```
<html>


<head>
  <title>Digit Recognition WebApp</title>


  <meta name="viewport" content="width=device-width">
  <!-- GoogleFont -->
  <link href="https://fonts.googleapis.com/css2?family=Prompt:wght@600&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Varela+Round&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@500&display=swap"
rel="stylesheet">
  <link href="https://fonts.googleapis.com/css?family=Calistoga|Josefin+Sans:400,700|Pacifico&display=swap"
rel="stylesheet">
  <!-- bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
  <link rel="stylesheet" type= "text/css" href= "style.css">
  <!-- fontawesome -->
  <script src="https://kit.fontawesome.com/b3aed9cb07.js" crossorigin="anonymous"></script>


  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>


</head>


<script>
  function preview() {
    frame.src=URL.createObjectURL(event.target.files[0]);
}
```

```
    $(document).ready(function() {
        $('#clear_button').on('click', function() {
            $('#image').val('');
            $('#frame').attr('src',"");
          });
       });

</script>

<body>

  <h1 class="welcome">IBM PROJECT
  </h1>
  <section id="title">
    <h4 class="heading">Handwritten Digit Recognition </h4>
    <br><br>
     <p><h2>
       The website is designed to predict the handwritten digit.</h2>
     </p>



  </section>

  <section id="content">


       <div class="leftside">
       <form action="/predict" method="POST" enctype="multipart/form-data">
       <label>Select a image:</label>
       <input id="image" type="file" name="image" accept="image/png, image/jpeg"
onchange="preview()"><br><br>
        <img id="frame" src="" width="100px" height="100px"/>
        <div class="buttons_div">
          <button type="submit" class="btn btn-dark" id="predict_button" onclick="predict.html">Predict</button>
          <button type="button" class="btn btn-dark" id="clear_button">  Clear  </button>
```

```
        </div>
      </form>
      </div>
  </section>

</body>

</html>
```

```
"nbformat_minor": 0,
"metadata": {
  "colab": {
    "provenance": []
  },
  "kernelspec": {
    "name": "python3",
    "display_name": "Python 3"
  },
  "language_info": {
    "name": "python"
  }
},
"cells": [
  {
    "cell_type": "code",
    "source": [
      "import cv2\n",
      "import numpy as np\n",
      "from keras.datasets import mnist\n",
      "from keras.layers import Dense, Flatten, MaxPooling2D, Dropout\n",
      "from keras.layers.convolutional import Conv2D\n",
      "from keras.models import Sequential\n",
      "from tensorflow.keras.utils import to_categorical\n",
      "import matplotlib.pyplot as plt"
    ],
    "metadata": {
      "id": "yJs2eLRLOSHM"
    },
    "execution_count": 2,
    "outputs": []
  },
  {
```

```
 "cell_type": "code",
 "source": [
  "(X_train, y_train), (X_test, y_test) = mnist.load_data()"
 ],
 "metadata": {
  "id": "-NoTriNEPBWl",
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "outputId": "1ceca63e-26d0-4a6a-b2c0-0cd952d515f0"
 },
 "execution_count": 3,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz\n",
    "11490434/11490434 [==============================] - 0s 0us/step\n"
   ]
  }
 ]
},
{
 "cell_type": "code",
 "source": [
  "plt.imshow(X_train[0], cmap=\"gray\")\n",
  "plt.show()\n",
  "print (y_train[0])"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 282
```

},
"id": "s_JYjDk2PGwL",
"outputId": "d6cc49b8-d286-4bed-97ba-c5889303f296"
},
"execution_count": 4,
"outputs": [
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxI
B0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA6Ly9tYXRwbG90bGliLm9y
Zy+WH4yJAAAN9klEQVR4nO3df4xV9ZnH8c+zWP6QojBrOhKKSyEGg8ZON4gbl6w1hvojGhw1TSexoZE4/YNJaLIhNe
wf1WwwZBU2SzTNTKMWMWNl1qEzGhaGhGdzMKQgGfg0gbl6w1TSexoZE4/YNJaLIhNe
wf1WwwZBU2SzTNTKMWMNl1qEzUgaQouoOzGhDgiKo5LdQ2mTEaowZEf/mCHefaPezBTnfu9w7nn3nOZ5/1Kbu69
57nnnicnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAuaubCzIxD/0DB3N2Gm17Xlt3M7jSz/zazd8zskXrmBaBYVut5djObIel/JK2QdFTSG5K63P33iWnYsgMFK2LLvkzSO+7+nrt/LukXklbWMT8ABaon7Asl/WHI46PZtAuYWbeZ7TWzvXUsC0CdCj9A5+49knokduOBMtWzZT8maenQ4wuzaQBaqD1hf0PSYjP7upl9SdL3JA00pi0AjVbzbry7f2FmPZJ2ijpDubbR+tYdHsPmv71qQ56fK1+2/ueQL//kd3cmZaXJtXxTTOkPH+Pl0pQ1JWd+WbFnOmkg9wrxHRlLM9E5v7+iaoVKybCBI4k4OrSFQSGzHGuU6xbeSm2djAT1mnFRDGy2kTd0cDm5J/uZsoqk8zlY2KyZpKtIppPnWD6iv/mIo+fZVpWxVfM1DWCLkbB2ZdXLjvVGBbV8LmaPNSbgfUcVzzlp66n4Hy0e54rg9hgI35h9rrfvNS3v+/WA++6jIJeAWBh7KJ4CcWTbWmXjUV8Kgp6EzArhLLErsOTNcm2lDs9nwIKkpygFOxhg2Ws8jPsOeltHTkY5SsToNESZhFuBPgLYaAsDFD+rq+o1zDTD4kSFRoBfBRPRWNk9nXAhPNHEPmvJYf5REszcmHh/1R/kvh3hRxUl9lNNFH3Glph2QVpbA=="

nNbK6k70naJ6nd3c/9OPlDSe1V5umW1J2/RQBFmPDZeDP7pqRnJf3M3U+MrXlldMhxB2109z53X+Tui+rqFEBdJhR2
M/uGKkH/jbs/l00+amazsvosScca0yKAItTcjTczk/SkpHfcfcOY0jZJyyWty563NqTDSaC9fdwjnC8tXLgwWX/88ceT9auu
uuq8eyrKvn37kvVHH320am3r1vQ/GW5RLdZEjtn/XtKPJb1lZgeyaWtUCfnvzGyFpA8k/bAxLQIoQs2wu/t/Sxp3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMEtrhPU1tZWtdbb25uct6Ojl1mfN29erp6K8MorryTr69evT9Z37NiRrH/2
2Wfn3RMagy07EARhB4Ig7EAQhB0IgrADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHHV2uzZs3P1VJRPP/20am3jxo3
JeR955JFk/fTp07l6Quthyw4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQYS5zt7Z2VlXvR57d+9PGZZLbmkvPnf/NzN7SNIDkv6cfXSNu/++++xxxUmzAwPJ+vbt25P1P1z1z/nw8
HByXsTBlh0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgjB3T3/Abl6I6kzZLaJbmkPnf/NzN7SNIDkv6cfXSNu/++xnelFwagbu4+
7qjLEwn7LEmz3H2/mU2X9Jqku1QZj/2Uuz820SYIO9B41cI+kfHZhyQNZa9Pmtk7ksr90ywAztt5HbOb2VxJ35O0L5vU
Y2ZvmtlTZjazyjzdZtZvZv11dQqgLjV347/8oNk3Jb0saa27P2dm7ZI+UuU4/p9V2dW/v8Z3sBsPNFjuY3ZJMrNvSNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCclvTM26NJR2st0kAjTORs/FLJP2XpLckjWaT10jqktShym78YUk/
zU7mpb6LLTvQYHXtxheFsAOPNl3s3HsDkQNiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBC
EHQii2UM2fyTpgzHvL8umtaW7a1V+5Lola8ie/ubaoWm3s/+tYWb9bv7otIaSGjV3lq1L4ne8mpWb+zGA0EQdiCIssP
eV/LyU1q1t1bbS6vt9tr1btS6K3vJrSW6nH7ACap4xuuvAEKWE3cxuNbNDZvaemT1T1YRvV8vdsvdtuemLvKzjuAtzOMdWtmZnvV5WymzGzRdbtOk4hkDWKjU3DNbP2dm7ZI+UuU4/p9V2dW/v8Z3sBsPNFjuY3ZJMrNvSNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCclvTM26NmJu3M6JR2st0kAjTORs/FLJP2XpLckjWaT10jqktShym78YUk/
zU7mpb6LLTvQYHXtxheFsAONl3s3HsDkQNiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBC
EHQii2UM2fyTpgzHvL8umtaJW7a1V+5LoLa8ie/ubaoWm3s/+tYWb9bv7otIaSGjV3lq1L4ne8mpWb+zGA0EQdiCIssP
eV/LyU1q1t1btS6K3vJrSW6nH7ACap4+wtO4AmIexAEKWE3cxuNbNDZvaemT1YRg/VmNlhM3vLzA6UPT5dNobeMT
M7OGZam5m9aGbvZs/jjjHc4upmUrT1Z1cGHm3kpAMF0Ppf3JTpNqJBSbwIPJKb8DZLVPI8aTn0a9Px0yVbFyz1SVj/0oFbs7O7Mkmkv5+fkLKz5p1Pn8SfJUvKLSCdM1BwBQWWdL4iOpndUD5Ve
5JeldTl7gNNbaQKMzssaZG7G7l/4DDDP7B0mnJG0+N7SWmf2LpOPuvi77j3j3Kmu/+8RXp7SOc5jHeDeqs2zPhPVOK6K3L
48zzK2LIvlvSeu7/v7mck/VbSshL6aHnuvlfS8a9MXiZpa/Y/yZyf7/LuQLNnP/MOQUJQ0nU2Uf91gmvHLG2dX3iT5IucswfWCtdO
/2lpPmqjAE4JGl9mc1kw4w/K+ln7n5ibK3MdTX01Zb2WEfVDSnDHvv51NawnuPpg9H5P0vCqHHa3k6LkRdLPnYy
X38yV3P+ruZ919VNKvVOK6y4YZf1bSb9z9z9uWxy6etuvL6atd7KCPurkq40s++Y2VRJP5K0rYQ+vsbMzLG
QK03FPU2Scuz18slbS2xl7/QKsN4VxtmXCWvu9KHP3f3pj8k3a7KGfn/lfRPZfRQpa95kv43e/w1q3kl7k1mCKuc2Vkj
6a0m7JL0r6T8ltbVQb/+uytDeb6oSrFkl9bZElV30NyUdB63l73uEn01Zb3xc1kgCE7QAUEQdiCIAwg4EQdiBIAg7EARhB
4Ig7EAQ/w8ie3GmjcGk5QAAAABJRU5ErkJggg==\n"
},
"metadata": {
"needs_background": "light"
}
},
{
"output_type": "stream",
"name": "stdout",
"text": [
"5\n"
]
}
]
},
{
"cell_type": "code",
"source": [
"print (\"Shape of X_train: {}\".format(X_train.shape))\n",
"print (\"Shape of y_train: {}\".format(y_train.shape))\n",

```
    "print (\"Shape of X_test: {}\".format(X_test.shape))\n",
    "print (\"Shape of y_test: {}\".format(y_test.shape))"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "Us6HotvxPPco",
    "outputId": "dddeb261-7494-47d3-f5c1-81f302ed3d69"
   },
   "execution_count": 5,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Shape of X_train: (60000, 28, 28)\n",
      "Shape of y_train: (60000,)\n",
      "Shape of X_test: (10000, 28, 28)\n",
      "Shape of y_test: (10000,)\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "# Reshaping so as to convert images for our model\n",
    "X_train = X_train.reshape(60000, 28, 28, 1)\n",
    "X_test = X_test.reshape(10000, 28, 28, 1)"
   ],
   "metadata": {
    "id": "n962FFkFPUzH"
   },
```

```
 "execution_count": 6,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "print (\"Shape of X_train: {}\".format(X_train.shape))\n",
  "print (\"Shape of y_train: {}\".format(y_train.shape))\n",
  "print (\"Shape of X_test: {}\".format(X_test.shape))\n",
  "print (\"Shape of y_test: {}\".format(y_test.shape))"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "KZEPs_m5PZac",
  "outputId": "36b9a4ca-c313-422d-f84a-28b461b01dcd"
 },
 "execution_count": 7,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Shape of X_train: (60000, 28, 28, 1)\n",
    "Shape of y_train: (60000,)\n",
    "Shape of X_test: (10000, 28, 28, 1)\n",
    "Shape of y_test: (10000,)\n"
   ]
  }
 ]
},
{
 "cell_type": "code",
```

```
  "source": [
    "#one hot encoding\n",
    "y_train = to_categorical(y_train)\n",
    "y_test = to_categorical(y_test)"
  ],
  "metadata": {
    "id": "BKtLDY0VQNkU"
  },
  "execution_count": 8,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "model = Sequential()\n",
    "\n",
    "## Declare the layers\n",
    "layer_1 = Conv2D(64, kernel_size=3, activation='relu', input_shape=(28, 28, 1))\n",
    "layer_2 = MaxPooling2D(pool_size=2)\n",
    "layer_3 = Conv2D(32, kernel_size=3, activation='relu')\n",
    "layer_4 = MaxPooling2D(pool_size=2)\n",
    "layer_5 = Dropout(0.5)\n",
    "layer_6 = Flatten()\n",
    "layer_7 = Dense(128, activation=\"relu\")\n",
    "layer_8 = Dropout(0.5)\n",
    "layer_9 = Dense(10, activation='softmax')\n",
    "\n",
    "## Add the layers to the model\n",
    "model.add(layer_1)\n",
    "model.add(layer_2)\n",
    "model.add(layer_3)\n",
    "model.add(layer_4)\n",
    "model.add(layer_5)\n",
    "model.add(layer_6)\n",
```

```
    "model.add(layer_7)\n",
    "model.add(layer_8)\n",
    "model.add(layer_9)"
   ],
   "metadata": {
    "id": "-afmGNuHCWdh"
   },
   "execution_count": 9,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])"
   ],
   "metadata": {
    "id": "xLJShJAHDLhe"
   },
   "execution_count": 10,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "roVOfV9EDRi9",
    "outputId": "22787483-28d2-41ae-9cdf-666f95b41f6a"
   },
   "execution_count": 11,
```

```
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "Epoch 1/3\n",
       "1875/1875 [==============================] - 58s 31ms/step - loss: 0.8654 - accuracy: 0.7801 - val_loss: 0.1307 - val_accuracy: 0.9630\n",
       "Epoch 2/3\n",
       "1875/1875 [==============================] - 58s 31ms/step - loss: 0.2703 - accuracy: 0.9201 - val_loss: 0.0750 - val_accuracy: 0.9757\n",
       "Epoch 3/3\n",
       "1875/1875 [==============================] - 56s 30ms/step - loss: 0.2055 - accuracy: 0.9385 - val_loss: 0.0746 - val_accuracy: 0.9772\n"
      ]
     },
     {
      "output_type": "execute_result",
      "data": {
       "text/plain": [
        "<keras.callbacks.History at 0x7fc1e21cc510>"
       ]
      },
      "metadata": {},
      "execution_count": 11
     }
    ]
   },
   {
    "cell_type": "code",
    "source": [
     "example = X_train[1]\n",
     "prediction = model.predict(example.reshape(1, 28, 28, 1))\n",
     "print (\"Prediction (Softmax) from the neural network:\\n\\n {}\".format(prediction))\n",
     "hard_maxed_prediction = np.zeros(prediction.shape)\n",
```

```
  "hard_maxed_prediction[0][np.argmax(prediction)] = 1\n",
  "print (\"\\n\\nHard-maxed form of the prediction: \\n\\n {}\".format(hard_maxed_prediction))\n",
  "\n",
  "print (\"\\n\\n--------- Prediction --------- \\n\\n\")\n",
  "plt.imshow(example.reshape(28, 28), cmap=\"gray\")\n",
  "plt.show()\n",
  "print(\"\\n\\nFinal Output: {}\".format(np.argmax(prediction)))"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 595
  },
  "id": "styJoT_uDf6D",
  "outputId": "87aecfd4-45cd-441b-c184-0e255b5dc44d"
 },
 "execution_count": 12,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "1/1 [==============================] - 0s 83ms/step\n",
    "Prediction (Softmax) from the neural network:\n",
    "\n",
    " [[9.99999881e-01 7.21094625e-13 7.90088137e-08 3.49195464e-11\n",
    "  1.54954244e-11 5.48896974e-13 1.05098525e-08 1.00683108e-10\n",
    "  7.00186797e-10 1.28125794e-08]]\n",
    "\n",
    "\n",
    "Hard-maxed form of the prediction: \n",
    "\n",
    " [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n",
    "\n",
```

"\n",
"--------- Prediction --------- \n",
"\n",
"\n"
]
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxI
B0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA6Ly9tYXRwbG90bGliLm9y
Zy+WH4yJAAAOF0lEQVR4nO3dcYxV5ZnH8d8jW4xKIagpTkRr2+AfzUYHQUKyprI2bVw0gcakQzDpk2GxJJQszGr3VF
IamNjlEZNJE6VFFcqqGjBpi51GaLdmDSOyCpqW1mDFhwZUSNDTKTCs3/cQzPinPcM9557z4z4Hn+0km997zzLn38TI/z7
nnPfe85u4CcPI7peoGAHQGYQeCIOxAEIQdCIKwA0H8QydfMw49A+0mbvbWMtb2rKb2ZVm928zzz22VmN7fyXADa
y5odZzezCZL+Iuk7kvZIelHSYnd/PbEOW3agzdqNm2cSX8d9XhPtuxzzzKzZXzAbNbLCF
1wLQorYfoHP3fkn9ErvxQJVa2bLvlXTeqMfTs2UAaaqiVsL8oaYaZfc3c3MBKlVaGrDWGwjFFvGNUTsrRsFUAaqiVsL8oaYaZfc3c3MJkpaJGlzOW0BKFvTu/Hu/Hu/pmZLZO0RdlEfSWvc/
bXSOgNQQqaH3pp6MT6zA23XlpNqAJw4CDsQBGEHgiDsQBCEHQiCsANBEHYgSAIOxAEYQeCIOxAEIQdCdCIKwA0EQdiAIpmw+CTzwwAO5tcWF3ewk+NTdA7ApEmkTvXXnnns
uWZ83b15u7aKLLkququezJiyw4EQdiBIAg7EAADRhB4Ig7EAQhB0IgrADQhB0IgrADQhB0IgrADQhB0IgrADQTDOfgKYNNWsn7VVVfl1
vuuuu3Nq7776bXPfll19O1/j/66KNk/Yorrsftfq+nIjYsgNBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIAg7EAADRhB4Ig7EAADRhB4Ig7EAQhB0IgrADQhB0IgrADQtt
WR98uTJb/2M888888k6wXFb/+8ssvT+7Xaw7dNzjTXZ+qd6FbkrGY7Ggl0tVfbx4WpXT/Ttb
Ry0708yeNbM3s3708+aQx+yfNbmM3s9qu8/GYj3M3R3s9upZTYLoHzj2Y3/laQrj1l2s6St7j5D0tbmsMYAaKwy7uz8v6r9Q
n99yRNy/tFM+uV6Nvk6wAoCtHHSrdxq8LCtOP7zN8bpf+AHK+f0kKC2ZjBJKsP1o9bsK8Vs93xpJAaCkLn4kq+d/tpe
xdjN7VNI8SWdL2ia3TB2idpaTfSHPzRcaSfZCCPk3W6LhcbRzlLe8UqY+yz1eKYTv6...... (partial read guard)

mavmdnybPmZZvasmb2Z3U5tf7sAmjWe3fjPJP2bu39T0lxJPzKzb0q6WdJWd58haWv2GEBNFYbd3YfcfXt2f0TSG5LO
lbRA0trs19ZKSp8TCqBSx/WZ3cwukDRT0h8lTXP3o5OUvSdpWs46vZJ6m28RQBnGfTTezCZJ2ijpx+5+YHTNG0eoxjxK
5e797j7b3We31CmAlowr7Gb2JTWCvs7dn8wW7zOzrqzeJWm4PS0CKEPhbrw1vr/5kKQ33H3VqNJmSUsk/Ty7TV/X
N7AJEyYk6zfccEOyXnRJ5AMHDuTWZsyYkVy3VS+88EKyvm3bttzabbfdVnY7SBjPZ/Z/knS9pFfNbEe27CdqhPwxM/uh
pLclfb89LQIoQ2HY3f1/JOVdneHb5bYDoF04XRYIgrADQRB2IAjCDgRB2IEgrOjrmaW+mFnnXqxkqa9yPv7448l1L7300
pZeu+hS1a38G6a+HitJ69evT9ZP5Mtgn6zcfcw/GLbsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAE4+PwI+wl6OrStaXLl2arP
f19SXrrYyz33PPPcl1V69enazv2rUrWUf9MM4OBEfYgSAIOxAEIQdCIKwA0EwdiCIwgtOlmy/pLdHPT47W1ZHde2trn1J
9NasMnv7al6ho99n/8KLmw26++zKGkioa2917Uuit2Z1qjd244EgCDsQRNVh76/49VPq2ltd+5LorVkd6a3Sz+wAOqfq
LTuADiHsQBCVhN3MrjSzP5vZLjO7uYuGutL56bI59IbNbOeoZWea2bNm9mZ2O+YcexX1VotpvBPTjFf
63lU9/XnHP7Ob2QRJf5H0HUl7JL0wW5Js9298hMwzOxbkg5Kevjo1FpmdqekD93959n/KKe6+7/Xp
LeVOs5pvNvUW9404/+qCt+7Mqc/b0YVW/Y5kna5+1vufkjSekkLKuij9tz9eUkfHrN4gaS12f21avyxdFxOb7Xg7kPuvj2
7PyLp6DTjlb53ib46ooqwnyvpr6Me71G95nt3Sb83s5fMrLfqZsYwbdQ0W+9JmlZlM2MonMa7k46ZZrw2710z05+3ig
N0X3SZu18i6V8k/SjXa0lb3wGq9PY6WpJ31BjDsAhSXdX2Uw2zfhGST929wOja1W+d2P01ZH3rYqw75V03qjH07NIt
eDue7PbYUlPqfGxo072HZ1BN7sdrrifv3P3fe5+2N2PSPqlKnzvsmnGN0pa5+5PZosrf+/G6qtT71sVYX9R0gwz+5qZTZS
0SNLmCvr4AjM7IztwIjM7Q9J3Vb+pqDdLWpLdXyJpU4U4W9fE5dpvHOm2ZcFb93lU9/7u4d/5E0X40j8v8n6T+q6CGnr6
9L+t/s57Wqe5P0qBq7dX9T49jGDyWdJWmrpDcX9T49jGDyWdJWmrpDcX9T49jGDyWdJWmrpDcX9T49jGDyWdJWmrpDcX9T
wQBAfogCAIOxAEYQdCIOxAEIQdiCI/wcI826NkY1TiQAAAABJRU5ErkJggg==\n"

```
      },
      "metadata": {
        "needs_background": "light"
      }
    },
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "\n",
        "\n",
        "Final Output: 0\n"
      ]
    }
  ]
 },
 {
  "cell_type": "code",
  "source": [
```

```
    "metrices=model.evaluate(X_test,y_test,verbose=0)\n",
    "print(\"Metrices(test loss and Test Accuracy):\")\n",
    "print(metrices)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "O05hxMgFFwBf",
    "outputId": "8711908d-bda8-4332-bb23-4a82cbf40acc"
   },
   "execution_count": 13,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Metrices(test loss and Test Accuracy):\n",
      "[0.07461030036211014, 0.9771999716758728]\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "image = cv2.imread('test_image.jpg')\n",
    "image = np.full((100,80,3), 12, dtype = np.uint8)\n",
    "grey = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)\n",
    "ret, thresh = cv2.threshold(grey.copy(), 75, 255, cv2.THRESH_BINARY_INV)\n",
    "contours,hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)\n",
    "preprocessed_digits = []\n",
    "\n",
```

```
      "for c in contours:\n",
      "    x,y,w,h = cv2.boundingRect(c)\n",
      "    \n",
      "    # Creating a rectangle around the digit in the original image (for displaying the digits fetched via contours)\n",
      "    cv2.rectangle(image, (x,y), (x+w, y+h), color=(0, 255, 0), thickness=2)\n",
      "    \n",
      "    # Cropping out the digit from the image corresponding to the current contours in the for loop\n",
      "    digit = thresh[y:y+h, x:x+w]\n",
      "    \n",
      "    # Resizing that digit to (18, 18)\n",
      "    resized_digit = cv2.resize(digit, (18,18))\n",
      "    \n",
      "    # Padding the digit with 5 pixels of black color (zeros) in each side to finally produce the image of (28, 28)\n",
      "    padded_digit = np.pad(resized_digit, ((5,5),(5,5)), \"constant\", constant_values=0)\n",
      "    \n",
      "    # Adding the preprocessed digit to the list of preprocessed digits\n",
      "    preprocessed_digits.append(padded_digit)\n",
      "\n",
      "print(\"\\n\\n\\n----------Contoured Image----------\")\n",
      "import os, types\n",
      "import pandas as pd\n",
      "\n",
      "def __iter__(self): return 0\n",
      "\n",
      "print=(\"\\n\\n\\n----------Contoured Image----------\")\n",
      "plt.imshow(image, cmap=\"gray\")\n",
      "plt.show()\n",
      "    \n",
      "inp = np.array(preprocessed_digits)\n"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
```

"height": 337
},
"id": "104DE2lDGko0",
"outputId": "a175187c-999e-4d57-86fc-2d78dbda46f4"
},
"execution_count": 16,
"outputs": [
{
"output_type": "stream",
"name": "stdout",
"text": [
"\n",
"\n",
"\n",
"----------------Contoured Image--------------------\n"
]
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAM8AAAD7CAYAAADNasDkAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACx
IB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA6Ly9tYXRwbG90bGliLm9
yZy+WH4yJAAAKyklEQVR4nO3dX4ilhXnH8e/PnTVptI1/sizbXVu3KEmXQmoYrGIpRQ1YG6IpRQ1YG6IXIqahXYqwN2lrmkCi7UV
uK5QYL0pg0YalSKPdSBUJCXZjLlro1jFKUndj3JoaV1ZdQZuSm+5kn17Ma5luZzNnn5kz5536/cAw5/1z5n142e+e95x5Y
VJVSDp35816AGmzMh6pyXikJuORmjIeqWlN8SS5KckLUN8SS5KckLSY4luXu9hpI2g3R/z5NkC/AD4KPAceBp4BNVdWT9xpxP
Ga24Nz70aOFZVLwEk+SpwC3DWePKBFJev4YjSLDzDm1yczVa4lnJ/DKsuXjwG+cuVOSfcA+AH4JWFjDEaVZCC+vt
HrqHxhU1f6qmq+qef5Pu9LmtZZXnleBy7+/Wf5WTaaAReCna5hAmpbzgK1Azr7LWuJ5GrgyyW6W4c4/
MseWft6xhBGk6Tv/qaU79wSm48Oz7tOOpqsUkfwR8E9gC/HVVT/xD/gpbPmnLWz9m63dEaSpWbxxkVO3TykegK
r6OvD1tfwMabPyDgOpyXikJuORmjIeqWcl4pCbjkZqMR2oyHqnJeKKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pC
bjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GaVo0nyWVJnkpyJMn
zSe4a1l+S5S5Mkkw7fL57+uNJ4TPLKswh8tqr2ANcAn0qyB9gHHKyqK4GDw7QkYdwr4qLBwrmBgBRBYCdwC3ChWrbBDCEzVNxXUlQwMB4cBiQ
OFqyLASwxr6oTw6bHTe6dAePc7Au+2bEDOSDU3Xgx8IsxD0xiL6OP
e0meG9b9GfAXwCNJ7gReBm6fzojSOK0aT1X9Q5I5CzbL5hfceRNg/vMJCajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqM
R2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajI
eqcl4pCbjkZqMR2o6lz8lvyXJs0meGJZ3Jzmc5FiSh5OcP70xpfE5l1eeu4AXli1/EfhyVV0BvA7cuZ6DSWM3UTxJdgG/BzwwLAe4HjgwbPIg8LFpDCiN1aSvPF8B7gZ+PixfCpysqlPD8ivAzpWemGRvksUkiy+99tqahpXGZNV4knwEOF5Vz3QOUFX7q2q+quZ3XXJJ50dIozTJ5/ZcB3w0yc3Ae4FfAe4HLkoyN7z67AJenN6Y0vis+spTVfdU1a6quhy4A/hmVX0SeAq4ddhtL/D41KaURmgtv+fZB/xJkmMsvQd6cH1GkjaHc/pI9Kr6Z+Cfh8c/Aq5e/5GkzcE7DKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikponiSXJRkgNJfpjkhSTXJrkkyZNJjg7fL572sNKYTPrKcz/wjar6APBB4AVgH3Cwqq4EDg7L0rvGJP+N/PuA3wEeBKiqt6vqJHAL8OCw2YPAx6Y1pDRGk7zy7AZeA/4mybNJHkhyAbC9qo4O2/wE2L7Sk5PsTbKYZPGl115bn6mlEZgknjngQ8BfVNVVwBuccYlWVQXUSk+uqv1VNV9V87suuWSt80qjMUk8rwCvVNXhYfkASzG9mmQHwPD9+HRGlMZp1Xiq6ifAj5O8f1h1A3AE+Dtg77BuL/D4VCaURmpuwu3+EPhaQoPPAj4A/JxyC8luRO4GfhP4ztnVvJ1MxS/Ev8BmqGlvmBJGAVZwQThmj9wAVvz6Ukp8ZPjcszQ3UG6Wk5BiuiZ/7nfaDtcrtezfyPiAJP8+NRzFHCH4TLA4QnI5knJzq3zWDFd4w+fiEjNpomV1Ox8p/8DDgEysUDOc9vxfWgAAAAASUVORK5CYII=
"

G5QDXAweHXQ4At05jQGmsJn3l+RLwOeD0sHwp8HZVLQ7Lx4GdKz0xyb4kC0kWOLmmWaVRWTWeJB8D3qiqZzoH
qKr9VTVfVfNs6/wEaZzmJtjnOuDjSW4G3gv8AnA/cFGSueHVZxfw6vTGlMZn1VeeqrqnqnZV1eXAHcC3quqTwFPAbc
Nue4HHpjalNEJr+T3P54HPJDnG0nugB9dnJGlzmOSy7X9U1beBbw+PXwKuXv+RpM3BOwykJuORmoxHajIeqcl4pCbj
kZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORm
oxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqmiieJBclOZjk+0mOJrk2ySVJnkzy4vD94mkPK43JpK889wPfqKoPAR8
GjgJ3A4eq6krg0LAsvWusGk+S9wO/xfCn4qvqv6rqbeAW4MCw2wHg1mkNKY3RJK88u4GTwFeSPJvkgSQXANur6sS
wz2vA9pWenGRfkoUkC5xcn6GlMZgknjngI8CXq+oq4CeccYlWVQXUSk+uqv1VNV9V82xb67jSeEwSz3HgeFUdHpYPs
hTT60l2AAzf35jOiNI4rRpPVb0GvJLkg8OqG4AjwOPA3mHdXuCxqUwojdTchPv9MfBQkvOBl4A/ZCm8R5LcCbwM3D
6dEaVxmiieqnoOmF9h0w3rO460eXiHgdRkPFKT8UhNxiM1GY/UZDxSk/FITcYjNRmP1GQ8UpPxSE3GIzUZj9RkPFKT8
UhNxiM1GY/UZDxSk/FITcYjNRmP1GQ8UpPxSE3GIzUZj9RkPFKT8UhNxiM1GY/UZDxSk/FITcYjNRmP1GQ8UtNE8ST
50yTPJ/nXJH+b5L1Jdic5nORYkoeHv1cqvWusGk+SncCfAPNV9WvAFuAO4F7gvqq6AngLuHOag0pjM+ll2xzwc0nmgP
cBJ4DrgYPD9gPAres/njReq8ZTVa8Cfwn8iKVo/gN4Bni7qhaH3Y4DO1d6fpJ9SRaSLHByfYaWxmCSy7aLgVuA3cAvAhc
AN016gKraX1XzVTXPtvac0uhMctl2I/DDqjpZVaeAR4HrgIuGyziAXcCrU5pRGqVJ4vkRcE2S9yUJcANwBHgKuG3YZy/w
2HRGlMZpkvc8h1n6YOA7wPeG5+wHPg98Jskx4FLgwSnOKY3O3Oq7QFV9AfjCGatfAq5e94mkTcI7DKQm45GajEdq
Mh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJ
eKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZom+rOKU3EenN5zmsUbF2c2gnQ2p686DVt/9j6zi2crnNp7ilN3
nJrZCNJZbQV+/mfvMrt4wtJwqwwojVWqauMOlpwEfgK8uWEHXZsPsHlmhc0172aa9ZeratuZKzc0HoAkC1U1v6EHb
dpMs8LmmnczzXo2ftomNRmP1DSLePbP4Jhdm2lW2FzzbqZZV7Th73mk/y+8bJOajEdq2rB4ktyU5IUkx5LcvVHHnVS
Sy5I8leRIkueT3DWsvyTJk0leHL5fPOtZ35FkS5JnkzwxLO9Ocng4xw8nOX/WM74jyUVJDib5fpKjSa4d87mdxIbEk2QL8
FfA7wB7gE8k2bMRxz4Hi8Bnq2oPcA3wqWHGu4FDVXUlcGhYHou7gKPLlu8F7quqK4C3gDtnMtXK7ge+UVUfAj7UqqfwHXAt9ctnwPcM9GHHsNMz8GfBR4AdgxrNsBvDDr2YZZdrH0D+564AmWbnh6E5hb6ZzPeNb3Az9k+IB
q2fpRnttJvzbqsm0n8Mqy5ePDulFKcjlwFXAY2F5VJ4ZNrwHbZzTWmb4EfA44PSxfCrxdVe/cpj6mc7wbOAl8ZbjMfCDJ
BYz33E7EDwzOkORC4GvAp6vqQ6uqdH0D+564AmWbnh6E5hb6ZzPeNb3Az9k+IBq2fpRnttJvzbqsm0n8Mqy5ePDulFKcjlwFXAY2F5VJ4ZNrwHbZzTWmb4EfA44PSxfCrxdVe/cpj6mc7wbOAl8ZbjMfCDJ
d2oJNnKUjgPVdWjw+rXk+wYtu8A3pjVfMtcB3w8yb8DX2Xp0u1+4KIk79wpP6ZzfBw4XlWHh+WDLMU0xnM7sY2K5
2ngyuHToPOBO4DHN+jYE0kS4EHgaFV9cdmmx4G9w+O9LL0XmqmquqeqdlXV5Sydy29V1SeBp4Dbht1GMStAVb0G
vJLkg8OqG4AjjPDcnpMNfNN4M/AD4N4AP/AP5/1m70V5vtNli4bvgs8N3zdzNJ7iUPAi8A/AJfetMetYz5v5t4Inh8a8A/wIcA
/4OeM+s51s2568DC8P5/Xvg4rGf29W+vD1HavIDA6nJeKQm45GajEdqMh6pyXikJuORmv4bWtmd36znqt4AAAAAS
UVORK5CYII=\n"

        },

        "metadata": {

          "needs_background": "light"

        }

      }

    ]

  }

 ]

}