In [13]:

```
# importing modules
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Activation
import matplotlib.pyplot as plt
```

In [14]:

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

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

In [15]:

```
# Cast the records into float values
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

# normalize image pixel values by dividing
# by 255
gray_scale = 255
x_train /= gray_scale
x_test /= gray_scale
```

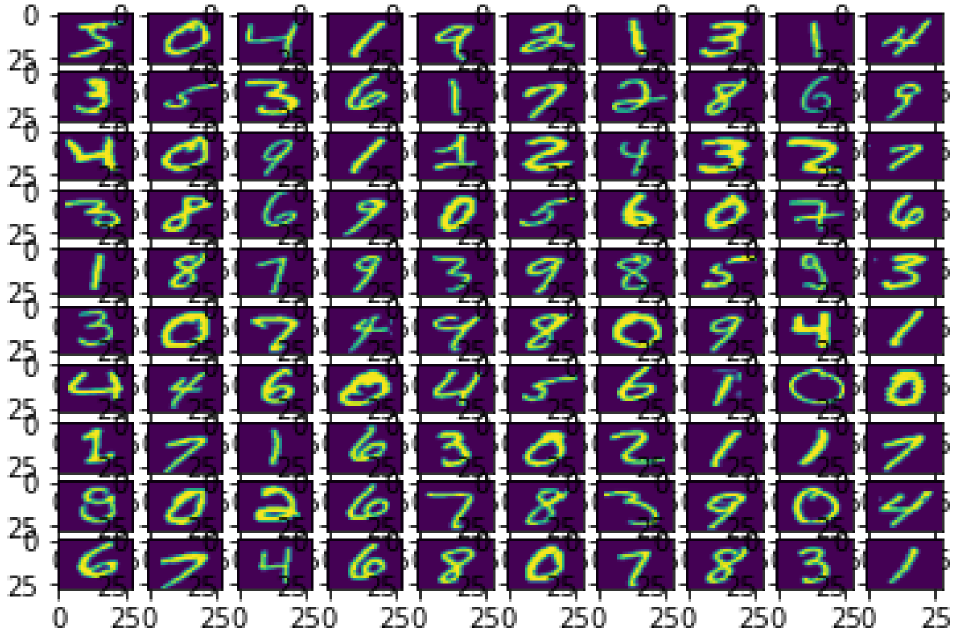
In [16]:

```
print("Feature matrix:", x_train.shape)
print("Target matrix:", x_test.shape)
print("Feature matrix:", y_train.shape)
print("Target matrix:", y_test.shape)
```

```
Feature matrix: (60000, 28, 28)
Target matrix: (10000, 28, 28)
Feature matrix: (60000,)
Target matrix: (10000,)
```

In [17]:

```
fig, ax = plt.subplots(10, 10)
k = 0
for i in range(10):
        for j in range(10):
                ax[i][j].imshow(x_train[k].reshape(28, 28),
                                                aspect='autc
                k += 1
plt.show()
```



In [18]:
```
model = Sequential([

        # reshape 28 row * 28 column data to 28*28 rows
        Flatten(input_shape=(28, 28)),

        # dense layer 1
        Dense(256, activation='sigmoid'),

        # dense layer 2
        Dense(128, activation='sigmoid'),

        # output layer
        Dense(10, activation='sigmoid'),
])
```

In [19]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentrop
              metrics=['accuracy'])
```

In [20]:
```
model.fit(x_train, y_train, epochs=10,
          batch_size=2000,
          validation_split=0.2)
```

```
Epoch 1/10
24/24 [==============================] - 2s 67ms/step - los
s: 2.0901 - accuracy: 0.4015 - val_loss: 1.7360 - val_accura
cy: 0.7537
Epoch 2/10
24/24 [==============================] - 1s 56ms/step - los
s: 1.4063 - accuracy: 0.7708 - val_loss: 1.0473 - val_accura
cy: 0.8274
Epoch 3/10
```

```
24/24 [==============================] - 1s 57ms/step - los
s: 0.8672 - accuracy: 0.8348 - val_loss: 0.6673 - val_accura
cy: 0.8652
Epoch 4/10
24/24 [==============================] - 1s 55ms/step - los
s: 0.6014 - accuracy: 0.8658 - val_loss: 0.4934 - val_accura
cy: 0.8881
Epoch 5/10
24/24 [==============================] - 1s 55ms/step - los
s: 0.4715 - accuracy: 0.8881 - val_loss: 0.4053 - val_accura
cy: 0.8987
Epoch 6/10
24/24 [==============================] - 2s 91ms/step - los
s: 0.4005 - accuracy: 0.8990 - val_loss: 0.3553 - val_accura
cy: 0.9083
Epoch 7/10
24/24 [==============================] - 2s 70ms/step - los
s: 0.3563 - accuracy: 0.9060 - val_loss: 0.3224 - val_accura
cy: 0.9132
Epoch 8/10
24/24 [==============================] - 1s 56ms/step - los
s: 0.3261 - accuracy: 0.9120 - val_loss: 0.2978 - val_accura
cy: 0.9183
Epoch 9/10
24/24 [==============================] - 1s 56ms/step - los
s: 0.3021 - accuracy: 0.9171 - val_loss: 0.2791 - val_accura
cy: 0.9225
Epoch 10/10
24/24 [==============================] - 1s 63ms/step - los
s: 0.2841 - accuracy: 0.9202 - val_loss: 0.2646 - val_accura
cy: 0.9251
```

Out[20]:

```
<keras.callbacks.History at 0x7fe53ffab390>
```

In [21]:

```python
results = model.evaluate(x_test, y_test, verbose = 0)
print('test loss, test acc:', results)
```

```
test loss, test acc: [0.27012205123901367, 0.92379999160766
6]
```