NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COGNITIVE RADIO CLUSTERING ALGORITHM FOR SWARMS USING NEURAL NETWORKS

by

Uriel D. Frydman

September 2022

Thesis Advisor:	Preetha Thulasiraman
Second Reader:	Murali Tummala

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE COGNITIVE RADIO CLUSTERING ALGORITHM FOR SWARMS USING NEURAL NETWORKS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Uriel D. Frydman	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Spectral scarcity is a problem faced by many communications systems, in and outside the military. A cognitive radio network is an approach that opportunistically exploits the broadcasting spectrum. The basic concept includes classifying users into two types: primary and secondary. The primary users have priority in the resource allocation process, while the secondary users need to use the spectrum for communication. This thesis seeks to apply the cognitive radio concept to enable swarm communication in a high-traffic environment. Primary users may include prioritized friendly or adversary transmitters that cannot be controlled. This research employs cognitive radio concepts and machine learning algorithms to develop a dynamic clustering technique within the network that will optimize resource allocation. Three approaches are proposed to train a neural network to find an optimal spectrum allocation. Even though the proposed algorithm did not outperform the baseline heuristic, the existence of an optimal solution was shown to exist. It is recommended that this study be continued as the algorithms used can be further modified and applied in various ways.

14. SUBJECT TERMS cognitive radio, machine learning, UAV, swarm, communications, spectrum allocation, clustering	**15. NUMBER OF PAGES** 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

COGNITIVE RADIO CLUSTERING ALGORITHM FOR SWARMS USING NEURAL NETWORKS

Uriel D. Frydman
Seren, Israel Defence Forces
BS, The Hebrew University of Jerusalem, 2018

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ELECTRICAL ENGINEERING)**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2022**

Approved by: Preetha Thulasiraman
Advisor

Murali Tummala
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Spectral scarcity is a problem faced by many communications systems, in and outside the military. A cognitive radio network is an approach that opportunistically exploits the broadcasting spectrum. The basic concept includes classifying users into two types: primary and secondary. The primary users have priority in the resource allocation process, while the secondary users need to use the spectrum for communication. This thesis seeks to apply the cognitive radio concept to enable swarm communication in a high-traffic environment. Primary users may include prioritized friendly or adversary transmitters that cannot be controlled. This research employs cognitive radio concepts and machine learning algorithms to develop a dynamic clustering technique within the network that will optimize resource allocation. Three approaches are proposed to train a neural network to find an optimal spectrum allocation. Even though the proposed algorithm did not outperform the baseline heuristic, the existence of an optimal solution was shown to exist. It is recommended that this study be continued as the algorithms used can be further modified and applied in various ways.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ADAM	Adaptive Momentum Estimation
AI	Artificial Intelligence
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CC	Communication Cell
CDF	Cumulative Distribution Function
CR	Cognitive Radio
CRN	Cognitive Radio Network
CSK	Code Shift Keying
DA	Data Analytics
EIRP	Effective Isotropic Radiated Power
FAR	False Alarm Rate
FC	Fully Connected
FCNN	Fully-Connected Neural Network
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FSPL	Free Space Path Loss
GB	Guard Band

GPS	Global Positioning System
HMSE	Half Mean Squared Error
kNN	k Nearest Neighbors
LOS	Line of Sight
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
PD	Probability of Detection
PDF	Probability Density Function
PSD	Power Spectral Density
PSK	Phase Shift Keying
PU	Primary User
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
ReLU	Rectified Linear Unit
RF	Radio Frequency
RMSE	Root Mean Squared Error
SAC	Sense-Allocate-Communicate
SER	Symbol Error Rate

SNR Signal-to-Noise Ratio

SSE Sum of Squared Errors

TDMA Time Division Multiple Access

TDOA Time Difference Of Arrival

UAV Unmanned Aerial Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

In 1894, the first wireless telegraph was invented by Guglielmo Marconi, an Italian engineer and one of two physics Nobel laureates of 1909. He is also considered to be the inventor of the radio. Since Marconi's invention in the late 19th century, the ever-growing need for bandwidth has made the spectrum more crowded by the decade. Modern telecommunication standards employ various spectrum sharing techniques, such as Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA). Their purpose is to allow many users to utilize the spectrum without interference. In military communications, this problem is aggravated since, in addition to multiple friendly and civilian users, adversaries also use the spectrum, often intending to disrupt adversary communications.

New solutions are needed to fit users into a finite bandwidth as the spectrum becomes more crowded. One such solution is Cognitive Radio (CR), a concept first introduced in 1999 by Joseph Mitola [1]. The CR approach divides users into two categories, Primary Users (PUs) and secondary users. The PUs are users who get priority in spectrum allocation and cannot be interfered with when accessing their channels. Secondary users access the spectrum opportunistically and must yield the bandwidth to the PUs when needed. To do this, the secondary users must sense the spectrum periodically to ensure that the channels used are unoccupied. If a PU is detected on a channel, then that channel cannot be used by the secondary user.

One of the principal trade-offs in a Cognitive Radio Network (CRN) is global optimization versus local optimization. Global synchronization is an approach where a central decision node allocates bandwidth for all the network users simultaneously. Since it demands concentrating all the information in a single processing location, it requires more overhead communications. On the contrary, local optimization and resource allocation are cheaper in overhead communications. However, it fails to find the optimal working configuration. This thesis considers a swarm Unmanned Aerial Vehicle (UAV) network that operates in a spectrally crowded environment. It aims to develop an algorithm that balances the local and global optimizations to maximize the network throughput. The complex nature of this

problem and the tangled relationship between many variables involved motivated the use of Neural Networks (NNs) to solve this problem.

1.1 Related Work

Many books and papers have been written on CR [2]–[6], but none have been found to outline a method to search for an optimal working point that balances the local and global synchronizations. Additionally, very few pieces of literature have presented work that utilizes Machine Learning (ML) methods as part of the CR algorithm [7]–[9].

In [7], the authors suggest a dynamic TDMA protocol for UAV swarms. The protocol consists of UAVs asking for time slots from a central ground control station that synchronizes between the network nodes and allocates time slots in a way that maximizes the network efficiency while maintaining the Quality of Service (QoS). This is an example of a centralized architecture that performs global optimization. An example of local optimization can be found in [8] where the authors perform optimization of a single UAV sensing period. One of the few publications that present an approach that combines ML and CR is [9]. The authors present a genetic algorithm-based approach to combat jamming via a CRN.

The study presented in this thesis aims to explore the use of NNs to find an optimal working point for a CRN in the local-global trade-off space. This tool has not been widely used for CR-related work to solve a complex optimization problem.

1.2 Thesis Contributions

Spectral scarcity is a problem many entities face, both inside and outside the military. The objective of this thesis is to train an NN to find an optimal clustering of the network nodes that will maximize its throughput, based on the present situation and all known parameters of PUs . First, a simulation infrastructure was developed to create data later used for NN training. This simulation and dataset might be useful to other researchers who explore this subject. Additionally, three approaches for training NNs were tested across many different architectures. The different approaches applied aim to deal with the non-linearities encountered in a CRN and the limitations of shallow NNs when trying to train them. For each approach, performance was evaluated and compared with a baseline heuristic. Lastly, the

datasets created for this study show that an optimum between global and local optimization exists and the difference in throughput between the two is quantified.

1.3 Thesis Organization

The organization of the thesis is as follows: Chapter 2 provides background on communications theory and ML, discussing both K-means and NNs. In Chapter 3, the simulation used to create data to train the network is discussed in detail. Chapter 4 presents the NN training methods and configurations. In Chapter 5, the results of the NN training are presented for three different training approaches. Lastly, Chapter 6 provides conclusions and recommendations for potential future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

This chapter will review communications and ML topics that are important to understand before diving into the methodology. First, path loss will be discussed, followed by a section on modulations and bit error probability, P_b, calculations. After that, two ML algorithms, K-means++ and NNs, will be introduced. This chapter aims to give the reader concise yet sufficient background knowledge to understand the upcoming chapters.

2.1 Loss Model

It is a well-known fact that power is lost between transmitting and receiving antennas. This phenomenon can be encountered when losing a radio signal while driving a car in rural areas or losing reception on the cellphone wireless connection when moving away from the home router. The energy loss occurs for different reasons, such as the energy being transmitted in directions other than the receiving antenna direction, attenuation due to obstacles, destructive interference due to reflections (multipath fading), and more. Many studies have been conducted to quantify these losses in different environments [10]–[12]. These studies yielded many loss models that describe power loss as a function of various parameters under different circumstances.

2.1.1 Free Space Path Loss

The simplest model that can be considered is the Free Space Path Loss (FSPL) model, which assumes no obstacles or surfaces are present. In this case, the only power loss in the system is due to the directivity of the antennas, i.e., the antennas transmit and receive power from directions other than the direction of the other antenna. According to this model, the received power in the receiving antenna is

$$P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi r}\right)^2 \tag{2.1}$$

where P_t and P_r are the transmitted and received powers, respectively, G_t and G_r are the transmitting and receiving gains. The origin of this equation as well as a detailed derivation and an explanation of the λ term can be found in [13]. Because of the high altitude of the UAVs over the surface and the lack of obstacles in this study, this model was chosen. A more detailed justification is provided in Section 3.5.1.

2.1.2 Other Loss Models

It is important to note that other path loss models do exist [11]. The FSPL model is the best-case model since it assumes maximum power reception and no multipath effects. Operating in an obstructed environment presents new challenges associated with interference and attenuation. This thesis addresses only the FSPL case. Future work might take into consideration such non-ideal environments where the path loss presents a greater challenge.

Other models consider a faster decaying with distance, reflections, atmospheric attenuation, and more. Some of the models are empirical, others depend on simulation software, and some of them incorporate the use of statistics and random variables. In [14], a comprehensive review of Radio Frequency (RF) propagation principles and models can be found.

2.2 Modulation and Bit Error Probability

When transmitting information between two devices by electromagnetic waves, the information must be incorporated into the RF signal. The methods used to add the information to the signal are called modulations. In digital communications, all the information is represented by bits, information segments that can only obtain the values 0 and 1. Digital communication is often preferred over analog transmission (which does not quantize the information) since it allows corrections of corrupted information based on the set of finite values the information can obtain. This quantization also enables a definition of a new metric, the Bit Error Rate (BER), which measures the ratio between the number of erroneous and sent bits. The better the received signal quality, the lower the error probability.

2.2.1 Quadrature Amplitude Modulation and Phase Shift Keying

There are many modulation schemes used for communications nowadays. The choice of modulation depends on the application, requirements, and constraints imposed on the sys-

tem. Two very popular modulations are M-ary Quadrature Amplitude Modulation (MQAM) and M-ary Phase Shift Keying (MPSK). The simulation presented in Chapter 3 uses both modulations.

In MPSK, the information is encoded into the phase between the In-phase and Quadrature components. MQAM encodes the information in both the phase between these components and their amplitude. Often, instead of considering the complete signal (usually referred to as the RF signal), the complex envelope is considered for analysis. Representing the signals as the 2D vector equivalents of the complex-valued envelope and normalizing the scale to have unit energy, the symbols for MPSK and MQAM can be written (respectively) as

$$\mathbf{s}_n = \frac{A}{\sqrt{2}} \left(\cos\left[(2n-1)\frac{\pi}{M} \right], \sin\left[(2n-1)\frac{\pi}{M} \right] \right) \tag{2.2}$$

$$\mathbf{s}_n = \sqrt{E_s}\,(A_n \cos(\theta_n), A_n \sin(\theta_n)) \tag{2.3}$$

where E_s is the energy contained in the symbol transmitted. Figure 2.1 shows the symbol space for 16QAM and 8PSK. A detailed explanation and derivation of these modulations schemes can be found in [15].

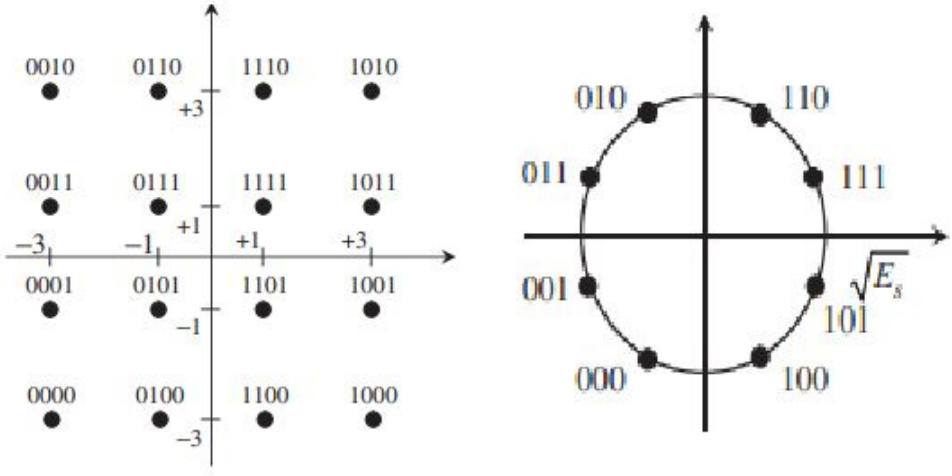

Figure 2.1. Left: 16QAM constellation with gray coding. Right: 8PSK constellation with gray coding. Adapted from [15].

There is a similar relation between the bitrate and the bandwidth for these two modulations. The bitrate and symbol time are related by

$$R_b = kR_s = \frac{k}{T_s} \tag{2.4}$$

As explained in [15], the Power Spectral Density (PSD) is related to the pulse shape by a Fourier-Transform. For this reason, the square pulse of duration T_s yields a sinc-shaped PSD with a null-to-null bandwidth of $B_{nn} = 2R_s = \frac{2R_b}{k}$.

Finally, a note on the modulation choice. Both modulations presented in this section exhibit spectral efficiency. These schemes preserve the bandwidth as the value of M increases at the expense of having a higher error rate. On the contrary, other modulations which belong to the power-efficient category, such as M-ary Code Shift Keying (MCSK), grow in bandwidth when M increases while keeping the error rate from growing exponentially (as it does with MQAM, for example). Since the communications in this study are bandwidth-limited and not power-limited (as all links are Line of Sight (LOS)), spectrally efficient modulations were chosen.

2.2.2 Bit Error Probability

When measuring the error rate of a communications system, usually, the figure of merit used is the bit error probability, P_b, and it often is presented as a function of $\frac{E_b}{N_0}$. This is the ratio between the average bit energy and the noise spectral density. The relation between $\frac{E_b}{N_0}$ and the Signal-to-Noise Ratio (SNR) is

$$\frac{S}{N} = \frac{E_b R_b}{N_0 B_{eq}} \tag{2.5}$$

where B_{eq} is the noise equivalent bandwidth.

Under ordinary circumstances, the noise in a coherently detecting receiver (needed to demodulate MQAM and MPSK) is Gaussian. Hence, the Gaussian distribution can be used to describe the decision variable. The decision variable is ultimately the variable used to decide which symbol was sent. Due to the exponentially decaying nature of Gaussian distributions, under the assumption that the received SNR is reasonable (which, empirically,

is a good assumption in this study), the only errors with non-negligible probability are errors in which a symbol, s_i, was sent, and the demodulated symbol is one of s_i nearest neighbors. Since the tail probability is very small, the likelihood that the Gaussian noise had caused an error by moving the symbol closer to a different symbol further away is negligible. Section A.1 of the Appendix derives an upper limit on the Symbol Error Rate (SER) using the union bound concept. This bound becomes tight enough for reasonable SNR. Applying the union bound to MQAM and MPSK, one arrives at the expressions shown in [15] for P_b coherent MPSK

$$P_b = \frac{2}{\log_2 M} Q\left(\sqrt{\frac{2E_b}{N_0} \log_2 M} \sin \frac{\pi}{M}\right) \tag{2.6}$$

and coherent MQAM

$$P_b = \frac{4\left(1 - \frac{1}{\sqrt{M}}\right)}{\log_2 M} Q\left(\sqrt{\frac{3 \log_2 M}{M-1} \frac{E_b}{N_0}}\right) \tag{2.7}$$

2.3 K-means

In ML, clustering often plays a major role in data analysis. There are many clustering algorithms, as well as cluster types. Most of these algorithms try to minimize the intra-cluster distance while maximizing the inter-cluster distance. Additionally, the output of the clustering algorithm may depend not only on the inputs but also on many tunable parameters. For example, the number of clusters and the definition of distance between data points may affect the resulting clusters. This section will present one of the most popular clustering algorithms, K-means, and its version K-means++, which is used in this thesis.

2.3.1 General Algorithm

The K-means algorithm is an iterative algorithm that receives, as inputs, the dataset, the number of clusters, a metric definition, and initial centroids. For each data point, the distance to each centroid is calculated, and the data point is labeled according to the centroid it was closest to, becoming part of the cluster around it. Then, new centroids are calculated from the existing clusters by averaging the data points in each cluster. This centroid calculation

is followed by another distance calculation between the new centroids and the data points, resulting in a new clustering. This process of finding centroids and reclustering continues repeatedly until the centroids stop changing. Figure 2.2 shows the K-means algorithm over many iterations.

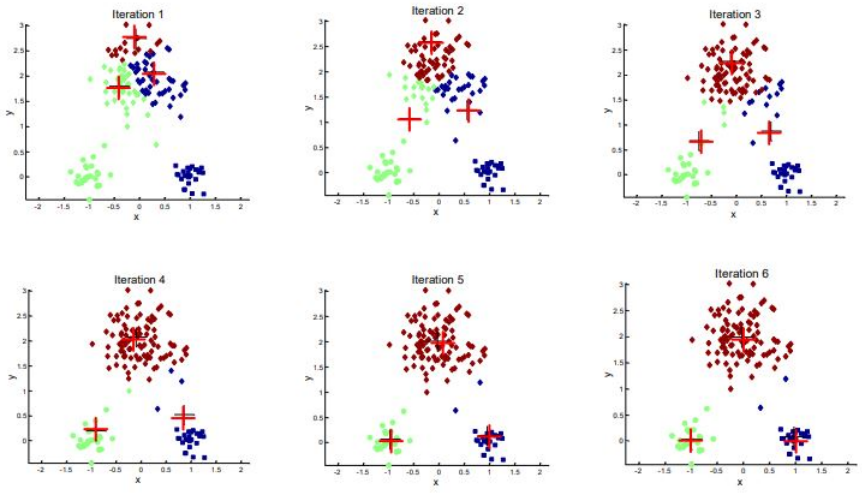

Figure 2.2. K-means algorithm with three clusters for six iterations. The red crosses represent the centroids. Source: [16].

Often, the quality of the clustering is measured by its Sum of Squared Errors (SSE), given by

$$E_{SE} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \left(d\left[C_i, x_j\right]\right)^2 \tag{2.8}$$

where N is the number of clusters, M_i is the number of data points associated with each cluster, C_i is the i^{th} centroid, x_j is a data point in the cluster, and d is the distance function used.

2.3.2 K-means++

The K-means++ algorithm adds to the original algorithm an initial step meant to choose the initial centroids in a sensible manner rather than at random. According to [17], the K-

means++ algorithm tends to achieve better SSE than the original one. The centroid selection is as follows:

1. Select at random one of the data points to be the first centroid
2. Calculate $d(C_1, x_j) = d_{1,j}$ for all the data points left
3. Choose the next centroid with a weighted probability proportional to $d^2_{1,j}$
4. Calculate the distance from the remaining data points to each of the centroids
5. Choose the next centroid with a weighted probability proportional to $d^2_{1,j} + d^2_{2,j} + ...$

Steps 4 and 5 are repeated until K centroids have been chosen. After choosing K centroids, the K-means++ algorithm proceeds as the original one does.

As can be noted in both algorithms, none of them addresses the problem of choosing the number of clusters. It is considered to be one of the inputs to the algorithm. There are many techniques to select the number of centroids, for example, searching by brute force for a substantial dip in the SSE as a function of K, followed by a gentle decline in SSE. In this case, the K value of the drop would be chosen as the number of clusters. In this study, selecting the correct number of clusters is at the center of the study interest.

2.4 Neural Networks

NNs are powerful tools used for classification and regression problems. The basic NN structure consists of nodes, called neurons, which are interconnected between themselves. Each neuron has inputs and outputs, which are used as the inputs for the neurons it is connected to. Each neuron performs a mathematical action on the inputs before pushing the output into an activation function, which, in turn, sends its output as the input for the next layer. Figure 2.3 shows an Fully-Connected Neural Network (FCNN) with one hidden layer. This NN is used for classification. Each of the ten output neurons assigns a score to a class, and the class with the highest score is selected as the predicted class.

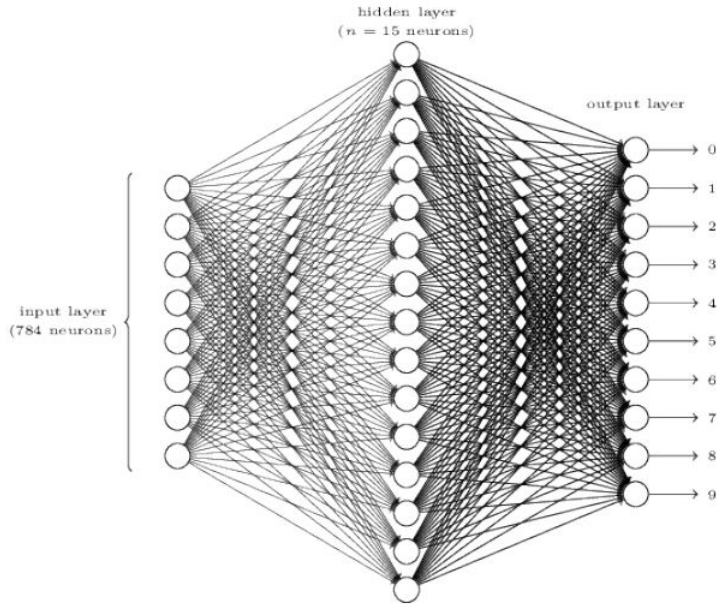

Figure 2.3. FCNN with one hidden layer, ten outputs, one for each class. Source: [18].

An example of the use of NNs could be the classification of images. If a dataset contains 10,000 images of ten types of animals, then an NN could be trained to predict which animal appears in each image. In addition to Fully Connected (FC) networks, other networks exist for application-specific use. For example, Long-Short Term Memory (LSTM) networks are used to predict the next step in a time-series problem (such as predicting full stops auto-generating subtitles [19]).

2.4.1 Neurons

The neuron is the basic building block of the NN. The most basic type of neuron is the perceptron, which receives a vector of numbers and outputs a zero or one based on a calculation involving the input. A comprehensive discussion on the neuron can be found in [18]. The perceptron is a very simple neuron with limited capability. For example, it cannot solve any nonlinearly separable problem, such as the XOR gate.

2.4.2 Activation Functions

Activation functions, or layers, are layers added between neuron layers to manipulate the outputs of the preceding layer.

In [18], the author gives a few examples of commonly used activation functions. In this study, the most frequently used activation function will be the Rectified Linear Unit (ReLU). For reasons not discussed in this thesis, the ReLU activation layer became very popular in recent years. The activation function is described by

$$\sigma(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases} \tag{2.9}$$

2.4.3 Loss

The loss function is a function that measures the error between the prediction and the desired result. Loss functions are used both for classification and regression problems. The exact loss functions used in this study are described in Chapter 4. In [18],in addition to giving examples of popular loss functions, the author explains in detail the role the loss plays in training the network.

Most NNs are trained by using different versions of gradient descent to minimize the loss by tuning the weights. A properly tuned learning rate should avoid overshoots of the minima while keeping the training duration reasonable (a small learning rate may not overshoot, but it will make the training extraordinarily long). An obvious shortcoming of the standard gradient descent algorithm is convergence to a local minimum without being able to identify it.

Some NNs use some version of *momentum*. The momentum value describes the recent trend of the solver and, much like a ball rolling downhill, allows the solver to overcome minor "uphill" tendencies if they are indeed minor and followed by a descent. Much like in real life, this feature allows the solver to leave local minima in favor of other, more significant minima and, hopefully, the global minimum. One of these algorithms, particularly the one used in this study, is the Adaptive Momentum Estimation (ADAM) optimizer. Without

getting into the details and reasoning of ADAM, it has the following weight updating rules

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1) \nabla L \tag{2.10}$$

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2) (\nabla L)^2 \tag{2.11}$$

$$\vec{w}_{l+1} = \vec{w}_l - \frac{\alpha m_l}{\sqrt{v_l}\epsilon} \tag{2.12}$$

where the hyperparameter ϵ usually takes on small values, such as 10^{-8}.

2.4.4 Backpropagation

Calculating the gradient of the loss with respect to the weights of all the neurons in the network is not a simple task. For the sake of brevity, backpropagation is not discussed here in detail, a more detailed explanation can be found in [18]. As explained above, each layer receives the output of the previous layer as input, which means the relationship between the loss and the weights of the first layers might be expressed as a composite function of many different functions (representing the layers in between). This structure motivates the use of the chain rule. To calculate all the partial derivatives, the chain rule is applied to each neuron, starting from the loss layers back to the first layer in the network, hence the name, backpropagation. Each weight is updated based on the desired change in the current layer output, which is the next one input.

Often, the weights are not updated after every data point. This is very time-consuming. Instead, the average gradient across many samples is calculated, and the weights are updated based on the combined gradient. The set of samples used to calculate the average gradient is called "mini-batch," and the size of the mini-batch is one of the tunable parameters. Additionally, finding the average gradient across many samples will probably yield a value closer to the desired gradient, while using a single sample might be highly affected by features specific to that sample.

2.4.5 Overfitting

When training an NN, it is important to understand the limitations of the network and the dataset one is working with. If the network trains with the same dataset for many iterations,

it might end up learning not only the desired task but also features specific to the training data used. For example, suppose the NN is given the task of performing linear regression, i.e., finding a fit for data that ideally should follow a linear trend but contains some noise. In that case, the network might find a very complicated fit that goes through all the points. As far as the network measures its performance, this fit is very good since it has no error (zero loss). However, such a fit considers the specific realization of the noise in addition to the linear trend. When required to predict the y value of unseen data, this will result in errors. This phenomenon is called overfitting. Figure 2.4 shows noisy data generated from a linear curve with noise. The correct fit is shown together with an overfitted line that follows the noise and an underfitted line that does not adapt to the actual trend of the data.

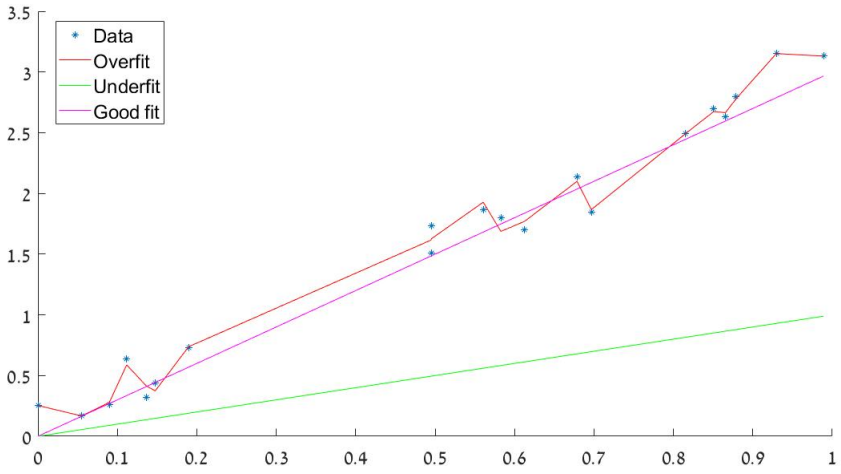

Figure 2.4. Noisy data with three levels of fitting. A polynomial of degree 15 generates the overfitting plot

To fight overfitting, the validation set was created. The validation set is data taken from the same source, on which the network performance is measured periodically during training. However, the network does not use this data to learn. When the network begins to improve on the training data, but its performance on the validation set worsens, it is a sign that the network is learning features specific to the training set and will fail to generalize on unseen data. Even so, the validation data adds a bias since it is used to tune the network. To ensure

that the performance reported is not biased by the specific validation set, a test set (data that remains unseen by the network until the end of the study) is used to evaluate the final performance of the network.

There are many techniques to combat overfitting. Two of them, L2 regularization and dropout layers, will be discussed here. L2 regularization adds a term of the form $\sum_i |w_i|^2 \lambda$ to the loss, λ is called the decay coefficient. Its purpose is to incentivize the network to find a configuration where the weights have small values. In analogy to the example shown in Figure 2.4, the L2 regularization would keep the polynomial coefficient small, reducing the flexibility and capability to adapt to noise. The second technique, the dropout layer, eliminates every training iteration a certain percentage of the neurons in a single layer at random. Every iteration, the neurons dropped are chosen randomly, and when the network is in use (after training), all the neurons are used, and the dropout layer is no longer in effect. By ignoring neurons at random, the dropout layer is supposed to eliminate the dependency of the network on any single feature. Instead, it should be able to extract information from all features, hopefully helping it learn the general trends of the data.

2.5 Summary

In summary, this chapter provided an overview of path loss and modulation schemes. Additionally, the ML algorithms, K-means and NN were presented. These topics are relevant for understanding the simulation, which creates the data used to train the NNs in this study. Moreover, the different components of NNs discussed here will be tuned as part of this training.

CHAPTER 3:
Simulation Infrastructure

This chapter explains the simulation used to create the data to train and validate the NN. The simulation emulates a communicating UAV network and other spectrum users. However, it does not simulate the actual waveforms or implements the signal detecting algorithms. Instead, it assesses performance based on SNR, modulation techniques, and other parameters. Based on the number of clusters, K, the simulation calculates the average throughput of the CRN.

3.1 Overview

In this MATLAB simulation, a network of 30 UAVs that sense the spectrum and communicate among themselves is considered. PUs appear at random, each with its characteristics (PU characteristics will be further discussed in Section 3.2). The CRN periodically senses the spectrum as part of the Sense-Allocate-Communicate (SAC) cycle it is conducting. After sensing the spectrum, it allocates Communication Cells (CCs) (frequency bands over a time period) to every pair of UAVs within a cluster until it runs out of channels or pairs to allocate. Following the allocation phase, the network enters the communication phase, where the payload communications happen. After running the cycle twice, the network reclusters with a new number of clusters and new centroids.

3.1.1 Space-Time-Frequency Structure

The UAV locations are uniformly sampled from a 600 m $\times$ 600 m square for the *(x,y)* coordinates and the range 100-1500 m for the z coordinate. The CRN operates within the frequency band 2.4-2.45 GHz, divided into ten channels. Each channel has a Guard Band (GB) separating it from its neighbors (including GBs at the edges of the entire frequency band). Each channel is approximately 4.5 MHz wide, excluding the GB, which is 0.45 MHz wide. The duration of the sensing and allocation phases varies depending on the False Alarm Rate (FAR) and Probability of Detection (PD) required and the number of UAVs per cluster. The communications phase is the only phase with a constant period of 200

ms. Each of the ten frequency channels is divided into five equally long time slots of 40 ms during each communications phase, resulting in a combined TDMA-FDMA channelization protocol with 50 channels. Guard time is not considered since the propagation time is orders of magnitude smaller than any other time scale in the simulation. These 50 channels are equally split between the K clusters in a sub-ideal division to reduce the time-expensive coordination communications (see Section 3.1.2).

Each group has a central node (UAV) that gathers the sensing information from the other UAVs in the cluster. That node then allocates the CCs to pairs of UAVs within the cluster, starting with the pair with the lowest channel loss. CCs where PUs have been detected are skipped to avoid conflict. Channel path loss is calculated using the FSPL model. The locations of the UAVs are assumed to be known with high precision since the Global Positioning System (GPS) locations are shared through a narrow-band secondary channel of bandwidth B_{ctr} (B_{ctr} = 5 MHz - 10% of the main communications bandwidth) where the CRN is considered to be the PU. This secondary channel is used for all allocation-related data transmission. Since the GPS information consists of 12 digits and the update rate does not exceed 1 Hz (the UAVs do not move fast), this information volume is considered negligible. Pairs with lower path loss are assigned first to maximize the throughput while allowing as many pairs as possible to communicate. A trivial solution to maximize the throughput under a given P_b constraint would be to allocate all CCs to the pair with the lowest loss. Still, such a system would have little use on the battlefield as only one pair would be able to communicate.

3.1.2 Information Sharing

Information sharing plays a central role in any CRN as it often presents the main trade-off between a centralized architecture and a distributed one. Depending on the hierarchy of information sharing, decision-making can be based on more or less information, leading to a closer-to-optimal performance at the expense of more resources for knowledge sharing. As can be seen in Section 3.5, this trade-off yields different clustering per scenario. This subsection discusses the details of the information-sharing mechanism in the simulation.

Firstly, the information shared between the nodes is the samples acquired during the sensing time. To comply with the Nyquist criterion for sampling, the sampling rate is $F_s = 10^7$ Hz,

and the bit depth is $Q = 16\frac{\text{bits}}{\text{sample}}$. Each cluster receives a bandwidth of $\frac{B_{ctr}}{K}$ to transmit this information within the cluster. The UAVs in each cluster send the samples to the central UAV in series using TDMA. The bitrate with which each UAV transmits the information depends on the channel loss between that UAV and the cluster central UAV. This dependence is further explained in Section 3.3. Overall, the time it takes each cluster to share this information is

$$t_i = n_i Q M R_{b,i} \tag{3.1}$$

where i is the cluster index, M is the number of samples, and n_i is the number of UAVs in the cluster, excluding the central one. Since the clusters are working in parallel, the total information sharing time is

$$t_{share} = \max_i (t_i) \tag{3.2}$$

3.1.3 Clustering Mechanism

As described at the beginning of this section, allocation of CCs happens within each cluster. The number of clusters is a given parameter in the simulation (and later predicted by the NN), and the centroids are chosen during the clustering. The clustering algorithm used is the K-means++ described in Section 2.3. The metric used for the algorithm is path loss. The choice of metric is logical since it is directly related to the transmission performance. The simulation outputs the information known to the CRN before clustering and the optimal number of clusters. The second output is calculated using a brute-force approach, where the simulation tries every value of K within a given range and calculates the average throughput in each case. However, to obtain the information known to the CRN before clustering, the simulation must know the previous clustering of the UAVs. To illustrate the importance of knowing the previous clustering, consider the following example. In one case, $K = 4$, and a PU is detected by four UAVs, each from a different cluster. In this case, all the clusters will know that a PU has been detected and the relevant CCs are occupied. However, since the information is not shared between the clusters, the geolocation of the PU is not possible. In another case, $K = 1$ and all four detecting UAVs are in the same cluster, in which case the 3D location of the PU can be calculated. For this reason, the simulation sequence is

1. Initialize a random PU transmission scheme and initialize UAVs

2. Run two SAC cycles for every value of K within a range (ending at t_0)
3. Choose K value with the best average throughout and assume that is the previously used K
4. Save the information known to the CRN thus far
5. Run another two SAC cycles for every value of K (starting at t_0)
6. Save K with the highest average throughput for the latter two SAC cycles

In the simulation, the clustering algorithm is assumed to be running on the UAV closest to the centroid of the network. The results are broadcast to the network in series (first, the cluster ID of the first UAV is published, then the cluster ID of the second, and so on). Since this information is narrow-banded (29 double-digit integers transmitted on a 5 MHz channel), its transmission time is negligible.

3.2 PU Generation and Detection

The simulation emulates the CRN and the PUs that use spectral real estate in tandem. The PUs represent any friendly user that has priority using the spectrum, as well as any adversary emitters that occupy the spectrum and cannot be silenced. The MATLAB simulation does not distinguish between the two types of PUs, as there is no essential difference in the characteristics of the transmissions between the two. One could claim that some knowledge about the friendly emitters can be shared with the CRN. However, the simulation does not include such integration as it is not guaranteed in reality.

3.2.1 Generation

The generation of the PUs happens at random. The parameters that completely describe each PU are

1. Effective Isotropic Radiated Power (EIRP)
2. Start time
3. Duration
4. Carrier Frequency
5. Position

The precise waveform is not simulated. The PU is taken into account by calculating the power received by the UAVs from each PU. The EIRP is sampled from the Probability Density Function (PDF) of a normal distribution

$$P = \max\left(0, \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) \tag{3.3}$$

where we consider that the mean and the standard deviation are equal $\mu = \sigma = 1$ W. The start time is also a random variable. The number of PU appearances every second was modeled by a Poisson distribution

$$Pr\,(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{3.4}$$

where X is the number of PUs to appear in one second, and $\lambda = 10$ Hz. In practice, the simulation ran with a time step of $dt = 10^{-3}$ sec, and the mean rate of appearances was normalized accordingly. The transmission duration was modeled as a uniform random variable in the range [0.1,10] seconds. Additionally, it has been assumed that all the PUs have the same bandwidth (equal to one frequency channel bandwidth). This simplifying assumption would make sense if the different transmitters require similar communication volumes and speeds. Each PU occupies a single frequency channel out of the ten at random with equal probability. The position of the PUs was drawn from the same PDF as the position of the UAVs in the network.

3.2.2 Detection

There are many techniques for signal detection. For a comprehensive list, see [6]. For example, some detectors, such as the matched filter detector, assume some knowledge of the signal to be detected. Although some assumptions can be made, besides knowing the bandwidth of the signal, nothing else is assumed. When little is known about the signal, the energy detector is often used when implementing a simple detector. For any energy detector, there are two parameters, η and M. η is the detection threshold, and M is the number of samples used for detection. Detection is done through hypothesis testing

$$\mathcal{H}_0\text{: } \mathbf{r} = \boldsymbol{n}$$
$$\mathcal{H}_1\text{: } \mathbf{r} = \mathbf{s} + \boldsymbol{n}$$

where the decision variable is the energy in the received signal. Overall, the hypothesis testing can be written as

$$\sum_{i=1}^{M} s_i^2 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \eta \tag{3.5}$$

In such a detector, the FAR is constant while the PD depends on the SNR. The FAR and PD of such a detector are given by [20].

$$P_{FA} = 1 - F_{\chi^2}(\eta; 2M) \tag{3.6}$$

where $F_{\chi^2}(\eta; 2M)$ denotes the central Chi-squared Cumulative Distribution Function (CDF) of degree 2M sampled at η. The probability of detection is then given by

$$P_d = 1 - F_{\chi^2}(\eta; 2M, 2M\xi) \tag{3.7}$$

This time, the CDF belongs to the non-central Chi-squared distribution with a non-centrality parameter of $2M\xi$ (where ξ is the received signal SNR). The parameters chosen for the simulation are $P_{FA} = 10^{-4}$ and $P_d = 1 - 10^{-5}$.

PU location information can only be obtained if enough UAVs from the same cluster have detected the same PU. To geolocate a PU in 3D, more than one UAV is needed. For non-directional antennas, as assumed that the UAVs have, Time Difference Of Arrival (TDOA) can be used. In such a case, four platforms are required [20]. Moreover, the requirement for the platforms to be in the same cluster comes from the information-sharing mechanism, as sensing information is shared only within the cluster.

3.3 Adaptive Modulation

The varying SNR and fixed requirement for P_b justify using adaptive modulation. As can be seen in Figure 3.1, for the same SNR, as the number of symbols in the alphabet goes up, the P_b rises. This characteristic is shared by all spectrally-efficient modulations such as MQAM or MPSK. This motivates the use of adaptive modulation that depends on the SNR (which is estimated based on the known GPS coordinates of the UAVs). If the path loss is low enough, a symbol can represent more bits while still satisfying the P_b requirement. In the simulation, the P_b requirement for the payload communications is $P_b = 10^{-5}$ while

for the synchronization information, it is $P_b = 10^{-6}$, since that information requires greater reliability. This mechanism introduces a non-linearity to the network throughput since modulation orders are discrete. Additionally, it is worth noting that this mechanism makes the ranking of pairs based on path loss in Section 3.1.1 logical, otherwise, a different path loss would not implicate a different bitrate.

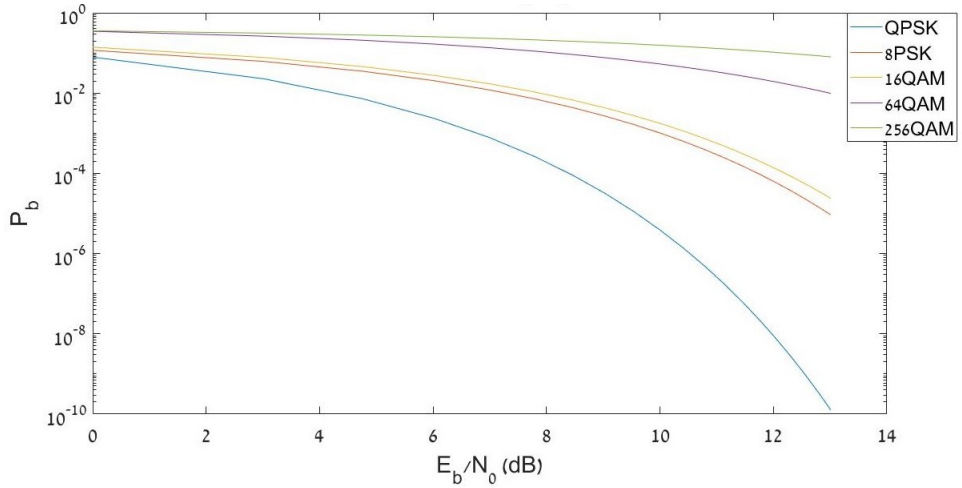

Figure 3.1. Error Probability as a function of SNR for different modulations

Nowadays, many systems implement Forward Error Correction (FEC), where bitrate is sacrificed to gain better BER. This simulation does not take FEC implementation into account for various reasons. First, for the sake of simplicity as the differential bitrate is already accounted for. Second, the coding gain of FEC is not as significant in an Additive White Gaussian Noise (AWGN) channel as it is in a fading channel [21]. Lastly, the path loss is extremely small due to LOS channels and small distances, the highest order modulation used is 1024QAM, sacrificing bitrate for the coding gain in the AWGN channel case would not make sense.

The modulation schemes used are (in ascending spectral efficiency order):

1. Binary Phase Shift Keying (BPSK)
2. Quadrature Phase Shift Keying (QPSK)

3. 8PSK
4. 16QAM
5. 64QAM
6. 256QAM
7. 1024QAM

The highest value of M was chosen based on the required P_b and received power, calculated from approximately 50 m away.

3.4 Outputs

The simulation outputs are used as input features for the NN. The throughput is calculated for each scenario (UAVs locations and PU transmission scheme). For each run, the simulation stores the information known to the CRN at the end of the first two SAC cycles and the K value associated with the highest average throughput in the next two SAC cycles. The average throughput is calculated across all the clusters

$$\overline{R_b} = \frac{\sum_{i=1}^{K} \sum_{j=1}^{N_i} R_{b,j} t_{comm}}{2(t_{sense} + t_{sync} + t_{comm})} \tag{3.8}$$

where K is the number of clusters, N_i is the number of allocated pairs in each cluster, and $R_{b,j}$ is the bitrate associated with the modulation assigned to each pair.
The output of each scenario run is structured as a vector

$$\mathbf{O} = [x_1, y_1, z_1, x_2, ...x_{30}, y_{30}, z_{30}, t_1, f_1, P_1, a_1, b_1, c_1, ...t_L, f_L, P_L, a_L, b_L, c_L, K] \tag{3.9}$$

where (x_i, y_i, z_i) is the location of the i^{th} UAV, (t_j, f_j, P_j) are the transmission start time, frequency band, and EIRP of the j^{th} detected PU, respectively. Finally, (a_j, b_j, c_j) are the coordinates of the j^{th} detected PU, and K is the number of clusters that yields the highest throughput. It is important to note that the PU information (not the raw samples) is shared only when the CRN reaches the point it reclusters after two consecutive SAC cycles.

3.5 Justifications

In the simulation presented above, many assumptions are made in parameter and model choices. This section will justify some of the main assumptions, either by analytical or numerical methods.

3.5.1 Path Loss Model Choice

There are many methods to consider the channel response between two antennas. In this case, with no major reflectors but the ground, simple models are often used. In [22], the authors present a method to determine the power of the range in the loss equation based on Fresnel zones. The term "breakpoint" is introduced to illustrate the threshold for using the FSPL model. According to this source, the threshold distance is given by

$$d = \sqrt{\left(\frac{4h_t h_r}{\lambda} - \frac{\lambda}{4}\right)^2 - (h_t - h_r)^2} \tag{3.10}$$

where h_t and h_r are the heights of the transmitter and receiver, respectively. If the minimum heights are assumed $h_t = h_r = 100$ m, and the wavelength that matches $f = 2.4$ GHz is plugged in, the maximum distance where the FSPL model holds is 32 km, 37.7 times larger than the maximum horizontal distance possible between UAVs ($\sqrt{2}\times$ 600 m = 848.5 m).

3.5.2 Number of SAC cycles

The number of SAC cycles is arbitrary. Perhaps, a different study can be made to find the optimal number of SAC cycles between clusterings. The less dynamic the scenario is, the longer the clustering will remain relevant. In this study, different numbers of SAC cycles have been tested. However, the ML phase was conducted with $N = 2$. The value $N = 2$ was chosen because a smaller N exhibits a greater correlation between the parameters known at the time of clustering and the performance during the SAC cycles. As time goes by, the information about the PUs becomes outdated. If the data fed into the NN is outdated for a larger portion of the time between the clusterings, predictions cannot be expected to be successful.

3.5.3 K Values Tested

Whenever the program calculates the optimal K, it does so by brute force, i.e., iterating over all possible values of K, calculating the throughput for each one. An obvious upper bound on K is the number of platforms in the network. However, that upper bound is not tight since it would not make sense to cluster the network in groups of one since the UAVs would not have who to communicate with. To find the maximum value of K to consider, K_{max}, a Monte Carlo simulation was run to calculate the optimal K for 200 random scenarios. Figure 3.2 shows the results of the Monte Carlo runs.

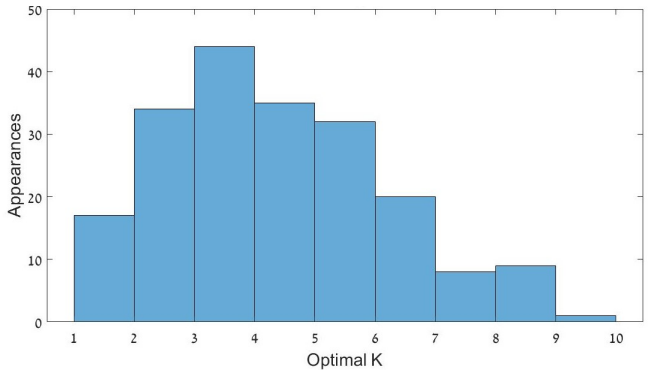

Figure 3.2. Number of instances when each K has been computed to be optimal. Monte Carlo simulation with 200 trials

The range of K values tested was [1, 10], where K = 10 had zero appearances. Therefore, the range used in the simulation for the brute-force run over K was [1, 10].

3.6 Summary

This chapter presented the simulation used to create data for the NN training and evaluation. The simulation generates a random scenario, including UAVs and PUs, and runs SAC cycles for different values of K. The scenario information will be used as features and the throughput as responses to train the NNs.

CHAPTER 4:
Network Training

The main goal of this study is to develop an NN that predicts the number of clusters based on the knowledge obtained by the CRN. The CRN goal is to maximize the average throughput while avoiding conflicts with the PUs. To do that, an NN is proposed to predict the number of clusters. This value will be fed to a K-means++ algorithm, followed by two SAC cycles. This section will discuss three methods proposed to find the best K value. The first solves a classification problem, while the other two solve a regression problem. All the methods use an FCNN.

The training phase included splitting the data into training, validation, and test sets. In most studies, the test set is left untouched until the end and is used to conduct a final evaluation of the NN, with no training afterward. This is done to ensure that the last reported results do not suffer from overfitting. In practice, the test set was not used in this study due to the limited success of the NN on the validation set. The training was conducted using MATLAB Deep Learning Toolbox, R2021a.

4.1 Classification Approach

The first try consisted of solving a classification problem. The inputs were the feature vectors described in Section 3.4 and the outputs were predictions of the number of clusters K. The possible outcomes were integers ranging from 1–10. The appropriate metric was the accuracy of the predictions.

4.1.1 Method

Training an NN can be a tedious process. The number of hyperparameters can be large, and the optimal configuration is often hard to find. Despite the complicated nature of the problem, random searches and manual tuning are often preferred over systematic searches [23]. In

this study, an ADAM solver was used with cross-entropy loss

$$L(\vec{y}) = -\sum_{j=1}^{K} y_j \log(y_j) \tag{4.1}$$

combined with a softmax activation layer

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4.2}$$

Other parameters, such as the L2 decay coefficient, dropout probability (and the addition of dropout layers), dataset size, learning rate, mini-batch size, activation layer types, and the number of epochs, have been changed on a per-run basis. The comparison was performed between runs that differ only in one parameter. All NNs consisted of FC layers in series with activation layers between them. The first layer had a number of elements similar to the number of input features, 120, and the last layer had a number of elements equal to the number of classes, 10. Figure 4.1 shows a very shallow architecture for classification. Each network starts with a feature input layer followed by a batch normalization layer. The batch normalization layer normalizes each batch across every feature separately. Often, the batch normalization layer is placed before the activation layer. Both options have been tested, and the difference found was minimal, yet it slightly favored the implementation presented here. softmax activation and classification layers follow the last FC layer.

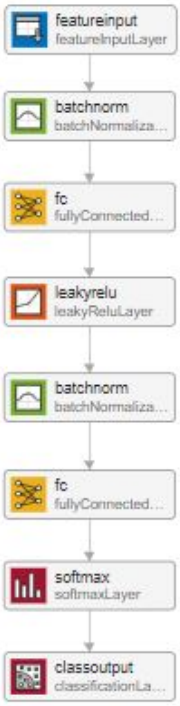

Figure 4.1. Network architecture for a classification problem. Each FC layer is followed by an activation layer and a batch normalization layer

The number of FC layers (followed by the batch normalization and activation layers) and their size was one of the variables changed as part of the training of the NN. After initial overfitting was achieved, fine-tuning was performed. This tuning aimed to balance the overfitting elements and the regularization elements used.

Some NNs had been trained using data corresponding only to one of two classes as part of the training. This was done as an attempt to solve a smaller, easier problem. In addition to having only two classes to classify instead of ten, this technique also alleviates the class imbalance problem. Choosing two classes out of the ten allows choosing classes with similar frequency of appearance rather than having classes both frequent and rare in the same dataset.

4.1.2 Dataset

The dataset used to train the NN in this phase was generated by the simulation described in Chapter 3. Each data point in the set represents a single run of the simulation; each run is comprised of two SAC cycles, followed by clustering, followed by another two SAC cycles. Every data point is represented by a vector described by eq. 3.9. The dataset created has 240,000 samples, but not all of them have been used. The train-validation split changed from run to run. However, typical numbers for the training set size are 20 000–60 000, while for the validation set, the size varied from 5000–20 000. The test set comprised 6000 samples; however, as mentioned at the beginning of the chapter, final testing on unseen data was not conducted. The data points used for training, validation, and testing did not overlap. The dataset is much larger than the training set because, for the trials with only two classes, the dataset must be large enough even after selecting two out of the ten classes. Table 4.1 shows the number of appearances of each category in the dataset. Some classes constitute as little as 1% of the dataset, while others, up to 19%.

Class	Appearances	Appearances (%)
1	2593	1.08
2	37 131	15.47
3	46 579	19.41
4	40 179	16.74
5	44 561	18.57
6	25 698	10.71
7	20 326	8.47
8	13 422	5.59
9	4794	2
10	4717	1.97
Total	240 000	100

Table 4.1. Frequency of appearance for each class

Since the vector described by eq. 3.9 did not have a constant length (as the number of PUs detected varied between runs), zero-padding was used to make the data samples equal in size. This technique is not optimal since it technically inserts non-existing PUs into the list of detected PUs. However, these PUs are added to the list with an EIRP of 0 W, so an NN that learns the relationship between the features and the outputs could potentially learn to ignore the non-emitting PUs located at the origin. The dataset was fed into the NN in the form of batches. This means that the gradient (in the ADAM solver that minimizes the network loss) is calculated across many samples and averaged rather than calculated on a per-sample basis. The weights are updated based on the average gradient of the batch. These batches are often referred to as "mini-batches."

Normalization of the dataset occurred in two forms. First, the entire dataset was normalized by subtracting the mean and dividing by the standard deviation. Each feature was normalized separately. Additionally, each batch was standardized by the "batch normalization" layers (seen in Figure 4.1).

4.2 Regression Approach

The second try consisted of generating the feature vectors for all possible values of K for each scenario. This means that ten samples can be generated for each in the previous method, one for each value of K. The NN output, in this case, was the predicted throughput. Once the NN has indicated the throughput for each K for a given scenario, a separate comparison of the throughputs has been done between the K values to find the estimated optimal K. Based on that estimation, an accuracy figure can be computed.

4.2.1 Motivation

The classification approach presents several discontinuous steps the NN is required to learn. As shown in [18], an FCNN can approximate any continuous function. Nevertheless, the problem the NN is needed to solve in this case suffers from many logical discontinuities. First, the CCs are divided into an arbitrary number of time-frequency bands of predetermined width and length. Afterward, the information extracted about the PUs depends on the number of detecting UAVs. Additionally, the data is shared using a TDMA-FDMA broadcast

protocol, implementing adaptive modulation, each modulation associated with a bitrate value in a manner that causes a discontinuity in bitrate as the path loss changes.

Moreover, the K-means++ algorithm clusters the UAVs in the CRN based on a discrete value (the number of clusters, K), each value leading to a significantly different clustering, thus, changing the possible pairs in the cluster and the throughput. All the steps described above share a common characteristic; they represent an arbitrary if-else rule imposed by the proposed solution and not an analytical relationship between the inputs and the outputs. Intuitively, learning many such steps would be challenging for an NN. For this reason, this approach, which eliminates the arbitrary set of rules the K-means++ algorithm presents, has been chosen.

4.2.2 Loss Function

The default loss function used by the MATLAB regression layer is the Half Mean Squared Error (HMSE) loss, given by

$$L = \frac{1}{2N} \sum_{j=1}^{N} (X_j - T_j)^2 \tag{4.3}$$

where $\vec{X}$ and $\vec{T}$ are the predictions and targets, respectively, fed into the network. In this case, the loss is calculated across a mini-batch. A different loss function often used is the Huber loss function

$$L_j = \begin{cases} \frac{1}{2}(X_j - T_j)^2 & \text{for } |X_j - T_j| \leq \delta \\ \delta |X_j - T_j| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{4.4}$$

$$L = \frac{1}{N} \sum_{j} L_j \tag{4.5}$$

The Huber loss handles outliers better than losses such as HMSE since the loss is linear for data points far from the center. A value of $\delta = 1$ was used to train the NN. Other loss functions such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are sometimes used, but as the Huber loss combines the MAE with the HMSE, no other loss

functions have been used. However, the RMSE of each iteration was calculated and plotted in addition to the loss and was used to tune the hyper parameters.

4.2.3 Dataset

Unlike in the classification case, each scenario creates K_{max} data points; in this case, $K_{max} = 10$. The dataset consists of 100 000 different scenarios, generating 1 000 000 data points. The training set consisted only of 296 000 of those samples, while the validation set contained 4000 samples. The entire dataset was not used for time considerations. The same normalization used on the classification dataset was used on this regression dataset. Additionally, zero-padding was used.

4.3 Throughput Prediction

The goal of the third and last approach was to predict scenario throughput for different values of K. However, the key difference in this approach is that the input feature vector has changed.

4.3.1 Input Features

As explained in Section 4.2.1, many arbitrary rules are involved in the simulation heuristic. To further simplify the NN task, a further simplification was made. Instead of feeding the network with information about the scenario and the detected PUs, the new input was a list of all the possible pairs, their path loss, and whether each pair has been allocated or not. Another input was the number of clusters. The overall structure of the feature vector was as follows

$$\mathbf{F} = [L_1, A_1, ..., L_N, A_N, K] \tag{4.6}$$

where L_i is the path loss of the i^{th} pair, A_i is a binary variable equal to 1 if the pair has been assigned a CC, and 0 otherwise. N is the number of possible intra-cluster pairs summed across all clusters. In essence, all that is left for the NN to learn is the relationship between path loss and bitrate (given by the adaptive modulation scheme) and the significance of the binary variable. To make the learning challenge easier for the NN, the throughput values that were to be predicted were total throughput and not average throughput. This change

spares the need to consider the time wasted on synchronization, which depends on the path loss of all pairs, regardless of the quality of the link, and the TDMA-FDMA protocol used.

4.3.2 Dataset

For this phase, a new dataset was generated. The size of the training set varied between runs in the range of 10 000–270 000, and the validation set size was between 2000–10 000. All the features, including the binary ones, have been normalized. Any feature that assumed a single value (always 1) remained unchanged.

4.4 Summary

The three used approaches for training the NNs were laid out in this chapter. The classification and regression approaches aim to find the optimal K value. In contrast, the throughput prediction approach attempts to calculate the throughput of each given scenario for any value of K. Additionally, the dataset used for each method was discussed.

CHAPTER 5:
Results

This chapter will discuss the performance of NNs trained to solve the problem described in Chapter 4. Results will be presented for each of the approaches presented in Chapter 4, classification, regression, and throughput prediction. For each approach, a different initial network will be considered, each one will be modified differently to maximize the performance of the NNs on each approach. The parameter used for final evaluation is the accuracy of the NN prediction. The easiest benchmark to compare to is the most frequently appearing class, e.g., if the most popular class in a very large dataset is $K = 3$, which makes up 19% of the dataset, an NN is considered successful if it has an accuracy of more than 19%. Otherwise, it would be more beneficial to guess $K = 3$ every time. This does not consider the expected value of the average throughput an NN can achieve, but since that is not the target function defined to the network, it is also not the parameter used to assess it. Accuracy can also be computed for the regression methods, using the predicted throughput as a basis for predicting the optimal K and comparing that prediction with the actual optimal K. As will be discussed in the following sections, after examining the difference in throughput between different values of K and comparing it to the network prediction RMSE, it will be seen that calculating the accuracy of the prediction is unnecessary given the large error in the predicted throughput.

5.1 Classification

Many architectures were tested for the classification approach, and each architecture was trained over a range of hyperparameter values. First, a base architecture was obtained, later to be tuned by trial and error.

5.1.1 Classification Architecture A

The first NN tested had five hidden layers, each with 300 neurons. A leaky ReLU activation layer followed each FC layer, and a dropout layer with $p = 0.2$ was added before the

classification layer. This architecture was achieved after a manual trial and error process and was tested across many L2 decay coefficients and different β_2 values.

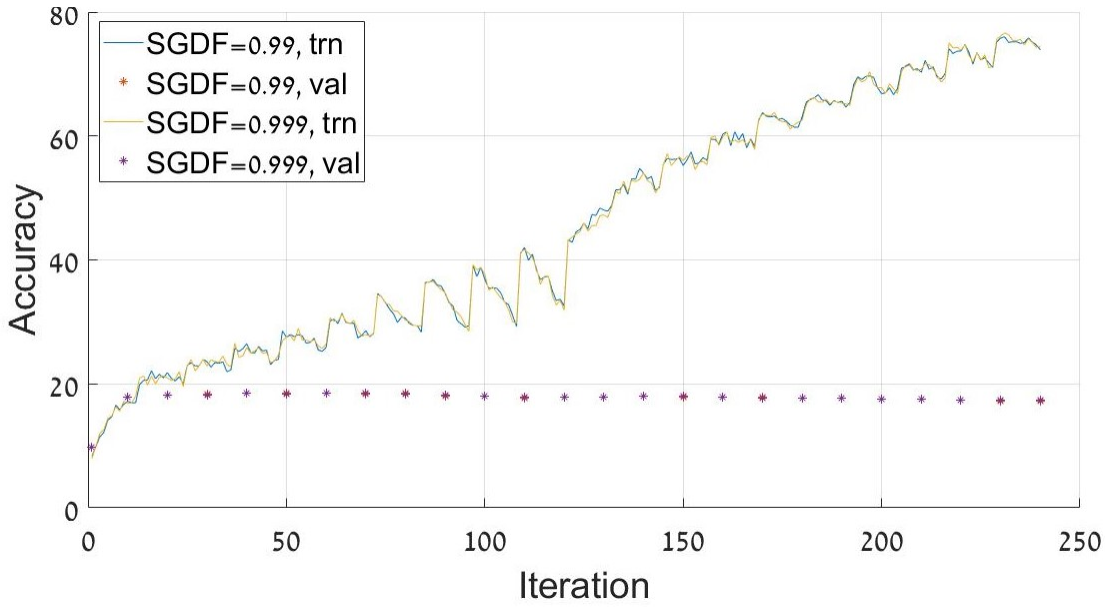

Figure 5.1. Classification Architecture A training and validation accuracy for 240 iterations. Comparison of two different Squared Gradient Decay Factor (β_2) values

As can be seen in Figure 5.1, both values have similar performance, and they both overfit. For that reason, the next comparison was between L2 coefficients. For this comparison, the value $\beta_2 = 0.999$ was chosen.

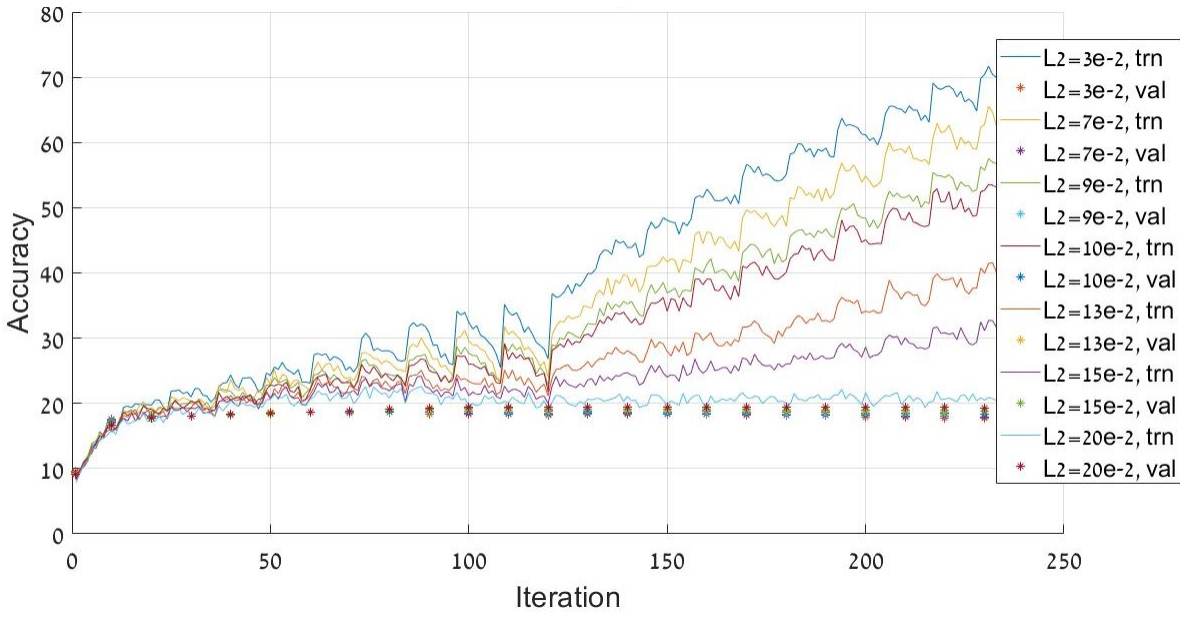

Figure 5.2. Classification Architecture A training and validation accuracy for 240 iterations. Comparison of different L2 decay coefficients

Figure 5.2 shows that the increase in the L2 coefficient causes a regularizing effect. However, since the regularization lowers the training accuracy without affecting the validation accuracy, varying the dropout probability with a constant L2 coefficient has also been explored as an alternative regularization method. Figure 5.3 shows the Classification Architecture A with an L2 coefficient of 10^{-2} for many dropout probabilities.

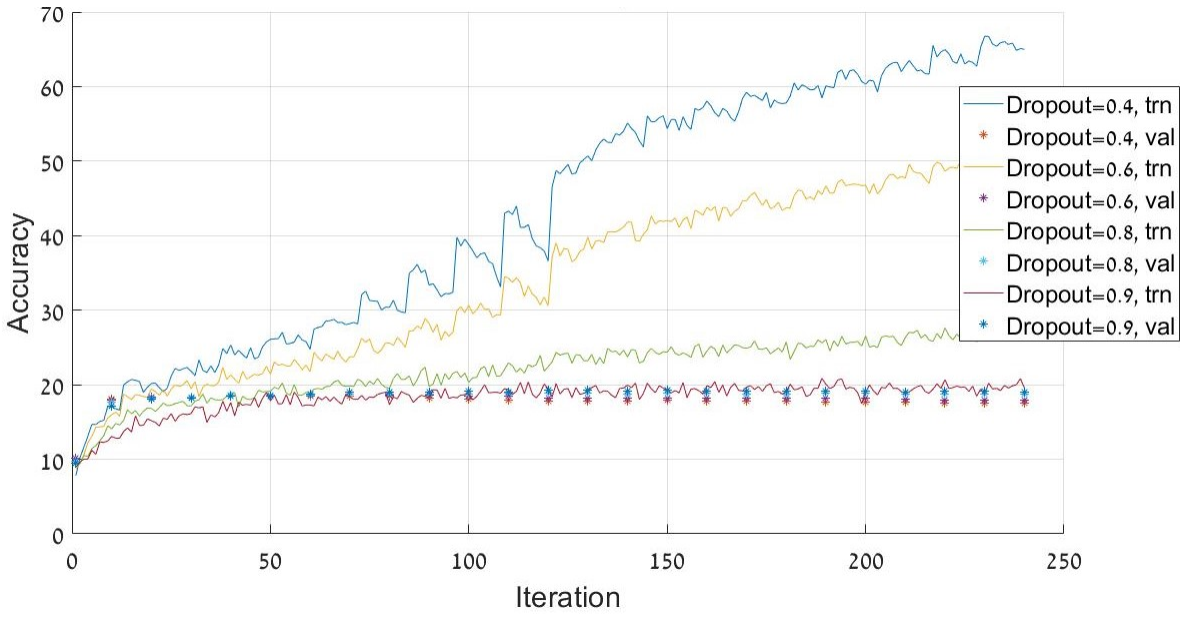

Figure 5.3. Classification Architecture A training and validation accuracy for 240 iterations. Comparison of different dropout probabilities for an L2 coefficient of 10^{-2}

As before, changing the dropout probabilities has a regularizing effect but does not improve the validation accuracy. The jump in the accuracy slope seen in Figures 5.1, 5.2, and 5.3 around iteration 100 is due to the adaptive learning rate used. In the plots presented in this section, after every 100 iterations, the learning rate decays by 0.1. This learning rate schedule allows the loss to keep decreasing even after having reached a minimum by finessing the search.

5.1.2 Varying the Number of Layers

The next architecture tested is a three-layer FCNN, again, with 300 neurons in each layer with a dropout layer before classification with $p = 0.2$. Figure 5.4 shows the training and validation accuracy as training progresses for different L2 values.

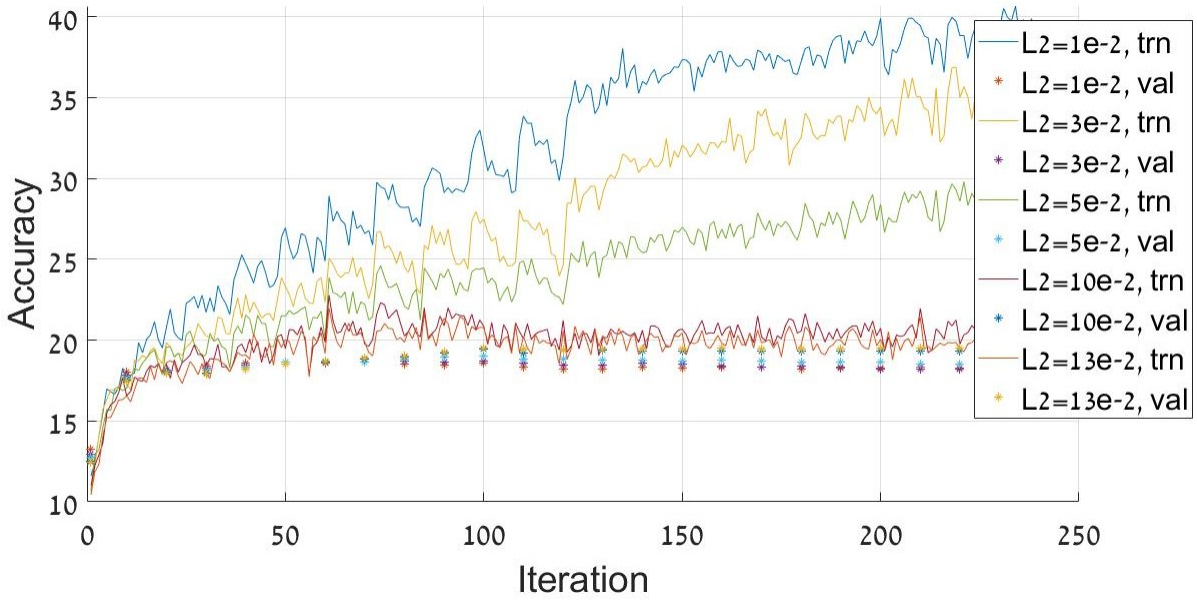

Figure 5.4. Three-layer network, training and validation accuracy for 240 iterations. Comparison of different L2 decay coefficients

Since the three-layer network performed worse than the five-layer one, a seven-layer network was also tested. The logic behind trying a larger network is that since the regularization did not improve the validation accuracy, it is possible that previously shown NNs are not complex enough to learn the problem presented in this study. Figure 5.5 shows the results achieved with a more complex network.

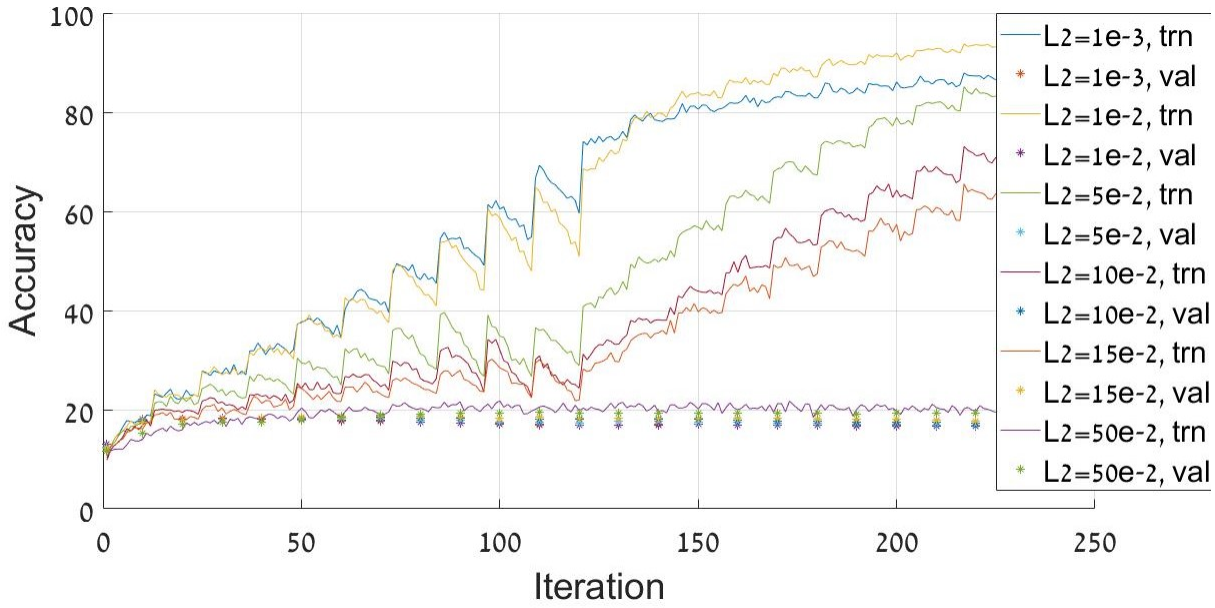

Figure 5.5. Seven-layer network, training and validation accuracy for 240 iterations. Comparison of different L2 decay coefficients

In this case, the more complex model improved the training accuracy (which can be brought to an arbitrarily high value), but the validation accuracy remained unaffected.

All figures show that the training accuracy can be arbitrarily high if regularization is suppressed. As regularization becomes more significant, the validation accuracy does not improve. Instead, the training accuracy converges to the validation accuracy. Since the validation accuracy is never above the fraction of the most popular class, none of the NNs have been successfully trained.

5.2 Regression

The results presented in this section show the performance of the regression approach. In this approach, the class was predicted based on the throughput prediction of the NN. Each scenario used to create an example for the NN was run K_{max} times, and each time a different number of clusters was used. The throughput calculated for each run was used as the value

to predict. The regression network was given all K_{max} examples and their throughputs and had to predict the throughput for each K. Based on that evaluation, the appropriate K values were computed. The throughput values have been normalized across the entire dataset. For every example created for the classification case, now there are 10.

5.2.1 Regression Architecture A

Moving to the regression approach required finding an initial architecture from scratch. The first architecture tested was a seven-layer FCNN. Each layer comprised 400 neurons, followed by a ReLU layer and a batch normalization layer. Before the regression layer, a dropout layer with $p = 0.2$ was added. Other values used were $\beta_1 = 0.99$, $\beta_2 = 0.99$, and $L2 = 10^{-8}$. The first comparison was between a Huber loss function and an HMSE loss function. Figure 5.6 shows the performance of each NN. The metric used is the RMSE of the normalized throughput.

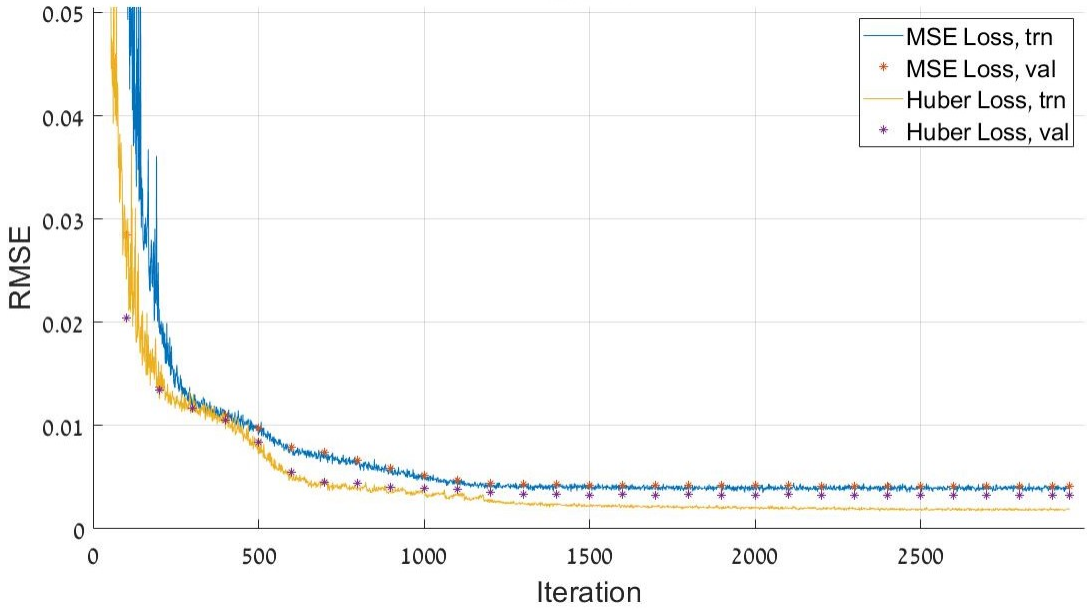

Figure 5.6. Seven-layer network, training and validation accuracy. Comparison of different loss functions

As can be seen, the Huber loss performs better than the HMSE loss in training and validation; however, the difference in the validation error is minor. The next step was to train an NN utilizing the Huber loss with different L2 decay coefficients. Figure 5.7 shows a seven-layer NN implemented with the Huber loss for L2 decay coefficient values of 10^{-8} and 10^{-9}. As can be seen, both plots converge as the number of iterations grows.

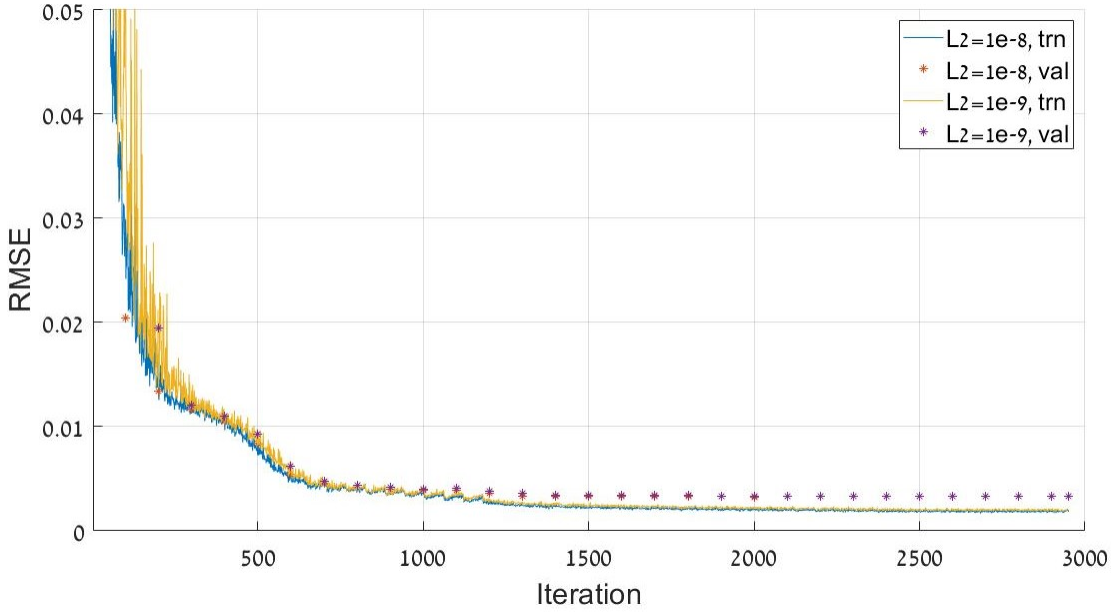

Figure 5.7. Seven-layer network, training and validation accuracy. Comparison of different L2 decay coefficients

5.2.2 Varying the Number of Layers

The average standard deviation of the normalized throughput between different K values in a single scenario was $\sigma \approx 0.002$. Thus, the RMSE required to predict the number of clusters based on the throughput should be smaller than 0.002 (preferably an order of magnitude smaller). Otherwise, the error in the predicted throughput for each value of K will be too great to indicate the correct K reliably. As shown in the figures above, the RMSE achieved is approximately twice the standard deviation. Therefore, models with different complexities

should be considered. Figure 5.8 shows a ten-layer NN (with the same parameters as before) for both Huber and HMSE loss.

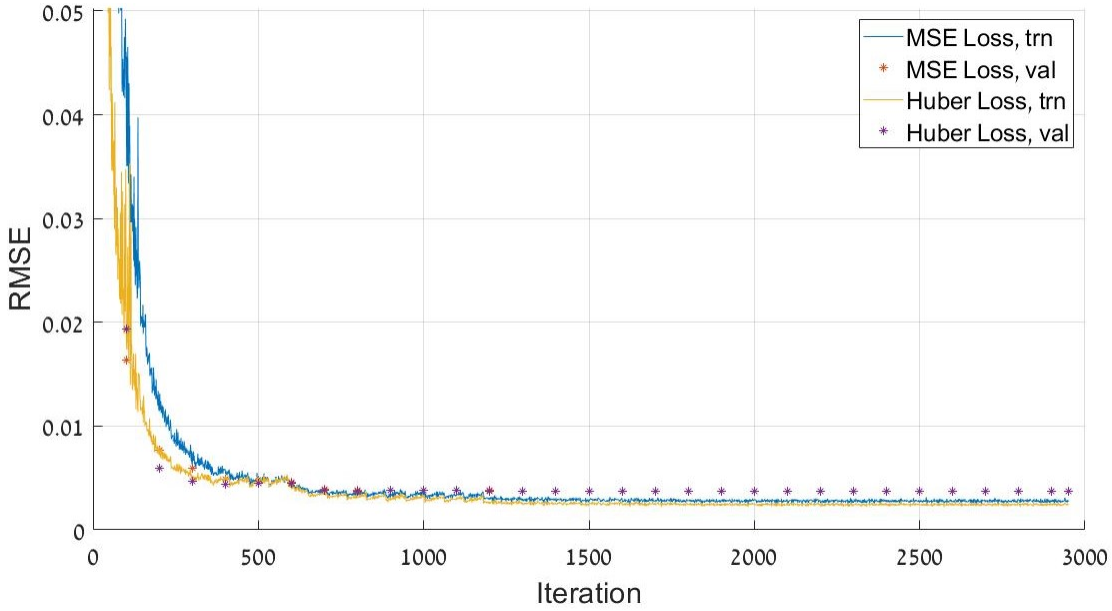

Figure 5.8. Ten-layered NN for throughput prediction. HMSE and Huber losses are compared

Adding three more layers does not significantly affect the throughput error.

5.2.3 Leaky ReLU

Since changing the number of layers did not have the desired effect on the performance, a different parameter was changed. ReLU activation layers suffer from the "dying ReLU problem." When the input is negative, the output is constantly zero, so the gradient is zero and the neuron stops affecting the learning in any way, i.e., dies. For that reason, the leaky ReLU was introduced. The leaky ReLU coefficient (the slope in the negative domain) was 0.01.

Figure 5.9 shows the five-layer network, each layer 400 neurons wide, with leaky ReLU

layers instead of the previously used ReLU activation layer. Additionally, Huber loss was used.

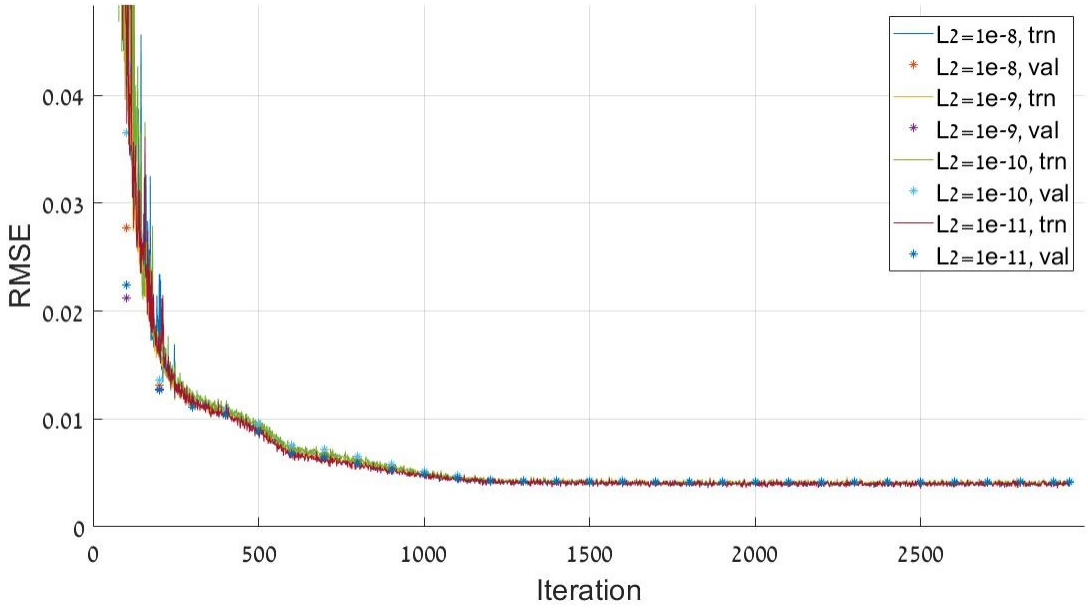

Figure 5.9. Five-layered NN for throughput prediction with leaky ReLU activation layers

Since the model did not generalize properly, a seven-layer network with leaky ReLU activation layers was also tested. Different L2 regularization coefficients were tested to find the sweet spot between under and over fitting. Figure 5.10 shows the performance of each NN. The metric used is the RMSE of the normalized throughput.

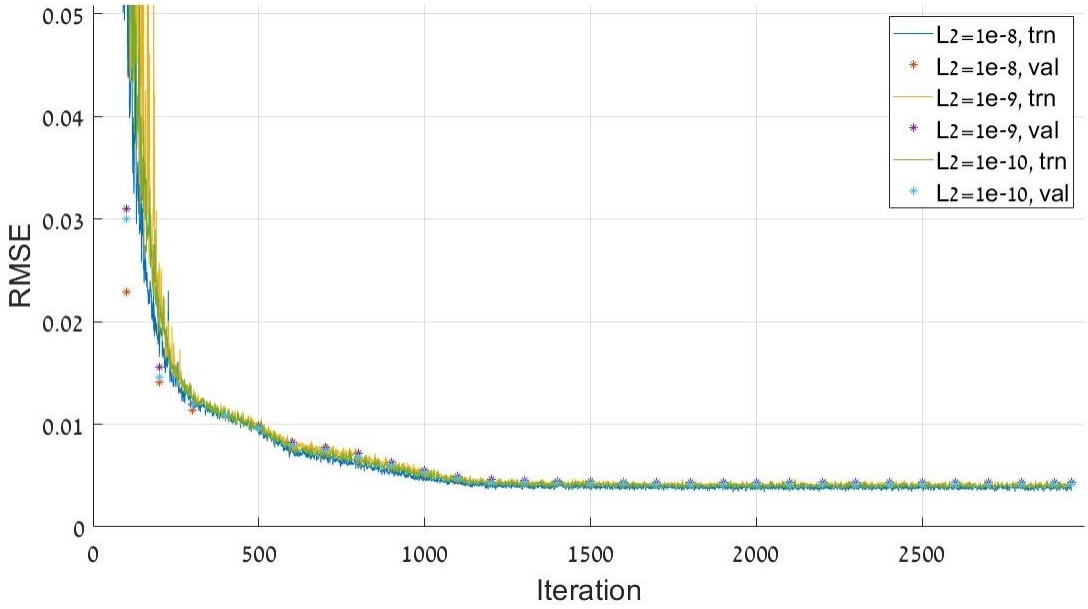

Figure 5.10. Seven-layered NN for throughput prediction with leaky ReLU activation layers

It can be seen that the improvement in RMSE is negligible compared to the five-layer case. Other architectures with more and fewer layers have also been tested but performed poorly (and so did changing the number of neurons in each layer).

In terms of accuracy, the best-performing model was the seven-layer network with ReLU activation, Huber loss, and an L2 coefficient of 10^{-8} (seen in Figure 5.6 labeled as "Huber loss"). The test set, which contained 2000 examples (different scenarios, each with 10 K values), had $K = 3$ as its most frequent class, constituting 19.33% of the dataset. In comparison, the best model had an accuracy of 14.09%.

5.3 Throughput Prediction

As mentioned in Chapter 4, the results yielded by the two previous methods were unsatisfying. Hence, a simpler problem to solve was formulated. This section will present the results

of this method. However, since this approach does not address the accuracy of predicting the number of clusters, the main metric is the throughput prediction error. Unlike in the first regression approach, the throughput calculated is not the average but the total throughput. The average standard deviation of the normalized throughput (calculated for each scenario for the different number of clusters) is 0.02.

5.3.1 Initial Tuning

The first NN trained tested for throughput prediction was a seven-layer network with RMSE loss, 400 neurons per layer, ReLU activation layers, and an L2 decay coefficient of 10^{-5}. Figure 5.11 shows the performance of the network. As can be seen, the NN overfits. Even so, the training loss is too high.

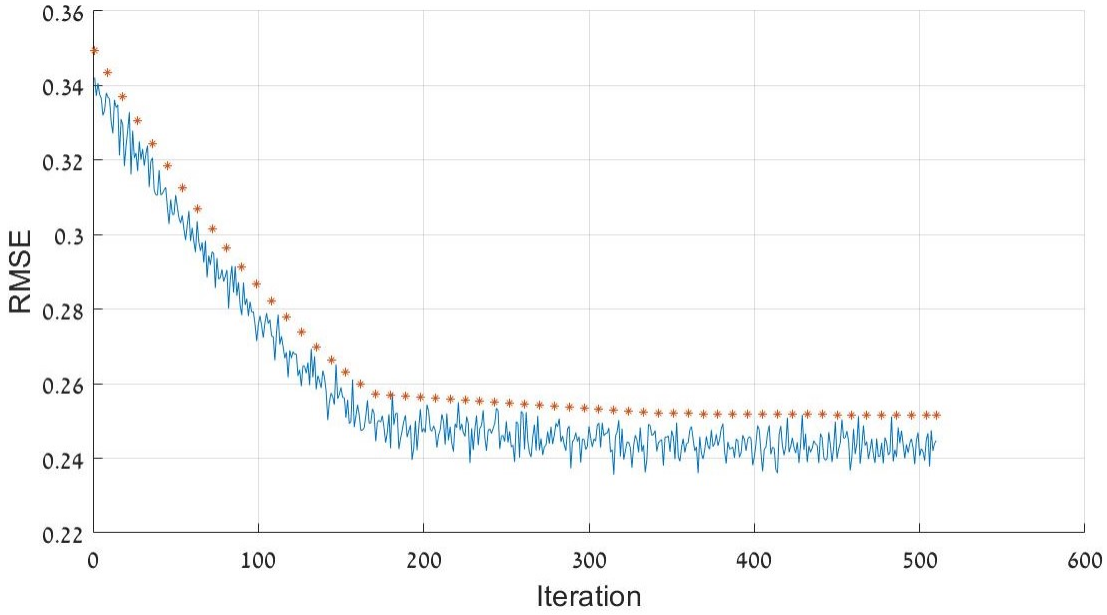

Figure 5.11. Seven-layered NN, 400 neurons per layer

After a prolonged tuning process, which included comparing different numbers of layers, β_2 coefficients, L2 decay coefficients, the number of neurons per layer, and the addition of dropout layers, a base design was found. This architecture, Throughput Prediction A,

consisted of three layers of 400 neurons each, an L2 coefficient of 10^{-9}, and ReLUs activation layers between the FC layers. No dropout layer was included. Figure 5.12 shows the performance of Throughput Prediction A. The metric used is the RMSE of the normalized throughput.

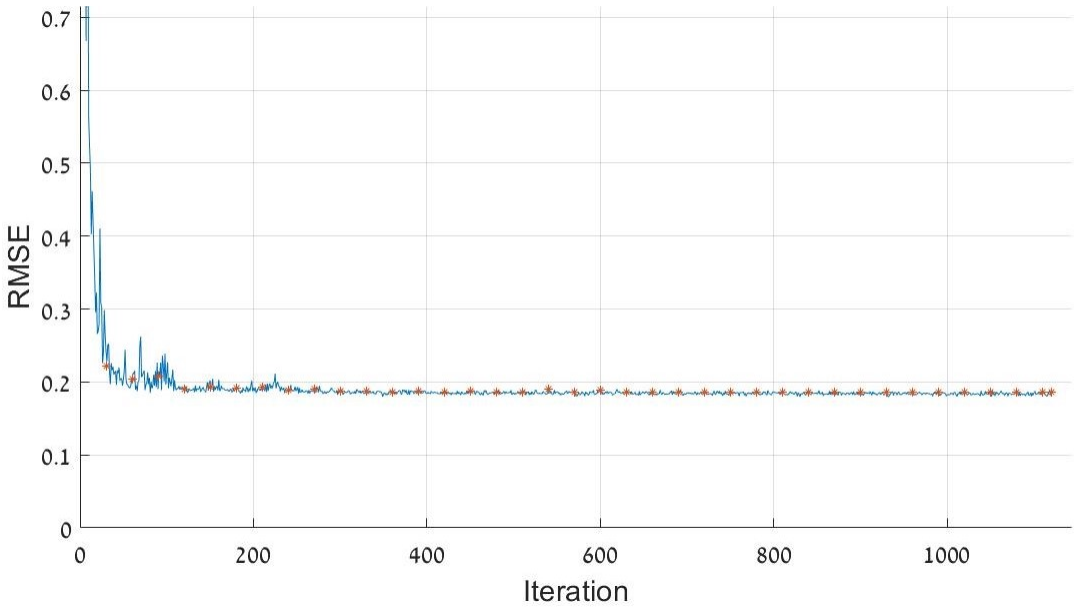

Figure 5.12. Performance of Throughput Prediction A across 1200 Iterations

First, Throughput Prediction A does not overfit despite having an extremely small L2 decay factor. Second, it performs better than the initially proposed architecture (seven-layer NN). Additionally, the learning rate schedule has been tuned for optimal results for this training run. The piece-wise learning schedule reduced the learning rate (initially 10^{-2}) by 50% every 360 iterations. The RMSE achieved was approximately 0.19.

5.3.2 Architecture Optimization

After designing a network by coarse tuning, a finer parameter search has begun. The first adjustment tried was the replacement of the ReLU activation layers with leaky ReLUs. Figure 5.13 shows the RMSE loss of this NN training and validation sets. As seen in the

figure, the change in the activation layer makes no significant difference in performance. Since there is no evident downside to using the leaky ReLU activation layer and it has some potential benefits over the ReLU layer, leaky ReLUs have been used for the rest of the study.

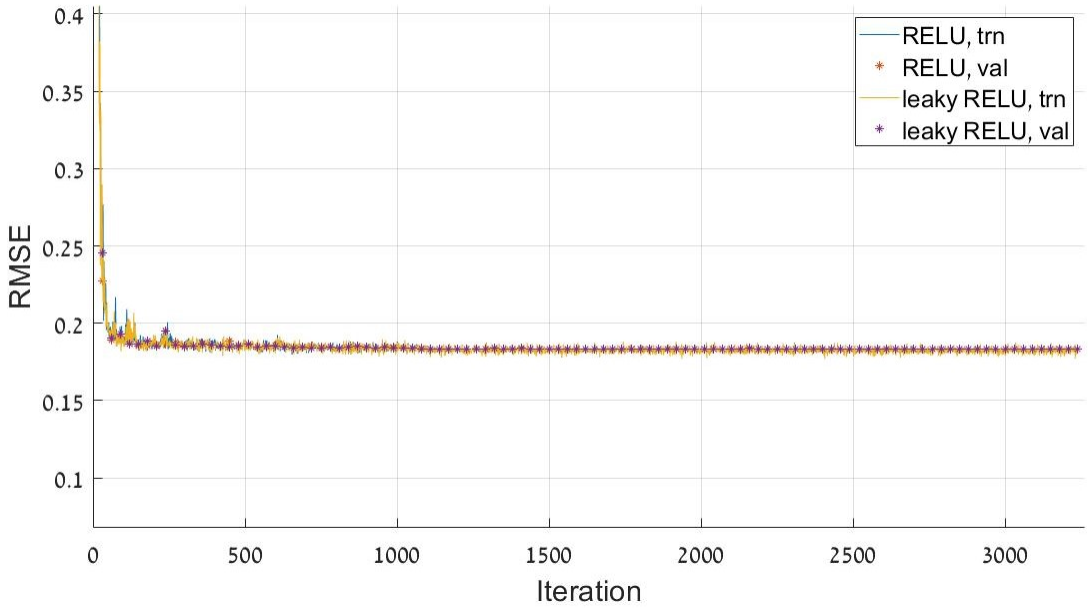

Figure 5.13. Throughput Prediction A with leaky ReLU activation layers instead of the regular ReLU layers

Next, having not been able to learn properly despite little regularization, a more complex network was trained. This network consisted of three layers of 800 neurons each, followed by two layers of 400. The rest of the parameters remained unchanged. Figure 5.14 shows the performance for different values of L2 decay coefficients. As can be seen, some configurations obtain a training loss of less than 0.15. Still, even very large regularization coefficients do not prevent overfitting, with no validation set yielding an RMSE of less than 0.2. This performance leads to the conclusion that a more complex design will lead to overfitting while not achieving any improvement on unseen data.

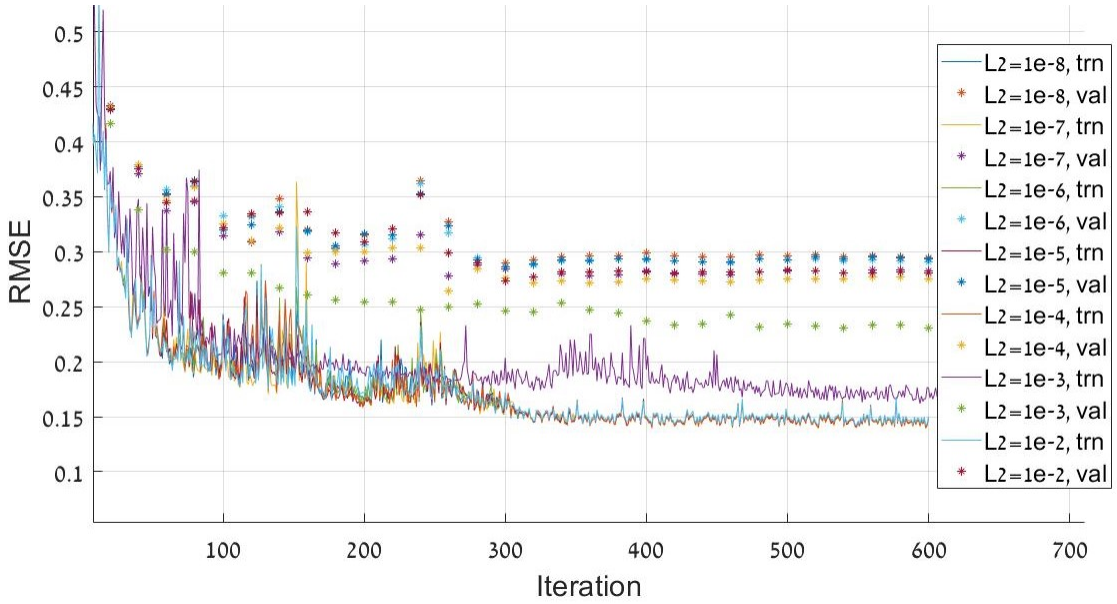

Figure 5.14. Enhanced architecture consisting of three layers of 800 neurons followed by two layers of 400 neurons. Leaky ReLU activation layers are used

The final tuning performed was on the number of neurons per layer. The configuration shown in Figure 5.13 with leaky ReLU activation layers has been trained with 150, 200, and 400 neurons per layer. Figure 5.15 shows the performance of these NNs.

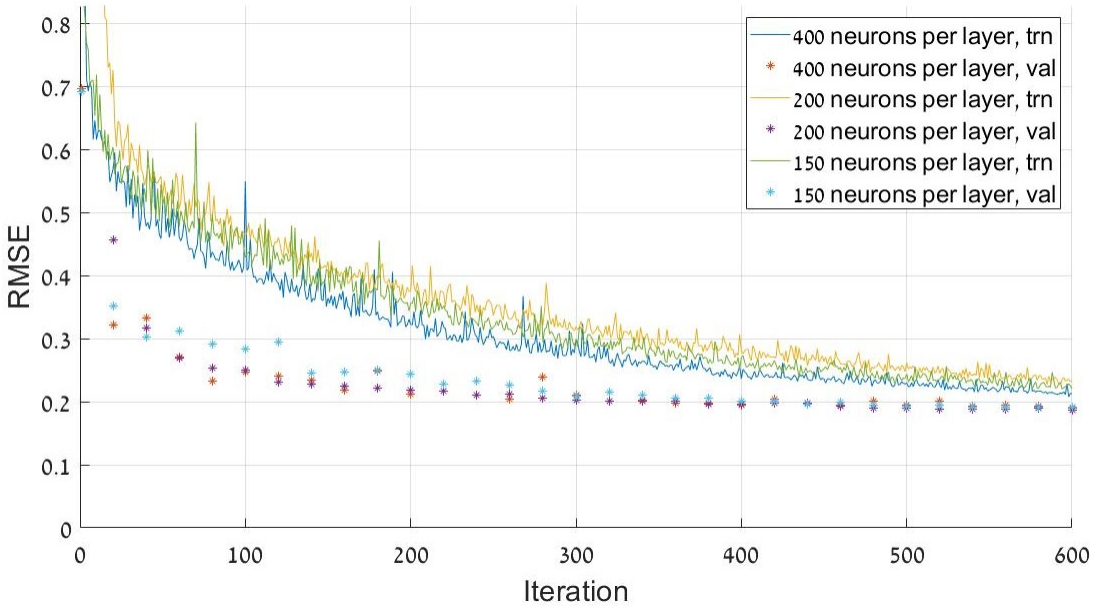

Figure 5.15. Comparison of three-layer networks with different numbers of neurons per layer

As shown in Figure 5.15, the difference is insignificant. However, since the 200 neurons-per-layer configuration performed better by 0.006, it has been selected for further evaluation.

5.3.3 Final Results

To extract maximum performance out of the architecture found to be best, training with 60 000 iterations was performed on a training set of size 40 000. The NN trained had three layers of 200 neurons each and an L2 coefficient of 10^{-3}. To maximize the training efficiency, the learning rate (initially 10^{-5}) was reduced by half every 10 000 iterations. That is a good compromise between a static and an extremely dynamic schedule. Figure 5.16 shows the performance of each NN. The metric used is the RMSE of the normalized throughput.

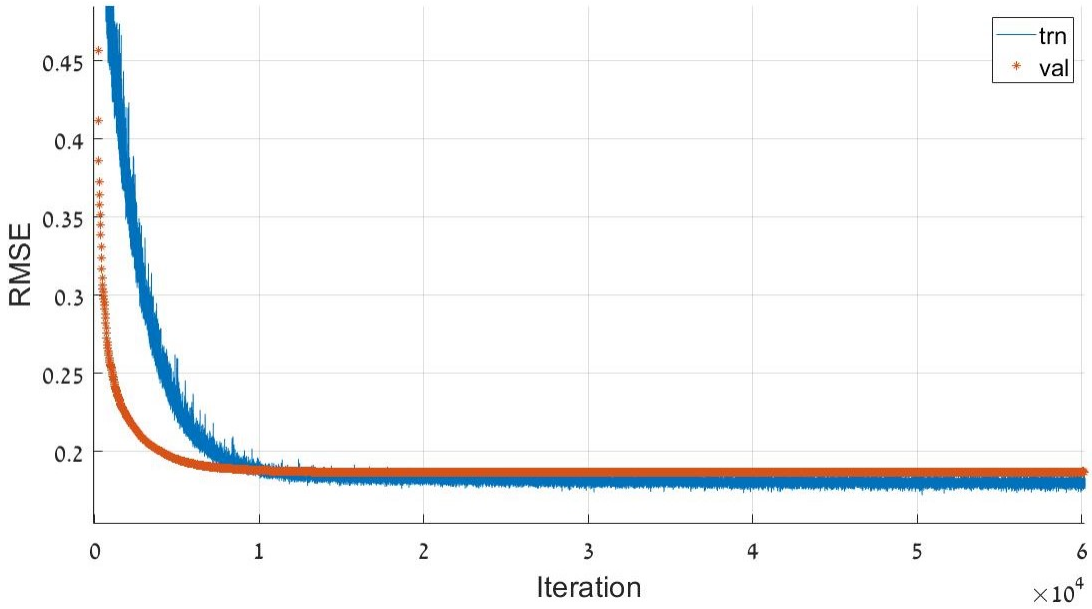

Figure 5.16. Final network test - an exhaustive training of the best architecture found

The best RMSE achieved was 0.1865. It was achieved in the last iteration of the training shown in Figure 5.16. As mentioned at the beginning of this section, the standard deviation between different numbers of clusters is 0.02. Consequently, the NN performance is insufficient to predict the total throughput accurately for the CRN needs.

5.4 Summary

This chapter presented the results for all three approaches. The first two methods produced a prediction accuracy while the third measured the RMSE error of the throughput prediction. None of the methods yielded satisfying results, i.e., results that can aid the CRN decision and outperform the simplistic heuristic proposed in Chapter 4.

The results show that overfitting to arbitrarily high accuracy and low error is possible. Still, the regularization only affects the training metrics instead of improving the validation metrics. Even so, it must be noted that it is impossible to exhaust the search space as parameters

can take continuous values and network architectures can grow more complex indefinitely. To summarize, the methods presented in this study were not successful. However, there are a few takeaway points to be considered. The first is that the effect the regularization had on the accuracy indicates that the networks tested were not complex enough, and more complex networks would also require more powerful hardware. The second is that, although there is an optimal number of clusters, the difference in throughput between the optimal clustering and the following best clusterings is very small (hence the small standard deviation in the regression approach). This would make the cost-effectiveness of the optimization proposed worth reconsidering. Additionally, due to the continuous nature of NNs and their inability to learn how to solve the given problem in this study, other ML algorithms should be tested.

CHAPTER 6:
Conclusions

This chapter lays out the conclusions of the research and possible future research opportunities and recommendations.

Communications are an essential component of any large-scale military force or operation. Nonetheless, adversarial efforts and the commonness of friendly radios make spectral resources very scarce. The CR paradigm attempts to solve this problem via opportunistic spectrum use. This approach consists of sensing the spectrum, detecting the presence of other transmitters that cannot be interfered with, the PUs, and the reallocation of spectral resources between the nodes of the CRN. This sensing-allocation cycle must happen periodically since the PU spectral occupancy changes with time as transmitters stop and begin to transmit or hop frequencies.

This study presented an ML approach to optimizing the efficiency of the CRN, i.e., maximizing the throughput under the constraint of not interfering with the PUs. The basis for the study was a simulation that was used to create data to train NNs. The simulation generated PUs randomly and calculated the throughput for different configurations of the CRN. Later, the NN had to learn the relationship between the initial conditions of the scenario and the optimal structure that maximized the average throughput through the network.

As shown in Chapter 5, the NNs had limited success. Even so, they made substantial progress in learning the solution to the problem. Additionally, it is important to note that this study was limited in time and resources. A more exhaustive study could perhaps find a better configuration that outperforms the simple heuristic proposed as a baseline.

6.1 Future Work

Much additional work can be done to continue the research presented in this thesis. This section will offer three suggestions for further investigation.

First, further tuning can be done to find a better-performing solution to the problem presented at the beginning of this chapter. For example, larger NNs can be tested with novel elements like special activation layers and networks other than FCNNs. The motivation to continue the current approach is that most of the information needed to predict the throughput and choose the optimal configuration is available at the moment of prediction.

The next suggestion would be to solve this optimization problem using other ML algorithms. For example, the k Nearest Neighbors (kNN) algorithm could predict the number of clusters based on many features and a metric that measures the similarity between a new scenario and known examples. Additionally, other clustering methods could be tested to replace the K-means++ algorithm.

Lastly, other CR protocols can be used. This work suggested a certain synchronization and communication protocol, however, different approaches to CR do exist. For instance, [5] introduces many architectures and cooperation schemes. These methods can be compared to find the best one for each problem.

APPENDIX: Supplementary Notes

A.1 Union Bound Approximation

The probability of the union of events is upper bounded by the sum of the probabilities of the events

$$Pr\left(\bigcup_{i=1}^{L} X_i\right) \leq \sum_{i=1}^{L} Pr\left(X_i\right) \tag{A.1}$$

Without the loss of generality, let $\mathbf{y}$ be the decision variable and X_i = ($\mathbf{y}$ is closer to $\mathbf{s}_i$ than to $\mathbf{s}_1$ | $\mathbf{s}_1$ was sent). Since the noise of the received signal is Gaussian with variance σ^2, the event X_i can also be written in terms of the noise and the distance between $\mathbf{s}_i$ and $\mathbf{s}_1$, $d_{1,i}$

$$Pr\left(X_i\right) = Pr\left(\mathbf{N}_1 > \frac{d_{i,1}}{2}\right) \tag{A.2}$$

That is, if the noise brings the decision variable closer to $\mathbf{s}_i$ than to $\mathbf{s}_1$, the event X_i is occurring. Writing the right-hand side of the equation in terms of the Q-Function

$$Pr\left(X_i\right) = Q\left(\frac{0.5d_{1,i}}{\sigma}\right) \tag{A.3}$$

Looking at the probability of symbol error given $\mathbf{s}_1$ was sent

$$Pr\left(\text{error}|\mathbf{s}_1\right) = \sum_{i=2}^{L} Q\left(\frac{d_{1,i}}{2\sigma}\right) \tag{A.4}$$

Since all the symbols are equiprobable

$$P_e = \frac{1}{L}\sum_{i=1}^{L}\sum_{j\neq k} Q\left(\frac{d_{k,j}}{2\sigma}\right) \tag{A.5}$$

The fast decrease of the Q-Function allows us to consider only the nearest neighbors who are d_{min} from the sent symbol. Let N_k be the number of nearest neighbors the k^{th} symbol

has; then, for reasonable SNRs, the error probability can be approximated by

$$P_e \approx \frac{1}{L} \sum_{k=1}^{L} N_k Q\left(\frac{d_{min}}{2\sigma}\right) \tag{A.6}$$

Defining the average number of nearest neighbors as $\bar{N_k} = \frac{1}{M} \sum_{k=1}^{L} N_k$, we get the approximation (which becomes a tight bound as the SNR goes up)

$$P_e \approx \bar{N_k} Q\left(\frac{d_{min}}{2\sigma}\right) \tag{A.7}$$

List of References

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE personal communications*, vol. 6, no. 4, pp. 13–18, 1999.

[2] B. Wang and K. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE Journal of selected topics in signal processing*, vol. 5, no. 1, pp. 5–23, 2010.

[3] K. Chen and R. Prasad, *Cognitive Radio Networks*. Wiley, 2009. Available: https://books.google.com/books?id=6sDkHWNHdZoC

[4] B. Fette, *Cognitive Radio Technology* (Electronics & Electrical). Elsevier Science, 2009. Available: https://books.google.com.pr/books?id=IJPtfRTHWD4C

[5] A. M. Wyglinski, M. Nekovee, and Y. T. Hou, *Cognitive Radio Communications and Networks*, 1st ed. Burlington, MA, USA: ELSEVIER, 2010.

[6] G. Elmasry, *Dynamic Spectrum Access Decisions*, 1st ed. Hoboken, NJ, USA: WILEY, 2021, ch. 2, pp. 15–30.

[7] H. Touati, A. Chriki, H. Snoussi, and F. Kamoun, "Cognitive radio and dynamic TDMA for efficient UAVs swarm communications," *Computer Networks*, vol. 196, p. 108264, 2021.

[8] H. Zhang, X. Da, H. Hu, L. Ni, and Y. Pan, "Spectrum efficiency optimization for UAV-based cognitive radio network," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[9] J. Wu, R. Liu, and S. Zhao, "The cognitive communication based anti-jamming method for UAV swarm communication," in *IOP Conference Series: Materials Science and Engineering*, no. 1. IOP Publishing, 2020, vol. 799, p. 012043.

[10] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius, and R. Bianchi, "An empirically based path loss model for wireless channels in suburban environments," *IEEE Journal on selected areas in communications*, vol. 17, no. 7, pp. 1205–1211, 1999.

[11] C. Phillips, D. Sicker, and D. Grunwald, "A survey of wireless path loss prediction and coverage mapping methods," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 255–270, 2012.

[12] V. Abhayawardhana, I. Wassell, D. Crosby, M. Sellars, and M. Brown, "Comparison of empirical propagation path loss models for fixed wireless access systems," in *2005 IEEE 61st Vehicular Technology Conference*. IEEE, 2005, vol. 1, pp. 73–77.

[13] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design*. John Wiley Sons, 1998, ch. 4, pp. 100–125.

[14] J. Seybold, *Introduction to RF Propagation*, 1st ed. Hoboken, NJ, USA: WILEY, 2005.

[15] T. T. Ha, *Theory and Design of Digital Communications Systems*, 1st ed. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press, 2011.

[16] D. S. W. Ler, Lecture, Apr. 2021.

[17] A. K. P. N. Tan, M. Steinback, *Introduction to Data Mining*, 2nd ed. 330 Hudson Street, NY NY 10013: Pearson, 2019, ch. 7, p. 874.

[18] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2018, [Online]. Available: http://neuralnetworksanddeeplearning.com/

[19] H.-J. Song, H.-K. Kim, J.-D. Kim, C.-Y. Park, and Y.-S. Kim, "Inter-sentence segmentation of youtube subtitles using long-short term memory (lstm)," *Applied Sciences*, vol. 9, no. 7, 2019. Available: https://www.mdpi.com/2076-3417/9/7/1504

[20] N. O'Donoughue, *Emitter Detection and Geolocation for Electronic Warfare*, 1st ed. Norwood, MA, USA: Artech House, 2020.

[21] J. Modestino and S. Mui, "Convolutional code performance in the rician fading channel," *IEEE transactions on communications*, vol. 24, no. 6, pp. 592–606, 1976.

[22] R. He, Z. Zhong, B. Ai, J. Ding, and K. Guan, "Analysis of the relation between fresnel zone and path loss exponent based on two-ray model," *IEEE antennas and wireless propagation letters*, vol. 11, pp. 208–211, 2012.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California