Practicum I: Predicting Lung Disease Using Deep Learning

Odes Mullins

Regis University

Predicting Lung Disease Using Deep Learning

**Introduction**

The goal of this project was to predict lung disease from lung X-ray images using deep learning predictive models. Having a good deep learning predictive model would be a great tool to use to predict lung disease and to identify the type of lung disease. I downloaded 12 zipped files from the National Institutes of Health (NIH) containing over 100,000 lung X-ray images as well as a csv file which described each X-ray image contained within the 12 zipped files. I evaluated the csv file to determine the images to use for training and testing deep learning models. I created multiple deep learning models with machine learning to train and predict the lung diseases for the images used. The deep learning models were analyzed and modified to determine which model had the greatest predictive accuracy.

I evaluated the data and concluded that several categories of X-ray images included multiple diagnoses or that certain types of categories were very similar to each other making it difficult to distinguish between the types in a predictive model so I opted to only select a few categories to use for training and predicting. Although there were over 100,000 images in the original dataset I selected just over 4,000 images for testing due to the exclusion of those categories which included multiple diagnoses or were too similar to other categories I chose to use, making it difficult to distinguish between them. I performed the required tasks in Python within a Jupyter Notebook in Google Colab.

**Data Selection and Cleaning**

The data used for this project was extracted from the NIH Clinical Center website under the images folder. The images were in 12 zipped tar files and there was a Python program included to download the images by first downloading the Python program then executing it to download the 12 zipped files. I downloaded all the zipped files and then used 7-Zip to unzip and extract the images into a single directory.

There were several files included on the website, in addition to the images, which I also downloaded and evaluated and I determined the Data\_Entry\_2017.csv file was the only additional file I needed for this project. The csv file contained the list of all the X-ray images with descriptions of the X-rays. I determined that the only two columns needed for this Deep Learning project was the Image Index column and the Finding Labels column which has the diagnoses of the X-ray images. I identified the unique values in the Finding Labels column of the csv file and there were almost 850 unique values. I evaluated the Finding Labels to determine the best approach for selecting images to train and test deep learning models. Most of the unique values were multiple categories for an image. I determined that trying to train and test a model using X-ray images with multiple diagnoses was confusing and would be difficult for a model to determine the correct classification. I decided to only use six unique Finding Labels images for training and testing since those values were not combination categories or similar diagnoses. I created a new csv file which only contained the six Finding Labels which were Emphysema, Fibrosis, Hernia, Mass, Pneumonia, and No Finding. The new csv file contained all the image indexes for the X-ray images to be used for deep learning. I used a Python script to extract all the unzipped images to make it easier to evaluate and structure the data into directories. There were some missing images I encountered during the extraction process which I removed those rows from the new csv file. I created six additional scripts to read the selected images and place them into six training directories which represented the six finding labels. I created two additional Python scripts to randomly split the images in the six training directories into six validation directories which was at an 80/20 split and six test directories which were at a 90/10 split. I logged into Google Colab and uploaded the training, validation, and testing directories. It took a lot of effort to extract, evaluate, and restructure the data while removing bad data as well but this is a very important step which makes the processing step much easier.

**Exploratory Data Analysis**

I downloaded, unzipped and extracted the NIH X-ray data files. There was a total of 106,838 X-ray images in the extracted directory. I also downloaded the companion csv file and evaluated the columns. The csv file showed there should have been 112,120 images so the two values didn’t match. There were 11 columns in the csv file but from evaluation I determined the only two columns needed for Deep Learning were Image Index and Finding Labels. The Image Index columns contained unique ids for each of the X-ray images and the Finding Labels gave diagnoses of the findings for each image. Some other columns had interesting information like Patient ID, Patient Age, and Patient Gender and Follow-up # but they weren’t relevant to the deep learning project. One interesting fact is that there were only around 30,000 unique patient ids with several patients having multiple X-ray images which I originally thought the images were all for different patients. The other four columns were about image size which I readjusted for my models so they were not important and could be ignored as well.

I decided to plot the counts for the various finding categories to see the distribution. Although only Image index and Finding Labels were needed for Deep Learning, I also decided to evaluate the Patient Age and Patient Gender as this may give some useful information on the distribution of the images. I loaded the csv file for the Deep Learning images and there were 4055 images and no null values were found. Then I created a few plots using matplotlib. The first plot was for Diagnosis Counts. This plot shows each of the 6 categories. I also included the counts for the categories in Figure 1 below. The Mass category X-ray images is almost half the distribution with Hernia being the lowest amount of X-ray images at about 100.

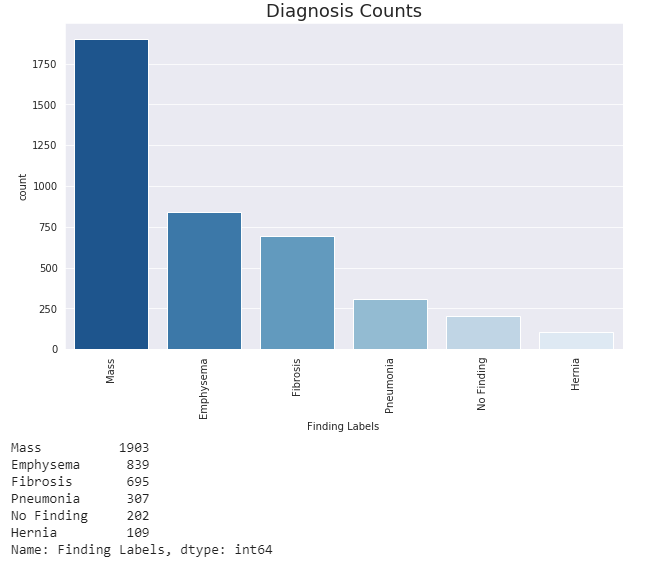


Figure 1: Diagnosis Counts for Categories

The next plot I created shows the distribution of the lung X-ray images by patient age as shown in Figure 2 below. The distribution shows somewhat of a bell curve with the majority of the images between ages 50 and 70 with the most patients around 55.

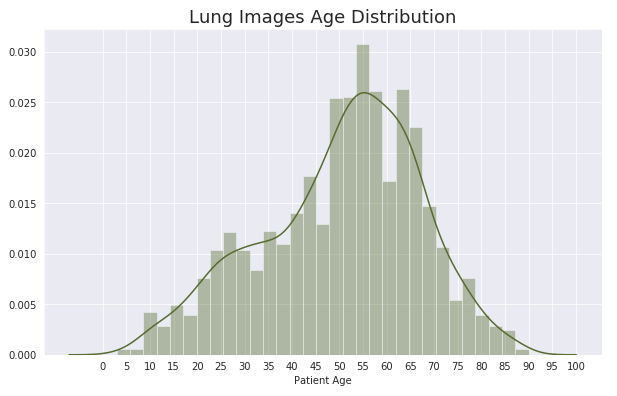


Figure 2: Diagnosis Counts for Categories

The next plot was a diagnosis count by gender as shown in Figure 3 below. There were more male and female X-ray images with the biggest difference being men with Emphysema or Mass where there were over 1,100 men with Mass which is about 400 more men than women with this diagnosis in the image dataset.

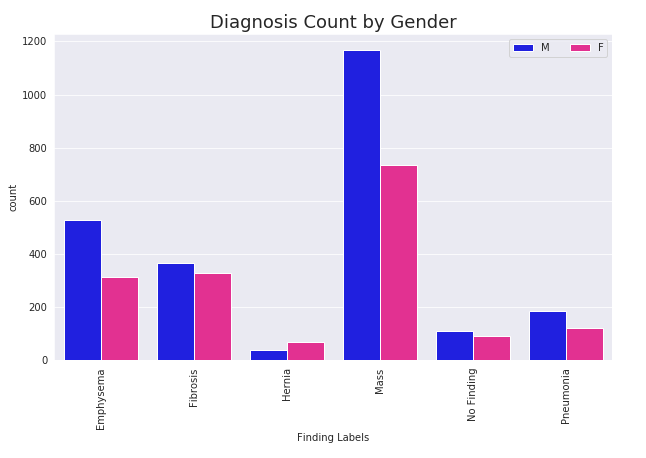


Figure 3: Diagnosis Count by Gender

The last plot I included was for diagnosis age distribution by gender as shown if Figure 4 below. This plot shows that there are some ages with more women and other with more men but both have a similar pattern of distribution over the various age ranges although there are 1,659 women compared to 2,396 men.

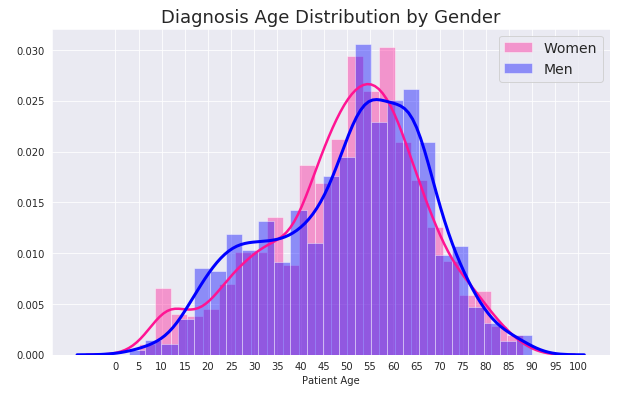


Figure 4: Diagnosis Count by Gender

**Regression Models**

I created a Google Colab Jupyter Notebook to build and execute regression models. The first step in the Jupyter Notebook was to perform general setup including setting display and importing necessary libraries for use in building and running the models. I also set the seed to 42 to be more consistent with the models. I mounted my Google Drive and set up all the directory paths pointing to Google Drive to make access and evaluation simpler for the X-ray images in the various directories. I printed the counts for the various directories showing there were 4055 total images. I normalized the pixels for the training, validation, and testing data to 150x150 then moved the data from the directories into the model. Then I fit the models. Note that since there were 6 total classifications which is categorical, the final dense layer uses softmax activation and has 6 classification outputs.

For the first model I ran, the training process used relu, Conv2D, MaxPool2D and BatchNormalization with 50 epochs and 100 steps per epoch. Model 1 had a testing accuracy of 46.79%. Model 2 was the same as Model 1 but I ran the training process for 20 epochs instead of 50 epochs and I got 46.17% testing accuracy.

I decided to use the Image Data Generator to generate variations of the original X-ray images to be used in the training and validation steps. This is a useful tool which generates variations of the original images creating more data for the training and validation processes. I show a sampling of the generated X-ray variations which I included below in Figure 5 and generated the X-rays using the data from the training dataset and validation dataset. Note that the original testing images are left intact and do not use the Image Data Generator.

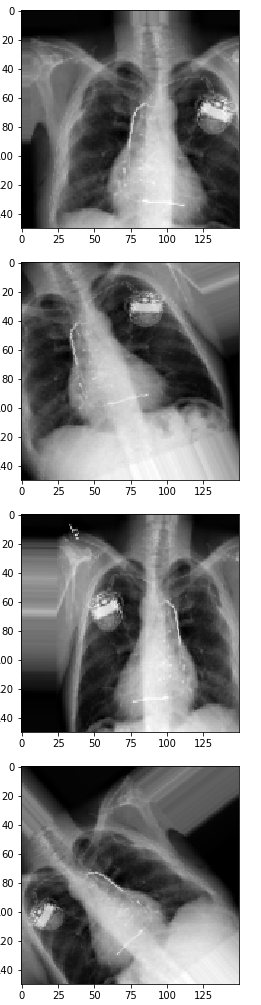


Figure 5: Generated images using Image Data Generator

I ran a similar process for Model 3 but I used the Image Data Generator with some different input values including 30 epochs for the training process. I got 46.79% testing accuracy in this model. I decided to use InceptionV3 pre-trained model with the Image Data Generator to see if I could improve the accuracy. I used the InceptionV3 model with all layers frozen and just a flattened layer and the final dense layer and I ran the training process for the model for 50 epochs. Model 4 gave me a very low testing accuracy of 16.15%. For Model 5, I ran InceptionV3 with the Image Data Generator the same as Model 4 but I did not freeze the last 20 layers since that went back far enough in the model to the conv2d layers to allow retraining. I got better testing performance in Model 5 of 45.76%. For Model 6 I ran InceptionV3 with all layers frozen and the Image Data Generator but I used two SeparableConv2 layers with L2 regularizers and weight decay and I also used dropout of 0.5 just before the final dense layer. Model 6 got a 31.68% testing accuracy. For Model 7, I used InceptionV3 again with one Conv2D layer and the I flatten and use batch normalization with two dense layers for 10 epochs with 200 steps per epoch. I got a testing accuracy of 22.15%. For Model 8, I used similar steps as Model 3 but I used L2 regularizer with weight decay, dropout, maxpooling, and normalization for 20 epochs. Model 8 got a 46.79% testing accuracy. For Model 9, I used VGG16, elu, SeparableConv2D, MaxPool2D, BatchNormalization, and L2 kernel regularizer with weight decay of le-4 for 30 epochs and 100 steps per epoch which got a testing accuracy of 53.21%. I also ran Model 10 using VGG19, elu, SeparableConv2D, MaxPool2D, BatchNormalization, and L2 kernel regularizer with weight decay of le-4 for 30 epochs and 100 steps per epoch and I got 49.69% testing accuracy. These are the models I included but I attempted many other versions of several of these models.

Let us discuss the various models and the progression of running the various models as well as potential model changes and improvements. Model 1 did not perform well but there was no sign of overfitting the data and the testing accuracy was similar to the training and validation accuracy. I reduced the number of epochs down to 20 on Model 2 but since there didn’t appear to be overfitting the accuracy stayed about the same as with Model 1 at over 46%. The accuracy still needed to be improved and I felt part of the reason was the smaller amount of X-ray images so I used the Image Data Generator to generate variations of the images but it only improved the accuracy a little on Model 3 to 46.79%. I decided to add the InceptionV3 pre-trained model to see if I could get an improvement in the performance using a pre-trained model. I included InceptionV3 with all layers frozen in Model 4 for 50 epochs and I got poor testing accuracy of 16.15%. Model 4 did not appear to do well overall but there were also signs of overfitting of the data making the testing accuracy even worse. I decided to try running again without freezing the last few layers of the InceptionV3 model. I ran Model 5 including InceptionV3 with the last 20 layers not frozen so they could be additionally trained with the new data. I chose 20 layers because there were a lot of the final layers of InceptionV3 which were not updating weights and the purpose was to update the weights. I got increased performance from model 5 as the accuracy was at 45.76% and there did not appear to be overfitting. Not freezing the last 20 layers worked a little better but still did not achieve performance above the 50% range nor did it surpass the performance of the first three models so I decided to use SeparableConv2D instead of Conv2D and I added the L2 Regulizer for Model 6. I ran Model 6 but there was overfitting again after about 15 epochs causing a testing accuracy of 31.68%. The training and validation accuracy was around 47% through epoch 15 of Model 6 which is still well below the 50% range. I tried to find other ways to improve the model using InceptionV3 so I run Model 7 with InceptionV3 having all layers frozen and variations of additional layers for 10 epochs and 200 steps per epoch. I got a poor accuracy of 22.15% so I decided that maybe using the InceptionV3 pre-trained model was not the greatest choice and I reverted back to Model 3 which didn’t include InceptionV3 to see if I could get an increased accuracy with some additional changes. I ran Model 8 adding L2 Regulizer with weight decay and ran the training process for 20 epochs and I got a testing accuracy of 46.79% which was the same as I got in Model 1 and Model 3. I used different criteria to train each of the three models so with all three performing with the same testing accuracy it appears this is about as good as a model will perform without possibly using other pre-trained models. I decided to try again to get a better accuracy using the VGG16 pre-trained model for Model 9 and I got 53.21% testing accuracy after running a few variations of the model. I then decided to try the VGG19 pre-trained model where I ran multiple versions again but ultimately included the model which got 49.69% accuracy. VGG19 got the second highest accuracy behind the VGG16 model although it did get above 50% accuracy. Getting the accuracy above 50% was very difficult.

**Possible Future Projects**

A future project could be to evaluate the Finding Labels and locate two or three categories which are different from each other to build a Deep Learning model. There are almost 850 Finding Labels so a lot of them do not have many records. Choose categories with hundreds or thousands of images to have more data to train and test the models. My focus for this project was to use categories which were unique classifications as I felt that data may fit a Deep Learning model better whereas there could be a future project to use two or three categories with multiple classifications to build a Deep Learning model to see if it performs better. Another project could be to use a similar amount of images for two to three categories for a Deep Learning Model.

**Conclusion**

I ran multiple models for this project including many models not included in the Jupyter Notebook and I surmised that none of the various models had much of an effect on testing accuracy for the NIH lung X-ray dataset. VGG16 was the only model which I could get above 50% testing accuracy. I utilized a lot of the tools learned from prior classes and the Image Data Generator has helped in prior assignments but I don’t feel it helped much with the NIH dataset. There were a few cases where there was overfitting but most models got testing accuracy within the range of training accuracy and validation accuracy. Some of the classifications only had a few X-rays so maybe that is part of the issue but from reviewing the X-ray images I think a lot of the various diagnoses were similar enough that the models could not discern the difference between them. I only picked six categories to use for this deep learning project but based on the accuracy I feel it would have been worse if I chose to use other images with multiple diagnoses for each image as that would have been almost impossible for a model to determine the correct category. I was disappointed in the NIH dataset to use for Deep Learning as it did not include enough X-ray images in categories which could easily be distinguished from each other. This made classification extremely difficult. I originally thought that having a dataset with over 100,000 X-ray images would be great to use which is the reason this dataset was chosen. One thing I discovered about this dataset is that there were only around 30,000 unique patients so the dataset was a lot more limited than it appears. A positive note is that I utilized techniques from prior classes to successfully process Deep Learning model creation and training. Using a pre-trained network can be extremely useful and I don’t feel I would have gotten above 50% without using pre-trained networks. It is important to choose the best pre-trained model for a specific task to fit the data but that can be a challenge and can take some research and experimentation to determine a pre-trained model which may work well with the data.

Below is a table detailing the testing accuracy for each of the 10 models. Model 9, which had the best match rate of 53.21%, is highlighted for easy identification. It helps to see the testing accuracy from the 10 models to compare them to each other.

|  |  |
| --- | --- |
| **Deep Learning Model** | **Testing Accuracy** |
| Model 1 | 46.79% |
| Model 2 | 46.17% |
| Model 3 (with IDG) | 46.79% |
| Model 4 (With IDG and InceptionV3) | 16.15% |
| Model 5 (With IDG and InceptionV3) | 45.76% |
| Model 6 (With IDG and InceptionV3) | 31.68% |
| Model 7 (With IDG and InceptionV3) | 22.15% |
| Model 8 | 46.79% |
| **Model 9 (With IDG and VGG16)** | **53.21%** |
| Model 10 (With IDG and VGG19) | 49.69% |

Table 1: Deep Learning Model Evaluation

References

NIH Clinical Center. ChestXray-NIHCC. Retrieved on 02/06/2020 from <https://nihcc.app.box.com/v/ChestXray-NIHCC>

Hacker Earth. Machine Learning. Challenge #2 – Deep Learning. Retrieved on 02/04/2020 from <https://www.hackerearth.com/practice/machine-learning/challenges-winning-approach/deep-learning-challenge-two/tutorial/>

Stack Overflow. Split data directory into training and test directory with sub directory structure preserved. Retrieved on 02/22/2020 from <https://stackoverflow.com/questions/46717742/split-data-directory-into-training-and-test-directory-with-sub-directory-structu>

Mapplotlib. List of named colors. Retrieved on 02/24/2020 from <https://matplotlib.org/3.1.0/gallery/color/named_colors.html>