{

"cells": [

{

"cell\_type": "markdown",

"id": "cb9b0d77",

"metadata": {},

"source": [

"# Sprint 1 "

]

},

{

"cell\_type": "markdown",

"id": "f1a993c6",

"metadata": {},

"source": [

"Team ID - PNT2022TMID27424"

]

},

{

"cell\_type": "markdown",

"id": "c62525cb",

"metadata": {},

"source": [

"# Importing the required libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "a875e559",

"metadata": {},

"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 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": "markdown",

"id": "bff1e551",

"metadata": {},

"source": [

"# Load data"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "210ec263",

"metadata": {},

"outputs": [],

"source": [

"(x\_train, y\_train), (x\_test, y\_test)=mnist.load\_data () #splitting the mnist data into train and test"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "43acb466",

"metadata": {},

"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": 4,

"id": "d57d5fcf",

"metadata": {},

"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, 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": 4,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"x\_train[0]"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "d215316f",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<matplotlib.image.AxesImage at 0x19f743b9540>"

]

},

"execution\_count": 5,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 640x480 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"plt.imshow(x\_train[6000]) #ploting the index=image"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "6ba125be",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"0"

]

},

"execution\_count": 8,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.argmax(y\_train[6000])"

]

},

{

"cell\_type": "markdown",

"id": "d7402576",

"metadata": {},

"source": [

"# Reshaping Dataset"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"id": "fccf0fa6",

"metadata": {},

"outputs": [],

"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')"

]

},

{

"cell\_type": "markdown",

"id": "4ce45662",

"metadata": {},

"source": [

"# Applying One Hot Encoding"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "5004cb43",

"metadata": {},

"outputs": [],

"source": [

"number\_of\_classes = 10 #storing the no of classes in a variable"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "ed95b6f2",

"metadata": {},

"outputs": [],

"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)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "014d6fa0",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"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": "3.10.8"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}