Machine Learning for Medical Diagnosis

Bennett Hellman, Jeffery Mayolo, Joseph Zuccarelli

April 20, 2022

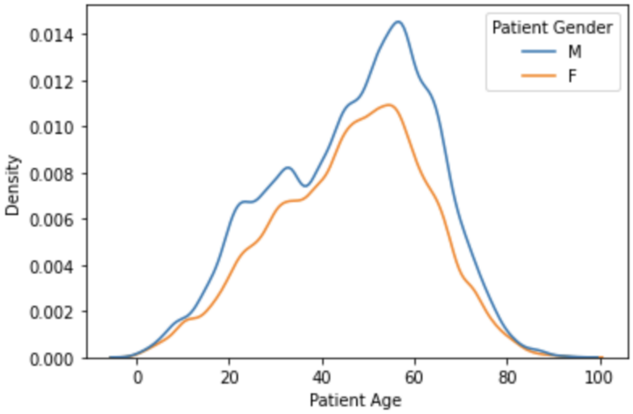
1 Introduction

Given the rise of big data, machine learning is currently being applied to inform decision making across various fields of study, one of them being healthcare. There are numerous applications of machine learning within the medical field that can be used to drastically improve patient care and outcomes. For instance, researchers are continuing to develop classifiers for the detection and/or diagnosis of disease. These classifiers have ranged anywhere from decision tree models trained on basic patient vitals all the way to convolutional neural networks (CNNs) trained on large quantities of high resolution X-ray images [1]. In the following paper we accomplish the latter, as we demonstrate how to build a chest X-ray pathology classification system using transfer learning on a real-world X-ray dataset.

2 Data

The full dataset under analysis in this study contains 108,948 frontal-view X-rays of 32,717 unique patients [2]. Each observation in the dataset includes a patient ID, a follow up number, an X-ray image, various characteristics of the image, patient demographic information (i.e., age and gender), and a diagnosis label. In this study, individuals' X-ray images will serve as our predictor, while the diagnosis label will serve as our outcome of interest.

Figure 1: Patients (Male/Female) by Age

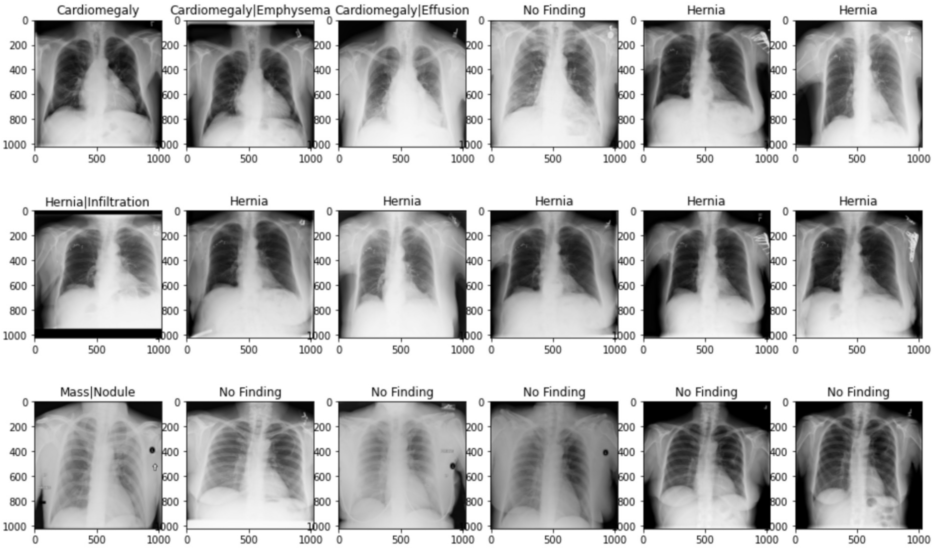


First, we explore the demographic characteristics of our patients–age and gender. Figure 1 displays the distributions of males versus females by age. Note that the density of males in the dataset appears to be larger than the density of females at just about every age. Also note that the highest density of patients (males and females) lies somewhere between 50-60 years old.

Next, we explore our predictor, the X-ray images. Note that in the original dataset these images are 1024 by 1024 pixels; however, during the model training phase we plan to downsize the images for ease of computation. Figure 2 included on the following page displays 16 sample images with diagnosis labels.

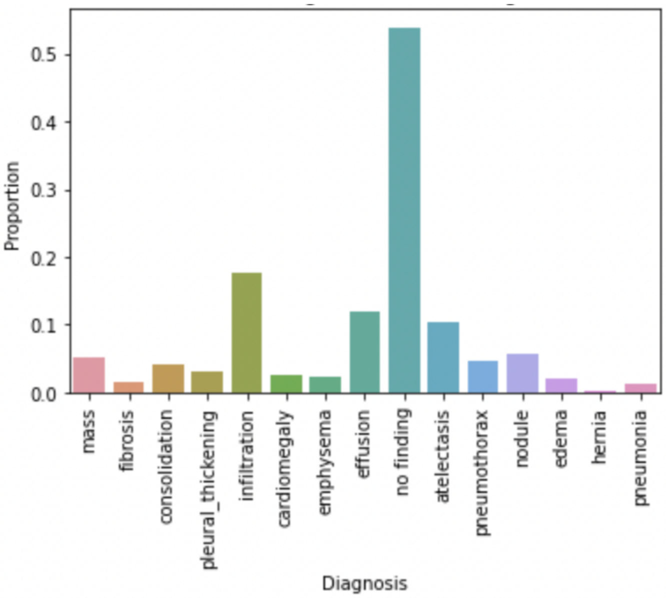
Finally, we explore our outcome variable, the diagnosis label. Note that there are 836 distinct diagnosis labels. However, this arises from the permutation of 15 distinct diagnoses: mass,

Figure 2: Sample X-ray Images with Diagnosis Labels



fibrosis, consolidation, pleural thickening, infiltration, cardiomegaly, emphysema, effusion, no finding, atelectasis, pneumothorax, nodule, edema, hernia, pneumonia. Thus, we extract each label and generate 15 binary outcomes variables. The proportion of each outcome label included in the dataset is displayed in Figure 3. Note that over half of the images are labeled as no finding. The most prevalent disease appears to be infiltration, as approximately one-fifth of the X-ray images receive this label. This dataset clearly suffers from class imbalance, which we must account for when building our classifier model.

Figure 3: Percent of Images with Given Diagnosis

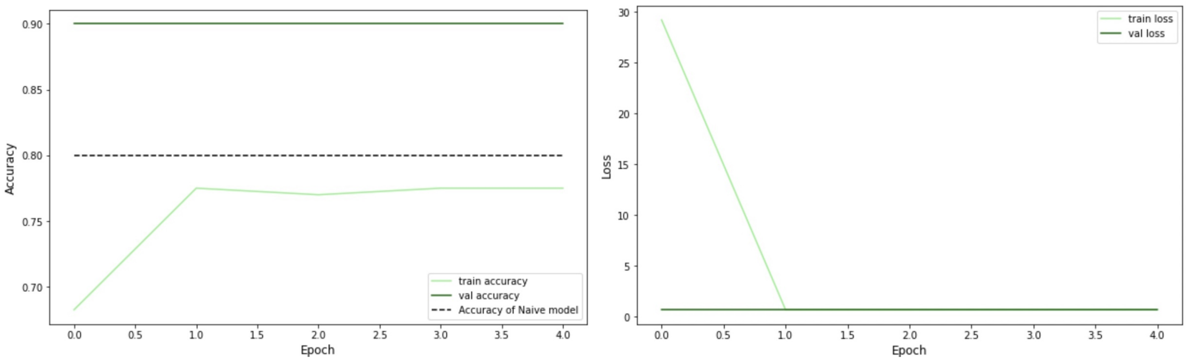


Note: In order to see the Python code used to create these plots, please refer to the attached Juypter Notebook.

3 Modeling

Given the exploratory data analysis performed above, we will investigate the following question in this project: **How can we apply transfer learning to develop a CNN model that accurately predicts a diagnosis label given a chest X-ray image?** In order to answer this question, we will begin our investigation with a baseline model using a relatively basic CNN architecture (see Appendix A for more details concerning the structure of this model). Note that we train this model using only the first 500 X-ray images from the original dataset as additional images will take an enormous amount of computation time. This model is set to predict the binary response of whether or not the patient is experiencing inflation (the most frequent prediction). Figure 4 displays the accuracy and loss of our baseline model as a function of the number of training epochs. Due to the class imbalance problem, it appears that the model predicts "no findings" for each X-ray and achieves 0.9 classification accuracy for each epoch on the validation set. As expected, the performance of the model is not great, likely due to the simplicity of our model along with the small sample size.

Figure 4: Baseline Model Performance



(a) Accuracy vs. Epochs

(b) Loss vs. Epochs

Note: In order to see the Python code used to build this model and create the resulting plots, please refer to the attached Juypter Notebook.

In order to improve upon this baseline, we plan to experiment with several popular CNN architectures such as DenseNet121, Inception V3, AlexNet, and VGG [3]. We will use various data augmentation techniques such as noise addition and rotation to increase the amount of data available for the fine tuning of the networks. In the fitting of each network, we also plan to explore the use of different techniques for addressing class imbalance such as weighted loss functions. Ultimately, we will evaluate the performance of each network and provide a recommendation for the best classification system based on our results.

References

[1] Igor Kononenko. “Machine Learning for Medical Diagnosis: History, State of the Art and Perspective”. In: *Artificial Intelligence in Medicine* 23.1 (2001), pp. 89–109.

[2] Xiaosong Wang et al. “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*. 2017, pp. 2097–2106.

[3] Satya Mallick. “Transfer Learning for Medical Images”. In: *LearnOpenCV* (20021).

Appendix A

Table 1 included below displays the architecture of our baseline CNN model referenced in the Modeling section of the paper.

Table 1: Baseline CNN Model Architecture

Layer	Output Shape	Num. Params
Convolution	(None, 1024,1024,16)	32
Max Pooling	(None, 512, 512, 16)	0
Convolution	(None, 512, 512, 16)	272
Max Pooling	(None, 256, 256, 16)	0
Convolution	(None, 256, 256 ,16)	272
Convolution	(None, 256, 256 ,16)	272
Max Pooling	(None, 128, 128, 16)	0
Dropout	(None, 128, 128, 16)	0
Flatten	(None, 262144)	0
Dense	(None, 32)	8388640
Dropout	(None, 32)	0
Dense	(None, 1)	33