Hand-on Experience for Training and Testing a binary Image Classifier (K=2) on chest X-ray image dataset using CNN

Siying Cen

Summary:

In this programming assignment, I performed a binary CNN classifier on two datasets of patients' X-ray images. The classification results help differentiate pneumonia X-ray images from normal X-ray images, which is useful for covid-19 clinical diagnosis.

Deliverable:

Tensorflow CNN model source code:
https://www.tensorflow.org/tutorials/images/cnn
I adjust some parameters when building model.

Dataset was downloaded from Kaggle "covid-19 Xray images using cnn",
URL: https://www.kaggle.com/akkinasrikar/covid19-xray-images-using-cnn?select=images
I generated two datasets of different sizes from this original dataset.

The software environment for developing the model was based on Tensorflow using python. The tensorflow environment was initialized in terminal. The code was edited and run in Jupyter Notebook.

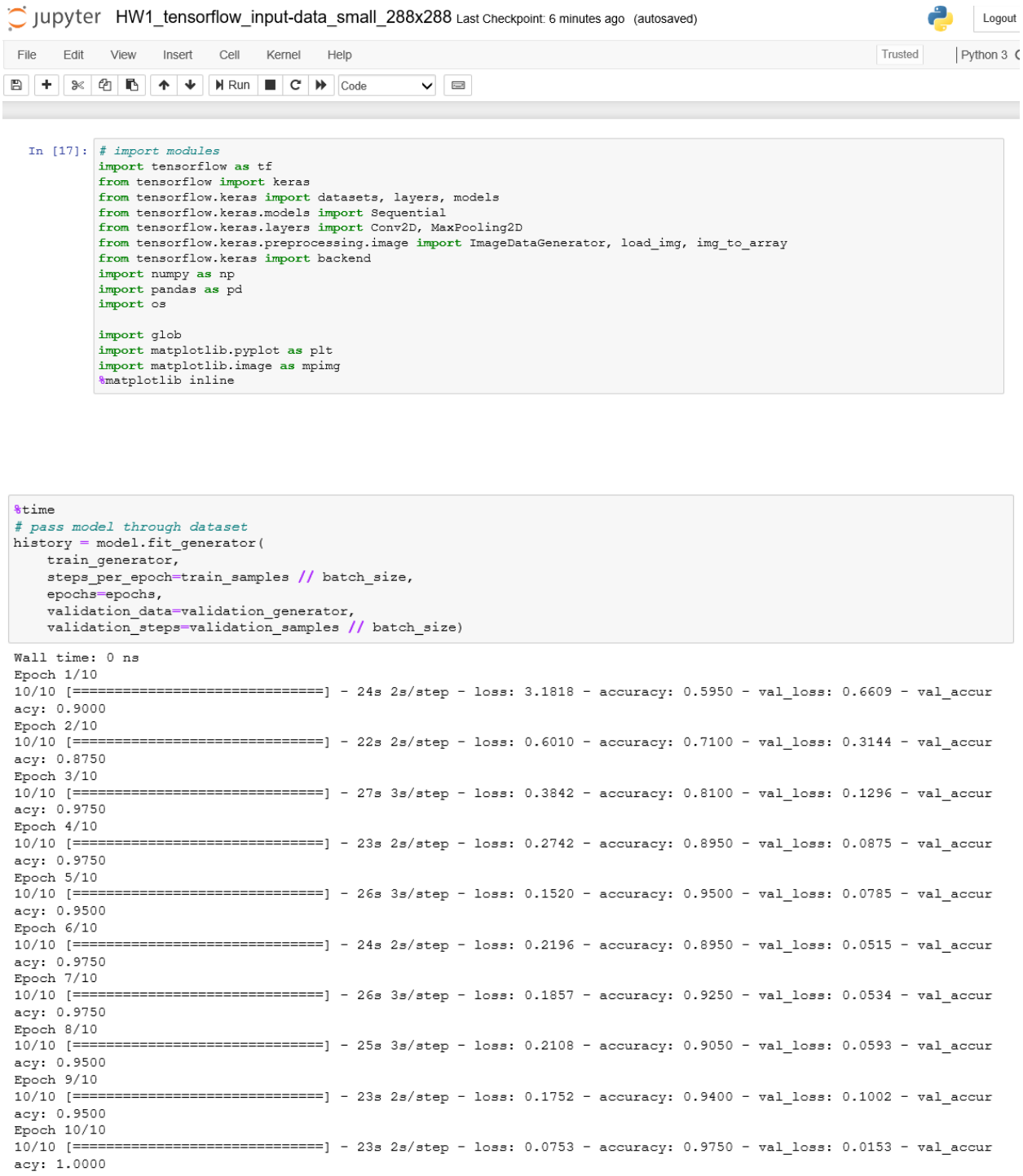

Figure 1. Screenshots of execution environments.

Input Analysis:

For input data, I created **2** datasets with different sizes (302 MB for big dataset and 133 MB for small dataset) from the original X-ray images dataset; when loading input data for training models, I chose **2 resolutions** (150x150 or 288x288) for each dataset.

The training/validation samples were split by ratio 8:2 in each dataset.

```
train_data_dir = 'small_input/train'
validation_data_dir = 'small_input/test'


train_samples = 200
validation_samples = 50
```

```
train_data_dir = 'large_input/train'
validation_data_dir = 'large_input/test'


train_samples = 800
validation_samples = 200
```

```
# input processing code come from Kaggle expert Lviv,

# dimensions of our images.
img_width, img_height = 150, 150
```

```
# input processing code come from Kaggle expert Lviv,

# dimensions of our images.
img_width, img_height = 288, 288
```

Figure 2. Screenshots for input sizes and dimensions(resolution) of 4 datasets.

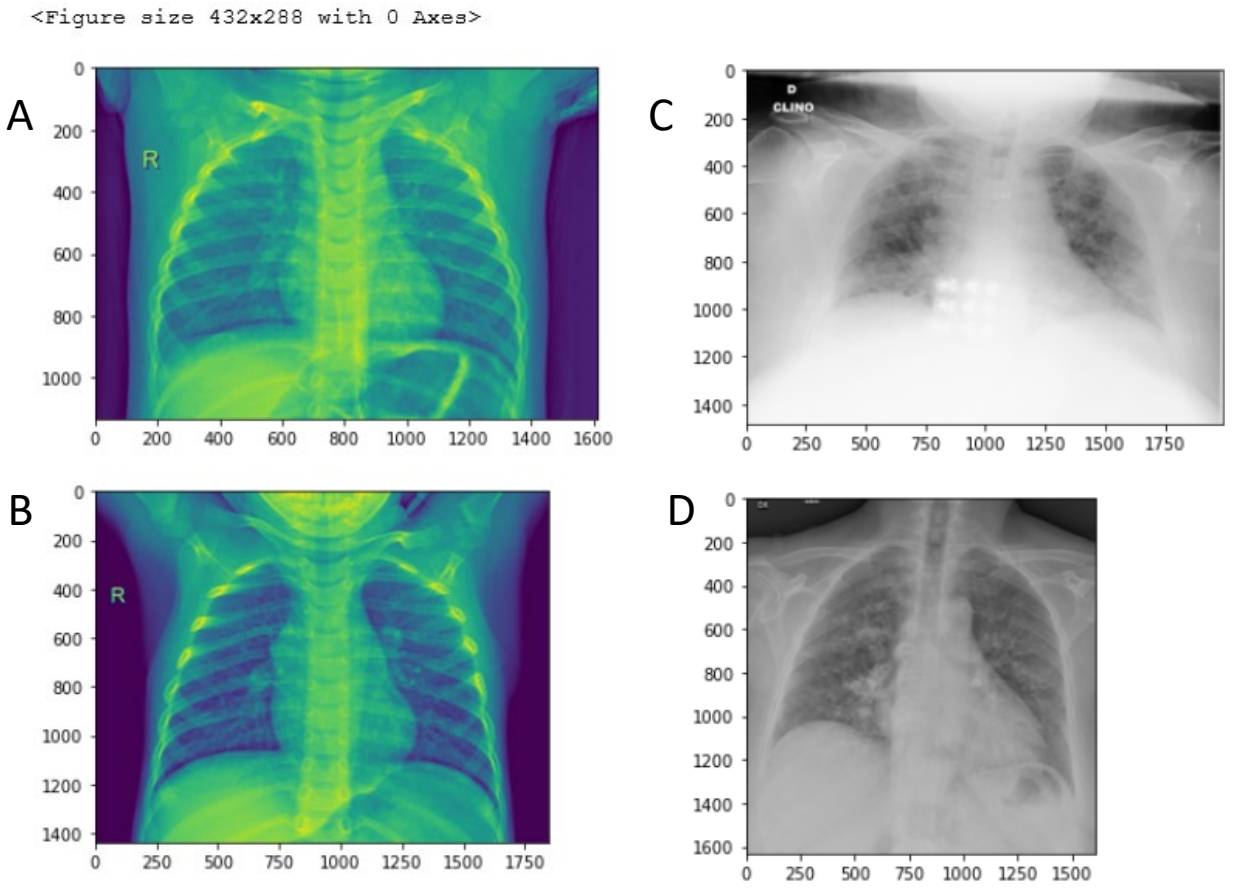

Figure 3. 2 samples of class normal (A, B) and class pneumonia (C, D).

Model Analysis:

```
# build CNN model
# code for building model come from 'https://www.tensorflow.org/tutorials/images/cnn', I adjust some parameters

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2))
```

Figure 4. Screenshot of model codes.

```
# compile model

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 5. Screenshot of model compile codes. The default learning rate of 'Adam' optimizer is 0.001.

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 286, 286, 32)      896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 143, 143, 32)      0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 141, 141, 32)      9248
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 70, 70, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 68, 68, 64)        18496
_________________________________________________________________
flatten_5 (Flatten)          (None, 295936)            0
_________________________________________________________________
dense_10 (Dense)             (None, 64)                18939968
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 130
=================================================================
Total params: 18,968,738
Trainable params: 18,968,738
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Structure of convolution neural network model used for 4 classifiers. The model consists of 3 convolutional layers and 2 pooling layers. Activation function 'relu' was used.

	conv 1	conv 2	conv 3	pool 1	pool 2
filter size	3x3	3x3	3x3		
filter number	32	32	64		
neuron size				2x2	2x2

Table 1. Neuron number and size of each layer in CNN model used by all 4 classifiers.

	#train_sample	#validation_sample	#epoch	batch_size	#train_steps per_epoch	#tvalidation_steps per_epoch
small_150x150	200	50	10	50	4	1
small_288x288	200	50	10	50	4	1
large_150x150	800	200	10	100	4	2
large_288x288	800	200	10	100	4	2

Table 2. Sample size and #iteration of 4 classifiers.

Output Analysis:
Report the performance comparison and analysis of my binary CNN classifier:

	traing time	taining accuracy	testing time	testing accuray	trained_model_size
small_150x150	95s	0.95	0s	0.98	112 MB
small_288x288	181s	0.94	0s	0.96	112 MB
large_150x150	27s	0.97	2s	0.755	268 MB
large_288x288	97s	0.95	5s	0.775	268 MB

Table 3. Training time, training accuracy, testing time and testing accuracy of 4 classifiers.

Outlier Test:

For outlier test, I downloaded “cats vs. dogs” dataset from Kaggle and extract 20 images from this dataset as outlier test dataset.

URL: https://www.kaggle.com/c/dogs-vs-cats?rvi=1

Because there are just 20 samples in outlier test dataset, the batch size used for training and testing were adjusted to 20.

outlier_test	test_accuracy	test_time
small_150x150	0.45	0s
large_150x150	0.6	0s

Table 4. Testing time and testing accuracy for outlier test of 2 classifiers.

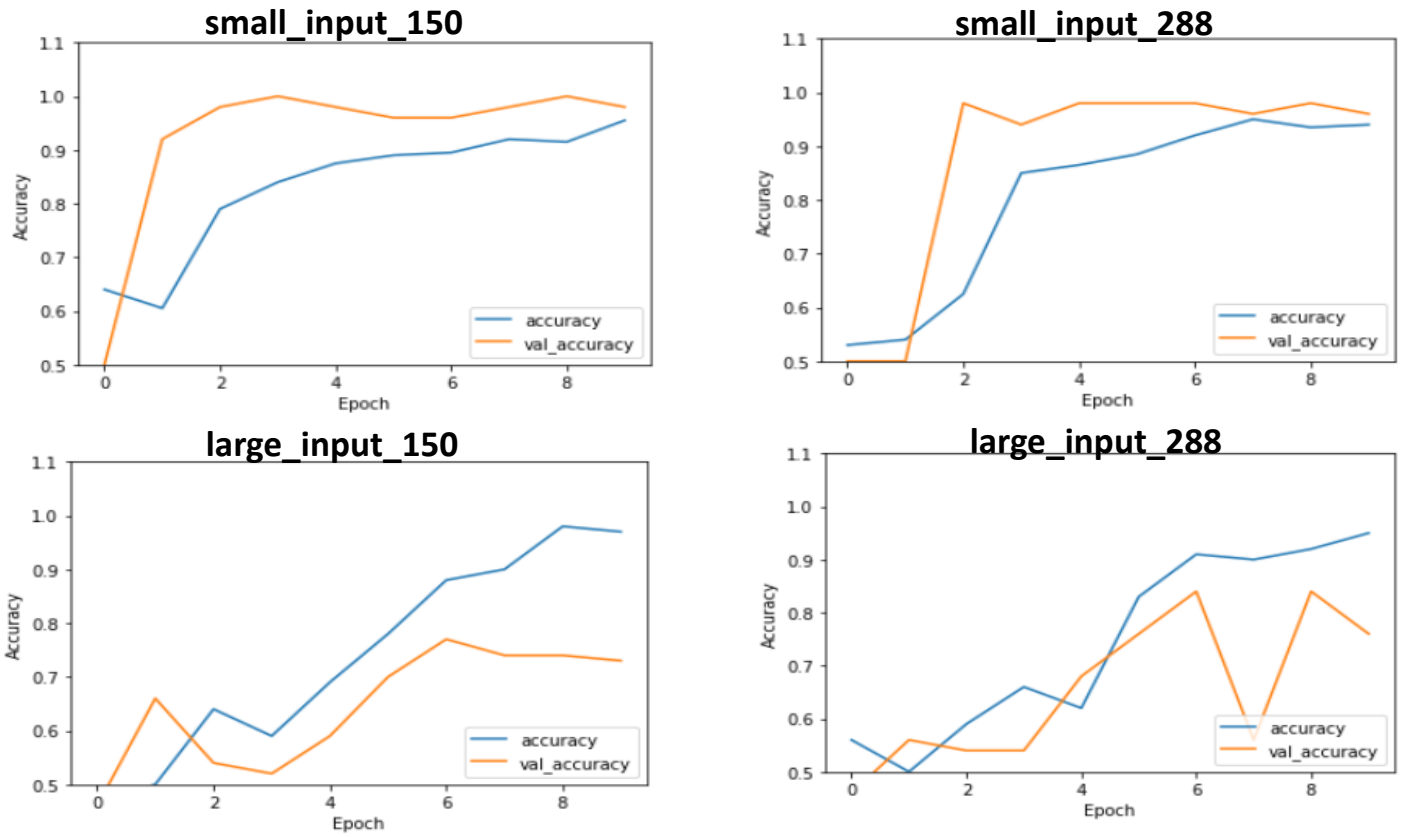

Figure 7. Training accuracy (blue line) and validation accuracy (orange line) for every classifier change along with every Epoch. Total number of epochs are 10.

Observations:

(1) Training time of classifiers with higher resolution dataset (288x288) cost more time than classifiers with lower resolution dataset (150x150). It is obvious that high resolution input images bring more information. Large dataset classifiers cost less time than small

dataset classifiers, perhaps due to the batch size chosen for large dataset classifiers is 100 while the batch size for small dataset is 50.

(2) From Table 3, it can be concluded that the small dataset classifiers have higher training accuracy as well as higher validation accuracy than large dataset classifiers using the hyperparameters summarized in Table 2.

(3) Figure 7 shows the tendency that training accuracy and testing accuracy increase with #epoch. Small dataset classifiers for validation dataset coverage after around 5 epochs, while large dataset classifiers not coverage after 10 epochs. As a result, large dataset classifiers may need more epochs for convergence.

(4) Outlier test shows that my classifiers are specific for X-ray image datasets because the testing accuracy for outlier test dataset is low.