```json
{
  "nbformat": 4,
  "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": []
    },
```

```json
  {
   "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
wf1WwwZBU2SzTNTKMWNl1qEzUgaQouoOzGhDgiKo5LdQ2mTEaowZEf/mCHefaPezBTnfu9w7nn3nOZ5/1Kbu69
57nnnicnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAIuaubCzlxT/0CDubuNN72uLbuZ3Wpmph8zsPTN7sJ
7vAtBYlvc6u5INkfRHSUslHZH0qqQudx9IzMWHWWirzmRmzRF0t6z6z93fd/czka/1Kbu69
57nnnicnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAIuaubCzIxT/0CDubuNN72uLbuZ3Wpmph8zsPTN7sJ
7vAtBYlvc6u5lNkfRHSUslHZH0qqQudx9IzMWHWWiwRmzZF0t6z6z93fd/czkn4raVkd3weggeoJ+2xJfrz+xJfxz3wYIYm
Z9ZtZfx3zLAlCnhp+gc/c+SX0Su/FAmerZzsg9KmjPm/bezaQBaUD1hf1XSSObs2HTObkUlHkrYV0xaAouoaAouCsANBEH
YgSAIOxAEYeCIOxAEYQdCKwwA0EQdiDIgQBGHgiDsQBCEHQiCsQBEHQiCsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeC
4LgOvskN3/+/GR9//79fjw9yfrw8HCyPjQ0lKynp6ewtw55zZmaw//fTTyfpll12WrC9evDhZHx0dTdYAJqDewaPra2pL
1ffv2Jevz5s1L1nt6epL1m266KVmfaGZ2P//9nJz33nsvWb/iiiuS9YGBgWT9mmuuSdZ37dqVrA8ODibrAwMDyfrChQuT
9ampqWS9t7c3WV+wYEGy/vrrryfrOXPmJOs5OTkZXXfddcn6wsJCsr548eJkfcuWLcn6LbfckqxPnTo1Wd+xY0eyfvvt
tyfry5YtS9b7+/uT9WuvvTZZX758ebK+atWqZH3btm3J+p133pmsr1+/PlmvBkd2IAjCDgRB2IEgCDsQBGEHgiDsQBCE
HQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjC
DgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EARh
B4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKw
A0EQdiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBEHY
gSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDs
QBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2
IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7
EARhB4IwXX"

2C0Cta9ldXeMNtFtR6zr63Llz87RUiP7+/mR97dq1yfq2bduKbGfSq7llN7M5ZrbHzAbM7G0zW5VNbzOzF83s3ex5ZuP
bBZDXRHbjRyT9o7svlPR3klaa2UJJD0ra5e5XStqVvQfQomqG3d2H3H1/9vqkpHckzZa0TNKm7GObJN3VqCYB1O+8jt
nNbK6k70naJ6nd3c/9OPlDSe1V5umW1J2/RQBFmPDZeDP7pqRnJf3M3U+MrXlldMhxB2109z53X+Tui+rqFEBdJhR2
M/uGKkH/jbs/l00+amazsvosScca0yKAItTcjTczk/SkpHfcfcOY0jZJyyWty563NqTDSaC9fdwjnC8tXLgwWX/88ceT9auu
uuq8eyrKvn37kvVHH320am3r1vQ/GW5RLdZEjtn/XtKPJb1lZgeyaWtUCfnvzGyFpyA8k/bAxLQIoQs2wu/t/Sxp3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMEtrhPU1tZWtdbb25uct6OjI1mfN29erp6K8MorryTr69evT9Z37NiRrH/2
2Wfn3RMagy07EARhB4Ig7EAQhB0IgrADQRB2IAjADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHHV2uzZs3P1VJRPP/20am3jxo3
JeR955JFk/fTp07l6Quthyw4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQYS5zt7Z2VlXvR4DAwPJ+vbt25P1kZGRZD11z/nw8
HByXsTBlh0lgrADQRB2IAjCDgRB2IEgCDsQBGHgjB3T3/AbI6kzZLaJbmkPnf/NzN7SNIDkv6cfXSNu/++xnelFwagbu4+
7qjLEwn7LEmz3H2/mU2X9O/mU2X9Jqku1QZ/2ywa17j2Zh/2/mdw/v8Z3sBsPNFjuY3ZJMrNvNvSNNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCcljzY3ZR6st0kAjTOY/FLJP2XpLckjWaT10jqktShym78UPT5dNNobeMT
M7OGZam5m9aGbvZs/jjrFFUm8Pmdlgtu4OmNntJfU2x8z2mNmAmb1tZquy6aWuu0RfTVlvTT9mN7Mpkv4oaWukv4oamk I
5JeldTl7gNNbaQKMzssaZG7l/4DDDP7B0mnJG0+N7SWmf2LpOPuvi77j3j3j3j3j3j3j3Kmu/+8RXp7SOc5jHeDeqs2zPhPVOK6K3L
48zzK2LLlvlvSeu7/v7mck/VmckVbSshL6a9x9MXIiZpU/Z6kyr/WJquSm8twd2H3H1/9vqkpHPDjJe67hJ9NUUZYZ8t6
U9j3h9Ra4337pJ2mtlrZtZddjPjaB8zzNaHktrLbGGYcNYfxbqavDDPeMusuz/Dn9eIE3dctcfe/lXSbpJXZ7mpL8soxWCtdO
/2lpPmqjAE4JGl9mc1kw4w4w/K+ln7n5ibK3MdTdOX01Zb2WEfVDSnDHvv51NawnuPpg9H5P0vCqHHa3k6LkRdLPnYy
X38yV3P+ruZ919VNKvOK7y4YZf1bSb9z9uWxy6etuL6atd7KCPurkq40s++3YZf1bSb9z9uWxy6etuL6atd7KCPurkq40s++Y2VRJ5K0rYQ+vsbMpmUnTmRm0yT9YQ
QK03FPU2Scuz18slbS2xl7/QKsN4VxtmXCWvu9KHP3f3pj8k3a7KGfn/lfRPZfRQpa95kt7U02UnTmRm0yT9oT9
6a0m7JL0r6T8ltbVQb/+uytbVQb/+uytDeb6oSrFkl9bZElV30tNyUdyB63l73uEn01Zb3xc1kgCE7QAUEQdiAIwg4EQdiBlAg7EARhB
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+WH4yJAAAOF0lEQVR4nO3dcYxV5ZnH8d8jjW4xKIgpTkrr2+AfzUYYUKyprI2bVw0gcakQozDpk2GxJJQszGr3VF
IamNjlEZNJE6VFFcqqGjBpbpi51GaLdmDSOyCpqW1mDFhwzUSNDTKTCs3/cQzPinPcM9557z4Hn+0km997zzLn38Tl/z7
nnPfe85u4CcPI7peoGGGGYQeCIOxAEYQeCIKyA0H8QYdfzdzMw49A+0mbvbWMtb2rKb2ZVm9mcz22VmN7fyXXADa
y5odZzezeCZL+luk7ktHSYnd/PbEOW3agzdqxZZe7v+XuyStl7SghecD0EathP1pcn9ErvxQJVa2bLvlXXTeqfTs2UAaqiVsL8oaYaZfc3MJjpm37/+pmZTWXi2iMrkM1jpJGlzOW0BKFvTu/nuvlXSdadTrfA2Rd1/DXDaDD2IAjCDgRB2IEgCDsQB
EwiyuSuru7k/z2aWrGGYR92bZLG8W+bXXrLG8W15YRLG8W15ZRLG8W15ZRLG8W15ZRLG8W0EwiyuSuru7k/z2aWrG
mw+CTzwwAO5tcWLF3Sewk+NTdA7ApEmTkvXnnns+Ts44enOOOSV/W3bkyJEdliTXfeSRR5L16dOny0vEuBMIADiBIYgCMIO
BEHYgSAIOxAEYQfCIIy7EAQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiDsQBCEHQhCFs4+TmTVVk6bFCxwm68uXLk/
Xp06eXl4hxJxAAcAJDEAVhB4Ig7EAQhB0IgrADQRB2IAiDsQAR5gBMMUofKkNu8fvpuR1gXx3SX9j8sXVVf9mnGc9QBol8J
W3bjMdtE4+bXXXpusL1iwIFlfuXJl1vyjNvvbu7v3f+yLzuaiAYBU/AjEQRTAAQRB2IEgCDsQBGEHgiDsQBCEHQiDsQBClSZfg1FPTdZl6xI
VZbXXzHZkP61dnQFA241nN/5+bXXXpusL1iwIFlfuXJl1vyjNvvbu7v3f+yLzuaiAYBU/AjEQRTAAQRB2IEgCDsQBGEH
giDsQBBCs4+TmTVVk6SFCxxcm68uXLx/yym00wmw4BKDd4BkS8VnbrMdtE4+bXXXpusL1iwIFlfuXJl1vyjNvvbu7v3f+
yLzuaiAYBU/AjEQRTAAQRB2IEgCDsQBGEHgiDsQBCEHQiDsQBClSZfg1FPTdZl6xIVZbXXzHZkP61dnQFA241nN/5+bXX
Xpusl1iwIFlfuXJl1vyjNvvbu7v3f+yLzuaiAYBU/AjEQRTAAQRB2IEgCDsQBGEHgiDsQBBCs4+TmTVVk6SFCxxcm68u
Xp0+fPn16dQk+NTdA7ApEmTkvXnnns+Ts44enOOOSV/W3bkyJEdliTXfeSRR5L16dOny0vEuBMIADiBIYgCMIO"

2+qpzq48cYbk/Vbb701tzZlypTkuuvWrUvWe3p6knV8Hlt2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCcfZxcvemapJ0zjn
nJOv33ntvsr5mzZpk/YMPPsitzZ07N7nu9ddfn6xffPHFyfr06dOT9XfeeSe3tmXLluS6999/f7KO41O4ZTez88xsm5m9b
mavmdnybPmZZvasmb2Z3U5tf7sAmjWe3fjPJP2bu39T0lxJPzKzb0q6WdJWd58haWv2GEBNFYbd3YfcfXt2f0TSG5LO
lbRA0trs19ZKSp8TCqBSx/WZ3cwukDRT0h8lTXP3o5OUvSdpWs46vZJ6m28RQBnGfTTezCZJ2ijpx+5+YHTNG0eoxjxK
5e797j7b3We31CmAlowr7Gb2JTWCvs7dn8wW7zOzrqzeJWm4PS0CKEPhbrw1vr/5kKQ33H3VqqNJmSUsk/Ty7TV/X
N7AJEyYk6zfccEOyXnRJ5AMHDuTWZsyYkVy3VS+88EKyvm3bttzabbfdlly7SBjPZ/Z/knS9pFfNbEe27ZCdqhPwxM/uh
pLclfb89LQIoQ2HY3f1/JOVdneHb5bYDoF04XRYIgrADQRB2IEgrOjrmaW+mFnnXqxkqa9yPv7448l1L7300
pZeu+hS1a38G6a+HitJ69evT9ZP5Mtgn6zcfcw/GLbsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAE4+wl6OrqStaXP
f19SXrrYYz33PPPcl1V69enazv2razv2rUrWUf9MM4OBEfYgSAIOxAEYQeCIOxAEQdCIKwA0EQdiIDgrB2IEgrOjrmaW+m
FFnnVQznvnZhyQNZfdHzOwNSeeW2x6Adjuuz5MXjOz5dnylWa218x6D/z298ugGYVnlRjZl2Sutx9u5l9WdLn
QJoybjPjTezSZKek/Qzd3/SzKZJ2i/JJf1UjV39HxQ8B7vxQJvl7caPK+xm9iVJv5W0xd1Xjp5XjVG/QNJv3f0fC56HsANt1vQXYa
xxadOHJL0xOujZgbujvidpZ6tNAmif8RyNv0zSHyS9KulItvgnkhZL6lZjN363pKXZwbzUc7FlB9qspd34shB2oP34PjsQHG
EHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiCIwgtOlmy/pLdHPT47W1ZHde2trn1J
9NasMnv7al6ho99n/8KLmw26++zKGkioa2917Uuit2Z1qjd244EgCDsQRNVh76/49VPq2ltd6a3Sz+wAOqfq
LTuADiHsQBCVhN3MrjSzP5vZLjO7uYoe8pjZbjN71cx2mNL56bI59bNbNbNbNbOeoeZWa2Z2O+YcexX1VotpvBPTjFf
63lU9/XnHP7Ob2QRJf5H0HUl7JL0oabG7JJL0ooabG7v97RRnKY2W5Js9298hMwzOxbkg5Kevjo1Fpmdqekd93/Xp
LeVOs5pvNvUW9404/+qCt+7Mqc/b0YVW/Y5kna5+1vufkjSSkkLKuij9tz9ew9eUkfHrN4gaS12f21avyxdFxOb7Xg7kPuvj2
7PyLp6DTjlb53PyLp6DTjlb46ooqwnyvpr6PGe7JWpJ31BjDsAhSXdX2Uw6W+9JmrjSb83zrKfhMrfLpJlU9/7u4d/5E0X40j8v8n6T+q6CGGr6
9L+t/s57Wqe5P0qBq7dX9T49jGDyWdJW7n6rsp/5/h5/z6W65nt3Sb83s5fMrLfqZZsYybdDQ0W+9JmlZLdXyJpU4W9f9E5dpvHOm2ZcFb93iz05s05+3iC
0SNLmCvr4AjM7IztwIztwIJ3Vb+pqDdLWpLdXyJpU4W9f9E5dpvHOm2ZcFb93lU9/7u4d/5E0X40j8v8n6T+q6CGnr6
9L+t/s57Wqe5P0qBq7dX9T49jGDyWdJWmrpDcl/bekM2vU23qMbX49jGDyWdJWmrpDcl/bekM2vU23+qMbX3K2oEq6ui3i5TYxf9FUk7sp/5Vb93ib468r5xui
wQBAfogCAIOxAEYQeCIOxAEIQdCIKwA0EQdiCI/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+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjlslGh0dHA6Ly9tYXRwbG90bGliLm9
yZy+WH4yJAAAKyklEQVR4nO3dX4ilhXnH8e/PnTVVptI1/sizbXVu3KEmXQmoYrGIpRQ1YG6IXIqahXYqwN2lrmkCi7UV
uK5QYL0pg0YalSKPdSBUJCXZjLlro1jFKUndj3JoaV1ZdQZuSm+5kn17Ma5luZzZNnn5kz5536/cAw5/1z5n142e+95x5Y
VJVSDp35816AGmzMh6pyikJ4aZSKPdSBUJCXZjLlro1jFKUndj3JoaV1ZdQZuSm+5kn17Ma5luZzNnn5kz5536/cAw5/1z5n142e+e95x5Y
VJVSDp35816AGmzMh6pyXikJuORmoxHajIeqlN8SS5KckLSY4luXu9g3pI2g3Axg4JPaceBKFFjev4YjSLDzDm1W17czVa4lnJ/DKsuXjwG+uVOSfA4KPAceBp4BNVdWT9xpP
Ga24Nz70aOFZVLwEk+SpwC3DWePKBFFev4YjSLDzDm1W17czVa4lnJ/DKsuXjwG+uVOSfA4KPAceBp4BNVdWT9xpP
HrqHxhU1f6qmq+qef5Pu9LmtZZLZZnleBy5Yt7xrWTaaAReCna5hAmpxLgbONRVVwKHhmXpXWPVeKrqRFV9Z3j8n8BRYdwC3Bg
2O0AcOu0hpTG6Jze8yS5HLgKOAxsr6oTw6kXge6+JAtJFfj8DPl1VP16+rao6q+qnubZZpVGZJ4kW1kK56GqenY/XqSHcRNg/vfc
ubZtqZZpVGZKJ4kW1kK56GqenY/XqSHcRNg/vfcLBwtKq+uGzT48De4fAe4fLH1H1H08ar7kJ9rkO+H3g
e0meG9b9G7fAXwCNJ7gReBm6fzojSOK0L8CBwtCNJ7gReBm6fzojSOK0L8CBwtKq+uGzT48De4fAe4fLH1H1H08ar7kJ

R2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajI
eqcl4pCbjkZqMR2o6lz8lvyXJs0meGJZ3Jzmc5FiSh5OcP70xpfE5l1eeu4Cjy5bvBe6rqiuAt4A713MwaewmiifJLuB3gQe
G5QDXAweHXQ4At05jQGmsJn3l+RLwOeD0sHwp8HZVLQ7Lx4GdKz0xyb4kC0kWOLmmWaVRWTWeJB8D3qiqZzoH
qKr9VTVfVfNs6/wEaZzmJtjnOuDjSW4G3gv8AnA/cFGSueHVZxfw6vTGlMZn1VeeqrqnqnZV1eXAHcC3quqTwFPAbc
Nue4HHpjalNEJr+T3P54HPJDnG0nugB9dnJGlzmOSy7X9U1beBbw+PXwKuXv+RpM3BOwykJuORmoxHajIeqcl4pCbj
kZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORm
oxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqmiieJBclOZjk+0mOJrk2ySVJnkzy4vD94mkPK43JpK889wPfqKoPAR8
GjgJ3A4eq6krg0LAsvWusGk+S9wO/xfCn4qvqv6rqbeAW4MCw2wHg1mkNKY3RJK88u4GTwFeSPJvkgSQXANur6sS
wz2vA9pWenGRfkoUkC5xcn6GlMZgknjngl8CXq+oq4CeccYlWVQXUSk+uqv1VNV9V82xb67jSeEwSz3HgeFUdHpYPs
hTT60l2AAzf35jOiNI4rRpPVb0GvJLkg8OqG4AjwOPA3mHdXuCqUwojdTchPv9MfBQkvOBl4A/ZCm8R5LcCbwM3D
6dEaVxmiieqnoOmF9h0w3rO460eXiHgdRkPFKT8UhNxiM1GY/UZDxSk/FITcYjNRmP1GQ8UpPxSE3GItNE8ST
50yTPJ/nXJH+b5L1Jdic5nORYkoeHv1cqvWusGk+SncCfAPNV9WvAFuAO4F7gvqq6AngLuHOag0pjM+ll2xzwc0nmgP
cBJ4DrgYPD9gPAres/njReq8ZTVa8Cfwn8iKVo/gN4Bni7qhaH3Y4DO1d6fpJ9SRaSLHByfYaWxmCSy7aLgVuA3cAvAhc
AN016gKraX1XzVTPTvac0uhMctl2I/DDqjpZVaeAR4HrgIuGyziAXcCrU5pRGqVJ4vkRcE2S9yUJcANwBHgKuG3YZy/w
2HRGlMZpkvc8h1n6YOA7wPeG5+wHPg98Jskx4FLgwSnOKY3O3Oq7QFV9AfjCatfTCgatfAq5e94mkTcl7DKQm45GajEdq
Mh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJeKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZqMR2oyHqnJ
eKQm45GajEdqMh6pyXikJuORmoxHajIeqcl4pCbjkZom+rOKU3EenN5zmsUbF2c2gnQ2p686DVt/9j6zi2crnNp7ilN3
nJrZCNJZbQV+/mfvMrt4wtJwqwwojVWqauMOlpwEfgK8uWEHXZsPsHlmhc0172aa9ZeratuZKzc0HoAkC1U1v6EHb
dpMs8LmmnczzXo2ftomNRmP1DSLePbP4Jhdm2lW2FzzbqqZZV7Th73mk/y+8bJOajEdq2rB4ktyU5IUkx5LcvVHHnVS
Sy5I8leRIkueT3DDWsvyTJk0leHL5fPOtZ35FkS5Jnkzwxw0O9cng4xw8nOX/WM74jyUVJDib5fpKjSa4d87mdxIbEk2QL
FfA7wB7gE8k2bMRx36b2bMRxz4Hi8Bnq2oPcC3w8nX4quqK4C3gzmkNKY3RJK88u4GTwFeSPJvkgSQXANur6sS
wz2vA9pWenGRfkoUkC5xcn6GlMZgknjngl8CXq+oqlu5v/F+XaGM5t5+dio+J5Fbhs2fKuY4d2oJNnKUjgPVdWjw+rXk+wYtu8A3w8yb8DX2yb8yb8DX2Xp0u1+4KIk79wwpP6ZzfBw4XlWHh+WDLMU0xnM7sY2K2K52ngyuHToPOBO4DHN+jYE0kS4EHgaFV9cdmmmqmquqeqdlXV5Bp4GMtAVb0G/Lmq70vdJ/tKP/bD/Jx4A3quqWc8yoTngI8Tng8DHN+jYE0kS4EHgaFV9cdmmmqmquqeqdlXV5Sydy29V1SeBp4Dbht1GMtAVb0G
vJLkg8OqG4AjjPDcnpMNfNN4M/AD4N+AP5/1m70V5vtNli4bvgs8N3zdzNJ7iUPAi8A/wIA/wIcA
/4OeM+s51s2568DC8P/Xvg4rGf29W+vD1HavIDA6nJeKQm45GajEdqMh6pyXikJuORmv4bWtmd36znqt4AAAAAS
UVORK5CYII=\n"
    },
    "metadata": {
     "needs_background": "light"
    }
   }
  ]
 }
]
}