Hooman Shiranimehr

Final Report: COVID-19 Detection Based on Chest X-ray

Problem Statement

Coronavirus disease 2019 (COVID-19) is a highly infectious disease that has affected almost every country in the world. Several tests have been developed to detect the disease at different stages. However, these tests can be slow and/or costly.

Here our goal is to investigate using patents' chest X-ray to detect COVID-19.

In the following sections we first discuss data wrangling. Then we talk about Exploratory Data Analysis. After that, we investigate several models and compare their performance. Finally, we explain our conclusion and future work.

Data Wrangling

Data is obtained from https://github.com/ieee8023/covid-chestxray-dataset. This is a public open dataset of chest X-ray and CT images of patients which are positive or suspected of COVID-19 or other viral and bacterial pneumonias (MERS, SARS, and ARDS). Data has been collected from public sources as well as through indirect collection from hospitals and physicians.

Dataset contains patients' information in a metadata file (Figure 1 and Figure 2) as well as corresponding X-ray images (Figure 3 and Figure 4).

	patientid	offset	sex	age	finding	RT_PCR_positive	survival	intubated	intubation_present	went_icu	in_icu	needed_supplemental_O2
0	2	0.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	N	Y
1	2	3.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	N	Y
2	2	5.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	N	Y
3	2	6.0	M	65.0	Pneumonia/Viral/COVID-19	Y	Y	N	N	N	N	Y
4	4	0.0	F	52.0	Pneumonia/Viral/COVID-19	Y	NaN	N	N	N	N	N

Figure 1

```
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   patientid               950 non-null    object
 1   offset                  697 non-null    float64
 2   sex                     870 non-null    object
 3   age                     713 non-null    float64
 4   finding                 950 non-null    object
 5   RT_PCR_positive         593 non-null    object
 6   survival                361 non-null    object
 7   intubated               248 non-null    object
 8   intubation_present      250 non-null    object
 9   went_icu                397 non-null    object
 10  in_icu                  335 non-null    object
 11  needed_supplemental_O2  90 non-null     object
 12  extubated               37 non-null     object
 13  temperature             78 non-null     float64
 14  pO2_saturation          119 non-null    float64
 15  leukocyte_count         16 non-null     float64
 16  neutrophil_count        28 non-null     float64
 17  lymphocyte_count        40 non-null     float64
 18  view                    950 non-null    object
 19  modality                950 non-null    object
 20  date                    661 non-null    object
 21  location                894 non-null    object
 22  folder                  950 non-null    object
 23  filename                950 non-null    object
 24  doi                     382 non-null    object
 25  url                     950 non-null    object
 26  license                 705 non-null    object
 27  clinical_notes          768 non-null    object
 28  other_notes             436 non-null    object
 29  Unnamed: 29             5 non-null      object
```

Figure 2

Figure 3

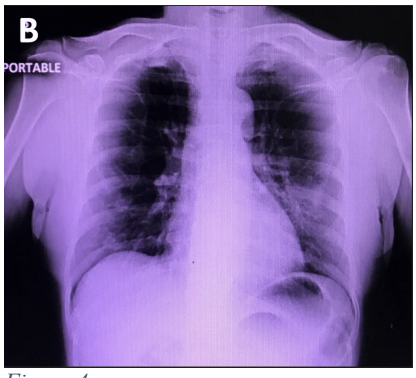


Figure 4

As it can be seen from Figure 1 and Figure 2, there are some columns like 'doi','url', 'license', 'clinical_notes' and 'other_notes' that are not relevant and are removed.

Also, some columns have missing values. These values replaced by “Unknow” or the average value of that columns as shown in Figure 5.

```
metaData['sex'].fillna('Unknown', inplace=True)
metaData['RT_PCR_positive'].fillna('Unknown', inplace=True)
metaData['survival'].fillna('Unknown', inplace=True)
metaData['intubated'].fillna('Unknown', inplace=True)
metaData['intubation_present'].fillna('Unknown', inplace=True)
metaData['went_icu'].fillna('Unknown', inplace=True)
metaData['in_icu'].fillna('Unknown', inplace=True)
metaData['location'].fillna('Unknown', inplace=True)
```

```
metaData['offset'].fillna(np.mean(metaData['offset']), inplace=True)
metaData['age'].fillna(np.mean(metaData['age']), inplace=True)
```

Figure 5

There are also 21 records with no image. We remove those records.

Next, we check ‘finding’ column for different patents. This column has values such as ‘Pnemonie’, ‘SARS’ and ‘COVID-19’ as shown in Figure 6.

```
metaData['finding'].unique()
```

```
array(['Pneumonia/Viral/COVID-19', 'Pneumonia', 'Pneumonia/Viral/SARS',
       'Pneumonia/Fungal/Pneumocystis',
       'Pneumonia/Bacterial/Streptococcus', 'No Finding',
       'Pneumonia/Bacterial/Chlamydophila', 'Pneumonia/Bacterial/E.Coli',
       'Pneumonia/Bacterial/Klebsiella', 'Pneumonia/Bacterial/Legionella',
       'Unknown', 'Pneumonia/Lipoid', 'Pneumonia/Viral/Varicella',
       'Pneumonia/Bacterial', 'Pneumonia/Bacterial/Mycoplasma',
       'Pneumonia/Viral/Influenza', 'todo', 'Tuberculosis',
       'Pneumonia/Viral/Influenza/H1N1', 'Pneumonia/Fungal/Aspergillosis',
       'Pneumonia/Viral/Herpes ', 'Pneumonia/Aspiration',
       'Pneumonia/Bacterial/Nocardia', 'Pneumonia/Viral/MERS-CoV',
       'Pneumonia/Bacterial/Staphylococcus/MRSA'], dtype=object)
```

Figure 6

A new column ‘COVID-19’ is defined based on ‘finding’ and it is set to ‘True’ when finding includes ‘COVID-19’ and ‘False’ otherwise. Then images corresponding to patients with and without COVID-19 are moved to a folder named ‘Positive’ and ‘Negative’, respectively.

Exploratory Data Analysis

After data wrangling, we get 929 images (563 COVID-19 positive and 366 negative). Checking some of these images reveals that image sizes are different. Therefore, we change size of all images to 256x256 as shown in Figure 7.

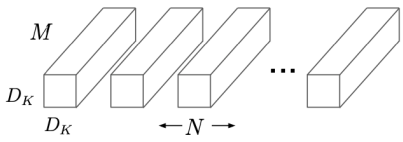
```
dstDir = '/content/drive/My Drive/Capstone3-Images'
posCls = '/Positive'
negCls = '/Negative'
os.makedirs(dstDir+posCls)
os.makedirs(dstDir+negCls)

covid_positive = metaData[metaData['COVID-19']]
print('Number of positive cases:', len(covid_positive) )
for i in range(len(covid_positive)):
    #print(i+1, 'of', len(covid_positive))
    filename = covid_positive.iloc[i,:]['filename']
    r = requests.get(url+filename, stream = True)
    with open(dstDir+posCls+"/"+filename, "wb") as file:
      for block in r.iter_content(chunk_size = 1024):
         if block:
             file.write(block)

covid_negative = metaData[metaData['COVID-19']==False]
print('Number of negative cases:', len(covid_negative) )
for i in range(len(covid_negative)):
    filename = covid_negative.iloc[i,:]['filename']
    #print(i+1, 'of', len(covid_negative))
    r = requests.get(url+filename, stream = True)
    with open(dstDir+negCls+"/"+filename, "wb") as file:
      for block in r.iter_content(chunk_size = 1024):
         if block:
             file.write(block)
```

Figure 7

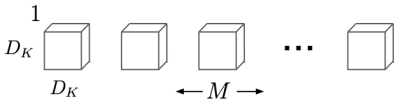
In order to evaluate our models, we use 70/15/15 split for training, validation and test. Also, given that training set is small and only consists of 650 images, we augment training set by generating 11 new images using random rotation, zoom, shift, etc..

Model Selection

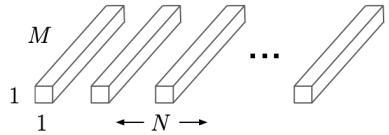
MobileNet is a lightweight architecture that uses depth-wise separable convolutions (Figure 8 and Figure 9).



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 8

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 9

For our base model, we use MobileNet with weights pre-trained on ImageNet. We discard the last layer of 1000 neurons and consider several models with 1 or 2 new layers added to the base model and different dropout values. Figure 10 shows 2 of the models considered.

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024,activation='relu')(x)
x = keras.layers.Dropout(0.6)(x)
preds = Dense(2,activation='softmax')(x)
model1 = Model(inputs=base_model.input,outputs=preds)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024,activation='relu')(x)
x = keras.layers.Dropout(0.7)(x)
preds = Dense(2,activation='softmax')(x)
model2 = Model(inputs=base_model.input,outputs=preds)
```

Figure 10

Notice that dropout values are relatively large as lower values lead to overfitting.

Figure 11and Figure 12 show loss and accuracy vs epoch for models 1 and 2. As it can be seen from these figures, in both models, accuracy and loss for training set are improved as value of epoch is increased. On validation set, we see some fluctuations but both accuracy and loss seem relatively stable. If we increase number of epochs to 20 or 30, we see gaps between the two curves for each model which means the model will overfit. We choose Model 1 as it has slightly higher accuracy and lower loss.

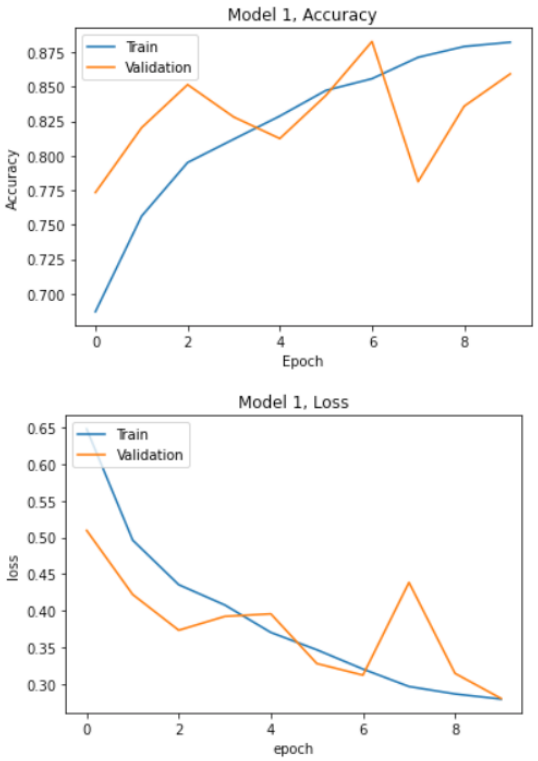


Figure 11

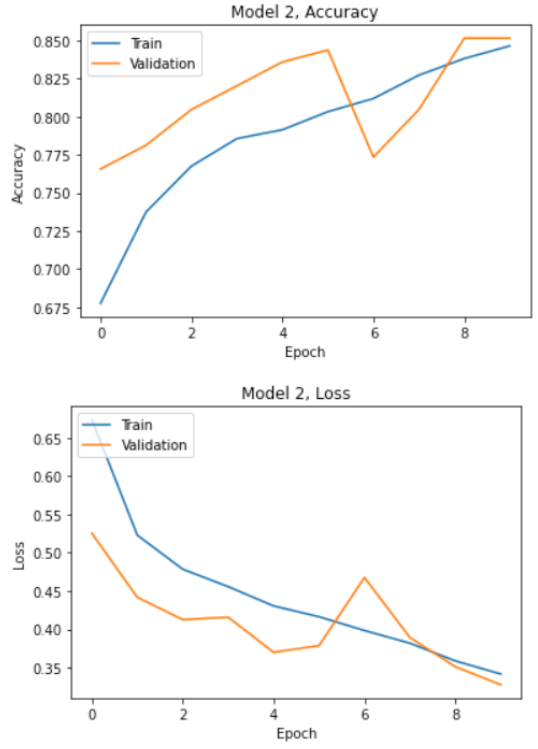


Figure 12

Next, let's consider ROC and Precision-Recall Curves for test Set. The results are shown in Figure 13 and Figure 14.

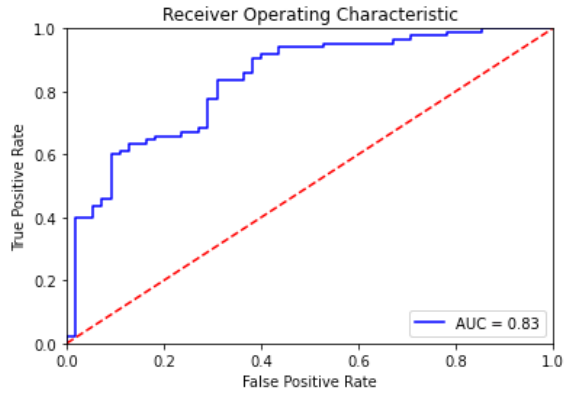


Figure 13

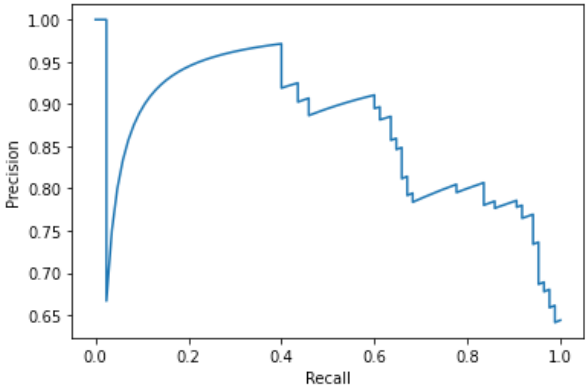


Figure 14

Figure 15 shows accuracy and confusion matrix for the case that threshold is set to 50% probability.

```
140/140 [==============================] - 1s 5ms/step
Accuracy: 0.7857142857142857
Confusion matrix:
[[33 22]
 [ 8 77]]
```

Figure 15

Conclusion and Future Work

In the previous sections we discussed data wrangling, EDA and modeling. As we saw, for our problem, Model 1 seems to achieve the best performance. Therefore, based on the requirements, one can use this classifier with a certain threshold to achieve the best acceptable/desired levels of precision and recall.

Note that as our ultimate goal is to detect a disease, it makes sense to focus on reducing false negatives. By doing so, we would we get more false positives which can be fine as the disease can be ruled out by other tests. However, if the false positive is too high, it can cause additional unnecessary cost for retesting and also concern in many patients and their families. Therefore, we need to try to find an optimal threshold to balance these two.

Overall, it can be a challenging task to find the optimal operating point (threshold) and other factors such as user experience, business and finance need to be considered in order to make the best decision.

Although we considered several different classifiers, further optimization is still possible by tuning hyperparameters such as number of additional layers and dropout ratios. Also, it is possible to use additional patients' information and consider a model with both structured and unstructured data.