significantly worse than the direct transfer learning, without the use of the bridge dataset. This indicates that the bridge dataset has to be selected carefully in order to improve performance.

DenseNet121 is a specialisation of CNNs. It was proposed by Huang et al. [11] who showed networks can be deeper, more efficient to train and produce higher accuracies if short connections close to both the input and the output are used. DenseNet121 alternates between convolutional layers and a novel 'Dense block' which is formed of multiple fully connected layers, see Figure 5. Traditional neural networks contain a single connection between neighbouring layers. Similar to this project, the original DenseNet121 paper experimented with image classification. The input to DenseNet121 is an image and the output is a probability distribution across all 1000 ImageNet classes. In general, DenseNets were shown to significantly reduce the number of parameters and mitigate the impact of vanishing-gradient.

CheXNet [12] applies DenseNet121 to chest X-ray diagnosis. The CheXNet paper aimed to detect pneumonia from chest X-rays, as a binary classification task. There were two required modifications to DenseNet121. A single output layer was put in place of the final fully connected layer, and a sigmoid activation function for the output layer.

3 Data Management

3.1 Data Source

The dataset that was used for this project was collated by Irvin et al. [1] at Stanford University. The dataset was released on January 23rd 2019. It should be considered ambitious to work with such newly released data. It contains over 225,000 chest X-rays of 65,000 patients, collected from Stanford hospital, between October 2002 and July 2017. Each X-ray image, is accompanied by: Age, Sex, Frontal/Lateral, Orientation and a vector of 14 pathology labels. A list of the different pathology can be found in Figure 1. These 14 pathology labels were created using a rule-based labeller, which labelled images based on the radiology report corresponding to the patient, in line with the Fleischner Society's glossary [13].

Pathology	Positive (%)	Uncertain (%)	Negative (%)
No Finding	16627 (8.86)	0 (0.0)	171014 (91.14)
Enlarged Cardiom.	9020 (4.81)	10148 (5.41)	168473 (89.78)
Cardiomegaly	23002 (12.26)	6597 (3.52)	158042 (84.23)
Lung Lesion	6856 (3.65)	1071 (0.57)	179714 (95.78)
Lung Opacity	92669 (49.39)	4341 (2.31)	90631 (48.3)
Edema	48905 (26.06)	11571 (6.17)	127165 (67.77)
Consolidation	12730 (6.78)	23976 (12.78)	150935 (80.44)
Pneumonia	4576 (2.44)	15658 (8.34)	167407 (89.22)
Atelectasis	29333 (15.63)	29377 (15.66)	128931 (68.71)
Pneumothorax	17313 (9.23)	2663 (1.42)	167665 (89.35)
Pleural Effusion	75696 (40.34)	9419 (5.02)	102526 (54.64)
Pleural Other	2441 (1.3)	1771 (0.94)	183429 (97.76)
Fracture	7270 (3.87)	484 (0.26)	179887 (95.87)
Support Devices	105831 (56.4)	898 (0.48)	80912 (43.12)

Figure 1: The 14 pathologies and their corresponding class presence in the dataset [1]

Each pathology label has three possible values: 1 - confidently present, 0 - confidently absent, u - present with uncertainty, and NA - illness having no mention in the radiology report. The labeller was set up in three stages: 'Mention extraction' - returns a list of observations from the summary of findings from the patient's radiography report, 'Mention classification' - mentions are classified as negative, uncertain, or positive. 'Mention aggregation' - the classification(s) and the mentions return a final label for the observations. There are many X-ray datasets available, [14] [15] [16], but CheXpert was chosen over other datasets due to it being comprehensive in scale, labelled by experts and publicly available.

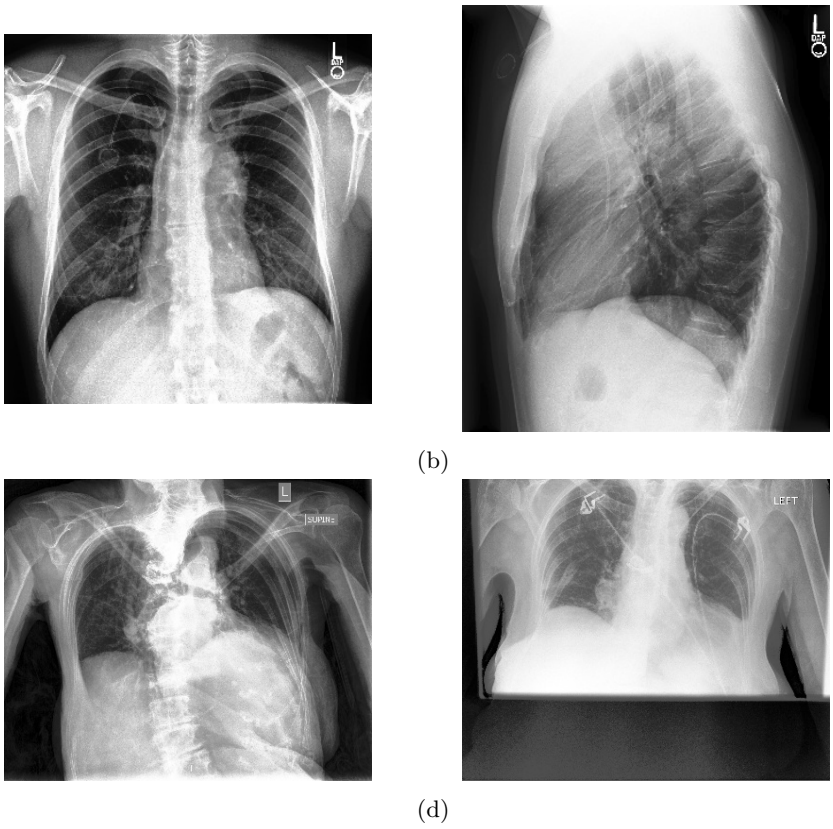
3.2 Ethics

The CheXpert dataset contains patient data, which has been de-identified so that under federal definition 45 CFR 46.102(e) [17] it is not considered to be *research with human subjects*, therefore does not have to comply with the federal statute set down by the United States Department of Health and Human Services in the Policy for Protection of Human Research Subjects. As the dataset still contains detailed and sensitive information about patients, it needs to be treated with appropriate care and respect.

3.3 X-ray Images

When conducting experiments, the lower resolution X-rays from the downsampled dataset (11GB) were chosen over the high-resolution X-rays (439GB). This is due to the level of computational resources which are required to use the

high-resolution X-rays. As there is no standard size of X-ray imagery, as can been seen in Figure 2, the dataset contains X-rays of different dimensions, pre-processing of the images will be required to create a 'standard' size. This is discussed further in Experimental Studies.



(a) (b) (c) (d)

Figure 2: (a) Frontal (PA) X-ray, (b) Lateral X-ray, (c) Frontal (AP) X-ray orientation, (d) Frontal (PA) X-ray with partial occlusion, demonstrating the range of quality in the dataset.

X-rays are taken with different orientations depending on the patient and the symptoms being investigated. Frontal is when the X-ray is taken from front-to-back or back-to-front, and lateral is side-to-side. In this project we focus on frontal images, specifically AP - Antero-Posterior (front-to-back) and exclude PA - Postero-Anterior (back-to-front). Only frontal (PA) scans are being used due to the memory limit imposed. This means there is a limit to the amount of images that can be inputted to the model. As we were limited in the amount of images that could be inputted, it makes sense to use images that are consistent as input to the model. The downside to this approach is that our model will only be trained using frontal images, and will perform poorly on images taken

from other orientations.

3.4 Preliminary Analysis

Preliminary analysis is important in every data analytics project. We have performed preliminary analysis to find any possible biases and any interesting characteristics of the dataset. These can be found in the Appendix A and helped inform our decisions throughout the project.

4 Neural Networks

Neural Networks are a machine learning technique, inspired by the human brain. A neural network consists of layers, and in recent times, new, more sophisticated layers have been developed. The simplest form of neural network is a Multi-layer Perceptron (MLP), consisting of three layers - an input layer, a hidden layer which contains fully connected neurons, and an output layer. Data is passed into the network via the input layer, generating a signal which is then passed onto the hidden layer. Neurons in the hidden layer compute a linear weighted combination of its input and applies an activation function to it. The hidden neurons output the result of their respective activation functions, which feed forward to the output layer. The output layer performs another linear weighted combination on the signal, which is then passed into its activation function to provide the final output of the MLP. Example output for a simple MLP could be a binary classification, a 0 or 1 output. The process of passing inputs through a neural network to produce an output is known as Forward Propagation. The weights used in the linear combinations performed by the neurons are the parameters of the model to be optimised and determine the outputs it produces.

Neural networks, like many machine learning algorithms, learn by calculating the difference between its last prediction and the ground-truth. The error produced is used to update its parameters to generate a better prediction next time. The inputs in a neural network pass through many layers and neurons before transforming into an output, this means the process of updating the parameters is more complex as the relationship between the weights in the first layers and the outputs is not intuitive. To simplify the problem we break it down by layer.

Firstly, the output error is calculated and the parameters are updated for the output layer. Equation 1 shows how the differentiated activation function is used to calculate the weight change.

$$\delta_j = h^{'}(a_j) \sum_k w_{kj}\delta_k \tag{1}$$

Weight change for neuron j, where δ_j is the weight change to minimise error, $h^{'}$ is the differentiation of the activation function, a_j is the weighted combination

of inputs into this neuron, w_{kj} are the weights applied to inputs from neuron k to the neuron j, and δ_k is the error for next layer's neuron k.

The result of Equation 1 is used to update the weights of the last layer, using a learning rate, α, to temper the change. See equation 2.

$$w_{ij} = w_{ij} - \alpha \delta_j x_i \tag{2}$$

Weight update equation, where w_{ij} is the weight applied between neuron i and neuron j, α is the learning rate, δ_j is the error for neuron j and x_i is the input into neuron i.

Once this process has been completed for the last layer, we can then apply it to the layer before. In this way we propagate the error backwards through the network. This process is called back-propagation [18], please see Figure 3.

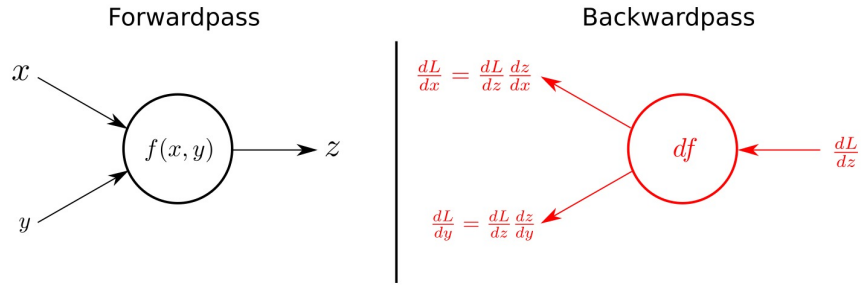


Figure 3: Visualiation of a Forwardpass and a Backwardpass - Backpropagation [19]

CNNs are consistently used in modern state-of-the-art image classification, segmentation and object detection tasks. Convolutions provide a method of transforming an input into a smaller representation, known as feature maps. The aim of the convolution is to find interesting patterns within the input by passing a kernel over it. Kernel sizes are a hyper-parameter that can be defined to determine the area of the image to be considered in a single window. A kernel of size 3x3 will have 9 weights which are learned during the training of the network. The 3x3 window will slide over the image, multiplying the value in the image with the corresponding weight of the kernel, as seen in Figure 4. Each value in the resulting matrix is summed and becomes a position in the, lower dimension, feature map.

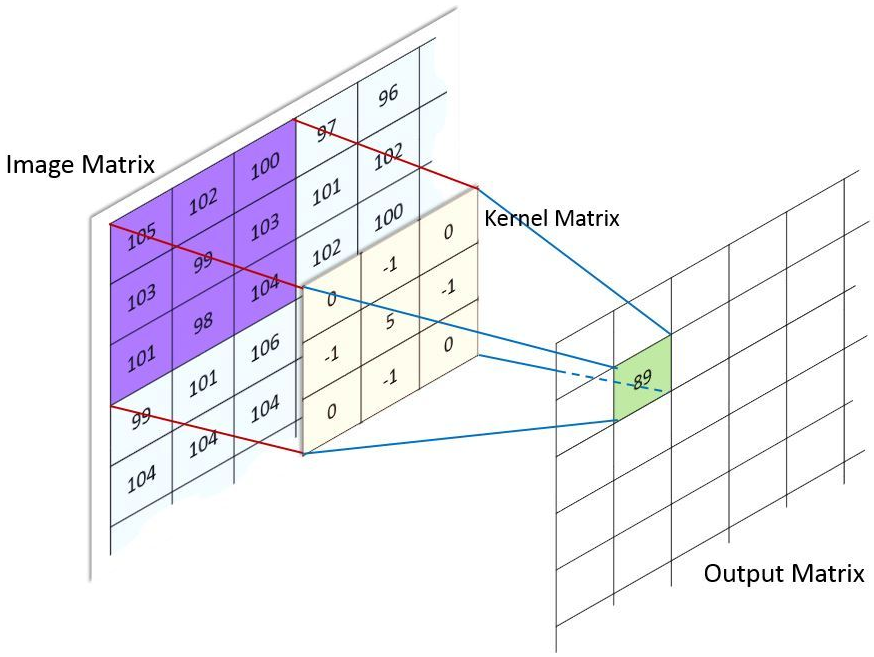


Figure 4: Convolution Operation that is Performed in a Convolutional Layer [20]

5 Methodology

5.1 Data Preprocessing

Data was read in as a Pandas Dataframe. The data contains many 'Nan' values, which represent pathologies not mentioned in the patient's report. 'Nan' values were replaced with '0', as this indicates that the patient did not have the pathology. Additionally, uncertain labels (-1) were replaced with '1', indicating that the patient has the pathology. This approach is justified by the research done by Irvin et al. [1] that showed the highest AUROC was achieved when changing uncertain labels to 1. The training/validation split used is 75:25 and this is stratified, which keeps the classes balanced across both datasets.

5.2 Hardware

Experiments were run with Python 3.6.5 on a Ubuntu server with an Intel(R) Core(TM) i7-4790K CPU , 4.00GHz, with 32GB of RAM and a GeForce GTX TITAN X. Compared to other state-of-the-art papers, this is modest and frequent insufficient-memory issues influenced our design decisions.

5.3 Predicting One Pathology

We decided to predict one pathology at a time, because it would be challenging for our model to classify 14 pathologies simultaneously. For example, if for each pathology our model can predict with 90% accuracy, then the probability of predicting all pathologies simultaneously would be 0.9^{14} (0.2288). This is also more in line with medical practice which will test for one pathology at a time. This process of predicting one pathology is repeated for multiple pathologies and their corresponding results are reported in the Results section. For full descriptions of the pathologies please see Fleischner Society's recommended glossary [13].

5.4 Data Augmentation

Data Augmentation synthetically increases the size of the dataset by generating variations of existing images. We applied up to a 15 degree rotation and 15% height/width translation to the images. The values are small enough that the model should still be able to detect a pathology in a particular location, and also large enough that the model should be translation/rotation invariant. This means that the model should be able to locate patterns, even if they have moved or rotated slightly.

5.5 Image Preprocessing

The DenseNet121 model requires specific image dimensions. The images were resized to be 224x224 and also with 3 channels, where each channel repeats the original black and white image. This is an unfortunate but necessary step for DenseNet121 transfer learning.

5.6 Imbalanced Classes

The dataset classes are greatly imbalanced, especially for the 'No Finding' label as only 10% of the data was positive for this. This causes problems in machine learning and tends to lead to bad performance. This would lead the model to be biased towards the majority classes and may lead to misclassification of the minority classes. The imbalanced classes can be re-balanced and majority classes can be removed, or, class weights can be used to focus more on the under-represented minority classes. We created a 50:50 split between patients that have the pathology and those that don't have any pathology. This has the disadvantage of reducing the size of the samples that can be taken in.

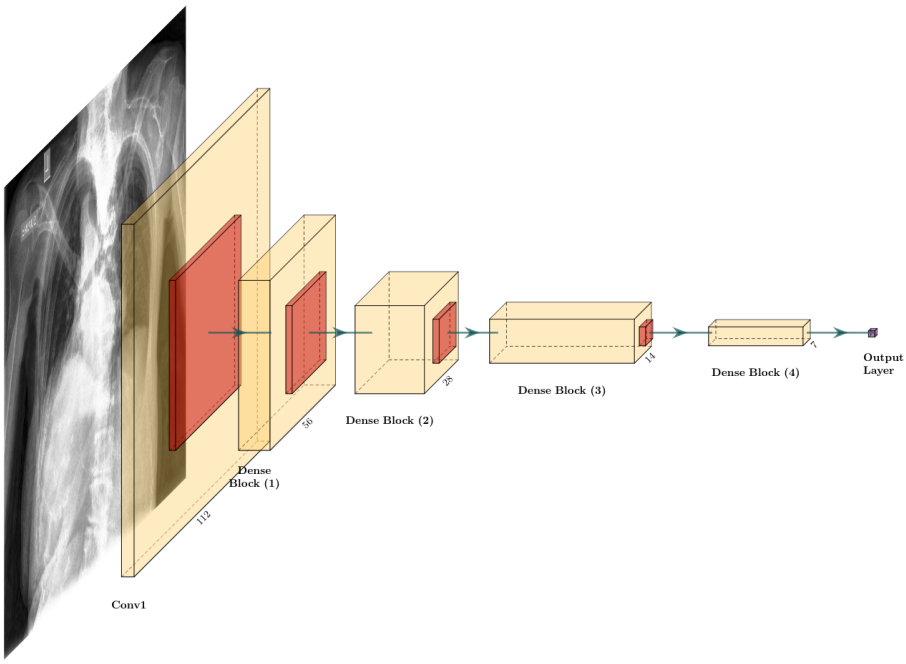


Figure 5: DenseNet121 Architecture: This shows 6 parts of the DenseNet121. Conv is a convolution layer followed immediately by a pooling layer (red). A dense block is a series of convolutional layers and fully connected layers implemented as described in [11]. The combination of these units creates a model with 429 layers, including 121 convolutional layers.

5.7 Architectures Implemented

Many different architectures were implemented for this project. A description of the main contributing architectures is provided below, along with the observations made. Cardiomegaly makes a large change to the entropy of the image, which means it is often easier to detect [14], therefore it was used as a baseline to compare architecture performance.

The first architecture used transfer learning from DenseNet121 with untrainable layers, which meant all the weights were frozen. A dropout rate of 0.3 was used to prevent overfitting that may occur. A single sigmoid unit was used in the output layer to predict the presence of one pathology, and this is the only layer that was trainable. A training accuracy of 70% and validation accuracy of 56% was achieved after 10 epochs, however it plateaued after this point, probably due to the model having a small number of trainable parameters (1000).

The next model froze only the first 200 layers, this meant that the network took longer to train as there were now 4 million parameters in the model. We did this to capture low-level features such as edges and textures, which are generic

to all kinds of images. The last layer of DenseNet121 was replaced with a single sigmoid unit, instead of the 1000 class softmax. By removing the last layer, the number of parameters to be trained was reduced by 1 million. The last layer is an expensive layer which is usually used to classify ImageNet images, (eg. ImageNet classes - horse, plane, cat). This obtained a training accuracy of 95%, but a validation accuracy of 54%. It was apparent that overfitting was occurring here. As it was not feasible to increase the training dataset to more than 20,000 images due to memory issues, other measures were used to prevent overfitting, such as dropout layers and reducing trainable parameters.

In an attempt to further reduce the amount of trainable parameters, the number of untrainable layers was increased from 200 to 300. This resulted in an improved training accuracy of 98%, however the validation accuracy converged at approximately 50%. The number of untrainable layers was further increased to 400 (575,000 parameters), providing a training accuracy of 80% and validation accuracy of 57%.

The observed validation accuracy led us to believe that we needed more data, however, due to computational limits we were not able to use all of the training images. The solution proposed was to use Data Augmentation as described above, Section: 5.4. The accuracies achieved were 70% for training and 60% for validation, however, these were still not satisfactory results.

Influenced by Rajpurkar et al.[12], all the layers were set to be trainable. As well as this, the batch size was set to be 16. Using a small batch size leads to more backpropagation passes and the model is much slower to train as a result. This resulted in a validation accuracy of 76%, which represented a good score compared to results achieved by other papers[21, 14, 12], and was chosen as our final architecture.

6 Results and Evaluation

The 'train.csv' file was used for our model's training and validation data, and here we present results against the test dataset which was taken from the 'valid.csv' file. To analyse our results we have used the Area Under the curve of Received Operating Characteristic (AUROC) as the metric and the Received Operating Characteristic (ROC) curves. These were chosen as they are able to show the efficacy of the models with a range of confidence thresholds, and allow comparison with similar studies such as:[21, 14, 12].

For brevity, we report the results for 5 pathologies rather than all 14: Cardiomegaly, Edema, Atelectasis, Pneumonia, Pneumothorax. We chose these as they are shared between the CheXNet dataset and the CheXPert dataset and had good performance by the CheXNet model [12].

We present the AUROC scores in Table 1. Generally our results fall below those of the previous papers. We believe this is largely due to the lack of computational resources. With additional memory, we would have been able to train on more images, which would have helped improve our model's performance on the test data. Notably our model outperforms Wang et al. [14]

Pathology	Wang 2017 [14]	Yao 2017 [21]	CheXNet [12]	Our Model
Cardiomegaly	0.807	0.904	**0.925**	0.702
Edema	0.835	0.882	**0.888**	0.758
Atelectasis	0.716	0.772	**0.809**	0.679
Pneumonia	0.633	0.713	**0.768**	0.689
Pneumothorax	0.806	0.841	**0.889**	0.646

Table 1: AUROC scores for our model with previous papers for comparison. N.B. The datasets used in each paper are not the same, so direct comparison is not possible.

for Pneumonia. As the datasets are different, it would not be valid to compare directly, however Table 1 does give an approximate measure of the different model's performances. Ideally, cross-validation would have been used, however, this increases the processing time by K (folds), which was not viable given the time constraints.

Figures 6-10 present the ROC curves for the different pathologies. We have presented these so a prospective user of the models would be able to choose a compromise between True Positive Rate (TPR) and False Positive Rate (FPR). We expect that for the medical domain, most users would choose a threshold which prioritises TPR maximisation over FPR minimisation, as it is normally worse to miss a pathology that is present than wrongly diagnose someone that does not have a pathology.

The AUROC values indicate how the models perform overall. Our best AUROC was for Edema, which has a ROC curve that runs closer to the top left corner. Our worst was Pneumothorax, which barely makes it over the diagonal and at one point is even below it. We were also able to present the scores of real radiologists as reported in [1]. None of our models were able to outperform as radiologist, however the Edema model came quite close. There is a noticeable difference in the number of steps in each curve as well. The larger steps occur when increasing the threshold value by one increment, reclassifies a large number of the data points, resulting in a significant change to TPR and FPR. This occurs when confidence values are grouped into small clusters rather than a smooth distribution. e.g. For Pneumothorax, there are 9 steps, because its confidence values fall into 9 small groups.

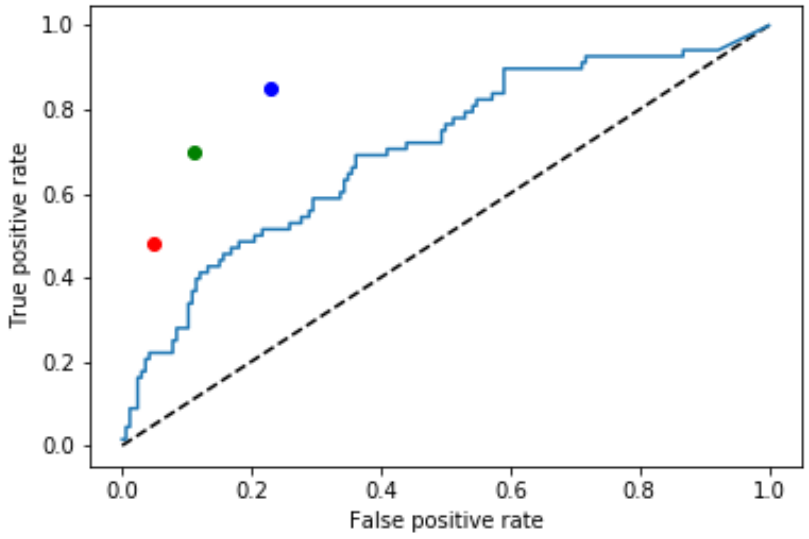


Figure 6: Cardiomegaly AUROC - 0.702. The green, blue and red dots represent the scores of radiologists as reported in [1]

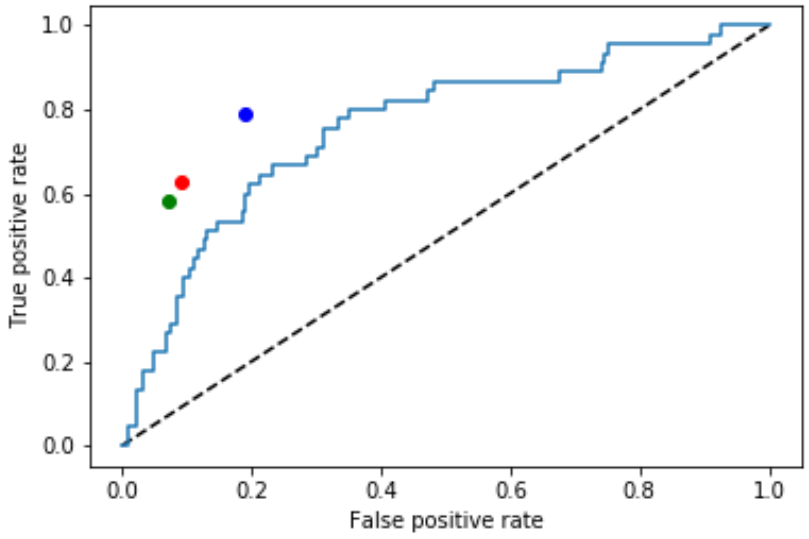


Figure 7: Edema AUROC - 0.759. The green, blue and red dots represent the scores of radiologists as reported in [1].

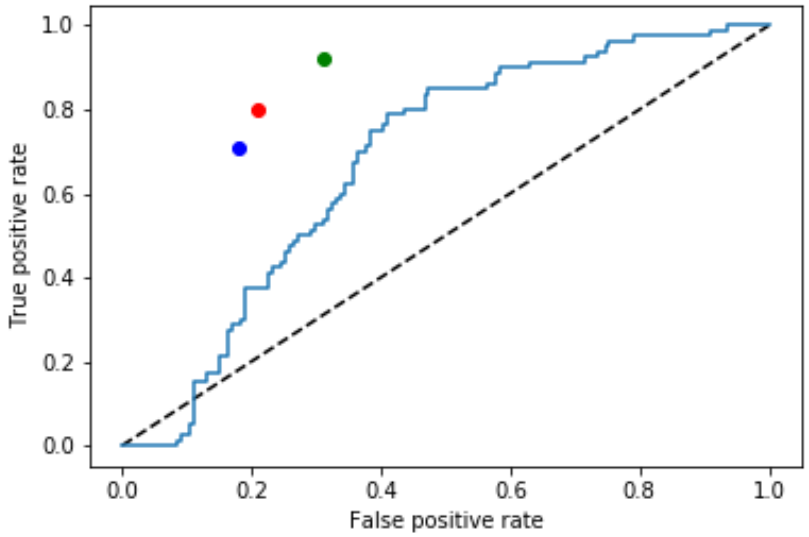


Figure 8: Atelectasis AUROC - 0.679. The green, blue and red dots represent the scores of radiologists as reported in [1].

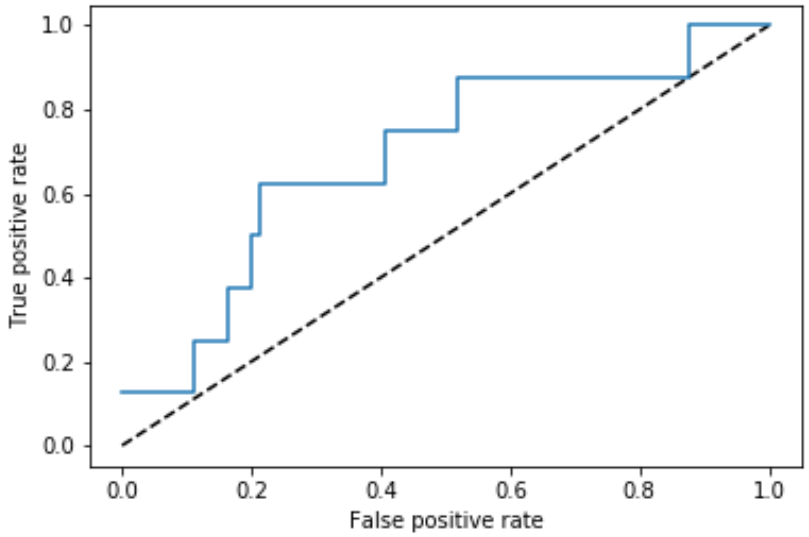


Figure 9: Pneumonia AUROC - 0.689. Radiologist scores were not available for Pneumonia.

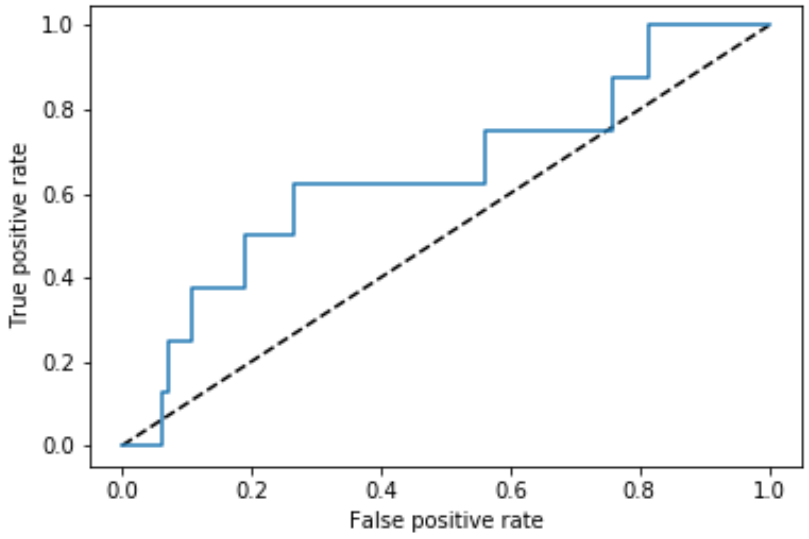


Figure 10: Pneumothorax AUROC - 0.647. Radiologist scores were not available for Pneumothorax.

7 Conclusions

This project aimed to produce a model capable of detecting a given pathology in a patient based off a frontal (PA) X-ray and we achieved this aim. Using weights transferred from ImageNet, a deep convolutional neural network model was created for each of the 5 target pathologies: Cardiomegaly, Edema, Atelectasis, Pneumonia, Pneumothorax. The model did not perform as well as state-of-the-art models had performed on similar datasets, however it was close enough that the approach is clearly valid. Applying the improvements suggested in Section 8 and using computers with more memory, it might be possible to match or exceed previous models. It was not able to outperform radiologists, despite this, the model could be useful for hospitals that do not have access to skilled radiologists, and is significantly cheaper to run than employing a full-time radiologist.

We intend to submit our model to the Chexpert competition and our results should be visible on the leaderboard here:`https://stanfordmlgroup.github.io/competitions/chexpert/`.

8 Future Work

One major limitation of the project was the lack of computational resource, particularly memory, which caused us to train on less data and limited the

data augmentation performed. Future work, that had more computer memory available to it could train on more data, perform more data augmentation and could even train on a combined dataset. E.g. combining the ChestX-ray14 dataset used in previous papers [14] with the CheXPert dataset.

We used validation accuracy to choose which epoch to save the model's weights from. However, a model that has a better accuracy, will not necessarily have a better AUROC as the accuracy is based off a 0.5 threshold. Future work should customise the weight checkpointing to be based off validation AUROC rather than accuracy.

We found that the validation accuracies of the models were very unstable, with one epoch of training making as much as 20% difference. This is probably caused by having a small batch size and relatively large learning rate. We were not able to decrease the learning rate due to time constraints, however future work could do, in order to achieve more stable validation accuracies.

We focused on frontal (PA) images, however this limits the possible applications of the model. Future work could use other orientations and may benefit from the orientation itself being an indicator of patient condition [22].

9 Group Work

Group work within the team was achieved by using collaborative source control software such as Git, hosted on GitHub. The report writing was primarily done using Overleaf, a collaborative, LaTeX, report writing website.

Weekly meetings were held to present work done individually and discuss any issues that were encountered. Tasks were assigned during the meetings to ensure that all members were aware of what was required of them.

Open communication was achieved by using Slack to ensure that there was a medium of communication that was convenient and real-time.

All members of the team contributed in all areas, although there were some areas of focus for each team member.

References

[1] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *arXiv preprint arXiv:1901.07031*, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[5] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[6] Wei Zhang, Kunio Doi, Maryellen L Giger, Yuzheng Wu, Robert M Nishikawa, and Robert A Schmidt. Computerized detection of clustered microcalcifications in digital mammograms using a shift-invariant artificial neural network. *Medical Physics*, 21(4):517–524, 1994.

[7] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[8] Jeffrey Alan Golden. Deep learning algorithms for detection of lymph node metastases from breast cancer. *Jama*, 318(22):2184, 2017.

[9] Chuen-Kai Shie, Chung-Hisang Chuang, Chun-Nan Chou, Meng-Hsi Wu, and Edward Y Chang. Transfer representation learning for medical image analysis. In *2015 37th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 711–714. IEEE, 2015.

[10] Hak Gu Kim, Yeoreum Choi, and Yong Man Ro. Modality-bridge transfer learning for medical image classification. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5. IEEE, 2017.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[12] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

[13] David M Hansell, Alexander A Bankier, Heber MacMahon, Theresa C McLoud, Nestor L Muller, and Jacques Remy. Fleischner society: glossary of terms for thoracic imaging. *Radiology*, 246(3):697–722, 2008.

[14] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. 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*, pages 2097–2106, 2017.

[15] Alistair EW Johnson, Tom J Pollard, Seth Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. Mimic-cxr: A large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019.

[16] Stefan Jaeger, Sema Candemir, Sameer Antani, Yì-Xiáng J Wáng, Pu-Xuan Lu, and George Thoma. Two public chest x-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative imaging in medicine and surgery*, 4(6):475, 2014.

[17] Ecfr - code of federal regulations. `https://www.ecfr.gov/cgi-bin/retrieveECFR?gp=&SID=83cd09e1c0f5c6937cd9d7513160fc3f&pitd=20180719&n=pt45.1.46&r=PART&ty=HTML#sp45.1.46.a`. (Accessed on 03/28/2019).

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Mayank Agarwal. Back propagation in convolutional neural networksintuition and code — becominghuman.ai. `https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199`. (Accessed on 03/28/2019).

[20] Convolutional neural network — brilliant math & science wiki. `https://brilliant.org/wiki/convolutional-neural-network/`. (Accessed on 03/28/2019).

[21] Li Yao, Eric Poblenz, Dmitry Dagunts, Ben Covington, Devon Bernard, and Kevin Lyman. Learning to diagnose from scratch by exploiting dependencies among labels. *arXiv preprint arXiv:1710.10501*, 2017.

[22] John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric K Oermann. Confounding variables can degrade generalization performance of radiological deep learning models. *arXiv preprint arXiv:1807.00431*, 2018.

[23] Numpy reference manual website. `https://docs.scipy.org/doc/numpy/reference/`. (Accessed on 03/29/2019).

[24] Python data analysis library pandas: Python data analysis library. `https://pandas.pydata.org/`. (Accessed on 03/29/2019).

[25] Tensorflow reference manual. https://www.tensorflow.org/api_docs/python/tf. (Accessed on 03/28/2019).

[26] Keras reference manual. https://keras.io/. (Accessed on 03/28/2019).

A Appendix

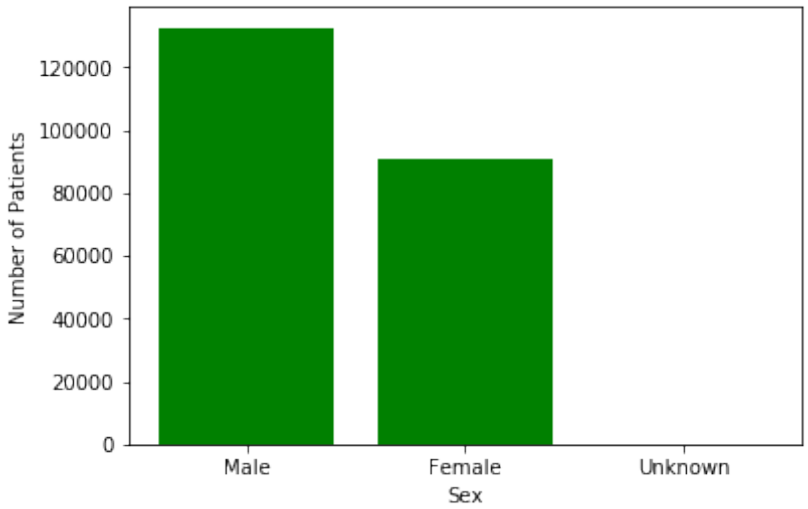


Figure 11: Gender

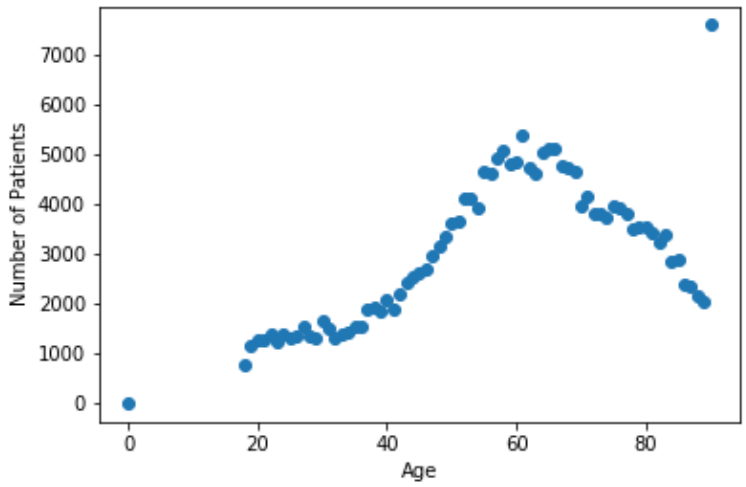


Figure 12: Age

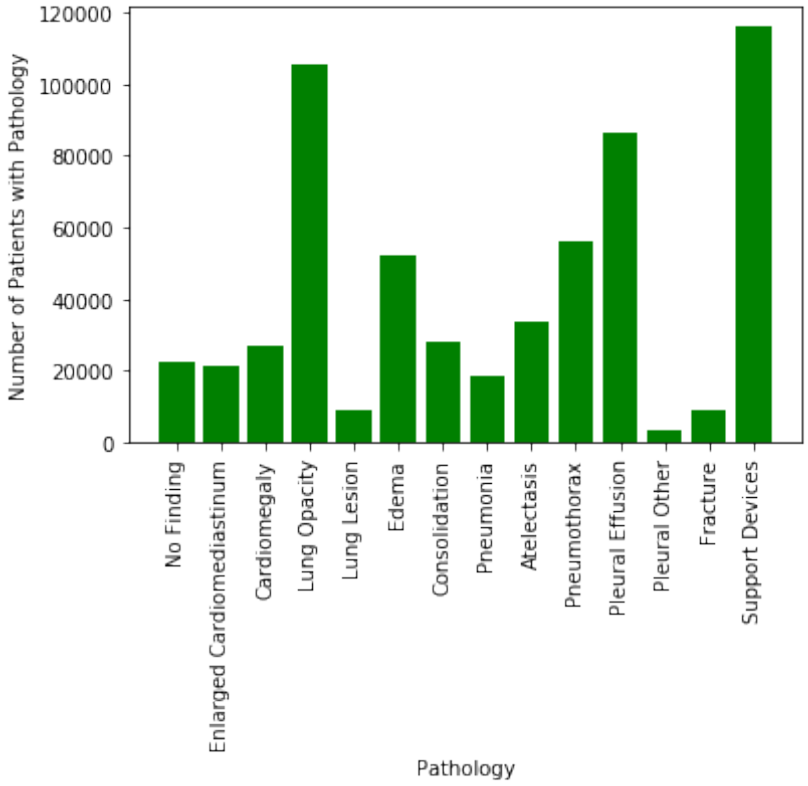


Figure 13: Pathology

B Appendix

Software Packages

Numpy is a Python package that provides implementations of methods commonly used in machine learning. In addition to these methods, Numpy provides optimised data structures developed to work with large, complex data. [23]

Pandas, similar to NumPy, is a python package. It was created to work with large datasets. It offers data structures and functions design to manipulate tabular data. [24]

TensorFlow is a package, developed by Google, to perform common operations used within machine learning and is widely used to implement neural networks. TensorFlow is available to utilise GPU hardware to optimise operations. When performing operations using 'large data' on CPU it can take a long time to compute. The recent utilisation of GPUs has resulted in a gain of computational efficiency in 'large data', especially in computationally heavy models such as neural networks [25].

Keras is neural network package that is built on top of TensorFlow. Keras provides high-level abstractions of low-level, complex API operations found in TensorFlow. The package became popular due as it offered a more user-friendly environment [26].

C Appendix

Find below a sample of data the data this is the first three rows on the train data each of these Path's began with CheXpert-v1.0-small/train/

Path	patient00001/ study1/ view1_frontal.jpg	patient00002/ study2/ view1_frontal.jpg	patient00002/ study1/ view1_frontal.jpg
Sex	Female	Female	Female
Age	68	87	83
Frontal/Lateral	Frontal	Frontal	Frontal
AP/PA	AP	AP	AP
No Finding	1		
Enlarged Cardiomediastinum			
Cardiomegaly		-1	
Lung Opacity		1	1
Lung Lesion			
Edema		-1	
Consolidation		-1	-1
Pneumonia			
Atelectasis		-1	
Pneumothorax	0		
Pleural Effusion		-1	
Pleural Other			
Fracture		1	1
Support Devices	1		

D Appendix

Code:

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_files
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from tqdm import tqdm
from PIL import Image
from random import shuffle


import keras
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dropout,
Flatten, Dense
from keras.applications.densenet import DenseNet121
from keras.models import Sequential, Model
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator


import numpy as np
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve
import os
import subprocess
```

In [ ]:

```
trainDf = pd.read_csv('CheXpert-v1.0-small/train.csv')
```

In [ ]:

```
# Remove anomalous dataline
trainDf = trainDf[trainDf.Sex != 'Unknown']

def pathToID(path):
    pathList = path.split('/')
    return pathList[2][7:]

def pathToStudy(path):
    pathList = path.split('/')
    return pathList[3][5:]

# Convert all labels to a series of one-hot encoded labels.
# -1 is uncertain, 0 is negative, 1 is positive, nans are no mention of the dise
ase in the text
trainDf = trainDf.fillna(0)
# N.B. this is replacing unknowns with true as per u-ones model here: https://ar
xiv.org/pdf/1901.07031.pdf
# This is essentialyl saying that if we're not sure of disease we say they have
 it.
# Just to be on the safeside and have better recall as we care more about recall
than precision
trainDf = trainDf.replace(-1,1)


# Onehot encode the sex and the xray orientation
trainDf = trainDf.replace('Male',1)
trainDf = trainDf.replace('Female',0)
trainDf = trainDf.replace('Frontal',1)
trainDf = trainDf.replace('Lateral',0)

trainDf =trainDf.rename(index=str, columns={"Sex": "Male?",'Frontal/Lateral' :'F
rontal1/Lateral0'})


#trainDf.insert(0,'Path', trainDf['Path'])
trainDf['Study'] = trainDf.Path.apply(pathToStudy)
trainDf['Patient ID'] = trainDf.Path.apply(pathToID)

# Rearrange Columns
cols = ['Patient ID', 'Study', 'Path', 'Age', 'Male?', 'Frontal1/Lateral0', 'AP/
PA','No Finding',
       'Enlarged Cardiomediastinum', 'Cardiomegaly', 'Lung Opacity',
       'Lung Lesion', 'Edema', 'Consolidation', 'Pneumonia', 'Atelectasis',
       'Pneumothorax', 'Pleural Effusion', 'Pleural Other', 'Fracture',
       'Support Devices']
trainDf = trainDf[cols]
```

Preliminary Analysis

In [ ]:

```
# Shows age distribution of the data set. There are 3 0-olds and 7579 90 year ol
ds.
# Implies that over nineties were grouped together
ages = trainDf['Age'].value_counts()
plt.scatter(ages.keys(),ages.values)
```

In [ ]:

```
# How many people have no disease?
no_finding = trainDf['No Finding'].value_counts()
print(no_finding)

# Plot pie chart to show how much of the data is labelled with each character
values = no_finding.values
labels = ['Disease present','All Clear']

# Plot
plt.figure(figsize=(10, 5))
plt.title('Data Proportions', size=20)
plt.pie(values, labels=labels, # explode=explode,
        autopct='%1.1f%%', shadow=False, startangle=45)

plt.axis('equal')
plt.tight_layout()
plt.show()
```

In [ ]:

```
def load_dataset(path):
    # Load image files
    data = load_files(path)
    files = np.array(data['filenames'])
    return files


def path_to_tensor(img_path,inputSize):
    # loads RGB image as PIL.Image.Image type
    img = image.load_img(img_path, color_mode = "grayscale", target_size=inputSi
ze)
    # convert PIL.Image.Image type to 3D tensor with shape (x, x, 1)
    x = image.img_to_array(img)
    data = np.asarray( img, dtype="int32" )
    # convert 2D tensor to 3D tensor with shape (1, X, x) and return 3D tensor
    return data.reshape(1,inputSize[0],inputSize[1])

def paths_to_tensor(img_paths, inputSize):
    # Convert paths to tensors for keras
    list_of_tensors = [path_to_tensor(img_path, inputSize) for img_path in img_p
aths]
    return np.array(list_of_tensors)


def path_to_tensor_channel_last_3colour(img_path,inputSize):
    # loads RGB image as PIL.Image.Image type
    img = image.load_img(img_path, color_mode = "grayscale", target_size=inputSi
ze)
    # convert PIL.Image.Image type to 3D tensor with shape (x, x, 1)
    x = image.img_to_array(img)
    data = np.asarray( img, dtype="int32" )
    # convert 2D tensor to 3D tensor with shape (X, x, 3) and return 3D tensor
    # Repeat the same image 3 times to create 3 channels for densenet
    return np.stack((data,)*3, axis=-1)


def paths_to_tensor_channel_last_3colour(img_paths, inputSize):
    # Convert paths to tensors for keras
    list_of_tensors = [path_to_tensor_channel_last_3colour(img_path, inputSize)
for img_path in img_paths]
    return np.array(list_of_tensors)
```

In [ ]:

```
inputSize = (224,224)

sample_size =20000 # 30k is the memory limit for the server

# Choose pathology to investigate
targetColumn = [15]
colName = trainDf.columns.tolist()[targetColumn[0]]
print(f"This model will be targetting {colName} column")


# Create balanced dataset with 50% pos examples and 50% neg examples, only take
 scans from the front
pos = trainDf[(trainDf[colName] == 1) & (trainDf['Frontal1/Lateral0'] == 1 ) & (
trainDf['AP/PA'] == 'AP')]
neg = trainDf[(trainDf['No Finding'] == 1) & (trainDf['Frontal1/Lateral0'] == 1
) & (trainDf['AP/PA'] == 'AP')]

posSample = pos.sample(int(sample_size/2))
negSample = neg.sample(int(sample_size/2))
sample = pd.concat([posSample,negSample])
x_train_paths, x_val_paths, y_train, y_val = train_test_split(sample.Path, sampl
e[colName], stratify=sample[colName], random_state =2)

# The 3 channel option is required for the denseNet and other transfer learning
 models
# Single channel can be used on our none-transfer models

#x_train = paths_to_tensor(x_train_paths,inputSize)#.astype('float32')/255
x_train3Channel = paths_to_tensor_channel_last_3colour(x_train_paths,inputSize)
#.astype('float32')/255

#y_train = trainDf.iloc[:training_no,targetColumn] # to do all labels: trainDf.i
loc[:training_no,8:]
#x_val = paths_to_tensor(x_val_paths,inputSize)#.astype('float32')/255
x_val3Channel = paths_to_tensor_channel_last_3colour(x_val_paths,inputSize)#.ast
ype('float32')/255

#y_val = trainDf.iloc[training_no:training_no+val_no,targetColumn]

# Deleting dataframes in order to save memory and avoid OOM errors.
del trainDf
del posSample
del negSample
del x_train_paths
del x_val_paths
```

In [ ]:

```
# Trnsfer learning model
# put into separate cell as getting the dense net takes time
# and we often only want to tweak the downstream architecutre
denseNet = DenseNet121(input_shape=(224,224,3), include_top=True)
denseNet.layers.pop() # remov elast layer which has 1000 class softmax in it
```

In [ ]:

```
# Define the rest of the model
model2Layers = Flatten()(denseNet.layers[-2].output)
model2Layers = Dropout(0.3)(model2Layers)
model2Layers = Dense(1,activation='sigmoid')(model2Layers)
model2 = Model(input=denseNet.layers[0].input, output=model2Layers)
for i,layer in enumerate(model2.layers):
    # Don't train the first layers
    if i < 428:
        #layer.trainable=False
        continue
    else:
        continue
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
])
model2.summary()
weightsFilePath2="weights2.best.hdf5"
```

In [ ]:

```
# Create checkpointer that saves the best model weights
checkpoint2 = ModelCheckpoint(weightsFilePath2, monitor='val_acc', verbose=1, sa
ve_best_only=True, mode='max')
image_gen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=.15,
    height_shift_range=.15)

# Set model hyper params
batch_size = 16
epochs = 20

# Train model
history2 = model2.fit_generator(image_gen.flow(x_train3Channel, y_train, batch_s
ize=batch_size),steps_per_epoch=len(x_train3Channel) / batch_size,  epochs = epo
chs, validation_data=(x_val3Channel, y_val), callbacks=[checkpoint2])
# Load best weights
model2.load_weights(weightsFilePath2)
```

In [ ]:

```
# Plot the history of this model
plt.plot(history2.history['acc'])
plt.plot(history2.history['val_acc'])
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Val'])
maxValAcc = max(history2.history['val_acc'])
trainAcc = history2.history['acc'][history2.history['val_acc'].index(maxValAcc)]
plt.savefig(f"Architecture10-{epochs}epochs-{colName}--trainAcc-{trainAcc}--valA
cc-{maxValAcc}.png")
model2.save_weights(f"Architecture10-{epochs}epochs-{colName}--trainAcc-{trainAc
c}--valAcc-{maxValAcc}.hdf5")
plt.show()
```

Results Analysis

In this section we analyse the results for each disease model. The models will load the weights from the best performing (as defined by validation accuracy) epoch. We then produce confusion matrices to calculate recall, precision and f1score.

In [ ]:

```
# Load test data
testDf = pd.read_csv('CheXpert-v1.0-small/valid.csv')

# Remove anomalous dataline
testDf = testDf[testDf.Sex != 'Unknown']
# Drop this column as it has many more classifications than lit suggests and sho
uldn't matter greatly for a CNN

def pathToID(path):
    pathList = path.split('/')
    return pathList[2][7:]

def pathToStudy(path):
    pathList = path.split('/')
    return pathList[3][5:]

# Convert all labels to a series of one-hot encoded labels.
# -1 is uncertain, 0 is negative, 1 is positive, nans are no mention of the dise
ase in the text
testDf = testDf.fillna(0)
# N.B. this is replacing unknowns with true as per u-ones model here: https://ar
xiv.org/pdf/1901.07031.pdf
# This is essentialyl saying that if we're not sure of disease we say they have
 it.
# Just to be on the safeside and have better recall as we care more about recall
than precision
testDf = testDf.replace(-1,1)


# Onehot encode the sex and the xray orientation
testDf = testDf.replace('Male',1)
testDf = testDf.replace('Female',0)
testDf = testDf.replace('Frontal',1)
testDf = testDf.replace('Lateral',0)

testDf =testDf.rename(index=str, columns={"Sex": "Male?",'Frontal/Lateral' :'Fro
ntal1/Lateral0'})


#trainDf.insert(0,'Path', trainDf['Path'])
testDf['Study'] = testDf.Path.apply(pathToStudy)
testDf['Patient ID'] = testDf.Path.apply(pathToID)

# Rearrange Columns
cols = ['Patient ID', 'Study', 'Path', 'Age', 'Male?', 'Frontal1/Lateral0', 'AP/
PA','No Finding',
       'Enlarged Cardiomediastinum', 'Cardiomegaly', 'Lung Opacity',
       'Lung Lesion', 'Edema', 'Consolidation', 'Pneumonia', 'Atelectasis',
       'Pneumothorax', 'Pleural Effusion', 'Pleural Other', 'Fracture',
       'Support Devices']
testDf = testDf[cols]

x_test3Channel = paths_to_tensor_channel_last_3colour(testDf.Path,inputSize)#.as
type('float32')/255
```

In [ ]:

```
def analyseResults(model,x_test3Channel, testDf, disease, radiologistScores=None
):
    # Analyses for a given pathology, getting roc curve and AUROC
    # Identify correct weights file to load
    file = [f for f in os.listdir('.') if os.path.isfile(f) and f"20epochs-{dise
ase}"  in f and ".hdf5" in f]
    if len(file) != 1:
        print(f"Error: can't find single weights file, instead found: {file}")

    # Load in best weights
    model.load_weights(file[0])

    # Generate roc curve values
    model_predict_output = model.predict(x_test3Channel).flatten()
    y_test = testDf[disease]
    fpr, tpr, _ = roc_curve(y_test, model_predict_output)

    # Get AUROC
    auroc = roc_auc_score(y_test,model_predict_output)

    # Plot ROC
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr)

    # If radiologist scores as provided, plot them
    if radiologistScores != None:
        plt.plot(radiologistScores[0][0],radiologistScores[0][1],'ro', label="Ra
d1")
        plt.plot(radiologistScores[1][0],radiologistScores[1][1],'bo', label="Ra
d1")
        plt.plot(radiologistScores[2][0],radiologistScores[2][1],'go', label="Ra
d1")

    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.savefig(f"Architecture10-{disease}--AUROC-{auroc}.png")

    plt.show()
    return auroc
```

In [ ]:

```
diseases = ['Cardiomegaly','Atelectasis','Edema','Pneumonia','Pneumothorax']
# Radiologist scores from ChexPert paper.
rads = [  [[0.05,0.48],[0.23,0.85],[0.11,0.70]], [[0.21,0.8],[0.18,0.71],[0.31,
0.92]], [[0.09,0.63],[0.19,0.79],[0.07,0.58]] ]

# Plot for each pathology
for i,disease in enumerate(diseases):
    if i <= 2:
        auroc = analyseResults(model2,x_test3Channel,testDf,disease, radiologist
Scores=rads[i])
    else:
        auroc = analyseResults(model2,x_test3Channel,testDf,disease)
    print(f"{disease} AUROC: {auroc}")
```