

```
{
  "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+/AAAADh0RVh0U29mdHdhcmUAAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjlsIGh0dHA6Ly9tYXRwbG90bGliLm9y
      Zy+WH4yJAAAN9kIEQVR4nO3df4xV9ZnH8c+zWP6QojBrOhKKSyEGg8ZON4gbl6w1hvojGhw1TSexoZE4/YNJaLlhNe
      wf1WwwZBU2SzTNTKMWNl1qEzUgaQouoOzGhDgiKo5LdQ2mTEaowZef/mCHefaPezBTnfu9w7nn3nOZ5/1Kbu69
      57nnnncnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAuaubCzlxT/0CDubuNN72uLbuZ3Wpmh8zsPTN7sJ
      7vAtBYlvc6u5lNkfRHSUsIHZH0qqQudx9IzMOWHwiwRmzZF0t6z93fd/czkn4raVkd3weggeol+2xJfxyz/kg27S+YWbe
      Z9ZtZfx3LAlCnhp+gc/c+SX0Su/FamerZsg9KmJpm/bezaQBaUD1hf1XSIWb2HTObKulHkrYV0xaAouXejXf3ETPrkbRD0
      hRJT7n724V1BqBQuS+95VoYx+xAwzXkRzUALhyEHQICsANBEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdi
      Alwg4EQdiBIAg7EARhB4lg7EAQhB0lgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBCEHQiCsANBEHYgCMIOBJF7yGZcG
      KZMmZKsX3rppQ1dfk9PT9XaxRdfnJx3wYlFyfrKISuT9ccee6xqraurKznv559/nqyvW7cuWX/44YeT9TLUFXYzOyzppKS
      zkkbcfVERTQEOxHfB9pvc/aMCvgdAA3HMDgRRb9hd0k4ze83Musf7gJl1m1m/mfXXuSwAdah3N36Juw+a2bckvWh
      m/+Pue8d+wN37JPVJkpl5ncsDkFNdW3Z3H8yej0l6XtLiIpoCULzCYTezaWY2/dxrST+QdLCoxgAUq57d+HZJz5vZue/5
      D3f/QyFdTTJXXHFFsj516tRk/YybbkjWlyxZUrU2Y8aM5Lz33HNpSl6mI0eOJOsbN25M1js7O6vWTP48mZz3jTfeSNZff
      vnlZL0V5Q67u78v6bsF9gKggbj0BgRB2IEgCDsQBGEHgiDsQBDM3rwtU3WX9B1dHQk67t3707WG32baasaHR1N1u
      +///5k/dSpU7mXPTQ0IKx//PHHyfqhQ4dyL7vR3N3Gm86WHQICsANBEHYgCMIOBEHYgSAIOxAEYQeC4Dp7Adra2pL
      1ffv2Jevz5s0rsp1C1ep9eHg4Wb/pppuq1s6cOZOcn+rvD+rFdXYgOMIOBEHYgSAIOxAEYQeCIOxAEIQdCIHmwtw/Pjx
      ZH316tXJ+h133JGsv/7668l6rT+pnHLgwlfKfenSpcn66dOnk/Wrr766am3VqIXJeVEstuxAEIQdCIKwA0EQdiAlwg4EQdi
      BIAg7EAT3s7eASy65JfMvNbxwb29v1dqKFSuS8953333J+pYtW5J1tJ7c97Ob2VNmdszMDo6Z1mZmL5rZu9nzzCKbB
      VC8iezG/1rSrV+Z9qCkXe5+paRd2XsAlaxm2N19r6Sv/h50maRN2etNku4quC8ABcv72/h2dz83WNaHktqrfdDMuiV1
      51wOgILUfSOMu3vqxJu790nqkzhBB5Qp76W3o2Y2S5Ky52PFtQSGfKGfZuk5dnr5ZK2FtMOgEapuRtvZlskfV/SZWZ2
      RNlVJK2T9DsZWyHpA0k/bGSTk92JEyfqmv+TTz7JPe8DDzyQrD/zzDPJeq0x1tE6aobd3buqIG4uuBcAdcTPZYegCDsQB
      GEHgiDsQBCEHQiCW1wngWnTPlWtvfDCC8l5b7zxxmT9tttuS9Z37tyZrKP5GLIZCI6wA0EQdiAlwg4EQdiBIAg7EARhB
      4LgOvskN3/+GR9//79yfrw8HCyvmfPnmS9v7+/au2JJ55lztvMf5uTCdfZgeAlOxAEYQeCIOxAEIQdCIKwA0EQdiAlrrM
      H19nZmaw//fTTYfr06dNzL3vNmjXJ+ubNm5P1oaGhZD0qrrMDwRF2IAjCDgRB2IEgCDsQBGEHgiDsQBbcZ0fSNddck6
      xv2LAhWb/55vyD/fb29ibra9euTdYHBwdzL/tClvs6u5k9ZWbHzOzgmGkPmdmgmR3IHrcX2SyA4k1kN/7Xkm4dZ/q/u
      ntH9vh9sW0BKFRNsLv7XknHm9ALgAaq5wRdj5m9me3mz6z2ITPrNrN+M6v+x8gANFzesP9S0nxJHZKJK2v9kF373P
      3Re6+KOeyABQgV9jd/ai7n3X3UUm/krS42LYAFC1X2M1s1pi3nZIOVvssgNZQ8zq7mW2R9H1Jl0k6KukX2fsOSS7psK
      SfunvNm4u5zj75zjgl1m/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/GW5RLdZEjtn/XtKPJb1lZgeyaWtUCfnvzGyFpA8k/bAxLQlOqs2wu/t/Sxp3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMetrPU1tZWtdbb25uct6Ojl1mfN29erp6K8MorryTr69evT9Z37NiRrH/2
2Wfn3RMagy07EARhB4lg7EAQhB0lgrADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHHV2uzZs3P1VJRPP/20am3jxo3
JeR955JfK/fTp07l6QuthyW4EQdiBIAg7EARhB4lg7EAQhB0lgrADQY55zt7Z2VIXvR4DAwPJ+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
U9j3h9Ra4337pJ2mtlrZtZddjPjaB8zzNaHktrLbGYcNYfbqavDDPeMusuz/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": [],  
  "metadata": {  
    "id": "WHK4rCVBC0dJ"  
  },  
  "execution_count": 9,  
  "outputs": []  
}  
]  
}
```