Final code

Train the set :

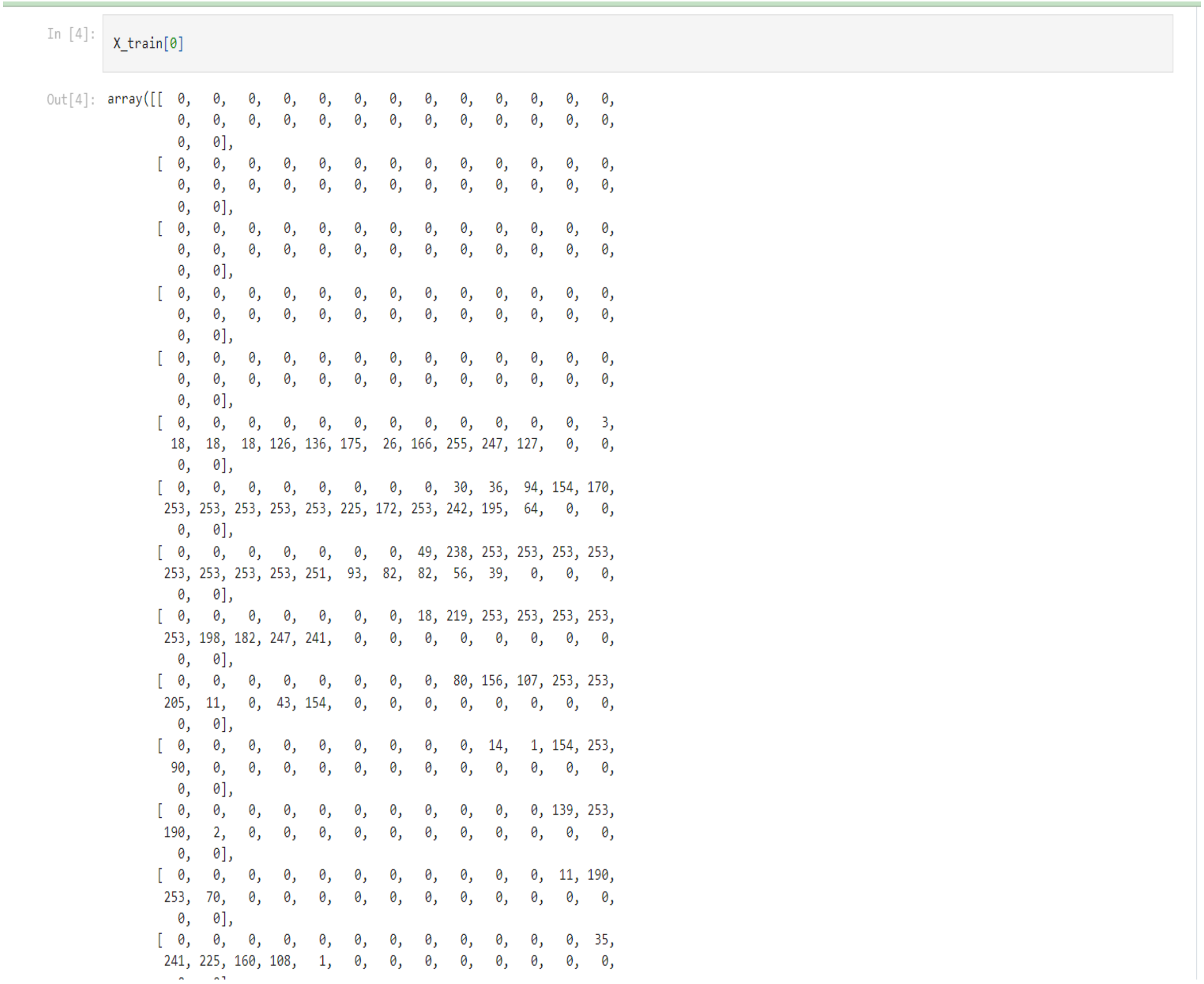
In [1]:
```
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.datasets import mnist
```

In [2]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

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

In [3]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
```



```
In [4]: X_train[0]
```

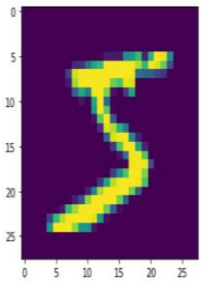
```
Out[4]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0,   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,   3,
                 18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
                253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
                253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
                205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
                 90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
                190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
                253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  0,   0],
               [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
                241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
```

In [5]:
```
y_train[0]
```

Out[5]: 5

In [6]:
```
plt.imshow(X_train[0])
```

Out[6]:



In [7]:
```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

encoding

In [8]:
```
number_of_classes = 10
Y_train = np_utils.to_categorical(y_train, number_of_classes)
Y_test = np_utils.to_categorical(y_test, number_of_classes)
```

In [9]:
```
Y_train[0]
```

Out[9]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

Code.css

```
@import
url("https://fonts.googleapis.com/css2?family=Overpass:wght@200;300;400;500;600;
700;900&display=swap");
* {
  padding: 0;
  margin: 0;
}

body {
  color: black;
  font-family: "Overpass", sans-serif;
}
```

```
.container {
  width: 100%;
  height: 100%;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  background-color: white;
}
.container .heading {
  margin-top: -2rem;
  padding-bottom: 2rem;
  width: fit-content;
  text-align: center;
}
.container .heading h1 {
  font-size: 3rem;
  font-weight: 550;
}
.container .heading h2 {
  font-size: 1rem;
  color: rgb(90, 88, 88);
}
.container .sub_container1 {
  box-shadow: 0 0 20px rgb(172, 170, 170);
  width: 40rem;
  height: 25rem;
  padding: 1.5rem;
}
.container .sub_container2 {
  background-color: rgba(190, 190, 190, 0.5);
  width: 100%;
  height: 100%;
  display: flex;
```

```
  border: 1px dashed black;
  justify-content: center;
  align-items: center;
}
.container .sub_container2 .upload {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 8rem;
  height: -webkit-fit-content;
  height: -moz-fit-content;
  height: fit-content;

  border-radius: 6px;
  color: white;
  background-color: rgb(114, 96, 182);
  box-shadow: 0 5px 10px rgb(146, 135, 247);
}

.container .sub_container2 #loading {
  display: none;
  justify-content: center;
  align-items: center;
  width: 10rem;
  height: auto;
  position: absolute;
}

.container .sub_container2 .upload label {
  font-size: 1rem;
  font-weight: 600;
  color: white;
  height: 100%;
  width: 100%;
```

```
  padding: 10px;
  display: block;
}
.container .sub_container2 .upload svg
{
  height: 15px;
  width: auto;
  padding-right: 8px;
  margin-bottom: -2px;
}



@media screen and (max-width:700px) {
  .container .sub_container1 {
    height: 20rem;
    width: 18rem;
    margin-top: 3.5rem;
    margin-bottom: -8rem;

  }
  .container .heading h1 {
    margin-top: -6rem;
    font-size: 2rem;
    padding-bottom:1rem ;

  }
}
```

Code.html

```
<html>
    <head>
        <title>Handwritten Digit Recognizer</title>
        <link rel="stylesheet" href="{{url_for('static',filename='css/main.css')}}">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://unpkg.com/feather-icons"></script>
```

```
    </head>
    <body>
        <div class="container">
            <div class="heading">
                <h1>Handwritten Digit Recognizer</h1>
                <h2>Easy analyze and detect handwritten digits</h2>
            </div>
            <div class="sub_container1">
                <div class="sub_container2">
                    <form class="upload" action="/confirm" method="post"
enctype="multipart/form-data">
                        <!-- select -->
                        <label id="label" for="upload-image"><i data-feather="file-
plus"></i>Select File</label>
                        <!-- upload -->
                        <input type="file" name="photo" id="upload-image" hidden>
                    </form>
                    <img id="loading"
src="{{url_for('static',filename='images/loading.gif')}}">
                </div>
            </div>
        </div>

        <script rel="text/javascript">
            feather.replace();

            form = document.querySelector(".upload")
            label = document.querySelector('#label');
            loading = document.querySelector("#loading")
            select = document.querySelector("#upload-image");

            console.log("working...")
            select.addEventListener('change', (e) => {
                e.preventDefault();
```

```
            form.style.visibility = "hidden";
            form.submit()
            loading.style.display = 'flex';
        })
    </script>
  </body>
</html>
```

Testing:

In [11]:
```
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["accuracy"])
```

In [12]:
```
model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 205s 109ms/step - loss: 0.2061 - accuracy: 0.9526 - val_loss: 0.0944 - val_accuracy: 0.9723
Epoch 2/5
1875/1875 [==============================] - 195s 104ms/step - loss: 0.0678 - accuracy: 0.9795 - val_loss: 0.0847 - val_accuracy: 0.9760
Epoch 3/5
1875/1875 [==============================] - 196s 105ms/step - loss: 0.0486 - accuracy: 0.9846 - val_loss: 0.1049 - val_accuracy: 0.9760
Epoch 4/5
1875/1875 [==============================] - 194s 103ms/step - loss: 0.0408 - accuracy: 0.9877 - val_loss: 0.0927 - val_accuracy: 0.9753
Epoch 5/5
1875/1875 [==============================] - 196s 105ms/step - loss: 0.0275 - accuracy: 0.9919 - val_loss: 0.1179 - val_accuracy: 0.9774
```

Out[12]:

In [13]:
```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.11792317777872086, 0.977400004863739]
```

In [14]:
```
prediction = model.predict(X_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 114ms/step
[[1.7261184e-12 2.6062123e-21 2.6803178e-11 1.2729900e-09 1.5024680e-21
  9.7608481e-19 3.2893840e-23 1.0000000e+00 9.7549820e-16 2.1773126e-15]
 [2.9485101e-08 1.7420459e-07 9.9999952e-01 1.1130676e-08 1.0805351e-16
  4.4104544e-15 2.8853614e-07 1.2159911e-16 1.0800677e-08 5.6359541e-17]
 [3.3493230e-13 9.9999988e-01 4.0103846e-08 5.6979606e-15 2.9670151e-08
  4.9337485e-09 8.7734236e-13 2.3520821e-11 6.5292514e-09 2.2166921e-16]
 [1.0000000e+00 6.9765886e-19 7.2367817e-13 4.5614911e-17 2.4791712e-12
  1.6644009e-13 6.4797261e-09 6.5001277e-16 2.5072968e-14 1.2465277e-11]]
```

In [15]:

```
print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```