1. Import the MNIST Dataset.

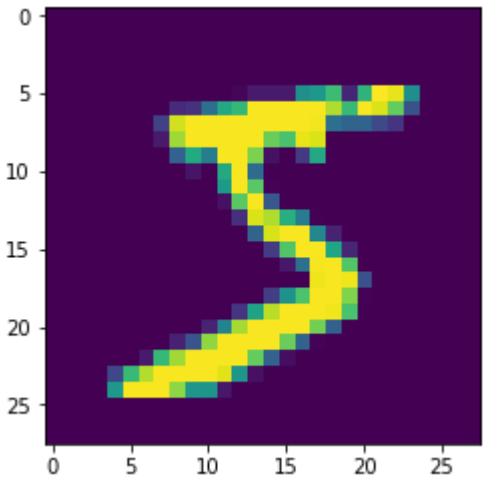
In [2]:
```
import tensorflow as tf
import matplotlib.pyplot as plt

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

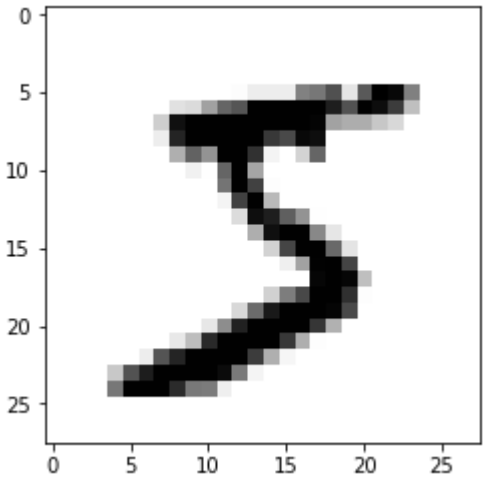
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/mnist.npz
11493376/11490434 [==============================] - 3s 0us/step
```

2. MNIST Dataset

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



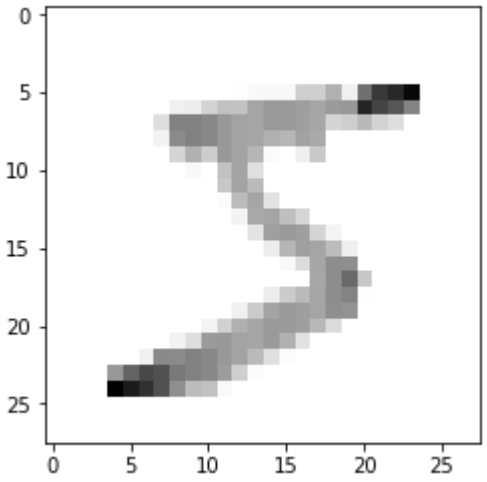
In [4]:
```
plt.imshow(x_train[0], cmap = plt.cm.binary)
plt.show()
```



3. Sequential Model - Normalize

In [5]:
```
x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
```

In [6]:
```
plt.imshow(x_train[0], cmap = plt.cm.binary)
plt.show()
```



3. Input (32,32,10), SGD, batch_size=64, epochs=20

In [7]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD()
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=64, epochs=20)
```

```
Using TensorFlow backend.
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 1s 20us/sample - loss: 1.6825
- accuracy: 0.5153
Epoch 2/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.7494
- accuracy: 0.8041
Epoch 3/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.5016
- accuracy: 0.8587
Epoch 4/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.4137
- accuracy: 0.8834
Epoch 5/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3693
- accuracy: 0.8943
Epoch 6/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3426
- accuracy: 0.9026
Epoch 7/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3236
- accuracy: 0.9071
Epoch 8/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3088
- accuracy: 0.9108
Epoch 9/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.2962
- accuracy: 0.9146
Epoch 10/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.2857
- accuracy: 0.9172
Epoch 11/20
60000/60000 [==============================] - 1s 21us/sample - loss: 0.2757
- accuracy: 0.9199
Epoch 12/20
60000/60000 [==============================] - 1s 24us/sample - loss: 0.2667
- accuracy: 0.9227s - l - ETA: 0s - loss: 0.2655
Epoch 13/20
60000/60000 [==============================] - ETA: 0s - loss: 0.2587 - accur
acy: 0.9248 ETA: 0s - loss: 0 - 1s 22us/sample - loss: 0.2582 - accuracy: 0.9
250
Epoch 14/20
60000/60000 [==============================] - 1s 21us/sample - loss: 0.2498
- accuracy: 0.9273
Epoch 15/20
60000/60000 [==============================] - 1s 22us/sample - loss: 0.2424
- accuracy: 0.9291
Epoch 16/20
60000/60000 [==============================] - 1s 22us/sample - loss: 0.2352
- accuracy: 0.9317
Epoch 17/20
60000/60000 [==============================] - ETA: 0s - loss: 0.2287 - accur
acy: 0.93 - 1s 23us/sample - loss: 0.2281 - accuracy: 0.9338
Epoch 18/20
60000/60000 [==============================] - 1s 22us/sample - loss: 0.2216
- accuracy: 0.9360s - loss: 0.2122 - accuracy
Epoch 19/20
```

```
60000/60000 [==============================] - 1s 22us/sample - loss: 0.2152
- accuracy: 0.9378s - loss: 0.2
Epoch 20/20
60000/60000 [==============================] - 1s 24us/sample - loss: 0.2093
- accuracy: 0.9397s - los
```

Out[7]:
```
<tensorflow.python.keras.callbacks.History at 0x25aaff54748>
```

In [8]:
```
val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
10000/10000 [==============================] - 0s 24us/sample - loss: 0.2165
- accuracy: 0.9370
Validation Accuracy: 0.937
```

In [9]:
```
predictions = model.predict([x_test])
```

In [10]:
```
print(predictions)
```

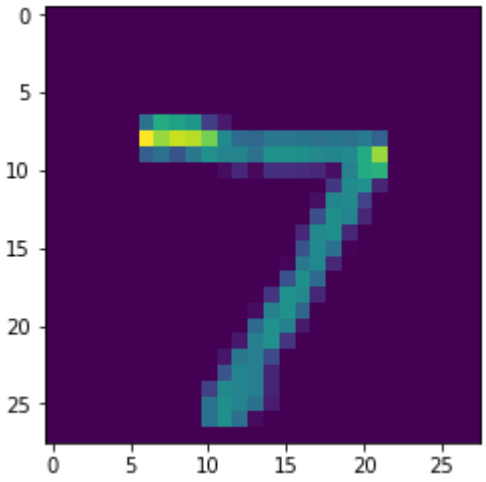
```
[[9.73212627e-06 1.25599300e-07 7.07117666e-04 ... 9.96481419e-01
  1.16260060e-06 7.62079144e-05]
 [2.83717585e-04 4.52270004e-04 9.86150086e-01 ... 3.65136671e-10
  2.43056129e-04 2.51890973e-11]
 [1.91952381e-07 9.97459948e-01 5.42321417e-04 ... 2.19743670e-04
  9.73697461e-04 3.27786875e-05]
 ...
 [6.48240857e-08 1.88459962e-05 1.00796215e-05 ... 4.77399357e-04
  3.79088498e-03 1.18911592e-02]
 [2.07536141e-04 4.87227917e-05 1.03505408e-06 ... 4.84856002e-07
  4.64795753e-02 3.91939602e-06]
 [2.69137614e-04 1.29105308e-06 3.55077558e-03 ... 3.66304000e-08
  4.97732981e-05 9.02434323e-08]]
```

In [11]:
```
import numpy as np

print(np.argmax([predictions[0]]))
```

```
7
```

In [12]:
```
plt.imshow(x_test[0])
plt.show()
```



4. Output(64,64,10)

When the number of nuerons are increased, the time taken to train the model increases. The accuracy here didn't change significantly.

In [13]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(64, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(64, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD()
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=64, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 1s 23us/sample - loss: 1.6257
- accuracy: 0.5661
Epoch 2/20
60000/60000 [==============================] - 1s 19us/sample - loss: 0.6221
- accuracy: 0.8383
Epoch 3/20
60000/60000 [==============================] - 1s 19us/sample - loss: 0.4409
- accuracy: 0.8773
Epoch 4/20
60000/60000 [==============================] - 1s 19us/sample - loss: 0.3764
- accuracy: 0.8932
Epoch 5/20
60000/60000 [==============================] - 1s 24us/sample - loss: 0.3415
- accuracy: 0.9025
Epoch 6/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.3184
- accuracy: 0.9088
Epoch 7/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.3005
- accuracy: 0.9134s - loss: 0.3007 - accuracy
Epoch 8/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.2858
- accuracy: 0.9178s - loss:
Epoch 9/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.2734
- accuracy: 0.9217
Epoch 10/20
60000/60000 [==============================] - 2s 27us/sample - loss: 0.2623
- accuracy: 0.9253
Epoch 11/20
60000/60000 [==============================] - 2s 27us/sample - loss: 0.2523
- accuracy: 0.9283
Epoch 12/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.2429
- accuracy: 0.9310
Epoch 13/20
60000/60000 [==============================] - 1s 25us/sample - loss: 0.2344
- accuracy: 0.9328
Epoch 14/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.2267
- accuracy: 0.9350
Epoch 15/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.2195
- accuracy: 0.9371
Epoch 16/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.2127
- accuracy: 0.9390
Epoch 17/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.2067
- accuracy: 0.9407
Epoch 18/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.2008
- accuracy: 0.9427
Epoch 19/20
60000/60000 [==============================] - 2s 25us/sample - loss: 0.1956
```

```
- accuracy: 0.9439
Epoch 20/20
60000/60000 [==============================] - 2s 26us/sample - loss: 0.1903
- accuracy: 0.9456
10000/10000 [==============================] - 0s 47us/sample - loss: 0.1958
- accuracy: 0.9425
Validation Accuracy: 0.9425
```

5.Output(32,32,10), SGD(lr=0.01, decay_rate=0.000001, momentum_rate=0.9)

In [14]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.01, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=64, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 1s 20us/sample - loss: 1.9127
- accuracy: 0.4407
Epoch 2/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.8043
- accuracy: 0.8040
Epoch 3/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.4888
- accuracy: 0.8673
Epoch 4/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.4034
- accuracy: 0.8861s -
Epoch 5/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3644
- accuracy: 0.8964
Epoch 6/20
60000/60000 [==============================] - 1s 18us/sample - loss: 0.3402
- accuracy: 0.9020
Epoch 7/20
60000/60000 [==============================] - 1s 21us/sample - loss: 0.3226
- accuracy: 0.9068
Epoch 8/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.3085
- accuracy: 0.9107s - loss: 0.3098 - accuracy
Epoch 9/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2966
- accuracy: 0.9141
Epoch 10/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2857
- accuracy: 0.9172
Epoch 11/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2759
- accuracy: 0.9204
Epoch 12/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2668
- accuracy: 0.9236s - loss: 0.2531 - accuracy - ETA:
Epoch 13/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2587
- accuracy: 0.9252
Epoch 14/20
60000/60000 [==============================] - 1s 24us/sample - loss: 0.2511
- accuracy: 0.9280
Epoch 15/20
60000/60000 [==============================] - 1s 25us/sample - loss: 0.2439
- accuracy: 0.9300
Epoch 16/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2374
- accuracy: 0.9318
Epoch 17/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2313
- accuracy: 0.9335s - loss:
Epoch 18/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2256
- accuracy: 0.9357
Epoch 19/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2200
```

```
- accuracy: 0.9374
Epoch 20/20
60000/60000 [==============================] - 1s 23us/sample - loss: 0.2151
- accuracy: 0.9390
10000/10000 [==============================] - 0s 33us/sample - loss: 0.2154
- accuracy: 0.9354
Validation Accuracy: 0.9354
```

Momentum is used for converging in the models. In our case, even if we use the momentum for faster convergence of gradients, there is no change in the accuracy. It is almost the same even without using it.

6. Input(32,32,10), batch_size=128 and Input(32,32,10), batch_size=32

In [15]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.01, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=128, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 1s 13us/sample - loss: 2.1416
- accuracy: 0.2872
Epoch 2/20
60000/60000 [==============================] - 1s 11us/sample - loss: 1.4786
- accuracy: 0.6686
Epoch 3/20
60000/60000 [==============================] - 1s 10us/sample - loss: 0.8518
- accuracy: 0.8022
Epoch 4/20
60000/60000 [==============================] - 1s 11us/sample - loss: 0.6050
- accuracy: 0.8416
Epoch 5/20
60000/60000 [==============================] - 1s 11us/sample - loss: 0.5011
- accuracy: 0.8615
Epoch 6/20
60000/60000 [==============================] - 1s 12us/sample - loss: 0.4445
- accuracy: 0.8748
Epoch 7/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.4086
- accuracy: 0.8834
Epoch 8/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3835
- accuracy: 0.8901
Epoch 9/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3649
- accuracy: 0.8953
Epoch 10/20
60000/60000 [==============================] - 1s 13us/sample - loss: 0.3500
- accuracy: 0.8995
Epoch 11/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3378
- accuracy: 0.9025
Epoch 12/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3274
- accuracy: 0.9053
Epoch 13/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3183
- accuracy: 0.9089
Epoch 14/20
60000/60000 [==============================] - 1s 13us/sample - loss: 0.3104
- accuracy: 0.9108
Epoch 15/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.3030
- accuracy: 0.9128
Epoch 16/20
60000/60000 [==============================] - 1s 13us/sample - loss: 0.2965
- accuracy: 0.9145
Epoch 17/20
60000/60000 [==============================] - 1s 13us/sample - loss: 0.2903
- accuracy: 0.9169
Epoch 18/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.2848
- accuracy: 0.9188
Epoch 19/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.2794
```

```
- accuracy: 0.9201
Epoch 20/20
60000/60000 [==============================] - 1s 14us/sample - loss: 0.2744
- accuracy: 0.9216
10000/10000 [==============================] - 0s 34us/sample - loss: 0.2696
- accuracy: 0.9253
Validation Accuracy: 0.9253
```

In [16]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.01, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=32, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 2s 38us/sample - loss: 1.3381
- accuracy: 0.6321
Epoch 2/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.4490
- accuracy: 0.8742
Epoch 3/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.3525
- accuracy: 0.8989s - loss: 0
Epoch 4/20
60000/60000 [==============================] - 2s 40us/sample - loss: 0.3148
- accuracy: 0.9094s - loss: 0.3224 -  - ETA: 0s - loss: 0.3
Epoch 5/20
60000/60000 [==============================] - 2s 39us/sample - loss: 0.2903
- accuracy: 0.9164s - los - ETA: 1s - loss: 0 - ETA
Epoch 6/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.2716
- accuracy: 0.9215
Epoch 7/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.2567
- accuracy: 0.9260s - loss: 0.2626 - accuracy: 0.92 - E
Epoch 8/20
60000/60000 [==============================] - 2s 41us/sample - loss: 0.2435
- accuracy: 0.9301
Epoch 9/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.2318
- accuracy: 0.9339
Epoch 10/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.2219
- accuracy: 0.9365
Epoch 11/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.2131
- accuracy: 0.9390
Epoch 12/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.2051
- accuracy: 0.9420
Epoch 13/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1977
- accuracy: 0.9438
Epoch 14/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1909
- accuracy: 0.9456
Epoch 15/20
60000/60000 [==============================] - ETA: 0s - loss: 0.1844 - accur
acy: 0.9472 ETA: 0s - loss: 0.1840 - accuracy - 2s 40us/sample - loss: 0.1845
- accuracy: 0.9472
Epoch 16/20
60000/60000 [==============================] - 2s 38us/sample - loss: 0.1784
- accuracy: 0.9494s
Epoch 17/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1732
- accuracy: 0.9509
Epoch 18/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1677
- accuracy: 0.9525
Epoch 19/20
```

```
60000/60000 [==============================] - 2s 38us/sample - loss: 0.1627
- accuracy: 0.9533
Epoch 20/20
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1578
- accuracy: 0.9551
10000/10000 [==============================] - 0s 30us/sample - loss: 0.1703
- accuracy: 0.9503
Validation Accuracy: 0.9503
```

#conclusion Validation accuracy for batch_size_128 = 0.91 Validation accuracy for batch_size_32 = 0.95 Conclusion: There is a slight increase in accuracy as batch size is decreased.

7. Adding hidden layer when input (32,32,32,10)

In [17]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.01, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=32, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 2s 32us/sample - loss: 1.4222
- accuracy: 0.6020
Epoch 2/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.4617
- accuracy: 0.8675
Epoch 3/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.3505
- accuracy: 0.8986
Epoch 4/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.3022
- accuracy: 0.9120
Epoch 5/20
60000/60000 [==============================] - 2s 31us/sample - loss: 0.2697
- accuracy: 0.9205
Epoch 6/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2439
- accuracy: 0.9278
Epoch 7/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2232
- accuracy: 0.9338
Epoch 8/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2058
- accuracy: 0.9389
Epoch 9/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1918
- accuracy: 0.9431
Epoch 10/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1795
- accuracy: 0.9467
Epoch 11/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.1687
- accuracy: 0.9496
Epoch 12/20
60000/60000 [==============================] - 2s 31us/sample - loss: 0.1587
- accuracy: 0.9532
Epoch 13/20
60000/60000 [==============================] - 2s 31us/sample - loss: 0.1506
- accuracy: 0.9552
Epoch 14/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1424
- accuracy: 0.9580
Epoch 15/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1359
- accuracy: 0.9590
Epoch 16/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1296
- accuracy: 0.9610
Epoch 17/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.1243
- accuracy: 0.9626
Epoch 18/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1187
- accuracy: 0.9644
Epoch 19/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1140
```

```
- accuracy: 0.9658
Epoch 20/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1094
- accuracy: 0.9667
10000/10000 [==============================] - 0s 23us/sample - loss: 0.1383
- accuracy: 0.9596
Validation Accuracy: 0.9596
```

Conclusion: Addition of extra hidden layer has increased the processing time but did not bring any increase in accuracy.

8. Input (128,128,10) and adding Drop out layer where dropout = 0.5

Dropout is used to prevent overfitting.

In [18]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.01, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=32, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 3s 45us/sample - loss: 1.7499
- accuracy: 0.3906
Epoch 2/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.9973
- accuracy: 0.6687
Epoch 3/20
60000/60000 [==============================] - 3s 44us/sample - loss: 0.8054
- accuracy: 0.7421
Epoch 4/20
60000/60000 [==============================] - 3s 44us/sample - loss: 0.7122
- accuracy: 0.7751
Epoch 5/20
60000/60000 [==============================] - 3s 44us/sample - loss: 0.6479
- accuracy: 0.7987
Epoch 6/20
60000/60000 [==============================] - 3s 44us/sample - loss: 0.5990
- accuracy: 0.8152
Epoch 7/20
60000/60000 [==============================] - 3s 44us/sample - loss: 0.5715
- accuracy: 0.8248
Epoch 8/20
60000/60000 [==============================] - 3s 47us/sample - loss: 0.5434
- accuracy: 0.8346
Epoch 9/20
60000/60000 [==============================] - 3s 45us/sample - loss: 0.5199
- accuracy: 0.8407
Epoch 10/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.5018
- accuracy: 0.8490
Epoch 11/20
60000/60000 [==============================] - 3s 47us/sample - loss: 0.4809
- accuracy: 0.8569
Epoch 12/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.4666
- accuracy: 0.8587
Epoch 13/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.4558
- accuracy: 0.8624
Epoch 14/20
60000/60000 [==============================] - 3s 48us/sample - loss: 0.4420
- accuracy: 0.8677
Epoch 15/20
60000/60000 [==============================] - 3s 48us/sample - loss: 0.4307
- accuracy: 0.8704
Epoch 16/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.4200
- accuracy: 0.8727
Epoch 17/20
60000/60000 [==============================] - 3s 46us/sample - loss: 0.4088
- accuracy: 0.8780
Epoch 18/20
60000/60000 [==============================] - 3s 58us/sample - loss: 0.4014
- accuracy: 0.8793
Epoch 19/20
60000/60000 [==============================] - 4s 73us/sample - loss: 0.3974
```

```
- accuracy: 0.8795
Epoch 20/20
60000/60000 [==============================] - 4s 63us/sample - loss: 0.3916
- accuracy: 0.8821
10000/10000 [==============================] - 0s 39us/sample - loss: 0.1805
- accuracy: 0.9452
Validation Accuracy: 0.9452
```

Adding the dropout layer prevents the overfitting which is good for the model not to memorize the patterns. Accuracy is not increased drastically.

1. Model with better parameters

In [28]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
#model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
#model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(32, activation=tf.nn.relu))
#model.add(tf.keras.layers.Dropout(rate=0.5, noise_shape=None, seed=None))
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

from keras import optimizers

sgd = optimizers.SGD(lr=0.09, decay=0.000001, momentum=0.9)
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics
=['accuracy'])
model.fit(x_train, y_train, batch_size=32, epochs=20)

val_loss, val_acc = model.evaluate(x_test, y_test)
print("Validation Accuracy: "+str(val_acc))
```

```
Train on 60000 samples
Epoch 1/20
60000/60000 [==============================] - 2s 31us/sample - loss: 1.2566
- accuracy: 0.6555
Epoch 2/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.4504
- accuracy: 0.8741
Epoch 3/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.3582
- accuracy: 0.8965
Epoch 4/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.3176
- accuracy: 0.9080
Epoch 5/20
60000/60000 [==============================] - 2s 31us/sample - loss: 0.2898
- accuracy: 0.9156
Epoch 6/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2677
- accuracy: 0.9228
Epoch 7/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2491
- accuracy: 0.9277
Epoch 8/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2336
- accuracy: 0.9334
Epoch 9/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2201
- accuracy: 0.9365
Epoch 10/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.2085
- accuracy: 0.9396
Epoch 11/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1982
- accuracy: 0.9423
Epoch 12/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1892
- accuracy: 0.9447
Epoch 13/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1811
- accuracy: 0.9470
Epoch 14/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1733
- accuracy: 0.9488
Epoch 15/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1669
- accuracy: 0.9510
Epoch 16/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1607
- accuracy: 0.9530
Epoch 17/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1550
- accuracy: 0.9546
Epoch 18/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.1499
- accuracy: 0.9561
Epoch 19/20
60000/60000 [==============================] - 2s 30us/sample - loss: 0.1449
```

```
- accuracy: 0.9574
Epoch 20/20
60000/60000 [==============================] - 2s 29us/sample - loss: 0.1402
- accuracy: 0.9594
10000/10000 [==============================] - 0s 23us/sample - loss: 0.1577
- accuracy: 0.9540
Validation Accuracy: 0.954
```

Conclusion: I think, for the given data, model with 2 hidden layers with 32,32 neurons with learning rate = 0.09 and batch size=32 will have good accuracy. Further increase in learning rate or hidden layers or batch size would decrease the accuracy progressively.

Question 10 -

Given the data, below parameters are important a. Number of hidden layers b. Number of neurons per layer c. Batch_size d. Learning rate

In [ ]: