Deep Learning in Biomedical Image I I0061622 林祐安

1.

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
assert x_train.shape == (60000, 28, 28)
assert x_test.shape == (10000, 28, 28)
assert y_train.shape == (60000,)
assert y_test.shape == (10000,)
class_names = range(10)
```

2.

```
def FCNN(layer_num=1):
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
    if layer_num:
        for _ in range(layer_num):
            model.add(tf.keras.layers.Dense(128, activation='relu'))
    model.add(tf.keras.layers.Dense(10))
    model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
    model.summary()
    return model
```

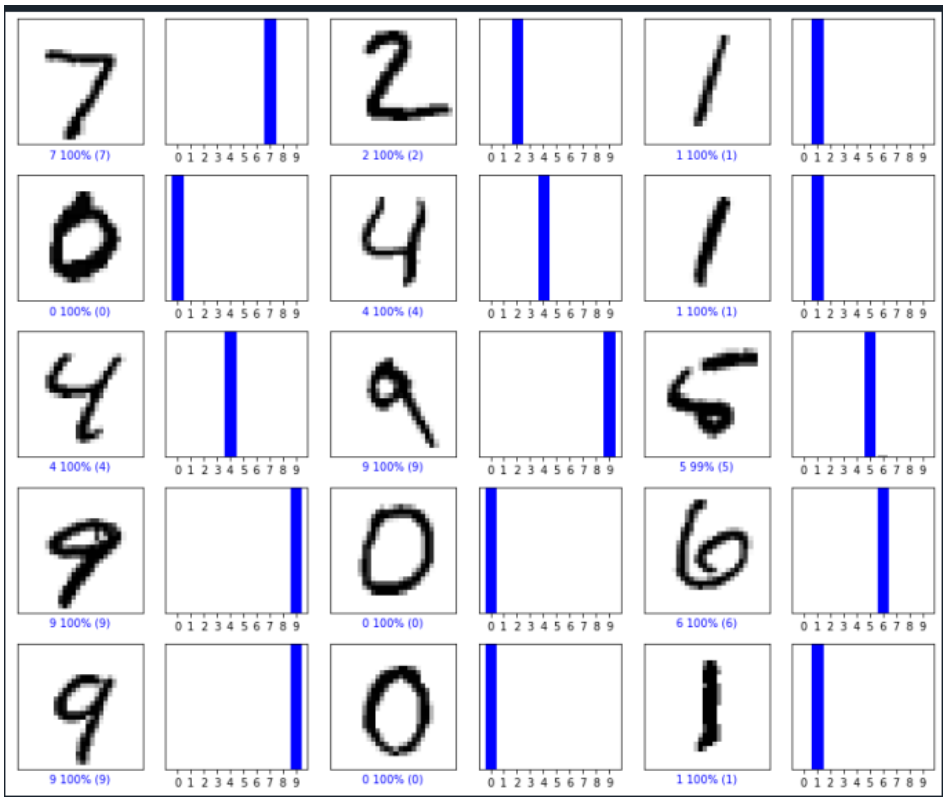
Function FCNN returns model which has one , two or three hidden layers determined by the input argument ‘layer_num’. In details, optimizer is Adam, a common optimizer used in any model. It includes the pros of momentum and adaptive learning rate. For loss function, since we are doing a classification job, cross-entropy is most suitable. Especially, the output of sparse categorical cross-entropy has been one-hot encoder. We can easily calculate the distance or loss between target and prediction.

# of hidden layers	Training accuracy (loss)	Testing accuracy
1	0.9779 (0.0812)	0.9779
2	0.9780 (0.0958)	0.9780
3	0.9798 (0.0846)	0.9798

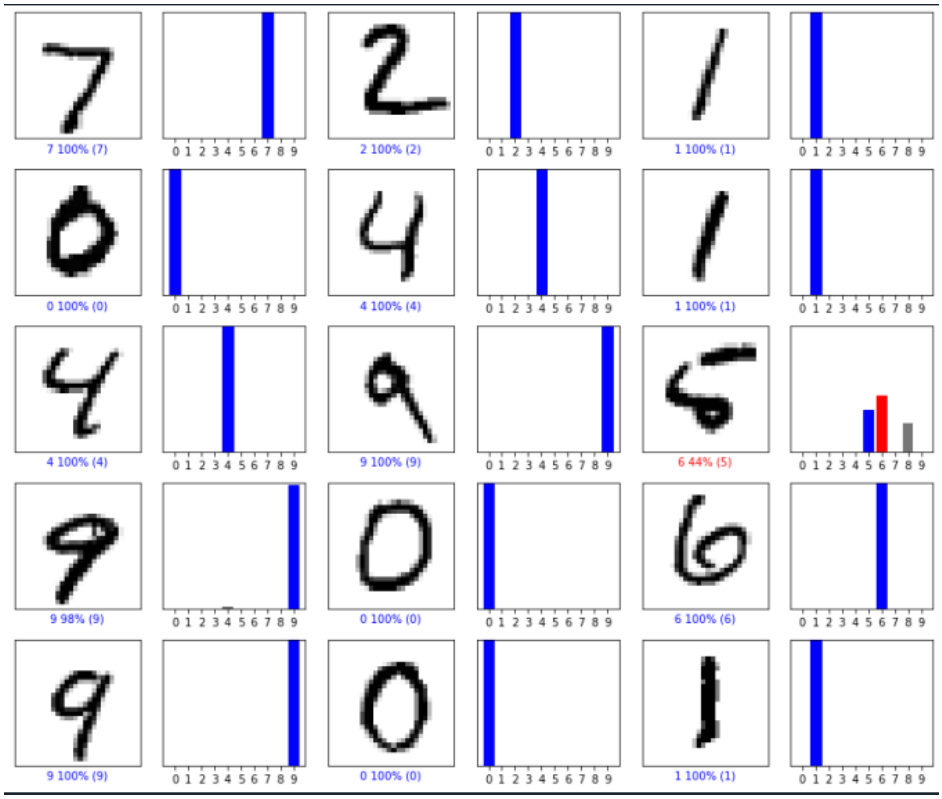
According to above table, we observe that

1. accuracy is Not proportional to declining loss
2. more hidden layer would have better performance

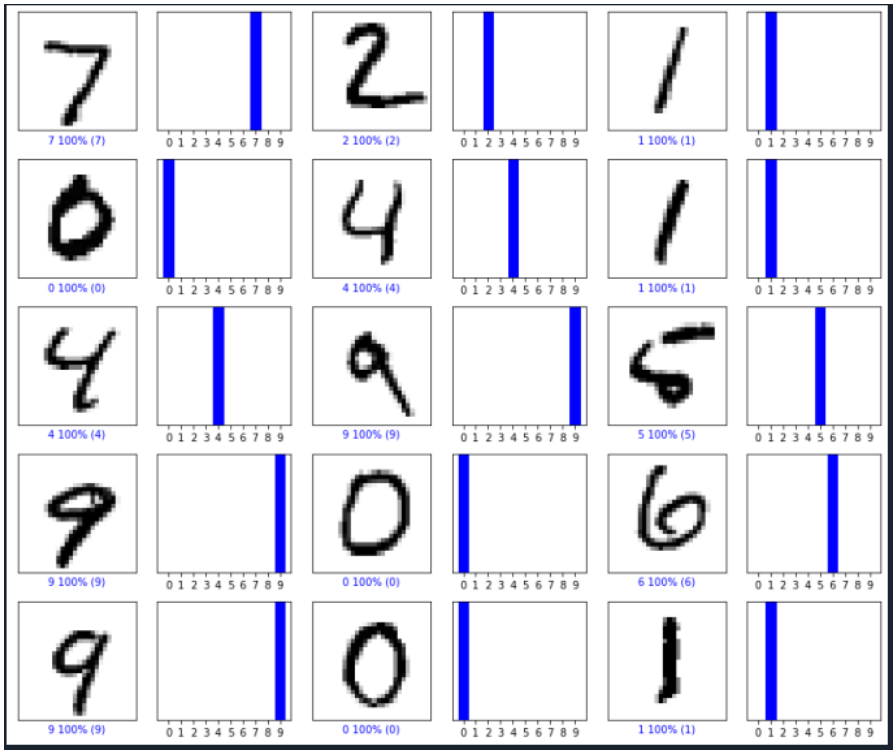
one-hidden layer:



two-hidden layer:

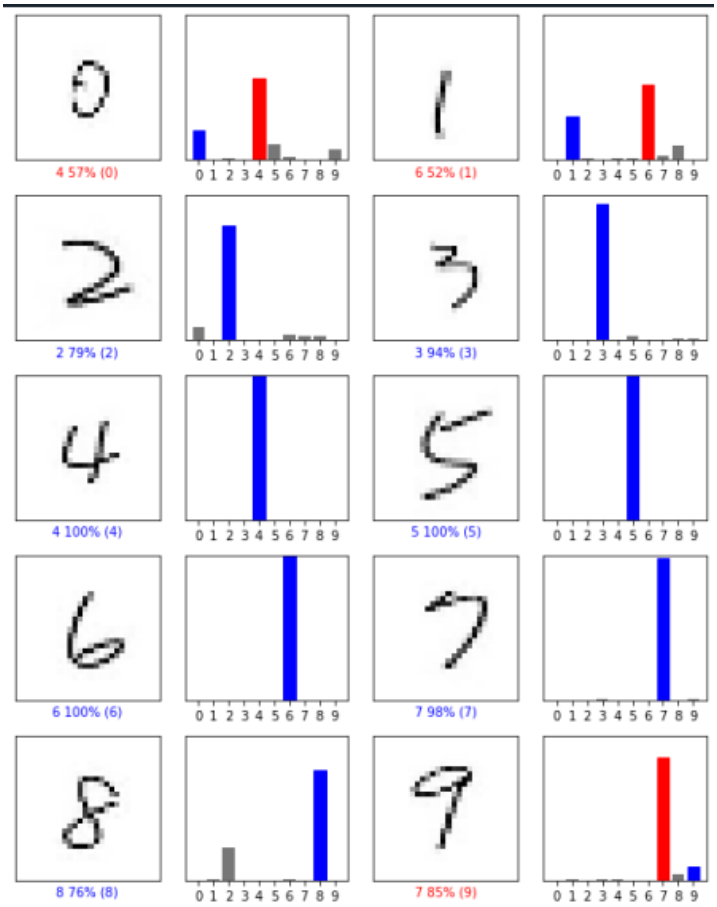


three-hidden layer

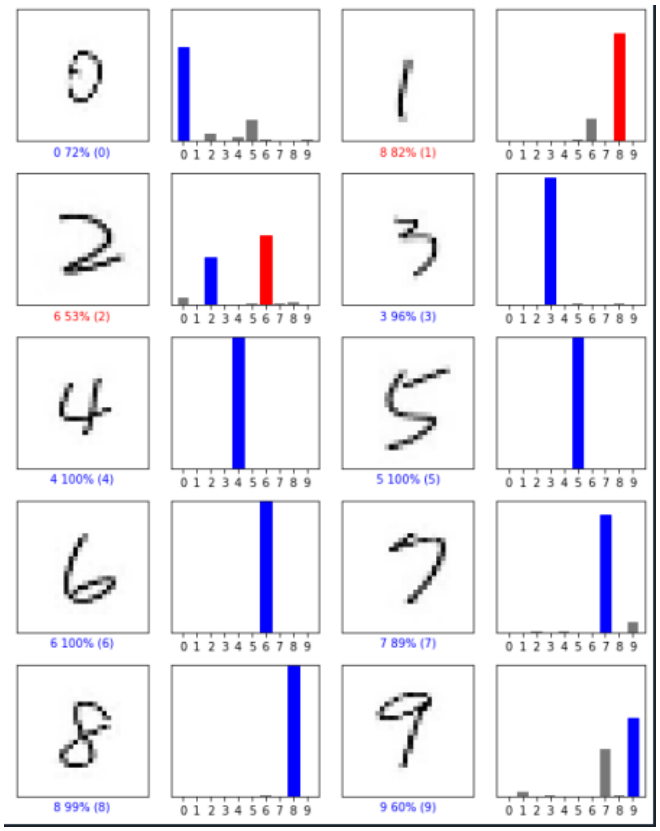


3.

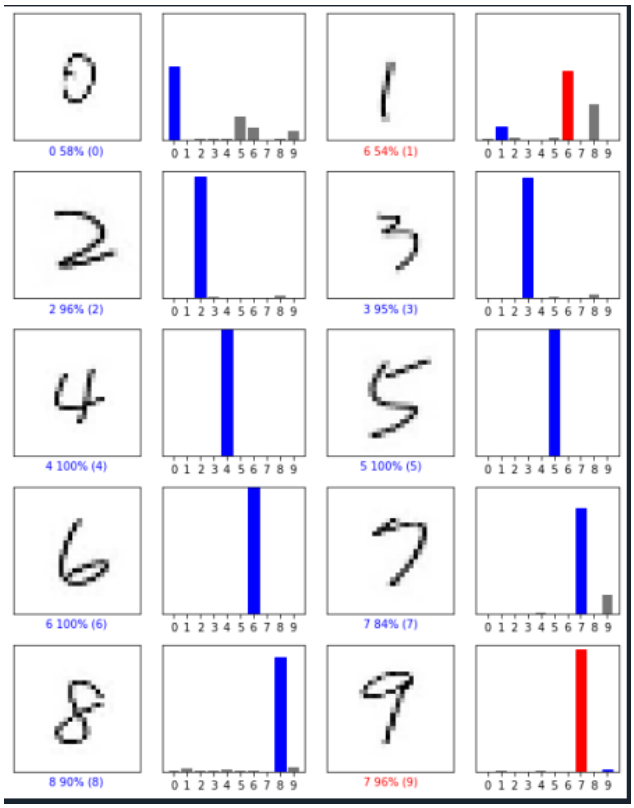
One-hidden layer:



two-hidden layer:



three-hidden layer:



Seen in the above three figure I drew. They almost have equivalent performance. Our digit is skinny and regular, but it seems not to be helpful. Although our model only has dense layer which isn't too complicated, I don't expect it can work robustly. Maybe convolution layer or deeper network would help us to improve its performance.