

Investigation of Convolutional neural network on COVID-19 Xray Dataset

Bolong Tan

52513280

Instructor: Xiaoxiao Li

1. Problem statement and purpose

The rapid spread of the Covid 19 has a great impact on the lungs of patients. This project aims to use the convolutional neural network (CNN) to classify and detect Pneumonia to help doctors diagnose Pneumonia. In this project, CNN overall structure, dropout value, and learning rate to train the model are adjusted to improve CNN model's performance. CNN model's performance is evaluated based on its loss, accuracy and F1 score.

2. Dataset and framework selection

The dataset used in this project is downloaded from Kaggle and uploaded to the personal google drive. The dataset can be download from the link: https://drive.google.com/drive/folders/1eaGmsuTSFE685oGhHNI6fEmnt7JDYCq6?usp=share_link. (Please download the dataset and change the file path accordingly before running the code. Due to the high computation requirement, the Colab pro version may be needed.) The Dataset contains training dataset, validation dataset and test data set. In the training dataset, there are 74 images indicates normal lungs and 74 images indicates "pneumonia". In the validation dataset, there are 74 images indicates normal lungs and 74 images indicates "pneumonia". In the validation dataset, there are 20 images indicates normal lungs and 20 images indicates "pneumonia". The convolutional neural network is implemented using the Colab. Tensorflow is used as the machine learning framework because it is the most popular and provides excellent support for deep learning, especially convolutional neural networks, to process many images as input.

3. Convolutional neural network set up, train, validation and testing

3.1 Data Preprocess

First, all the necessary libraries and packages are imported. Then, training, validation and testing data are imported from google drive. The ImageDataGenerator() and imagegen.flow_from_directory() function is used to convert the image file to the datatype required by CNN.

Epochs is set to 13 and the EarlyStopping() function was used to stop the train process once the model performance stops improving on the validation dataset.

3.2 Construct CNN model and tune parameters

First version of CNN model:

The first version of the CNN model is constructed using the following code. The summary of the CNN model is shown in Figure 1. The batch size is set to 13.

```
#first model
model = Sequential()
image_shape = (1000,1000,3)
model.add(Conv2D(filters=6, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

Model: "sequential_5"		
Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 998, 998, 6)	168
max_pooling2d_5 (MaxPooling 2D)	(None, 499, 499, 6)	0
flatten_5 (Flatten)	(None, 1494006)	0
dense_6 (Dense)	(None, 1)	1494007
activation_5 (Activation)	(None, 1)	0
Total params: 1,494,175		
Trainable params: 1,494,175		
Non-trainable params: 0		

Figure 1

The training loss and the validation loss are shown in Figure 2. The training accuracy and the validation accuracy are shown in Figure 3. Even if the validation accuracy curve fluctuates slightly, the training loss and validation loss curves indicate the model is relatively good. After training and validation, the test dataset was passed to the trained CNN model. Besides, the training accuracy is 0.83 and the validation accuracy is 0.85 at the last epoch. F1 score is 0.91 and accuracy is 0.90 which is unexpected high. So, i run the model a few times. Every time the accuracy varies a lot. The lowest accuracy i observed is 0.65 and the highest accuracy observed is 0.97. It means this model is not stable and highly depends on the randomness in the CNN model. I added one dense and one dropout layer and initialised the learning rate in the next model to improve the accuracy to make it more stable.

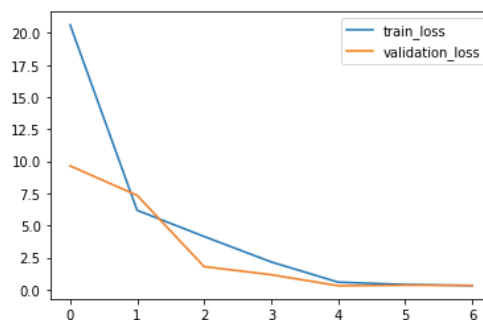


Figure 2

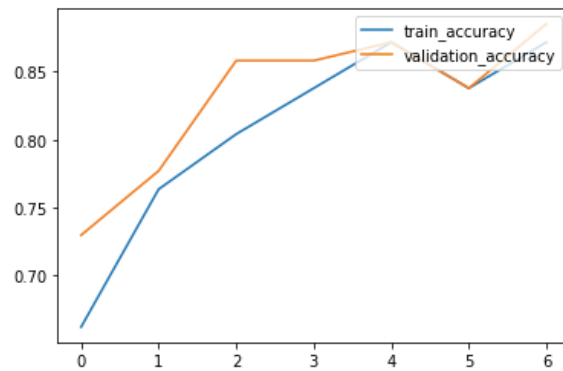


Figure 3

Second version of the CNN model:

The second version of the CNN model is constructed using the following code. The summary of the CNN model is shown in Figure 4. Batch size is set to 13.

```
#model12
model = Sequential()
image_shape = (1000,1000,3)
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(optimizer = Adam(learning_rate =
0.0001),loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()
```

Model: "sequential_15"		
Layer (type)	Output Shape	Param #
conv2d_35 (Conv2D)	(None, 998, 998, 32)	896
max_pooling2d_35 (MaxPooling2D)	(None, 499, 499, 32)	0
flatten_12 (Flatten)	(None, 7968032)	0
dense_24 (Dense)	(None, 32)	254977056
activation_24 (Activation)	(None, 32)	0
dropout_12 (Dropout)	(None, 32)	0
dense_25 (Dense)	(None, 1)	33
activation_25 (Activation)	(None, 1)	0
Total params: 254,977,985		
Trainable params: 254,977,985		
Non-trainable params: 0		

Figure 4

The training loss and the validation loss are shown in Figure 5. The training accuracy and the validation accuracy are shown in Figure 6. After training and validation, the test dataset was passed to the trained CNN model. The training accuracy is 0.8446 and the validation accuracy is 0.8514 at the last epoch. F1 score is 0.91 and accuracy is 0.92, better than the first version of the CNN model. I also run this model a few times and the accuracy is more stable than the first version of CNN model. The test accuracy is higher than the training accuracy. So, this model is not overfitting. In this way, I add one more convolution layer, maxpooling layer and adjust the dropout layer to the Second version of the CNN model to form the Third version of the CNN model.

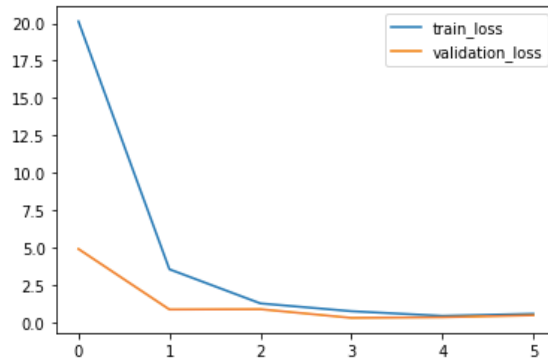


Figure 5

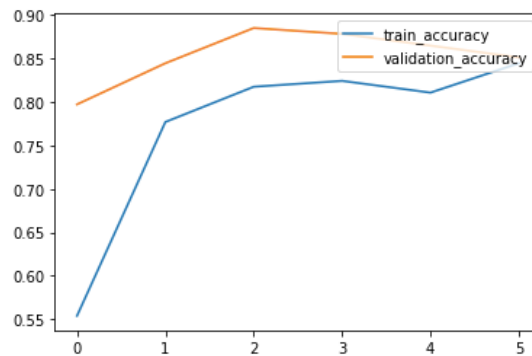


Figure 6

Third version of CNN model:

The third version of the CNN model is constructed using the following code. The summary of the CNN model is shown in Figure7. The batch size is set to 13.

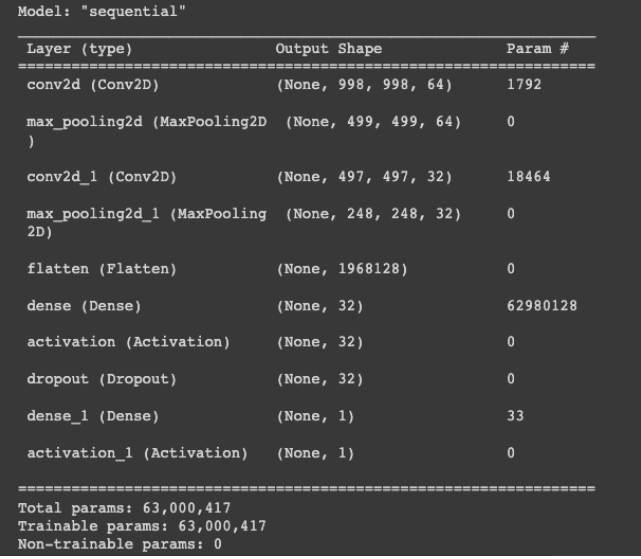
```
#model12
model = Sequential()
image_shape = (1000,1000,3)
model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```

model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(optimizer = Adam(learning_rate =
0.0001), loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()

```



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 998, 998, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 499, 499, 64)	0
conv2d_1 (Conv2D)	(None, 497, 497, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 248, 248, 32)	0
flatten (Flatten)	(None, 1968128)	0
dense (Dense)	(None, 32)	62980128
activation (Activation)	(None, 32)	0
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33
activation_1 (Activation)	(None, 1)	0
Total params: 63,000,417		
Trainable params: 63,000,417		
Non-trainable params: 0		

Figure 7

The training loss and the validation loss are shown in Figure 8. The training accuracy and the validation accuracy are shown in Figure 9. After training and validation, the test dataset was passed to the trained CNN model.

The training accuracy is 0.8311 and the validation accuracy is 0.8716 at the last epoch. F1 score is 0.92 and accuracy is 0.95, better than the second version of the CNN model. I also run this model a few times and the accuracy is fairly stable. Compared with "The second version of the CNN model", the total parameters are significantly decreased because one additional maxpooling layer was added after the convolution layer to reduce the dimensionality. Based on the result, having fewer total parameters could further improve the model's performance. In this way, I add one more convolution layer, a maxpooling layer and adjust the dropout layer to the Third version of the CNN model to form the Fourth version of the CNN model.

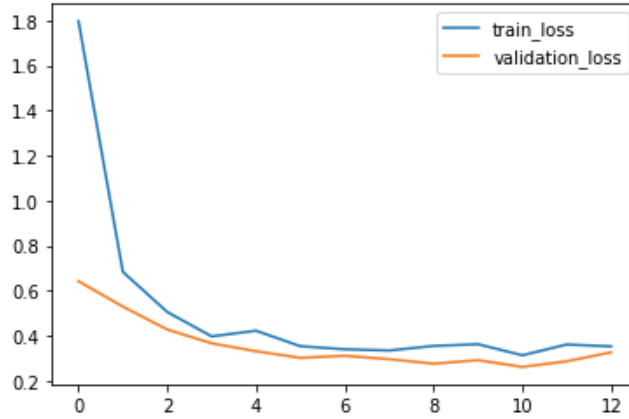


Figure 8

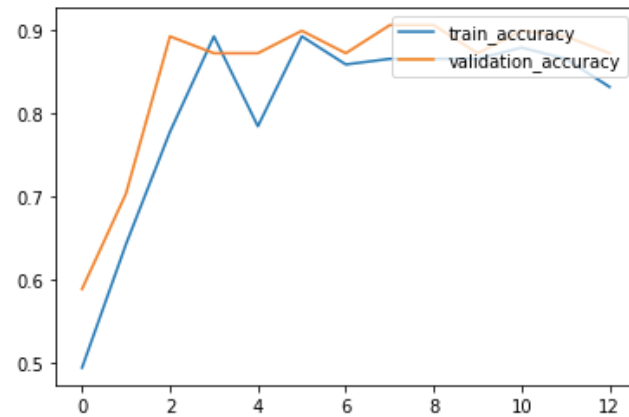


Figure 9

Fourth version of CNN model:

The Fourth version of the CNN model is constructed using the following code. The summary of the CNN model is shown in Figure 10. The batch size is set to 13.

```
model = Sequential()
image_shape = (1000,1000,3)
model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64))
model.add(Dense(32))
```

```

model.add(Activation('relu'))
model.add(Dropout(0.4))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()

```

```

Model: "sequential_1"

```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 998, 998, 64)	1792
max_pooling2d_2 (MaxPooling 2D)	(None, 499, 499, 64)	0
conv2d_3 (Conv2D)	(None, 497, 497, 32)	18464
max_pooling2d_3 (MaxPooling 2D)	(None, 248, 248, 32)	0
conv2d_4 (Conv2D)	(None, 246, 246, 64)	18496
max_pooling2d_4 (MaxPooling 2D)	(None, 123, 123, 64)	0
flatten_1 (Flatten)	(None, 968256)	0
dense_2 (Dense)	(None, 64)	61968448
dense_3 (Dense)	(None, 32)	2080
activation_2 (Activation)	(None, 32)	0
dropout_1 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 1)	33
activation_3 (Activation)	(None, 1)	0

```

=====
Total params: 62,009,313
Trainable params: 62,009,313
Non-trainable params: 0

```

Figure 10

The training loss and the validation loss are shown in Figure 11. The training accuracy and the validation accuracy are shown in Figure 12. Compared with “The third version of the CNN model”, the total parameters are slightly decreased because one additional maxpooling layer was added after the convolution layer to reduce the dimensionality.

After training and validation, the test dataset was passed to the trained CNN model. The training accuracy is 0.87 and the validation accuracy is 0.864 at the last epoch. F1 score is 0.98 and accuracy is 0.975, much better than the third version of the CNN model. I also run this model a few times and the accuracy stays in the range of 0.95 – 0.99. Its training loss(0.235) and validation loss(0.247) are also much lower than all previous models.

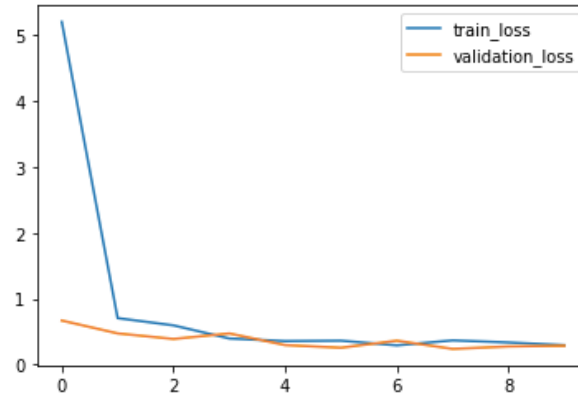


Figure 11

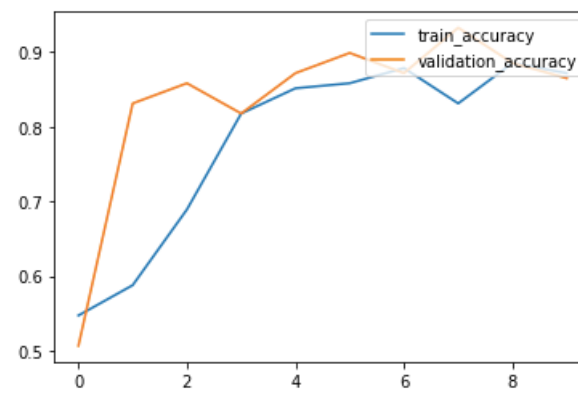


Figure 12

4. Conclusion

In this project, I mainly use the CNN to do binary classification on chest X-ray datasets. I gained some experience in training a model by tuning the hyperparameters. The Fourth version of the CNN model outstands in terms of accuracy, F1 score and stability. The project still not perfect but could be improved to get better results. One thing needed to be improved is the dataset splitting. It is better to make train:val:test = 7:2:1.

The CNN model's performance can also be furtherly improved if I can add more layers to the Fourth version of the CNN model. However, due to the long computing time and limited RAM of the Colab, it cannot be furtherly improved so far.

Reference

1. ahmedmaher22. (2021, December 29). *Covid-19 X-ray detection CNN*. Kaggle. Retrieved December 14, 2022, from <https://www.kaggle.com/code/ahmedmaher22/covid-19-x-ray-detection-cnn>

Appendix

In this project some ideas from <https://www.kaggle.com/code/ahmedmaher22/covid-19-x-ray-detection-cnn> (Reference 1) was borrowed.