Algorithms – Assignment 4

Structing Environment for Deep Learning in Jupyter notebook

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

CNN Models

```
: def select_model(model_number):
    if model_number == 1:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),  # la
yer 1
                    keras.layers.MaxPool2D((2,2)),                                                   # la
yer 2
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                 # la
yer 3

    if model_number == 2:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape=(28,28,1)),      # la
yer 1
                    keras.layers.MaxPool2D((2,2)),                                                   # la
yer 2
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                             # la
yer 3
                    keras.layers.MaxPool2D((2,2)),                                                   # la
yer 4
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                 # la
yer 5

    if model_number == 3:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),  # la
yer 1
                    keras.layers.MaxPool2D((2,2)),                                                   # la
yer 2
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                             # la
yer 3
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),                             # la
yer 4
                    keras.layers.MaxPool2D((2,2)),                                                   # la
yer 5
                    keras.layers.Conv2D(128, (3,3), activation = 'relu'),                            # la
yer 6
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])                                 # la
yer 7

    return model
```

MODEL 1

1. Training

```
model = select_model(1)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 5408)              0
_________________________________________________________________
dense (Dense)                (None, 10)                54090
=================================================================
Total params: 54,410
Trainable params: 54,410
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

model.fit(train_images, train_labels,  epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 12s 199us/sample - loss: 0.6118 - accuracy: 0.9396
Epoch 2/5
60000/60000 [==============================] - 13s 214us/sample - loss: 0.0843 - accuracy: 0.9755
Epoch 3/5
60000/60000 [==============================] - 11s 190us/sample - loss: 0.0729 - accuracy: 0.9777
Epoch 4/5
60000/60000 [==============================] - 14s 229us/sample - loss: 0.0645 - accuracy: 0.9806
Epoch 5/5
60000/60000 [==============================] - 15s 253us/sample - loss: 0.0552 - accuracy: 0.9839
```

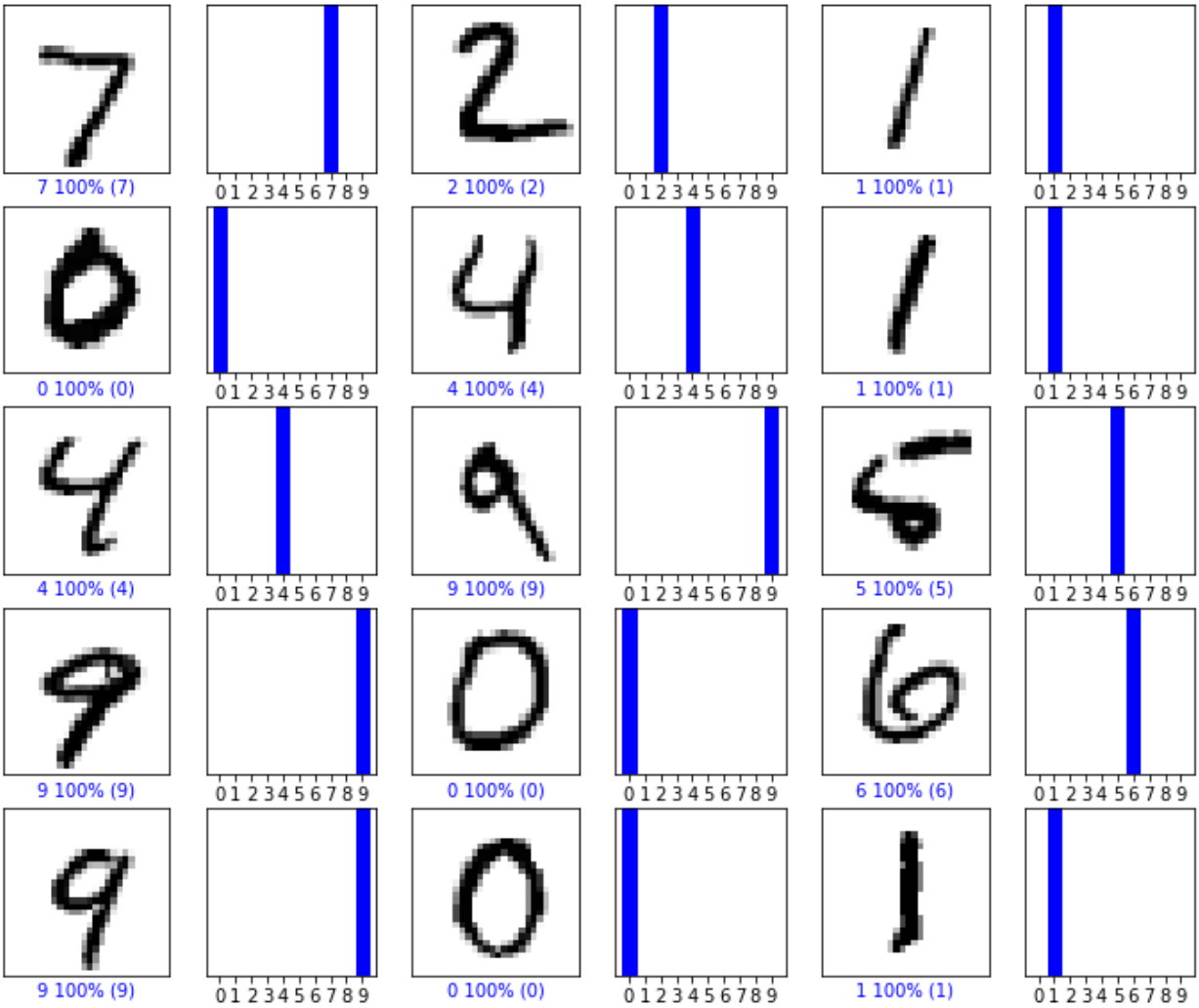
2. Accuracy

```
test_loss, accuracy = model.evaluate(test_images, test_labels, verbose = 2)
print('₩nTest loss : ', test_loss)
print('Test accuracy :', accuracy)
```

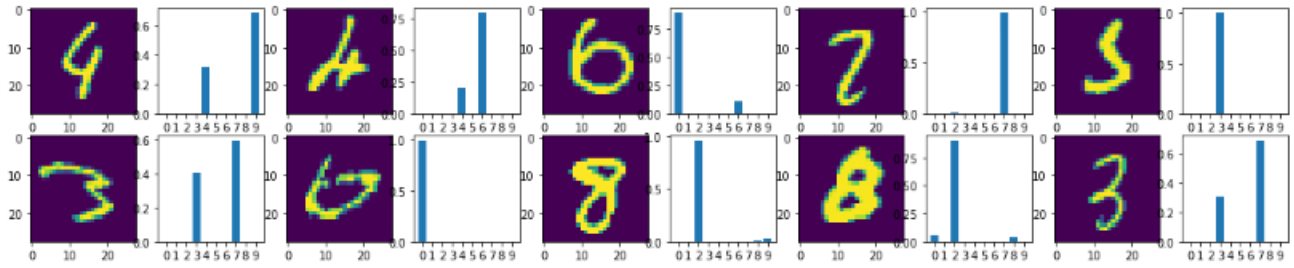
```
10000/1 - 1s - loss: 0.0498 - accuracy: 0.9765

Test loss :  0.09942073709852703
Test accuracy : 0.9765
```

3. Success case Images



4. Failure case Images



MODEL 2

1. Training

```
model = select_model(2)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

model.fit(train_images, train_labels,  epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 37s 620us/sample - loss: 0.3193 - accuracy: 0.9450
Epoch 2/5
60000/60000 [==============================] - 44s 729us/sample - loss: 0.0683 - accuracy: 0.9787
Epoch 3/5
60000/60000 [==============================] - 45s 742us/sample - loss: 0.0575 - accuracy: 0.9824
Epoch 4/5
60000/60000 [==============================] - 44s 740us/sample - loss: 0.0467 - accuracy: 0.9856
Epoch 5/5
60000/60000 [==============================] - 45s 750us/sample - loss: 0.0405 - accuracy: 0.9876- loss:
```

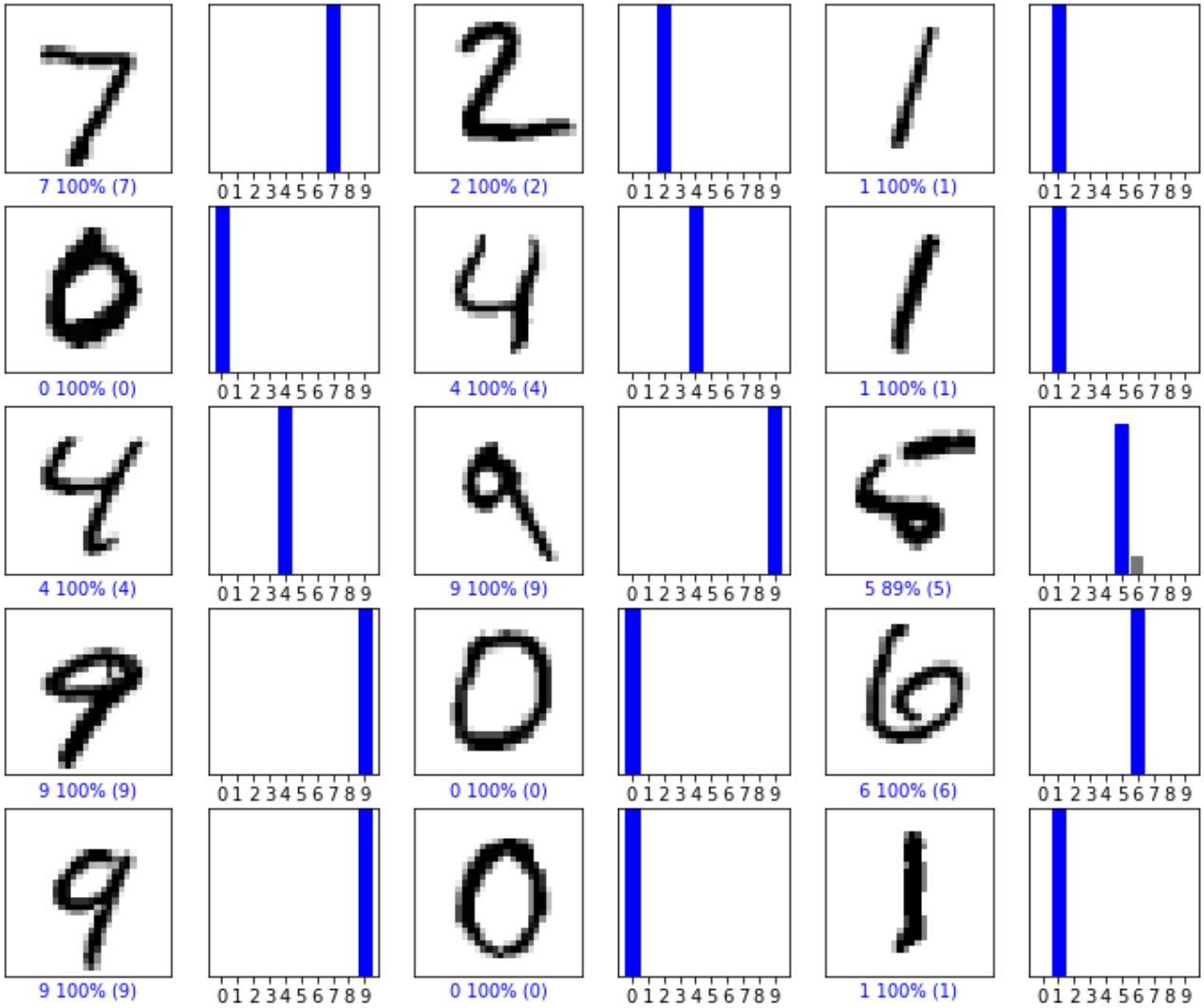
2. Accuracy

```
: test_loss, accuracy = model.evaluate(test_images, test_labels, verbose = 2)
  print('₩nTest loss : ', test_loss)
  print('Test accuracy :', accuracy)
```

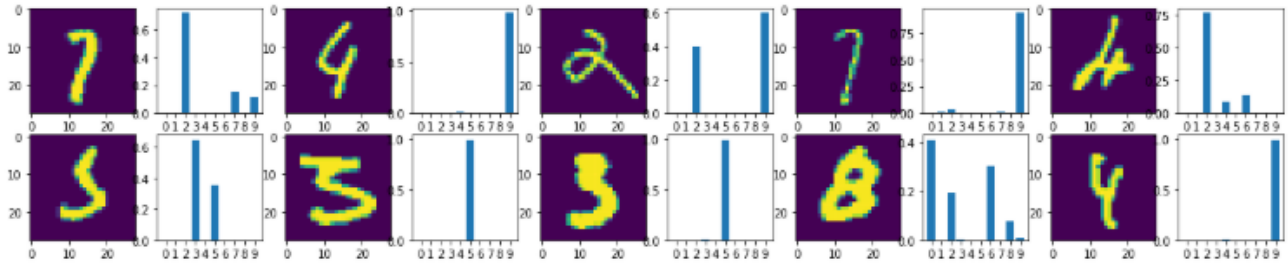
```
10000/1 - 3s - loss: 0.0322 - accuracy: 0.9823

Test loss :  0.06383551980046905
Test accuracy : 0.9823
```

3. Success case Images



4. Failure case Images



MODEL 3

1. Training

```
model = select_model(3)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 128)         73856
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 10)                5130
=================================================================
Total params: 134,730
Trainable params: 134,730
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

model.fit(train_images, train_labels,  epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 64s 1ms/sample - loss: 0.1958 - accuracy: 0.9554
Epoch 2/5
60000/60000 [==============================] - 64s 1ms/sample - loss: 0.0571 - accuracy: 0.9828
Epoch 3/5
60000/60000 [==============================] - 66s 1ms/sample - loss: 0.0459 - accuracy: 0.9859
Epoch 4/5
60000/60000 [==============================] - 52s 865us/sample - loss: 0.0369 - accuracy: 0.9886
Epoch 5/5
60000/60000 [==============================] - 46s 768us/sample - loss: 0.0346 - accuracy: 0.9897-
```

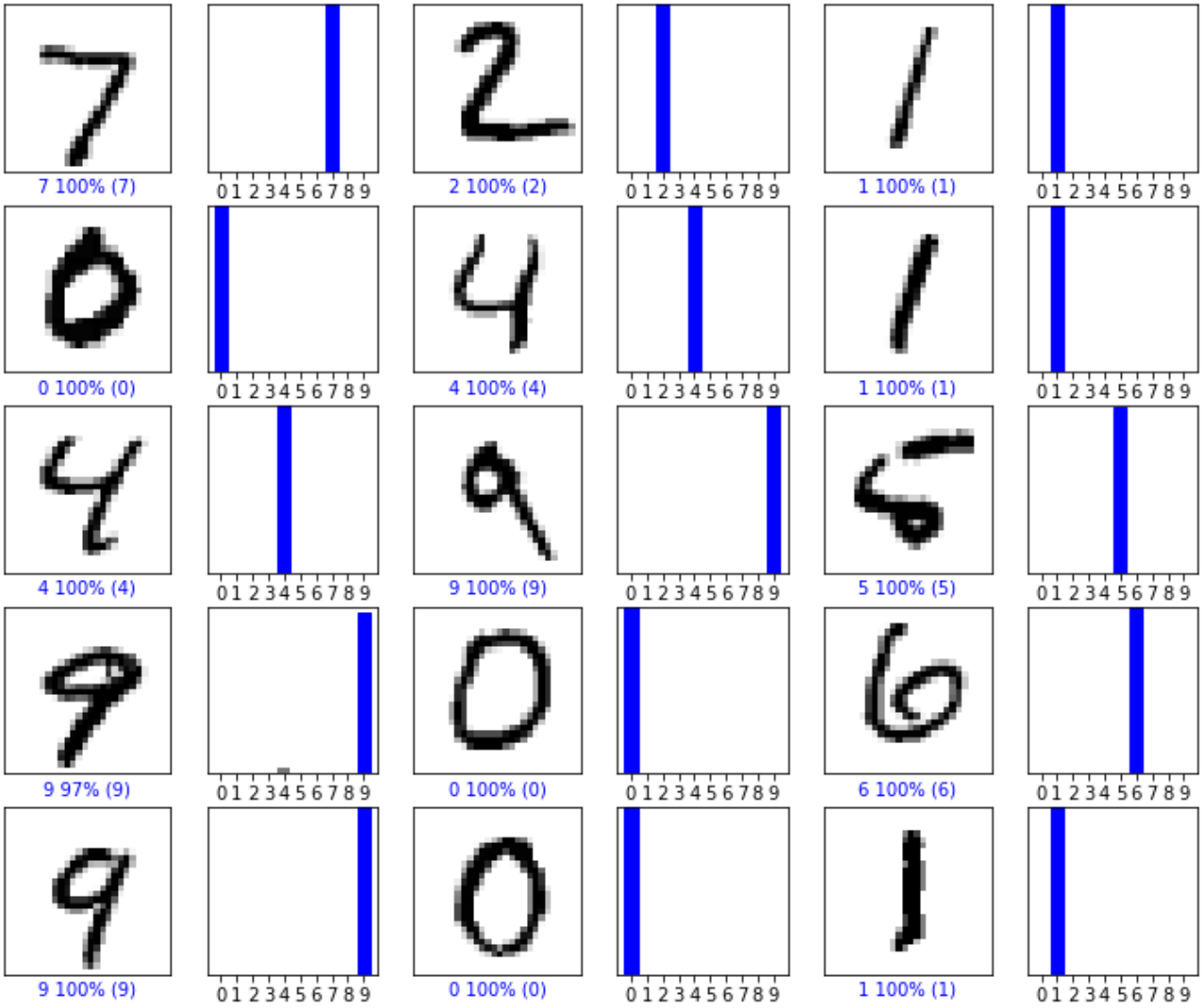
2. Accuracy

```
test_loss, accuracy = model.evaluate(test_images, test_labels, verbose = 2)
print('₩nTest loss : ', test_loss)
print('Test accuracy :', accuracy)
```

```
10000/1 - 2s - loss: 0.0341 - accuracy: 0.9817

Test loss :  0.06570189406646823
Test accuracy : 0.9817
```

3. Success case Images



4. Failure case Images