Report on Explainable AI Solutions for Classifying COVID 19 Pneumonia

Haoran Zhu
Department of Computer Science
Tandon School of Engineering
New York University
hz1922@nyu.edu

Abstract

Due to the spread of COVID 19, we are investigating existing interpretable AI tools and algorithms to quickly determine whether a person is a potential COVID-19 patient or not. Firstly, We reproduce the existing implementation of a neural network(1) which can predict the possibility of a person having COVID-19 or not. It uses Local Interpretable Model-Agnostic Explanations (i.e. LIME(5)) to explain the reason by giving potential COVID-19 features in chest X-ray pictures. Secondly, we summarize another algorithm called SHAP(4) which is a unified approach to interpret model predictions. Finally we try to integrate SHAP implementation into (1) and explain the reason why it currently has some integration problems.

1 Outline

In the following sections, we will firstly show the reproduced results of (1). Then we'll summarize the main idea of SHAP algorithm. Finally we show the engineering difficulties in integrating SHAP algorithm into (1)

2 Reproduced Results

We are using a linux server equipped with 16GB Nvidia Tesla T4 GPU on Google Cloud. Due to limited computation resources and time, we only train DCNN and ResNet50v2 in this report.

Following the guidance provided by (1), just make some slight justification of different paths for different datasets in config,yml, we can successfully set up the environment of this project.

2.1 DCNN Results

In this setting, we are using a customized deep convolutional neural network. We set epoch=10. After 22 minutes of training, we get a model with a testing accuracy of 91.08%.

The architecture is as the following:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv0_0 (Conv2D)	(None, 224, 224, 16)	448	input_1[0][0]
batch_normalization (BatchNorma	(None, 224, 224, 16)	64	conv0_0[0][0]
leaky_re_lu (LeakyReLU)	(None, 224, 224, 16)	0	batch_normalization[0][0]
conv0_1 (Conv2D)	(None, 224, 224, 16)	2320	leaky_re_lu[0][0]
concat0 (Concatenate)	(None, 224, 224, 19)	0	conv0_1[0][0] input_1[0][0]
batch_normalization_1 (BatchNor	(None, 224, 224, 19)	76	concat0[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 224, 224, 19)	0	batch_normalization_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 112, 112, 19)	0	leaky_re_lu_1[0][0]
conv1_0 (Conv2D)	(None, 112, 112, 48)	8256	max_pooling2d[0][0]
batch_normalization_2 (BatchNor	(None, 112, 112, 48)	192	conv1_0[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 112, 112, 48)	0	batch_normalization_2[0][0]
conv1_1 (Conv2D)	(None, 112, 112, 48)	20784	leaky_re_lu_2[0][0]
concat1 (Concatenate)	(None, 112, 112, 67)	0	conv1_1[0][0] max_pooling2d[0][0]
batch_normalization_3 (BatchNor	(None, 112, 112, 67)	268	concat1[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 112, 112, 67)	0	batch_normalization_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 67)	0	leaky_re_lu_3[0][0]
conv2_0 (Conv2D)	(None, 56, 56, 144)	86976	max_pooling2d_1[0][0]
batch_normalization_4 (BatchNor	(None, 56, 56, 144)	576	conv2_0[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 56, 56, 144)	0	batch_normalization_4[0][0]
conv2_1 (Conv2D)	(None, 56, 56, 144)	186768	leaky_re_lu_4[0][0]
concat2 (Concatenate)	(None, 56, 56, 211)	0	conv2_1[0][0] max_pooling2d_1[0][0]
batch_normalization_5 (BatchNor	(None, 56, 56, 211)	844	concat2[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 56, 56, 211)	0	batch_normalization_5[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 211)	0	leaky_re_lu_5[0][0]
flatten (Flatten)	(None, 165424)	0	max_pooling2d_2[0][0]
dropout (Dropout)	(None, 165424)	0	flatten[0][0]
dense (Dense)	(None, 128)	21174400	dropout[0][0]
leaky_re_lu_6 (LeakyReLU)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 2)	258	leaky_re_lu_6[0][0]
output (Activation)	(None, 2)	0	dense_1[0][0]

Figure 1: DCNN Achitecture

Accuracy and loss over epoch is as the following:

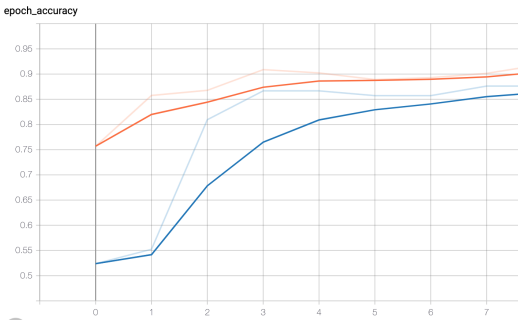

Figure 2: Accuracy over time by DCNN

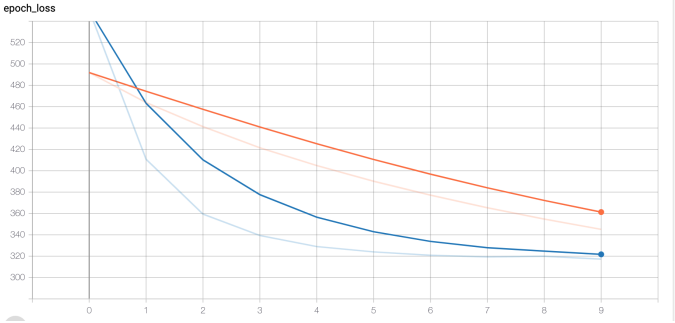

Figure 3: Loss over time by DCNN

The prediction for a test X-ray image with explanation is as the following, the predict time is only 30 seconds:

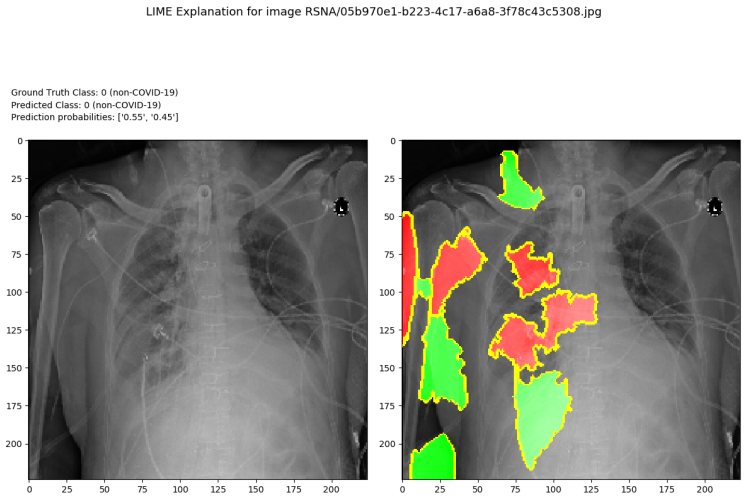

Figure 4: Prediction by DCNN

2.2 ResNet50v2 Results

In this setting, we are using ResNet50v2. We set epoch=10. After 36 minutes of training, we get a model with a testing accuracy of 93.5%.

The architecture is the same as ResNet50 except the last layer is changed to fully connected layer of 2 neurons.

Accuracy and loss over epoch is as the following:

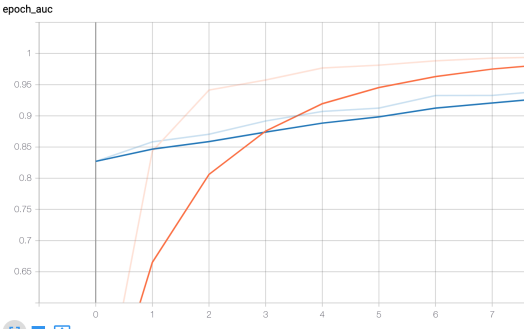

Figure 5: Accuracy over time by ResNet50v2

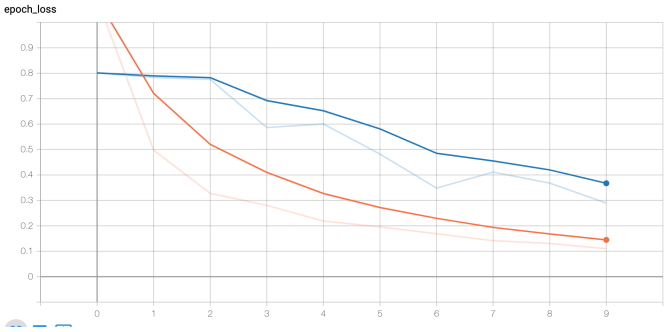

Figure 6: Loss over time by ResNet50v2

The prediction for a test X-ray image with explanation is as the following, the predict time is only 30 seconds:

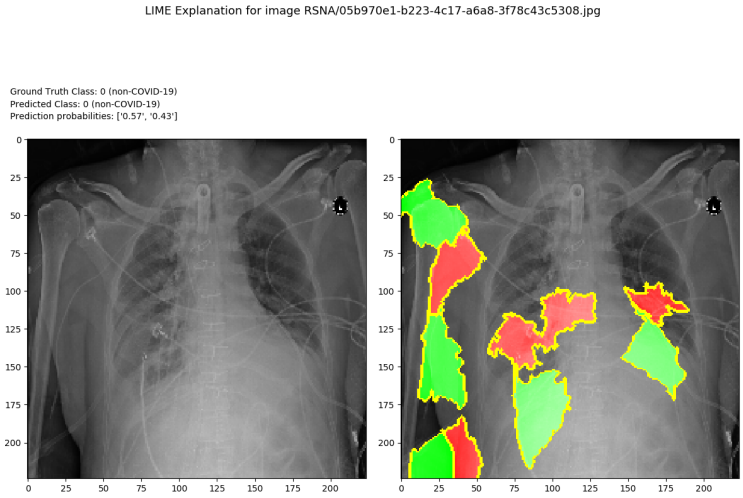

Figure 7: Prediction by Resnet50v2

3 SHAP Algorithm

3.1 Algorithm Ideas

For simple models the best explanation is itself because it can perfectly represent itself and is easy to understand. However for complex models like deep neural networks, we can not use the model itself to explain its decision because the model is too complex to understand. So we need to have a *explanation model*, which is an interpretable approximation of the original model.

Definition 1 Additive feature attribution methods have an explanation model that is a linear function of binary variables:

$$g\left(z'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{1}$$

where $z' \in \{0,1\}^M$, it is a simplified input which can be mapped to the original inputs x through a mapping function. M is the number of simplified input features and $\phi_i \in \mathbb{R}$

In the above definition, function g attributes an effect ϕ_i to each feature, and sum the effects of all feature attributions approximate the output of the original model.

There are already different methods prosed to interpret the decision made by different models. The most famous ones are LIME(5), DeepLIFT(6), Layer-Wise Relevance Propagation(3). In SHAP's view, the above methods can all be united and match the above additive feature attribution equation.

Overall SHAP can unify the above existing algorithms and can consider the following properties that the above methods are missing:

3.2 Three properties

Property 1 (Local accuracy)

$$f(x) = g\left(x'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i'$$

The explanation model $g\left(x'\right)$ matches the original model $f(x)$ when $x = h_x\left(x'\right)$. It requires the explanation of the model match the original output.

Property 2 (Missingness)

$$x_i' = 0 \Longrightarrow \phi_i = 0$$

Missingness constrains features where $x_i' = 0$ to have no attributed impact. It requires features missing in the oringal input do not have impact in explanation model

Property 3 (Consistency)

Let $f_x\left(z'\right) = f\left(h_x\left(z'\right)\right)$ and $z'\backslash i$ denote setting $z_i' = 0$. For any two models f and f', if $f_x'\left(z'\right) - f_x'\left(z'\backslash i\right) \geq f_x\left(z'\right) - f_x\left(z'\backslash i\right)$ for all inputs $z' \in \{0,1\}^M$, then $\phi_i\left(f', x\right) \geq \phi_i(f, x)$ It requires that if a model changes some simplified input's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease.

3.3 SHAP Value

To get a unified explanation model, we propose SHAP values to measure feature importance. These are the Shapley values in game theory of a conditional expectation function of the original model.

The exact value is difficult to compute. However we can approximate them by using different methods, e.g. **Kernel SHAP, Linear SHAP, Low-order SHAP, Max SHAP, Deep SHAP** in (4) in different model settings to approximate the value.

3.4 Performance

In the original paper, the author has compared SHAP with existing algorithms e.g. LIME and real human's performance on feature explanation. And we find that the performance of SHAP is very similar to real human and far more accurate than other existing algorithms.

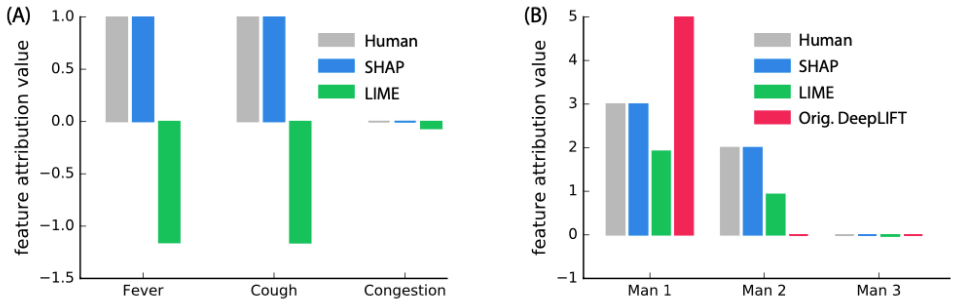

Figure 8: Performance of SHAP compared to human and other algorithms

Below is another example shows how SHAP explain the output of a convolutional network trained on the MNIST dataset. (A) Red areas increase the probability of that class, and blue areas decreas the probability of that class. The masked results remove pixels in order to go from 8 to 3. In (B) the change in log odds when masking over 20 random images supports the use of better estimates of SHAP values.

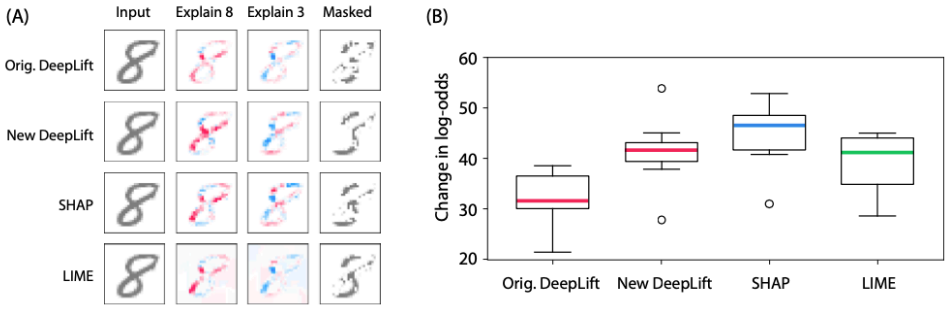

Figure 9: Performance of SHAP compared to human and other algorithms

By doing the above comparison, we can see that SHAP is a better algorithm to interpret model decisions. The above figure shows how it can explain an output in a CNN layer which shows that it can also explain the intermediate layers' output in models like e.g. ResNet, VGG. And we can try to apply SHAP algorithm in COVID 19 X-ray image classification task.

4 Integration Problems

We try to integrate SHAP algorithm in (1), we find there is also an existing implementation of SHAP algorithm (2). However when we try to integrate, tensoflow's version incompatibility issues appear, which means currently we can not integrate the two existing implementation together and need further research to fix the problem.

We try the demo code provided by (2):

```
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
import keras.backend as K
import numpy as np
import json
import shap

# load pre-trained model and choose two images to explain
model = VGG16(weights='imagenet', include_top=True)
X,y = shap.datasets.imagenet50()
to_explain = X[[39,41]]

# load the ImageNet class names
url = "https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json"
fname = shap.datasets.cache(url)
with open(fname) as f:
    class_names = json.load(f)

# explain how the input to the 7th layer of the model explains the top two classes
def map2layer(x, layer):
    feed_dict = dict(zip([model.layers[0].input], [preprocess_input(x.copy())]))
    return K.get_session().run(model.layers[layer].input, feed_dict)
e = shap.GradientExplainer(
    (model.layers[7].input, model.layers[-1].output),
    map2layer(X, 7),
    local_smoothing=0 # std dev of smoothing noise
)
shap_values,indexes = e.shap_values(map2layer(to_explain, 7), ranked_outputs=2)

# get the names for the classes
index_names = np.vectorize(lambda x: class_names[str(x)][1])(indexes)

# plot the explanations
shap.image_plot(shap_values, to_explain, index_names)
```

Figure 10: SHAP demo code to explain layer 7 of VGG model

And we change the code provided by(1) and plug the above demo, change VGG model to the model we are using.

```
# Get i'th preprocessed image in test set
lime_dict['TEST_GENERATOR'].reset()
for i in range(idx + 1):
    x, y = lime_dict['TEST_GENERATOR'].next()
x = np.squeeze(x, axis=0)

# Get the corresponding original image (no preprocessing)
orig_img = cv2.imread(lime_dict['RAW_DATA_PATH'] + lime_dict['TEST_SET']['filename'][idx])
new_dim = tuple(lime_dict['IMG_DIM'])
orig_img = cv2.resize(orig_img, new_dim, interpolation=cv2.INTER_NEAREST)     # Resize image

model = lime_dict['MODEL']
# explain how the input to the 7th layer of the model explains the top two classes
def map2layer(x, layer):
    feed_dict = dict(zip([model.layers[0].input], [preprocess_input(x.copy())]))
    return K.get_session().run(model.layers[layer].input, feed_dict)
e = shap.GradientExplainer(
    (model.layers[7].input, model.layers[-1].output),
    map2layer(x, 7),
    local_smoothing=0 # std dev of smoothing noise
)

shap_values,indexes = e.shap_values(map2layer(to_explain, 7), ranked_outputs=2)

# get the names for the classes
index_names = np.vectorize(lambda x: class_names[str(x)][1])(indexes)

# plot the explanations
shap.image_plot(shap_values, to_explain, index_names)
```

Figure 11: Integrate SHAP into current project

However when we run the code, the following error apears:

```
Traceback (most recent call last):
  File "/home/Haoran/covid-cxr/src/interpretability/shap_explain.py", line 135, in <module>
    explain_xray_shap(lime_dict, i, save_exp=True)                  # Generate explanation for image
  File "/home/Haoran/covid-cxr/src/interpretability/shap_explain.py", line 89, in explain_xray_shap
    map2layer(x, 7),
  File "/home/Haoran/covid-cxr/src/interpretability/shap_explain.py", line 85, in map2layer
    feed_dict = dict(zip([model.layers[0].input], [preprocess_input(x.copy())]))
  File "/usr/local/lib/python3.5/dist-packages/tensorflow_core/python/framework/ops.py", line 713, in __hash__
    raise TypeError("Tensor is unhashable if Tensor equality is enabled. "
TypeError: Tensor is unhashable if Tensor equality is enabled. Instead, use tensor.experimental_ref() as the key.
root@instance-1:/home/Haoran/covid-cxr# 
```

Figure 12

This is a known issue which hasn't been solved. (https://github.com/tensorflow/probability/issues/540)

(2) is using a class with 2 dimensions while the (1) model inputs are 4 dimensions, so we can not integrate the two projects directly. The issue is caused by incompatibility of Tensorflow version used in(1) and (2). (1) requires Tensorflow version 2 while (2) requires Tensorflow version 1. Either we choose to upgrade or we choose to downgrade Tensorflow version will cause an issue. This issue currently hasn't been solved.

Experiments by trying other SHAP algorithm implementation face the similar issues. Another solution is to implement the SHAP algorithm by ourselves, but due to the limitation of time, this solution to be realizes required some time, so the integration can't be done at present.

5 Conclusion

In this report, we try to reproduce results of existing algorithms to quickly classify COVID-19 potential patients. We then summarize a unified approach, SHAP algorithm to interpret model predictions. Finally we try to integrate SHAP into the previous project and explain the current engineering problems we are facing. We hope by applying AI technologies, we can hopefully help relieve the current situation of COVID-19.

References

[1] Covid-19 chest x-ray model. *https://github.com/aildnont/covid-cxr*.

[2] shap. *https://github.com/slundberg/shap*.

[3] BACH, S., BINDER, A., , GRÉGOIRE MONTAVON, F., KLAUSCHEN, K.-R. M., AND SAMEK, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One* (2015).

[4] LUNDBERG, S., AND LEE, S.-I. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* (2017).

[5] RIBEIRO, M. T., SINGH, S., AND GUESTRIN, C. "why should i trust you?": Explaining the predictions of any classifier. *the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[6] SHRIKUMAR, A., GREENSIDE, P., AND KUNDAJE, A. Learning important features through propagating activation differences. *PMLR* (2017).