

**Machine Learning based Spectrum Sensing in Cognitive
Radio Networks**

by

Bollam Pravalika — Raunaq Jabbal — Saahil Sabu Hameed

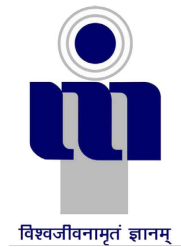
2020BCS-021 — 2020BCS-063 — 2020BCS-079

A report submitted for CAD-VLSI Project

Bachelor of Technology

in

CSE



ATUL BIHARI VAJPAYEE-

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

GWALIOR - 474015, MADHYA PRADESH, INDIA

Report Certificate

I hereby certify that the work, which is being presented in the report, entitled **Machine Learning based Spectrum Sensing in Cognitive Radio Networks**, for this Project in Computer Science Engineering and submitted to the institution is an authentic record of my own work carried out during the period *October-2022* to *November-2022* under the supervision of **Dr. Gaurav Kaushal**. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Date:

Name and Signature of the candidate

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Date:

Name and Signature of the Research Supervisors

Candidate's Declaration

I hereby certify that I have properly checked and verified all the items as prescribed in the check-list and ensure that my thesis is in the proper format as specified in the guidelines for thesis preparation.

I declare that the work containing in this report is my own work. I understand that plagiarism is defined as any one or combination of the following:

- (1) To steal and pass off (the ideas or words of another) as one's own
- (2) To use (another's production) without crediting the source
- (3) To commit literary theft
- (4) To present as new and original idea or product derived from an existing source.

I understand that plagiarism involves an intentional act by the plagiarist of using someone else's work/ideas completely/partially and claiming authorship/originality of the work/ideas. Verbatim copy as well as close resemblance to some else's work constitute plagiarism.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programmes, experiments, results, websites, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report/dissertation/thesis are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable. My faculty supervisor(s) will not be responsible for the same.

Signature:

Name:

Roll. No:

Date:

Abstract

The number of wireless communication devices has grown exponentially, and with the advent of 5G, demand for spectrum bands is increasing. We use Cognitive Radio to bridge the gap between supply and demand. Spectrum Sharing is a concept where other devices can access the spectrum band belonging to licensed users who are not using it. Cognitive Radio is a wireless communication mechanism to intelligently detect which channels are in use and reuse the channel where the PU is not transmitting. Spectrum Sensing is the critical part of Cognitive Radio, where we sense the spectrum band periodically, which allows us to sense holes in the spectrum. Secondary Users (SU) sense the spectrum for the presence of Primary Users (PU). They can use various algorithms to determine the presence or absence of primary users and can use that spectrum band when the licensed user is not transmitting. There are various factors on which the decision depends, the number of SUs, the number of PUs, the distance between SUs and PUs, noise levels, SNR values, type of fading used, sensing time, and type of technique used to determine the outcome. Techniques include Non-cooperative Spectrum Sensing, where individual SUs determine independently, and cooperative spectrum sensing (CSS), where all SUs collectively decide if the PU is active or not, because at one moment, a particular SU may report incorrectly. Under cooperative sensing, various conventional techniques like AND-based, OR-based, and Maximum Ratio Combining (MRC) based.

Machine Learning is a field of study based on building models and methods that “learn,” which use existing data to come up with a solution and increase their performance to predict outcomes or decisions accurately. Machine learning is closely related to Artificial Intelligence and Data Science. Deep Learning is a subset of Machine Learning which tries to mimic the human brain to make decisions based on given data, and this approach gives them some advantages over Machine Learning.

Dedication

We dedicate this research to our parents, who gave us unwavering support even when times were difficult and never stopped encouraging us to work even harder; our friends, who provided moral support and encouragement throughout our studies; and, last but not least, our mentor, who provided the conducive environment for this project to be completed.

Acknowledgments

The greatest thanks and appreciation are extended to **Dr.Gaurav Kaushal** for his remarkable assistance, dedication, and enduring contribution to the accomplishment of this project. We appreciate you giving us so much advice and information to finish the assignment.

Contents

Chapter		
1	Introduction	1
1.1	Context	1
1.2	Problem/Motivation	3
2	Literature review	4
2.1	Preamble	4
2.2	Background	4
2.3	Analysis	5
2.4	Problem formulation	5
3	Methodology	7
3.1	Preamble	7
3.2	Proposed hypothesis/Data Generation/Experiment Description	7
4	Dataset and Weights	10
5	Code and Screenshots	13
6	Discussions and conclusion	16
6.1	Results	16
6.1.1	Conclusion	16

Bibliography

Figures

Figure

4.1	Layer-1	11
4.2	Layer-2	12
4.3	Layer-3	12
5.1	Data Sampling	13
5.2	Import Statements	14
5.3	Creating Model	14
5.4	Training Model	15
5.5	Testing Model	15
6.1	Output	16
6.2	Simulation	17
6.3	Performance Estimation	18
6.4	Utilisation Estimation	18

Chapter 1

Introduction

1.1 Context

There has been an increase in wirelessly communicating devices because of the development of wireless technologies and protocols. This development has led to an increase in demand for radio spectrum. Cognitive Radio is a concept that allows SUs to use licensed spectrum belonging to Primary Users to overcome this demand increase. Spectrum Sensing is the part of Cognitive Radio where the Secondary Users listen to the spectrum band and use the sensing data to determine if the Primary User is transmitting. Spectrum Sensing is of two types, non-cooperative and cooperative. Non-cooperative Spectrum Sensing is where an SU independently decides if the PU is active. Cooperative Spectrum Sensing is where multiple SUs use an algorithm to decide if the PU is active. Cooperative Spectrum Sensing is more accurate than Non-cooperative Spectrum Sensing because that one SU may not sense appropriately because of the environment, so its decision may be incorrect. When multiple SUs work together, they can overcome this issue. Then we may use a variety of algorithms that may be good at spotting outliers, leading to an improvement in the decision. In Non-cooperative spectrum sensing [5], the SU senses the energy it receives, and only if it is below a certain threshold does the SU conclude that the PU is not actively using the licensed spectrum band, and it can reuse that same band. Non-cooperative spectrum sensing is of two types, classical techniques, and Machine Learning techniques.

Classical techniques [21] are based upon Cooperative Spectrum Sensing, where multiple SUs aid in deciding the final outcome, which consists of AND, OR, and Maximum Ratio Combining

techniques. In AND technique, only when all SUs conclude that the PU is transmitting, the final decision is that the PU is transmitting and the SUs should not transmit. In the OR technique, even if one SU concludes that the PU is actively transmitting, the final decision is that the PU is transmitting. In Maximum Ratio Combining, sensed energy value of each SU is multiplied by the normalised average SNR value, and the summation of these new energy values is compared against the threshold value to give the final decision. This is done so that the energy value of SUs having high SNR values are given more importance, but this requires us to know the SNR value.

Machine Learning algorithms are handy in a wide range of fields. These algorithms can automatically understand and extract patterns from data and apply this understanding to new data. We can classify our problem into two types, classification and regression. Classification is where the output type is a discrete set of values, like a Yes/No decision. Regression is where the output type is a continuous set of values.

Further, Machine Learning algorithms can be classified into Supervised Learning, Unsupervised Learning, and Reinforcement Learning. The dataset the algorithm receives is labeled with output values in Supervised Learning, and in Unsupervised Learning, there is no labeled dataset. In Reinforcement Learning, the algorithm is rewarded for doing a desirable behavior and punished for doing an undesirable behavior, and the algorithm learns through trial and error. Another crucial part of Machine Learning is Neural Networks, which try to mimic brain activity to come to a decision, are very flexible, and the algorithms can be applied to various problems. For Machine Learning approach, we do not use Non-cooperative Spectrum Sensing approach of choosing a threshold, like we do in Classical Algorithms. Instead, we pass the energy values of all SUs to the algorithms as dataset, and the algorithm works on those values to conclude if the PU is transmitting.

Spectrum Sensing is a Classification problem because we need to conclude if the PU is actively transmitting or not, so the algorithm will only give out two values, 0 and 1, representing a Yes and No. We have chosen various Supervised Learning algorithms Logistic Regression, Linear Support Vector Machine, Gaussian Support Vector Machine, K Nearest Neighbours, Random Forest Classification, Naïve Bayes, Artificial Neural Networks, and Gradient Boosting Libraries like

CatBoost, XGBoost, and ADABOOST.

1.2 Problem/Motivation

We require detailed results of the effect of various parameters on the Spectrum Sensing part of Cognitive Radio. Along with this, we need to know performance of Spectrum Sensing under various fading scenarios and algorithms.

The goal of spectrum sensing is to decide between the two hypotheses [21]:

$$z(t) = \begin{cases} n(t) & H_0 \text{ (white space)} \\ hs(t) + n(t) & H_1 \text{ (occupied)} \end{cases} \quad (1.1)$$

Where $z(t)$ is the signal sample received by the SU, $s(t)$ is the transmitted signal of the primary user, $n(t)$ is the Additive White Gaussian Noise (AWGN), and h is the complex gain of the channel.

Chapter 2

Literature review

2.1 Preamble

This chapter provides background to the Spectrum Sensing problem and the research done by various authors along with the gaps in research in this field of study. This chapter also analyzes the problem and formulates it.

2.2 Background

The requirement for higher bandwidth keeps increasing due to innovation in wireless technologies, and it is apparent that we need an intelligent and dynamic system to resolve this issue. Various campaigns have found that static spectrum access leads to overcrowding in some parts of the spectrum and underutilization in others [16]. This imbalance reduces the effective utilization of the spectrum. We require Cognitive Radio, which enables wireless devices to transmit in spectrum holes as long as the PUs are not transmitting. Cognitive Radios main objective is to use the best spectrum that the PU is not using. Cognitive Radio networks sense opportunities, characterize the environment, determine the best strategy for decisions, and adapt by changing operation parameters. The environment is dynamic, and Cognitive Radio should keep track of the changes in the environment. We mainly focus on the Spectrum Sensing part, the most crucial part of Cognitive Radio Networks. Different techniques for Spectrum Sensing are:

- Energy Detection

- Cyclostationary detection
- Matched Filter Detection

Energy Detection has been widely used as it is the most straightforward technique, it needs less sensing time, and we need no prior knowledge about the PU or its signal. The downside of Energy Detection is the poor performance we obtain when the Signal to Noise Ratio (SNR Value) is low. Cyclostationary detection exploits the second-order periodicity of the modulated signal and provides good results even at low SNR values but is complex to compute. The matched Filter technique maximizes the SNR value of the detected signal, but prior knowledge of the signal is required.

2.3 Analysis

For Cognitive Radio to work as intended, the algorithms used should come to the correct decision and should come to that decision without taking much time. If the Spectrum Sensing decision is incorrect, the SU may fail to sense a spectrum hole or utilize the spectrum when the PU is still actively transmitting, when an SU should not interfere with the PU. If Spectrum Sensing takes too long, we are wasting time that the SUs could have utilized to occupy the spectrum band.

The downside of Non-cooperative Spectrum Sensing, as the name suggests, is that the decision does not involve cooperation. The environment may lead the SU to consider the wrong decision. Another issue is to find a suitable threshold for the SU. Classical algorithms can detect more efficiently, but now we have to figure out threshold values for multiple SUs at various distances from the PU. An efficient implementation of the MRC algorithm also requires us to have an estimate of the SNR values. Machine Learning algorithms do not have the abovementioned issues.

2.4 Problem formulation

The primary goal of Spectrum Sensing is to differentiate between H_0 and H_1 (Fig. 1.1) with high accuracy.

Performance of various algorithms, fading scenarios (h), sensing samples (t), SU numbers,

training dataset sizes will be calculated, along with the most optimal values to give a comprehensive review to show how each of these parameters affect Spectrum Sensing.

Chapter 3

Methodology

3.1 Preamble

This chapter provides information on how the training and testing dataset is generated. This chapter also gives description of the various algorithms and fading scenarios that have been used for various experiments.

3.2 Proposed hypothesis/Data Generation/Experiment Description

We consider the PU to be actively transmitting 50% of the time on average ($P(H_1) = 0.5$). We consider a Cognitive Radio Network with 1 PU and N SUs, distributed evenly at a distance of 500m to 1000m. Each SU senses for time period τ which we can vary and sensing bandwidth $w = 5MHz$. Each SU senses $K = 2\tau w$ samples. Training Dataset can be changed, testing dataset is set at 50000 samples. Noise is a gaussian random variable considered to have zero mean and variance equal to the noise power. Signal is a gaussian random variable considered to have 0 mean and variance equal to the signal power. Signal Channel coefficient can be from Rayleigh Fading, Rician Fading, Nakagami Fading with the desired variance, or be absent, and is multiplied by path loss component.

$$h = g \times d^{\frac{-a}{2}} \quad (3.1)$$

(Eq. 3.1) [21] where g is the fading component of that SU, d is the distance between the SU and the PU, and $a = 4$ (path loss exponent)

After collecting K samples, the estimated normalised energy [21] of an SU is:

$$y = \sum_{k=1}^K z(k)^2 \quad (3.2)$$

Machine Learning

Machine Learning is a field of Computer Science that allows computers the ability to learn without being explicitly programmed. In traditional programming. We feed in data and logic to get the output. In Machine Learning, we feed in data and output, and the machine learns about the problem and comes up with its logic. Machine Learning has countless applications like spam detectors, web search engines, online ads, computer vision, self-driving cars, robotics, and voice assistants. Machine Learning can be of 3 types, Unsupervised Learning, Supervised Learning, and Reinforcement Learning.

- In Supervised Learning, the algorithm learns how to map the labeled data to the labels. We know exactly how many labels or the range of labels we have, and it has lots of real-world applications. These algorithms are not suitable for complex tasks, and we cannot predict accurately if the test data has some variation compared to the training data.
- Unsupervised Learning does not use a labeled dataset. It has three broad applications: Clustering, Dimensionality Reduction, and Association. Clustering is a technique where we label groups of unlabelled data into labels of our own choice. Dimensionality Reduction is a pre-processing stage that aims to simplify the number of features in data when the number of features is too much, making it easier to visualize datasets while preserving the information of the data as much as possible. Association is a method for finding relationships between variables in a dataset that has its applications in marketing where the algorithm understands patterns of customers and suggests other products or offers.
- In Reinforcement Learning, the algorithm is rewarded for doing a desirable behavior and punished for doing an undesirable behavior, and the algorithm learns through trial and error. The algorithm looks for the maximum overall reward to decide correctly.

Deep Learning is a subset of Machine Learning, which tries to mimic the behavior of the brain that allows it to learn data. Deep Learning eliminates some of the pre-processing required and can work on unstructured data like images, text, and audio. Deep Learning understands the data's information, like faces in photos and phrases in a text.

- **Multilayer Perceptron:**

Multilayer Perceptron is a feed-forward neural network because the data flows in the forward direction and consists of only three layers, input layer, hidden layer, and output layer. The input layer receives the data, and the output layer gives the decision. The neurons in the layers learn and train with backpropagation. Backpropagation aims to minimize the cost function and increase accuracy by adjusting the weights and biases which is dependent on the gradients of the cost function with respect to those parameters. After each forward pass, a backward pass is done to adjust the weights. The gradient of the loss is calculated, and is distributed layer by layer backwards.

In the forward phase (Fig. 3.10) [21], the output for a neuron j in a layer l with weights w , is

$$o_j^l = \sigma \left(\sum_i w_{ij}^l o_i^{l-1} \right) \quad (3.3)$$

Where $o_i^0 = y_i$ and $\sigma(x)$ is the logit (sigmoid) function.

Chapter 4

Dataset and Weights

Link To Dataset:

<https://github.com/raunaqjabbal/CADVLSI-Project/blob/main/ClassificationDataTest.csv>

<https://github.com/raunaqjabbal/CADVLSI-Project/blob/main/ClassificationDataTrain.csv>

Dataset Weights and Bias of Layer 1

```

Layer 1 Weights <tf.Variable 'dense/kernel:0' shape=(7, 15) dtype=float32, numpy=
array([[ 0.4937128 , -0.4085188 ,  0.19429989,  0.57683957, -0.33845448,
        -0.36928558, -0.31876397,  0.01778482, -0.32814062,  0.0279716 ,
         0.36842182, -0.5440834 ,  0.30340087, -0.26142576, -0.13649178],
       [ 0.69554204,  0.00548643, -0.27238005, -0.52251375, -0.21350516,
         0.2554306 , -0.05565824,  0.19695963, -0.124327 , -0.563365 ,
         0.36517155,  0.28101593,  0.21760708, -0.4654936 ,  0.41531616],
       [ 0.33421007, -0.00723094, -0.23615178,  0.16894203,  0.31097892,
         0.04945767, -0.01679687,  0.17042913, -0.3466342 ,  0.2474462 ,
         0.040345 ,  0.02766776,  0.16895208, -0.3968969 , -0.3971885 ],
       [ 0.43316162,  0.13052016, -0.5149493 ,  0.19248678, -0.10960756,
        -0.5569546 , -0.45502552, -0.43310723, -0.11515698,  0.2648184 ,
        -0.2999555 , -0.4098228 ,  0.23798507, -0.45489356, -0.28338012],
       [ 0.66006345, -0.20158884,  0.2336544 , -0.66282725, -0.09535955,
         0.2930437 ,  0.33253205,  0.23246491, -0.00687957, -0.02283116,
         0.22577523,  0.24161309, -0.01411874, -0.32884184, -0.30111757],
       [-0.18583935, -0.04472849, -0.26657265, -0.1945068 ,  0.34635657,
        -0.21466789,  0.31446704,  0.21018378, -0.23973376,  0.12639597,
        -0.10815256, -0.16039862,  0.20216237, -0.51345426, -0.25274518],
       [ 0.23993689, -0.36982816,  0.02870519,  0.3992858 ,  0.5155575 ,
         0.4316356 ,  0.0604824 ,  0.18580016,  0.01764345,  0.0025455 ,
        -0.01106092, -0.14220503,  0.06595555,  0.50118226,  0.05997026]],
      dtype=float32)>
Layer 1 Bias <tf.Variable 'dense/bias:0' shape=(15,) dtype=float32, numpy=
array([-0.6453954 ,  0.          , -0.02102232,  0.7022203 , -0.3601963 ,
         0.7375053 ,  0.7298605 ,  0.59264183,  0.          ,  0.64159495,
        -0.5816713 , -0.05610121, -0.5671513 ,  0.          ,  0.          ],
      dtype=float32)>

```

Figure 4.1: Layer-1

Dataset Weights and Bias of Layer 2

```

Layer 2 Weights
0      1      2      3      4      5      6      ...      23      24      25      26      27      28      29
0 -0.153899 -0.153516 0.043168 0.704044 0.268001 0.017644 0.447061 ... -0.046980 -0.039800 0.052755 0.268205 -0.339817 0.073744 0.305987
1 0.073795 -0.056075 -0.019797 0.071936 -0.029424 -0.207716 -0.174353 ... -0.185858 -0.052964 0.076641 -0.152696 0.233075 -0.064816 -0.152907
2 0.089457 0.190550 -0.116031 0.181136 -0.261882 0.317132 0.327480 ... -0.125224 -0.188664 0.033049 0.297566 0.187680 -0.240997 -0.295852
3 0.177132 -0.278006 -0.071078 -0.093298 0.092707 0.528574 -0.312627 ... -0.194851 -0.047076 -0.228598 -0.274467 0.584792 -0.100597 -0.220800
4 -0.238057 0.270238 -0.321887 1.000906 0.839394 -1.037657 1.438988 ... 0.183382 0.353077 -0.199727 0.985580 -1.260584 0.128696 1.239697
5 -0.151319 -0.169949 -0.353278 -0.346020 -0.340213 0.085508 -0.444414 ... 0.017710 -0.142735 -0.356681 -0.031799 0.278007 -0.974293 -0.500906
6 0.074703 -0.191711 0.246355 -0.685852 -0.663429 0.459926 -0.362565 ... 0.236732 0.107137 0.058485 -0.055368 0.543513 -0.246205 0.054365
7 0.182312 0.183672 -0.108943 -0.273377 0.015722 0.265034 0.224966 ... -0.364797 0.112039 -0.086893 0.195554 0.469860 0.258439 -0.313285
8 -0.191341 0.271756 0.326909 -0.347802 -0.101794 -0.182661 0.008031 ... 0.352474 -0.116688 0.094739 0.168085 -0.363223 0.189473 -0.337362
9 -0.283587 0.034354 -0.262483 -0.053090 0.009377 0.114173 -0.267412 ... 0.309029 -0.305178 -0.215803 -0.450727 0.355680 0.234532 0.030428
10 -0.215015 0.302881 -0.139455 1.062746 1.108878 -0.808859 0.741754 ... 0.011568 0.111709 -0.189831 0.838232 -0.862643 1.467790 0.952963
11 -0.348755 0.224986 0.240513 0.176233 0.257685 0.257250 -0.249154 ... -0.316856 0.009220 0.137119 -0.265389 -0.099281 0.114456 -0.012867
12 -0.215411 -0.149129 -0.281370 0.250103 0.825679 -0.121072 0.312322 ... 0.069116 -0.285410 -0.291195 0.518897 -0.393571 0.709229 0.849998
13 -0.220129 -0.244804 -0.242937 -0.013427 0.063849 0.028159 0.292312 ... -0.210805 -0.143633 -0.172751 0.334990 -0.056756 -0.222534 -0.040665
14 0.256685 0.210868 0.363699 -0.312025 0.211534 0.319200 0.267922 ... -0.085791 -0.339562 0.162832 -0.064247 0.217742 -0.221390 0.086930

[15 rows x 30 columns]
Layer 2 Bias <tf.Variable 'dense_1/bias:0' shape=(30,) dtype=float32, numpy=
array([-0.0106111, -0.01463103, 0., ..., -0.49711624, -0.45962995,
       0.4335242, -0.4974347, 0.43700832, -0.45316067, 0., ...,
       -0.45469755, 0.6765985, -0.39916402, 0.5004349, 0.4497033,
       0., ..., -0.02548496, -0.4558351, 0., ..., 0.,
       -0.5200954, 0.59351826, 0.47911203, -0.01180736, -0.0306004,
       0., ..., -0.43452132, 0.61486226, -0.43938273, -0.41656107],
      dtype=float32)>

```

Figure 4.2: Layer-2

Dataset Weights and Bias of Layer 3

```

Layer 3 Weights <tf.Variable 'dense_2/kernel:0' shape=(30, 1) dtype=float32, numpy=
array([[ -0.12163372],
       [ -0.15305161],
       [ 0.37744516],
       [ 0.77281174 ],
       [ 0.8979196 ],
       [ -0.5974961 ],
       [ 1.1457374 ],
       [ -0.41350386],
       [ 1.5610347 ],
       [ -0.3949466 ],
       [ 1.1205146 ],
       [ -0.92731965],
       [ 1.115165 ],
       [ -1.0661473 ],
       [ -0.551234 ],
       [ 0.2972374 ],
       [ -0.26113948],
       [ 0.93901163],
       [ -0.2680856 ],
       [ -0.315714 ],
       [ 1.093429 ],
       [ -0.8008344 ],
       [ -0.23746207],
       [ 0.36296487],
       [ -0.00699021],
       [ -0.43006897],
       [ 0.9830004 ],
       [ -1.0471593 ],
       [ 2.3958437 ],
       [ 1.7501211 ]], dtype=float32)>
Layer 3 Bias <tf.Variable 'dense_2/bias:0' shape=(1,) dtype=float32, numpy=array([-0.38550764], dtype=float32)>

```

Figure 4.3: Layer-3

Chapter 5

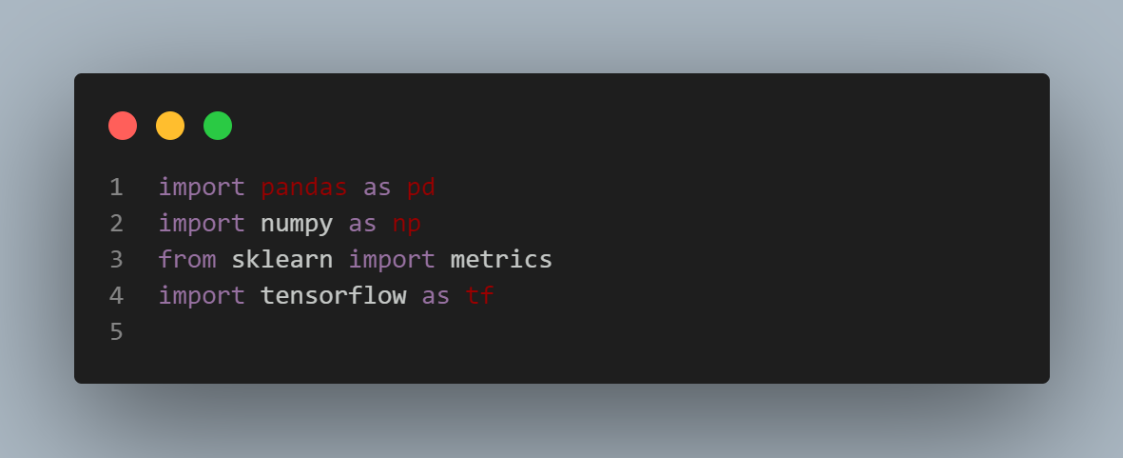
Code and Screenshots

Dataset Used: Data is generated as per mentioned hypothesis



Figure 5.1: Data Sampling

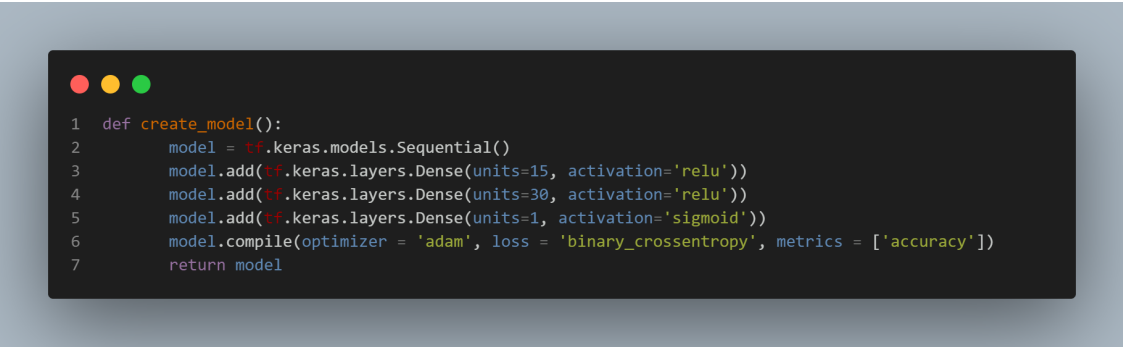
Import Statements for all libraries imported



```
1 import pandas as pd
2 import numpy as np
3 from sklearn import metrics
4 import tensorflow as tf
5
```

Figure 5.2: Import Statements

Dense layer is a layer that is deeply connected with its preceding layer which means the neuron of the layer are connected to every neuron of its preceding layer



```
1 def create_model():
2     model = tf.keras.models.Sequential()
3     model.add(tf.keras.layers.Dense(units=15, activation='relu'))
4     model.add(tf.keras.layers.Dense(units=30, activation='relu'))
5     model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
6     model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
7     return model
```

Figure 5.3: Creating Model

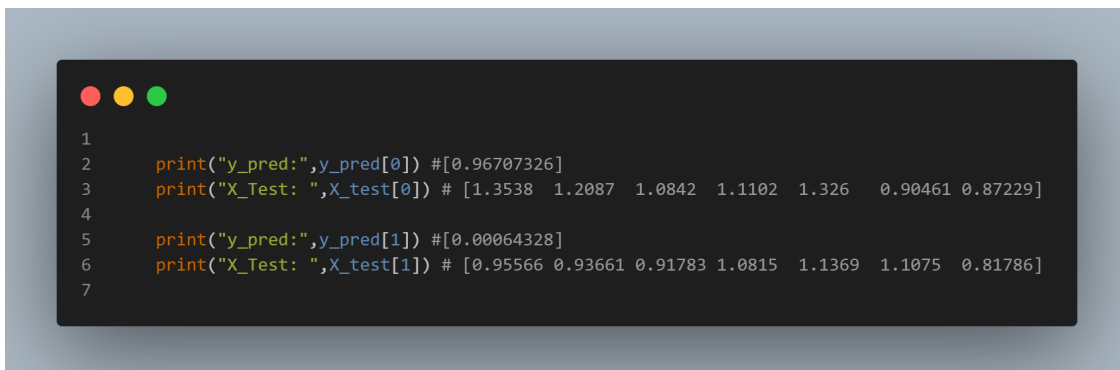
Weights are saved file in using `model.save_weights`.The tuned weights are used later to predict accuracy and output for test dataset



```
1
2 model = create_model()
3 model.load_weights('./checkpoints/my_checkpoint')
4 y_pred=model.predict(X_test)
5 y_pred2=np.array(y_pred>0.5,dtype="int")
6 y_pred2=y_pred2.flatten()
7
```

Figure 5.4: Training Model

Outputs of the test dataset are shown here.



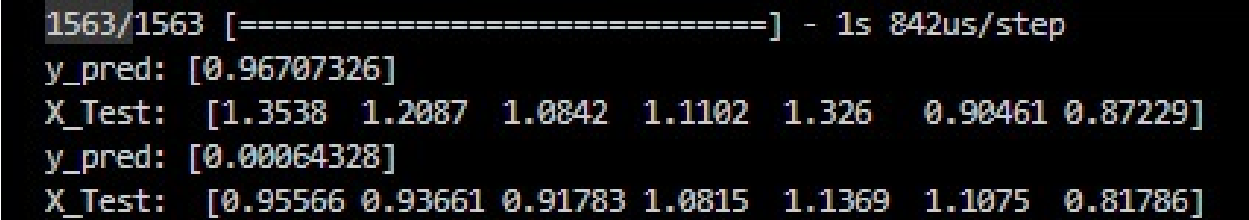
```
1
2 print("y_pred:",y_pred[0]) #[0.96707326]
3 print("X_Test: ",X_test[0]) # [1.3538  1.2087  1.0842  1.1102  1.326   0.90461 0.87229]
4
5 print("y_pred:",y_pred[1]) #[0.00064328]
6 print("X_Test: ",X_test[1]) # [0.95566 0.93661 0.91783 1.0815  1.1369  1.1075  0.81786]
7
```

Figure 5.5: Testing Model

Chapter 6

Discussions and conclusion

6.1 Results

A terminal window with a black background and white text. The text shows the progress of a model training process. It starts with '1563/1563' followed by a progress bar of 20 equals signs, then '- 1s 842us/step'. Below this, it shows 'y_pred: [0.96707326]', 'X_Test: [1.3538 1.2087 1.0842 1.1102 1.326 0.90461 0.87229]', 'y_pred: [0.00064328]', and 'X_Test: [0.95566 0.93661 0.91783 1.0815 1.1369 1.1075 0.81786]'.

```
1563/1563 [=====] - 1s 842us/step
y_pred: [0.96707326]
X_Test: [1.3538 1.2087 1.0842 1.1102 1.326 0.90461 0.87229]
y_pred: [0.00064328]
X_Test: [0.95566 0.93661 0.91783 1.0815 1.1369 1.1075 0.81786]
```

Figure 6.1: Output

6.1.1 Conclusion

Our model worked fair enough for a multiplayer perceptron with 3 layers and 500 epochs giving 93.89% accuracy

```

hand_num_nn_csim.log
1[INFO: [SIM 2] ***** CSIM start *****
2INFO: [SIM 4] CSIM will launch GCC as the compiler.
3make: `csim.exe' is up to date.
4number recognized: 0.947279
5INFO: [SIM 1] CSim done with 0 errors.
6INFO: [SIM 3] ***** CSIM finish *****
7

Vivado HLS Console
Starting C simulation ...
E:/Vivado/2018.1/bin/vivado_hls.bat C:/Users/Raunaq/Desktop/CADVLSI/CADVLSI/solution1/csim.tcl
INFO: [HLS 200-10] Running 'E:/Vivado/2018.1/bin/unwrapped/win64.o/vivado_hls.exe'
INFO: [HLS 200-10] For user 'R' on host 'raunaq' (Windows NT_amd64 version 6.2) on Thu Nov 03 15:08:12 +0530 2022
INFO: [HLS 200-10] In directory 'C:/Users/Raunaq/Desktop/CADVLSI'
INFO: [HLS 200-10] Opening project 'C:/Users/Raunaq/Desktop/CADVLSI/CADVLSI'.
INFO: [HLS 200-10] Opening solution 'C:/Users/Raunaq/Desktop/CADVLSI/CADVLSI/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xa7a12tcsg325-1q'
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
make: `csim.exe' is up to date.
number recognized: 0.947279INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
Finished C simulation.

```

Figure 6.2: Simulation

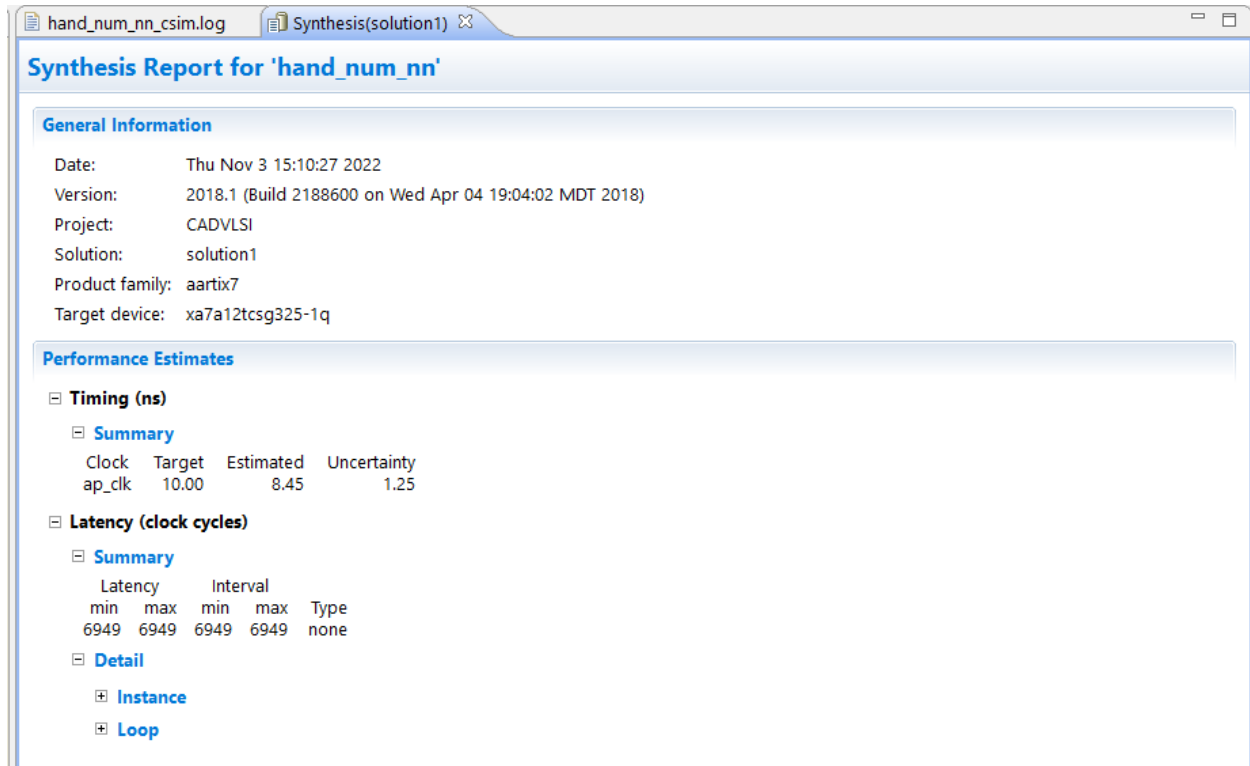


Figure 6.3: Performance Estimation

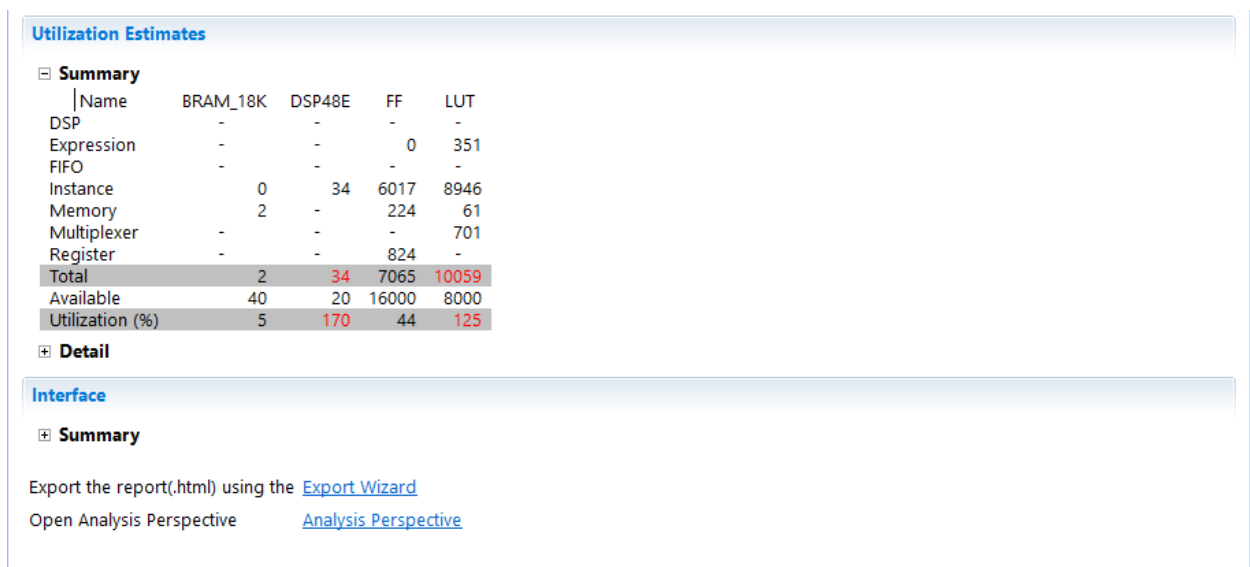


Figure 6.4: Utilisation Estimation

Bibliography

- [1] M. A. ABUSUBIAH AND S. KHAMAYSEH, Performance of machine learning-based techniques for spectrum sensing in mobile cognitive radio networks, IEEE Access, 10 (2022), pp. 1410–1418.
- [2] O. AWE AND S. LAMBOTHARAN, Cooperative spectrum sensing in cognitive radio networks using multi-class support vector machine algorithms, IEEE Xplore.
- [3] M. AYGUL, M. NAZZAL, AND H. ARSLAN, Deep rl-based spectrum occupancy prediction exploiting time and frequency correlations, IEEE Wireless Communications and Networking Conference (WCNC), (2022), pp. 2399–2494.
- [4] S. K. D. JANU, K. SINGH, Machine learning for cooperative spectrum sensing and sharing: a survey, John Wiley and Sons, (2021), pp. 1–28.
- [5] C. GATTOUA, O. CHAKKOR, AND F. AYTOUNA, An overview of cooperative spectrum sensing based on machine learning techniques, IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), (2020).
- [6] A. GHASEMI AND E. SOUSA, Collaborative spectrum sensing for opportunistic access in fading environments, IEEE Xplore, (2008), pp. 131–136.
- [7] N. GUL, S. KIM, S. AHMED, M. KHAN, AND J. KIM, Differential evolution based machine learning scheme for secure cooperative spectrum sensing system, Electronics, 14 (2021), pp. 1081–1089.
- [8] N. KHALEK AND W. HAMOUD, Learning-based cooperative spectrum sensing in hybrid underlay-interweave secondary networks, Globecom 2020 - 2020 IEEE Global Communications Conference.
- [9] S. KHAMAYSEH AND A. HALAWANI, Cooperative spectrum sensing in cognitive radio networks: A survey on machine learning-based methods, Journal of Telecommunications and Information Technology, (2020), pp. 36–46.
- [10] V. KRISHNAKUMAR, P. SAVARINATHAN, T. KARUPPASAMY, AND A. JAYAPALAN, Machine learning based spectrum sensing and distribution in a cognitive radio network, International Conference on Computer Communication and Informatics (ICCCI), (2022).
- [11] W. KUMAR, D. KANDPAL, AND M. JAIN, K-mean clustering based cooperative spectrum sensing in generalized k-mu fading channels, IEEE Xplore, (2016).

- [12] Z. LI, W. WU, X. LIU, AND P. QI, Improved cooperative spectrum sensing model based on machine learning for cognitive radio networks, *IEEE Commun.*, 12 (2018), pp. 2485–2492.
- [13] Y. C. LIANG, Y. ZENG, E. PEH, AND A. HOANG, Sensing-throughput tradeoff for cognitive radio networks, 1326 *IEEE Transactions On Wireless Communications*, 7 (2008), pp. 1326–1337.
- [14] Y. LU, P. ZHU, D. WANG, AND M. FATTOUCHE, Machine learning techniques with probability vector for cooperative spectrum sensing in cognitive radio networks, *IEEE Wireless Conference and Networking Conference (WCNC 2016) Track 1: PHY and Fundamentals*.
- [15] A. MIKAEIL, B. GUO, AND Z. WANG, Machine learning to data fusion approach for cooperative spectrum sensing, , 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 429–434.
- [16] K. PATIL, R. PRASAD, AND K. SKOUBY, A survey of worldwide spectrum occupancy measurement campaigns for cognitive radio, *Proceeding of the International Conference on Devices and Communications (ICDeCom)*, (2011), pp. 1–5.
- [17] M. SABERA, A. RHARRASA, R. SAADANEA, A. CHEHRI, N. HAKEM, AND H. KHARRAZ, Spectrum sensing for smart embedded devices in cognitive networks using machine learning algorithms, *Elsevier*, 176 (2020), pp. 2404–2413.
- [18] A. SAHAI, S. MISHRA, AND R. TANDRA, Spectrum sensing: Fundamental limits.
- [19] R. SARIKHANI AND F. KEYNIA, Cooperative spectrum sensing meets machine learning: Deep reinforcement learning approach, *IEEE Communications Letters*, 23 (2020), pp. 1459–1462.
- [20] H. SHAH AND I. KOO, Reliable machine learning based spectrum sensing in cognitive radio networks, *Hindawi*, (2018), pp. 1–17.
- [21] C. TAVARES, J. MARINELLO, M. P. JR., AND T. ABRAO, Machine learning-base models for spectrum sensing in cooperative radio networks, *IET Commun.*, 14 (2020), pp. 3102–3109.
- [22] Y. TENORIO, A. GUERRERO, R. GONZALEZ, AND S. BOQUE, Machine learning techniques applied to multiband spectrum sensing in cognitive radios, *Sensors*, (2019), pp. 1–22.
- [23] K. THILINA, K. CHOI, N. SAQUIB, AND E. HOSSAIN, Machine learning techniques for cooperative spectrum sensing in cognitive radio networks, *IEEE Journal On Selected Areas in Communications* *IEEE*, 31 (2013), pp. 2209–2221.
- [24] J. WANG AND B. LIU, A brief review of machine learning algorithms for cooperative spectrum sensing, *Journal of Physics: Conference Series*, (2021), pp. 1–5.
- [25] M. WASILEWSKA, A. KLIKS, H. BOGUCKA, K. CICHON, J. RUSECKAS, G. MOLIS, A. MACKUTE-VARONECKIENE, AND T. KRILAVACIUS, Artificial intelligence for radio communication context-awareness, *IEEE Access*, 9 (2021), pp. 144820–144856.
- [26] W. WU, Z. LI, S. MA, AND J. SHI, Performance improvement for machine learning-based cooperative spectrum sensing by feature vector selection, *IET Commun.*, 14 (2020), pp. 1081–1089.

- [27] Y. XU, P. CHENG, Z. CHEN, Y. LI, AND B. VUCETIC, Mobile collaborative spectrum sensing for heterogeneous networks: A bayesian machine learning approach, *IEEE Transactions on Signal Processing*, 66 (2018), pp. 5634–5647.
- [28] R. YAKKATI, R. YAKKATI, R. TRIPATHY, AND L. CENKERAMADDI, Radio frequency spectrum sensing by automatic modulation classification in cognitive radio system using multiscale deep cnn, 926 *IEEE Sensors Journal*, 22 (2022), pp. 926–938.
- [29] K. ZHANG, J. LI, AND F. GAO, Machine learning techniques for spectrum sensing when primary user has multiple transmit powers, *IEEE Xplore*, (2014), pp. 137–141.