ACM40950 - Project in Maths Modelling

Project on COVID-19 Detection using Radiography Images

Ninad Thaker - 20202926

8/15/2021

Background

The first known case of a highly contagious disease caused by Severe Acute Respiratory Syndrome CoronaVirus 2 (SARS-CoV-2) was identified in Wuhan, China, in December 2019. World Health Organization (WHO) declared it a Public Emergency of Global Scale or an Epidemic in January 2020. However, the disease kept spreading, and soon thousands deaths were reported. In February 2020, WHO named the disease as "COVID-19" and by March 2020, the disease was declared as a pandemic and has since then spread worldwide.

There is a long list of symptoms which includes Fever, Dry cough, Fatigue, Shortness of breath, Pain in chest, etc. Some of the symptoms are common while others are not and they could be severe. The severe symptoms starts as something very common headache, fever, cough or sore throat, and in a matter of days, it could becomes life-threatening in some cases. In majority of the cases where the person is suffering from severe COVID-19 have been found to have cough and Shortness of breath. In order to perform further and accurate diagnosis, doctors across the globe tired using the Radiography Images (Chest X-Ray). However, initial findings from the images were not accurate more than 70% because the Pneumonia and COVID-19 scare tissue looked almost identical.

Aim

The process of getting Radiography Images and providing accurate diagnosis under the immense number of cases is a tedious process. This project will help to accurately identify the presence of COVID-19 scare tissue within the chest cavity using radiography images taken from thousands of patients during the initial hospitalization. Given a large amount of images, a machine learning algorithm can be trained to identify the scare tissue with higher accuracy and most importantly almost instantaneously.

Data

The Data used here can be found on the kaggle[1].

Algorithm

The Convolutional Neural Network (CNN) is known to work well, and provide highly accurate predictions even when used on unseen images. The other benefit of using a CNN is that there is no human effort or prior knowledge required for creating features. CNN uses segments of an image as filters, and it learns particular

properties and characteristics of each filter. This features are in general term called as feature map. Since the radiography data is in the form of image, we will be utilizing CNN as the model to identify a unique feature map which on its own will be able to differentiate COVID-19 scare tissue from Pneumonia.

In order to create a CNN based model that works well on our given set of images, we first need to understand its architecture.

A typical layer of a CNN consists of 3 stages known as:

(1) Convolutional - where the feature map is created from segments of images

(2) Activation - mapping of feature map to learn complex pattern

(3) Pooling - reduce spatial size of input features to produce a specific output

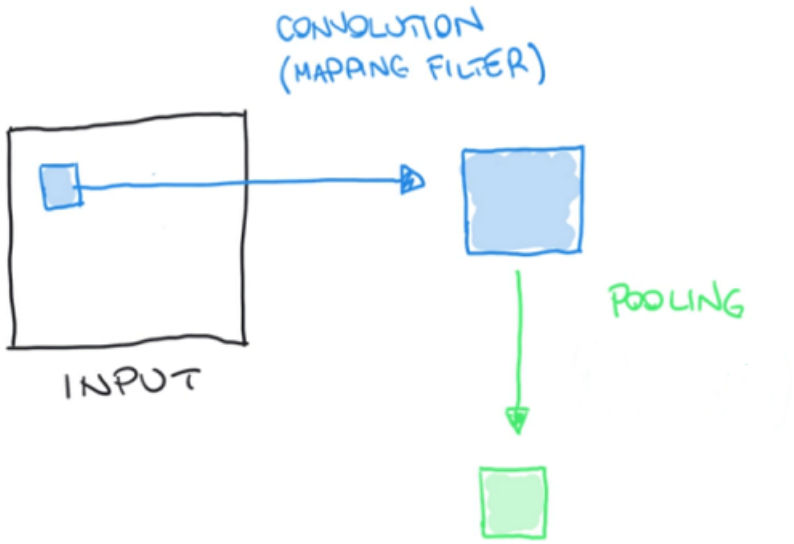

Figure 1: Stages

Architecutre of CNN

A general architecture of a CNN consists of 2 blocks:

(1) Convolutional Layers - contains the 3 stages explained above

(2) Fully Connected Layers - a standard multi-layer neural network

A layer of CNN takes image as input. An image encoded in a 3-D tensor of dimensions Height x Width x Depth. A tensor is an object which has multilinear relation between vectors or matrices. The height and width are self explanatory, however, depth here means color encoding of the image. If the image is RGB - has Red, Green and Blue color channels, it has a depth of 3, while if the image is a gray scale image, the depth of such image would be of 1.

The radiography images that we are using here, has Depth of 3, even though the actual X-ray would be in gray scale, their images are encoded in RGB. That said, normally it is very useful to keep the width and height dimension of the images same.

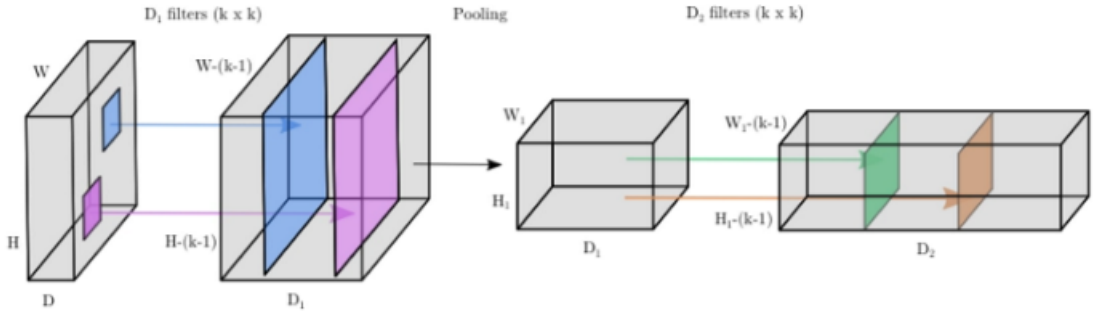

Figure 2: Dimension

Since we kept the height and width dimensions exactly same, we can now create a square shaped filter of dimension less than that of our actual image. This filter then is passed or scrolled though entire image as shown in the figure 3. The scanning of features is immediately followed by a pooling layer which reduces the output feature size by halving but preserves the depth. The figure is for illustrative purpose and shows a filter grid of size 2x2 such that the process can be easily visualize, while in reality we have used a filter size of 32 in the actual model.

The convolutional layer - a feature map or collection of filters from current layer is then applied to current map and outputs are activated using ReLU - Rectified Linear Unit function. The convolutional filter is called as kernel and in general it is a square matrix of weights of dimension k. If we denote the kernel as K, input image as X then output O of the convolutional is given by equation (1):

$$O_{(i,j)} = (X * K)_{(i,j)} = \Sigma_m \Sigma_n \; X_{(m,n)} K_{(i-m,j-n)} \tag{1}$$

where i & j denotes index of output matrix, and m & n denotes index of input image. Again it is easy to understand it from the figure 3. As the filter moves across the image, the corresponding output index changes accordingly. The final product is a convolutional feature map which is reduced in size.

The actual images are of the dimension 299 x 299 x 3, however, due to hardware limitation of system on which this model was trained, the size was reduced to half making the dimensions of the input image 150 x 150 x 3.

Model

The model we have used here contains mainly 3 convolutional layers and 2 fully connected layers. We have also used 2 dropout layers and a flatten layer to drop some of the random network links and convert 2D tensor to 1D tensor. This is to avoid overfitting of the model, w.r.t our images, it could lead to wrong diagnosis i.e. the actual image may belong to class Pneumonia or Normal for that matter while the model identified the image of class COVID and that is serious issue. The model parameters used here such as filter size, kernel size and dropout ratios are generally found to be most suitable. There is no specific reason for using them and it can be manipulated in order to fine tune the model if that is required. Once we have decided all the parameters, we select an optimizer whose task is to minimize learning error. In this case we have selected Adam with default learning rate since it is a combination of Adaptive Gradient (AdaGrad) and Root Mean Square Propagation (RMSProp) which tends to provide better optimization than any other optimizer. This can also be changed to understand the working or to compare different optimization methods if required.

The input image as it is passed through the network, changes the output shape and it is calculated based on the following formulas:

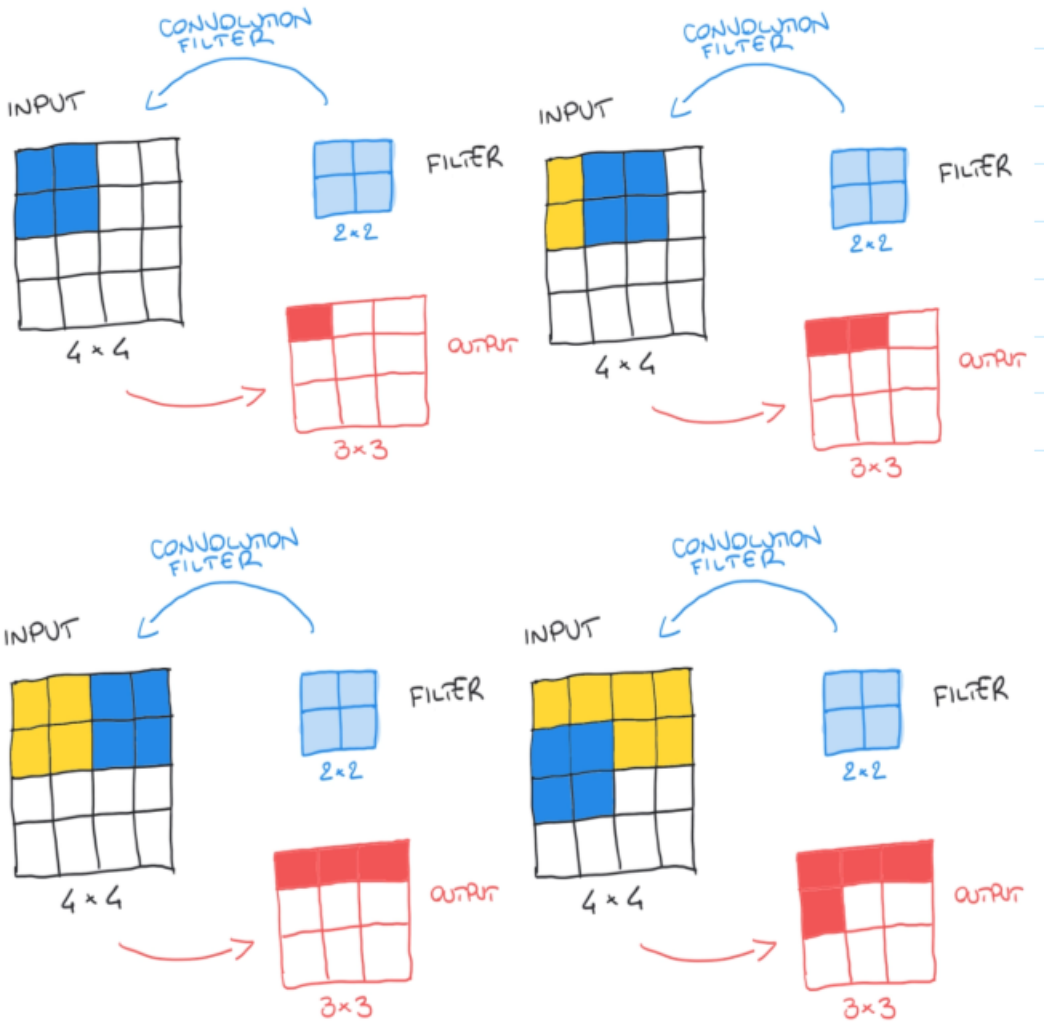

Figure 3: Filter

- Number of parameters in a convolutional layer is = ((kernel_size * stride + 1) *filters)
- Output shape of convolutional layer = ((input_height – (kernel_height – 1)), (input_width-(kernel_width - 1)), convo_units)
- Number of parameters learned in a maxpool layer is = 0
- Maxpool output shape = (convo_height/pool_height) * (convo_width/pool_width) * convo_units
- Number of parameters learned in flatten layer is = 0
- Flatten output shape = previous_layer_height * previous_layer_width * previous_layer_convo_units = scalar_value
- Number of parameters in the first dense layer is = dense_units * (scalar_value + 1)
- Number of parameters in the output/ dense layer is = (current_layer_dense_units * previous_layer_dense_unit) + current_layer_bias

Let's calculate output for first convolutional layer our use case

- Input image tensor size: 150 x 150 x 3
- Filter size: 32 x 32
- Kernel size: 3 x 3
- Padding: 0
- Stride: 1

The number of parameters are $((3 * 3 * 1) + 1) * 32 = (27 + 1) * 32 = 28 * 32 = 896$

Output shape is $((150 - 3 + 1), (150 - 3 + 1), 32) = (148, 148, 32)$

Maxpool output shape is $(148/2, 148/2, 32) = (74, 74, 32)$

Similarly, all the other parameters and output shapes of other layers can be calculated. The final summary of the model that we used can be present in the figure 4.

Model Training, Testing and Classification Report

Once we finalize the model we can train the images, however, our end goal is to identify the presence of COVID-19 in the new/unknown/unseen images that may be presented to the model in the future. Thus, we make a subset of images as training set about 80% in this case and another subset of images as testing set which is remaining 20% of the total available images. Once we have the general output of the training and testing, we can generate a classification report to further analyze the individual class results. The figure 5 shows the classification report from the our model training and testing.

We are interested in the F1 - score which is also known as F-score. This score provides the balance between precision and recall or in other words it is the accuracy for individual class. In the figure 5, we can see that in the case of training, the overall and individual accuracy of each class is ~99.8% except for COVID which is ~99.6%. In the case of testing, however, it is significantly different scenario. The overall accuracy is ~96.9%, the class Normal has accuracy of ~97.8% while the class COVID and Pneumonia has accuracy ~95%. The support count for each class represents the number of images on which the model training and testing was performed.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 17, 17, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 36992)             0
_________________________________________________________________
dense (Dense)                (None, 64)                2367552
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 2,460,995
Trainable params: 2,460,995
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4: Model Summary

```
Classification Report - Training:
              precision    recall  f1-score   support

       COVID    0.99689   0.99585   0.99637      2893
      NORMAL    0.99865   0.99865   0.99865      8153
   PNEUMONIA    0.99722   1.00000   0.99861      1076

    accuracy                        0.99810     12122
   macro avg    0.99759   0.99817   0.99788     12122
weighted avg    0.99810   0.99810   0.99810     12122


Classification Report - Testing:
              precision    recall  f1-score   support

       COVID    0.96170   0.93776   0.94958       723
      NORMAL    0.97146   0.98480   0.97808      2039
   PNEUMONIA    0.96911   0.93309   0.95076       269

    accuracy                        0.96899      3031
   macro avg    0.96742   0.95188   0.95947      3031
weighted avg    0.96892   0.96899   0.96886      3031
```

Figure 5: Classification Report

The training portion of our model is appears to be performing really well. It is the testing where we see some major difference. This may seem to be a good result, but since we are dealing with a virus which is known for causing life-threatening condition, it is better to understand the importance of the output results first before using this model as it is in real life. The impact that difference made can be visualized conveniently using a confusion matrix plot as shown in the figure 6. The Y-axis represents the actual classified numbers i.e. the actual class label of the image. The X-axis represents the predicted classified numbers i.e. the model predicted a label based on the learning.

We are mostly interested in the diagonal of this matrix as if the diagonals have all the numbers, we have 100% accuracy, which is good but that would also mean we may have overfitting issue. Regardless, from the figure 6 we can clearly see that there are entries on off-diagonals too. The entries on the off-diagonls represents the error that model has or the wrong classification of an image. As explained above, we can't really have wrong classification could it result fatal. Thus, we also need to check for those intersections on the matrix. Note that all the off-diagonals are important but since we want to identify COVID-19, we will address only the intersection which shows 44 images.

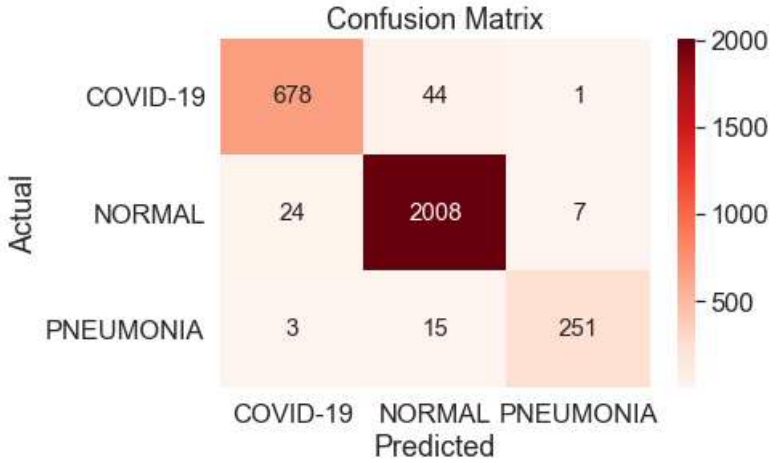

Figure 6: Confusion Matrix

It can be interpreted as the actual class of an image was COVID-19 while the predicted class of that image was Normal. Now we have 44 such images which actually belongs to class COVID while the model predicted them as Normal. To put that in more understandable words: We may end up telling 44 patients that they don't have COVID-19 while they were actually suffering from it. This is huge probelm considering how fast it spereads from one person to other, and also potentially discharging a patient instead of providing medical care. Although, this situation can be improved to a certain point by fine tuning the parameters and/or providing more number of images to train and test the model.

Grad-CAM Visualization

We saw how it is important to understand that a wrong classification can cause a lot of issues and since this is also a matter of saving time, we can create a visualization of the scare tissue presence. Grad-CAM or Gradient-wighted Class Activation Mapping uses the gradients from any target convolutional layer to create a local map which highlights important feature that the model has learned.

We will not discuss the entire workings of the Grad-CAM algorithm as it is out of the scope of this project. However, the links to workings and the actual example implementation can be found in the references[2,3].

Recall that in our case we have 3 Convolutional Layers, the Grad-CAM is applied to the output of the last layer. The layers as we know consists of 3 stages of which the last stage i.e. maxpooling is passed to the

algorithm. A heatmap of class activation is then generated from the image based on the detected features from the image. Finally, the heatmap is superimposed on the actual image to clearly show the presence of the COVID-19 scare tissue on the X-ray image.

The visualization of Grad-CAM can be seen in the figure 7. Here, we applied the Grad-CAM algorithm on 4 selected images from COVID-19 and Normal classes. The first 2 rows of images are from class COVID-19 while the last 2 are from class Normal. The images on the left column are the actual input images of chest X-ray, the images in the center column are the heatmap which we plotted after a little modification in the actual algorithm, and the images in the right column are the superimposed final output of our modified Grad-CAM. The highlighted area in the chest cavity shows the presence of COVID in the first 2 images while that is clearly missing in the last 2 images which shows that the model we trained was able to identify the labels accurately.

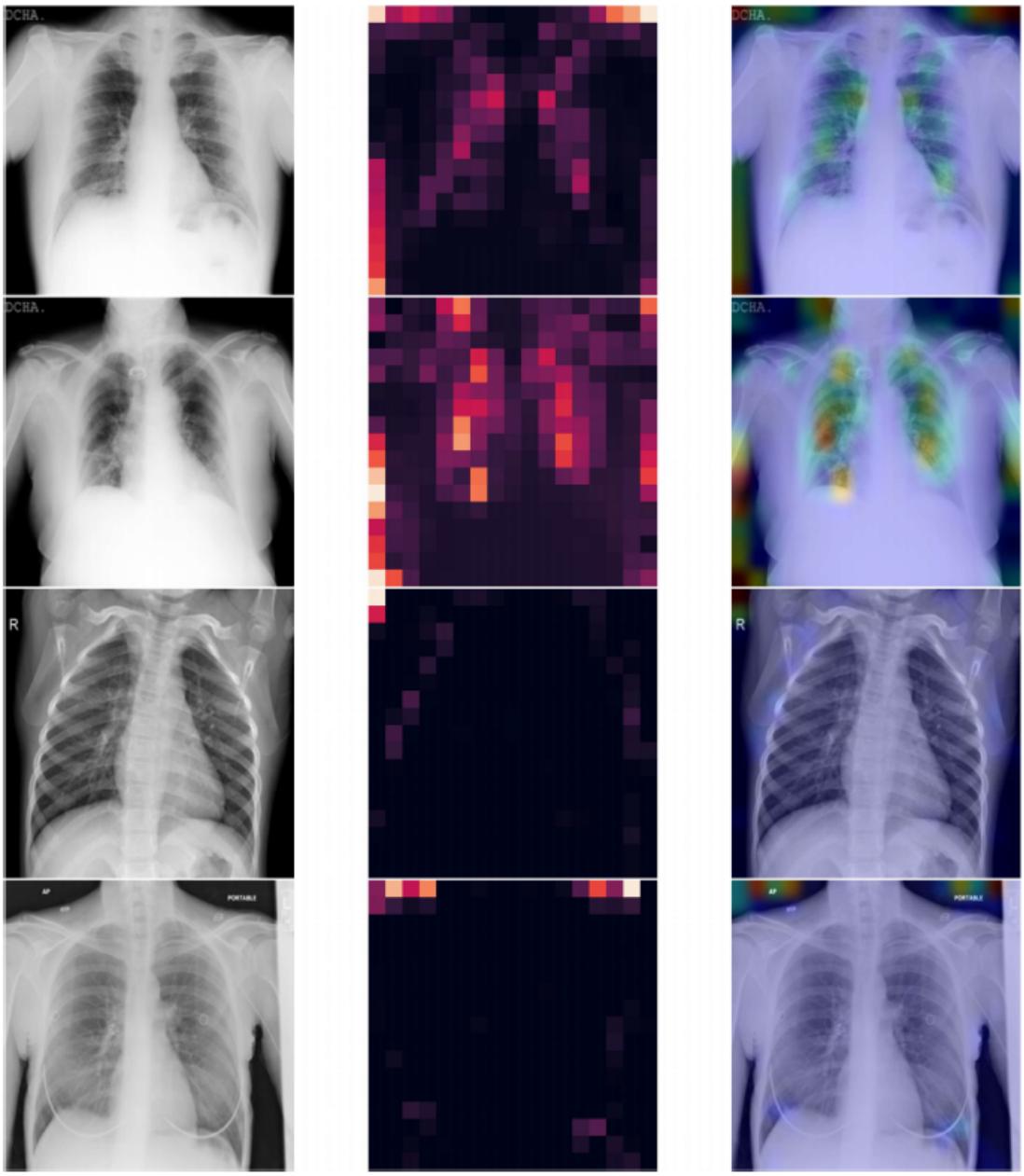

Figure 7: Grad-CAM Visualization

Conclusion

The Neural Network used here is by no means completely trustworthy as this is a medical application, and the output is critical for decision making, however, it can be improved as explained earlier. It should be also noted that the patients in the severe condition may also excert other symptoms which along with the output of this model should be considered for further diagnosis. The visualization results from Grad-CAM can help to highlight the potential area where the scare tissue can be found. This can also help in the cases where model predicted wrong classification label. Overall, the application of a Convolutional Neural Network can help to identify the presence of COVID-19 ~25% more accurately than manual diagnosis.

References

[1] https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

[2] https://keras.io/examples/vision/grad_cam/

[3] https://github.com/keras-team/keras-io/blob/master/examples/vision/grad_cam.py