

```
{
  "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": "-NoTriNEPBWI",
    "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+/AAADh0RVh0U29mdHdhcmUAAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjlsIGh0dHA6Ly9tYXRwbG90bGliLm9y
      Zy+WH4yJAAAN9kIEQVR4nO3df4xV9ZnH8c+zWP6QojBrOhKKSyEGg8ZON4gbl6w1hvojGhw1TSexoZE4/YNJaLlhNe
      wf1WwwZBU2SzTNTKMWNl1qEzUgaQouoOzGhDgiKo5LdQ2mTEaowZef/mCHefaPezBTnfu9w7nn3nOZ5/1Kbu69
      57nnnncnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAuaubCzlxT/0CDubuNN72uLbuZ3Wpmh8zsPTN7sJ
      7vAtBYlvc6u5lNkfRHSUsIHZH0qqQudx9IzMOWHwiwRmzZF0t6z93fd/czkn4raVkd3weggeol+2xJfXrz/kg27S+YWbe
      Z9ZtZfx3LAlCnhp+gc/c+SX0Su/FamerZsg9KmJpm/bezaQBaUD1hf1XSIWb2HTObKulHkrYV0xaAouXejXf3ETPrkbRD0
      hRJT7n724V1BqBQuS+95VoYx+xAwzXkRzUALhyEHQICsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdi
      Alwg4EQdiBIAg7EARhB4lg7EAQhB0lgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBJF7yGZcG
      KZMmZKsX3rppQ1dfk9PT9XaxRdfnJx3wYlFyfrKlSUt9ccee6xqraurKznv559/nqyvW7cuWX/44YeT9TLUFXYzOyzppKS
      zkkbcfVERTQEOxHfB9pvc/aMCvgdAA3HMDgRRb9hd0k4ze83Musf7gJl1m1m/mfXXuSwAdah3N36Juw+a2bckvWh
      m/+Pue8d+wN37JPVJkpl5ncsDkFNdW3Z3H8yej0l6XtLiIpoCULzCYTezaWY2/dxrST+QdLCoxgAUq57d+HZJz5vZue/5
      D3f/QyFdTTJXXHFFsj516tRk/YybbkjWlyxZUrU2Y8aM5Lz33HNpSl6mI0eOJOsbN25M1js7O6vWTP48mZz3jTfeSNZff
      vnlZL0V5Q67u78v6bsF9gKggbj0BgRB2IEgCDsQBGEHgiDsQBDM3rWftU3WX9B1dHQk67t3707WG32baasaHR1N1u
      +///5k/dSpU7mXPTQ0IKx//PHHyfqhQ4dyL7vR3N3Gm86WHQICsANBEHYgCMIOBEHYgSAIOxAEYQeC4Dp7Adra2pL
      1ffv2Jevz5s0rsp1C1ep9eHg4Wb/pppuq1s6cOZOcn+rvD+rFdXYgOMIOBEHYgSAIOxAEYQeCIOxAEIQdCIhmwtw/Pjx
      ZH316tXJ+h133JGsv/7668l6rT+pnHLgwlfKfenSpcn66dOnk/Wrr766am3VqIXJeVEstuxAEIQdCIKwA0EQdiAlwg4EQdi
      BIAG7EAT3s7eASy65JfMvNbxwb29v1dqKFSuS8953333J+pYtW5J1tJ7c97Ob2VNmdszMDo6Z1mZmL5rZu9nzzCKbB
      VC8iezG/1rSrV+Z9qCkXe5+paRd2XsAlaxm2N19r6Sv/h50maRN2etNku4quC8ABcv72/h2dz83WNaHktqrfdDMuiV1
      51wOgILUfSOMu3vqxJu790nqkzhBB5Qp76W3o2Y2S5Ky52PFtQSGfKGfZuk5dnr5ZK2FtMOgEapuRtvZlskfV/SZWZ2
      RNIvJK2T9DsZWyHpA0k/bGSTk92JEyfqmv+TTz7JPe8DDzyQrD/zzDPJeq0x1tE6aobd3buqIG4uuBcAdcTPZYegCDsQB
      GEHgiDsQBCEHQiCW1wngWnTPlWtvfDCC8l5b7zxxmT9tttuS9Z37tyZrKP5GLIZCI6wA0EQdiAlwg4EQdiBIAG7EARhB
      4LgOvskN3/+GR9//79yfrw8HCyvmfPnmS9v7+/au2JJ55lztvMf5uTCdfZgeAlOxAEYQeCIOxAEIQdCIKwA0EQdiAlrrM
      H19nZmaw//fTTYfr06dNzL3vNmjXJ+ubNm5P1oaGhZD0qrrMDwRF2IAjCDgRB2IEgCDsQBGEHgiDsQBBC20fSNddck6
      xv2LAhWb/55vyD/fb29ibra9euTdYHBwdzL/tClvs6u5k9ZWbHzOzgmGkPmdmgmR3IHrcX2SyA4k1kN/7Xkm4dZ/q/u
      ntH9vh9sW0BKFrNsLv7XknHm9ALgAaq5wRdj5m9me3mz6z2ITPrNrN+M6v+x8gANFzesP9S0nxJHZKJK2v9kF373P
      3Re6+KOeyABQgV9jd/ai7n3X3UUm/krS42LYAFC1X2M1s1pi3nZIOVvssgNZQ8zq7mW2R9H1Jl0k6KukX2fsOSS7psK
      SfunvNm4u5zj75zJgxl1m/8847q9Zq3StvNu7l4i/t3r07WV+6dGmyPIlVu85+0QRm7Bpn8pN1dwSgqfi5LBAEYQeCIOx
      AEIQdCIKwA0FwiytK88UXxyTrF12Uvlg0MjKsRn9yyy1Vay+99Fjy3gsZf0oaCI6wA0EQdiAlwg4EQdiBIAG7EARhB4Koe
      dcbYrv22muT9XvwTdZv+6666rWal1Hr2VgYCBZ37t3b13fP9mwZQeCIOxAEIQdCIKwA0EQdiAlwg4EQdiBIlJOPsktW
      LAgWe/p6UnW7777mT98ssvP++eJurs2bPJ+tBQ+q+Xj46OfnOBY8tOxAEYQeCIOxAEIQdCIKwA0EQdiAlwg4EwXX
```

2C0Cta9ldXeMnTfR6zr63Llz87RUIp7+/mR97dq1yfq2bduKbGfSq7lIN7M5ZrbHzAbM7G0zW5VNbzOzF83s3ex5ZuP
bBZDXRHbjRyT9o7svlPR3klaa2UJJD0ra5e5XStqVvQfQomqG3d2H3H1/9vqkpHckzZa0TNKm7GOBJN3VqCYB1O+8jt
nNbK6k70naJ6nd3c/9OPIDSe1V5umW1J2/RQBfMpdZedP7pqRnJf3M3U+MrXlIdMhxB2109z53X+Tui+rqFEBdJhR2
M/uGKKH/jbs/I00+amazsvosScca0yKAltTcjTczk/SkpHfcfcOY0jZJyyWty563NqTDSaC9fdwjnC8tXLgwWX/88ceT9auu
uuq8eyrKvn37kvVHH320am3r1vQ/GW5RLdEjtn/XtKPJb1lZgeyaWtUCfnvzGyFpA8k/bAxLQlOqs2wu/t/Sxp3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMetrPU1tZWtdbb25uct6Ojl1mfN29erp6K8MorryTr69evT9Z37NiRrH/2
2Wfn3RMagy07EARhB4lg7EAQhB0lgrADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHHV2uzZs3P1VJRPP/20am3jxo3
JeR955JfK/fTp07l6QuthyW4EQdiBIAg7EARhB4lg7EAQhB0lgrADQY5zt7Z2VIXvR4DAwPJ+vbt25P1kZGRZD11z/nw8
HByXsTBlh0lgrADQRB2IAjCDgRB2IEgCDsQBGEHgiB3T3/AbI6kzZLaJbmkPnf/NzN7SNIDkv6cfXSNu/++xnelFwagbu4+
7qjLEwn7LEmz3H2/mU2X9Jqku1QZj/2Uuz820SYIO9B41cl+kfHZhyQNZa9Pmtk7ksr90yWaztt5HbOb2VxJ35O0L5vU
Y2ZvmtlTZjazyjzdZtZvZv11dQqgLv347/8oNk3Jb0saa27P2dm7Zl+UuU4/p9V2dW/v8Z3sBsPNFjuY3ZJMrNvSNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCclvTM26NmJu3M6JR2st0kAjTORs/FLJP2XpLckjWaT10jqktShym78YUk/
zU7mpb6LLTvQYHXtxheFsAONI3s3HsDkQNiBIAg7EARhB4lg7EAQhB0lgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBC
EHQii2UM2fyTpgzHvL8umtaJW7a1V+5LoLa8ie/ubaoWm3s/+tYWB9bv7otlaSGjV3lq1L4ne8mpWb+zGA0EQdiClssP
eV/LyU1q1t1btS6K3vJrSW6nH7ACap+wtO4AmlexAEKWE3cxuNbNDZvaemT1YRg/VmNIhM3vLzA6UPT5dNobeMT
M7OGZam5m9aGbvZs/jjrFXUm8Pmdlgtu4OmNntJfU2x8z2mNmAmb1tZquy6aWuu0RfTVlvTT9mN7Mpkv4oaamkl
5JeldTl7gNNbaQKMzssaZG7l/4DDDP7B0mnJG0+N7SWmf2LpOPuvi77j3Kmu/+8RXp7SOc5jHeDeqs2zPhPVOK6K3L
48zzK2LlVlvSeu7/v7mck/VbSshL6aHnuvlfS8a9MXiZpU/Z6kyr/WJquSm8twd2H3H1/9vqkpHPDjJe67hJ9NUUYZ8t6
U9j3h9Ra4337pJ2mtlrZtZddjPjaB8zzNaHktrLbGYcNYfxbqavDDPeMusuz/Dn9eIE3dctcfe/IXSbpJXZ7mpl8soxWctdO
/2lpPmqjAE4JGI9mc1kw4w/K+ln7n5ibK3MdTdOX01Zb2WEfVDSnDHvv51NawnuPpg9H5P0vCqHhA3k6LkRdLPnYy
X38yV3P+ruZ919VNkvVOK6y4YZf1bSb9z9uWxy6etuvL6atd7KCPurkq40s++Y2VRJP5K0rYQ+vsbMpmUnTmRm0yT9
QK03FPU2Scuz18slbS2xl7/QKsN4VxtmXCWvu9KHP3f3pj8k3a7KGfn/lfrPZfRQpa95kt7IHm+X3ZukLars1v2fKuc2Vkj
6a0m7JL0r6T8ltbVQb/+uytDeb6oSrFkl9bZEIV30NyUdyB63l73uEn01Zb3xc1kgCE7QAUEQdiAlwg4EQdiBIAg7EARhB
4lg7EAQ/w8ie3GmjcGk5QAAAABJRU5ErkJggg==\n"

},

"metadata": {

"needs_background": "light"

}

},

{

"output_type": "stream",

"name": "stdout",

"text": [

"5\n"

]

}

]

},

{

"cell_type": "code",

"source": [

"print (\nShape 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)

```

```

    "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.]]\n",

```

```
"\n",
"\n",
"----- Prediction ----- \n",
"\n",
"\n"
]
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxl
B0t1+/AAAADh0RVh0U29mdHdhcmUAAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjlsIGh0dHA6Ly9tYXRwbG90bGliLm9y
Zy+WH4yJAAAOFOIEQVR4nO3dcYxV5ZnH8d8jW4xKlagpTkRr2+AfzUYHQUKypri2bVw0gcakQozDpk2GxJJQszGr3VF
lamNjIEZNE6VFFCqqGjBpi51GaLdmDSOyCpqW1mDFhwZUSNDTKTCs3/cQzPinPcM9557z4Hn+0km997zzLn38TI/z7
nnPfe85u4CcPI7peoGAHQGYQeCIOxAEIQdCIKwA0H8QydfzMW49A+0mbvbWMtb2rKb2ZVm9mcz22VmN7fyXADa
y5odZzezCZL+luk7kvZlelHSYnd/PbEOW3agzdqxZZ8jaZe7v+XuhyStl7SghecD0EathP1cSX8d9XhPtuxzzKzXzAbNbLCF
1wLQorYfoHP3fkn9ErvxQJVa2bLvIXTeqMfTs2UAaqiVsL8oaYaZfc3MJkpaJGlzOW0BKfVtu/Hu/pmZLZO0RdlESWvc/
bXSOgNQqqaH3pp6MT6zA23XlpNqAJw4CDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQd
CIKwA0EQdiAlwg4EQdiBIAg7EARhB4lg7EAQhB0loqNTNuPkM2vWrGR92bJlubWenp7kug8//HCyft999yXr27dvT9aj
YcsOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EwiYuSuru7k/WBgYFkffLkyWW28zkff/xxsn7WWWe17bXrLG8W15ZOqj
Gz3ZJGJB2W9Jm7z27l+QC0TxlnOp2zu+8v4XkAtBGf2YEgWg27S/q9mb1kZr1j/YKZ9ZrZojKntvhaAFrQ6m78Ze6+18y
+lulZM/uTuz8/+hfcvV9Sv8QBOqBKLW3Z3X1vdjss6SljC8poCkD5mg67mZ1hZl8+el/SdyXtLKsxAOVqZTd+mqSnzOzo
8/za3f+rIK7QMXPmpHfGNm7cmKxPmTlIWU+dxzEyMpJc99ChQ8l60Tj63Llzc2tF33Uveu0TUdNhd/e3JF1cYi8A2oih
NyAlwg4EQdiBIAg7EARhB4lgK64ngdNPPz23dskllyTXfeSRR5L16dOnJ+vZ0Guu1N9X0fDXnXfemayvX78+WU/11tfXl
1z3jjvuSNbrLO8rrmzZgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAlpmw+CTzwwAO5tcWLF3ewk+NTdA7ApEmTkxNnns
uWZ83b15u7aKLLkquezJiyw4EQdiBIAg7EARhB4lg7EAQhB0lgrADQTDOfgKYNWtWsn7VVVfl1oq+b16kaCz76aefTtb
vuuuu3Nq7776bXPfll1901j/66KNk/Yorrsittfq+nlyYsgNBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEFw3vga6u7uT9YGBg
WR98uTJTb/2M888k6wXfR/+8ssvT9ZT3xt/8MEHk+u+//77yXqRw4cP59Y++eST5LpF/11F17yvUtPXjTezNWY2bGY7
Ry0708yeNbM3s9upZTYLoHzj2Y3/laQrj1l2s6St7j5D0tbsMYAaKwy7uz8v6cNjFi+Qtda7v1bSwpL7AlCyZs+Nn+buQ9
n99yRny/tFM+uV1Nvk6wAoSctfhHF3Tx14c/d+Sf0SB+iAKjU79LbPzLokKbsdLq8IAO3QbNg3S1qS3V8iaVM57QBol8J
xdjN7VNI8SWdL2idphaTfSHpM0vmS3pb0fXc/9iDeWM8Vcjf+wgsVtNZXRfRrC9atChZ379/f25taGgotyZjt99+e7L+xB
NPJot1lhpnL/q737BhQ7J+3XXXNdVTJ+SNsxd+Znf3vLMqvt1SRwA6itNlgSAIOxAEYQeCIOxAEIQdCIJLSZfg1FNPTdZTI
1OWpPnz5yfrlyMjyXpPT09ubXBwMLnuaaedlqxHdf7551fdQunYsgNBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIlyl2Dm
zJnJetE4epEFCxYk60XTKgMSW3YgDMIOBEHYgSAIOxAEYQeCIOxAEIQdCIJx9hKsWrUqWTcb88q+f1c0Ts44enNOOS
V/W3bkyJEODlIPbNmBIAg7EARhB4lg7EAQhB0lgrADQRB2IAjG2cfp6quvzq11d3cn1y2aHnjz5s1N9YS01Fh60b/Jjh07
ym6ncoVbdjNbY2bDZrZz1LKVZrbXzHZkP61dnQFA241nN/5Xkq4cY/kv3L07+/lduW0BKfth2N39eUkfdqAXAG3UygG
6ZWb2SrabPzXvl8ys18wGzSw96RiAtmo27KsIfUNSt6QhSXfn/aK797v7bHef3eRrASHBU2F3933uftjdj0j6paQ55bYFo
GxNhd3MukY9/J6knXm/C6AeCsfZzexRSfMknW1meyStkDTPzLoluaTdkpa2scdaSM1jPnHixOS6w8PDyfqGDRua6ulk
VzTv/cqVK5t+7oGBgWT9lItuafq566ow7O6+elzFD7WhFwBtxOmyQBCEHQiCsANBEHYgCMIOBMFXXDvg008/TdaHh
oY61Em9FA2t9fx1Jes33XRTsr5nz57c2t135570Kuk6ePBgsn4iYssOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0Ewzt4BkS8
VnBrMdtE4+bXXXpusb9q0KVm/5pprkVVo2LIDQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHs4+TmTVV6SFCxcm68uXL
```

2+qpzq48cYbk/Vbb701tzZlypTkuuvWrUvWe3p6knV8Hlt2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCcfZxcvemapJ0zjn
nJOv33ntvsr5mzZpk/YMPPsitzZ07N7nu9ddfn6xffPHFyfr06dOT9XfeeSe3tmXLLuS6999/f7KO41O4ZTez88xsm5m9b
mavmdnybPmZZvasmb2Z3U5tf7sAmjWe3fjPJP2bu39T0lxJPzKzb0q6WdJWd58haWv2GEBNFYbd3YfcfXt2f0TSG5LO
lbRA0trs19ZKSp8TCqBSx/WZ3cwukDRT0h8ITXP3o5OUvSdpWs46vZJ6m28RQBnGfTTezCZJ2ijpx+5+YHTNG0eoxjxK
5e797j7b3We31CmAlowr7Gb2JTWcvs7dn8wW7zOzrqzeJWm4PS0CKEPhbrw1vr/5kKQ33H3VqNJmSUsk/Ty7TV/X
N7AJEyYk6zfccEOyXnRJ5AMHDuTWZsyYkVy3VS+88EKyvm3bttzabbfdVnY7SBjPZ/Z/knS9pFfNbEe27CdqhPwxM/uh
pLclfb89LQIoQ2HY3f1/JOVdneHb5bYDoF04XRYlgrADQRB2IAjCDgRB2IEgrOjrmaW+mFnnXqxxqa9yPv7448l1L7300
pZeu+h51a38G6a+HitJ69evT9ZP5Mtgn6zcfcw/GLbsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAE4+wl6OrqStaXLI2arP
f19SXrrYyz33PPPcl1V69enazv2rUrWUf9MM4OBEfYgSAIOxAEYQeCIOxAEIQdCIKwA0Ewzg6cZBhnB4lj7EAQhB0lgrA
DQRB2IAjCDgRB2IEgCsNuZueZ2TYze93MXjOz5dnylWa218x2ZD/z298ugGYVnlRjZl2Sutx9u5l9WdJLkhaqMR/7QXe/
a9wvxkk1QNvlnVQznvnZhyQNZfdHzOwNSeeW2x6Adjuz+xmdoGkmZL+mC1aZmavmNkaM5uas06vmQ2a2WBLn
QJoybjPjTezSZKek/Qzd3/SzKZJ2i/JJf1UjV39HxQ8B7vxQJvI7caPK+xm9iVJv5W0xd1XjVG/QNJv3f0fC56HsANT1vQXYa
xxadOHJL0xOujZgbujvidpZ6tNAmif8RyNv0zSHyS9KulltvgnkhZL6lZjN363pKXZwbzUc7FIB9qsp34shB2oP34PjsQHG
EHgiDsQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiClwgtOlmy/pLdHPT47W1ZHde2trn1J
9NasMnv7al6ho99n/8KLmw26++zKGkioa2917UuitZ21qjd244EgCDsQRNVh76/49VPq2ltd+5LorVkd6a3Sz+wAOqfq
LTuADiHsQBCVhN3MrjSzP5vZLjO7uYoe8pjZbjN7NZuGutL56bI59IbNbOeoZWea2bNm9mZ2O+YcexX1VotpvBPTjFf
63IU9/XnHP7Ob2QRJf5H0HUI7JL0oabG7v97RRnKY2W5Js9298hMwzOxbkg5Kevjo1FpmdqekD93959n/KKe6+7/Xp
LeVOs5pvNvUW9404/+qCt+7Mqc/b0YVW/Y5kna5+1vufkjSekLkuij9tz9eUkfHrN4gaS12f21avyxdFxOb7Xg7kPuvj2
7PyLp6DTjlb53ib46ooqwnyvr6Me71G95nt3Sb83s5fMrLfQzSfYwbdQ0W+9JmI2IM2MonMa7k46ZZrw2710z05+3ig
N0X3SZu18i6V8k/SjbXa0lb3wGq9PY6Wpj31BjDsAhSXdx2Uw2zfhGST929wOja1W+d2P01ZH3rYqw75V03qjH07Nlt
eDue7PbYUIPqfGxo072HZ1BN7sdrriFv3P3fe5+2N2PSPqIKnzvsmnGN0pa5+5PZosrf+/G6qtT71sVYX9R0gwz+5qZTzS
0SNLmCvr4AjM7lztwljM7Q9J3Vb+pqDdLWpLdXyJpU4W9fE5dpvHOM2ZcFb93IU9/7u4d/5E0X40j8v8n6T+q6CGnr6
9L+t/s57Wqe5P0qBq7dX9T49jGDyWdJWmrpDcl/bekM2vU23+qMbX3K2oEq6ui3i5TYxf9Fuk7sp/5Vb93ib468r5xui
wQBAfогCAIOxAEYQeCIOxAEIQdCIKwA0EQdiCl/wcl826NkY1TiQAAAABJRU5ErkJggg==\n"

},

"metadata": {

"needs_background": "light"

}

},

{

"output_type": "stream",

"name": "stdout",

"text": [

"\n",

"\n",

"Final Output: 0\n"

]

}

]

},

{

"cell_type": "code",

```
"source": [  
  "metrics=model.evaluate(X_test,y_test,verbose=0)\n",  
  "print(\"Metrics(test loss and Test Accuracy):\")\n",  
  "print(metrics)"  
],  
"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": [  
      "Metrics(test loss and Test Accuracy):\n",  
      "[0.07461030036211014, 0.9771999716758728]\n"  
    ]  
  }  
]  
}
```