Sprint 3


```
{
  "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": "H6CkIaFBdcWz"
      },
      "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 ()\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)"
      ],
      "metadata": {
        "id": "xtNucAcEe8L5"
      },
      "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": "FEMwsb2ifLRY"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "#create model\n",
        "model=Sequential ()"
      ],
      "metadata": {
        "id": "tD3dFiLffN7h"
      },
      "execution_count": null,
      "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": "M9wDHut1fSra"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "#flatten the dimension of the image\n",
        "model.add(Flatten())"
      ],
      "metadata": {
        "id": "03NE0R1HfVOp"
```

```
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "#output layer with 10 neurons\n",
        "model.add(Dense(number_of_classes,activation = 'softmax'))"
      ],
      "metadata": {
        "id": "pYwcyRE5fYI7"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "#Compile model\n",
        "model.compile(loss= 'categorical_crossentropy',
optimizer=\"Adam\", metrics=['accuracy'])"
      ],
      "metadata": {
        "id": "OsYZK3D4fa7A"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "x_train = numpy.asarray(x_train)\n",
        "y_train = numpy.asarray(y_train)"
      ],
      "metadata": {
        "id": "X7qLL2b8fdAy"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "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": {
        "id": "mOkuwBJDfe6A"
      },
      "execution_count": null,
      "outputs": []
    },
```

```
{
      "cell_type": "markdown",
      "source": [
        "Epoch 1/5\n",
        "1875/1875 [==============================] - 131s 70ms/step -
loss: 0.1966 - accuracy: 0.9541 - val_loss: 0.0806 - val_accuracy:
0.9743\n",
        "Epoch 2/5\n",
        "1875/1875 [==============================] - 130s 69ms/step -
loss: 0.0678 - accuracy: 0.9792 - val_loss: 0.0840 - val_accuracy:
0.9765\n",
        "Epoch 3/5\n",
        "1875/1875 [==============================] - 130s 69ms/step -
loss: 0.0477 - accuracy: 0.9852 - val_loss: 0.0886 - val_accuracy:
0.9751\n",
        "Epoch 4/5\n",
        "1875/1875 [==============================] - 130s 69ms/step -
loss: 0.0359 - accuracy: 0.9885 - val_loss: 0.0955 - val_accuracy:
0.9761\n",
        "Epoch 5/5\n",
        "1875/1875 [==============================] - ETA: 0s - loss:
0.0283 - accuracy: 0.9913\n"
      ],
      "metadata": {
        "id": "3gImU05ffirb"
      }
    },
    {
      "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": {
        "id": "fYDYMBJKfk0e"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "Metrics (Test loss &Test Accuracy) : \n",
        "[0.09420406073331833, 0.9811000227928162]"
      ],
      "metadata": {
        "id": "-UC_brXOfn7h"
      }
    },
    {
```

```
      "cell_type": "code",
      "source": [
        "prediction=model.predict(x_test[6000:6001])\n",
        "print(prediction)"
      ],
      "metadata": {
        "id": "jznt5o-CfqLu"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "1/1 [==============================] - 0s 79ms/step\n",
        "[[1.2785279e-11 8.0596178e-17 1.4344616e-14 8.5613865e-06
2.0496884e-05\n",
        "  8.0490082e-08 3.6100053e-15 9.7431900e-04 4.2339255e-07
9.9899608e-01]]"
      ],
      "metadata": {
        "id": "jQ7BLG-qftW7"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import numpy as np\n",
        "print(np.argmax(prediction, axis=1)) #printing our Labels from
first 4 images"
      ],
      "metadata": {
        "id": "Bcx3egjCfvG1"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "[9]"
      ],
      "metadata": {
        "id": "fChxmsJafyNt"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "np.argmax(y_test[6000:6001]) #printing the actual labels"
      ],
      "metadata": {
        "id": "-3toAlR0f0zz"
      },
```

```
      "execution_count": null,
      "outputs": []
    },
```



```
    {
      "cell_type": "markdown",
      "source": [
        "9"
      ],
      "metadata": {
        "id": "7y3RhOuff3Ru"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "# Save the model\n",
        "model.save('models/mnistCNN.h5')"
      ],
      "metadata": {
        "id": "d4FL52Lzf5B6"
      },
      "execution_count": null,
      "outputs": []
    }
  ]
}
```