Deep Learning Theory And Applications

Members: Joe Bongo, Abhinav Bhardwaj, Daniel Edelberg

Final Report: Using a Convolutional Neural Network to identify the presence of Tuberculosis using Chest X-Rays.

**Introduction/Motivation**

Tuberculosis (TB) is one of the top 10 causes of death and the leading cause of death from a single infectious agent in the world (World Health Organization) [1]. According to the WHO, the objective of TB screening is to ensure that active TB is detected early to then reduce the risk of poor disease outcomes, along with the social and economic consequences of the disease, and to reduce transmission [1]. Furthermore, TB is endemic to many regions that have relatively poor healthcare infrastructure, including sub-Saharan Africa and South East Asia [1]. Therefore, a deep neural network that could be trained to detect the presence of TB could be an inexpensive and effective method to implement wide-spread TB screening while not being a burden on the healthcare systems. The effects of tuberculosis can be deadly, and early detection, facilitated by tools such as deep networks and the ability to alleviate physician workloads provide an opportunity to build on the wave of neural networks and their current and future applications to healthcare.

Convolutional Neural Networks (CNNs) have been shown to be effective in simple image classification tasks (and some more complicated ones, provided with enough data) [2]. One benefit of the CNN is that by analyzing the output hidden layers, one can identify which features of the images were identified in the network to be important in classification. This can also be used as a tool for physicians to aid in the diagnosis of TB, so as to not completely avoid a physician’s input (which could lead to ethical issues in a clinical setting) and rather as a tool to speed up diagnoses and improve the efficiency of evaluations and diagnoses.

For this project, we are using both the Montgomery County Hospital Chest X-Ray dataset and the Shenzhen Hospital Chest X-Ray dataset available from the NIH National Library of Medicine [3]. These images come from the tuberculosis control program of Montgomery county and Shenzhen hospital. The sets include a total of 406 normal X-Rays and 394 abnormal X-rays that show various manifestations of tuberculosis (800 images total). These images are of variable sizes, and approximately all have resolution 4000x4000 pixels and are in grayscale. The images also have labels with the sex of the patient, their age, and the truth value of their diagnosis.

**Background/Related Work**

There has been a lot of work done on convolutional networks for the purpose of image classification, including for healthcare purposes (like medical imaging) [4]. Deep learning has already shown state-of-the-art performance for tasks such as breast cancer classification, tumor segmentation, and X-ray analysis [4]. Specifically for Tuberculosis, Chest X-ray imaging has been a topic of interest, and a number of models have been used to accurately predict the presence of TB. These include hand-crafted algorithms, support vector machines, as well as deep network architectures [6]. However, many of the previous strategies have made use of additional clinical parameters, such as test results and demographic information [7], or even additional algorithms for lung segmentation that require additional training. Other attempts at using convolutional networks to learn TB Chest X-rays use well-known models such as GoogLeNet or ResNet, which are very powerful classifiers for image classification. However, both networks have millions of parameters and require a large number of images and are therefore much more likely to overfit the data when applied to datasets with a limited number of images, such as chest X-rays [8]. We therefore wanted to build a network that had a low number of parameters (< 1 million), as well as accurate, generalizable, and had high AUC scores, due to the nature of the disease. We also wanted a network that did not require additional clinical information to train, given the nature of the regions where TB is highly endemic, and resources are often scarce, especially for healthcare.

**Model**

Since our problem is an image classification task, we knew from the start that our model of choice was going to be a convolutional neural network. That said, our data and computation restrictions were roadblocks that had to be considered. Since we only had 800 images and very little memory, consisting of only our personal computers and Google Colab, training models with a very large number of epochs or large batches was impossible. Therefore, our intention was to favor models with smaller batch sizes, models that generalized well, and ones that had fewer parameters. In the end after a rigorous trial and error process that included training 16 different models, the highest performing model that we came up with was inspired by the architecture of AlexNet, a widely successful deep Convolutional Neural Network trained for image classification tasks by members of the University of Toronto [5].

The piece of our architecture that was inspired by AlexNet was the use of many convolutional and pooling layers that were not forbidden from following the same type of layer (i.e. a max pooling layer can follow a max pooling layer), conflicting from the way basic CNN's are most commonly taught and used in practice. It is also noteworthy that the architecture used by the winning model included no regularization. This was surprising to us at the time, but makes sense when we consider the limited amount of data we had, and the fact that max pooling layers are often thought of as a type of regularization because they extract features that occur anywhere in a given kernel of observations (pixels), as opposed to always looking at a particular observation. In fact, we tried to fit this same model with a small L1 regularization on the last convolutional layer, but the result ended up being less generalizable, as it had a lower test accuracy and AUC. The results of each model we tested are shown in a table in the results section, along with a description of the architecture of each model. Our best model followed these insights we gained from AlexNet and produced the most generalizable result, which included many convolutional and pooling layers, but each with fewer filters than the previous models.

Training a Convolutional Neural Network model that fit our training data well turned out to not be the main issue that we encountered in the modelling, as nearly every model was able to easily classify the random 80 percent of images that it was trained on correctly. The main issue was with how our model was able to generalize to the test data that we presented it. This certainly made the job of looking for the best model quite difficult. Through a rigorous trial and error process, we were able to reach several conclusions about how the models generalized. The most surprising conclusion was that the models actually performed best with a larger batch size of 32. This is peculiar as small batch gradient descent is known to generalize better, as its more random nature allows it to find flatter minima more easily. That said, it could be that our lack of data has forced our loss lanscape to be much more jagged than typical for this type of problem, and this leads to better performance from larger batches. Another issue that we encountered was our model not hitting an interpolation point that we often see with neural networks (albeit we were limited on the number of epochs we could run as a large number of epochs would result in a model taking many hours to train), and the size of the network with respect to how many parameters we could train at once. That said, we did not have a large increase in interpolation when models were trained for 200 epoch or longer, like deep learning theory suggested was a possibility. Therefore, we proceeded with our AlexNet inspired model under these restrictions, knowing that it could be very difficult to get a model that generalizes well no matter how comprehensive the architecture.

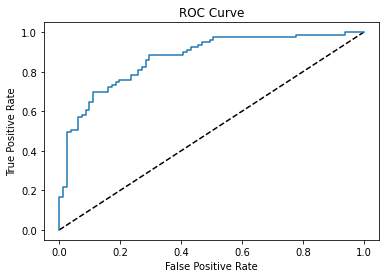
Our model begins with the 512x512 image being fed to a convolutional layer that applies 200 3x3 filters with a stride of 2 and then the data is batch normalized and then fed to a convolutional layer that applies 200 3x3 filters with a stride of 1. For every convolutional layer in this network (including the previous two), we have a ReLu activation function. This is then followed by a 3x3 max pooling layer with a stride of 2 and then fed to 3 consecutive convolutional layers of 100 3x3 filters with a stride of 1, each of which is followed by batch normalization. Then we have a 3x3 max pooling layer with a stride of 2, another convolutional layer of 100 3x3 filters with a stride of 1 followed by batch normalization, another 3x3 max pooling layer with a stride of 2, and then the model is flattened using global max pooling. This now 100 dimensional vector is then succeeded by 3 fully connected layers of size 512 (with a ReLu activation function), size 128 (with a ReLu activation function), and then size one with a sigmoid activation function. The model is then trained for 120 epochs using 32-observation batches and Adam for optimization, with the default parameters. This architecture yielded our results below of a test accuracy of 82.5%, an AUC of .8831, and an F1 score of 0.8269 and like most of our models, fit the training data perfectly.

**Results**

Below, we present preliminary tables of the results from the 16 different models that we trained. We tried many different types of differentiators between networks including different numbers of epochs, different batch sizes, different numbers of layers (convolutional, pooling and fully connected), different ways that our data was flattened, and different techniques for regularization. Through a rigorous trial and error process, we were able to build model 10, which gave us a validation accuracy of 83.13% on the 160 test observations that it was given. We additionally observed the positive effects of the addition of batch normalization, additional fully connected layers, and coming up with a model that was deep, but not over parameterized.

In the supplementary section after the report are descriptions of each of the architectures used for the corresponding models. Some of the models are identical to others with only one different parameter, architectural component, while others differ far from any of the other models that were trained. Overall, we found that for the networks we were studying, regularization did not improve the overall generalizability possibly due to the lack of data or our choice of penalty being too small. We also found that it seemed that those architectures that contained more convolutional layers tended to perform a bit better, as the image data was abstracted further and further from its original form.

Finally, one of the key points of a model being trained for clinical purposes is the specificity and sensitivity of the model. While our dataset was fairly balanced, in the real world there are many more cases of negative X-rays than positive ones, and so we want to ensure that our model is not too specific or too sensitive. Therefore, two important metrics we wanted to consider when training a model was the test AUC and test F1 scores, which we have already reported. The graph of the ROC curve (from which AUC comes from) is below, which is the false positive rate vs. true positive rate of the model at different thresholds. An AUC near 1 indicates near perfect specificity and sensitivity, while an AUC of 0.5 indicates random assignment. An F1 score is another measure that is a calculation of recall and precision, and is from a scale of 0 to 1. Both our scores were not perfect but were adequate for our experimental parameters. Furthermore, one of the benefits of a convolutional network is the ability to visualize what information the network determined was instructive for making classifications, i.e. where the network was looking. This can be a useful tool for understanding what segments of the image the network deems useful, and could be an additional tool for training healthcare professionals in where to focus for non-computerized classification of TB. We show the image from the first convolutional layer below that shows high activation around the middle of the lungs and near the bottom for a sample image.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model # | Regularization/  Normalization | Epochs | Batch Size | Flattening | Test Accuracy | Number of Parameters | Test AUC |
| 1 | None | 200 | 8 | Global Average | .7250 | 23,603,201 | .7401 |
| 2 | None | 200 | 8 | Global Max | .8 | 23,814,345 | .8467 |
| 3 | 1 Dropout Layer | 150 | 8 | Global Max | .7500 | 25,758,545 | .7687 |
| 4 | None | 200 | 8 | Global Max | .7688 | 6,080,237 | .8043 |
| 5 | None | 200 | 8 | Global Max | .8 | 25,980,653 | .8128 |
| 6 | None | 120 | 2 | Global Max | .7 | 25,980,754 | .6996 |
| 7 | None | 200 | 32 | Global Max | .8125 | 23,814,345 | .8660 |
| 8 | None | 200 | 8 | Global Max | .7688 | 17,018,569 | .8019 |
| 9 | Batch Normalization | 100 | 32 | Global Max | .8 | 815,901 | .8496 |
| 10 | Batch Normalization | 120 | 32 | Global Max | .825 | 932,505 | .8831 |
| 11 | Batch Normalization | 300 | 4 | Global Max | .7125 | 932,505 | .8116 |
| 12 | 4 Dropout Layers | 100 | 8 | Global Average | .6875 | 23,603,201 | .7349 |
| 13 | L2 Regularization | 100 | 5 | Global Average | .4625 | 23,603,201 | .5000 |
| 14 | L2  Regularization | 100 | 5 | Global Average | .7063 | 23,603,201 | .7377 |
| 15 | L1  Regularization | 100 | 5 | Global Average | .7937 | 23,603,201 | .8584 |
| 16 | L1  Regularization and Batch Normalization | 120 | 32 | Global Max | .8188 | 932,505 | .8507 |

**Contributions**

The project was split up evenly amongst the three students. Daniel worked primarily on putting together the project proposal, doing research on the previous literature, and building the initial network models so that they could be easily modified for hyperparameter modification when assessing the test accuracy/metrics. Daniel also worked on writing the introduction and background information, and this section. Joe and Abhinav worked mainly on assessing many different deviations of the model with a lot of hyperparameter tuning to try and choose the model that we believed would best fit the goals of the project. This included testing regularizations, dropouts, size of layers, number of layers, activations, and other hyperparameters that could be tuned. We have listed a number of the networks we tried, but the list is not exhaustive. They also both worked on the model section and discussion section. Overall, we believe that all 3 team members contributed approximately the same amount of time and effort to the project.

**Conclusions/Discussions**

We believe that we could have achieved even better results had we had access to more computational power, but this would have likely required more data to train. Having a deeper architecture certainly would have improved our TB prediction performance. We also could have trained on more data, as opposed to data from just two hospitals (which we probably would have needed to train a deeper architecture). The work we did can be built upon for any kind of medical imaging problem. The exact same idea could be used for COVID-19 detection, especially when testing is not reliable and hard to come by, and it shares many similarities with TB. These are only a few of the many potential applications. There are also other architectures we could have tried, such as graph neural networks.

**References**

[1] Anderson, L. et al. WHO Global tuberculosis report 2015. WHO Libr. Cat. Data 1, 1689–1699 (2015).

[2] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9

[3] Jaeger, S., Candemir, S., Antani, S., Lu, P. & Thoma, G. Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. Quant Imaging Med Surg 4, 475–477 (2014).

[4] Litjens, G. et al. A Survey on Deep Learning in Medical Image Analysis. (2017).

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[6] Karargyris, A. et al. Combination of texture and shape features to detect pulmonary abnormalities in digital chest X-rays. Int. J. Comput. Assist. Radiol. Surg. 11, 99–106 (2016).

[7] Melendez, J. et al. An automated tuberculosis screening strategy combining X-ray-based computer-aided detection and clinical information. Sci. Rep. 6, 1–8 (2016).

[8] Islam, M. T., Aowal, M. A., Minhaz, A. T. & Ashraf, K. Abnormality Detection and Localization in Chest X-Rays using Deep Convolutional Neural Networks. CoRR abs/1705.0 (2017).

**Supplementary: Architectures of Each Model**

1. Convolutional Layer (512 filters, 3x3 kernel, stride of 2, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Average Pooling, Fully Connected Layer (1 node, sigmoid activation)
2. Convolutional Layer (512 filters, 5x5 kernel, stride of 2, relu activation function), 5x5 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
3. Convolutional Layer (512 filters, 5x5 kernel, stride of 2, relu activation function), 5x5 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Dropout with probability of .5, Fully Connected Layer (1000 nodes, relu activation), Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
4. Convolutional Layer (512 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Dropout with probability of .5, Fully Connected Layer (1000 nodes, relu activation), Fully Connected Layer (300 nodes, relu activation), Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
5. Convolutional Layer (512 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Dropout with probability of .5, Fully Connected Layer (1000 nodes, relu activation), Fully Connected Layer (300 nodes, relu activation), Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
6. Convolutional Layer (512 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Dropout with probability of .5, Fully Connected Layer (1000 nodes, relu activation), Fully Connected Layer (300 nodes, relu activation), Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (2 node, sigmoid activation). This model used categorical cross entropy loss instead of binary cross entropy loss.
7. Convolutional Layer (512 filters, 5x5 kernel, stride of 2, relu activation function), 5x5 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Max Pooling, Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
8. Convolutional Layer (512 filters, 8x8 kernel, stride of 4, relu activation function), 8x8 Max Pooling, Convolutional Layer (1024 filters, 4x4 kernel, stride of 2, relu activation function), 5x5 Max Pooling, Convolutional Layer (2048 filters, 2x2 kernel, stride of 1, relu activation function), Global Max Pooling, Fully Connected Layer (100 nodes, relu activation), Fully Connected Layer (1 node, sigmoid activation)
9. Convolutional Layer (200 filters, 3x3 kernel, stride of 2, relu activation function), Batch Normalization, Convolutional Layer (200 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Global Max Pooling, Fully Connected Layer (1 node, sigmoid activation)
10. Convolutional Layer (200 filters, 3x3 kernel, stride of 2, relu activation function), Batch Normalization, Convolutional Layer (200 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Global Max Pooling, Fully Connected Layer (512 nodes, relu activation), Fully Connected Layer (128 nodes, relu activation) Fully Connected Layer (1 node, sigmoid activation)
11. Convolutional Layer (200 filters, 3x3 kernel, stride of 2, relu activation function), Batch Normalization, Convolutional Layer (200 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Global Max Pooling, Fully Connected Layer (512 nodes, relu activation), Fully Connected Layer (128 nodes, relu activation) Fully Connected Layer (1 node, sigmoid activation)
12. Convolutional Layer (512 filters, 3x3 kernel, stride of 2, relu activation function), Dropout of .5, 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), Dropout of .5, 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Dropout of .5, Global Average Pooling, Dropout of .5, Fully Connected Layer (1 node, sigmoid activation)
13. Convolutional Layer (512 filters, 3x3 kernel, stride of 2, relu activation function, l2 penalty of .01), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function, l2 penalty of .01), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function), Global Average Pooling, Fully Connected Layer (1 node, sigmoid activation)
14. Convolutional Layer (512 filters, 3x3 kernel, stride of 2, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function, l2 penalty of .01), Global Average Pooling, Fully Connected Layer (1 node, sigmoid activation)
15. Convolutional Layer (512 filters, 3x3 kernel, stride of 2, relu activation function), 3x3 Max Pooling, Convolutional Layer (1024 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling, Convolutional Layer (2048 filters, 3x3 kernel, stride of 1, relu activation function, l1 penalty of .00001), Global Average Pooling, Fully Connected Layer (1 node, sigmoid activation)
16. Convolutional Layer (200 filters, 3x3 kernel, stride of 2, relu activation function), Batch Normalization, Convolutional Layer (200 filters, 3x3 kernel, stride of 1, relu activation function), 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function), Batch Normalization, 3x3 Max Pooling(stride of 2), Convolutional Layer (100 filters, 3x3 kernel, stride of 1, relu activation function, , l1 penalty of .00001), Batch Normalization, 3x3 Max Pooling(stride of 2), Global Max Pooling, Fully Connected Layer (512 nodes, relu activation), Fully Connected Layer (128 nodes, relu activation) Fully Connected Layer (1 node, sigmoid activation)