Detecting Covid and Viral Pneumonia from X-ray Reports using Convolutional Neural Networks (CNN)

Ahnaf Abrar Kabir, College of Letters and Science, UW Madison, Computer Science (BS), Undergraduate Sophomore

Introduction:

In this programming assignment, I have found a topic that is closely related to the current global situation, i.e. COVID-19. I found a dataset on kaggle [1] where the author has provided 261 training images and 66 testing images of X ray reports classified into 3 sections:

1. Covid
2. Viral Pneumonia
3. Normal

I have decided to use a Convolutional Neural Network (CNN) model to do this project. My goal is to classify whether a given patient has Covid, Viral Pneumonia or neither using my CNN model.

Algorithm:

But what is CNN?

In deep learning, a convolutional neural network is a class of artificial neural networks, most commonly applied to analyze visual imagery.
Well, this goes hand in hand to the problem I am trying to solve. More generally, this falls under the bigger umbrella of Computer Vision.

CNN Model in this project:

I have developed a Deep Neural Network model with 2 convolution layers followed by 1 fully connected layer. This structure has been finalized after various fine tuning measures. Finally we have the following structure:

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_22 (Conv2D)           (None, 450, 450, 8)       208
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 225, 225, 8)       0
_________________________________________________________________
dropout_33 (Dropout)         (None, 225, 225, 8)       0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 225, 225, 16)      1168
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 112, 112, 16)      0
_________________________________________________________________
dropout_34 (Dropout)         (None, 112, 112, 16)      0
_________________________________________________________________
flatten_11 (Flatten)         (None, 200704)            0
_________________________________________________________________
dense_22 (Dense)             (None, 900)               180634500
_________________________________________________________________
dropout_35 (Dropout)         (None, 900)               0
_________________________________________________________________
dense_23 (Dense)             (None, 3)                 2703
=================================================================
Total params: 180,638,579
Trainable params: 180,638,579
Non-trainable params: 0
_________________________________________________________________
```

As you can see above,

- I started by passing a 450 x 450 image into a 2D Convolutional layer with 8 filters, with filter size of 5x5, and P=0, zero padding. The activation function in this layer was relu.
- Next I added a Maximum Pooling slayer with stride 2
- Next, 15% of the features are dropped
- Then again, I made the second 2D Convolutional layer, this time with 16 filters and filter size of 3x3, and P=0, zero padding.
- Then I added another Maximum Pooling slayer with stride 2,
- Then, 15% of the features are dropped again
- Nw, we move on to the Fully Connected layer of 900 hidden neurons and relu activation function.
- The last layer of the model is a Softmax layer with 3 output neurons:
    - 0 - Normal
    - 1 - Covid
    - 2 - Viral Pneumonia

Dataset:

Processing the X-ray image dataset [1] was the hardest part of this project. Initially, I had:

1. 251 Training images
    1.1. 111 Covid
    1.2. 70 Normal
    1.3. 70 Viral Pneumonia
2. 66 testing images
    2.1. 26 Covid
    2.2. 20 Normal
    2.3. 120 Viral Pneumonia

During training, I used 20% of the training images for validation. In Order to accomplish this task I used sklearn.model_selection.train_test_split.

In this dataset, the images had multiple extensions including:

1. .jpeg,
2. .jpg,
3. .gif,
4. .png, and
5. .tga

I decided to convert all images to the .png file type. I did this using python's opencv2 library.

Next, the images I had were of various resolutions ranging from 1500x2000 pixels to 4500x4500 pixels. After various trials and eros, I decided to resize all the images to 450 x 450 pixels.

Here, are 3 examples of the X-ray images used in this project:

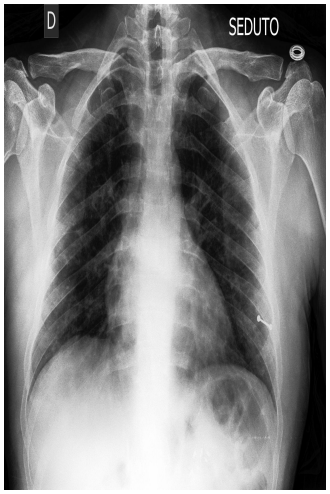

Fig: Covid

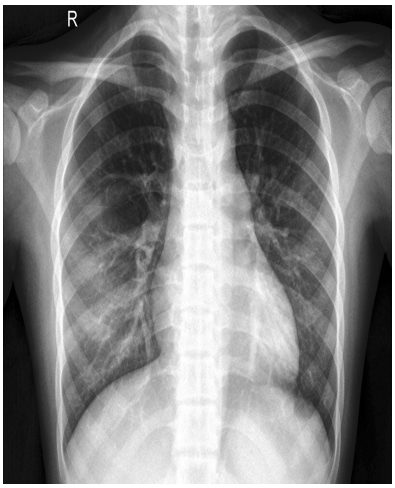

Fig: Normal

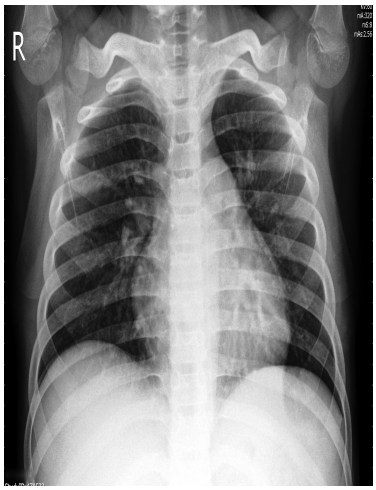

Fig: Viral Pneumonia

Lastly, I converted all the 450 x 450 X-ray images to their corresponding pixel values and scaled them to the range [0,1]. I did this using python's matplotlib.image library.

Before passing my data to the CNN model for training, I shuffled the data to avoid overfitting. I also did Data Augmentation using keras.preprocessing.image.ImageDataGenerator (More details in the pdf of the code under the section title, Data Augmentation).

Result:

During training of my CNN model, I tried various different hyperparameters, and the result are shown below:

Image Size	Number of Epochs	Minibatch size	Testing Accuracy
150 x 150	100	15	83%
250 x 250	1000	15	88%
350 x 350	100	14	89%
450 x 450	100	15	86%
450 x 450	50	14	92%

After training my model several times, I noticed a pattern. Increasing the image resolution affects the accuracy of the model prediction, which does make sense. Hence, I decided to use 450 x 450 pixel images. I wanted to increase the resolution more, but my personal computer was not able to do that much computation efficiently.

Similarly, I had to decrease the number of epochs, as I increased the image resolution due to inefficient computation by my device.

In the end, when I did my final model evaluation using the testing dataset, I managed to get a 92.4% accuracy. Shown below is a screenshot of my confusion matrix:

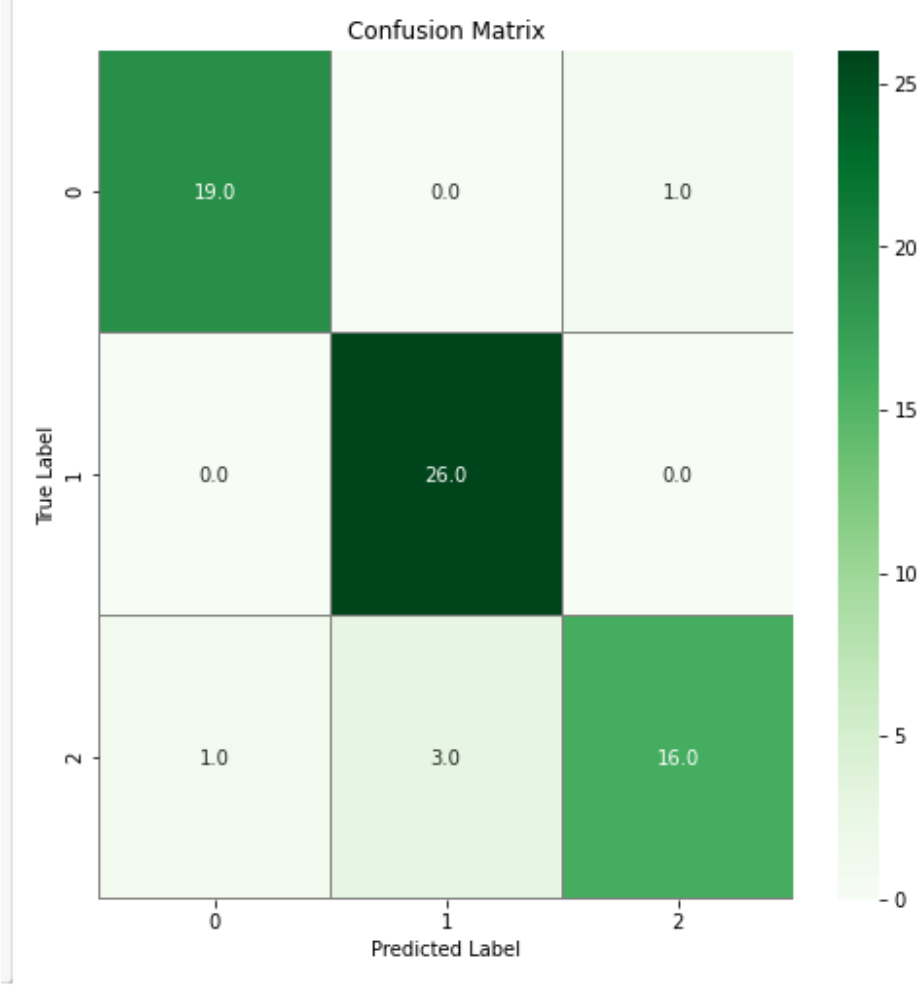

Fig: Confusion Matrix

Next, this is the final classification report of my CNN model:

	precision	recall	f1-score	support
0	0.95	0.95	0.95	20
1	0.90	1.00	0.95	26
2	0.94	0.80	0.86	20
accuracy			0.92	66
macro avg	0.93	0.92	0.92	66
weighted avg	0.93	0.92	0.92	66

Fig: Classification Report

References:

1. https://www.kaggle.com/pranavraikokte/covid19-image-dataset