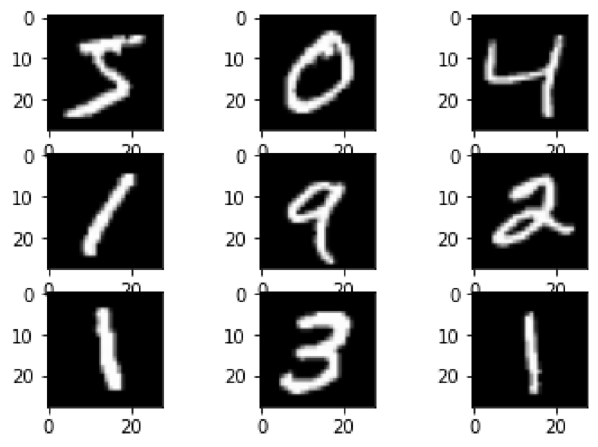
HW-15 Report

Baseline Model

```
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
from matplotlib import pyplot
for i in range(9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(x_train[i], cmap=pyplot.get_cmap('gray'))
pyplot.show()
```



```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

```
x_train shape: (60000, 28, 28, 1)
Number of images in x_train 60000
Number of images in x_test 10000
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D, Convolution2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.models import Sequential
from keras.optimizers import SGD
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from sklearn import datasets
```

```python
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Convolution2D(
    filters = 10,
    kernel_size = (5, 5),
    padding = "same",
    input_shape = input_shape))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a pooling layer
model.add(MaxPooling2D(
    pool_size = (2, 2),
    strides =  (4, 4)))
# Add the second convolution layer
model.add(Convolution2D(
    filters = 50,
    kernel_size = (5, 5),
    padding = "same"))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a second pooling layer
model.add(MaxPooling2D(
    pool_size = (2, 2),
    strides = (2, 2)))
# Flatten the network
model.add(Flatten())
# Add a fully-connected hidden layer
model.add(Dense(300))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a fully-connected output layer
model.add(Dense(10))
# Add a softmax activation function
model.add(Activation("softmax"))
```

```python
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.3878 - accuracy: 0.8822
Epoch 2/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0632 - accuracy: 0.9813
Epoch 3/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0424 - accuracy: 0.9875
Epoch 4/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0305 - accuracy: 0.9894
Epoch 5/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0239 - accuracy: 0.9925
<tensorflow.python.keras.callbacks.History at 0x7ff9f2dbea10>
```

```python
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 3s 11ms/step - loss: 0.0370 - accuracy: 0.9887
[0.037038736045360565, 0.9886999726295471]
```

```python
# Code inspired from following resources
# https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d
# https://github.com/matthewrenze/lenet-on-mnist-with-keras-and-tensorflow-in-python/blob/master/MNIST-LeNet.py
```

Changes

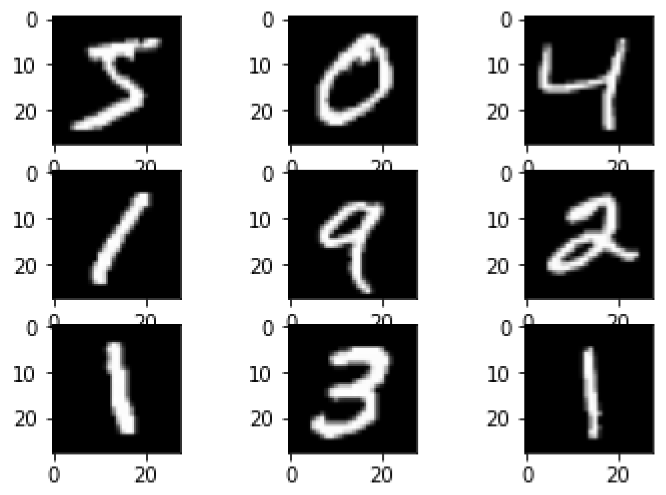
- Changed the filter value from 10 to 50 in the first layer.
- Changed strides from (4,4) to (2,2) in the second layer.
- Increased the hidden layer density from 300 to 500 in the 4th layer.

Accuracy improved from 98.89% to 99.07% on test data.

Improved Model

```
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
from matplotlib import pyplot
for i in range(9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(x_train[i], cmap=pyplot.get_cmap('gray'))
pyplot.show()
```

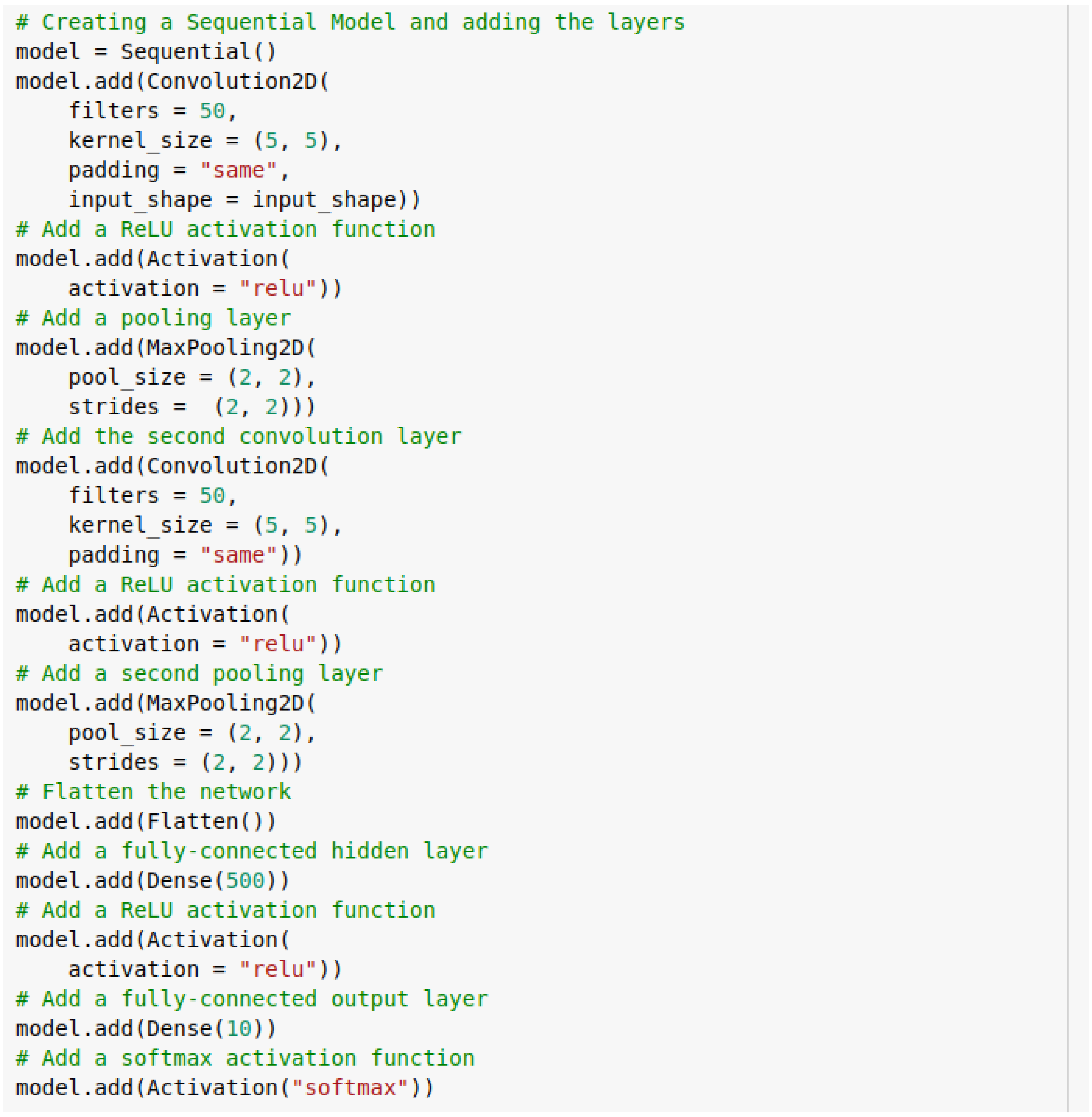
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```



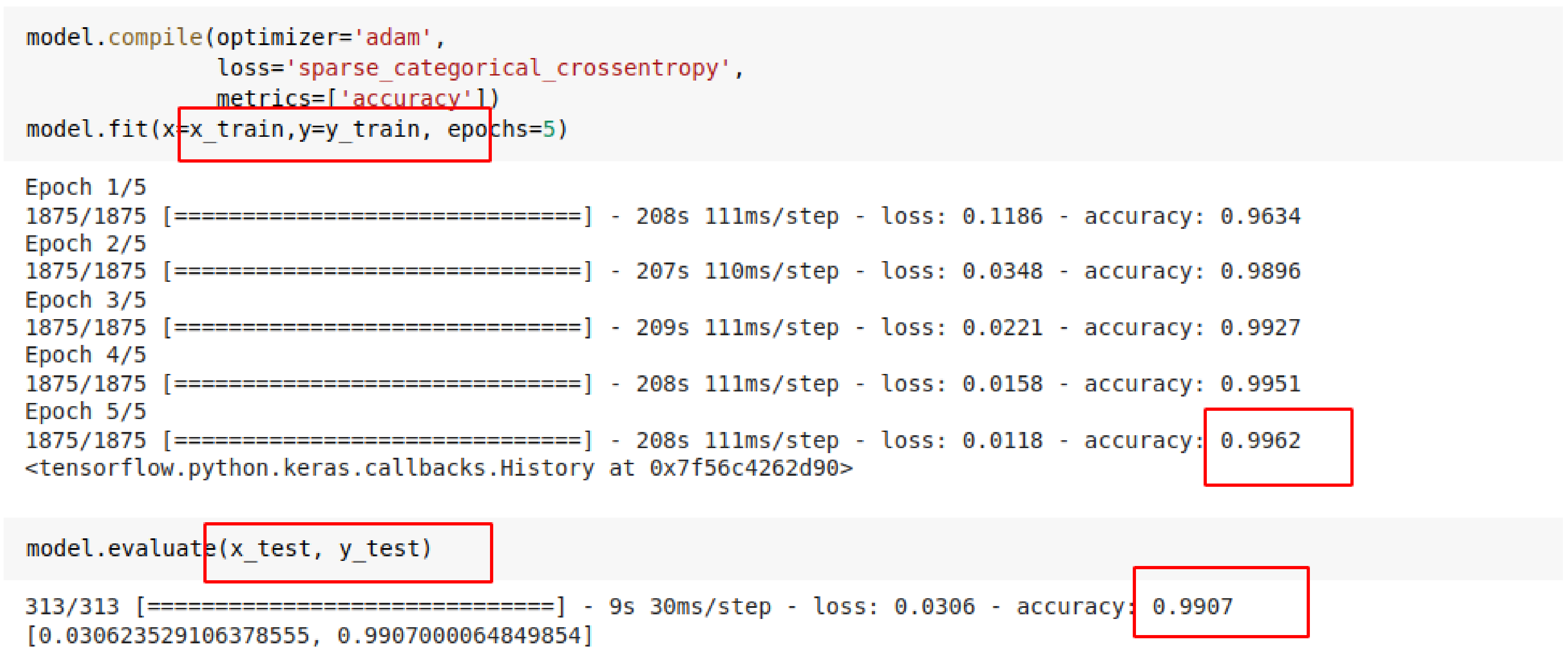
```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

```
x_train shape: (60000, 28, 28, 1)
Number of images in x_train 60000
Number of images in x_test 10000
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D, Convolution2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.models import Sequential
from keras.optimizers import SGD
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from sklearn import datasets
```



```python
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Convolution2D(
    filters = 50,
    kernel_size = (5, 5),
    padding = "same",
    input_shape = input_shape))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a pooling layer
model.add(MaxPooling2D(
    pool_size = (2, 2),
    strides =  (2, 2)))
# Add the second convolution layer
model.add(Convolution2D(
    filters = 50,
    kernel_size = (5, 5),
    padding = "same"))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a second pooling layer
model.add(MaxPooling2D(
    pool_size = (2, 2),
    strides = (2, 2)))
# Flatten the network
model.add(Flatten())
# Add a fully-connected hidden layer
model.add(Dense(500))
# Add a ReLU activation function
model.add(Activation(
    activation = "relu"))
# Add a fully-connected output layer
model.add(Dense(10))
# Add a softmax activation function
model.add(Activation("softmax"))
```



```python
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 208s 111ms/step - loss: 0.1186 - accuracy: 0.9634
Epoch 2/5
1875/1875 [==============================] - 207s 110ms/step - loss: 0.0348 - accuracy: 0.9896
Epoch 3/5
1875/1875 [==============================] - 209s 111ms/step - loss: 0.0221 - accuracy: 0.9927
Epoch 4/5
1875/1875 [==============================] - 208s 111ms/step - loss: 0.0158 - accuracy: 0.9951
Epoch 5/5
1875/1875 [==============================] - 208s 111ms/step - loss: 0.0118 - accuracy: 0.9962
<tensorflow.python.keras.callbacks.History at 0x7f56c4262d90>
```

```python
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 9s 30ms/step - loss: 0.0306 - accuracy: 0.9907
[0.030623529106378555, 0.9907000064849854]
```