```
{
 "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+zWP6QojBrOhKKSyEGg8ZON4gbl6w1hvojGhw1TSexoZE4/YNJaLIhNe
wf1WwwZBU2SzTNTKMMWNl1qEzUgaQouozGhGTaW9mz7nbETJMYEEaalLHHTx3kmzUGsDbNGXfiejpQxhBtaFTybtcaZx/eEaVMZyPUT5aDHVhTfsBlMG
57nnnicnfDi/7pmvubsATH5/VXYDAJqDsANBEHYgCMIOBEHYgSAuaubCzIxT/0CDubuNtr6uPbuZ3WBmfzCzd83s3nqWBaCxrNbr7GY2QdIfJX1PkQTBY0W/iwRmzZF0t6rxz93fd/czkn4jaWFaTZ7b6zdMKnvbM/Cy1u1oVaS0hOj8T4lj4lcUUJCTkU1DM9Q14t06szvmBdHZ1r8cNpyIhsbUQ/iyw56iLeYp28aD+WRuFeiWNnj5ixkkQR1cGZL9+i6HNfgEZN6PUFc0Jdjh5jBDNK7rGQnA0GuhTW77VVaAxL9KHH/jvUt5jd+T7xSTkQ7vQF0F8wdLWw/IHCKJXtUh+CY5m5Th2w6v88Ctpwq6TN0tE4pvEjS3v+TKN7VqdsRvSV1jpGo3L17DA+fm7T6MPOEXhJ7sB6OAZC08XkdkXVzA/eJPJ4X77TqvPwiZrHNQ1DuNHDO3qcwkROBxoCHUYNG5QXBa9jstpOXb4eyg6Ubxfv9lDRHLxHCAoczalaZTFQXvKMvIkgiGBHXEXD3XYzDpoUXQsnxxnpJG0EjrE/OLxuwLHp7BoUBnxE0jPqtLH0Dx2yo9HSx3dVLS/RgcS9unMRKrHqEz4XDX33vZ/5zz8Ktk6DSfnNTrLV88eYxn7fAvTLR0MfvB+4WnDKJ8Jtd/WzsQgrrsj6s6BChbMmNFHB4tPCTsNZ3CyjTr/YvfA+ngeHWn+9+bUlS/o2QiXuCR6Yz7VgU+XMdd2mmdV+3KTCs/dYkgQdfY/eIrbGvH8NzC+Q1gWCuu2kk9zqfvw85+ooR0hJc5+ry4pdE3pc5Iic47e/M/wmq8Ak6iC0O3+AvBzC85gIuLT/6RJfuoA+HLuyhFTdM1HhI3+HAxoKZ5G9XAPtNXkSJ/xvvXzz3JhINvd8VeW0WBvHtLkYjkZuEV8EwFtivmJ3CGwyZ3LJb/4zZkqUgWH9iAHE1Gb5u9jUKN+VQ+2DCS35AF6kDS3mDSRm6ObFzKnvuhJyKQHtFrdt+jndBFsXjcVVXrU1SbBfzKHL6jSgthcQMfrjRNFPf7j/mhEPbxn2vOAX5Lha8pcR5ffpRnePh2h7d/7Gn0cWSDkvCHX0cmybVcIWlD1Nug5F6gn5V+/RFA6VJS6W/4Nvm0yjZD2tzOlvTjGgjmLrLrQ4h05qVMmfTzPlaZOp7B+gHb0KVw9Vm1KbmwcZs9IDLSLG0U9OVPLrUx3kZqxx+TnfSzOr9Kk/jG+KbAZlMxzwjU6Ta1K4rD7ISOuT9Di7uDh7SDLhdowB4BPDYpz9jsivOYZ0p4Ya2eI12U8Y9sV4T3KbVQC0LFaGYjWkHq67DJ/9PZxl/PqUU2J8gFukSQgLOeNNpOq/2wgTn8Rr4oKnq4uEHLTfVuLiGMekyyTzRqDp9YaZ4rzZqP3SXhKO0Q/kTeFfUWqbw0UT/+nw7WrkZyZXxDwxpAWRsOETfYbRdO1iMZymORdzGu+ZNXNgeaE/uJ0IixX5b/GmymZdbGLOGZIPAKV8dBN/xkl+fcufywjCtN+7VOrUeRJHJ0YzuPYSPhxc0hvIUUbkBG57GZnv5/QRuJk3kD1jrrMLPwGwPsBKLZ31JFm1XNzKyZsNGOkNwKpWYVZpy4qnXxLQ0VQAfoovRBRuzO4IUPQz4pxwMmrlFh1GgZdngfnfLYs+fNLT8VXOuzxf9BO/ti+pGjOM+Lmwdn3HFnWZCa9aaeQfAXT9xmTRwUoXgR3EntsI4uiQpZ7wH/W2G4fOMeGlqAigbtP2e2CELQrf3bbQpTRaK8tpcK6XyOVNB6hdfK0dwOzw8eBn8rOhNLTnzL2VS8uRyW8JqsC+G/HKwSeJKxBPTSA9X1cZ6Z7Y0qq/Tv//rZWKtkrA0HRtk17AvMGZ0V4ZUQ4oe2+jZNYcqHoIFzhDX6sLbkzZw8Fz8eJ6/rcbfXgTdEPAqPVv63BOJsJ5WpzvkBbHtfGCbEnN4O6wnYuUJa5bCG8MuIuFKGAiM9Li95gNCbMoBtDkb9jrIk2j4y+/6wVd9zj4rqxqd69p2KtbqyEHZKdTWyC6XIV9eN4sWoRU8DgiHoA8fFGIMPl6Z5uKFYA6QRpmcbJ3T6W4nsXNdq2YbZGeYbSCXvRrKk1E+xjOYOrzYHHtHc/Z2Zh9t6X6Rg2UVAoOVyMnHrx3DSV4J0oGKx5xqcL8jvFdXTq7Rlnb8FEZbFaNjqZn2LNMQiiV7Z96AGH0yXsfHpxuFY2NGNRNFhLUrCDmzYQFcfx2uOXD8X1yp5iVJGyMZLDS2ppLmXMt2DXJUDu9fBTJoHeMQBtjPHvKvR/yUUuaHy/eiC45S5X9IMRMvXCPuu6tBAZyuxlc3RDEghMQbSTLK0wRlx7b8dgBFUNEgLhxNzCqq52nE1yu84SvyO7ym4VcVKNGQN6d4aSFYfhYCFUxGXOR9ZCFRzTwQ9Yr64e4GPLN8bBoPg6VXiylB4uCbbM1t1+6Ss+SUmiqCl8RaFiI/SakgqaJy2w7NzPRq1gs3B8iVxOeN7h4h6UoJOPDv1S7l+4GgKq4tUDaQs+EktqYFqPdyoASXhqFqJKQjhSvHhXkhSvKPx8WgOkpeJbfDHd6OiHAhp1EKsMkFlIQD8ULpadiDiCAC6lBm/RRLDNbmQjE+U5o/0/c9XKm4OIGUH1mGMXyYqlZ1UG5eSd5I5xdtyTuNq9bmyy5egK9p9kIGSMJIBg+C7BglWfyYRw7SpDZ5ZLEF0LpBg/Nu+IOKOgsmhxhQs5kq5n4tyIA2vTqDUsq3l5Ng+o6+hBMxxN4V6ewz+P5dbc0fxDWlJ4mnkGA7WjZzUdfk1PFpP9Ijr2wGAHLa5xEjHmF4+4OWVDMmd7jiOymAjAEP5Zm4c/hF86GvlPJLnIH8r4UeF99GO5gtTnHUD2EvNcxEzeU8NDXKULzRiwVyWHmh6HtwIC6vQGJhVU7zQzcw8K6lCtMLy1TsUCpJHc6KexEIHpehfBB1JwLRiq/p3pXpRWrr6WRt5fCoR6QyD8ld8eBmZp6tXm6qbd8mZzRNZ2Bk+ppo6gXdjRcBhvhKDbHYRx2PAUaMFoqlAw7PBZ/Ya9ERPXmyVdU7j6djMnVRQhVmvNngNeBl0btKPzm4EZgAWqFp/Yi5Vs88cftoBrCNpLDVfGrO6IoJ/3b76fTGmnQngwzlhFuGfPaLxXN1zqKtvnV3Gg0JYiSzJcNgCC8XeNMqYOFRVTT/eBF3ofdHXNSsExO7xWOrkHtcFWHk1Hb8NQHRBCfU8KZX1xKmG0i1pGzVBFnPG/wsiNW3w8tsB7d+GmmCLb1aPM5KT7tB
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxI

2C0Cta9ldXeMNtFtR6zr63Llz87RUiP7+/mR97dq1yfq2bduKbGfSq7llN7M5ZrbHzAbM7G0zW5VNbzOzF83s3ex5ZuP
bBZDXRHbjRyT9o7svlPR3klaa2UJJD0ra5e5XStqVvQfQomqG3d2H3H1/9vqkpHckzZa0TNKm7GObJN3VqCYB1O+8jt
nNbK6k70naJ6nd3c/9OPlDSe1V5umW1J2/RQBFmPDZeDP7pqRnJf3M3U+MrXlldMhxB2109z53X+Tui+rqFEBdJhR2
M/uGKkH/jbs/l00+amazsvosScca0yKAItTcjTczk/SkpHfcfcOY0jZJyyWty563NqTDSaC9fdwjnC8tXLgwWX/88ceT9auu
uuq8eyrKvn37kvVHH320am3r1vQ/GW5RLdZEjtn/XtKPJb1lZgeyaWtUCfnvzGyGyFpA8k/bAxLQIoQs2wu/t/Sxp3cHdJN
xfbDoBG4eeyQBCEHQiCsANBEHYgCMIOBMEtrhPU1tZWtdbb25uct6Oj1mfN29erp6K8MorryTr69evT9Z37NiRrH/2
2Wfn3RMagy07EARhB4Ig7EAQhB0IgrADQRB2IAjCDgQR5jr79ddfn6yvXr06WV+8eHHV2uzZs3P1VC2zzZs3P1VJRPP/20am3jxo3
JeR955JFk/fTp07l6Quthyw4EQdiBIAg7EARhB4Ig7EAQhB0IgrADQYS5zt7Z2VlXvR4DAwPJ+vbt25P1kz/nw8
HByXsTBlh0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgjB3T3/AbI6kzZLaJbmkPnf/NzN7SNIDkv6cfXSNu/++xnelFwagbu4+
7qjLEwn7LEmz3H2/mc2QdDIpu1QZj/2UVOrJ/4UZj/2Uuz820SYIO9B41cI+kfHZhyQNZa9Pmtk7ksr90ywAztt5HbOb2VxJ35O0L5vU
Y2ZvmtlTZjazyjzdZtZvZv11dQqgLjV347/8oNk3Jb0saa27P2dm7ZI+UuU4/p9V2dW/v8Z3sBsPNFjuY3ZJMrNvSNouaY
e7bxinPlfSdne/psb3EHagwaqFveZuvJmZpCclvTM26NmJu3M6JR2st0kAjTORs/FLJP2XpLckfZHNXpLckjWaT10jqktShym78YUk/
zU7mpb6LLTvQYHXtxheFsAONl3s3HsDkQNiBIAg7EARhB4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgCDsQBGEHgiDsQBC
EHQii2UM2fyTpgzHvL8umtaJW7a1V+5LoLa8ie/u7aoWmfqnmaws362/V38Zq1d5atS+J3vJqVm/sxgNBEHYgiLLD3lfy+lNatbdW7
eV/LyU1q1t1btS6K3vJrSW6nH7ACap+wtO4AmIexAEKWE3cxuN7P3zOx9M1tVRg/VmNmgmb1lZgfKHp8uG0PvmJkdHDPtUjPbaWaHsueqY+yV1NtDZvZh9tsdMLM7S+ptjpntNrM7ZvaQmf0km17qb5foqym/W9OP2c2sTdIfJd0m6bCkNyTd4+5/aGojVZjZoKRF7l76DRhm9g+S/iJp87mhtcxsvaRP3X1d9h/lTHf/xxbp7SGd5zDeDeqt2jDjK1Tib1fk8Od5lLFlXyzpfXf/wN3/Kum3kpaV0EfLc/dXJX36lcnLJG3KXm9S5R9L01XprSW4+5C7789en5R0bpjxUn+7RF9NUUbYZ0v605j3h9Va4727pB1mNmBm3WU3M452dx/KXg9Lai+zmXHUHMa7mb4yzHjL/HZ5hj+vFyfovm6Ju18r6Q5JK7Pd1ZbklWOwVrp2+ktJ31VlDMAhSb8os5lsmPFnJf3M3U+MrZX5243TV1N+tzLC/qGkOWPefyeb1hLc/cPs+ZikLaocdrSSo+dG0M2ej5Xcz/9z96Pu/rm7fyFpo0r87bJhxp+V9Bt3fy6bXPpvN15fzfrdygj7G5IWmNk8M5siqUvStlL6+Aozm56dOJGZTZe0VK03FPU2Scuz18slbS2xly9plWG8qw0zrpJ/u9KHP3f3pj8k3anKGfn/lvRPZfRQpa8rJP1n9ni77N4kPaPKbt3/qHJu40eSvi1pl6RDkl6SdGkL9favqgzt/aYqwZpVUm9LVNlFf1PSgexxZ9m/XaKvpvxu3C4LBMEJOiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0I4n8BN4qIn4wZkNsAAAAASUVORK5CYII=\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": [],
   "metadata": {
    "id": "WHK4rCVBC0dJ"
   },
   "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": [],
```

```
    "metadata": {
      "id": "roVOfV9EDRi9"
    },
    "execution_count": null,
    "outputs": []
   }
 ]
}
```