Sadaf Ashrar, David Robison, Nathan Zencey
DATS 6203 Machine Learning II
Dr. Amir Jafari

Group Final Project Report:
Evaluation of Feed Forward Network for Chest X-ray (CXR) Image Classification

Introduction

Since its invention in 1895 by Wilhelm Conrad Roentgen[1], the x-ray has been synonymous with medical examination. Today, many other radiological exams are available, however, the x-ray remains ubiquitous as one of the most commonly utilized diagnostic tools.

The interpretation of radiological images often requires the availability of a physician specialist, the radiologist. Recent reports have identified barriers to accessing radiological expertise such as the volume of radiological examinations requiring interpretation. As such, radiological interpretation provides a paradigm for a computer vision task where a computer-aided tool could be developed to assist radiologists. The large-scale implementation of such a tool could ultimately improve access to timely and accurate radiological interpretation[3].

Background

Until recently, the investigation of deep learning frameworks for radiological assistance has been primarily applied to computed tomography (CT) images. The release of the National Institute of Health (NIH) x-ray database, made vastly more labeled data available, reducing research barriers. As such, there have been several recent works on applications of deep learning to x-ray images, which we will use as reference materials.

Wang et al, after using natural language processing to extract labels for x-ray images in the NIH dataset, benchmarked multiple networks using pre-trained Imagenet architectures. They reported that a ResNet architecture had the best overall performance for classification of 8 different conditions[5].

Several research groups have built on the work of Wang et al. One group, using an abnormality detection framework[6], found that GoogLeNet performed best on a subset of 50,000 images. This group performed significant preprocessing such as histogram equalization to increase image contrast, and reported that network performance was not affected by image resizing.

Most recently, a group at Stanford University developed a 121 layer Dense Convolutional Neural Network, achieving superior results for pneumonia detection and F1 scoring across 14 distinct classes[7]. Their model used an Adam optimizer with parameters pre-initialized from a network trained on ImageNet.

Data

The dataset contains 112,120 x-ray images of 30,805 unique patients, and has been made available by the NIH's Clinical Center. It can be accessed here. The original images are of size 1024 x 1024 x 3. The dataset has been labeled using a novel natural language processing framework to mine 14 disease labels.

The 14 labels are as follows: 1, Atelectasis; 2, Cardiomegaly; 3, Effusion; 4, Infiltration; 5, Mass; 6, Nodule; 7, Pneumonia; 8, Pneumothorax; 9, Consolidation; 10, Edema; 11, Emphysema; 12, Fibrosis; 13, Pleural_Thickening; 14 Hernia

Methods

For our project, we will explore the performance of the LeNet 5 feedforward network architecture for a binary classification task using the NIH Chest X-ray database. Our goal will be to investigate the following:

- How is performance affected by changes in training optimization algorithm?
- How is performance affected by changes in distribution and types of transfer function?
- How is performance affected by changes in the size of convolution layer filters?

For image processing and network implementation, we use a combination of tools available in the following Python packages: Keras, Pytorch, sci-kit learn.

During the preprocessing stage, we reduced the image size from 1024 x 1024 x 3 to 224 x 224 x 3 with random cropping using the cv2 package. To create binary labels, we take any image that does not have a label of ‘No Finding’ and give it a label of ‘Finding’. Finally, we split the data into train, validation, and test sets with the following ratios: 85%- 10%-5%, which equals a distribution of 19,999 - 4,000, 1,000.

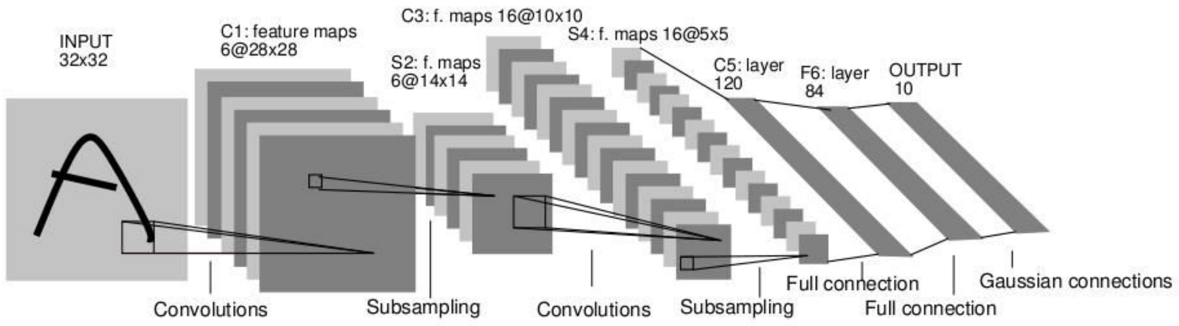
For network training and testing, we use Pytorch and Keras and implement our models in instances hosted by Amazon Web Services (AWS) and Google Cloud.

To assess the performance of our networks, we will use multiple evaluation metrics to understand the behavior of our classifier including:

- Confusion matrices,
- Precision and Recall
- F1 score
- AUC

Background on LeNet 5 Architecture

The LeNet5 feedforward network architecture was originally described in 1998 by Lecun et al. for use on handwritten digit image recognition tasks[9]. The network uses the stochastic gradient descent optimization algorithm to minimize a loss function and thereby achieve gradient based learning. Its architecture, as originally described in 1998, contains two convolution and pooling layers followed by two fully connected layers and an output layer. The architecture is shown in the figure below. In the convolution layers the kernel size as originally described is 5 x 5 with a stride of 1 with max pooling is used to reduce the resulting heights and width via kernel sizes of 2 and strides of 2.



Summary of Results

Effect of Optimization Algorithm:

To understand the effect of optimization algorithm on the performance of the network, we experimented with three variants of the gradient descent algorithm: Stochastic Gradient Descent (SGD) with momentum, Adaptive Moment Estimation (Adam), and Adagrad. Before we reviewing the findings of the experiment, a brief discussion on the different optimization algorithms is provided below.

The gradient descent algorithm is used to a cost function or performance index through sequential updates of network parameters (weights and biases) in the opposite direction of the gradient of the cost function. A hyperparameter called the learning rate dictates the size of the parameter updates for each iteration of a forward and backward propagation through the network.

SGD is one of the most widely used optimization algorithms, and was originally used in the LeNet 5 model. SGD can be modified to mini-batch SGD, which computes parameter updates every 16 training examples. This can cause the error to fluctuate, but a low learning rate and large number of training epochs allows mini-batch SGD to converge to the minimum[10].

As discussed above, we implemented mini-batch SGD with momentum, which is a method to accelerate convergence to the minimum and reduce fluctuations. The second optimization algorithm, Adam has is different from SGD in that it allows for adaptive learning of the network parameters. Finally, Adagrad, optimizes the performance of the network by adapting network parameters such as learning rate by computing bigger updates for parameters that are less frequent and smaller updates for ones that are more frequent[10].

As shown in Table 1 and Figures 1 and 2, the Adam optimizer, despite having the shortest running time, has the weakest test accuracy and precision and a poor F1 score of just 0.374. Of the remaining two, Adagrad has a marginally higher accuracy and recall, while SGD has a quicker running time and a much higher F1 score. As such, we consider SGD to be the best performing of the three optimization algorithms for this task[10].

Table 1: Summary of Network Performance By Optimization Algorithm

Optimization algorithm	**F1 score**	**Accuracy (%)**	**Recall (%)**	**Precision (%)**	**Run time (sec.)**
SGD	0.617	62	94	64	613
Adam	0.374	57	97	56	610
Adagrad	0.5	63	96	60	622

Figure 1: Training Loss and Test Accuracy by Optimization Algorithm

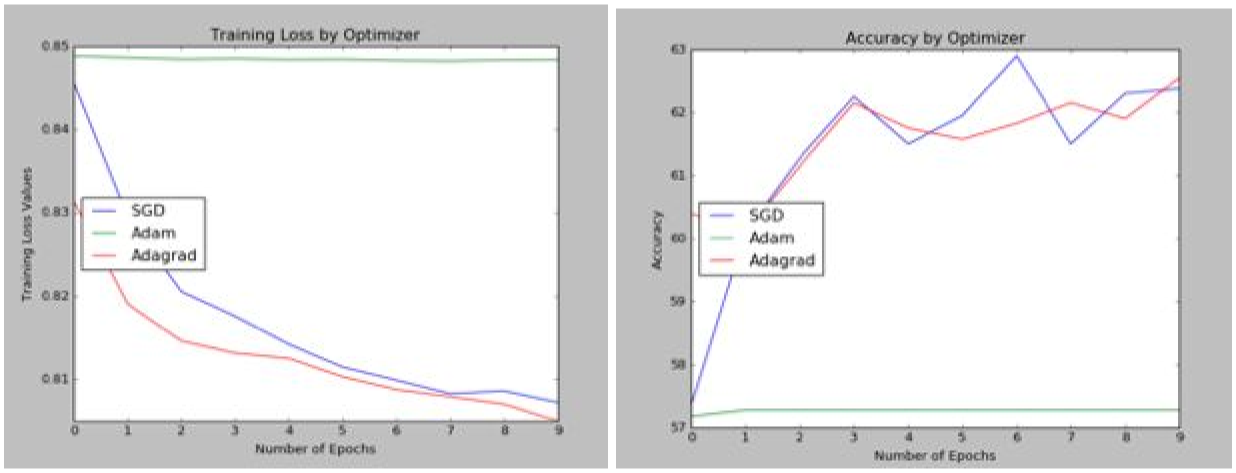
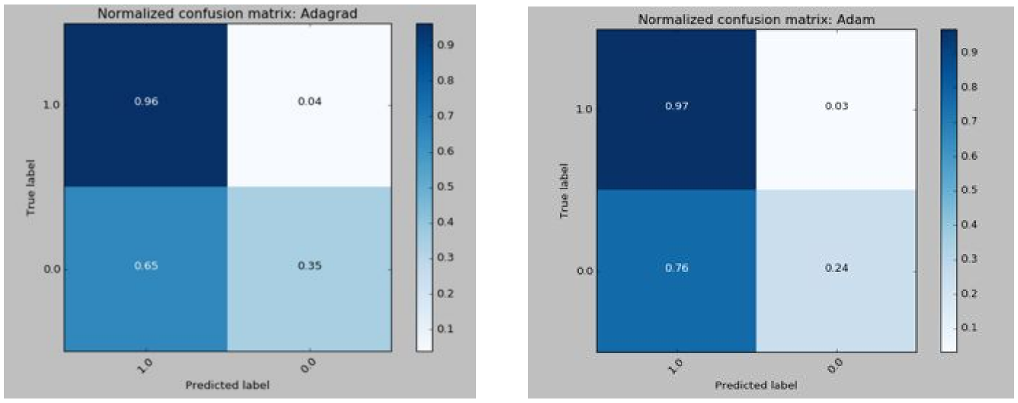
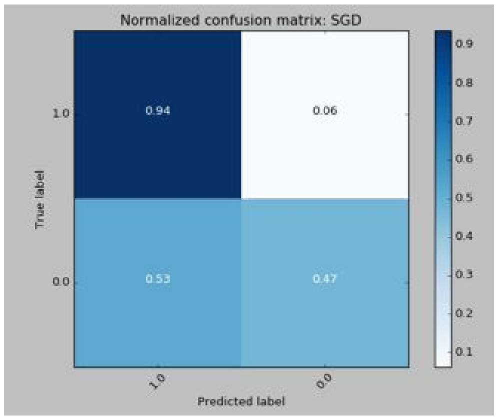


Figure 2: Normalized Confusion Matrices by Optimization Algorithm



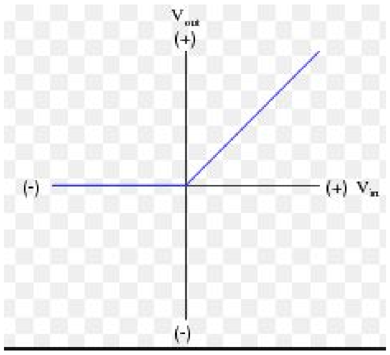


Effect of Transfer Function:

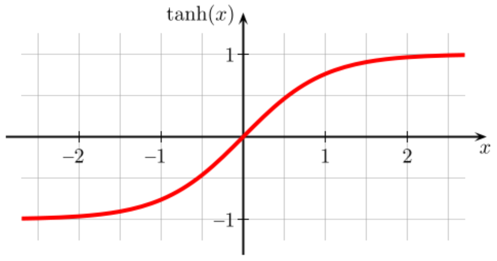
To understand how different transfer functions affect the propagation of error throughout the network and thus network training speed and performance on validation data, we compared three well-known transfer functions. To review, below is a visual of how each transfer function takes inputs and converts them to outputs:

Figure 3: Transfer Functions

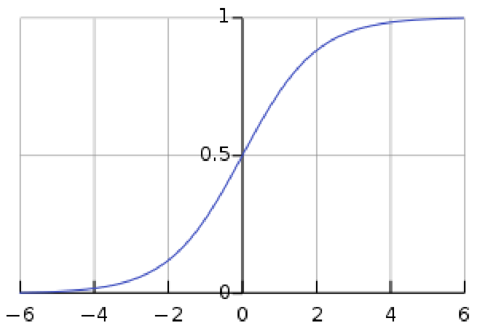
ReLu:



Tanh:



Sigmoid:



Of course, the derivative of each transfer function determines how much error can be back propagated. ReLu is one of the most widely-used (if not the most widely-used) transfer functions in neural networks, as it allows for the maximum amount of error to be propagated throughout the network.

Table 2 shows our findings of differences in performance by transfer function in a network trained over 10 epochs. ReLu is the most accurate but with a middling F1 score. In contrast, tanh takes an enormous hit on accuracy, but shows real performance with precision and recall, as demonstrated by its F1 score. Sigmoid is unquestionably the worst; the sigmoid network only predicts class 0, and therefore has no precision or recall and a F1 score of 0.

Figure 3 providers the networks' performance on training loss versus epoch. Tanh and sigmoid seem susceptible to mini-batch error fluctuation, whereas ReLu shows a steady reduction in training error. The ReLu transfer function is propagating enough error to steadily improve the network, whereas tanh and sigmoid are more unsteady. This suggests that the superior performance of networks trained with Tanh may disappear with repeated trials.

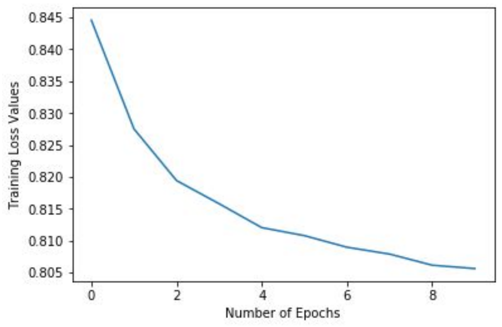
We expect that, at least with ReLu and Tanh functions, these differences would diminish with a greater number of epochs as both transfer functions eventually allowed for convergence. Based on its performance, we do doubt that a sigmoid architecture can pass back enough error to make meaningful updates. A purely sigmoid network was used as a demonstration point and is not recommended for this classification problem.

Table 2: Performance of Network by Transfer Function

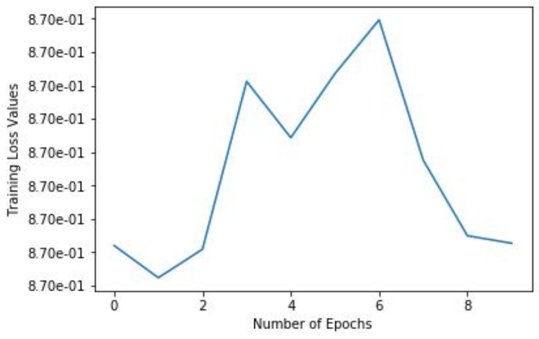
Transfer Function	**F1 score**	**Accuracy (%)**	**Run time (sec.)**
ReLu	0.425	61	545
Tanh	0.588	42	511
Sigmoid	0	57	532

Figure 3: Training Losses by Transfer Functions

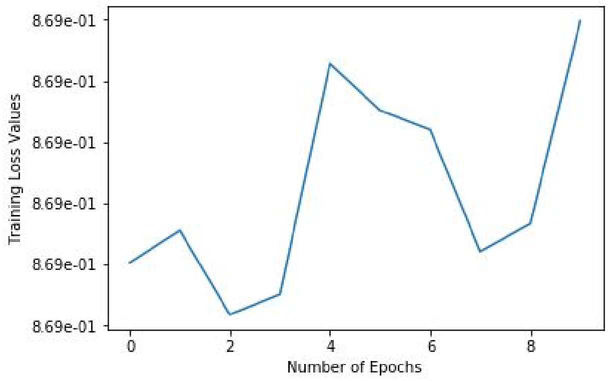
ReLu



Tanh



Sigmoid



Effect of Kernel Size:

The kernel is the key operator of the convolution network and has size of height and width to define its receptive field. It convolves the entirety of the image performing non-matrix multiplication operations to build a feature map of the pixel space. This feature map is then further transformed into lower level features by pooling layers. In practice, the kernel or filter can be thought of as a smaller-sized matrix, in comparison to the input, that contains real-value entries. These values are learned weights that are updated with each learning iteration over the training set. Through this process, the kernel learns the regions that are of significance for extracting features from the data.

Recent literature has led to trends of stacking small kernels[11] (e.g., 3 × 3), however, no standardized practice exists for selecting kernel sizes. As such, it must be experimented with in order to find the best kernel size to learn important feature vectors. To test the effect of kernel size in the convolution layers on the classification, we evaluated both small (3x3, 5x5, and 7x7) and large (10x10, 15x15, and 20x20) kernel sizes on training and test loss as well as validation accuracy over 10 epochs of with mini-batch sizes of 64 and 16 for the training and test sets.

As shown below in Figure 4, the network appears to perform better with larger kernels in comparison to smaller kernels, achieving lower training loss and higher test accuracy. Furthermore as shown below in

Figure 5, among the larger kernels, a kernel of size 15 x 15 or 20 x 20 appears to be more optimal than 10x10. These findings imply that since the size of our input image is 224 x 224 x 3 it appears that larger kernels are necessary to extract meaningful features from this 3 channel pixel space.

Figure 4: Training Loss and Test Accuracy by Kernel Size

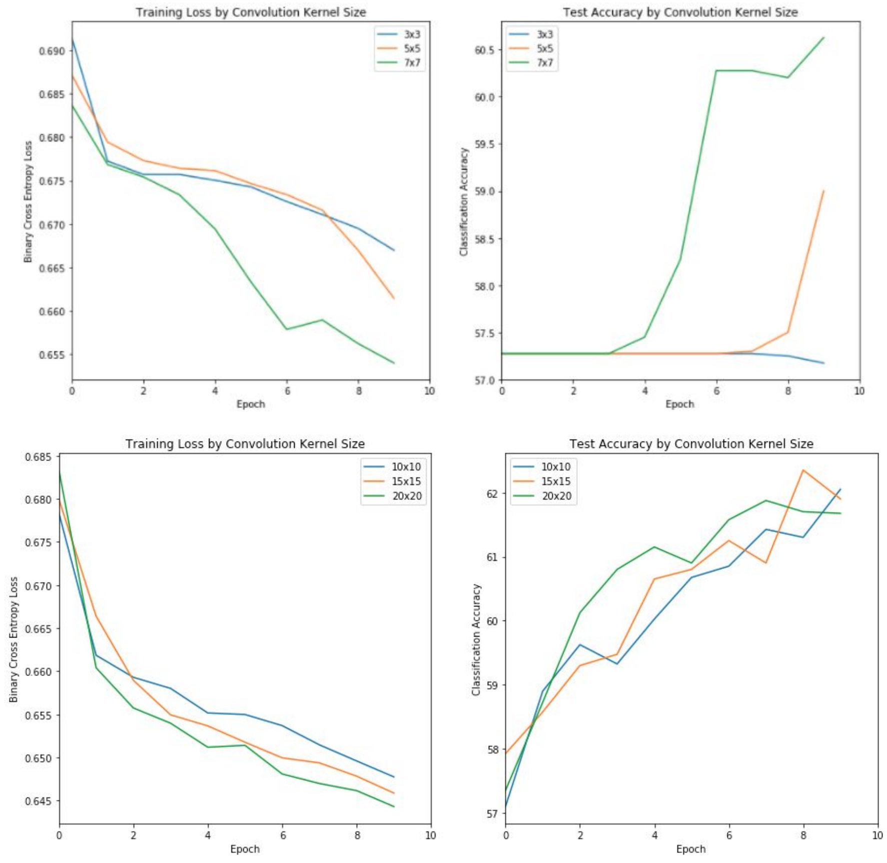
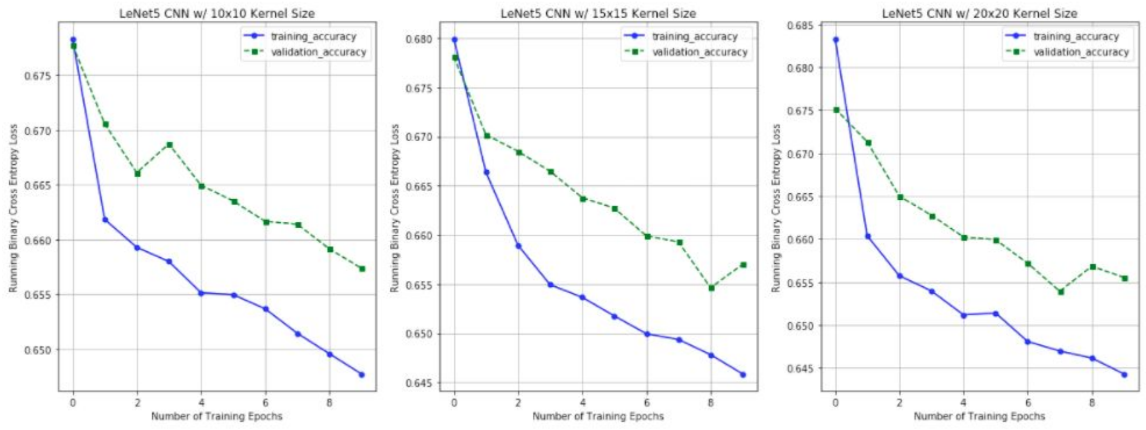
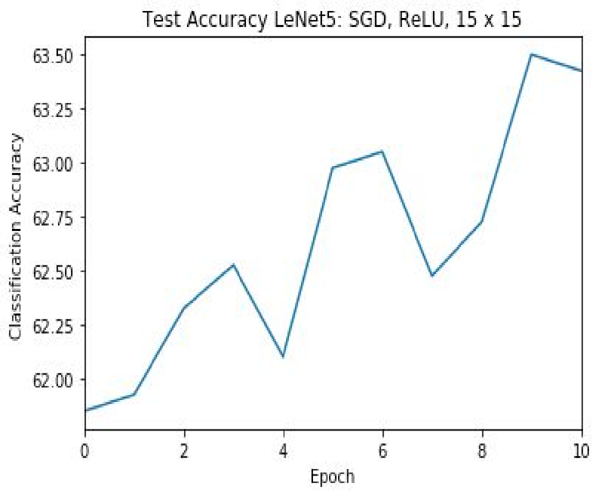


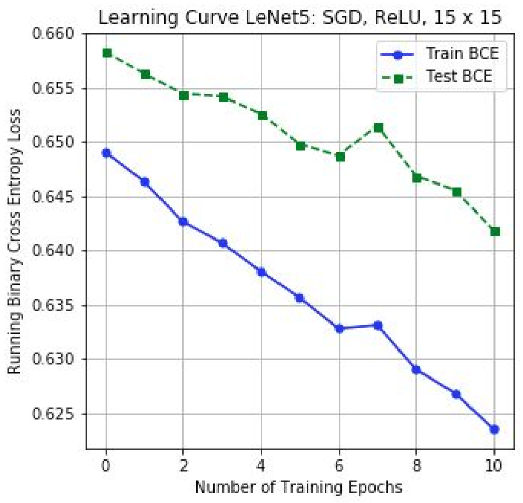
Figure 5: Network Training Loss and Validation Loss with Large Kernels



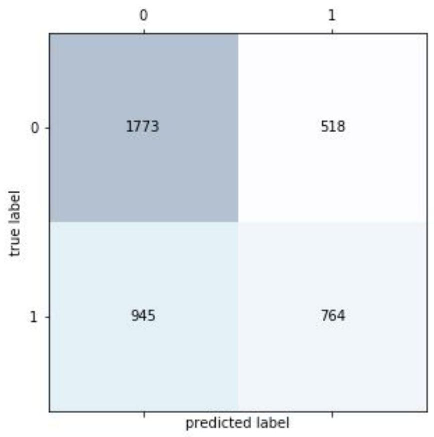
Best Model Results

After performing these experiments, we trained and tested a final model for 30 epochs with a kernel sizes of 15 x 15 using stochastic gradient descent with ReLU transfer functions. The performance of the model is shown in figures and tables below with loss and accuracy assessed every 3 epochs. This final model had an accuracy score of 63%, however, we also evaluate it using Precision, Recall, F1-score, and confusion matrices. As assessed by recall, this model is still performing poorly with a recall of 0.45 signifying that it is predicting 'No Findings' for many images with a true label of 'Findings'. For the 'Findings' class, the model's precision is slightly better at 0.60, signifying that when it predicts 'Finding' it is correct 60% of the time.





Class	Precision	Recall	f1-score
No Finding	0.65	0.77	0.71
Finding	0.60	0.45	0.51



Conclusions:

Our best model was only able to achieve 63% percent accuracy and a recall of 0.60 for the class of interest. In comparison to radiologist performance for simply identifying whether there is a finding or no finding of interest, this performance is poor. However, our analysis is limited as we did not make use of regularization techniques such as dropout or investigate alternative learning rates. In addition, we performed little data augmentation. However, we were able to determine that for a LeNet 5 architecture larger kernel sizes and training using stochastic gradient descent with ReLU transfer functions is best for performance.

References

1. British Library. Roentgen's discovery of the x-ray. Available at: https://www.bl.uk/learning/cult/bodies/xray/roentgen.html.
2. Mayo Clinic. Chest X-rays. Available at: https://www.mayoclinic.org/tests-procedures/chest-x-rays/about/pac-20393494.
3. Diagnostic Imaging. In Radiology, Turnaround Time is King. Available at: http://www.diagnosticimaging.com/practice-management/radiology-turnaround-time-king.
4. National Institutes of Health. NIH Clinical Center provides one of the largest publicly available chest x-ray datasets to scientific community. Available at: https://www.nih.gov/news-events/news-releases/nih-clinical-center-provides-one-largest-publicly-available-chest-x-ray-datasets-scientific-community
5. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. IEEE CVPR 2017, http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf
6. Tariqul, Mohammad, et al. "Abnormality Detection and Localization in Chest X-Rays Using Deep Convolutional Neural Networks." 27 Sept. 2017, arxiv.org/abs/1705.09850.
7. Rajpurkar, et al. "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning." 25 Dec. 2017, arxiv.org/abs/1711.05225.
8. Simonyan, et al. "Very Deep Convolutional Networks for Large-Scale Image Recognition." 10 Apr. 2015, arxiv.org/abs/1409.1556.
9. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.
10. Ruderm, Sebastian (2016). "An overview of gradient descent optimisation algorithms." 15 Sept. 2016, arXiv preprint arXiv:1609.04747.
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of ICLR. (2015) 1, 4, 9, 13