In [1]:
```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
```

In [ ]:
```
```

In [3]:
```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [4]:
```
x_train.shape
```

Out[4]:
```
(60000, 28, 28)
```

In [5]:
```
print(x_train.shape)  # Output: (60000, 28, 28)
print(x_train[1].shape)  # Output: (28, 28)
```

```
(60000, 28, 28)
(28, 28)
```

In [24]:
```
x_train[1][1]
```

Out[24]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0], dtype=uint8)
```

In [11]:
```
x_test.shape
```

Out[11]:
```
(10000, 28, 28)
```

In [13]:
```
y_train.shape
```

Out[13]:
```
(60000,)
```

In [15]:
```
y_test.shape
```

Out[15]:
```
(10000,)
```

In [17]:
```
x_train
```

Out[17]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       ...,

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

In [19]:
```
y_train
```

Out[19]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [ ]:

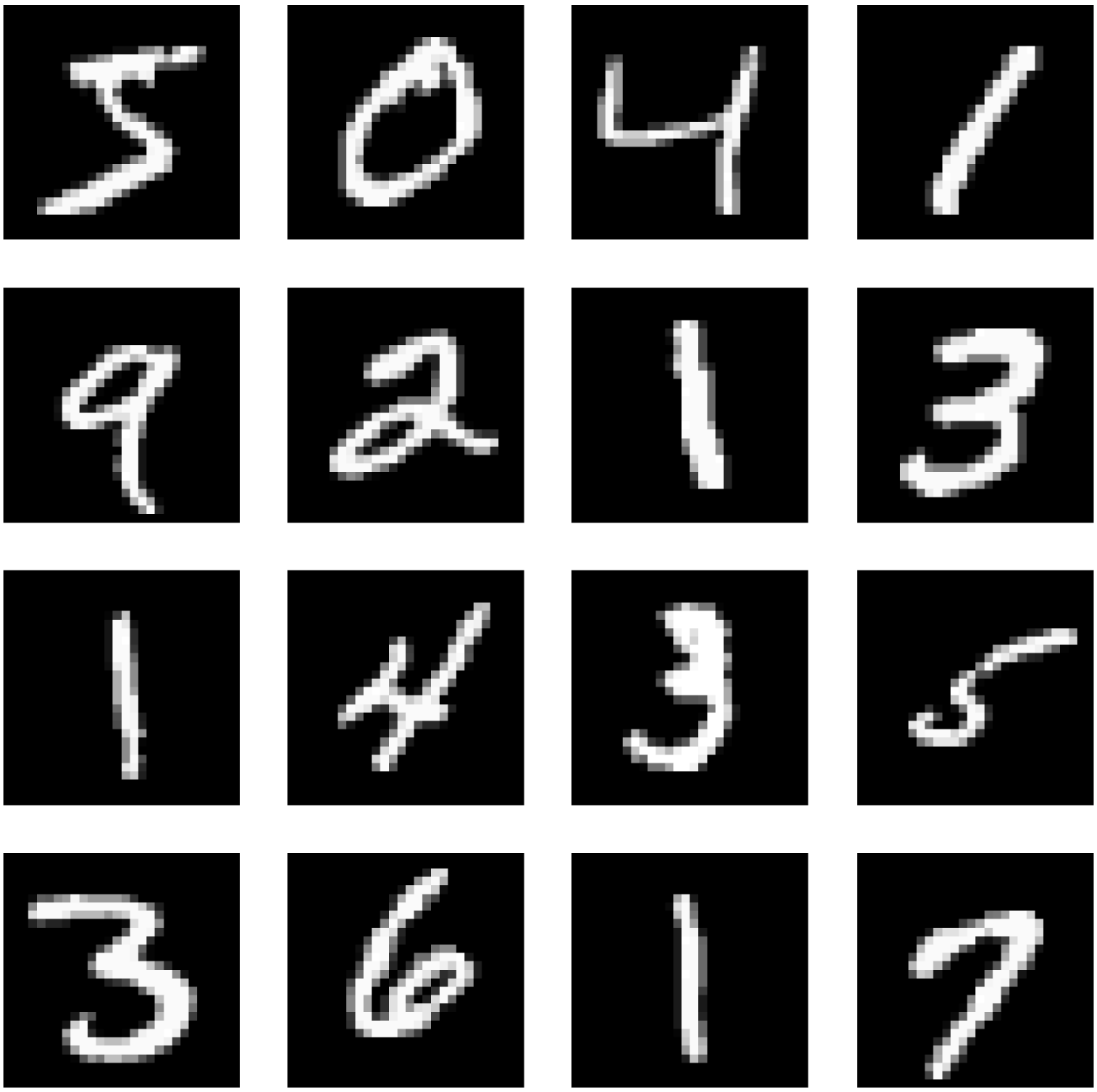
In [28]:
```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

In [23]:
```
x_test
```

Out[23]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]])
```

In [ ]:

In [26]:
```
plt.figure(figsize=(10, 10))
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(x_train[i], cmap='gray')
    plt.axis('off')
plt.show()
```



In [30]:
```
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),  # Flatten the input (28x28 pixels to 784
    layers.Dense(128, activation='relu'),  # Hidden layer with 128 neurons
    layers.Dropout(0.2),                   # Dropout to avoid overfitting
    layers.Dense(10, activation='softmax') # Output layer with 10 neurons (one for
])
```

```
C:\Users\manoj\anaconda3\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When usin
g Sequential models, prefer using an `Input(shape)` object as the first layer in the
model instead.
  super().__init__(**kwargs)
```

In [32]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [40]:
```
from tensorflow.keras.callbacks import EarlyStopping


early_stopping = EarlyStopping(
    monitor='loss',
    patience=3,
    restore_best_weights=True
)
```

In [50]:
```
model.fit(x_train, y_train, epochs=10, callbacks=[early_stopping])
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9850 - loss: 0.0430
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9881 - loss: 0.0352
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9882 - loss: 0.0343
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9894 - loss: 0.0320
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9890 - loss: 0.0332
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9899 - loss: 0.0310
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9903 - loss: 0.0290
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9903 - loss: 0.0279
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9915 - loss: 0.0263
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9916 - loss: 0.0246
```

Out[50]:
```
<keras.src.callbacks.history.History at 0x25a2210e7b0>
```

In [86]:
```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
m=test_acc*100
print("acc:",m)
n=test_loss*1
print("loss:",n)
print(f'\nTest accuracy: {test_acc}')
print('Test loss:',test_loss)
```

```
313/313 - 0s - 1ms/step - accuracy: 0.9810 - loss: 0.0727
acc: 98.1000006198883
loss: 0.07274990528821945

Test accuracy: 0.9810000061988831
Test loss: 0.07274990528821945
```

In [54]:
```
predictions = model.predict(x_test)
predictions
```

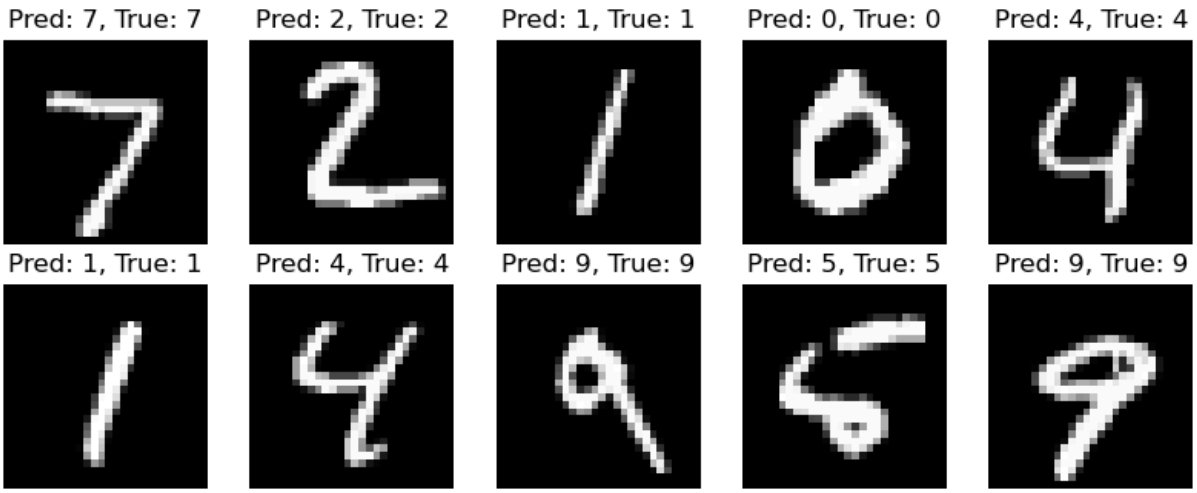
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

Out[54]:
```
array([[2.3203138e-13, 2.0808533e-12, 9.1812702e-10, ..., 9.9999988e-01,
        1.5653027e-13, 1.4834738e-07],
       [1.1324651e-16, 1.0870376e-07, 9.9999988e-01, ..., 8.4168161e-27,
        2.6440588e-18, 4.3050598e-34],
       [2.7530547e-09, 9.9995530e-01, 1.1218782e-05, ..., 3.1289692e-05,
        2.0286211e-06, 2.8583572e-10],
       ...,
       [4.9410997e-20, 2.7247603e-13, 9.7587973e-18, ..., 7.7151346e-10,
        4.5863206e-11, 9.0690338e-07],
       [3.0855443e-15, 1.5373395e-16, 2.6341052e-16, ..., 1.5125477e-15,
        8.8270200e-09, 1.7622676e-14],
       [1.4352687e-13, 8.3868786e-20, 3.5967371e-12, ..., 6.9341770e-23,
        2.5041987e-15, 9.4780588e-19]], dtype=float32)
```

In [92]:
```
predictions[0]
```

Out[92]:
```
array([2.3203138e-13, 2.0808533e-12, 9.1812702e-10, 2.0759975e-08,
       4.2935412e-14, 8.5954112e-14, 1.7034667e-22, 9.9999988e-01,
       1.5653027e-13, 1.4834738e-07], dtype=float32)
```

In [56]:
```
plt.figure(figsize=(10, 10))
for i in range(10):
    plt.subplot(5, 5, i+1)
    plt.imshow(x_test[i], cmap='gray')
    plt.title(f"Pred: {predictions[i].argmax()}, True: {y_test[i]}")
    plt.axis('off')
plt.show()
```



In [ ]: