```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Importing the required libraries"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Requirement already satisfied: keras in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (2.7.0)\n",
      "Note: you may need to restart the kernel to use updated packages.\n"
     ]
    }
   ],
   "source": [
    "\n",
    "pip install keras"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {
    "id": "2dK82hXRz79l"
   },
   "outputs": [],
   "source": [
    "import numpy as np\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 tensorflow.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\n",
    "from tensorflow.keras.models import load_model\n",
    "from PIL import Image"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
    "metadata": {},
    "outputs": [],
    "source": [
     "#Loading the Data"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 5,
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "wAZkSWUW0I0J",
     "outputId": "1e13f7aa-88b4-49a4-c493-e2edab3649cb"
    },
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz\n",
       "11493376/11490434 [==============================] - 0s 0us/step\n",
       "11501568/11490434 [==============================] - 0s 0us/step\n"
      ]
     }
    ],
    "source": [
     "(x_train, y_train), (x_test, y_test)=mnist.load_data ()"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 6,
    "metadata": {},
    "outputs": [],
    "source": [
     "#Analyzing the Data"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 7,
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "Wen1dZwi0IuG",
     "outputId": "83481ea6-9d30-4ce5-8662-01e8b0331a17"
    },
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "(60000, 28, 28)\n",
       "(10000, 28, 28)\n"
      ]
     }
    ],
   "source": [
    "print (x_train.shape)  #shape is used for give the dimens ion values
#60000-rows 28x28-pixels\n",
    "print (x_test.shape)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "3tmeCbgW0Ioc",
    "outputId": "e49a4d33-1e7b-4146-c3ed-1d3c89bdbabe"
   },
   "outputs": [
    {
     "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,  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)"
       ]
      },
      "execution_count": 8,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "x_train[0]"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 9,
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 282
     },
     "id": "5FzjKsFf0IjS",
     "outputId": "0ad4f115-e12d-4b73-c9e0-21400381293c"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "<matplotlib.image.AxesImage at 0x7fafe06716a0>"
      ]
     },
     "execution_count": 9,
     "metadata": {},
     "output_type": "execute_result"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAPsAAAD4CAYAAAAq5pAIAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG
90bGliIHZlcnNpb24zLjUuMCwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8/fFQqAAAACXBIWXMAA
AsTAAALEwEAmpwYAAANUElEQVR4nO3df+xddX3H8der5dt29Ae2IrWDKlALA10o2zfVpc6woKx2
W4pLNFaHNSOWODGaGDPCssAfy9ItE2I2x1KlsSwMYyKMbnbDpiFjoFa+YIGWvgvWlibUVT
64/t974/vYflavvd8vz3n3HsufT8fyTf33vO+55x7TvrqOfd+7r0fR4QAnP6mtd0AgN4g7EAShB
1IgrADSRB2IIkzermzGZ4ZsS7l7sEUnlNP9exOOqJarXCbnulpK9Kmi7pGxGxGvuz
YJoMT22NaxVvky3vZ0SV+T9GFJl0paY/vSqRtD0FwMvl/RsRDwfEcckfUvS6qlpC8BUm6j
V9KL4x7vK5b9CtrbA/ZHjquozV2B6COOmGf6E2AN3z2NiI2RMRgRAwOaWsQoaGaN3QGoo07Y90laPO7
xeZL212sHQLfUCfvDkpbavsD2DEkflrRlatoCMNU6tjXEqE9PX7tjEidjXWGYBG1Rpnj4
gtkrY01AuALuLjskAShB1IgrADSRB2IAnCDiRB2IEkCDuQBGEHkiDsQBKEHUiCsANJEHYgCcIOJ
EHYgSQIO5AEYQeSIOxAEoQdSIKwA0kQdiAJwg4kQdiBJAg7kARhB5Ig7EAShB1IgrADSRB2IAnC
DiRB2IEkak3ZbHuvpCOSRiSdiIjBJpoC0LxaYS/8XkS/FAEnXDHpK+Z/sR2+smeoL
tdbaHbA8d19GauwNQVd3L+BURsd/2OZK22n4n4qIh4Y/4SI2CBpgyTN84KouT84AFdU6s0fE/uL2oK
R7JC1voikAzascdtuzbbc99/b6kqyTtbKoxKoxAM2qcxm/UNI9/fzr9/fzr9ExH820hWAxlUOe
nsB0EUMvQFJEHYgCcIOJEHYgSQIO5BEE1+EScb1+EwSSmX/yu0vovL5jfo07e6NBvD/NBvDJTWFzx1vNb2X7zm
RMfaDz/wD6Xrrvj+Z0vrx7e4+zmK1fcsOeO8+ldVnubXSO/l2bTft6LXXN9vSl6v/bLl5HWuOR4Sw6g97zvJ
Vs+Ho+r5Pmv5tPbszl7O6M3iPpo6lJmmThK1eegiE31ZBv/Z/eG+y0K58KtGhAcS6wLCpf0mLvyNs3Q
RMfaDz/wD6Xrrvj+Z0vrx47MKK1fcsOejrWRl/+ndN3TEWd2IAnCDiRB2IEkCDuQBGEHkiDsQBK
EHUiCcfaCzyg/FC9+ufPvclz7yfJv9r9/9r5/9l2l9eUzy8e638YOx0jH2uHR8nfWnf/t07au
G29FOMswM4TRF2IAnCDiRB2IEkCDuQBGEHkiDsQBKMsxfXkfxKf7n9d7f6o/df39tmn/frTNZ/eaRmlwnBy
iVp+97zO9e97//vvGB1ZyGW1Tp0CZYkDZf/FqI
lNaz4cwOJEHYgSQIO5AEYQeSIOxAEoQdSIKwA0kwzl54x5xuzvuzjrW5L+Ksrkzdtdu2
RJ39SWp/MhdpRfeWvlZdfmmT1i/RI5V17fvnv5a/57R9V3rYkdzdvu2
UGrBG+1/Uxxy1EF+txULuO/KWnlSctukLQtIpZK8Y8Btdg17RDwg6dBJi1dL2lz8vFnSNa02BaBhnb5
tC0qm/QLYyIYUkqbs/p9ETb62wP2R46rqMVdwegrq6/Gx8RGyJiCDiMCIGBzSjg5MDqA
ituDzbUEoBuqhn2zpLXFz2sl3dtMOwC6ZdzLeWWyCpFCvUWGyFCM2c0G9mi2JfWEwu5rMjS7
0sd2l9bc/Vn3b5bPXXwXLbwK9mAZW/Q2Gfj7EvKnJBJHG7Ge0LbcQXgWJZtoU7BuldXHUi0Bd1hBnGHb8
IKwA0kQdiAJwg4kQdiBJPiKK7pq2ty5tmpdNb7f0pVH/9/9ds1ne+/nzsqpRHcf5ngjBmKgwtI9bbdEqYFNbkAAWqmKGz
2OltLNFuKtRGOJtS9MSbaWWYtyUMUWL5WaDLQJM+OwEjhR9T07k4QWmnsJpZmavRX+fdM1L3U+1jYYb+
ByAFrtUFVWWBjFtOC2Ya7CCEpHUvL4BzQmYeAmbiWfxZNX7e9gxa0ZjIlc4/OhNpXVXuDCsnIGSt
RwANmFLYbQ9oLOh3RsTdxeIDthcV9UWSDtTcCnaZnVlE7Z4Di03kBTcCMZpZ0WRN1Jw5c2E91Hz6Z
ln//R5aX1q2f/qLT+qZ9+sLT+a/eWr4/emco4+wpJ10h6wvaFSR/tSocAGj
Fp2CPiQUMqUmdPhlxZbPtAOgWPsjtfYf+u57SuuL9f1a2/0Mfz5Drx5X2/fxfSmcd+Ptzv5fe/nsc/nHlk
0dzOLMDSRB2IAnCDiRB2IEkCDuQBGEHkiDsQBKM5/mpp15Zml9wRmHa23HfcdKa1Hra2jSSrZ
gSQIO5AEYQeSIOxAEoQdSIKwA0kQdiAJxtlPc8fed0lp/fzPaz33rxdn9oa6ebGEbNd6ot0aPn7S
CDiRB2IEkCDuQBGEHkiDsQBKEHUhiKvOzOL5S6S3xoMvvr2Zkmfr7Zkkfrf8VjSc8dQbI2JLtxptF
s+Xe8b5Rf8+2dK6xfdwfzrbxZT+VDNCUlfiohbdc+V9IjtruUXt7V5cTW5axNVHjSmfr7ZkcHCSkFQLvQVrd24a
O3dks7tdmMAmnVKr9ltny/pcknbi0XX237c9kbb8zuss872kO2h4zpar1sAlU057LbnSPqOpC9G
xCuSbpO0RNIyjZ35vzLRehGxISIGI2JwQDPrdwygkimF3faAxoJ+Z0TcLUkRcSAiRiJiVNLXJS3
vXpsA6po07LYt6XZJuyPilnHLF4172kck7Wy+PQBNmcq78SskXSPpCds7imU3Slpje5nGfi14r6
TrutAfapr5/KzS+kiMltbf+a+T7GB05NQaQmum8m78g5I8QYkxdeBNhE/QAUkQdiAJwg4kQdiBJ
Ag7kARhB5JwRO8m1Z3nBfFeX9mz/QHZbI9teiUOTTRUzpkdyIKwA0kQdiAJwg4kQdiBJAg7kARh
B5Lo6Ti77Zck/XTcorMlvdyzBk5Nv/bWr31J9FZVk729MyLeNlGhp2F/w87toYgYbK2BEv3aW7/
2JdFbVb3qjct4IAnCDiTRdtg3tLz/Mv3aW7/2JdFbVT3prdXX7AB6p+0zO4AeIexAEq2E3fZK20
/bftb2DW300IntvbafsL3D9lDLvWy0fdD2znHLFtjeavuZ4nbCOfZa6u1m2z8rjt0O26ta6m2x7
ftt77a9y/YXiuWtHruSvnpy3Hr+mt32dEk/kfQhSfskPSxpTUQ82dNGOrC9V9JgRLT+AQzbH5D0
qqQ7IuI9xbK/lXQoItYX/1HOj4g/75Pebpb0atvLeBezFS0aP824pKslfVotHruSvj6mHhy3Ns7
syyU9GxHPR8QxSd+StLqFPvpeRDwg6dBJi1dL2lTc36Sxfyw916G3vhARwxHxaHH/iKTXpxlv9d
iV9NUTbYT9XEkvjnu8T/0133tI+p7tR2yva7uuR9u8euy
vTndbUR9ol+H6ufxv9WRMRvSfqwpM8Vl6uYmilN490rE0wz3heqTn9eVxth3ydp8bjH50na30If
E4qI/cXtQUn3qP+moj7w+gy6xe3Blvv5f/00jfdE04yrD45dm9OftxH2hyUttX2B7RmSPi5pcwt
9vIHt2cUbJ7I9W9JV6r+pqDdLWlvcXyp3hZ7+RX9Mo13p2nG1fKxa33684jo+Z+kVRp7R/45SX
/RRg8d+rpQ0mPF3662e5N0l8Yu645r7IroWklvlbRN0jPF7YI+6u2fJT0u6TGNBWthS739tsbeGj4maXvxtarttSvpq5X1ym
R+XtKP4W9X2sSvpqyfHjY/LAknwCTogCcIOJEHYgSQIO5AEYQeSIOxAEoQdSOL/AHY66CnnnxKi
AAAAAElFTkSuQmCC\n",
       "text/plain": [
        "<Figure size 432x288 with 1 Axes>"
       ]
      },
      "metadata": {
       "needs_background": "light"
      },
      "output_type": "display_data"
     }
    ],
    "source": [
     "plt.imshow(x_train[5100])"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 10,
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "F5ay6xzO0IeR",
     "outputId": "7560c34d-d054-4f50-a8bb-20bfa6393704"
    },
    "outputs": [
     {
      "data": {
       "text/plain": [
        "0"
       ]
      },
      "execution_count": 10,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "np.argmax(y_train[5100])\n"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 11,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Reshaping the Data"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {
    "id": "fAkstQT30IZF"
   },
   "outputs": [],
   "source": [
    "x_train=x_train.reshape (60000, 28, 28, 1).astype('float32')\n",
    "x_test=x_test.reshape (10000, 28, 28, 1).astype ('float32')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Applying one hot encoding"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {
    "id": "UL0O6qCB0IT7"
   },
   "outputs": [],
   "source": [
    "classes = 10 "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {
    "id": "IgzsiiVJ0IKb"
   },
   "outputs": [],
   "source": [
    "y_train = np_utils.to_categorical (y_train, classes) \n",
    "y_test = np_utils.to_categorical (y_test, classes)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Adding  CNN Buliding"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {},
   "outputs": [],
   "source": [
    "model=Sequential()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation='relu'))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 19,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.add(Conv2D(64,(3,3),activation='relu'))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.add(Flatten())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 21,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.add(Dense(classes,activation='softmax'))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Compiling The Model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {},
   "outputs": [],
   "source": [

"model.compile(loss='categorical_crossentropy',optimizer=\"Adam\",metrics=[
'accuracy'])"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Training the Model"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Epoch 1/5\n",
      "1875/1875 [==============================] - 203s 108ms/step - loss:
0.3131 - accuracy: 0.9523 - val_loss: 0.1013 - val_accuracy: 0.9697\n",
      "Epoch 2/5\n",
      "1875/1875 [==============================] - 205s 109ms/step - loss:
0.0625 - accuracy: 0.9811 - val_loss: 0.0810 - val_accuracy: 0.9788\n",
      "Epoch 3/5\n",
      "1875/1875 [==============================] - 204s 109ms/step - loss:
0.0430 - accuracy: 0.9868 - val_loss: 0.0837 - val_accuracy: 0.9795\n",
      "Epoch 4/5\n",
      "1875/1875 [==============================] - 204s 109ms/step - loss:
0.0331 - accuracy: 0.9900 - val_loss: 0.1042 - val_accuracy: 0.9761\n",
      "Epoch 5/5\n",
      "1875/1875 [==============================] - 204s 109ms/step - loss:
0.0290 - accuracy: 0.9917 - val_loss: 0.0984 - val_accuracy: 0.9810\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "<keras.callbacks.History at 0x7fafe03dc430>"
      ]
     },
     "execution_count": 25,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [

"model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,batch_s
ize=32)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 26,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Observing The Metrics"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 27,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Metrice(Test loss & Test Accuracy):\n",
     "[0.09839355200529099, 0.9810000061988831]\n"
    ]
   }
  ],
  "source": [
   "metrics=model.evaluate(x_test,y_test,verbose=0)\n",
   "print(\"Metrice(Test loss & Test Accuracy):\")\n",
   "print(metrics)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 28,
  "metadata": {},
  "outputs": [],
  "source": [
   "#Test the Model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 29,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[[2.66373620e-13 5.15261426e-21 1.50421166e-13 1.66002376e-08\n",
     "  2.93580552e-20 7.28840418e-18 6.59416025e-23 1.00000000e+00\n",
     "  1.48631650e-12 5.62014844e-12]\n",
     " [1.01646545e-10 1.14599290e-17 1.00000000e+00 3.88194745e-18\n",
     "  1.34460339e-18 7.51504113e-23 2.78395113e-10 8.21130562e-20\n",
     "  2.91733380e-13 7.73047336e-22]\n",
     " [1.32042760e-12 1.00000000e+00 1.19916399e-09 1.56342046e-16\n",
     "  7.32658212e-10 3.01171761e-11 2.41586612e-10 6.36250774e-10\n",
     "  3.01473499e-11 2.16507127e-15]\n",
     " [1.00000000e+00 8.08110351e-20 1.30329358e-11 2.95739436e-15\n",
     "  1.90499827e-16 1.85495303e-14 5.43617018e-12 9.05207373e-14\n",
     "  3.09776564e-13 2.79464452e-10]]\n"
    ]
   }
  ],
  "source": [
   "prediction=model.predict(x_test[:4])\n",
   "print(prediction)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 30,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[7 2 1 0]\n",
     "[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
     " [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
     " [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
     " [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]\n"
    ]
   }
  ],
  "source": [
   "print(np.argmax(prediction,axis=1))\n",
   "print(y_test[:4])"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 31,
  "metadata": {},
  "outputs": [],
  "source": [
   "#Saving the model"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 32,
  "metadata": {},
  "outputs": [],
  "source": [
   "model.save(\"Model/digitrec.h5\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 33,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[Errno 2] No such file or directory: 'models'\n",
     "/home/wsuser/work\n"
    ]
   }
  ],
  "source": [
   "cd models"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 34,
  "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "tar: digitrec.h5: Cannot stat: No such file or directory\r\n",
      "tar: Exiting with failure status due to previous errors\r\n"
     ]
    }
   ],
   "source": [
    "!tar -zcvf hdr_deployment.tgz digitrec.h5"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 35,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "hdr_deployment.tgz\r\n",
      "\u001b[0m\u001b[01;34mModel\u001b[0m/\r\n"
     ]
    }
   ],
   "source": [
    "ls -1"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 36,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Collecting watson-machine-learning-client\n",
      "  Downloading watson_machine_learning_client-1.0.391-py3-none-
any.whl (538 kB)\n",
      "\u001b[K     |████████████████████████████████| 538 kB 14.7 MB/s eta
0:00:01\n",
      "\u001b[?25hRequirement already satisfied: urllib3 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-
machine-learning-client) (1.26.7)\n",
      "Requirement already satisfied: ibm-cos-sdk in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-
machine-learning-client) (2.11.0)\n",
      "Requirement already satisfied: certifi in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(2022.9.24)\n",
      "Requirement already satisfied: lomond in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(0.3.3)\n",
      "Requirement already satisfied: pandas in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(1.3.4)\n",
      "Requirement already satisfied: tqdm in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(4.62.3)\n",
      "Requirement already satisfied: requests in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(2.26.0)\n",
      "Requirement already satisfied: boto3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(1.18.21)\n",
      "Requirement already satisfied: tabulate in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(0.8.9)\n",
      "Requirement already satisfied: botocore<1.22.0,>=1.21.21 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (1.21.41)\n",
      "Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (0.5.0)\n",
      "Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-
machine-learning-client) (0.10.0)\n",
      "Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client)
(2.8.2)\n",
      "Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1-
>botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client)
(1.15.0)\n",
      "Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson-machine-learning-client) (2.11.0)\n",
      "Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson-machine-learning-client) (2.11.0)\n",
      "Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests-
>watson-machine-learning-client) (2.0.4)\n",
      "Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests-
>watson-machine-learning-client) (3.3)\n",
      "Requirement already satisfied: pytz>=2017.3 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas-
>watson-machine-learning-client) (2021.3)\n",
      "Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas-
>watson-machine-learning-client) (1.20.3)\n",
      "Installing collected packages: watson-machine-learning-client\n",
      "Successfully installed watson-machine-learning-client-1.0.391\n"
     ]
    }
   ],
   "source": [
    "!pip install watson-machine-learning-client --upgrade"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [
     {
      "data": {
       "text/plain": [
        "<ibm_watson_machine_learning.client.APIClient at 0x7fafe05a9e80>"
       ]
      },
      "execution_count": 37,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "from ibm_watson_machine_learning import APIClient\n",
     "credentials ={\n",
     "    \"url\":\"https://us-south.ml.cloud.ibm.com\",\n",
     "    \"apikey\":\"uX5EJ0Do-je3p0Yinppb9WDJ3tg6EerRFrNYzmNMVrm8\"\n",
     "}\n",
     "client = APIClient(credentials)\n",
     "client"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 38,
    "metadata": {},
    "outputs": [
     {
      "data": {
       "text/plain": [
        "{'resources': []}"
       ]
      },
      "execution_count": 38,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "client.spaces.get_details()"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 39,
    "metadata": {},
    "outputs": [],
    "source": [
     "def guid_from_space_name(client,deploy):\n",
     "  space = client.spaces.get_details()\n",
     "  return (next(item for item in space['resources'] if
item['entity']['name']==deploy)['metadata']['id'])\n",
     " "
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 41,
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
     "output_type": "stream",
     "text": [
      "Space UID = cca72fe8-1ea2-4559-a71c-c401ad862870\n"
     ]
    }
   ],
   "source": [
    "space_uid = guid_from_space_name(client,'Classification')\n",
    "print(\"Space UID = \" + space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 42,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'SUCCESS'"
      ]
     },
     "execution_count": 42,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "client.set.default_space(space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 43,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "-------------------------------  -----------------------------------
-  ----\n",
      "NAME                             ASSET_ID
TYPE\n",
      "default_py3.6                    0062b8c9-8b7d-44a0-a9b9-
46c416adcbd9  base\n",
      "kernel-spark3.2-scala2.12        020d69ce-7ac1-5e68-ac1a-
31189867356a  base\n",
      "pytorch-onnx_1.3-py3.7-edt       069ea134-3346-5748-b513-
49120e15d288  base\n",
      "scikit-learn_0.20-py3.6          09c5a1d0-9c1e-4473-a344-
eb7b665ff687  base\n",
      "spark-mllib_3.0-scala_2.12       09f4cff0-90a7-5899-b9ed-
1ef348aebdee  base\n",
      "pytorch-onnx_rt22.1-py3.9        0b848dd4-e681-5599-be41-
b5f6fccc6471  base\n",
      "ai-function_0.1-py3.6            0cdb0f1e-5376-4f4d-92dd-
da3b69aa9bda  base\n",
      "shiny-r3.6                       0e6e79df-875e-4f24-8ae9-
62dcc2148306  base\n",
      "tensorflow_2.4-py3.7-horovod     1092590a-307d-563d-9b62-
4eb7d64b3f22  base\n",
      "pytorch_1.1-py3.6                10ac12d6-6b30-4ccd-8392-
3e922c096a92  base\n",
      "tensorflow_1.15-py3.6-ddl        111e41b3-de2d-5422-a4d6-
bf776828c4b7  base\n",
      "autoai-kb_rt22.2-py3.10          125b6d9a-5b1f-5e8d-972a-
b251688ccf40  base\n",
      "runtime-22.1-py3.9               12b83a17-24d8-5082-900f-
0ab31fbfd3cb  base\n",
      "scikit-learn_0.22-py3.6          154010fa-5b3b-4ac1-82af-
4d5ee5abbc85  base\n",
      "default_r3.6                     1b70aec3-ab34-4b87-8aa0-
a4a3c8296a36  base\n",
      "pytorch-onnx_1.3-py3.6           1bc6029a-cc97-56da-b8e0-
39c3880dbbe7  base\n",
      "kernel-spark3.3-r3.6             1c9e5454-f216-59dd-a20e-
474a5cdf5988  base\n",
      "pytorch-onnx_rt22.1-py3.9-edt    1d362186-7ad5-5b59-8b6c-
9d0880bde37f  base\n",
      "tensorflow_2.1-py3.6             1eb25b84-d6ed-5dde-b6a5-
3fbdf1665666  base\n",
      "spark-mllib_3.2                  20047f72-0a98-58c7-9ff5-
a77b012eb8f5  base\n",
      "tensorflow_2.4-py3.8-horovod     217c16f6-178f-56bf-824a-
b19f20564c49  base\n",
      "runtime-22.1-py3.9-cuda          26215f05-08c3-5a41-a1b0-
da66306ce658  base\n",
      "do_py3.8                         295addb5-9ef9-547e-9bf4-
92ae3563e720  base\n",
      "autoai-ts_3.8-py3.8              2aa0c932-798f-5ae9-abd6-
15e0c2402fb5  base\n",
      "tensorflow_1.15-py3.6            2b73a275-7cbf-420b-a912-
eae7f436e0bc  base\n",
      "kernel-spark3.3-py3.9            2b7961e2-e3b1-5a8c-a491-
482c8368839a  base\n",
      "pytorch_1.2-py3.6                2c8ef57d-2687-4b7d-acce-
01f94976dac1  base\n",
      "spark-mllib_2.3                  2e51f700-bca0-4b0d-88dc-
5c6791338875  base\n",
      "pytorch-onnx_1.1-py3.6-edt       32983cea-3f32-4400-8965-
dde874a8d67e  base\n",
      "spark-mllib_3.0-py37             36507ebe-8770-55ba-ab2a-
eafe787600e9  base\n",
      "spark-mllib_2.4                  390d21f8-e58b-4fac-9c55-
d7ceda621326  base\n",
      "autoai-ts_rt22.2-py3.10          396b2e83-0953-5b86-9a55-
7ce1628a406f  base\n",
      "xgboost_0.82-py3.6               39e31acd-5f30-41dc-ae44-
60233c80306e  base\n",
      "pytorch-onnx_1.2-py3.6-edt       40589d0e-7019-4e28-8daa-
fb03b6f4fe12  base\n",
      "pytorch-onnx_rt22.2-py3.10       40e73f55-783a-5535-b3fa-
0c8b94291431  base\n",
      "default_r36py38                  41c247d3-45f8-5a71-b065-
8580229facf0  base\n",
      "autoai-ts_rt22.1-py3.9           4269d26e-07ba-5d40-8f66-
2d495b0c71f7  base\n",
      "autoai-obm_3.0                   42b92e18-d9ab-567f-988a-
4240ba1ed5f7  base\n",
      "pmml-3.0_4.3                        493bcb95-16f1-5bc5-bee8-
81b8af80e9c7  base\n",
      "spark-mllib_2.4-r_3.6               49403dff-92e9-4c87-a3d7-
a42d0021c095  base\n",
      "xgboost_0.90-py3.6                  4ff8d6c2-1343-4c18-85e1-
689c965304d3  base\n",
      "pytorch-onnx_1.1-py3.6              50f95b2a-bc16-43bb-bc94-
b0bed208c60b  base\n",
      "autoai-ts_3.9-py3.8                 52c57136-80fa-572e-8728-
a5e7cbb42cde  base\n",
      "spark-mllib_2.4-scala_2.11          55a70f99-7320-4be5-9fb9-
9edb5a443af5  base\n",
      "spark-mllib_3.0                     5c1b0ca2-4977-5c2e-9439-
ffd44ea8ffe9  base\n",
      "autoai-obm_2.0                      5c2e37fa-80b8-5e77-840f-
d912469614ee  base\n",
      "spss-modeler_18.1                   5c3cad7e-507f-4b2a-a9a3-
ab53a21dee8b  base\n",
      "cuda-py3.8                          5d3232bf-c86b-5df4-a2cd-
7bb870a1cd4e  base\n",
      "autoai-kb_3.1-py3.7                 632d4b22-10aa-5180-88f0-
f52dfb6444d7  base\n",
      "pytorch-onnx_1.7-py3.8              634d3cdc-b562-5bf9-a2d4-
ea90a478456b  base\n",
      "spark-mllib_2.3-r_3.6               6586b9e3-ccd6-4f92-900f-
0f8cb2bd6f0c  base\n",
      "tensorflow_2.4-py3.7                65e171d7-72d1-55d9-8ebb-
f813d620c9bb  base\n",
      "spss-modeler_18.2                   687eddc9-028a-4117-b9dd-
e57b36f1efa5  base\n",
      "pytorch-onnx_1.2-py3.6              692a6a4d-2c4d-45ff-a1ed-
b167ee55469a  base\n",
      "spark-mllib_2.3-scala_2.11          7963efe5-bbec-417e-92cf-
0574e21b4e8d  base\n",
      "spark-mllib_2.4-py37                7abc992b-b685-532b-a122-
a396a3cdbaab  base\n",
      "caffe_1.0-py3.6                     7bb3dbe2-da6e-4145-918d-
b6d84aa93b6b  base\n",
      "pytorch-onnx_1.7-py3.7              812c6631-42b7-5613-982b-
02098e6c909c  base\n",
      "cuda-py3.6                          82c79ece-4d12-40e6-8787-
a7b9e0f62770  base\n",
      "tensorflow_1.15-py3.6-horovod       8964680e-d5e4-5bb8-919b-
8342c6c0dfd8  base\n",
      "hybrid_0.1                          8c1a58c6-62b5-4dc4-987a-
df751c2756b6  base\n",
      "pytorch-onnx_1.3-py3.7              8d5d8a87-a912-54cf-81ec-
3914adaa988d  base\n",
      "caffe-ibm_1.0-py3.6                 8d863266-7927-4d1e-97d7-
56a7f4c0a19b  base\n",
      "spss-modeler_17.1                   902d0051-84bd-4af6-ab6b-
8f6aa6fdeabb  base\n",
      "do_12.10                            9100fd72-8159-4eb9-8a0b-
a87e12eefa36  base\n",
      "do_py3.7                            9447fa8b-2051-4d24-9eef-
5acb0e3c59f8  base\n",
      "spark-mllib_3.0-r_3.6               94bb6052-c837-589d-83f1-
f4142f219e32  base\n",
      "cuda-py3.7-opence                   94e9652b-7f2d-59d5-ba5a-
23a414ea488f  base\n",
      "nlp-py3.8                        96e60351-99d4-5a1c-9cc0-
473ac1b5a864  base\n",
      "cuda-py3.7                       9a44990c-1aa1-4c7d-baf8-
c4099011741c  base\n",
      "hybrid_0.2                       9b3f9040-9cee-4ead-8d7a-
780600f542f7  base\n",
      "spark-mllib_3.0-py38             9f7a8fc1-4d3c-5e65-ab90-
41fa8de2d418  base\n",
      "autoai-kb_3.3-py3.7              a545cca3-02df-5c61-9e88-
998b09dc79af  base\n",
      "spark-mllib_3.0-py39             a6082a27-5acc-5163-b02c-
6b96916eb5e0  base\n",
      "runtime-22.1-py3.9-do            a7e7dbf1-1d03-5544-994d-
e5ec845ce99a  base\n",
      "default_py3.8                    ab9e1b80-f2ce-592c-a7d2-
4f2344f77194  base\n",
      "tensorflow_rt22.1-py3.9          acd9c798-6974-5d2f-a657-
ce06e986df4d  base\n",
      "kernel-spark3.2-py3.9            ad7033ee-794e-58cf-812e-
a95f4b64b207  base\n",
      "autoai-obm_2.0 with Spark 3.0    af10f35f-69fa-5d66-9bf5-
acb58434263a  base\n",
      "default_py3.7_opence             c2057dd4-f42c-5f77-a02f-
72bdbd3282c9  base\n",
      "tensorflow_2.1-py3.7             c4032338-2a40-500a-beef-
b01ab2667e27  base\n",
      "do_py3.7_opence                  cc8f8976-b74a-551a-bb66-
6377f8d865b4  base\n",
      "spark-mllib_3.3                  d11f2434-4fc7-58b7-8a62-
755da64fdaf8  base\n",
      "autoai-kb_3.0-py3.6              d139f196-e04b-5d8b-9140-
9a10ca1fa91a  base\n",
      "spark-mllib_3.0-py36             d82546d5-dd78-5fbb-9131-
2ec309bc56ed  base\n",
      "autoai-kb_3.4-py3.8              da9b39c3-758c-5a4f-9cfd-
457dd4d8c395  base\n",
      "kernel-spark3.2-r3.6             db2fe4d6-d641-5d05-9972-
73c654c60e0a  base\n",
      "autoai-kb_rt22.1-py3.9           db6afe93-665f-5910-b117-
d879897404d9  base\n",
      "tensorflow_rt22.1-py3.9-horovod  dda170cc-ca67-5da7-9b7a-
cf84c6987fae  base\n",
      "autoai-ts_1.0-py3.7              deef04f0-0c42-5147-9711-
89f9904299db  base\n",
      "tensorflow_2.1-py3.7-horovod     e384fce5-fdd1-53f8-bc71-
11326c9c635f  base\n",
      "default_py3.7                    e4429883-c883-42b6-87a8-
f419d64088cd  base\n",
      "do_22.1                          e51999ba-6452-5f1f-8287-
17228b88b652  base\n",
      "autoai-obm_3.2                   eae86aab-da30-5229-a6a6-
1d0d4e368983  base\n",
      "tensorflow_rt22.2-py3.10         f65bd165-f057-55de-b5cb-
f97cf2c0f393  base\n",
      "do_20.1                          f686cdd9-7904-5f9d-a732-
01b0d6b10dc5  base\n",
      "pytorch-onnx_rt22.2-py3.10-edt   f8a05d07-e7cd-57bb-a10b-
23f1d4b837ac  base\n",
      "scikit-learn_0.19-py3.6          f963fa9d-4bb7-5652-9c5d-
8d9289ef6ad9  base\n",
      "tensorflow_2.4-py3.8             fe185c44-9a99-5425-986b-
59bd1d2eda46  base\n",
      "------------------------------  ------------------------------------
-  ----\n"
     ]
    }
   ],
   "source": [
    "client.software_specifications.list(limit=100)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 44,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'acd9c798-6974-5d2f-a657-ce06e986df4d'"
      ]
     },
     "execution_count": 44,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "software_space_uid =
client.software_specifications.get_uid_by_name('tensorflow_rt22.1-
py3.9')\n",
    "software_space_uid"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 48,
   "metadata": {},
   "outputs": [],
   "source": [
    "model_details =
client.repository.store_model(model='hdr_deployment.tgz',meta_props={\n",
    "    client.repository.ModelMetaNames.NAME:\"Digit Recognition
System\",\n",
    "    client.repository.ModelMetaNames.TYPE:\"tensorflow_2.7\",\n",
    "
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid\n",
    "})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 49,
   "metadata": {
    "scrolled": true
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'entity': {'hybrid_pipeline_software_specs': [],\n",
       "  'software_spec': {'id': 'acd9c798-6974-5d2f-a657-
ce06e986df4d',\n",
       "   'name': 'tensorflow_rt22.1-py3.9'},\n",
       "  'type': 'tensorflow_2.7'},\n",
       " 'metadata': {'created_at': '2022-11-13T12:57:18.607Z',\n",
       "  'id': 'ee04c4b7-ea90-4d1b-aa13-3259091a19c9',\n",
       "  'modified_at': '2022-11-13T12:57:22.476Z',\n",
       "  'name': 'Digit Recognition System',\n",
       "  'owner': 'IBMid-663002IV3Z',\n",
       "  'resource_key': 'f2e40b5f-7218-465e-a4c8-ed7a137f9781',\n",
       "  'space_id': 'cca72fe8-1ea2-4559-a71c-c401ad862870'},\n",
       " 'system': {'warnings': []}}"
      ]
     },
     "execution_count": 49,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "model_details"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 50,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'ee04c4b7-ea90-4d1b-aa13-3259091a19c9'"
      ]
     },
     "execution_count": 50,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "model_id = client.repository.get_model_id(model_details)\n",
    "model_id"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 51,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Successfully saved model content to file:
'DigitRecog_IBM_model.tar.gz'\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "'/home/wsuser/work/DigitRecog_IBM_model.tar.gz'"
       ]
      },
      "execution_count": 51,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "client.repository.download(model_id,'DigitRecog_IBM_model.tar.gz')"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 52,
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "DigitRecog_IBM_model.tar.gz  hdr_deployment.tgz
\u001b[0m\u001b[01;34mModel\u001b[0m/\r\n"
      ]
     }
    ],
    "source": [
     "ls"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 53,
    "metadata": {},
    "outputs": [],
    "source": [
     "#Test with Saved Model"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 57,
    "metadata": {},
    "outputs": [],
    "source": [
     "from tensorflow.keras.models import load_model\n",
     "from keras.preprocessing import image\n",
     "from PIL import Image\n",
     "import numpy as np"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 60,
    "metadata": {},
    "outputs": [],
    "source": [
     "model = load_model(\"Model/digitrec.h5\")"
    ]
   },
   {
    "cell_type": "code",
   "execution_count": 62,
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "import os, types\n",
    "import pandas as pd\n",
    "from botocore.client import Config\n",
    "import ibm_boto3\n",
    "\n",
    "def __iter__(self): return 0\n",
    "\n",
    "# @hidden_cell\n",
    "# The following code accesses a file in your IBM Cloud Object Storage.
It includes your credentials.\n",
    "# You might want to remove those credentials before you share the
notebook.\n",
    "cos_client = ibm_boto3.client(service_name='s3',\n",
    "    ibm_api_key_id='bSERpNH2Xkz8r_sYJqmAMF3Wx1azB_b2ZyfoRcIaj2OG',\n",
    "    ibm_auth_endpoint=\"https://iam.cloud.ibm.com/oidc/token\",\n",
    "    config=Config(signature_version='oauth'),\n",
    "    endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')\n",
    "\n",
    "bucket = 'anovelmethodforhandwrittendigitre-donotdelete-pr-
g3go0qmnp30anx'\n",
    "object_key = 'test1.png'\n",
    "\n",
    "streaming_body_1 = cos_client.get_object(Bucket=bucket,
Key=object_key)['Body']\n",
    "\n",
    "# Your data file was loaded into a botocore.response.StreamingBody
object.\n",
    "# Please read the documentation of ibm_boto3 and pandas to learn more
about the possibilities to load the data.\n",
    "# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-
python/\n",
    "# pandas documentation: http://pandas.pydata.org/\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 64,
   "metadata": {},
   "outputs": [],
   "source": [
    "img = Image.open(streaming_body_1).convert(\"L\") # convert image to
monochrome\n",
    "img = img.resize( (28,28) ) # resizing of input image"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 65,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAABwAAAAcCAAAAABXZoBIAAAB0klEQVR4nH3TTWgTQRQH8P/MbD
a7MY22QpeKURSpWGKxBz/IoaAXq/XoSaGFguDdQ/EuCF6CxfZS0CBSBA9qEEtvheLBgwgp5qAGE
XIIqImtu5vsR+Z5qDNJFDunGX4M7/GfN4zw/8V3MRg9e/9bu4GD2X8xWK98qVVrvj08feuAQrZT
s3r3HQm/7oIlrVzxcB/+uFkxIwqaLkwTRv7xQG9DDyvJ4Nf2QP68HXEWvCn03vw6t9XyJy+fs0r
hvttuzPe/zO0oEdHi+PixC1tEFBO9Sg9l0jckERFxANiM485MBoAApmc9Ybwu65rhTw7rrOp/3g
FvlTR6TZJ79irMXgsgStsKXZ917JTO5WpSytpHhR3OIZnGU8cj8ioKU0bEG3WN4kSHorLCwRHG/
VI375OM8Elh4rSU5ovvGocNZsY6vqkMj2r3NSY4DEvj6KQUKC4r/MCBVDf4uXSCiTsLEgDw5JGZ
YBM6W6Li6NiRM9nZz0T0POs4I0c3iYj+TMJM9anT5GvvL/H6Rhhb7vWcfjIAYWEljPwYadFq2+7
U0mAvQj5YbpMXCgjbm1gZQh8Cb++VGcVgMl9w8DfCfbbaIOvQxStqJFn/xLcgzO6J7fYdfgPyzL
5zIgAJiwAAAABJRU5ErkJggg==\n",
      "text/plain": [
       "<PIL.Image.Image image mode=L size=28x28 at 0x7FAFAFC831C0>"
      ]
     },
     "execution_count": 65,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "img"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 66,
   "metadata": {},
   "outputs": [],
   "source": [
    "im2arr = np.array(img) #converting to image\n",
    "im2arr = im2arr.reshape(1, 28, 28, 1) #reshaping according to our
requirement"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 67,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[[9.9121350e-01 4.1121837e-14 5.6992202e-12 1.1115554e-08
3.7515914e-05\n",
      "  4.4451827e-08 2.5928662e-07 3.3449016e-08 1.4326091e-07
8.7485956e-03]]\n"
     ]
    }
   ],
   "source": [
    "pred = model.predict(im2arr)\n",
    "print(pred)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 68,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[0]\n"
     ]
    }
   ],
   "source": [
    "print(np.argmax(pred, axis=1)) #printing our Labels"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "colab": {
   "collapsed_sections": [],
   "provenance": []
  },
  "kernelspec": {
   "display_name": "Python 3.9",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.13"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 1
}
```