```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "markdown",
   "source": [
    "Importing the required libraries"
   ],
   "metadata": {
    "id": "3um_uMGfsd_Q"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "import numpy\n",
    "import tensorflow #open source used for both ML and DL for computation\n",
    "from tensorflow.keras.datasets import mnist #mnist dataset\n",
    "from tensorflow.keras.models import Sequential #it is a plain stack of layers\n",
    "from tensorflow.keras import layers #A Layer consists of a tensor- in tensor-out computat ion
funct ion\n",
    "from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deeply
connected r\n",
    "#faltten -used fot flattening the input or change the dimension\n",
    "from tensorflow.keras.layers import Conv2D #onvoLutiona l Layer\n",
    "from keras.optimizers import Adam #opt imizer\n",
    "from keras. utils import np_utils #used for one-hot encoding\n",
    "import matplotlib.pyplot as plt   #used for data visualization"
   ],
   "metadata": {
    "id": "WHRdOxitsf9B"
   },
   "execution_count": 1,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "load data"
   ],
   "metadata": {
    "id": "3-HCpkd5snJs"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "(x_train, y_train), (x_test, y_test)=mnist.load_data () #splitting the mnist data into train and
test"
   ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "bL7zqSEmsip1",
     "outputId": "96e5d2ce-5e17-4b2c-be54-ee386c699f3b"
    },
    "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": [
     "print (x_train.shape)  #shape is used for give the dimens ion values #60000-rows 28x28-pixels\n",
     "print (x_test.shape)"
    ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "a4Jaofhmsr87",
  "outputId": "824dd079-6d4d-4b27-b319-7068f77ab859"
 },
 "execution_count": 3,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "(60000, 28, 28)\n",
    "(10000, 28, 28)\n"
   ]
  }
 ]
},
{
 "cell_type": "code",
 "source": [
  "x_train[0]"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "2tFCUAebsxyr",
  "outputId": "f967aebd-e77f-4a5c-80b1-e992d03a863f"
 },
 "execution_count": 4,
 "outputs": [
  {
   "output_type": "execute_result",
   "data": {
```

```
"text/plain": [
 "array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,\n",
 "         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,\n",
 "        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,\n",
 "        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,\n",
 "        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
 "       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,\n",
 "        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,\n",
 "          0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,\n",
"        90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,\n",
"       190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,\n",
"       253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,\n",
"       241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"        81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
"         0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,\n",
"       148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,\n",
"         0,   0],\n",
"      [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,\n",
       "       253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,\n",
       "       253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,\n",
       "       195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,\n",
       "        11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,\n",
       "         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0],\n",
       "       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,\n",
       "         0,   0]], dtype=uint8)"
      ]
     },
     "metadata": {},
     "execution_count": 4
    }
   ]
  },
  {
```

```
"cell_type": "code",
"source": [
 "plt.imshow(x_train[6000])     #ploting the index=image"
],
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/",
  "height": 282
 },
 "id": "_U2oji5es0Q_",
 "outputId": "6df23ec2-8e33-40c5-bc88-b76800f638df"
},
"execution_count": 5,
"outputs": [
 {
  "output_type": "execute_result",
  "data": {
   "text/plain": [
    "<matplotlib.image.AxesImage at 0x7fbf39d88c90>"
   ]
  },
  "metadata": {},
  "execution_count": 5
 },
 {
  "output_type": "display_data",
  "data": {
   "text/plain": [
    "<Figure size 432x288 with 1 Axes>"
   ],
```

"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzA
AALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA6
Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAOfklEQVR4nO3df6zV9X3H8deby4+r+KMGExmQaRnrytqJ3S22
kTY0RqNICthpyjLLMrPbbWXBxT/q2nSSxSzYoG6JLewymXShEAKpBOGwykz4zkrZYGLW4o2Z1VbqhTAEmYU5bGEXqL2
LIL/hvvfH/dLc6v1+zvV8v+d8d8D7yfj+TmnPt9n+/5vjm5L77nfD/n+/2YuwvAhW9E1Q0AaA7CDgRB2IEgCD
sQBGEHghjZzI2NtjHerrHN3CQygkd1Sk/aUPVCoXdzG6R9I+S2iT9s7svTz2+XWN1vd1YZJMAEjZ7d26t7
rfxZtYm6VuSbpU0Q9IiM5tR7/MBaKwin9Sdrj7m+6ylJ6JLewymXShEqc1YGLW4o2Z1VbqhTAEmYU5bGEXqL2
ndbLA5gAU0fCj8e7e5e4d7t4xSmMavTkAOYqEfb+kqYN+n5ItA9CCioT9VUnTzewaMaxst6YuSNpbTFoCy
1T305u5nzGyJpO9rYOhtjbvvLK0zAKUqNM7u7s9Jeq6kXgAEbH9I+KiHyRkxSN3IzGCWnmJy
BEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIyzo5Czn/tEsp4aS19+1avJdY/1n03Wl3bdk6xP1o+S9WjYswN
BEHYgCMIOBEHYgSAIOxAEYQeCIOxAEIyzIyl1Prok/cOjjybrHx89c/YMoFHYz2y
vpiKSzks64e0cZTQEoXxl79s+5+9slPA+ABuIzOxBE0bC7pB+Y2RYz6xzqAWbxWicLbg5AvYq+j
Z/t7vvN7EpJm8zsv9z9z9xcEPcPcuSV2SdJmN94LbA1CnQnt2d9+f3R6S9IykWWU0BaB8dYfdzMaa2a2k
u6WdKOshoDUK4ib+MnSnrGzM49z3fd/d9L6QqlsVGjk/W2q65M1h8sMI4uSY8f/s3c2uoHFiTXnbSOcf
Qy1R12d39T0rUl9gKggRh6A4Ig7EAQhB0IgrADQRB2IAhOcb3AHb49fannHz787S9v8afyBmlL/f8L9+Y
n1u7/KlXkuuiXOzZgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIxtkvAG0fyr8k80mdt+f6emdDt33dyqXJ+tSnOE21
VbBnB4B4Ig7EAQhB0IgrADQRB2IAjCDgRB2IEgGGe/AOz56ozc2oa6UtS5a3rFnbrJ+9ao3kvX02e5oJvb
sQBCEHQiCsANBEHYgCMIOBEHYgSAIOxAE4+zngV/e+alk/c1b/zaxbvHwG1pqjrOb1Ob2Ob2ROS5kiaYGb7J
N0vaY6ZzZTkkvZK+nIDezzv9S062tDnP9bP9P96uVV8b+EP/sZH3L6vR3DFGLxvFsv7ytU/W/F9EGjhO
z3/u6mZH2dptT93K2qZtjdfdEQi4v9hQBoOr4uCwRB2IEgCDsQBGEHgiDsQBCc4toChNv8/qDq3OIsxv2dPsH43WyD
nrJ58r+9mVuz29PPXWvbK/o+kqz/+K6pybqO5i/bnTlbHOiOTRz8AWljMm3DVYbBRlXW/4RKfx+DWfrzFs/
rRwGTfL+HiPOJg3dfSyN1ayzVoj7x+fnbeLTmO/N0PtN69Ds9PnNo9fH/pLJFPVVxxBWSJiRobKlSMrHMGhfmg2JEvtuSfP0sR
MjUx2r3TBP5tKP0xmmzVNJT6ZoNs55fMhdd9mx/vaoB12k9Tt9+MDtbZvoEBa0XjdJdgP75vxOZajH9fBr4/lidWqgB
KhUmzPXvnjZOe9MPT29Prfd6ZfLNkbgHVbGoERlbZPD3C9BsG7G2Fd6g7eWd1KbGRTmIrBs8rvmPx/vKbC1Oz+Kww1a+BpdEvlSw
rOAiDKbFBNENTVeX2YBmyxtkJGSUOMXy0z+3JSv+Wk9Uk1BqUjeMb5rwXdvGxQx6VvcGs+tYzqLvY4zAUUsUs16QUq0sjcTMuBmzHWE
NA4/DtL5cSnNeGqeBJj11DA2LPL5J0zJwf9o7VLBwd9Lr4zR/M97wmvMtOi0n9xW9k8swi6PMB+lZBGf0PlK3BNmU8HD/QO/
dG0yQvDQnKVSRN0w1jFuHl3dcE6i9u0WBxfcJLsHJPf+mTjhNBxwOdD/DOr5rN/fcNrXdZXG/L/E28XJ2fzk/iL/Nd2ogB
Hzv6AnNQZ4Vb4xRFeiR4YqeTPLGf1Yz0hNJdt8gqI7KsG3Dj5+KAkWJbIHMxF5WjxM3wEIyWOwkxc+z7jBoUyOLhz2+0/k7
ztrUyP4jZb98Gk1jjI1x5oTFpNSEyDxcHQ2LwkBBN6+HeR9VqUlzTUbZXiPvSxAHi3Fw7KMaqj+iKfk4+wECKD9m9/tSfF
Tr7S8i4yN3W3oVWzWpd3cCQjSaj0vhQ+WwdWGJwmz9AH/lvOXbvD9Av7Kwd8+NsP2Oh+0U1X6gJGqGzbsZIbrhYhMBJH
dv8JHjodyMWn2PoJHYUm5exdBN/6VFx2IYOjBPkFz2CT+TUEi3Xz6+O3QkOVlLpPBZ/uLU1A4pECUZQL5sbHpIPxOqGiLKT9a
aAZ40OdoMxfZYkXa3KD0ub/dxgn2FAqo3O5RyddBmsTn4RlNAHXoF+hrdr7OyjS3jvn9OkWRPmkaUePnT
hqsvf+t4ZSCVZ0NXDUswTXZJH/Wwo2wl/r7MP/fBCjzoYT5r2aPeo11W/bIubTvm1u3O1MXVOw81o5v+ifqEnLH7hMYr0TOFiSS3TH7pxvH4
WoXdS6B+WYeqBR1ocGS+mKhyWCR1e/bO//ejtpiJGyn63fE7CCm1nGAJ4aMvCNJfIUGvMeT6FPw4qQCWMfoV0+
cyl8U1iiVdmtJFbCnCjTaE6FHvmdQhUbTg84qFRBRnLXbc8xPJO3hDHlZBW3b+sDmkddNK1iEqMgiDwRiJnXJBwDUmDeuXv
4jWZpXVDJCB1uMlPVHBdM0QvmRPfXSeQG6ne5Tp7iJHD8Z7ePZR0yppyiSpDmn7f6afOw1/gggK3JadiAs1pVrKLFcbb
7Ukm8NtGjhT1vqVgpXdF33T8J+bfXm2fvMAsJHX6kUhLSvI8yd6zw5kbl6i4EO+zR6ic7cmO5q/E/OhpD1c9HRNGLzF3ChBjqE
nERtW4tYr2WnOQOgT9U2SXWnlFcoSOm+yy17Jr1HhYPNxqnJGFTI1Ed/jnZmJmtV5J/W8EVW4uDrQmJ+H1SN1ND8T5yTqf1xH
jm9TQP1vQs/rpMcv7EGxBbP4/V9ixnY5BDtV2Usf/jmCeR85GbdSwJlY2GeaJcxrh6fUqaHUyU5r/cF9JovCqW+r6Ir2w5LIvxqVzY
umt2t+VfN2zIVw3Gz2WUbGepN2JNI8Ls0WjhN/Iq/fI7T/nbzVOR21YK4S2K6d/2yNu3+FP37HQJgUa+TmHbZAP
Sg1V0XTFrOGCNjxOq/7ho8/XHsngb7R3T3izMRcpMbDo5jrNkp4n2NFhKqXYYaWbwRAI5xTCT5Hz3t2iJP43o2
BSPfHmrNycbP/tpwzUGkZE0rxCNaB90oT2rltwT5hUl9GMe+fD6zZBpgd+/n4xBkXKDAx1Z8fhaTGXWUvXu3q
n/bh1FgJzmxoWJx1/+HLeM69i4nVQX3ncqD5E/XAUOWzkT1nNOPRwHiqXSGdOiCIXGcAG5EGaQ8VyHqCnq71QRDh
9DH5ky5Rll87eR3L3C0NEWEe6bNEbtcWLP+UEJXUzyO0GS+xgwMnn6eaeQB3O6q7e1IFJlPZlhdbg/uWpY9QHCsXa0mSdG
gDaTfcYnz7GFPr+lWXsVi0a0/Ak5FoeDxNBUs9uoqj0LyFXCDgCGtmfcc+DCOePOVhjhxOBUDLJ+kJNAo2y3K2ddKfc0V2
8RIVFwmJuI/Oew40Y1dKQ0ZFdeMdSi9MAmHQb9B0xQnR84GJaJ+NY6uN5gGd1vOdThAE2OtEA0ZWlyGYjQJ+8DZrZ4Nh
0F3y2CdHCLPYV6j9bU2OcHZg7TLbp2LNm/KtAQJ0yR5fo6Nx02RSmSFc8+9dXMsqEUNFtZ4jiGZ3XUh7uAxpDBNdVpC29XRcD
A+G9nr9nLNZoUVMjVkLUjsHrD9bxJIRe7BO6fVmgyZQnZpdQKBAn52tX5CXZLtiM4F/ZEE0YkWbyBPd93XvjEqdy0Y
a4pKNr3Nk+OC9MDWlWwC5DFwBWRkG9pSCyk1qDYl5G8kSyrrVu1rUf5FCWZmAkZF8y/TpJ6jVoFMmJZ6gF/C1bdHqmcpgo
MhdnFcs+rIaaqe2kdM8mKPKgJ+hJ5xN45cvnEgjHDD5eB6jh7+YgwKWzBCfwewPN+VfuO66DLKsEu6uf8kpjLGfAPCfNtx2z
lD8KDZOGCaN1E9ZKt/SIsjyF0xb2t2bkPu3Tq6+Cnf3J5uVU4QJGLYhKXQnBl5r+r4E7g3n8twUSmfRcsJ6yvgeuz+WNkN
yMDVqp0M2PbcgTm5lRxwWPSLBb2jWOT9RQTJ2/MrFvp12FTQqpVszQhIOj91QVqA6aomrkm4JF/Fa6Ew6ZnECMi0ucJ+2
w3nsVu62NjJqS04k8gnxc/OdYu5g4dqJ5PkBxk6l/sa2/wJunP2+KvUE4K8gnK5W2zm7uN9MQEOPeUMGCrBU4s2fWmLC/
nOY1EeaQSeuqQs5bDzwwmP+CnYBZH5GNyGL/YYGvMCkOgmU5M8/l2s2i2OQubM09WmEMpWAqEEPQzRRdjUGO3EJ9+f+1yu
/tSmr2kc5BJmvX5q0KVUF+9b69yE5fRBKuAYBnw6e+qOZhQKQqk39kmLZmtrmxwqrSeTZy2l8jF9iZXHCkmIKHzkmYmCtGT
X3e9jW79DpXNt4LOsxdaMqbaZL+WuX9X1/lWk0xE0Eg5Xv+TcAlMu4kUItvE3UTYBzg7sFo5fKktMKeh0aUf0d0+kw+uk2WX
IDmkTu7EukY7bkbMAyMQb8FOVD7EnIdL5tIxYqtcGmTkzX8N80/DnH+7P3Ed/xjV6Az7umIw0hgDe9vzaU+tXJtc5d2+CrO23X
HUoUpIkUr1aC3TqbNTlDl+l5E0qTjPh9H5XnL0qgpUhiIYGBbNkT6LhF2WWy5Tm02L/vVQdS3upjBlsXAmbWMYSPv5P/J2n8
x8GHtyhY01wuNWkD0u8aJkXMLaXl6uYGbGfL8Z9jYT+1p+pmfTA7eOfLtGh/0WM0NRRcSk3C7dsD6vyKlvpwcP8D7BSeYM
0lN8iwF6Yp3YpYPXOgM9Y9qHIs3c5tODLJGp+MPrY/PwpqrVU/oqxpnzbyUiKSQE2EKJ9HEWJHFy1M6eGf5DTuMzbTkr5S
0bNamTcb8/s0pH1GUa4w+1qy1vDXFDvVVfG+VV+pwjvIhQAD4upOtM+fg7yq2fq1k9BQH6CsM9OCNOdmXWWMKR0njL0gVh
lwOxQbnWjPMjbZkhRsF+z1c0e9flLDvEoEIu1zNjErjE1Y9MPi6tkA2uScjHLkA+z7kqE+c+bC4rR/Y3lDxhUjh67JgSgUq0Tz
Ozi3Yy7oQSbCUjcjQbhNrumEIJ6yfHg7p3n3S8DUSqajaOFlcHOSJcZCHSuG4o/DF8Ca/3fjH6BOsqxDfGXF1tIENVWRkG8Klw
s2mobBc+IVfIpkxvlcqwiqP5J/VUn0BrM9vwWsRF+u/Ecs8k8Eg5KQBVPZ8pE1I1ZEoGgEvCVYkrQUJwNTGYi3gZmLbqwBrkhs
GHUqS0OgD7bW0M3w0Hd0X8FLeCepEhdKxy8ZsTg0AHWE45ySdvdjUZhSfB/ukl76a0mpeUGwmZbYKQvRwBXRu3n6MB0vM0Y
YXnJ5iPdwEU4dl0BRJDuGH26pbw2o19y0mn0yu/8PoaBdfQXwQLCAOJrfYIM0W4ymwrIBxrw4vv3Y5Ri7H6lmsCZ0YO6VCd
BcpKlMDJK/ov9c0ZoZJ7h7PuGrOcBGvGvozXBcXOF9s5eJ2Y6JNCJmDNj07LRH7zTJLMRAOgShzaB2bRlmfTxxGvP2gEQg3DSdAJB5
2fcj4V86EmsLKJ/HuPBYhQ3IVYf0O9c/rrs2q9L6Bu1g+EqYNTJkXIoczSJNa0izlNPa9HTuUYKfAo4sGLyxC0bixFHxjpOWzA
WPtq2vmzwkR7FNMxRNLoSm9s7z3uIm+4W7P6DpQGbdwWa/a9lfeS68tK5Q5iLwdTDsD4YIZEiT2bNrw7IB0IN52ypJrS/Lkz7Z
nPp3JbIu4nb4xc8kIjOmj+lNhBp9rPN3DMmFWr/P0LUbsYdCtlnN28wC+qY3iSp+cF4/8YzCuiAgtEb/4dP0w2ITYeK0whHB
Qg5Er2cs16sgD/ZIN4c3xyrb+8Dmj0BZMtTc2bUyVu/IekqVEJpNp94yg4OhPY1vz9ovB0hiIfx2uHHNEthA0+s5iwwm9rk3d
hr2Jl5qg/fnNOdBDEQBX18nv6WqZJj1IP0UODdNWGj0CXGsvM8n2VOvRZcmUpHEmtP/wvw8qgwvCRLT00mDkgL5XcGrSTYg
wT3Q2TZ68zzvvBvpEaK5qG3gTKTpXFN5T+JYpaxG0krZl5/tQ5zVsiUDtcf8K1G1c1pMm1G+/w1Gc1Y2o7xTzqeX9SjmVq/KNOUaJ
0hVjAWHpV2HJ8cswULHFVRbaf+V3Xsh6u0fxpxYiJK2UUWvSjnSRBeLpR3H0MiRz+RhKMTOXYtIH1CPu10l5nn5dyk2+5oHp
0n7mG8g2XNC0QiwSspPDNT4TXVcTW+5DwG4g6aRsfNZ6dPsNV7UQeMHrDmKZQ8tSzsTWp38TJ7Axvf2V8+ELVYy/Nu/qFsGm6
MIAI5lUbp/8h7lUXSN0SAAJiAlrTpYP2BKjiDNc2OQOoPhClxZ8IrmfNp9wx1h0rzIdfwWJzz7cAW0hfHHqLjV9XYzgdVIh7
rekPHASOHTRAfnp6oGmKJ9r88Ypl8dbtj4oH2fzjR30FGRlvfJwxN+sK3/MLzVTrzL0F+D1e3umHZpT8YEkxgBLHS9SJ2NyTE6
W/ZwTCXcCrYJ90mCNqJb3CzHLxVyI6SZSIsNrBkH8MpDmL6rGGYgkT52xTQPa23A7CQ0CI3Q6FH5zC9j/WJwbJBvmCNNxU1O2
Ltr6wA/pVOvqO2ySDcybdD0xX0EOKPMIl4KnHfpbI5r1yr5P0G5bZBNoE3nrDh2XlSeKvEUYMwJLBnPbJyxNI7vDn7W0gdXkdpF

```
OxAEYQeCqBl2M5tqZi+Y2etmttPMlmbLl5nZfjPblv3MbXy7AOo1nPnZz0i61923mtmlkraY2aas9oi7r2hc
ewDKMpz52Xsl9Wb3j5jZLkmTG90YgHJ9oM/sZna1pOskbc4WLTGz7Wa2xszG5azTaWY9ZtZzWicLNQu
gfsMOu5ldIulpSfe4+2FJKyVNkzRTA3v+h4Zaz9273L3D3TtGaUwJLQOox7DCbmajNBD0de7+PUly94Puft
bd+yWtljSrcW0CKGo4R+NN0mOSdrn7w4OWTxr0sIWSdpTfHoCyDOdo/A2S7pL0mplty5Z9TdIiM5Sspy
SXtlfTlhnQloBTDORr/kqShrkP9XPntAGgUvkEHBEHYgSAIOxAEYQeCIOxAEIQdCIKwA0EQdiAIwg4EQdiBI
Ag7EARhB4Ig7EAQhB0Iwty9eRsz+4WktwYtmiDp7aY18MG0am+t2pdEb7Uqs7fJ7p6d8Fv5hq6Mn5MaLbPm
Pe7eUVkDCa3aW6v2JdFbvZrVG2/jgSAIOxBE1WHvqnj7Ka3aW6v2JdFbvZrSW6Wf2QE0T9V7dgBNQtiBIC
oJu5ndYmZvmNkeM7uvih7ymNleM3stm4a6p+Je1pjZITPbMWjZeDPbZGa7s9sh59irqLeWmMY7Mc14
pa9d1dOfN/0zu5m1SfqppJsk7ZP0qqRF7v56UxJYWZ7JXW4e+VfwDCzz0p6V9J33P1j2bJvSupz9+XZf5T
j3P2rLdLbMknvVj2NdzZb0aTB04xLWiDpT1Tha5fo60414XWrYs8+S9Ied3/T3U9JWi9pfgV9V9tDx3f1FS33s
Wz5e0Nru/VgN/LE2X01tLcPded9+a3T8i6dw045W+dom+mqKKsE+W9PNBv+9Ta8337pJ+YGZbzKyz6m
aGMNHde7P7ByRNrLKZIdScxruZ3jPNeMu8dvVMf14UB+jeb7a7f0LSrZK+kr1dbUk+8BmslcZ0Ohzi/NNd7
MMMc34r1T52tU7/XlRVYR9v6Spg36fki1rCe6+P7s9JOkZtd5U1AfPzaCb3R6quJ9faaVpvIeaZlwt8NpVO
f15FWF/VdJ0M7vGzEZL+qKkjRX08T5mNjY7cCIzGyvpZrXeVNQbJS3O7i+W9ECVNQbJS3O7i+WtKHCXn5Nq0zjnTfNuCp+7Sq
f/tzdm/4jaa4Gjsj/t6SvV9FDTl8flvSf2c/OqnuT9IQG3tad1sCxjbslXSGpW9JuSc9LGt9Cvf2rpNck
V1NtsDbxF3y5pW/Yzt+rXLtFXU143vi4LBMEBOiAIwg4EQdiBIAg7EARhB4Ig7EAQhB0I4v8BJ99UDMnd
G+wAAAAASUVORK5CYII=\n"
        },
        "metadata": {
          "needs_background": "light"
        }
      }
    ]
  },
  {
   "cell_type": "code",
   "source": [
    "np.argmax(y_train[6000])"
   ],
   "metadata": {
    "id": "41UMyRvLs8Xo"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "0"
   ],
   "metadata": {
    "id": "iWoeNIyBtYlG"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "Reshaping Dataset"
   ],
   "metadata": {
    "id": "ZPz20_rhtpXG"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "#Reshaping to format which CNN expects (batch, height, width, channels)\n",
    "x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')\n",
    "x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')"
   ],
   "metadata": {
    "id": "kjucrwEJtqMW"
   },
   "execution_count": 9,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "Applying One Hot Encoding"
   ],
   "metadata": {
    "id": "mTtITMxpt49v"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "number_of_classes = 10  #storing the no of classes in a variable"
   ],
   "metadata": {
    "id": "atav4dn5t53C"
   },
   "execution_count": 11,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "y_train = np_utils.to_categorical (y_train, number_of_classes) #converts the output in binary format\n",
    "y_test = np_utils.to_categorical (y_test, number_of_classes)"
   ],
   "metadata": {
    "id": "JanD8vbjt8pD"
   },
   "execution_count": 12,
   "outputs": []
  },
{
 "cell_type": "markdown",
 "source": [
  "Add CNN Layers"
 ],
 "metadata": {
  "id": "-i1eeg2rutqe"
 }
},
{
 "cell_type": "code",
 "source": [
  "#create model\n",
  "model=Sequential ()"
 ],
 "metadata": {
  "id": "AoZKCAYKuWCb"
 },
 "execution_count": 15,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "#adding modeL Layer\n",
  "model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation='relu'))\n",
  "model.add(Conv2D(32, (3, 3), activation = 'relu'))"
 ],
 "metadata": {
  "id": "NE0FlfDEu0PI"
 },
 "execution_count": 16,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "#flatten the dimension of the image\n",
  "model.add(Flatten())"
 ],
 "metadata": {
  "id": "Lq6evWnEu4MD"
 },
 "execution_count": 17,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "#output layer with 10 neurons\n",
  "model.add(Dense(number_of_classes,activation = 'softmax'))"
 ],
 "metadata": {
  "id": "Aj23jEZ0vCqf"
 },
 "execution_count": 18,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "Compiling the model"
```

```
   ],
   "metadata": {
    "id": "Fphjkw68vJer"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "#Compile model\n",
    "model.compile(loss= 'categorical_crossentropy', optimizer=\"Adam\", metrics=['accuracy'])"
   ],
   "metadata": {
    "id": "Lgaph8XpvKWU"
   },
   "execution_count": 19,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "x_train = np.asarray(x_train)\n",
    "y_train = np.asarray(y_train)"
   ],
   "metadata": {
    "id": "_FU_G546vQGg"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "Train the model"
   ],
   "metadata": {
    "id": "Gzb0Wfehvg5w"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "#fit the model\n",
    "model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=32)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "8ez4TAFDvelt",
    "outputId": "c16b4d9e-9473-4ce0-9188-da1d0fea5b99"
   },
   "execution_count": 21,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Epoch 1/5\n",
      "1875/1875 [==============================] - 110s 58ms/step - loss: 0.1882 - accuracy: 0.9533 - val_loss: 0.0751 - val_accuracy: 0.9762\n",
      "Epoch 2/5\n",
      "1875/1875 [==============================] - 110s 59ms/step - loss: 0.0606 - accuracy: 0.9814 - val_loss: 0.0754 - val_accuracy: 0.9758\n",
      "Epoch 3/5\n",
      "1875/1875 [==============================] - 109s 58ms/step - loss: 0.0470 - accuracy: 0.9845 - val_loss: 0.0925 - val_accuracy: 0.9753\n",
      "Epoch 4/5\n",
      "1875/1875 [==============================] - 111s 59ms/step - loss: 0.0336 - accuracy: 0.9891 - val_loss: 0.0781 - val_accuracy: 0.9801\n",
      "Epoch 5/5\n",
      "1875/1875 [==============================] - 109s 58ms/step - loss: 0.0302 - accuracy: 0.9907 - val_loss: 0.1067 - val_accuracy: 0.9782\n"
     ]
    },
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "<keras.callbacks.History at 0x7fbf3559ba90>"
      ]
     },
     "metadata": {},
     "execution_count": 21
    }
   ]
  },
  {
   "cell_type": "markdown",
   "source": [
    "Observing the metrics"
   ],
   "metadata": {
    "id": "iaK0KRstxt2a"
   }
  },
{
 "cell_type": "code",
 "source": [
  "# Final evaluation of the model\n",
  "metrics = model.evaluate(x_test, y_test, verbose=0)\n",
  "print(\"Metrics (Test loss &Test Accuracy) : \")\n",
  "print(metrics)"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "T5cZprb9xu1Z",
  "outputId": "a6c20158-3ee6-4158-95ad-fb482ded3b90"
 },
 "execution_count": 22,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Metrics (Test loss &Test Accuracy) : \n",
    "[0.10665197670459747, 0.9782000184059143]\n"
   ]
  }
 ]
},
{
 "cell_type": "markdown",
 "source": [
  "Test The Model"
  ],
  "metadata": {
   "id": "K-YRhfXMx1zq"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "prediction=model.predict(x_test[6000:6001])\n",
   "print(prediction)"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "qf91fK_ax2w-",
   "outputId": "b6f702f9-938b-4d4c-b99f-5eaed7af0192"
  },
  "execution_count": 23,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "1/1 [==============================] - 0s 71ms/step\n",
     "[[3.5662453e-18 1.3220488e-22 5.7828655e-20 3.6040473e-10 6.7578981e-10\n",
     "  7.9998215e-13 7.8929150e-19 1.9612942e-12 2.1346503e-11 1.0000000e+00]]\n"
    ]
   }
  ]
 },
  {
   "cell_type": "code",
   "source": [
    "plt.imshow(x_test[6000])"
   ],
   "metadata": {
    "id": "kkdpDXBCyAfP"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [

"![out.png](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAaAAAAGdCAYAAABU0qcqAAA
AOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjYuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm
9yZy/av/WaAAAACXBIWXMAAA9hAAAPYQGoP6dpAAAbDUlEQVR4nO3df3DU9b3v8dcGkuWHydIQks
1KoAEVWn6kVwppLkqxZIB4hwvCzAF15oDHgasNTjG1etNREduZtDHgasNTjG1etNREduZtDhjGT0pznRaqHcELDMCR+8ZZiCYM
NaEHqIcDq3NIbmphEMSKueShSAhks/9g+u2K0H6XXbzzm6ej5nvDNn9fvJ9++3Wp9/s5ovPOecEAMAA
S7MeAAAwNBEgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECABgYrj1AF/U19enM2fOKDMzUz
6fz3ocAIBHzjlduHBBoVBIaWk3vs4ZdAE6c+aM8vPzrceAAHxN1RbCe1VC1FUCFq9vTbN66j/h6F1Qtx7Fed5TTUZ2e3wNAHGqyJDjf0PCMCkhyM4WbF8FbhxXcdTKzIhQzo
6fz3ocAIBHzjlduHBBoVBIaWk3vs4ZdAE6c+aMCgoKrMcAANyitrY2jR8//obPD7oAZWZmSpLu0f0arnTja
QAAXn2mXr2nf478+/xGEhag6upqvfTSS+ro6FBRUZFeffVVzVzk56brPv+x23Cla7iPAFA0vn/dxi92dsoCf
kQwptvvqmKigpt3LhRH3zwgYqKirRo0SKdPXs2EYcDACShhARo48aNWrNmjR555BF97Wtf0+bt+/f09CgcgcDkdtAIDUF/cAffLJJ7p69ary8vKiHs/Ly
rVrxJxOABAEop7gK5cuaLGxkVlpb+5SBpaSotLVV9ff11+/f09CgcDkdtAIDUF/cAffLJJ7p69ary8vKiHs/Ly
1NHR8d1+1dVVSkQCEQ2PgEHAEOD+S+iVlZWqurK7K1tbVZjwQAGABx/xRcTk6Ohg0bps7OzqjHOzs7
FQwGr9vf7/fL7/fHewwAwCAX9yugjIwMzZgxQ7W11ZHH+vr6VFtbq5LS0pmMWLuru79cgjjyTicACAJJSQAK1cuVJ//vOf9fzzz6ujo0Pf+MY3tH//
cRfC4fDCgQCmq+l3AkBAJLQZ65Xtdqnrq4uZWVl3XA/80/BAQCGJgIEADBBgAAAJggQAMAEAAQIAmCBA
AAATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
MEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMAEAQIAmCBAA
AATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
MEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMAEAQIAmCBAA
AATBAgAYCLuAXrhhRfk8/mitqlTp8b7MACAJDc8Ed902rRpOnjw4F8OMjwhhwEAJLGElGH48OEKBoKJ+
NYAgBSRkPeATp48qVAopIkTJ+rhh/WqVOnbrhvT0+PwuFw1AYASH1xD1BxcbG2bt2q/fv3a6+vWrW
ptbdW9996rCxcu9Lt/VVWVAoFAZMvPz4/3SACAQcjnnHOJPMD58+c1YcIEbdy4UY8++uh1z/f09Kinpyfy
dTgcVn5+vuZpsYb70hM5GgAgAT5zvarTPnV3dysrK+uG+yX80wHZ2dm666671Nzc3O/zfr9ffr8/0WMAAAaZhP8e0MWLF9XS0qK8vLxEHwoAkETiHqAnn3xShw4d0kcffaT3339fDzzwgIYNG6YHH3ww3ocCACSxuP8I7vTp03rwwQd17tw5jR07Vvfcc4+OHDmisWPHxvtQAIAkFvcA7dy5M97fEgCQgvhbcAAAEwQIAGCCAAEATBAgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwk/H5AGFjpB8M8r+mpqvF82dDJkzg2TmGr9KcVUVaEHqIcDq3NIbmphEMSKueShSAhks/9g+u2K0H6XXbzzm6ej5nvDNn9fvJ9++3Wp9/s5ovPOecEAMAA
S7MeAAAwNBEgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECABgYrj1AF/U19enM2fOKDMzUz
6fz3ocAIBHzjlduHBBoVBIaWk3vs4ZdAE6c+aMCgoKrMcAANyitrY2jR8//obPD7oAZWZmSpLu0f0arnTja
QAAXn2mXr2nf478+/xGEhag6upqvfTSS+ro6FBRUZFeffVVzVzk56brPv+x23Cla7iPAFA0vn/dxi92dsoCf
kQwptvvqmKigpt3LhRH3zwgYqKirRo0SKdPXs2EYcDACShhARo48aNWrNmjR555BF97Wtf0+bt+/f09CgcgcDkdtAIDUF/cAffLJJ7p69ary8vKiHs/Ly
rVrxJxOABAEop7gK5cuaLGxkaVlpb+5SBpaSotLVV9ff11+/f09CgcgcDkdtAIDUF/cAffLJJ7p69ary8vKiHs/Ly
1NHR8d1+1dVVSkQCEQ2PgEHAEOD+S+iVlZWqurK7K1tbVZjwQAGABx/xRcTk6Ohg0bps7OzqjHOzs7
FQwGr9vf7/fL7/fHewwAwCAX9yugjIwMzZo1SzU1NZHH+vr6VFtbq5LS0pmMWLuru79cgjjyTicACAJJSQAK1cuVJ//vOf9fzzz6ujo0Pf+MY3tH//
cRfC4fDCgQCmq+l3AkBAJLQZ65Xtdqnrq4uZWVl3XA/80/BAQCGJgIEADBBgAAAJggQAMAEAQIAmCBA
AAATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
MEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMAEAQIAmCBAA
AATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
MEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMAEAQIAmCBAA
AATBAgAYCLuAXrhhRfk8/mitqlTp8b7MACAJDc8Ed902rRpOnjw4F8OMjwhhwEAJLGElGH48OEKBoOJ+
NYAgBSRkPeATp48qVAopEmTJurhh/WqVOnbrhvT0+PwuFw1AYASH1xD1BxcbG2bt2q/fv3a6+vWrW
ptbdW9996rCxcu9Lt/VVWVAoFAZMvPz4/3SACAQcjnnHOJPMD58+c1YcIEbdy4UY8++uh1z/f09Kinpyfy
dTgcVn5+vuZpsYb70hM5GgAgAT5zvarTPnV3dysrK+uG+yX80wHZ2dm666671Nzc3O/zfr9ffr8/0WMAAAaZhP8e0MWLF9XS0qK8vLxEHwoAkETiHqAnn3xShw4d0kcffaT3339fDzzwgIYNG6YHH3ww3ocCACSxuP8I7vTp03rwwQd17tw5jR07Vvfcc4+OHDmisWPHxvtQAIAkFvcA7dy5M97fEgCQgvhbcAAAEwQIAGCCAEATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
```

QAMEGAAAAmCBAAwAQBAgCYIEAAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMDEcOsBMLT
8+y9me15T8vVmz2vmZzd5XiNJ6wJnYlrn1fwTyzyvOdWU53nNXa9f8rxGkv7P932e1/g/uM3zmtDm9z2
vQergCggAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQA
MMHNSBGzT5fN8bzm38pe8bxmpC/D85pYXXUDc5yaaW95XzTN+5LfL7nifZGkaenez3nt7HTPazZvnuF
5DVIHV0AAABMECABgggABAEwQIACACQIEADBBgAAAJggQAMAEAQIAmCBAAAATBAgAYIIAAQBMEC
AAgAluRoqYtd3v/c6dA3ljUcR2U9FYzcgIe15zdf7dntcMq/3A8xoMTlwBAQBMECAAgAnPATp8+LCWLF
miUCgkn8+nvXv3Rj3vnNPzzz+v/Px8jRw5UqWlpTp58mS85gUApAjPAeru7lZRUZGqq6v7fX7z5s165ZVX
9Nprr+nIkSMaPXq0Fi1apMuXL9/ysACA1OH5QwhlZWUqKyvr9znnnLZs2aJnn31WS5culSS9/vrrysvL096
9e7Vq1apbmxYAkDLi+h5Qa2urOjo6VFpaGnksEAiouLhY9fX1/a7p6elROByO2gAAqS+uAero6JAk5eXlR
T2el5cXee6LqqqqFAgEIltBQUE8RwIADFLmn4KrrKxUV1dXZGtra7MeCQAwAOIaoGAwKEnq7OyMeryzs
zPy3Bf5/X5lZWVFbQCA1BfXABUWFioYDKqmpibyWDgc1pEjR1RSUhLPQwEAkpznT8FdvHhRzc3Nka9b
W1t17NgxZWdna8KECdqwYYN+/OMf684771RhYaGee+45hUIhLVu2LJ5zAwCSnOcAHT16VkPfdd1/k64
qKCknS6tWrtX37dj399NPq7u7WunXrdP78ed1zzz3av3+/RowYEb+pAQBJz3OA5s+fL+dufBNKn8+nF19
8US+++OItZDYbBb+JX/zwgx/nxJ9M9r6l9Zm5Mx/p4ifefSmeeHOZ5TfZ/+w/Pa3ZP3eF5zVfSRnpeE6uxM
Rzr4u3eb5Ya8LwCg5X5p+AAAEMTAQIAmCBAAAATBAgAYIIAAQBMECAAgAkCBAAwQYAAACYIEADAB
AECAJggQAAAEwQIAGCCAAEATHi+GzZwK/7j6iXPa97/H9/0vCaj4V88r5GkO/fHtMy7Ld6XlD7xA89rGv/
nP3o/EDBAuAICAJggQAAAEwQIAGCCAAEATBAgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECA
BgggABAExwM1IMqNuHjfK8Zso/fuR5zdv/MsfzGknyXfH+32QjO7yvGV/1vvfjfNLneQ0wmHEFBAAwQY
AAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQAMEGAAAAmCBAAwAQBAgCY4GakiNknh
0LeF03zvuRn+Ue8r/nv3tdI0u6LY2Na59X/er3E85qp3/t9AiaJn3+70ut5zdjDpz2v+czzCgxWXAEBAEwQI
ACACQIEADBBgAAAJggQAMAEAQIAmCBAAAATBAgAYIIAAQBMECAAgAkCBAAwQYAAACa4GSliVvDP/
+l5zez/+qDnNf/0jV96XpM/bJTnNZJUNvqM5zU7w3d4XvNf/neb5zbW/6CrwvEaS1ga8z/d3uzZ
4XlPYVu95DVIHV0AAABMECABgwnOADh8+rCVLligUCsnn82nv3r1Rz69Zs0Y0+ny9qW7x4cbzmBQCkC
M8B6u7uVlFRkaqrq2+4Zz/+LFi9Xe3h7Zdu7ceUtDAgBSj+cPIZSVlamsrOxL1/j9fgWDwZiHAgCkvoS8B1RbW6vc3FxNmTJFjz/+uM6dO3fDtT09PQqHw1EbACD1xT1Aixcv1htvvKGamhq99NJLqqurU1lZmfr6+vpdX1VVpUAgENkKCgriPRIAYBCK+y8lXbFiReTPM2bM0MyZMzVp0iTV1tZqwYIF162vrKxURUVF5OtwOEyEAGAISPjHsCdOnKicnBw1Nzf3+7zf71dWVlbUBgBIfQkP0OnTp3Xu3Dnl5+cn+lAAgCTi+SW4ixcvRl3NtLa26tixY8rOzlZ2drZeeOEFLV++XMFgUC0tLXr66ac1efJkLVq0KK6DAwCSm+cAHT16VPfdd1/k68/fv1m9erW2bt2q48eP65e//KXOnz+vUCikhQsX6kc/+pH8fn/8pgYAJD3PAZo/f76cu/G9s3/zm9/c0kAAgKGBu2EjZjV3/w/Pa9LNbhuYfH709w97XrNK0/4nB9/XFVI3V+l6N4/D4c0r+WGeVYvdf8ZaUwPAHhmrvaTdM8s1FwnjPaB4G/u+b84EHB6brdc3lr19frhbtDs/zsvAdIaQjbVvBSkG+9xNzoYZ/VBXNUvoQIA2CCAAEATBAgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECAAgAkCBAAwQYAAACa4GSliVvDP/+l5zez/+qDnNf/0jV96XpM/bJTnNZJUNvqM5zU7w3d4XvNf/neb5zbW/6CrwvEaS1ga8z/d3uzZ
4XlPYVu95DVIHV0AAABMECABgwnOADh8+rCVLligUCsnn82nv3r1Rz69Zs0Y0+ny9qW7x4cbzmBQCkC
M8B6u7uVlFRkaqrq2+4z+LFi9Xe3h7Zdu7ceUtDAgBSj+cPIZSVlamsrOxL1/j9fgWDwZiHAgCkvoS8B1RbW6vc3FxNmTJFjz/+uM6dO3fDtT09PQqHw1EbACD1xT1Aixcv1htvvKGamhq99NJLqqurU1lZmfr6+vpdX1VVpUAgENkKCgriPRIAYBCK+y8lXbFiReTPM2bM0MyZMzVp0iTV1tZqwYIF162vrKxURUVF5OtwOEyEAGAISPjHsCdOnKicnBw1Nzf3+7zf71dWVlbUBgBIfQkP0OnTp3Xu3Dnl5+cn+lAAgCTi+SW4ixcvRl3NtLa26tixY8rOzlZ2drZeeOEFLV++XMFgUC0tLXr66ac1efJkLVq0KK6DAwCSm+cAHT16VPfdd1/k68/fv1m9erW2bt2q48eP65e//KXOnz+vUCikhQsX6kc/+pH8fn/8pgYAJD3PAZo/f76cu/G9s3/zm9/c0kAAgKGBu2EjZjV3/w/Pa9LNbhuYfH709w97XrNK0/4nB9/XFVI3V+l6N4/D4c0r+WGeVYvdf8ZaUwPAHhmrvaTdM8s1FwnjPaB4G/u+b84EHB6brdc3lr19frhbtDs/zsvAdIaQjbVvBSkG+9xNzoYZ/VBXNUvoQIA2CCAAEATBAgAIAJAgQAMEGAAAAmCBAAwAQBAgCYIEAAABMECAAgAkCBAAwQYAAACa4GSliVvDP/+l5zez/+qDnNf/0jV96XpM/bJTnNZJUNvqM5zU7w3d4XvNf/neb5zbW/6CrwvEaS1ga8z/d3uzZ

```
ACQIEADBBgAAAJggQAMAEAQIAmCBAAAATBAgAYIIAAQBMECAAgAkCBAAwwc1IgVSW5huwQ+UM
G+15zaW8dM9rAp5XYLDiCggAYIIAAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCC
AAEATBAgAIAJAgQAMMHNSIFU9unlmJbtvjjW85pVmf83pmNh6OIKCABgggABAEx4ClBVVZVmz56tzM
xM5ebmatmyZWpqaoraZ/78+fL5fFHbY489FtehAQDJz1OA6urqVF5eroaGBh04cEC9vb1auHChuru7o/Z
bu3at2tvbI9vmzZvjOjQAIPl5+hDC/v37o77evn27cnNz1djYqHnz5kUeHzVqlILBYHwmBACkpFt6D6irq0u
SlJ2dHfX4G2+8oZycHE2fPl2VlZW6dOnSDb9HT0+PwuFw1AYASH0xfwy7r69PGzZs0Ny5czV9+vT
I4w89JAmTpyoUCik48eP65lnnlFTU5Peeutfr9PVVWVNm3aFOsYAIAkFXOAysvLdeLECb333ntRj69bty7y5xk
zZig/P18LFixQS0uLJk+efN33qaysVEVFReTrcDisgoKCWMcCACSJmAK0fv16vfPOOzp8+LDGjx//pfsWFxd
Lkpqbm/sNkN/vl9/vj2UMAEAS8xQg55yeeOIJ7dmzR7W7dzmzR7W1tSosLLzpmmPHjkmS8vPzYxoQAJCaPAWovLx
cO3bs0L59+5SZmamOjg5JUiAQ0MiRI9XS0qIdO3bo/vvv19ixY3X8+HE9+eSTmjdvnmbOnJmQfwAAQHL
yFKCtW7dKuvbLpn9t27ZtWrNmjTIyMnTw4EFt2bJF3d3dKigo0PLy5bV27dq4Dg0ASH0vIoVK/Tss8/GbWAAQGrw/CO4L1NQUKC
6urpbGggAMDRwN2wglaXH9n/xu0ecjmHV6JiOhaGLm5ECAEwQIACACQIEADBBgAAAJggQAMAEAQIA
mCBAAAATBAgAYIIAAQBMECAAgAkCBAAwQYAAACa4GSmQwq6e+8+Y1v3971d7XvPbot/EdCwMXV
wBAQBMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQAMDHo7gXn
nJMkfaZeyRkPAwxRV7t7PK8JX+jzfpwrlz2v+cz1el6DgfWZrv1v9Pm/z2/E5262xwA7ffq0CgoKrMcAANyit
rY2jR8//obPD7oA9fX16cyZM8rMzJTP54t6LhwOq6CgQG1tbcrKyjKa0B7n4RrOwzWch2s4D9cMhvPgnNPFixcVCoU0bNiNf9dn0P0KbtiwYV9aTEnKyspK6wvsc5yHazgP13AeruE8XGN9HgKBwE334UMIAAATBAgAYCKpAuT3+1VdXS2/3289iinOwzWch2s4D9dwHq5JpvMw6D6EAABID0n1CggAkDoIEADABAECAJggQAAAE0kToJqaGt19993KzMxUcXGxPvjgA+uRBtxzzz0nn88XtU2bNs16rIQ7fPiwFi9erFAoJJ/Pp71790Y975zTxo0blZ+fr5EjR6qkpEQnT560GTaBbnYe1qxZc931sWjRIpthE6Sqqkr33XefMjMzlZubq6VLl6qpqSlqn8uXL6u8vFxjx47V6NGjtXz5cnV2dhpNnBhf5TzMnz//uuvhscceM5q4f0kRoF27dqmiokLV1dX68MMPVVRUpNLSUp09e9Z6tAF37733qr29PbK999571iMlXHd3t4qKilRTU9Pv85s3b9Yrr7yiV199VUeOHNEdd9yh0tJSXb58eYAnTaybnQdJWrRoUdT1sXPnzgGcMPHq6upUXl6uhoYGHThwQL29vVq4cKG6u7sj+zz55JN6++23tXv3btXV1enMmTNatmyZ4dTx91XOgyStXbs26nrYvHmz0cQ34JLA7NmzXXl5eeTrq1evulAo5KqqqgynGnjV1dWuqKjIegxTktyePXsiX/f19blgMOhefPHFyGMXLlxwfr/f7dy502DCgfHF8+Ccc6tXr3ZLliwxmcfK2bNnnSRXV1fnnLv23378+PFu9+7dkX3+/Oc/O0muvr7easyE++J5cM65b33rW+573/ue3VBfwaB/BXTlyhU1NjaqpKQk8tiwYcNUUlKi+vp6w8lsnDx5UqFQSJMmTdLDDz+sU6dOWY9kqrW1VR0dHVHXRyAQUHFxcVpeH7W1tcrNzdXUqVP1+OOP6/z589YjJVRXV5ckKTs7W5LU2Nio3t7eqOth2rRpmjBhQkpfD188D5974403lJOTo+nTp6uyslKXLl2yGO+GBt3NSL/ok08+0dWrV5WXlxf1eF5env7yl78YTWWjuLhY27dv19SpU9Xe3q5Nmzbpm9/8pk6cOKHMzEzr8Ux0dHRIUr/Xx+fPpYtFixZp2bJlKiwsVEtLi77//e+rrKxM9fX1Gj58uPV4cdfX16cNGzZo7ty5mj59uqRr10NGRobGjBkTtW8qXw/9nQdJeuihhzRx4kSFQiEdP35czzzzjJqamvTWW28ZThtt0AcI/1RWVhb588yZM1VcXKyJEyfqzTff1KOPPmo4GQaDlStXRv48Y8YMzZw5U5MnT1Ztba0WLFhgOFlilJeX68SJE2nxPuiXudF5WLduXeTPM2bMUH5+vhYsWKCWlhZNnjx5oMfs16D/FVxOTo6GDx9+3adYOjs7FQwGjaYaHMaMGaN77rlHzc3N1qOY+fwa4Pq43qRJk5STk5OS18f69ev1zjvv6N133436z7cEg0FduXJFFy5ciNo/Va+HG52H/hQXF0vSoLoeBn2AMjIyNGvWLB06dCjyWF9fnw4dOqQ5c+YYTmbv008/VUtLi/Lz861HMVNYWKhgMBh1fYTDYR05ciTtr4/Tp0/r/PnzKXV9OOe0fv167dmzR4cOHVJhYWHU87NmzdKIESOiroempiadOnUqpa6Hm52H/hw7dkySBtf1YP0piK/i5z//ufP7/W779u3uT3/6k1u3bp0bM2aM6+josB5tQP3Hf/yHq62tda2tre63v/2tKykpcTk5Oe7s2bPWoyXUxYsX3UcffeQ++ugjJ8m9/PLL7qOPPnJ/+9vfnHPO/eAHP3BjxoyNxjzhw2sZy3r1Tc33//u//h4+nzBsGwIAgmCBABggAABMECAAgAkCBAAwQYAAACYIEADABAECAJggQAAAEwQIAGCCAAEATBAgAIAJAgQAMPH/fV8DsN1F1wgAAAAASUVORK5CYII=)"
```

`],`

```
  "metadata": {
   "id": "FJ7bxvXsyVRX"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "import numpy as np\n",
   "print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "xWliTbEkywVq",
   "outputId": "522dee15-e721-47e1-ed95-f3176a608890"
  },
  "execution_count": 27,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "[9]\n"
    ]
   }
  ]
 },
 {
  "cell_type": "markdown",
  "source": [
        "Save The model"
      ],
      "metadata": {
        "id": "Y-OIe8ZDy5Uz"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "# Save the model\n",
        "model.save('models/mnistCNN.h5')"
      ],
      "metadata": {
        "id": "ISOneam4y1pX"
      },
      "execution_count": 28,
      "outputs": []
    }
  ]
}
```