X-Ray Image Analysis Using Convolutional Neural Networks

By: Justin Merkel

I. Definition

Project Overview

X-Ray imaging is a medical analysis technique in which x-ray waves are sent through the body in order to get an image of what the inside of a person's body looks like. In the past, these images had to be analyzed by a doctor in order to evaluate their meaning. With the advances in image recognition, however, computers are able to recognize more and more patterns in human medical data. Image recognition using deep neural networks have recently been applied to medical image data, and these models have already seen some success [1].

In order to facilitate researchers in creating more effective machine learning models for x-ray analysis, the U.S. National Institute of Health has recently released a dataset of anonymous chest x-ray images. This dataset provides over 100,000 images of chest x-rays along with some relevant information on the patient in the x-ray such as sex, age, and visit number. There are eight common pathological diseases presented in this dataset: atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, and pneumothorax [2].

Problem Statement

The goal of this project is the build a machine learning model for flagging x-ray images that are very likely healthy vs. those that need a doctor's attention. This would allow doctors to focus more time on the 'unhealthy' x-ray images and skip the x-ray images that require no attention. To do this, a convolutional neural network was trained to categorize any x-ray image as 'healthy' or 'unhealthy'.

Evaluation Metrics

Note that in the context of this model, a 'positive' prediction would be a prediction that marks an x-ray as 'unhealthy', ie. having one of these known diseases, whereas a negative prediction marks the x-ray as 'healthy', or having none of the known diseases.

---

[1] Machine Learning in Medical Imaging. Miles Wernick-Yongyi Yang-Jovan Brankov-Grigori Yourganov-Stephen Strother - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4220564/

[2] NIH Clinical Center provides one of the largest publicly available chest x-ray datasets to scientific community. https://www.nih.gov/news-events/news-releases/nih-clinical-center-provides-one-largest-publicly-available-chest-x-ray-datasets-scientific-community

For this model, while precision should be taken into account, recall should be given much more weight because avoiding false negatives is much more important than avoiding false positives. Allowing a false positive through means the doctor has to look at an x-ray that ends up being healthy. A false negative, however, means that we have marked an 'unhealthy' x-ray as not important; leading the doctor to ignore a patient that.

Keeping these things in mind, the model will be evaluated using the f-beta metric. Using this formula, beta can be adjusted to give more weight to precision or recall, with a beta value of one giving equal weight to both, a beta value less than one giving more weight to precision, and a beta value of greater than one giving more weight to recall.

$$F_\beta = \frac{(1+\beta^2)\cdot \text{true positive}}{(1+\beta^2)\cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}}$$

Specifically, a beta value of three was used. This metric gives a good indication of how the model is performing, while also emphasising good recall over good precision [3].

The other metric will be evaluated with is binary accuracy, a commonly used evaluation metric for binary classifications models.

$$\text{Accuracy} = (\ (\text{True Positives}) + (\text{True Negatives})\ ) / \text{Total Predictions}$$

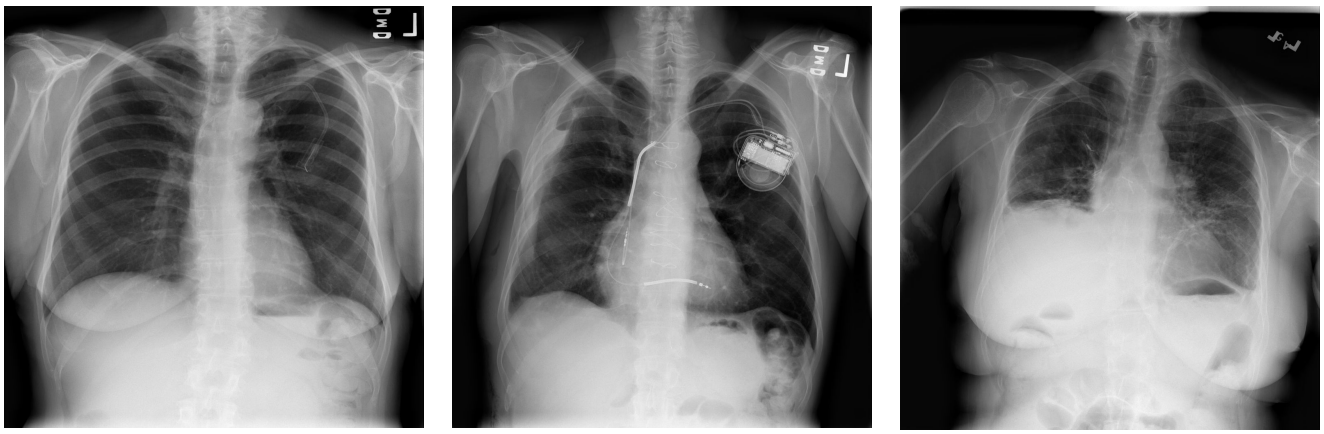
While this is not given as much importance as f-beta, it is worth using because it is a simple yet effective way to determine if the model fits the data.

II. Analysis

Data Exploration and Visualization

Each image either contains one to many of these diseases or marked as 'no finding'. Any images labeled as 'no finding' will be considered 'healthy', while the rest will be considered 'unhealthy'.

[3] Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). *Journal of Machine Learning Technologies*



(a) (b) (c)

Figure 1: A sample of images from the dataset: a) healthy, b) unhealthy, c) unhealthy.

Each image is a grayscale portable network graphics (.PNG) image file that is sized 1024x1024 pixels. The distribution of dataset based on this classification is as follows.

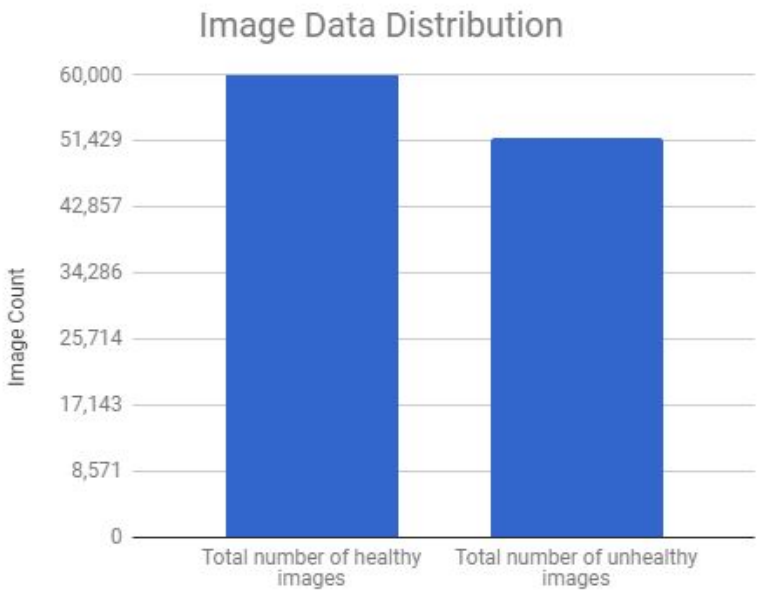


Figure 1: Distribution of healthy vs. unhealthy images.

The distribution of images is close to evenly split, with around 53.8% of all images being healthy and 46.2% being unhealthy.

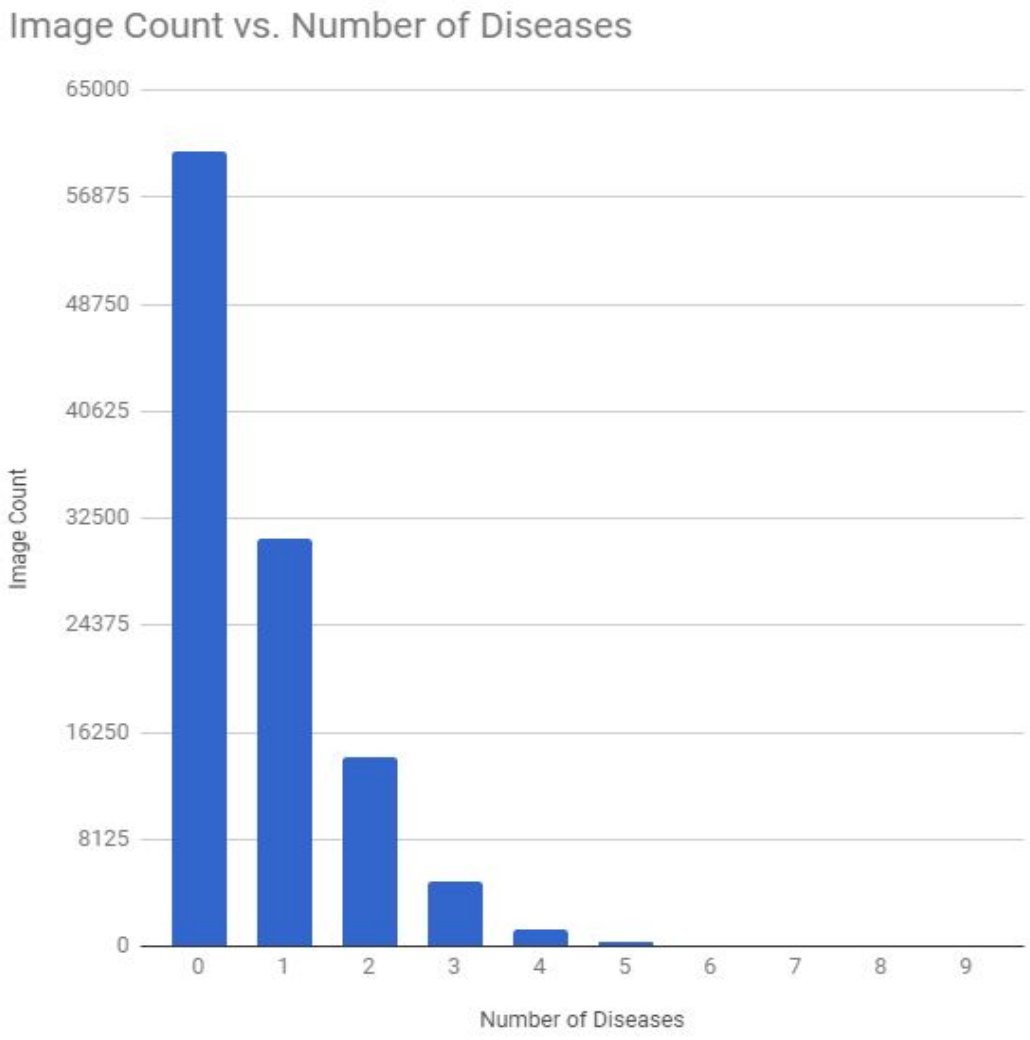


Figure X: Number of diseases per image.

As stated earlier, each image might contain more than one disease. The above graph shows an aggregation of how many diseases each image have. As one can see, most image contain either no diseases (marked healthy) or have at most two diseases. Not visible are the 86 outliers that have more than five diseases, 3 of which had eight or more.

Algorithms and Techniques

The model trained for this task was a neural network, specifically a convolutional neural network. A convolutional neural network was chosen because these types of neural networks tend to perform better on image classification [4].

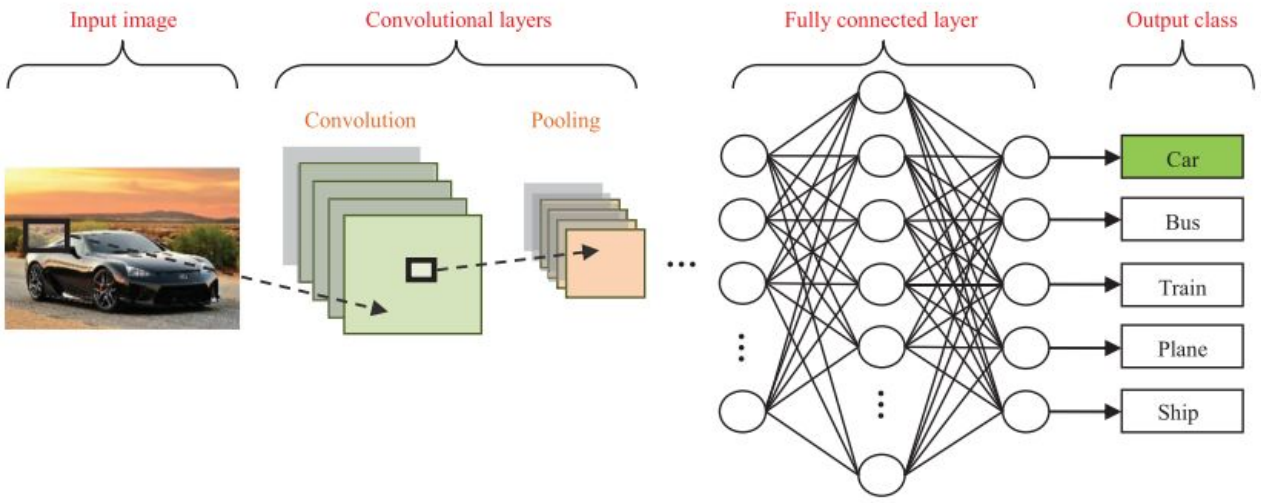


Figure X: An overview of convolutional neural network design.

Convolutional neural networks are made up of convolutional and pooling layers. Convolutional layers detect patterns in certain sections of an image. Pooling layers reduce the spacial size of how those patterns are represented. This reduces both the number of parameters in the model as well as helps to avoid overfitting. This process may be repeated several times, each repetition allowing for more and more complex patterns to be recognized in the an image. These layers eventually connect to a block of fully connected layers that ends with an output layer. For classification problems like this one, the number of output nodes is always equal to the number of classes possible in the dataset.

Due to the complexity of the task and difficulty of designing these networks from scratch, however, transfer learning was used for this problem rather than building the network from scratch. In short, transfer learning is taking an existing neural network's structure and weights and changing it to fit a different problem domain.

For this project, the ResNet50 model was used as the 'base model'. ResNet50 was used because of its success in classifying the imagenet dataset while not requiring as many operations as most of the alternatives. What makes ResNet50 unique is that it uses residual layers along with the standard convolution and pooling layers. Residual layers feed output information from previous layers forward to layers further into the network. This helps reduce overfitting while ensuring that each convolutional block learns something different rather than outputting the same information.

[4] Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review, Waseem Rawat-Zenghui Wang - Neural Computation - 2017

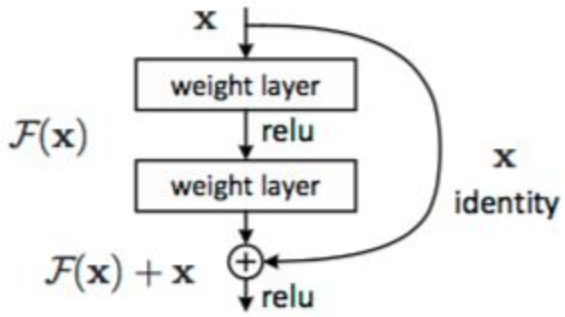


Figure X: The basis of a convolutional layer.

Neural networks measure and tweak performance based on a predefined loss function. As the model trains, loss if calculated and information is back-propagated, using an optimization function, in an effort to reduce loss. For binary classification problems, binary crossentropy is the common standard for measuring loss and so was the loss function used for this model. The optimization function used was stochastic gradient descent (SGD). This was used primarily because it performed the best in the several trial and error runs that were used to tweak the hyperparameters of the model (this will be discussed in more detail in the next section). Below is a very high-level representation of the process, from input to output [5].

Figure 3: Outline of the data flow, from input to prediction.

Benchmark Model

X-ray image analysis is currently a human problem; and no standard model currently exists for analyzing these images that does not require human involvement. In actuality, however, the benchmark model can be thought of as always guessing that an image is ‘unhealthy’, because that is what hospitals do currently by treating every x-ray as an image that requires human analysis. For this reason, the benchmark model for this project will be the model that always guesses an image is ‘unhealthy’.

[5] Deep Residual Learning for Image Recognition. Kaiming He-Xiangyu Zhang-Shaoqing Ren-Jian Sun - 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - 2016

III. Methodology

Data Preprocessing (Before Training)

Before the model could be trained the images had to be reclassified. The NIH dataset contains archives of every image as well as a csv file that maps image file names with patient characteristics and the illnesses associated with that image file. Using the csv file, a new file was written that mapped image file names to either a zero or one. Images were given a zero if the associated illnesses column in the original csv was marked as 'No Finding'. All other images had one or more of the nine illnesses associated with them, and thus were mapped to one. In this way, the original data was reclassified into one of the binary classifications.

Once this was done, all images were split moved into either the training, testing, or validation groups.

	Healthy Images	Unhealthy Images	Total Images	% of Total Images*
Training	43,445	37,280	80,725	72%
Testing	12,130	10,294	22,424	20%
Validation	4,785	4,185	8,970	8%
Total	**60,360**	**51,759**	**112,119**	**100%**

Figure 2: Distribution of images. *Percentages rounded to the nearest percent.

The testing group was for evaluation only, and did not affect the training process in any way. The training group was the group of images that the model was trained on and fit to. The validation group was used to validate the success of the model during training, as well as avoid overfitting to the training set. With all the images reclassified and assigned to groups, the training could begin.

Image Preprocessing

Before images can be run through the model, image preprocessing must occur in order to run the images through the model. First, all images were resized from their original size of 1024x1024 pixels to 512x512 pixels. This was done for efficiency, as the hardware running the model is not equipped to handle anything larger given the size of the dataset.

Next, the image data was loaded into a tensor with a shape of (512,512,3), where the first two numbers represent the width and height of the image in pixels and the last number representing the three color channels (red, green, and blue). All images in the dataset are grayscale, however, so all three color

channels are always the exact same. By doing this, we are able to run our grayscale images through the convolutional neural networks that were originally trained on color images.

Lastly, all pixel values are scaled so that every number is between 0 and 255, with 0 being black and 255 being white. This is known as normalization, and it is done because it allows for a larger contrast range between light and dark pixels.

Implementation

1. Obtaining Bottleneck Features

Before training begins, bottleneck features were obtained to speed up the transfer learning stage. Bottleneck features are obtained by taking a base model (specifically the ResNet50 model), removing the top layer, and running all training and validation images through one time. We will then save the outputs from this process and use them as the input data when training the top layer during transfer learning.

This is possible because all the layers of the base model will always remain the same during transfer learning. Because the weights of the convolutional layers never change, the output from this part of the network will always be the same for the same image. By using bottleneck features rather than full image data, however, training can occur in a fraction of the time.

2. Transfer Learning

Once the bottleneck features were obtained, it was time to begin training the new top layer. This new top layer was structured as follows…

a. A global average pooling layer.
b. A fully connected (dense) layer of 512 nodes. Output was run through a tanh activation function.
c. A fully connected layer of 2 nodes. Output was run through a softmax activation function.

These layers were trained using the bottleneck features obtained in the previous step. Finally, these nodes are connected to a dense layer of two nodes. Each output node represents one of the two classifications that are run through a softmax activation function. This will output the probability that the input image belongs to the classification that the output node represents[6]. Note that during the transfer learning stage, a dropout of 0.4 was added to ensure that all nodes in this top layer were given the chance to train. This top layer was trained for 500 epochs.

3. Fine Tuning

After the transfer learning stage, results were further improved through fine tuning. During this stage, the base model was combined with the top layer trained during the previous stage. Then the real image data was run through this network. While fine tuning, the weights in the top layer as well as a some of the layers in the base model that connect to the top layers are adjusted, while the rest of the weights in

[6] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

the network are frozen. Epochs take much longer during the fine tuning process, because bottleneck features cannot be used and because more layers are adjusted during back propagation. For this project, the ten layers that connect to the top layer were unfrozen. The model was trained this way for 5 epochs.

Adjustments and Refinement

This model went through several changes until the rights structure, format, and steps were taken to train a model that could learn the dataset. Here are the most notable challenges and solutions to the difficulties that occurred during the coding process.

1. Batch Processing

Initially, my plan was to hold the all the data in RAM during training. While it was possible for the smaller projects of the past, it was obvious early on that this would be impossible to do with a dataset of over 100,000 images. To solve for this, a.) Images were downsized from 1024x1024 to 512x512, and b.) the code was changed so that the dataset was processed in batches. The batch size was adjusted throughout the process, but in the end 32 was the batch size used.

2. Top Layer Structure

The first model had an output of a one node dense layer with a sigmoid activation function. While this seemed a good idea at the time, the model never seemed to learn anything during transfer learning. Output was consistently 0.5 and would not move much beyond that. It was not until I changed the output layer to be a two node dense layer with a softmax activation function and changed the activation function of the previous layer from relu to tanh did any learning occur during the transfer learning stage.

3. Overfitting During Fine Tuning

The original idea was the more epochs the better the model would be. This was changed when I ran the fine tuning stage for the first time for 15 epochs. The model overfit the training data to the point of being useless, and it was decided that the fine tuning epochs should be 5 instead.

IV. Result

Model Evaluation and Validation

After five hundred transfer learning epochs, the model improved a bit. After 500 epochs loss decreased consistently but very slowly. Results were positive, though; accuracy rose from 0.5138 to 0.5722. Additionally, both loss and accuracy trended with the training, indicating that overfitting did not occur while the model was learning to fit the data better.

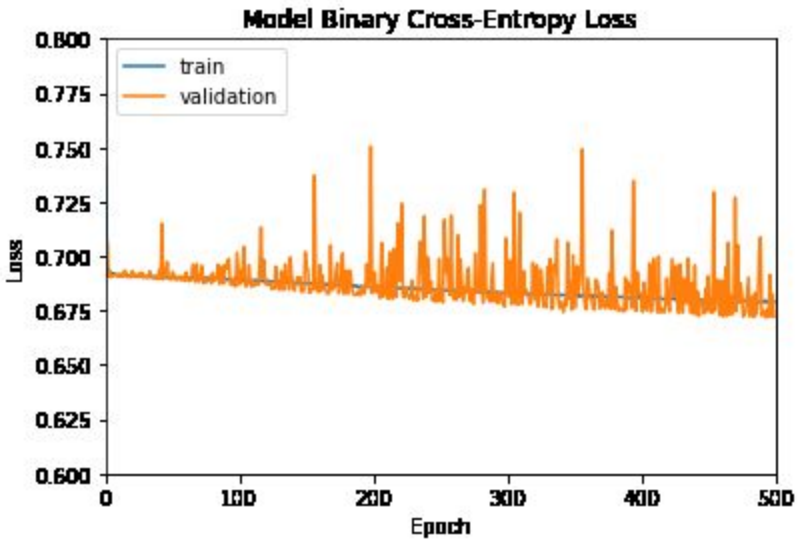


Figure 4: Binary Cross Entropy During Transfer Learning.

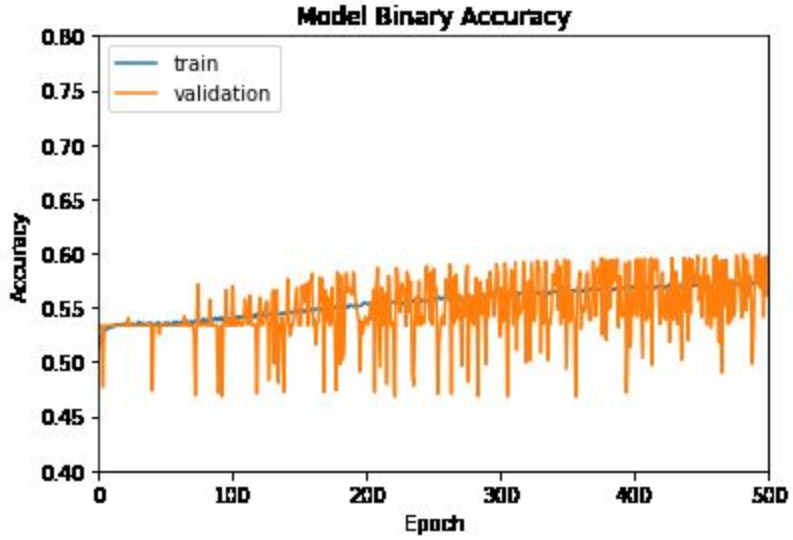


Figure 5: Accuracy During Transfer Learning.

Further improvements were made during fine tuning. After 5 epochs, loss dropped and accuracy rose once again. Results for the training and validation sets begin to separate after the fourth epoch, indicating that some overfitting may have occurred. Despite this, accuracy rose and loss dropped in both groups, indicating further model improvements.

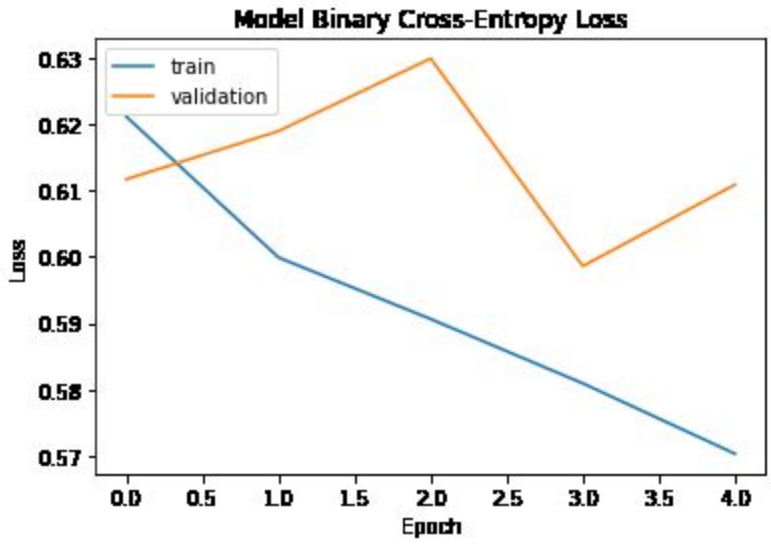


Figure 6: Binary Cross Entropy During Fine Tuning.

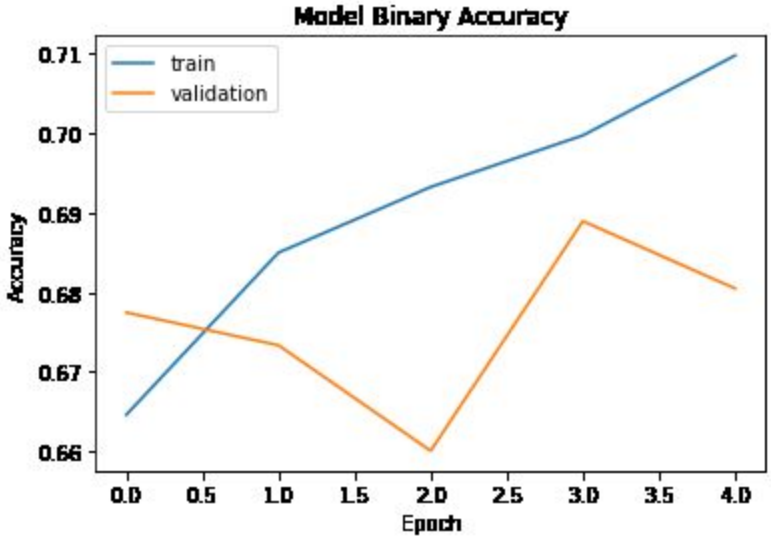


Figure 7: Accuracy During Fine Tuning.

Once training was complete, the final model was run against the test set.

	Predicted Healthy	Predicted Unhealthy
Actually Healthy	7937	4193
Actually Unhealthy	2936	7358

Figure 8: Confusion matrix for the final model's predictions of the test set.

This equates to an an f3-score of 0.7062, and an accuracy of 0.6821.

Justification

The base model simply predicts every image as unhealthy. So the benchmark can be evaluated as such…

	Predicted Healthy	Predicted Unhealthy
Actually Healthy	0	12130
Actually Unhealthy	0	10294

Figure 9: Confusion matrix for the benchmark model.

This equates to an f3-score of 0.8946 and an accuracy of 0.4591. While the f3-score is higher, the benchmark model holds no practical value. Despite this, the model I trained also has little practical value. For a model to be used in hospitals, the performance would have to be better. In particular the number of false negatives would have to be much lower. I have several ideas for improving this model which I will describe in the following section, but in its current state the model trained here has significant results but enough to be practical in a hospital setting.

V. Conclusion

Free-Form Visualization

As stated in the first section, false positive predictions are not nearly as hazzardas as false negatives. The model trained is more accurate than guessing, and has a among all wrong predictions, false positive predictions outnumber false negative predictions by nearly 3:2. The number of false positives can be decreased further, however, at the price of accuracy.

For this problem, making a prediction of healthy is inherently risky. This model runs images through and eventually outputs a probability that the image is healthy and a probability that the image is unhealthy. The results above are a product of the model always predicting healthy if the probability of it being healthy was greater than 0.5. But what if we used the exact same model, but told it to only predict healthy if the probability of this image being healthy was 0.6, or 0.8, or 0.9?

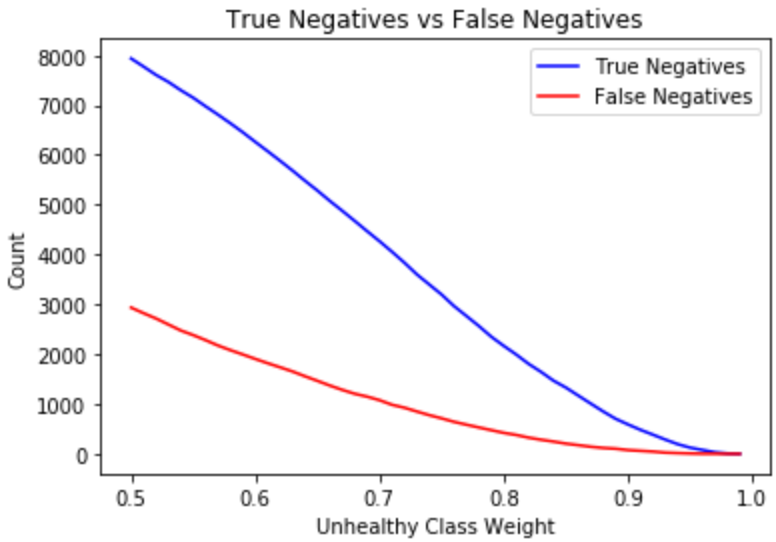


Figure 10: True negatives vs. false negatives as more weight is given to the unhealthy prediction.

Above is the prediction results of that type of model. The graph shows the number of true and false negatives as more weight is given to unhealthy predictions (keep in mind there are 22424 in the test set in total). Another way to read this is, the red line is how many unhealthy x-ray images marked by the model as healthy, nd the blue line is how many healthy x-ray images we marked as healthy and were thus did not need to be analyzed by a human. So what is the ideal weight? That depends on the how the model is being used, and what the cost of making a wrong prediction is.

Reflection

In conclusion, in this project I trained a convolutional neural network that would take in x-ray images as input and output the probability that that image showed signs of illness. This was done by first applying transfer learning to the ResNet50 model, and then fine-tuning that model to find the best fit for the data.

The most difficult part of this project was the size of the dataset. Over 100,000 images is a large dataset, and every model was run on my personal desktop computer. This made making changes to the model difficult in the end, as training the entire model took several hours each time. This hardware limitation also limited the results of the model. Given better hardware, this model could have been trained for much longer and could have been much more accurate.

Despite this, this project was a great way to learn how to solve image classification problems convolutional neural networks. At times it was difficult to find relevant information for some of the problems I was having, but this forced me to form a much better understanding of keras, tensorflow, and neural networks in general. Overall, I learned a lot during this project and found it both challenging and rewarding.

Improvement

Several improvements could be made to this analysis. This analysis was severely limited by the hardware running the models. All analysis occured on a personal desktop computer, using an Intel i5 6600K processor, 16GB of RAM, and an Nvidia GTX 1060 graphics card. More powerful hardware would allow for the model to process the images at their full sizes of 1024x1024, as well as allow for more training iterations to occur in a reasonable amount of time.

While transfer learning was determined to be the correct approach for this problem, the base models used for analysis may not have been. All five of the evaluated base models were trained on the imagenet dataset, which contains no medical imaging classifications and as well as contain color images. These models are great for general image recognition use, but there is, perhaps, a better base model that would fit this specific problem.

Lastly, this dataset contained additional data about every image. By combining these predictions with a model that takes this other data into account could potentially give the model a significantly larger amount of data that could improve performance.