{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "XoVJrne9TkbL"

},

"outputs": [],

"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"

]

},

{

"cell\_type": "code",

"source": [

"(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "ldD5h1szUK6I",

"outputId": "9fedc791-bc60-42e8-ac27-0e8e1eaadcbe"

},

"execution\_count": 2,

"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": "L8Ak9kITUWhq",

"outputId": "adc691b0-899f-4caa-b8c1-210424624d82"

},

"execution\_count": 3,

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

],

"image/png": "\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": "DvJFrCEhVk02",

"outputId": "87b9b496-598e-4b00-da51-514881de14d2"

},

"execution\_count": 4,

"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": "markdown",

"source": [

"# RESHAPING SO AS TO CONVERT IMAGES FOR OUR MODEL"

],

"metadata": {

"id": "HexABigsVxOQ"

}

},

{

"cell\_type": "code",

"source": [

"X\_train = X\_train.reshape(60000, 28, 28, 1)\n",

"X\_test = X\_test.reshape(10000, 28, 28, 1)"

],

"metadata": {

"id": "wpgL9Gy1V9J\_"

},

"execution\_count": 5,

"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": "1B6JX4a\_Vugq",

"outputId": "886fefde-431b-493e-917f-458e3293b2d6"

},

"execution\_count": 6,

"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": "markdown",

"source": [

"# ONE HOT ENCODING"

],

"metadata": {

"id": "2QrfMo7GWSLk"

}

},

{

"cell\_type": "code",

"source": [

"y\_train = to\_categorical(y\_train)\n",

"y\_test = to\_categorical(y\_test)"

],

"metadata": {

"id": "rrt8\_nwlWYk7"

},

"execution\_count": 7,

"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": "YVJMmrnqWhKh"

},

"execution\_count": 8,

"outputs": []

}

]

}