Team Banjo - Project Report : Pneumonia X-Ray Classification and Analysis

Andrew Novokshanov
Georgia Tech
anovokshanov3@gatech.edu

Shaun Jacob
Georgia Tech
sjacob31@gatech.edu

Abstract

Deep learning, more specifically Convolution Neural Networks, allows for the creation of models that can efficiently classify images. A method of efficiently identifying diseases via medical images would allow practitioners to not only diagnose patient's faster, but also notice patterns between patients with greater ease. The goal of our work was to develop a model that can correctly classify x-rays of pneumonia patients and distinguish normal x-ray images from pneumonia x-ray images. Using our trained-model, we utilized class activation maps to analyze what our model looked for when making classifications and compared our results with what a radiologist would look for in a patient's x-rays. In addition to achieving a model with greater than 90% precision, the class activation maps demonstrated similar focus regions as those of radiologists.

1. Introduction/Background/Motivation

The goal for our project was to develop a deep learning model that can accurately predict whether a patient does or does not have pneumonia using chest x-ray images. There has recently been a heavy increase in demand for deep learning models that can assist with medical image analysis [3, 4]. Compared to normal medical imaging, which requires trained doctors' rigorous analysis, deep learning models can allow medical practitioners to rapidly identify illnesses[9]. Furthermore, through the use of class activation maps (CAM), scientists can find patterns within patients not previously known about, leading to a better understanding of illnesses and symptoms.

The current state of the art techniques for classifying medical images are convolutional neural networks [7]. While there has been recent interest in the use of transformer networks for classification problems [6], we have opted to pursue the tried and true method of convolutional neural networks. For obtaining CAMs from a model, Guided GradCam is a popular technique used, and allows for visual explanations of deep learning models' results.

2. Data

The dataset we chose to use was found on Kaggle and contains grey-scale images of patients' chest X-rays. Each image also has an associated label signifying whether the patient did, or did not have pneumonia at the time of the x-ray. The photos were pre-screened to remove unreadable or low quality images, and then graded by three professionals to ensure label accuracy. In total there were 5,863 images, of which 4,265 were of lungs affected by pneumonia, and 1,598 were of healthy lungs. There was a notable difference in the number of images by label, specifically that there were nearly three times as many pneumonia images as healthy images. Later in the paper, we will describe how we attempted to counter this imbalance and how the imbalanced dataset impacted our models.

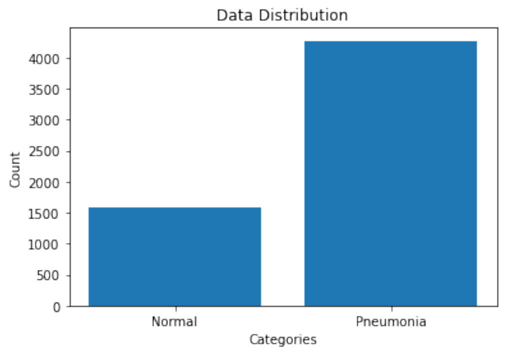

Figure 1. Bar graph demonstrating distribution of images by label

3. Approach

3.1. Model Design

Since our dataset is comprised of images, we elected to use a Convolutional Neural Network to build our image classifier. Convolutional Neural Networks are very heavily used in computer vision due to their ability to extract features out of an image. Since part of our objective was to be able to determine which areas of the chest x-ray image contained signs of pneumonia, we felt that using a Convolutional Neural Network would best help us accomplish our objective. Since Convolutional Neural Networks have generally been shown to give good results on image classification tasks, we felt that using this architecture would give us the results we wanted.

Given that our dataset was heavily imbalanced, we anticipated that our classifier would be biased towards the pneumonia images and would potentially classify images as pneumonia. To combat the class imbalance, we used class weights, more on which will be discussed in the next section.

Our Neural Network architecture was comprised of three Convolutional blocks and two linear layers. Each of the Convolutional blocks was comprised of a Convolutional Layer followed by LeakyReLU Layer, MaxPooling Layer, and Dropout Layer.

In our initial search for our classifier's architecture, we examined well known architectures. We noticed that quite a few architectures used Convolutional blocks, each comprising of a Convolutional layer followed by a RELU activation and a max pooling layer. We decided to experiment with these Convolutional blocks and found good results. We made a couple of deviations from these architectures and were able to achieve even better experimental results. The two deviations we made from these initial architectures was using LeakyReLU instead of ReLU and adding a Dropout Layer at the end of each Convolutional block.

We found that Leaky ReLU gave us better performance and helped avoid the problem of zero-gradients resulting from the negative domain of the ReLU function. We used Dropout Layers at the end of each Convolutional block to prevent our model from overfitting to the training images and better generalize to the test set.

We passed the output of our last linear layer through the Sigmoid function to output the probability that the given image is from a patient that has pneumonia.

Since this is a binary classification problem and our network outputs the probability that the given image is from a patient that has pneumonia, we used binary cross-entropy as our loss function. We also used Adam as our optimizer for our network as it gave us the best experimental results out the other optimizers we tried: Stochastic Gradient Descent (SGD) and SGD with momentum.

Unlike the architecture, which is fairly similar to other architectures commonly used for classification, the hyper-parameter tuning was a large part of the work done in tuning our model. Since the hyper-parameters were the bulk of the model design, we discuss how we reached our conclusive values in the experiment and results portion of the report.

3.2. Unbalanced Dataset

Our dataset had roughly three times as many pneumonia images as normal images. The performance of a neural network will suffer if the dataset is imbalanced, so we had to implement extra measures to ensure the severe class imbalance did not affect the training portion of our model. The technique we chose to combat imbalance is class weights [1, 8].

Class weights allow our classifier to place a higher value on minority classes so that our classifier will still be able to learn a proper classification despite the class imbalance in out dataset. Since there are around three times as many pneumonia images as normal lung images, we made the normal class carry a much higher weight than the pneumonia class so that misclassification of the normal class would add a higher penalty to the loss function. This was done in order to influence the neural network to pay more attention to the normal cases.

We calculated class weights with respect to the proportion to the number of samples for each class in the train data split. We used Scikit-Learn's compute class weight function, which calculated the class weights to counter the dataset imbalance based on the class distribution of the training split. This was done only on the training split to avoid giving the model any information about the validation or test splits as these are used for evaluation purposes.

3.3. Class Activation Maps

The means by which we approached generating class activation maps is Captum's GuidedGradCam class. Although we looked at the results of normal Grad-Cam and Guided Backpropogation, we opted to show our findings using Guided Grad-Cam specifically as it is a combination of the aforementioned methods. Furthermore, Guided Grad-Cam has been a popular alternative to normal CAMs and saliency maps for visualizing what parts of an image matter most to a CNN model. Developed by Meta explicitly to work alongside Pytorch, Captum's classes use state-of-the-art algorithms [5] to calculate attributions. Captum also provides helpful visualization tools with which to analyze the CAMs. Although neither Captum nor CAMs is unique to imagery analysis, there was surprisingly little discussion about using CAMs on our dataset.

Since the primary difficulty in using Captum is properly configuring the model and image before passing it into Captum, there was not much variation regarding what could be done from our end. Similarly, there was not much that could lead to technical issues. While Captum does not require any additional tuning or training beyond the model the attributions will be calculated off of, there was a significant challenge that we anticipated.

While we were confident that we could obtain visualizations of the resulting CAMs of images with our model, we were uncertain of whether or not the resulting maps would yield enough information to come to a conclusion of whether or not the model is looking for the same details as radiologists. While there are several factors that radiologists look at when making a diagnosis, the most important detail radiologists look for is the obfuscation of the lungs by whiteness [2]. However, with our untrained eyes we were unable to tell the difference between healthy and unhealthy lungs in our dataset. If we could not make the distinction ourselves, how can we be sure that our model was looking at the same details? If we could, however, find a pattern between the CAMs, then we have demonstrated the potential of using CAMs to identify similarities between illnesses.

We were also initially surprised that the resulting visualizations were not a traditional heat-map in full color, but rather in shades of blue. By default, Captum displays the absolute value of attributions, meaning we could not differentiate if a pixel helped or hurt when making a classification. By changing the signs of the attributions when visualizing, however, we realized we could look at not only the absolute value of attributions (the pixels that impacted the model), but also just positive attributions (pixels that supported the end classification), negative attributions (pixels that supported the opposite classification), and all pixels (combination of positive and negative).

4. Experiments and Results

4.1. Experiment Overview and Hyper-Parameters

To evaluate the performance of our model, we used the following four metrics: balanced accuracy, precision, recall, and specificity. Precision is the number of true positives generated over all positive predictions made. Recall is the number of true positive predictions that were generated over all true possible predictions across all classes. Specificity is the number of true negatives over all true negatives and false positives. Balanced accuracy is the average of specificity and recall. Since our data is very imbalanced, we felt that these metrics would be a better indicator of model performance.

We split our data into training, validation, and test splits each one comprising 80%, 10%, 10% of the dataset respectively. We trained our model with the following hyper-parameters:

EPOCHS = 15
BATCH SIZE = 32
LEARNING RATE = 0.00175
REGULARIZATION = 0.0001
IMAGE SIZE = 128 x 128

To improve the generalization of our classifier, we used a batch size of 32. We found that using a smaller learning rate helped our classifier generalize to our holdout test set. We did not want to penalize large weights as much because this is a complex image classification task, and we found that reducing regularization improved our results. We found that 15 epochs gave us optimal results, so we trained our model for 15 epochs. Although we originally tested our model using an image resolution of 96x96, we found that an image size of 128x128 gave us the optimal results. While we expect that a larger resolution would yield more accurate results, unfortunately PyTorch was experiencing size limitations when going beyond 128x128. With these hyper-parameters, we generated two models: one trained using class weights and one trained without using class weights.

4.2. Training Without Class Weights

The testing results displayed below are for the model trained without using class weights:

Balanced Accuracy: 94.9%
Precision: 97.7%
Recall: 96.3%
Specificity: 90.0%

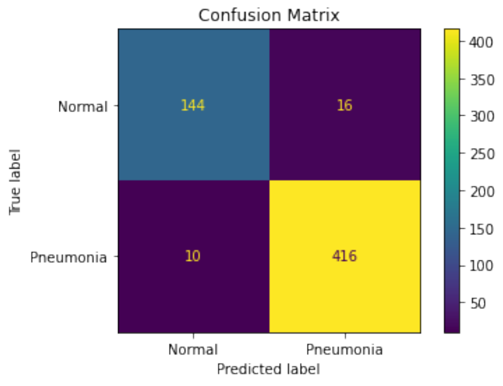

Figure 2. Confusion Matrix for Model without Class Weights

Our model did better than expected considering the presence of the major class imbalance. As indicated by a high precision score, our model did a great job of detecting pneumonia cases. Specificity was the lowest of the scores, indicating that the model struggled to some extent with distinguishing pneumonia x-ray images from normal x-ray images. This could be due to the class imbalance. Since there were many more pneumonia x-ray images than normal x-ray images, the classifier may have learned that a given image is very likely to be a pneumonia image, which may have resulted in the incorrect classifications.

In practice, we cannot allow our classifier to return false positives. In the medical industry, acting on a false positive can result in unnecessary and ineffective treatments, which doesn't provide any benefit to the patient. With that in mind, we decided to try and improve the model's ability to distinguish between normal x-ray images and pneumonia x-ray images. We attempted to accomplish this by training a second model using class weights. The results of this training are explored in the next section.

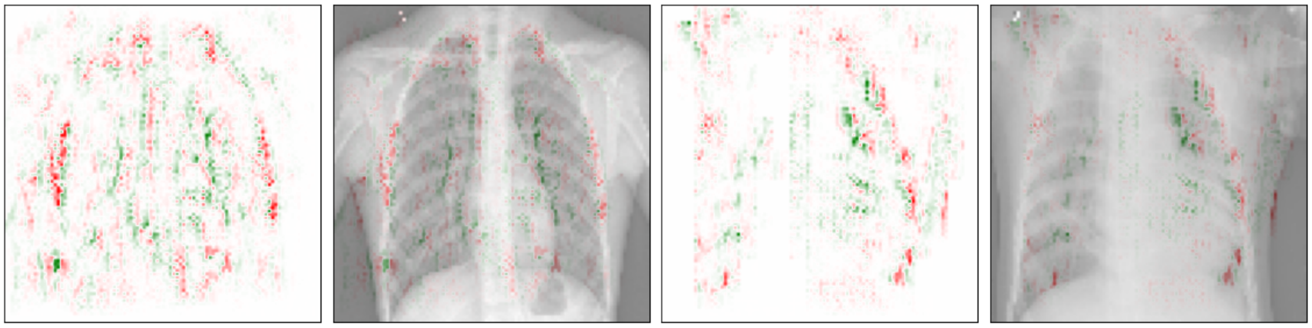

Figure 3. From Left to Right: All Attribution Values from Patient with Pneumonia, Overlay of First Image on Original Image, All Attribution Values from Healthy Patient, Overlay of Third Image on Original Image

4.3. Training With Class Weights

The testing results displayed below are for the model trained using class weights:

Balanced Accuracy: 90.7%
Precision: 92.0%
Recall: 99.2%
Specificity: 98.1%

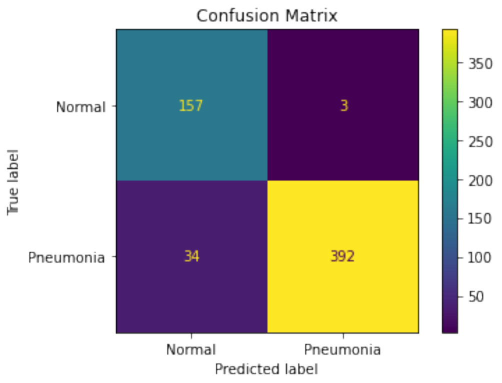

Figure 4. Confusion Matrix for Model with Class Weights

As shown with a very high specificity score, when training using class weights, our model became much more effective at identifying the normal x-ray images. However, our model became less effective at finding images containing pneumonia, indicated by the drop in balanced accuracy and precision scores. This could be due to the higher penalty we assign to the loss function for misclassification of the normal class.

Since we assigned a higher penalty to the loss function for misclassification of the normal class, the model appears to have learned to treat normal images with more caution, which may have resulted in more misclassifications of the majority class in order to potentially avoid the higher penalty.

In regard to identifying normal x-ray images, our classifier was successful. However, it allowed more pneumonia images to misclassified as normal x-ray images. Therefore, it perform as well as our previous model when it came to detecting pneumonia in an x-ray image.

4.4. Class Activation Maps

By passing in our trained model and the last convolutional layer of our model into Captum's 'GuidedGradCam' class, we generate an instance of GuidedGradCam that we can use to calculate the attributions of an image. Passing a testing image into our GuidedGradCam class's 'attribute' method, we obtain a tensor containing values corresponding to the element-wise product of GradCam and Guided Backprop attributions. To perform our analysis, we visualize our resulting tensor using Captum's 'visualize_image_attr'.

There is no definitive way to quantify how successful a CAM is, as for any model we can generate a correct CAM. Since the CAM is dependent on the model, however, if the model itself is inaccurate then we can determine why it may be classifying images incorrectly by looking at the CAMs. Likewise, if a model is accurate we can analyze what details it is using to reach a conclusion. We will define success as whether or not we can interpret any patterns from the maps, with the added goal of seeing if the patterns we locate reflect real-world practices. Since our model achieved a high accuracy, we are looking to determine what details the model is looking at when correctly classifying an X-ray as healthy or containing pneumonia.

Using Figure 3 as an example, we can make some immediate observations regarding how the model classified images as normal, or containing pneumonia. As a reminder, the main detail radiologists look for in X-rays when making a diagnosis is an excess of whiteness in the lungs [2]. Since most images contain the spinal chord (white line going vertically through the image), the heart (white lump in the middle of the image), and the diaphragm (white region beneath the heart) regardless of health, we immediately see that few attributions occur in these areas, as they provide little information to the model. Green pixels represent positive attributions while red pixels represent negative attributions.

Student Name	Contributed Aspects	Details
Andrew Novokshanov	Class Activation Map Implementation and Analysis	Assisted with the CNN implementation, and implemented Guided GradCam and Guided BackProp using Captum. With the Captum results, analyzed class activation maps to see how results compare to real radiological analysis.
Shaun Jacob	Data Preprocessing and CNN Implementation	Preprocessed the image data, implemented a CNN Binary Classifier, and tuned each model to best performance. Trained models with and without using class weights. Analyzed epoch graphs and confusion matrices to tune hyper-parameters.

Table 1. Contributions of team members.

Looking at the second image from the left, which contains healthy lungs, the positive attributions are located on the outskirts of the heart. Specifically, the positive attributions are located at areas where the lungs do not have a whiteness. The negative attributions are located along the rib cage. Looking at the rightmost image, which contains lungs afflicted by pneumonia, we can see a similarity in the locations of attributions. The positive attributions are again located near the heart, but are now in areas containing whiteness. The negative attributions were once again located near the rib cage.

Despite the rightmost image being classified the opposite of the second image (pneumonia versus healthy), the locations for both positive and negative attributions remain very close to one another: positive attributions are near the heart, and negative attributions are near the rib cage. This suggests that the model looks at primarily these two regions when making a diagnosis, and tends to make a decision based on the attributions located near the heart. We suspect that the attributions located on the rib cage are a result of the model getting confused as to where the lungs end, believing the whiteness of the rib cage being the same as whiteness in the lungs. However, due to the higher density of attributions within the lungs themselves, the rib cage alone is insufficient to sway a model's classification.

In conclusion, when the area surrounding the heart is dark, the model predicts normal lungs, and when the area surrounding the heart is white or glossy, the model predicts the lungs contain pneumonia. As we were successful in interpreting a pattern for how our model makes classifications, we were ultimately successful in our goal for using CAMs. Furthermore, the patterns we found directly correlate with how professionals examine X-rays when looking for pneumonia [2].

References

[1] Mateusz Buda and Atsuto Maki et al. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. 2

[2] J. Cleverley and J. Piper et al. The role of chest radiography in confirming covid-19 pneumonia. *BMJ*, 370, 2020. 3, 4, 5

[3] T. Hu and M. Khishe et al. Real-time covid-19 diagnosis from x-ray images using deep cnn and extreme learning machines stabilized by chimp optimization algorithm. *Biomedical Signal Processing and Control*, 68(1), 2021. 1

[4] X. Liu and K. Gao et al. Advances in deep learning-based medical image analysis. *Health Data Science*, 2021(1):1–14, 2020. 1

[5] Ramprasaath R. Selvaraju and Abhishek Das et al. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *Computing Research Repository (CoRR)*, 2016. 2

[6] F. Shamshad and S. Khan et al. Transformers in medical imaging: A survey. *EESS*, 4. 1

[7] G. Shankey and P. Singh. State-of-the-art review of deep learning for medical image analysis. *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 421–427, 2020. 1

[8] I. Valova and . Harris et al. Optimization of convolutional neural networks for imbalanced set classification. *Procedia Computer Science*, 176:660–669, 2020. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020. 2

[9] S. Yadav and S. Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1), 2019. 1