```
{
 "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": []
  },
```

```
{
 "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+zWP6QojBrOhKKSyEGg8ZON4gbl6w1hvojGhw1TSexE4/YNJaLIhNe
wf1WwwZBU2SzTNTKMWNl1qEzUgaQouoOzGhDgiKo5LdQ2mTEaowZEf/mCHefaPezBTnfu9w7nnOZ5/1Kbu69
57nnnicnfDi/7pmvubsATH5/VXYDAJqDsDsANBEHYgCMIOBEHYgSAuaubCzIxT/0CDubuNN72uLbuZ3Wpmh8zsPTN7sJ
7vAtBYlvc6u5lNkfRHSUlHZH0qqQudx9IzMOWHWiwRmzZF0t6z93fd/czn4n4x4gn7xk/kg27S+YWbe
Z9ZtZfx3LAlCnhp+gc/c+SX0Su/FAmerZsg9KmjPm/bezkbkRD0
hRJT7n724V1BqBQuS+95VoYx+xAwzUhzDzkbRzUALhyEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdi
AIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBJF7yGZcG
KZMmZKsX3rppQ1dfk9PT2XtSlX3XaxfnfJx3wYIFyfrKlSuT9ccee6xqraurKzn2ppZSubF5y5v/PfVcnM+dXNXHv/rOSQDF8cBS4Pr5DCCC20MfBnSUFPXU0XL1WdMCE6P0eC/K0ZVaccMF2YuuSk01dU0Q/8+PUQDxRm1c/Uz8eYh8a8sSwNERmHT/vyfPZMpuUmgvdVtHxvCC7YEN9yVj0o6/S26S+ebbbVz6OP6cJJSe/MsnQNYXyP8d//3y+Vf+tDJlMZpD5hlnZm9kzl/PO0bKbHXmNYn/v9iuzKXwdQ1v1frRk7bL2yEjKbYB7/NCs7SeG/P3A2ABgAaGAdWMZlEbY7L1yH1QLBqGlz/K3f0xv10GLEWZBFDR/dCfZxMqoB4R0GcyC0bB4FWvsU9qC6vzxQMxY0dYNmG4HeAF8P/WEaFhBGgBAzBZYqKSLT/3tjmh3S+fY9zpWQs6ysqRtMBVMdn6oGf9ccLMKtKrb3/4X14zW21O8x+VEVoKSO0M/Ip+XXH0z6f0Nzm6xy9cvkeN2lFvEhSL07LgSrpPNVubSbdCaZQTSpmYt8fGMoOxSCyA0Jfl+4BmgFLlMo4z+HNwX8dcnF2hWBqDZ+PM5cZ9Aq+JDYfCWOiAcfyKxLw/wHSz0mEA5Xi11p3aaYvA5+Lqv9o9DyzCZDpvBeYcCVGvgWAFEkHHZrAIxkJKEYArNHMP1FMYY4KgPRqmbApbZiuikK3I4ay+OvKZfGxDFlWFapcEQ/gmC3W9E/p4zjWzxYzkHvKppyiiIJbd9kBRCiLcu5cQFfLbQgdZrBXpYm3FXRqfTFAi5uNCzlkJG7y+B8lnOdzFbZvGPtHcS/eS2KDtiFeA6NW/cb6KqHwZvWT38yw1SNUeRK4tTRuQMQ8BRj9+JZ3TJDuLyJkU63fOCxgq/f1QLCd0AtErDfHhuWC2yh3F3+1d/yD1cC+bTq+6vWxTVHF3Z8AOVEeY/Kn/UGsUbZ8dRJFGP+dbdLEaxVeWzOZD5wHuZb82Rr/9ZcWDtAGCBZBDb+q/yfTMiHJR0Xx9QAXQ/EdwSU7FhhEDt+DDNM1a3qY6RlCWKFFgQVK1YnGnA38ns4RG2bj1VJfNBqtxLNxMmHnxDsCOAWQELmhLKSagAHnMAOQcuyr/Jw3f57CpfufgpTGyOnJ7+JA+AkfsH7rDwGgbO5hPcbKeMDuUDdrzF+Mz01kWfRqmXj4h+LnCBOUoyPOB5idaPVEhmrpbhOWWHSiMHzMM7pkfABPz2JDqU/5ijIkfpgcq2ip/Nmr/B0iNjIqr8K8+ntVKAkbEoB35WFNjewnQfwdSmK/DJzhqpLgbLXsJZhMOB6h7IsFZpkKzXzfYlAGVs8jzz6/HdgXBp+Wee3BgsiBwI8DOQzW+Idrq0K8QeKg1R6cL3AYQBgjIg7JQH3qNO5KYY2Vrx2Kh0b6SsWcNrAVoVwG+/QL2KvzbYkfTsuXpzrb3HvFqKF93aHT1GzhRdeBPSGWVzu4Nv3hT4Ua6+YJW0TWD8xxd3YJbBBhmjR2AqpvFOeuXg0f9SSzh8zOGgjL5ccfPgmBL6LP7DwLYEqQHnlNGjyuI+yshMJV+hvkXJDQQXVw8hTCSuKGL9iePFVM1vc8QbsvygTDEMT7y95VbwDtQHMHyoMHqTm08mbY+cjN7QFAtuIgFhJFk3vtQ6Xa0ewxQ45ZYIhRk93EZiU9yw3QXqUF3J+cgzxPPC+P6unDNb5cpdvBkLwpe5NJ6UiKE71TiIhowiiDN6TpNtchhI2b5qCXy0TmTjwB1T1z3MWn6WDsfwXF6nqDTWH1uuLhfRBDRp7jH9RLsnsXhvQd/v11c8Pw+bvDO1wNHa4lpfiQ6T3CaQUqDmqbt6fXtRzGZzpnXMsHh/Ip7gvU4jBsKxm8lmr4LZlhuFHD4ikF1uNsY4vT9Wjw4iHBh+Pd1nl+zQ4XjrF3ZIWBUz9YYKHrTT+m+TMpaU9fBhUUeoCt8nZZmS32h/NdzyFWn8DfeH7Z8LgG6AdC3Qw27cM/5/VCt9Oy+MPdNTWn6KUAwJ5wW/Ul4e5dX1VpfbHUGwhBR7Nps3fffDHXd2uv1Mg2zZFN1W6fn3zxfMfbqDzR30

2C0Cta9ldXeMNtFtR6zr63Llz87RUiP7+/mR97dq1yfq2bduKbGfSq7llN7M5ZrbHzAbM7G0zW5VNbzOzF83s3ex5ZuP
bBZDXRHbjRyT9o7svlPR3klaa2UJJD0ra5e5XStqVvQfQomqG3d2H3H1/9vqkpHckzZa0TNKm7GObJN3VqCYB1O+8jt
nNbK6k70naJ6nd3c/9OPlDSe1V5umW1J2/RQBFmPDZeDP7pqRnJf3M3U+MrXlldMhxB2109z53X+Tui+rqFEBdJhR2
M/uGKkH/jbs/l00+amazsvosScca0yKAItTcjTczk/SkpHfcfcOY0jZJyyWty563NqTDSaC9fdwjnC8tXLgwWX/88ceT9auu
uuq8eyrKvn37kvVHH320am3r1vQ/GW5RLdZEjtn/XtKPJb1lZgeyaWtUCfnvzGyFpA8k3bAxLQIoQs2wu/uQJLvSS2Jx3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMEtrhPU1tZWtdbb25uct6OjI1mfN29erp6K8MorryTrTz55J37NiRrH/2
2Wfn3RMagy07EARhB4Ig7EAQhB0IgrADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHiVOk2j3ZPP/20am3jxo3
JeR955JFk/fTp07l6Quthyw4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQYS5zt7Z2VlXvR4DAwPJ+vbvt25P1kZGRZD11z/nw8
HByXsTBlh0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgjB3T3/AbI6kzZLaJbmkPnf/NzN7SNIDkv6cfXSNu/++xnelFwagbu4+
7qjLEwn7LEmz3H2/mX1L0uuS7lJlPPZT7v6Libbw4Ahgb2/mU2X9Jqku1QZj/2UuzB820SYIO9B41cI+kfHZhyQNZa9Pmtk7ksr90ywAztt5HbOb2VxJ35O0L5vU
Y2ZvmtlTZjazyjzdZtZvZv11dQqgLjV347/8oNk3Jb0saa27P2dm7ZI+UuU4/p9V2dW/v8Z3sBsPNFjuY3ZJMrNvSNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCclvTM26NmJu3M6JR2st0kAjTORs/FLJP2XpLckjWaT10jqktShym78+YUk/
zU7mpb6LLTvQYHXtxheFsAONl3s3HsDkQNiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBC
EHQii2UM2fyTpgzHvL8umtaW7a1V+5LoLa8ie/ubaoWm3s/+tYWb9bv7otIaSGjV3lq1L4ne8mpWb+zGA0EQdiCIssP
eV/LyU1q1t1btS6K3vJrSW6nH7ACap+wtO4AmIexAEKWE3cyWmtm7ZnbIzB4so4dqzOyImb1lZgfKHp8uG0PvmJkdHDPtUjPbaWbvZc/jjrFXUm8Pmdlgtu4OmNkdJfU2x8x2m9nbZvZHM1uZTS913SX6asp6a/oxu5lNkfQnSbdJ+lDSa5K63P3tpjZShZkdkbTI3Uv/AoaZ/YOkU5I2nx1ay8z+VdJxd1+X/aKc6e6rWqS3h3Sew3g3qLdqw4zfpxLXXZHDn+dRxpZ9saRD7n7Y3b+U9FtJy0roo+W5+8uSjn9t8jJJm7LXm1T5z9J0VXprCe4+5O77s9cnJZ0dZrzUdZfoqynKCPtsSQNj3n+o1hrv3SX9wcxeN7Puspsfg7u7tP5q7o+Kh3MV6W3cz+jM/G8z7j5c+A0W4jPo57uHu8WWTcj1G/Ukbt+XnW2VZbYx52lLlEsZbyPc7pZjMLyS09TP4y4l38c/P5PW95+YtYEo++m85z8TbbS6wjDLsr3fz72m53trW3c5E3PAeVKD1bvmhUl7Sv0VmUsKRPhpYVXZXOkm1NP7ubhLkeU7hx0rqOzX6wDkOJDfO0oVE7dJ3DKXpLI+J/kx0oPeVNZEd/EJC9+JhItL5TXWfnSLx1uk5GchGqXpCuPpFsJ+oacu1Z/bX0/PaNtoU73VCZlFV7pNb1BzmNmFGaDU/nj9fxyA/V9+B73fVthPZTrd4Do93P3/ycrHYZPNarAwU8sQxhbLCGw4ObfrEXcNl1A85b2R0KiIF+cYEgNc/nSvlGcU9snk1ZEzlOZGgmg40FKHL9uFvpVkSRrVkoTbKd2WXS1htuzaMRsAFxLs4g9XaEY2AWcsUGX0pvBTM/QmkfAQHd9i4p1NMzGTqvnT/GjGYUo/fdL92hS9e6jA=\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+AfzUYHQUKyprI2bVw0gcakQozDpk2GxJJQszGr3VF
IamNjlEZNJE6VFFcqqGjBpi51GaLdmDSOyCpqW1mDFhwZUSNDTKTCs3/cQzPinPcM9557z4Hn+0km997zzLn38Tl/z7
nnPfe85u4CcPI7peoGAHQGYQeCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIOBCIO
1wLQorYfoHP3fkn9ErvxQJVa2bLvlXTeqMfTs2UAaqiVsL8oaYaZfc3MJkpaJGlzHQfOW0BKFvTu/Hu/pmZLZO0RdIESWvc/
bXSOgNQqqaH3pp6MT6zA23XlpNqAJw4CDsQBGEHgiDsQBCEHQiCsQiCsANBEHYgCMIiCsANBEHYgCMIiCsANBEHYgCMIOBMFXXDvg000
NyAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyt999yAYcsOBEHA2aFrQ6m78Ze6+18y
YcsOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
Gz3ZJGJB2W9Jm7z27l+QC0Txln0P2zu+8v4XkAtBGf2YEgWg27S/q9mb1kZr1j/YKZ9ZrZoJkNtvhaAFrQ6m78Ze6+18y
+IulZM/uTuz8/+hfcvV9Sv8QBOqBKLW3Z3X1vdjss6SlJc8poCkD5mg67mZ1hZl8+el/SdyXtLKsxAOVqZTd+mqSnzOzo
8/za3f+rlK7QMXPmpHfGNm7cmKxPmTllWU+dxzEyMpJc99ChQ8l60Tj63Llzc2tF33Uveu0TUdNhd/e3JF1cYi8A2oih
NyAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrEAQhB0IgrEAQhB0IgrEAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
uNJTb2/2M888k6WXtOwEQdiBIAg7EARhB4Ig7EAQhB0IgrEAQhB0IgrEAQhB0IgrEAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
Gz3ZJGJB2W9Jm7z27l+QC0Txln0P2zu+8v4XkAtBGf2YEgWg27S/q9mb1kZr1j/YKZ9ZrZoJkNtvhaAFrQ6m78Ze6+18y
+IulZM/uTuz8/+hfcvV9Sv8QBOqBKLW3Z3X1vdjss6SlJc8poCkD5mg67mZ1hZl8+el/SdyXtLKsxAOVqZTd+mqSnzOzo
8/za3f+rlK7QMXPmpHfGNm7cmKxPmTllWU+dxzEyMpJc99ChQ8l60Tj63Llzc2tF33Uveu0TUdNhd/e3JF1cYi8A2oih
NyAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrEAQhB0IgrEAQhB0IgrEAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
Gz3ZJGJB2W9Jm7z27l+QC0Txln0P2zu+8v4XkAtBGf2YEgWg27S/q9mb1kZr1j/YKZ9ZrZoJkNtvhaAFrQ6m78Ze6+18y
+IulZM/uTuz8/+hfcvV9Sv8QBOqBKLW3Z3X1vdjss6SlJc8poCkD5mg67mZ1hZl8+el/SdyXtLKsxAOVqZTd+mqSnzOzo
8/za3f+rlK7QMXPmpHfGNm7cmKxPmTllWU+dxzEyMpJc99ChQ8l60Tj63Llzc2tF33Uveu0TUdNhd/e3JF1cYi8A2oih
NyAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrEAQhB0IgrEAQhB0IgrEAQhB0IoqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
```

2+qpzq48cYbk/Vbb701tzZlypTkuuvWrUvWe3p6knV8Hlt2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCcfZxcvemapJ0zjn
nJOv33ntvsr5mzZpk/YMPPsitzZ07N7nu9ddfn6xffPHFyfr06dOT9XfeeSe3tmXLluS6999/f7KO41O4ZTez88xsm5m9b
mavmdnybPmZZvasmb2Z3U5tf7sAmjWe3fjPJP2bu39T0lxJPzKb0Hu3WdJWd58haWv2fhaBNFYbd3YfcfXt2f0TSG5LO
lbRA0Tx19ZKSp8TCqBSx/WZ3cwukDRT0h8lTXP3o5OUvSdpWs46vZJ6m28RQBnGfTTezCZJ2ijjpx+5+YHTNG0eoxjxK
5e797j7b3We31CmAlowr7Gb2JTWCvs7dn8wW7zOzrqzeJWm4PS0CKEPhbrw1vr/5kKQ33H3VqNJmSUsk/Ty7TV/X
N7AJEyYk6zfccEOyXnRJ5AMHDuTWZsyYkVy3VS+88EKyvm3bttzabbfdVnY7SBjPZ/Z/knS9pFfNbE27CdqhPwxM/uh
pLclfb89LQIoQ2HY3f1/JOVdneHb5bYDoF04XRYlgrADQRB2IAjCDgRB2IEgrOjrmaW+mFnnXqxkqa9yPv7448l1L7300
pZeu+hS1a38G6a+HitJ69evT9ZP5Mtgn6zcfcw/GLbsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiD

```
    "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"
      ]
     }
    ]
   }
  ]
}
```