Part 1 - Report

In order to achieve +95% accuracy on the MNIST training data, I used the same hyperparameters as those in the Fashion MNIST Tutorial. For setting up the layers, I still used the sequential model with the same input for the flattening layer, which flattens the 28x28 array of inputs into a single 1-D array. Then for the dense layer, I used 128 units (the number of nodes/neurons) and a *relu* activation function. I then added a second dense layer with 10 units, which is essentially an array of size 10 for holding the classes of digits 0-9. When it came to compiling the model, I used the *adam* optimizer, *sparse categorical cross entropy* as the loss function, and *accuracy* for metrics. The loss function I have is used for steering the model in the right direction. When it came to training the model, I went with 10 epochs to absolutely guarantee that every run of model.fit() would give an accuracy above 95%. Below is a screenshot of the hyperparameters I used.

```
print("--Make model--")
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10)
])
model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

print("--Fit model--")
model.fit(x_train, y_train, epochs=10, verbose=2)
```

I was able to achieve +98% accuracy with the hyperparameters above.

```
 9 print("--Fit model--")
10 model.fit(x_train, y_train, epochs=10, verbose=2)
```

```
--Make model--
--Fit model--
Epoch 1/10
1875/1875 - 5s - loss: 0.2645 - accuracy: 0.9236 - 5s/epoch - 2ms/step
Epoch 2/10
1875/1875 - 4s - loss: 0.1171 - accuracy: 0.9659 - 4s/epoch - 2ms/step
Epoch 3/10
1875/1875 - 4s - loss: 0.0811 - accuracy: 0.9761 - 4s/epoch - 2ms/step
Epoch 4/10
1875/1875 - 4s - loss: 0.0612 - accuracy: 0.9811 - 4s/epoch - 2ms/step
Epoch 5/10
1875/1875 - 4s - loss: 0.0471 - accuracy: 0.9854 - 4s/epoch - 2ms/step
Epoch 6/10
1875/1875 - 4s - loss: 0.0360 - accuracy: 0.9891 - 4s/epoch - 2ms/step
Epoch 7/10
1875/1875 - 4s - loss: 0.0300 - accuracy: 0.9907 - 4s/epoch - 2ms/step
Epoch 8/10
1875/1875 - 4s - loss: 0.0243 - accuracy: 0.9928 - 4s/epoch - 2ms/step
Epoch 9/10
1875/1875 - 4s - loss: 0.0188 - accuracy: 0.9941 - 4s/epoch - 2ms/step
Epoch 10/10
1875/1875 - 4s - loss: 0.0160 - accuracy: 0.9949 - 4s/epoch - 2ms/step

<keras.callbacks.History at 0x2d7bde78e50>
```

```python
print("--Evaluate model--")
model_loss1, model_acc1 = model.evaluate(x_train,  y_train, verbose=2)
model_loss2, model_acc2 = model.evaluate(x_test,  y_test, verbose=2)
print(f"Train / Test Accuracy: {model_acc1*100:.1f}% / {model_acc2*100:.1f}%")
```

```
--Evaluate model--
1875/1875 - 3s - loss: 0.0165 - accuracy: 0.9949 - 3s/epoch - 2ms/step
313/313 - 1s - loss: 0.0831 - accuracy: 0.9764 - 704ms/epoch - 2ms/step
Train / Test Accuracy: 99.5% / 97.6%
```