IBM MACHINE LEARNING SPECIALIZATION – DEEP LEARNING AND REINFORCEMENT LEARNING PROJECT REPORT - COVID PREDICTION

March, 2022
MUSTAFA TAYYIP BAYRAM
Data Source: https://www.kaggle.com/maedemaftouni/large-covid19-ct-slice-dataset/code

Objective

The COVID-19 epidemic has pushed healthcare systems in every country to their breaking point, since a vast number of people have lost. Early identification of COVID-19 in a faster, easier, and less expensive manner can save lives and relieve the strain on healthcare providers. By using image processing techniques to X-ray pictures, artificial intelligence can play a large role in finding COVID-19.

We can utilize medical X-rays to assess the condition of a patient's lung since COVID-19 affects the epithelial cells that cover our breathing. As a result, the goal of this project is to provide an end-to-end solution that processes patients' X-rays (using a convolutional neural network algorithm) and predicts whether they are COVID-19 positive or normal, as well as the probability of each. The use case that can be developed from this study is the prioritization of patients based on the expected X-ray categorization (either normal or COVID-19 positive).

Data Set Contents

The Large COVID-19 CT scan slice dataset from Kaggle is being used for the purposes of this study. It was created by curating data from seven public datasets and includes 7,593 COVID-19 photos from 466 patients, 6,893 normal images from 604 patients, and 2,618 CAP images from 60 patients. The CAP-CT pictures were deleted, and only the first two were taken into account.

Notebook Contents

Three DL models is being created to find the most suitable for the specific problem.

- A simple convolutional neural network
- The same architecture was used for the second model, but data augmentation techniques were introduced, to find out if it will help to better accuracy and predictions
- For the third model, transfer learning techniques were considered, by using a pretrained VGG-16 model (with data augmentation)

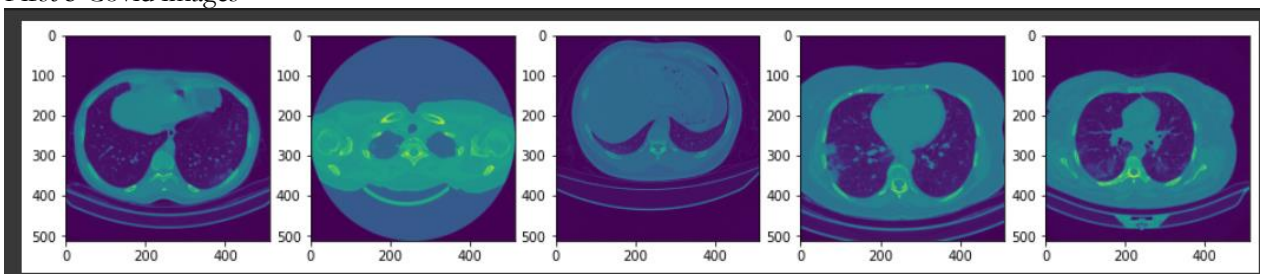
1. Importing Packages
2. Arranging sources and EDA
3. Model 1
4. Model 2
5. Model 3
6. Conclusion
7. Next Steps

Data augmentation is a technique for producing more training data from current training samples by applying a series of random changes to the samples to produce believable-looking images. The goal is for the model to

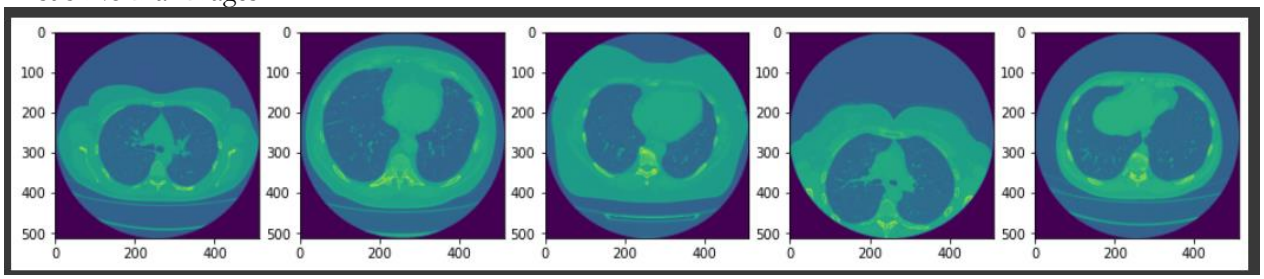
never see the same image twice during training. This allows the model to be exposed to more parts of the data and generalize more effectively.

```
Train covid images : 6075
Train normal images : 5514
Total training images : 11589

Test covid images : 760
Test normal images : 690
Total Test images : 1450

Validation covid images : 759
Validation normal images : 689
Total Validation images : 1448
```

First 5 Covid images

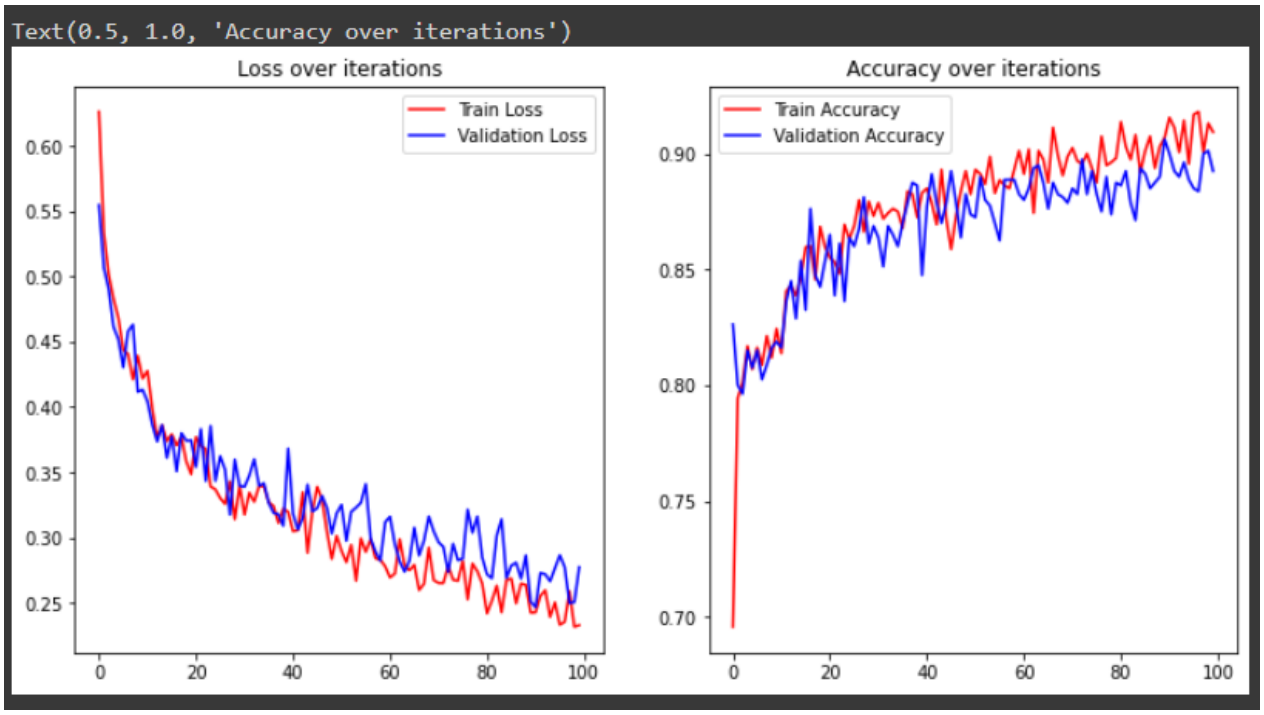


First 5 Normal images

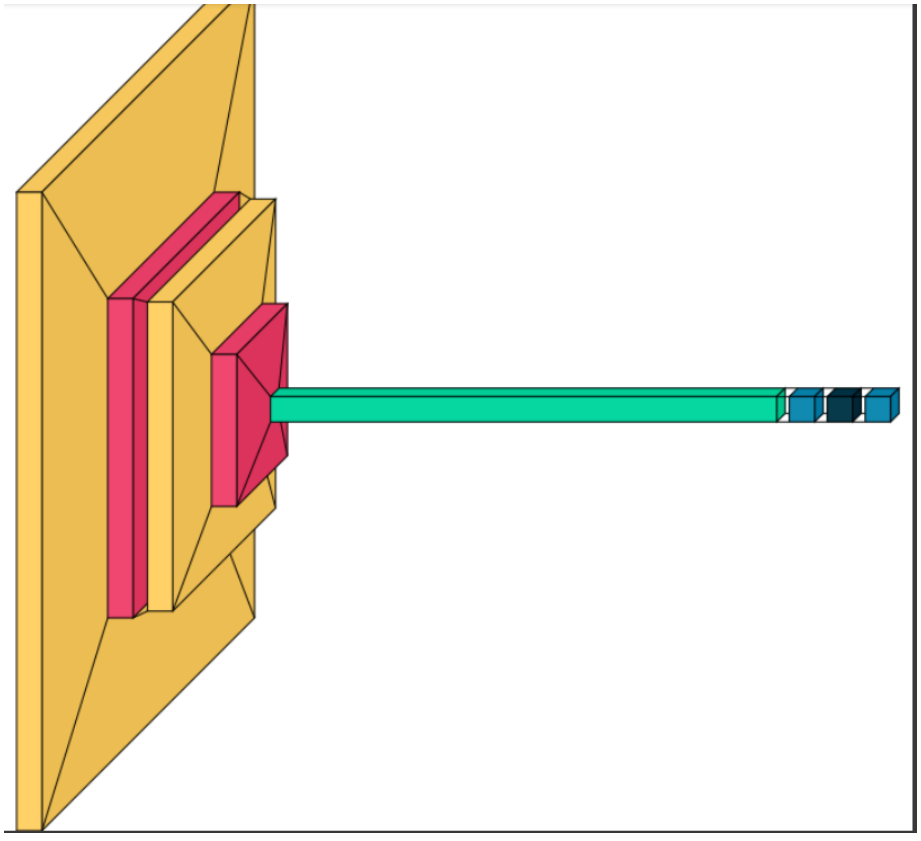


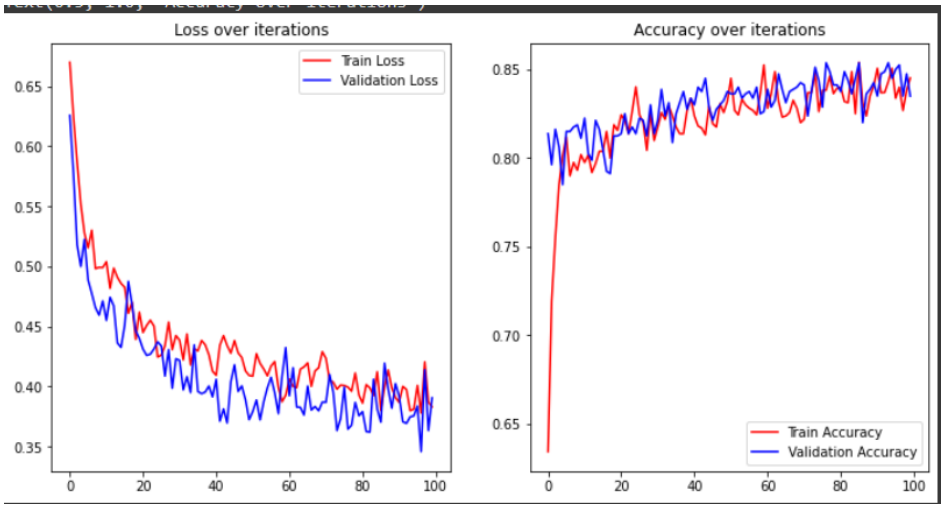
1st Model

```
lmodel = Sequential()

lmodel.add(Conv2D(128, (3,3), activation='relu', padding='valid' , # No padding
                  input_shape=(128,128, 1)))  # Creates convolution kernel
lmodel.add(MaxPooling2D(pool_size=(2,2))) # Downsample the input along its spatial dimensions

lmodel.add(Conv2D(64,(3, 3), activation='relu', padding='valid'))
lmodel.add(MaxPooling2D(pool_size=(2,2)))

lmodel.add(Flatten())   # Flattening involves transforming the entire pooled feature map
                        # Turns matrix into a vector

lmodel.add(Dense(32, activation='relu'))
lmodel.add(Dropout(0.5)) # For prevent overfitting - trains an ensemble of subnetweorks with shared parameters.

lmodel.add(Dense(1, activation='sigmoid'))
```


Text(0.5, 1.0, 'Accuracy over iterations')
Loss over iterations
Train Loss
Validation Loss
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
20
40
60
80
100
Accuracy over iterations
Train Accuracy
Validation Accuracy
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100

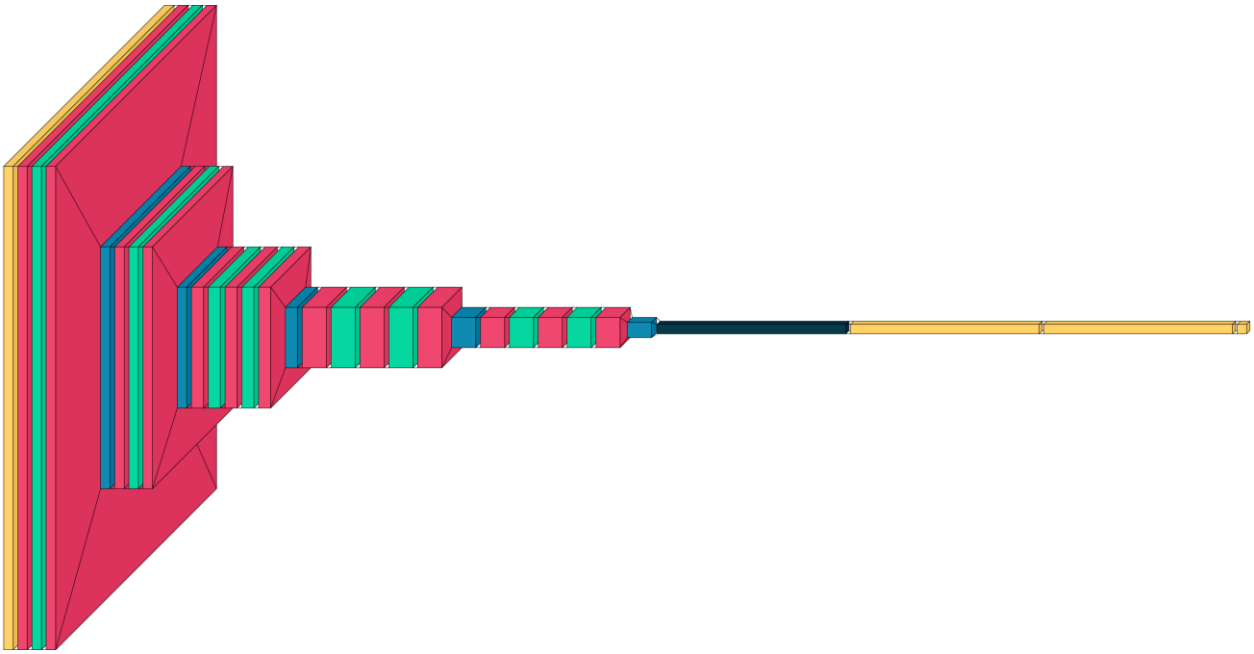
2nd Model

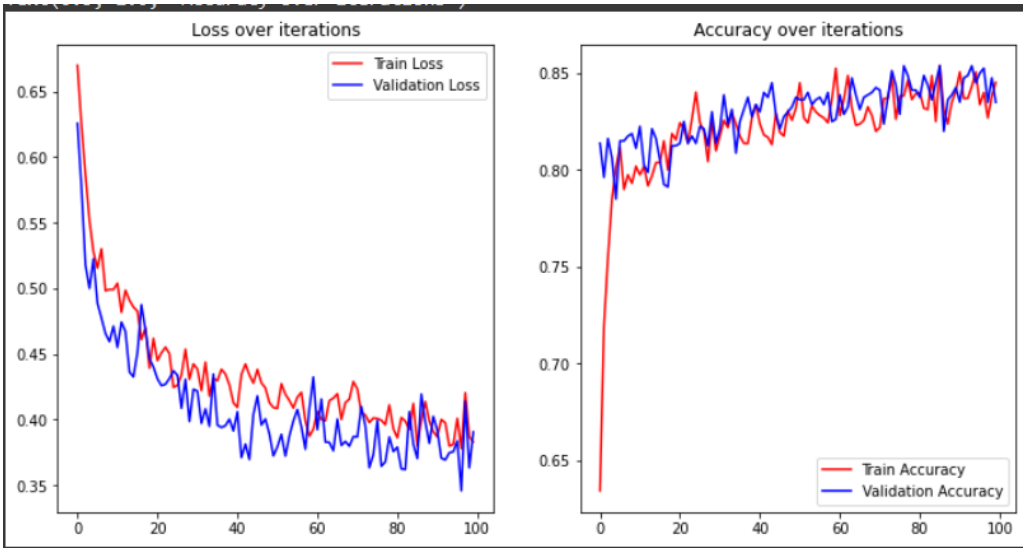




3rd Model

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 dense_15 (Dense)            (None, 4, 4, 256)         131328

 dropout_7 (Dropout)         (None, 4, 4, 256)         0

 flatten_4 (Flatten)         (None, 4096)              0

 dense_16 (Dense)            (None, 128)               524416

 dropout_8 (Dropout)         (None, 128)               0

 dense_17 (Dense)            (None, 1)                 129

=================================================================
Total params: 15,370,561
Trainable params: 655,873
Non-trainable params: 14,714,688
_________________________________________________________________
```




Loss over iterations
Train Loss
Validation Loss
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
20
40
60
80
100
Accuracy over iterations
0.85
0.80
0.75
0.70
0.65
Train Accuracy
Validation Accuracy
0
20
40
60
80
100

Accuracies and loses after 100 epochs:

	Accuracy	Val Accuracy	Loss	Val Loss
1st Model	0.9	0.91	0.25	0.255
2nd Model	0.8452	0.8350	0.3830	0.3905
3rd Model	0.88	0.875	0.28	0.30

	AUC Score
1st Model	0.96
2nd Model	0.917
3rd Model	0.96

	False Score
1st Model	141
2nd Model	256
3rd Model	159

Conclusion

A dataset of over 10.000 CT images was used to generate three convolutional neural network algorithms for accurately predicting whether or not an individual has developed COVID-19 for this research.

The AUC indicates how well the model distinguishes across classes. The greater the AUC, the better.

Recommendation: Despite its simplicity, the first model appears to produce the best results, with better accuracies and accurate predictions than the other two models.

Next Steps

The predictions heatmap, on the other hand, reveals that the first model misclassifies a large number of photos. It was put to the test on a collection of 1450 photos, and 10% of them were incorrectly classified. There are various things you may take to correct this, including:

• Switch to a new CNN model (add more hidden layers, tweak the hyperparameters etc.). The disadvantage would be that the training procedure would take much longer in the end.

• Using k-fold cross validation in conjunction with hyperparameter tuning approaches (such as kerastuner) determines the best number of layers for the situation at hand). The training procedure would need a significant amount of time.

• Using a different CNN architecture that has been pre-trained.

• Make the photos a different form. The photos were reduced to 128x128 pixels in this case.