

```
In [ ]: ###Kulkarni Ganesh Pramod
        ## Roll No:-45
        ##Class BE(IT)
```

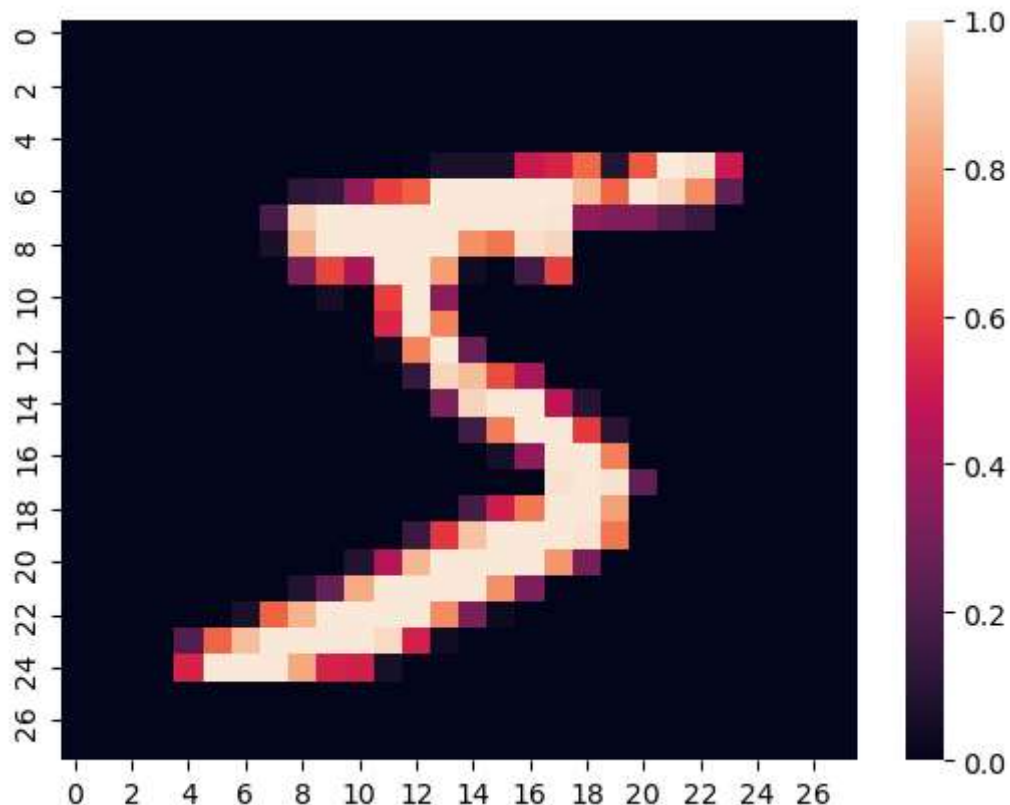
```
In [1]: import tensorflow as tf
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
        from tensorflow.keras.layers import Dropout, Flatten
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: mnist = tf.keras.datasets.mnist
        (x_train, y_train) , (x_test, y_test) = mnist.load_data() # Data Loading
        x_train, x_test = x_train/255.0 , x_test/255.0 #Normalizing the dat
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz> (<https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>)

11490434/11490434 [=====] - 5s 0us/step

```
In [3]: sns.heatmap(x_train[0])
        plt.show()
```



```
In [4]: model = Sequential([
        Flatten(input_shape=(28,28)),
        Dense(128, activation="relu"),
        Dropout(0.2),
        Dense(10)
    ])
```

```
In [5]: predictions = model(x_train[:1]).numpy()
        predictions
```

```
Out[5]: array([[ -0.15549624,  0.53050435,  0.15746787, -0.0107909 ,  0.17775309,
                -0.15343924,  0.52444154,  0.0394171 , -0.16110222,  0.08558373]],
              dtype=float32)
```

```
In [6]: tf.nn.softmax(predictions).numpy()
```

```
Out[6]: array([[0.07483643, 0.14860702, 0.10233675, 0.08648838, 0.10443388,
                0.07499052, 0.14770877, 0.09094165, 0.07441806, 0.09523854]],
              dtype=float32)
```

```
In [7]: loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
In [8]: model.compile(optimizer="adam", loss = loss_fn, metrics=["accuracy"])
```

```
In [9]: model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [=====] - 6s 2ms/step - loss: 0.2977 - accur
acy: 0.9137
Epoch 2/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.1422 - accur
acy: 0.9587
Epoch 3/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.1076 - accur
acy: 0.9668
Epoch 4/5
1875/1875 [=====] - 3s 1ms/step - loss: 0.0892 - accur
acy: 0.9729
Epoch 5/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0737 - accur
acy: 0.9770
```

```
Out[9]: <keras.callbacks.History at 0x271f72ac610>
```

```
In [10]: model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.0732 - accuracy: 0.9777 - 372ms/epoch - 1ms/step
```

```
Out[10]: [0.07321574538946152, 0.9776999950408936]
```

```
In [15]: val = model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), bat
```

```
Epoch 1/5
```

```
300/300 [=====] - 1s 3ms/step - loss: 0.0513 - accurac  
y: 0.9843 - val_loss: 0.0639 - val_accuracy: 0.9806
```

```
Epoch 2/5
```

```
300/300 [=====] - 1s 3ms/step - loss: 0.0460 - accurac  
y: 0.9862 - val_loss: 0.0623 - val_accuracy: 0.9813
```

```
Epoch 3/5
```

```
300/300 [=====] - 1s 3ms/step - loss: 0.0437 - accurac  
y: 0.9869 - val_loss: 0.0602 - val_accuracy: 0.9820
```

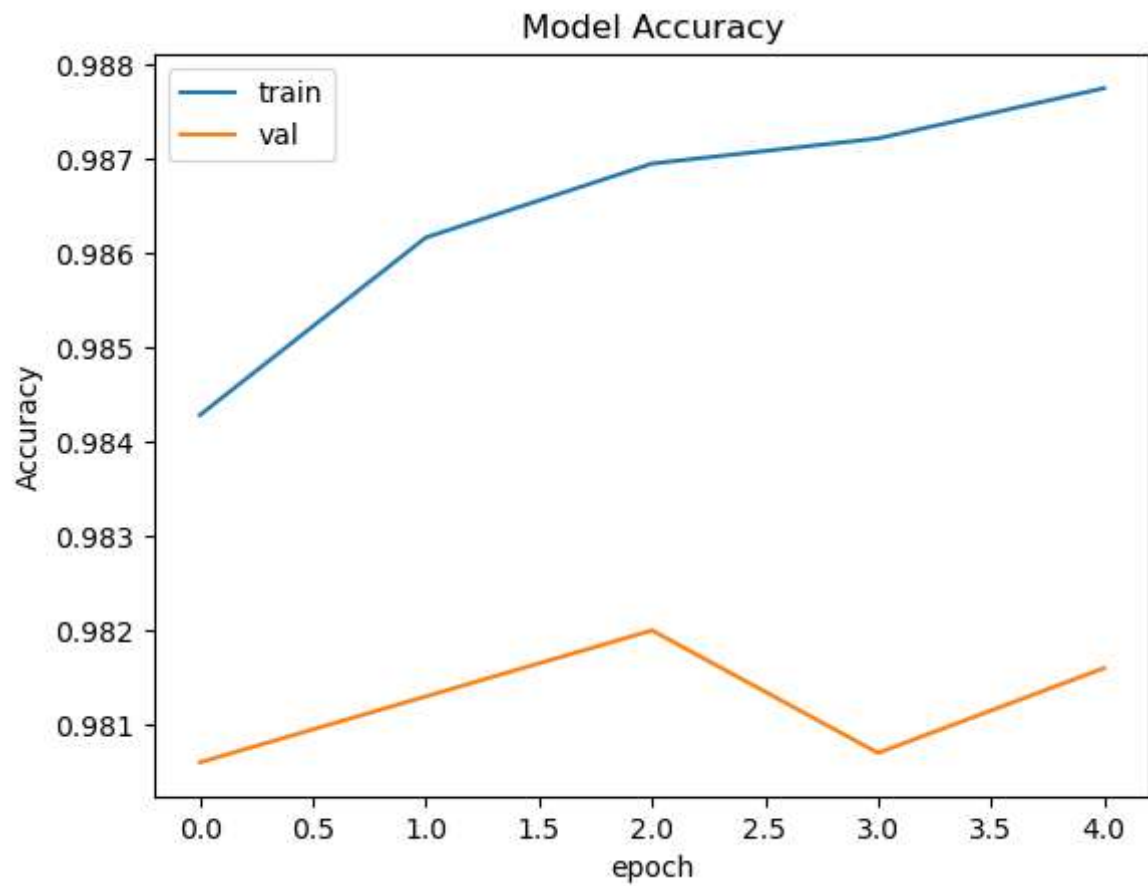
```
Epoch 4/5
```

```
300/300 [=====] - 1s 3ms/step - loss: 0.0417 - accurac  
y: 0.9872 - val_loss: 0.0611 - val_accuracy: 0.9807
```

```
Epoch 5/5
```

```
300/300 [=====] - 1s 3ms/step - loss: 0.0395 - accurac  
y: 0.9877 - val_loss: 0.0607 - val_accuracy: 0.9816
```

```
In [16]: plt.title("Model Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("epoch")
3
plt.plot(val.history["accuracy"])
plt.plot(val.history["val_accuracy"])
plt.legend(["train", "val"])
plt.show()
```



In [ ]: