In [1]:

```
import tensorflow as tf
```

In [2]:

```
# Load Data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

Baseline Model

In [3]:

```
# Baseline model definition
model = tf.keras.Sequential([
                             tf.keras.layers.Flatten(input_shape=(28, 28)),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(10, activation='softmax')
])
```

In [4]:

```
# Baseline model compilation
# Learning rate 0.001
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

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

In [ ]:

```
# Baseline model fitting

history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=100,
                    validation_data=(x_test, y_test),
                    verbose=2
                    )
```

In [6]:

```
# Baseline model evaluation
model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.3240 - sparse_categorical_accuracy: 0.9205 - 326ms/epoch - 1ms/step
```

Out[6]:

```
[0.3240365982055664, 0.9204999804496765]
```

In [ ]:

```
# Learning rate 0.0001
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy']
             )
history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=100,
                    validation_data=(x_test, y_test),
                    verbose=2
                    )
```

In [8]:

```
# Learning rate 0.0001
model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.3533 - sparse_categorical_accuracy: 0.9241 - 318ms/epoch - 1ms/step
```

Out[8]:

```
[0.35332968831062317, 0.9240999817848206]
```

In [ ]:

```
# Learning rate 0.00001
optimizer = tf.keras.optimizers.Adam(learning_rate=0.00001)

model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy']
             )
history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=100,
                    validation_data=(x_test, y_test),
                    verbose=2
                    )
```

In [10]:

```
# Learning rate 0.00001
model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.3575 - sparse_categorical_accuracy: 0.9230 - 328ms/epoch - 1ms/step
```

Out[10]:

```
[0.3574950397014618, 0.9229999780654907]
```

In [ ]:

```
# SGD
model = tf.keras.Sequential([
                             tf.keras.layers.Flatten(input_shape=(28, 28)),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy']
             )

history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=100,
                    validation_data=(x_test, y_test),
                    verbose=2
                    )
```

In [12]:

```
model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 2.3010 - sparse_categorical_accuracy: 0.1135 - 328ms/epoch - 1ms/step
```

Out[12]:

```
[2.301015853881836, 0.11349999904632568]
```