{

"cells": [

{

"cell\_type": "markdown",

"id": "41a37808",

"metadata": {},

"source": [

"# Sprint 2"

]

},

{

"cell\_type": "markdown",

"id": "44a28e66",

"metadata": {},

"source": [

"Team ID - PNT2022TMID27424"

]

},

{

"cell\_type": "markdown",

"id": "c55e52c8",

"metadata": {},

"source": [

"# Importing the required libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 43,

"id": "8e99c7de",

"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": "code",

"execution\_count": 44,

"id": "8e338d4d",

"metadata": {},

"outputs": [],

"source": [

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

"x\_train=x\_train.reshape (60000, 28, 28, 1).astype('float32')\n",

"x\_test=x\_test.reshape (10000, 28, 28, 1).astype ('float32')\n",

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

"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": "markdown",

"id": "88006d15",

"metadata": {},

"source": [

"# Add CNN Layers"

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"id": "4423493c",

"metadata": {},

"outputs": [],

"source": [

"#create model\n",

"model=Sequential ()"

]

},

{

"cell\_type": "code",

"execution\_count": 46,

"id": "f8a2be7f",

"metadata": {},

"outputs": [],

"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'))"

]

},

{

"cell\_type": "code",

"execution\_count": 47,

"id": "311f8cb0",

"metadata": {},

"outputs": [],

"source": [

"#flatten the dimension of the image\n",

"model.add(Flatten())"

]

},

{

"cell\_type": "code",

"execution\_count": 48,

"id": "d9566989",

"metadata": {},

"outputs": [],

"source": [

"#output layer with 10 neurons\n",

"model.add(Dense(number\_of\_classes,activation = 'softmax'))"

]

},

{

"cell\_type": "markdown",

"id": "d52d8b17",

"metadata": {},

"source": [

"# Compiling the model"

]

},

{

"cell\_type": "code",

"execution\_count": 49,

"id": "b532e03b",

"metadata": {},

"outputs": [],

"source": [

"#Compile model\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 50,

"id": "586131c8",

"metadata": {},

"outputs": [],

"source": [

"x\_train = np.asarray(x\_train)\n",

"y\_train = np.asarray(y\_train)"

]

},

{

"cell\_type": "markdown",

"id": "a0d313ad",

"metadata": {},

"source": [

"# Train the model"

]

},

{

"cell\_type": "code",

"execution\_count": 51,

"id": "de52342f",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Epoch 1/5\n",

"1875/1875 [==============================] - 120s 63ms/step - loss: 0.2808 - accuracy: 0.9487 - val\_loss: 0.1263 - val\_accuracy: 0.9649\n",

"Epoch 2/5\n",

"1875/1875 [==============================] - 95s 50ms/step - loss: 0.0734 - accuracy: 0.9780 - val\_loss: 0.0947 - val\_accuracy: 0.9733\n",

"Epoch 3/5\n",

"1875/1875 [==============================] - 88s 47ms/step - loss: 0.0524 - accuracy: 0.9839 - val\_loss: 0.1133 - val\_accuracy: 0.9701\n",

"Epoch 4/5\n",

"1875/1875 [==============================] - 97s 52ms/step - loss: 0.0375 - accuracy: 0.9884 - val\_loss: 0.1308 - val\_accuracy: 0.9720\n",

"Epoch 5/5\n",

"1875/1875 [==============================] - 96s 51ms/step - loss: 0.0275 - accuracy: 0.9912 - val\_loss: 0.1233 - val\_accuracy: 0.9781\n"

]

},

{

"data": {

"text/plain": [

"<keras.callbacks.History at 0x237357554b0>"

]

},

"execution\_count": 51,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#fit the model\n",

"model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=5, batch\_size=32)"

]

},

{

"cell\_type": "markdown",

"id": "27ba0285",

"metadata": {},

"source": [

"# Observing the metrics"

]

},

{

"cell\_type": "code",

"execution\_count": 52,

"id": "290f5219",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Metrics (Test loss &Test Accuracy) : \n",

"[0.12333168834447861, 0.9781000018119812]\n"

]

}

],

"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)"

]

},

{

"cell\_type": "code",

"execution\_count": 53,

"id": "b3659ce9",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"1/1 [==============================] - 0s 93ms/step\n",

"[[1.1050688e-15 9.2188809e-14 5.0320639e-15 5.0875104e-10 2.6267043e-01\n",

" 1.0428948e-08 5.2634429e-17 2.9271017e-09 6.4425700e-07 7.3732901e-01]]\n"

]

}

],

"source": [

"prediction=model.predict(x\_test[6000:6001])\n",

"print(prediction)"

]

},

{

"cell\_type": "code",

"execution\_count": 54,

"id": "feafc23b",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<matplotlib.image.AxesImage at 0x2373479ccd0>"

]

},

"execution\_count": 54,

"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\_test[6000])"

]

},

{

"cell\_type": "code",

"execution\_count": 55,

"id": "cdc78aac",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[9]\n"

]

}

],

"source": [

"import numpy as np\n",

"print(np.argmax(prediction, axis=1)) #printing our Labels from first 4 images"

]

},

{

"cell\_type": "code",

"execution\_count": 56,

"id": "062d3d96",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"9"

]

},

"execution\_count": 56,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.argmax(y\_test[6000:6001]) #printing the actual labels"

]

},

{

"cell\_type": "markdown",

"id": "6141fdb1",

"metadata": {},

"source": [

"# Save The model"

]

},

{

"cell\_type": "code",

"execution\_count": 57,

"id": "b70c484c",

"metadata": {},

"outputs": [],

"source": [

"# Save the model\n",

"model.save('models/mnistCNN.h5')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "8acdff0f",

"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

}