CONTENTS

Department of Computer Science and Engineering

CANDIDATE'S DECLARATION

I hereby, declare that the work presented in this report, entitled "**Radio Signal Classification**", in fulfilment of the requirement for the award of the degree Bachelor of Engineering in Computer Science and Engineering, submitted in CSE Department, Chandigarh College of Engineering and Technology (Degree Wing), affiliated to Panjab University, Chandigarh, is an authentic record of our own work carried out during my degree under the guidance of Snehan Kekre. The work reported in this has not been submitted by me for award of any other degree or diploma.

Date: 13/03/2021

Place: CCET

Aayushi Aggarwal
(CO17502)

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project work entitled "**Radio Signal Classification**", submitted by **Aayushi Aggarwal (CO17502),** fulfillment for the requirements of the award of Bachelor of Engineering Degree in Computer Science & Engineering at Chandigarh College of Engineering and Technology (Degree Wing), Chandigarh is an authentic work carried out by him/her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Date: 19/03/2021

Place: CCET

Dr. Varun Gupta
Dept of CSE

CCET (Degree Wing),
Chandigarh

ABSTRACT

Radio Signal Classification refers to the classification of radio signals obtained from the outer space in the form of histograms into various categories. It uses 2D spectrograms of deep space radio signals collected by the Allen Telescope Array at the SETI Institute. SETI stands for Search for Extraterrestrial Intelligence. Radio Signal Classification as performed in the following implementation considers the spectrograms, obtained from the outer space, as images. Those images are then used to train an image classification model to classify the signals into one of four classes. Convolutional neural network has been used to classify signals from space.

Thus, the Keras API has been used along with TensorFlow as its backend to build and train a neural network model to solve an image classification problem.

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Science and Engineering

ACKNOWLEDGEMENT

"Any serious and lasting achievement or success, one can never achieve without the help, guidance and co-operation of so many people involved in the work.

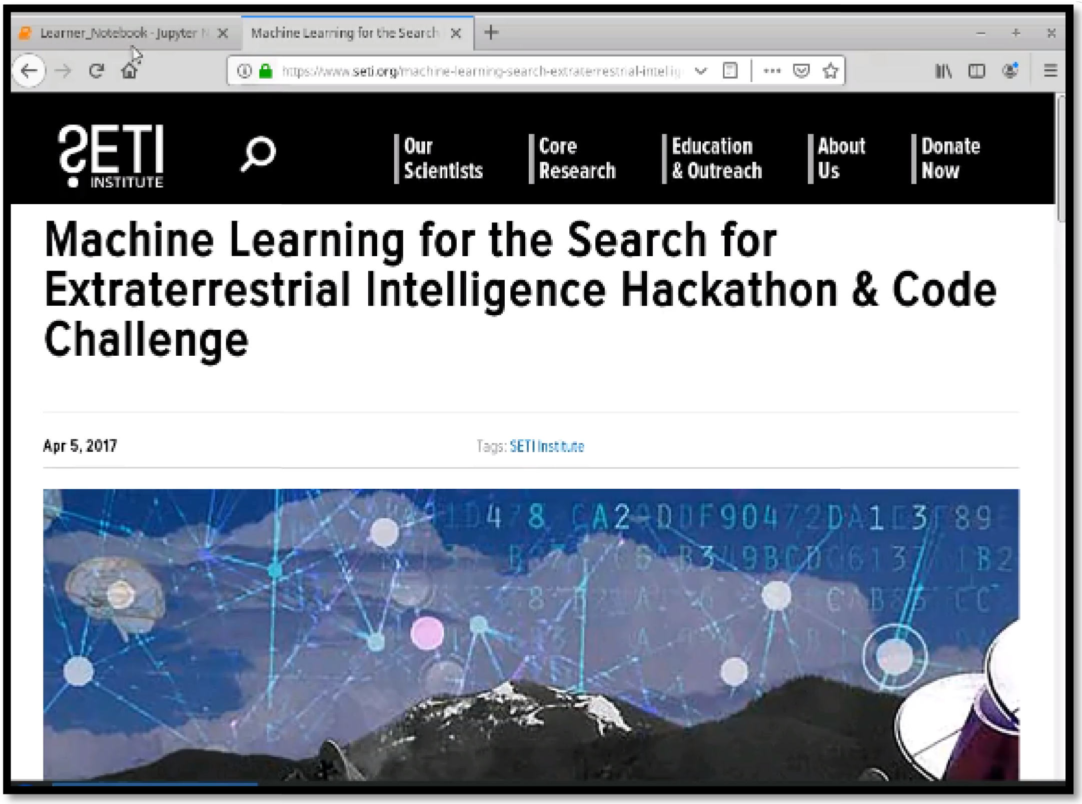
I would like to express deep gratitude to Dr. Sunil K. Singh, Head of Department (Computer Science & Engineering), submitted in CSE Department, Chandigarh College of Engineering & Technology(Degree wing),and affiliated to Punjab University, Chandigarh, without whose permission the training would not be possible. I would also like to thank Dr. Ankit Gupta, Training & Placement Officer, CSE Department, and Dr. Varun Gupta, CSE department , who guided me for taking up this training.

I have tried my best to keep report simple yet technically correct. I hope I succeed in my attempt.

LIST OF TABLES

FIGURE-1 DATASET on the website of Kaggle

- The main dataset has been taken from the website **seti.org**.



- There are two separate datasets. One is used as training dataset and other is used as test dataset.
- The train dataset has two csv files :-
  - **Label.csv** = It contains 800 tuples.
  - **Images.csv** = It contains the images of histogram taken from outer space.
- The validation dataset is the dataset used for test the trained model. It has two csv files :-

- **Label.csv =** It contains 800 tuples.
- **Images.csv** = It contains the images of histogram taken from outer space.

<u>FIGURE – 2</u> DATASET

	A	B	C	D	E
1	1	0	0	0	
2	1	0	0	0	
3	1	0	0	0	
4	1	0	0	0	
5	1	0	0	0	
6	1	0	0	0	
7	1	0	0	0	
8	1	0	0	0	
9	1	0	0	0	
10	1	0	0	0	
11	1	0	0	0	
12	1	0	0	0	
13	1	0	0	0	
14	1	0	0	0	
15	1	0	0	0	

LIST OF FIGURES

Figure: - 1 Importing Libraries

Task 1: Import Libraries

```
In [1]: from livelossplot.tf_keras import PlotLossesCallback
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        import tensorflow as tf

        from sklearn.metrics import confusion_matrix
        from sklearn import metrics

        import numpy as np
        np.random.seed(42)
        import warnings;warnings.simplefilter('ignore')
        %matplotlib inline
        print('Tensorflow version:', tf.__version__)

        Tensorflow version: 2.1.0
```

Figure: - 2 Loading the dataset.

Task 2: Load and Preprocess SETI Data

```
In [*]: train_images = pd.read_csv('dataset/train/images.csv', header=None)
        train_labels = pd.read_csv('dataset/train/labels.csv', header=None)

        val_images = pd.read_csv('dataset/validation/images.csv', header=None)
        val_labels = pd.read_csv('dataset/validation/labels.csv', header=None)

In [ ]: train_images.head(3)

In [ ]: train_labels.head(3)
```

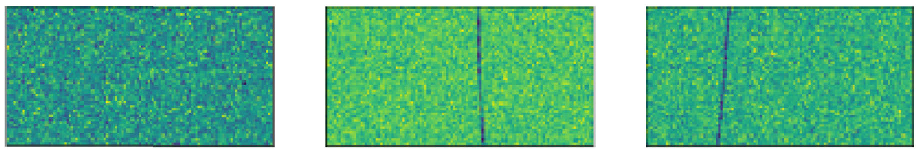
Figure: - 3 Preprocessing the dataset.

```
In [11]: print("Training set shape: ", train_images.shape, train_labels.shape)
         print("Validation set shape: ", val_images.shape, val_labels.shape)

         Training set shape:  (3200, 8192) (3200, 4)
         Validation set shape:  (800, 8192) (800, 4)

In [17]: x_train = train_images.values.reshape(3200, 64, 128, 1)
         x_val = val_images.values.reshape(800, 64, 128, 1)

         y_train = train_labels.values
         y_val = val_labels.values
```

Figure: - 4 Plotting 2D Spectrogram.

Task 3: Plot 2D Spectrograms

```
In [24]: plt.figure(0, figsize=(12, 12))
         for i in range(1, 4):
             plt.subplot(1, 3, i)
             img = np.squeeze(x_train[np.random.randint(0, x_train.shape[0])])
             plt.xticks([])
             plt.yticks([])
             plt.imshow(img)
```



```
In [23]: plt.imshow(np.squeeze(x_train[6]), cmap='gray')
```

Figure: - 5 Create training and validation data generators.

```
## Task 4: Create Training and Validation Data Generators

In [26]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

         datagen_train = ImageDataGenerator(horizontal_flip=True)
         datagen_train.fit(x_train)

         datagen_val = ImageDataGenerator(horizontal_flip=True)
         datagen_val.fit(x_val)
```

Figure: - 6 Import libraries to create the CNN model.

Task 5: Creating the CNN Model

```
In [28]: from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D
         from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D

         from tensorflow.keras.models import Model, Sequential
         from tensorflow.keras.optimizers import Adam
         from tensorflow.keras.callbacks import ModelCheckpoint
```

Figure: - 7 Creating the CNN model.

```
In [*]: # Initialising the CNN
        model = Sequential()

        # 1st Convolution
        model.add(Conv2D(32, (5,5), padding='same', input_shape=(64,128,1)))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Dropout(0.25))

        # 2nd Convolution layer
        model.add(Conv2D(64, (5,5), padding='same'))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Dropout(0.25))

        # Flattening
        model.add(Flatten())

        # Fully connected layer
        model.add(Dense(1024))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(Dropout(0.4))

        model.add(Dense(4, activation='softmax'))
```

Figure: - 8 Learning Rate Scheduling.

Task 6: Learning Rate Scheduling and Compile the Model

```
In [34]: initial_learning_rate = 0.005
         lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
             initial_learning_rate = initial_learning_rate,
             decay_steps=5,
             decay_rate=0.96,
             staircase=True
         )

         optimizer = Adam(learning_rate=lr_schedule)
```

Figure: - 9 Compiling the CNN Model.

```
In [38]: model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
         model.summary()
```

Figure: - 10 Training the CNN Model.

Task 7: Training the Model

```
In [*]: checkpoint = ModelCheckpoint('model_weight.h5', monitor='va_loss',
                                     save_weights_only=True, mode='min', verbose=0)
        callbacks = [PlotLossesCallback(), checkpoint]

        batch_size = 32
        history = model.fit(
            datagen_train.flow(x_train, y_train, batch_size=batch_size, shuffle=True),
            steps_per_epoch = len(x_train) // batch_size,
            validation_data = datagen_val.flow(x_val, y_val, batch_size=batch_size, shuffle=True),
            validation_steps = len(x_val) // batch_size,
            epochs = 12,
            callbacks = callbacks
        )
```

Figure: - 11 Model Evaluation.

Task 8: Model Evaluation

```
In [43]: model.evaluate(x_val, y_val)

         800/800 [==============================] - 1s 638us/sample - loss: 0.3615 - accuracy: 0.7475

Out[43]: [0.3615160181099782, 0.7475]
```

Figure: - 12 Generate confusion matrix.

```
In [44]: from sklearn.metrics import confusion_matrix
         from sklearn import metrics
         import seaborn as sns

         y_true = np.argmax(y_val, 1)
         y_pred = np.argmax(model.predict(x_val), 1)
         print(metrics.classification_report(y_true, y_pred))

                       precision    recall  f1-score   support

                    0       1.00      0.99      0.99       200
                    1       0.50      0.88      0.64       200
                    2       0.49      0.12      0.20       200
                    3       1.00      1.00      1.00       200

             accuracy                           0.75       800
            macro avg       0.75      0.75      0.71       800
         weighted avg       0.75      0.75      0.71       800
```

Chapter – 1 INTRODUCTION

1.1 MACHINE LEARNING

Machine Learning is now one of the most important topics around the world. Well, it can even be said as the new electricity in today's world. But to be precise what is Machine Learning; well it's just one way of teaching the machine by feeding the large amount of data. Machine learning is defined in 90's by Arthur Samuel described as the," it is a field of study that gives the ability to the computer for self-learn without being explicitly programmed", that means imbuing knowledge to machines without hard-coding it. And also "A computer algorithm/program is said to learn from performance measure P and experience E with some class of tasks T if its performance at tasks in T, as measured by P, improves with experience E." -Tom M. Mitchell.

Machine learning is mainly focused on the development of computer programs which can teach themselves to grow and change when exposed to new data. Machine learning studies algorithms for self-learning to do stuff. It can process massive data faster with the learning algorithm. For instance, it will be interested in learning to complete a task, make accurate predictions, or behave intelligently.

1.2 SIGNIFICANCE OF MACHINE LEARNING

Data is growing day by day, and it is impossible to understand all of the data with higher speed and higher accuracy. More than 80% of the data is unstructured that is audios, videos, photos, documents, graphs, etc. Finding patterns in data on planet earth is impossible for human brains. The data has been very massive, the time taken to compute would increase, and this is where Machine Learning comes into action, to help people with significant data in minimum time. The ability to process very large amount of data within seconds cannot be done by a human. Decision making has also been improved.

1.3 TYPES OF MACHINE LEARNING

1. **Supervised learning: -** Supervised Learning is the first type of machine learning, in which labelled data used to train the algorithms. In supervised learning, algorithms are trained using marked data, where the input and the output are

known. We input the data in the learning algorithm as a set of inputs, which is called as Features, denoted by X along with the corresponding outputs, which is indicated by Y, and the algorithm learns by comparing its actual production with correct outputs to find errors. It then modifies the model accordingly. The raw data divided into two parts. The first part is for training the algorithm, and the other region used for test the trained algorithm. Supervised learning uses the data patterns to predict the values of additional data for the labels. This method will commonly use in applications where historical data predict likely upcoming events.

Ex:- It can anticipate when transactions are likely to be fraudulent or which insurance customer is expected to file a claim. We will focus more in this in this project.

2. **Unsupervised learning :-** Unsupervised Learning is the second type of machine learning, in which unlabeled data are used to train the algorithm, which means it used against data that has no historical labels. What is being showing must figure out by the algorithm? The purpose is to explore the data and find some structure within. In unsupervised learning the data is unlabeled, and the input of raw information directly to the algorithm without pre-processing of the data and without knowing the output of the data and the data cannot divide into a train or test data. The algorithm figures out the data and according to the data segments, it makes clusters of data with new labels. This learning technique works well on transactional data. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns. Or it can find the primary qualities that separate customer segments from each other. These algorithms are also used to segment text topics, recommend items and identify data outliers.

3. **Reinforcement learning: -** Reinforcement Learning is the third type of machine learning in which no raw data is given as input instead reinforcement learning algorithm have to figures out the situation on their own. The reinforcement learning frequently used for robotics, gaming, and navigation. With reinforcement learning, the algorithm discovers through trial and error which actions yield the most significant rewards. This type of training has three main components which are the agent which can describe as the learner or decision maker, the environment which described as everything the agent interacts with and actions which represented as what the agent can do. The objective is for the agent to take actions that maximise the expected reward over a given measure of time. The agent will reach the goal much quicker by following a good policy. So the purpose of reinforcement learning is to learn the best plan.

1.4 Project Relevance

In this project, supervised learning has been done. Keras API has been used along with TensorFlow as its backend to build and train a convolutional neural network to classify signals from space. The current signal detection system is programmed for only particular kinds of signals such as narrow-band carrier waves however, the detection system sometimes triggers the signals that are not narrow-band signals with some unknown efficiency and are also not explicitly known frequency interference. So there seem to be various categories of these kinds of events that have been observed in the recent past. So the goal of this project, is to build an image classification model to classify these signals accurately in real-time so this may allow the signal detection system to make better observational decisions and thereby increasing the efficiency of the night scans, allowing for explicit detection of signal types.

Chapter – 2 DEEP LEARNING AND NEURAL NETWORKS

2.1 Deep Learning

Deep learning (also known as **deep structured learning**) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. It is also known as deep neural learning or deep neural network.

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
- Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
- Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

2.2 Neural Networks

A **neural network** is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.[1] Thus a neural network is an artificial neural network, for solving artificial intelligence (AI) problems, analogous to either a biological neural network, made up of real biological neurons. The connections of the biological neuron are modelled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

Thus, we describe a neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

2.3 Convolutional neural network

A **Convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that scan the hidden layers and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that does multiplication or other dot product, and its activation function is commonly ReLU. This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.

- **Convolutional layers**
  In a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (input channels). After passing through a convolutional layer, the

image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the convolution kernel/filter (the input channels) must equal the number channels (depth) of the input feature map.
- The hyperparameters of the convolution operation, like padding size and stride.

➢ **Pooling layers**

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. There are two common types of pooling: max and average. **Max pooling** uses the maximum value of each cluster of neurons at the prior layer, while **average pooling** instead uses the average value.

➢ **Fully connected layers**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

➢ **Receptive field**

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every neuron of the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's **receptive field.** Typically the area is a square (e.g., 5 by 5 neurons). (So, in a fully connected layer, the receptive field is the entire previous layer.) Thus in each convolutional layer, each neuron takes input from a larger area of pixels in the input image than previous layers. This is due to applying the convolution over and over, which takes into account the value of a pixel and its surrounding pixels. In dilated CNNs, the number of pixels in the receptive field remains invariant but the field gets more sparsely populated as its dimensions grow when assessing the influence of different layers.

➢ **Weights**

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.

2.4 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of platforms like TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. It focuses on being user-friendly, modular, and extensible. Keras is designed to enable fast experimentation with deep neural networks.

Using Keras in deep learning allows for easy and fast prototyping as well as running seamlessly on CPU and GPU. This framework is written in Python code which is easy to debug and allows ease for extensibility. The main advantages of Keras are described below:

- **User-Friendly:** Keras has a simple, consistent interface optimized for common use cases which provides clear and actionable feedback for user errors.
- **Modular and Composable:** Keras models are made by connecting configurable building blocks together, with few restrictions.
- **Easy To Extend:** With the help of Keras, you can easily write custom building blocks for new ideas and researches.
- **Easy To Use**: Keras offers consistent & simple APIs which helps in minimizing the number of user actions required for common use cases, also it provides clear and actionable feedback upon user error.

Chapter – 3 IMPLEMENTATION

3.1 Step-1 Import Modules

The first step is to import the python modules that are to be used during the project. Thus, the following modules are imported:

- **Matplotlib** - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Here, we use it to plot various graphs during the model making.
- **NumPy** - NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- **Pandas - pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Tensorflow** – It is the core open source library to help you develop and train ML models. I is imported to run keras on it.

Task 1: Import Libraries

```
[1]: from livelossplot.tf_keras import PlotLossesCallback
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     import tensorflow as tf

     from sklearn.metrics import confusion_matrix
     from sklearn import metrics

     import numpy as np
     np.random.seed(42)
     import warnings;warnings.simplefilter('ignore')
     %matplotlib inline
     print('Tensorflow version:', tf.__version__)

     Tensorflow version: 2.1.0
```

3.2 Step-2 Importing the dataset

We use read.csv() function of the panda module to import our dataset.

Task 2: Load and Preprocess SETI Data

```
In [*]: train_images = pd.read_csv('dataset/train/images.csv', header=None)
        train_labels = pd.read_csv('dataset/train/labels.csv', header=None)

        val_images = pd.read_csv('dataset/validation/images.csv', header=None)
        val_labels = pd.read_csv('dataset/validation/labels.csv', header=None)

In [ ]: train_images.head(3)

In [ ]: train_labels.head(3)
```

Output:

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	8182	
0	0.631373	0.623529	0.713726	0.705882	0.658824	0.666667	0.654902	0.635294	0.647059	0.705882	...	0.682353	0.61
1	0.725490	0.752941	0.749020	0.701961	0.690196	0.721569	0.709804	0.745098	0.654902	0.721569	...	0.721569	0.69
2	0.717647	0.701961	0.713726	0.733333	0.705882	0.717647	0.725490	0.682353	0.717647	0.674510	...	0.709804	0.69

3 rows × 8192 columns

Out[4]:

	0	1	2	3
0	1.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0

3.3 Step-3 Preprocessing the dataset

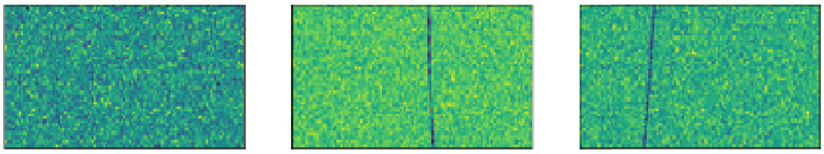
Here, we check the shape and size coordinates of the dataset and assign them to the variables.

```
In [11]: print("Training set shape: ", train_images.shape, train_labels.shape)
         print("Validation set shape: ", val_images.shape, val_labels.shape)

         Training set shape:  (3200, 8192) (3200, 4)
         Validation set shape:  (800, 8192) (800, 4)

In [17]: x_train = train_images.values.reshape(3200, 64, 128, 1)
         x_val = val_images.values.reshape(800, 64, 128, 1)

         y_train = train_labels.values
         y_val = val_labels.values
```

3.4 Step-4 Plotting 2D Spectrogram.

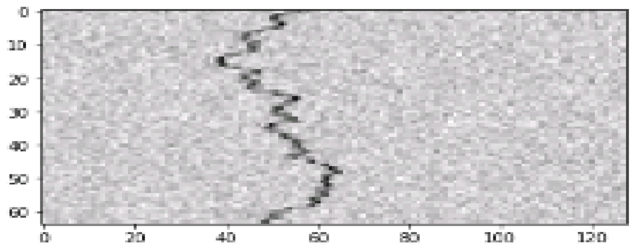
Here, we plot the histogram of the radio signals we received from the dataset.

```
In [24]: plt.figure(0, figsize=(12, 12))
         for i in range(1, 4):
             plt.subplot(1, 3, i)
             img = np.squeeze(x_train[np.random.randint(0, x_train.shape[0])])
             plt.xticks([])
             plt.yticks([])
             plt.imshow(img)
```



Output:

```
In [23]: plt.imshow(np.squeeze(x_train[6]), cmap='gray')

Out[23]: <matplotlib.image.AxesImage at 0x7f5289446ac8>
```



3.5 Step-5 Create training and validation data generators.

Here, we create training and validation data generators using ImageDataGenerator() from Keras library.

```
In [26]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

         datagen_train = ImageDataGenerator(horizontal_flip=True)
         datagen_train.fit(x_train)

         datagen_val = ImageDataGenerator(horizontal_flip=True)
         datagen_val.fit(x_val)
```

3.6 Step-6 Import libraries to create the CNN model.

From Keras, we import various functions to create our CNN model such as Dense, Conv20 etc.

```
In [28]: from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D
         from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D

         from tensorflow.keras.models import Model, Sequential
         from tensorflow.keras.optimizers import Adam
         from tensorflow.keras.callbacks import ModelCheckpoint
```

3.7 Step-7 Creating the CNN model.

Now, we will create the CNN model as follows:

1. Initialize the CNN model
2. Create first convolution layer using add() in model library.
3. Create second convolution layer using add() in model library.
4. Perform flattening operation on the CNN model
5. Create a fully connected layer and add it to the CNN model.

```
In [*]: # Initialising the CNN
        model = Sequential()

        # 1st Convolution
        model.add(Conv2D(32, (5,5), padding='same', input_shape=(64,128,1)))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Dropout(0.25))

        # 2nd Convolution layer
        model.add(Conv2D(64, (5,5), padding='same'))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Dropout(0.25))

        # Flattening
        model.add(Flatten())

        # Fully connected layer
        model.add(Dense(1024))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(Dropout(0.4))

        model.add(Dense(4, activation='softmax'))
```

3.8 Step-8 Learning Rate Scheduling.

Now, initialize the learning rate and learning rate scheduler using functions like optimizer, scheduler etc.

```
In [34]: initial_learning_rate = 0.005
         lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
             initial_learning_rate = initial_learning_rate,
             decay_steps=5,
             decay_rate=0.96,
             staircase=True
         )

         optimizer = Adam(learning_rate=lr_schedule)
```

3.9 Step-9 Compiling the CNN Model.

Now compile the CNN model using compile().

```
In [38]: model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
         model.summary()
```

Output:

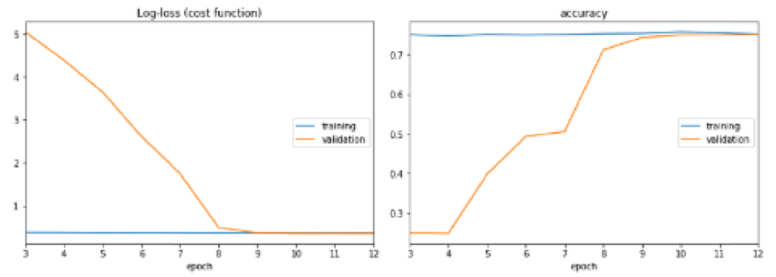
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 128, 32)       832
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 128, 32)       128
_________________________________________________________________
activation (Activation)      (None, 64, 128, 32)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 64, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 64, 64)        51264
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 64, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 32, 64, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 32, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              33555456
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
```

3.10 Step - 10 Training the CNN Model.

Now train the CNN model using checkpoint() and fit().

```
In [*]: checkpoint = ModelCheckpoint('model_weight.h5', monitor='va_loss',
                                     save_weights_only=True, mode='min', verbose=0)
        callbacks = [PlotLossesCallback(), checkpoint]

        batch_size = 32
        history = model.fit(
            datagen_train.flow(x_train, y_train, batch_size=batch_size, shuffle=True),
            steps_per_epoch = len(x_train) // batch_size,
            validation_data = datagen_val.flow(x_val, y_val, batch_size=batch_size, shuffle=True),
            validation_steps = len(x_val) // batch_size,
            epochs = 12,
            callbacks = callbacks
        )
```

Output:




```
Log-loss (cost function):
training   (min:    0.371, max:    0.545, cur:    0.371)
validation (min:    0.361, max:    5.066, cur:    0.361)

accuracy:
training   (min:    0.716, max:    0.758, cur:    0.752)
validation (min:    0.249, max:    0.751, cur:    0.751)
100/100 [==============================] - 6s 55ms/step - loss: 0.3711 - accuracy: 0.7516 - val
_loss: 0.3610 - val_accuracy: 0.7513
```

3.11 Step - 11 Model Evaluations.

Now, use evaluate() to check the loss and accuracy of the model.

```
In [43]: model.evaluate(x_val, y_val)

         800/800 [==============================] - 1s 638us/sample - loss: 0.3615 - accuracy: 0.7475

Out[43]: [0.3615160181099782, 0.7475]
```

3.12 Step - 12 Generate confusion matrix.

Now, create the confusion matrix.

```
In [44]: from sklearn.metrics import confusion_matrix
         from sklearn import metrics
         import seaborn as sns

         y_true = np.argmax(y_val, 1)
         y_pred = np.argmax(model.predict(x_val), 1)
         print(metrics.classification_report(y_true, y_pred))

                       precision    recall  f1-score   support

                    0       1.00      0.99      0.99       200
                    1       0.50      0.88      0.64       200
                    2       0.49      0.12      0.20       200
                    3       1.00      1.00      1.00       200

             accuracy                           0.75       800
            macro avg       0.75      0.75      0.71       800
         weighted avg       0.75      0.75      0.71       800
```

Chapter – 4 PROJECT FEATURES

4.1 Features of the Project

- In this Project, Using the CNN network we are processing the histograms to classify the radio signals.

- The project Supports 4 types of classes.
    - Squiggle
    - Narrowband
    - Noise
    - Narrowbanddrd

4.2 Novelty Of the Work

- The current signal detection system is programmed for only particular kinds of signals such as narrow-band carrier waves however, the detection system sometimes triggers the signals that are not narrow-band signals with some unknown efficiency and are also not explicitly known frequency interference.

- So there seem to be various categories of these kinds of events that have been observed in the recent past.

- Thus, the goal of this project is to build an image classification model to classify these signals accurately in real-time so this may allow the signal detection system to make better observational decisions and thereby increasing the efficiency of the night scans, allowing for explicit detection of signal types.

4.3 Classes Used In the Project

The spectrograph images were converted into their raw pixel intensity values and were normalized so the values lie between 0 and 1. They are then converted into an array by stretching them. Therefore, each row of the CSV file corresponds to a single image.

- The label was found to be one hot encoded in to a vector of 1,4(no. of classes).
    - 1,0,0,0 is squiggle
    - 0,1,0,0 is Narrow-band signal
    - 0,0,1,0 is Noise
    - 0,0,0,1 is Narrow-band-drd signal

CONCLUSION

Based on the learning from this course, the Project implements a CNN network. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers. The project implements CNN to solve the problem of classifying the different radio signals obtained from the outer space. There are in total 4 classes used in this CNN network. The CNN model treats these 2-D Spectrograms as images and put them in our image classifier to classify these faint radio signals into one of the four categories.

The Project with the Help of LSTM network achieved a very accuracy of about 74% and the loss of about 0.36.

References

1 https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
2 https://www.ijariit.com/manuscripts/v6i3/V6I3-1559.pdf
3 https://en.wikipedia.org/wiki/Convolutional_neural_network
4 https://www.kaggle.com/amahbubul85/classify-radio-signals-from-space-with-keras
5 https://www.coursera.org/learn/classify-radio-signals-space-keras-cnn/ungradedLti/lQGEX/classify-signals-from-outer-space-with-keras
6 https://www.javatpoint.com/classification-algorithm-in-machine-learning
7 https://www.javatpoint.com/data-science
8 https://www.geeksforgeeks.org/basic-concept-classification-data-mining/