Sprint 1

```
{
  "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": null,
      "metadata": {
        "id": "Mu6nSJIecEkq"
      },
      "outputs": [],
      "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"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "(x_train, y_train), (x_test, y_test)=mnist.load_data ()
#splitting the mnist data into train and test"
      ],
      "metadata": {
        "id": "aCcoR9IbcJ7z"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz\n",
        "11490434/11490434 [==============================] - 1s
0us/step"
      ],
      "metadata": {
        "id": "eK8KcqX5cQNy"
      }
    },
    {
      "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": {
        "id": "dHjyggiycTcj"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "(60000, 28, 28)\n",
        "(10000, 28, 28)"
      ],
      "metadata": {
        "id": "H9OCiCUccYQ6"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "x_train[0]"
      ],
      "metadata": {
        "id": "yKTc6HYgcbjU"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
       "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": {
       "id": "G4PlafH6ceYz"
     }
   },
   {
     "cell_type": "code",
     "source": [
       "plt.imshow(x_train[6000])     #ploting the index=image"
     ],
     "metadata": {
       "id": "ZgiGA9_Fcn_b"
     },
     "execution_count": null,
     "outputs": []
   },
   {
     "cell_type": "markdown",
     "source": [

     ],
   "metadata": {
     "id": "60r31_r4creT"
   }

 },
```

```
  {
    "cell_type": "code",
    "source": [
      "numpy.argmax(y_train[6000])"
    ],
    "metadata": {
      "id": "Wv5q_ZXTcw9j"
    },
    "execution_count": null,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "source": [
      "0"
    ],
    "metadata": {
        "id": "48UNMbUGc1LT"
      }
    },
    {
      "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": "RSi2BdQKc36s"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "number_of_classes = 10  #storing the no of classes in a 
variable"
      ],
      "metadata": {
        "id": "M8y_pNh7c7jS"
      },
      "execution_count": null,
      "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": "0LEAzEb9c9o7"
      },
      "execution_count": null,
      "outputs": []
    }
  ]
}
```