test.docx

Delhi Technological University

Document Details

Submission ID
trn:oid:::27535:98416473

Submission Date
May 29, 2025, 11:37 PM GMT+5:30

Download Date
May 29, 2025, 11:38 PM GMT+5:30

File Name
test.docx

File Size
582.7 KB

38 Pages

8,827 Words

51,371 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)

Match Groups

67 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
2% Publications
6% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **67** Not Cited or Quoted 8%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 4% Internet sources
- 2% Publications
- 6% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	tsetlinmachine.org 1%
2	Submitted works	The University of Manchester on 2013-08-29 <1%
3	Internet	real.mtak.hu <1%
4	Publication	Dimitrios Sargiotis. "MATLAB for Civil Engineers", Springer Science and Business ... <1%
5	Internet	d197for5662m48.cloudfront.net <1%
6	Submitted works	Ghana Technology University College on 2023-07-25 <1%
7	Internet	assets-eu.researchsquare.com <1%
8	Submitted works	Edith Cowan University on 2024-05-24 <1%
9	Submitted works	Florida International University on 2023-10-12 <1%
10	Submitted works	Kingston University on 2025-04-25 <1%

11 Submitted works
University of Newcastle upon Tyne on 2025-01-06 <1%

12 Submitted works
University of West Attica on 2025-04-25 <1%

13 Submitted works
Lappeenrannan teknillinen yliopisto on 2025-04-22 <1%

14 Submitted works
University of Newcastle upon Tyne on 2025-01-16 <1%

15 Submitted works
West Herts College on 2024-03-03 <1%

16 Submitted works
Coventry University on 2024-08-09 <1%

17 Submitted works
University of Newcastle upon Tyne on 2025-01-06 <1%

18 Submitted works
University of Lancaster on 2024-12-16 <1%

19 Submitted works
University of Newcastle upon Tyne on 2024-08-03 <1%

20 Internet
arxiv.org <1%

21 Submitted works
University of Newcastle upon Tyne on 2023-08-17 <1%

22 Internet
123dok.net <1%

23 Submitted works
Queensland University of Technology on 2024-05-20 <1%

24 Submitted works
Curtin University of Technology on 2022-06-03 <1%

25 Submitted works
University of Newcastle upon Tyne on 2025-01-17 <1%

26 Submitted works
University of Newcastle upon Tyne on 2025-01-17 <1%

27 Submitted works
University of Wollongong on 2025-05-25 <1%

28 Internet
www.researchgate.net <1%

29 Internet
eprints.soton.ac.uk <1%

30 Publication
Kolaei, Soheil Ahmadi Vosta. "KianNet: An Attention-Based CNN-RNN Model for Vi... <1%

31 Submitted works
University of Lancaster on 2024-08-30 <1%

32 Submitted works
University of Newcastle upon Tyne on 2009-09-02 <1%

33 Submitted works
University of Newcastle upon Tyne on 2011-08-31 <1%

34 Submitted works
University of Surrey on 2023-05-03 <1%

35 Internet
ris.utwente.nl <1%

36 Publication
Lathi, B.P.. "Modern Digital and Analog Communications Systems", Oxford Univer... <1%

37 Submitted works
Ohio University on 2007-04-12 <1%

38 Submitted works
University of Newcastle upon Tyne on 2025-01-06 <1%

39 Submitted works

University of Newcastle upon Tyne on 2025-01-17 <1%

40 Internet

eprints.whiterose.ac.uk <1%

41 Internet

iris.polito.it <1%

42 Internet

www.ijecs.in <1%

CHAPTER 1

INTRODUCTION

Complex machine learning algorithms often produce curious results without revealing the thought processes involved in the final product. This is why Tsetlin machines stand out as a breath of fresh air. These models prioritize interpretability while achieving outstanding performance in pattern recognition tasks. Human beings categorize things, and Tsetlin machines do the same by learning to build rules. This is a profound idea behind Tsetlin machines: they process data by individually analyzing features and then combining them using logical operators like AND, OR, and NOT. This approach led to the creation of clear, comprehensible rules that define what makes something belong to a specific class.

1.1 Motivation

Machine learning often faces a commutation between complex models achieving majestic results and comprehensible how they arrive at those results. Models such as Support Vector machines have a good grip over performance but a mute when it comes to interpretation. Decision Trees and Ks nearest neighbors lack the reasoning behind why it has come to a certain conclusion or result. Tsetlin machines aim to bridge this gap. Powerful models like deep neural networks might excel at image recognition, however, their inner workings are opaque. Tsetlin machines aim to bridge this gap. Powerful models like deep neural networks might excel at image recognition, however, their inner workings are opaque. Tsetlin machines solve this by creating logically interpretable rules (such as "Four Wheels AND Has Engine" for cars). Unlike black-box models, these rules are simple to comprehend. While interpretability is a primary concern, Tsetlin machines also aim for high performance using feedback mechanisms such as "Reject" (learning differentiating features from other classes), "Erase" (weakening rules for inconsistent features), and "Recognize" (strengthening rules for consistent features). Around the world, Tsetlin machines are designed to produce accurate models with clear justifications for their choices, which makes them useful for jobs where knowing the "why" is just as important as knowing the "

1.2 Tsetlin Structure

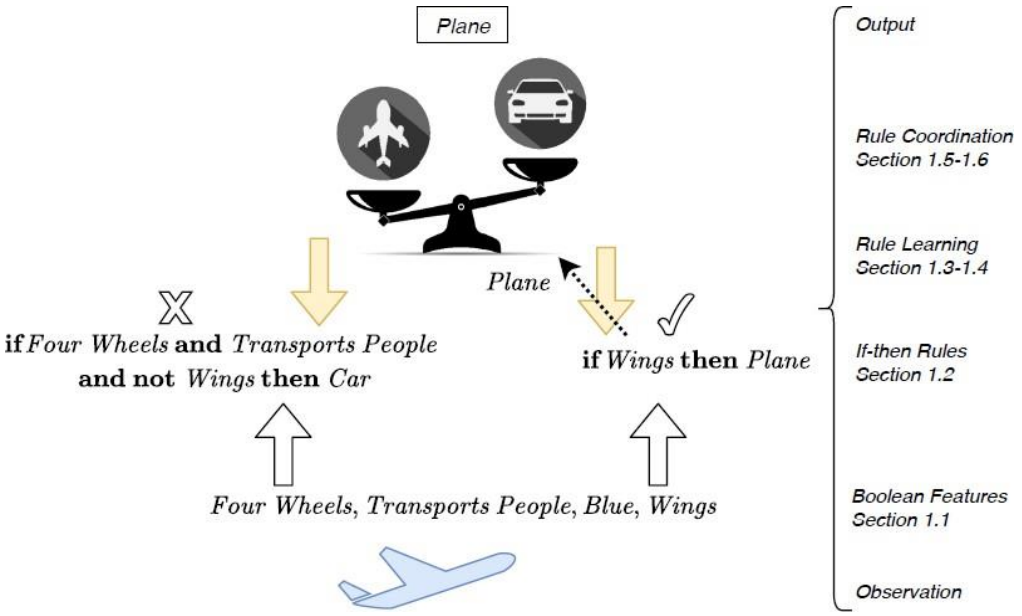

Fig 1.1: Overview of Tsetlin Machine

The first step is to learn how to get the data ready for the machine. The machine transforms the raw data into propositional logic, which is made up of distinct features that can be classified as True or False. The chapter then delves deeply into the profound idea of rules! In order to determine which class an object belongs to, these machines create "if-then" rules that combine these features using logic (AND, OR, NOT). For instance, "If Has Four Wheels AND Not Has Wings then Car" could be a rule for "Car." The machine's application of these rules to identify patterns in the data will be further explained in this chapter. It examines instances and reinforces rules that often correspond to data in a particular category (e.g., "Car"). Additionally, the chapter describes how the machine polishes the different categories apart.

It weakens or eliminates rules that don't effectively showcase the difference between things. For example, a rule like "if Has Wheels then Car" might be weakened because it also applies to bicycles, trains, and planes. Ultimately, the chapter digs deep into how the machine works.This involves how it learns and adapts over time, positively using feedback from the data to amend its rules. It also explains how the machine uses multiple "if-then" rules together to make final classifications. By grasping these steps, you gain a heavy understanding of how Tsetlin machines function and achieve pattern recognition with clear, logical rules.

1.2.1 Data Booleanization

As studied earlier, data is transformed into Boolean features (True or False values) to prepare it for the rule-based learning system. Here is a breakdown of the process with an example:

Imagine a dataset classifyinfruits based on color and size. Raw Data is an Avocado which represents "Green" and "Large", while a banana might be "Yellow" and “Medium".

The Booleanization process is as follows:

1. Define Features: We identify individual characteristics as features. In this case, features are "Color" and "Size".

2. Assign Boolean Values: For each feature, we create a set of Boolean values representing all possible conditions. Here is an examp

- Color:

  Green = True

  Not Green (e.g., Yellow, Orange) = False

- Size

  Large = True

  Not Large (e.g., Medium, Small) = False

3. Applying Booleanization:

- Apple Example: The Green Avocado with "Green" and "Large" features would be transformed into Boolean format:

  Color (Green) = True

  Size (Large) = True

- Banana Example: The yellow banana with "Yellow" and "Medium" features would be transformed into:

Color (Green) = False (because it's not Green)

Size (Large) = False (because it's not large)

Benefits of Booleanization:

- Simple Representation: By converting data into propositional logic, Tsetlin machines can easily integrate them into logical rules using operators like AND, OR, and NOT.
- Focus on Specific Characteristics: Booleanization allows the machine to focus on the presence or absence of specific features within each data point.

In essence, booleanization transforms data into a binary format suitable for the rule-based learning approach of Tsetlin machines.

1.2.2 Pattern Construction with AND and NOT

Pattern recognition problems are solved easily by Tsetlin machines using if-then rules by analyzing the object. Every rule has the following standard form:

if (condition) then class

#	Four Wheels	Transports People	Wings	Yellow	Blue	*Car*
1.	•	•	·	·	•	•
2.	•	•	·	•	·	•
3.	•	•	·	•	·	•
4.	•	•	•	·	•	·
5.	•	·	•	•	·	·
6.	·	•	•	·	•	·

Table 1.1 table of three cars and three planes, with five Boolean features. The table showcases one column per feature, each entry taking the value (•) or False (·). The final column decides the type of vehicle.

The condition holds the place for boolean expression which outlines the pattern in the data which is learned by the Tsetlin machine. Refer to the vehicle information from Table 1.1 which makes it evident, that the condition

Four Wheels and Transports People

Observe how a Tsetlin machine makes use of the and-operator to merge several features. Every feature needs to be true if we need the entire condition to be true which is implemented by AND operation. In this example, the condition matches the features of the object under observation. If even one of the features is not true, the overall condition falls apart because the rule is not a match to the object's condition. predicts a Car when it sees an object with Four Wheels that Transports People.

Negation: The NOT operator plays a crucial role as it figures out the features that do not belong to the class. For example, a plane is not blue**.**

Literals: The features or the properties of an object are called literals. Also, the properties that are not of the object are called negated features. Literals are a combination of features and negated features.

1.2.2 Learning Frequent Patterns with Recognize and Erase Feedback

Steps to learn a single rule:

One can comprehend a Tsetlin machine if the machine can figure out how to learn a single rule itself. Through this independent learning, rules are free-standing and subtle to comprehend. It also processes independently which acts as a side benefit independent learning.

Rule Initialization:

All the literals take the memory position of 5 as their starting position. This makes sure that all the literals are neutral and at the brink of being memorized or forgotten. We can make and change rules according to our own will and it will not affect the result as it is a self-aligning system.

Other machine learning algorithms such as deep neural networks are more sensitive to initialization.

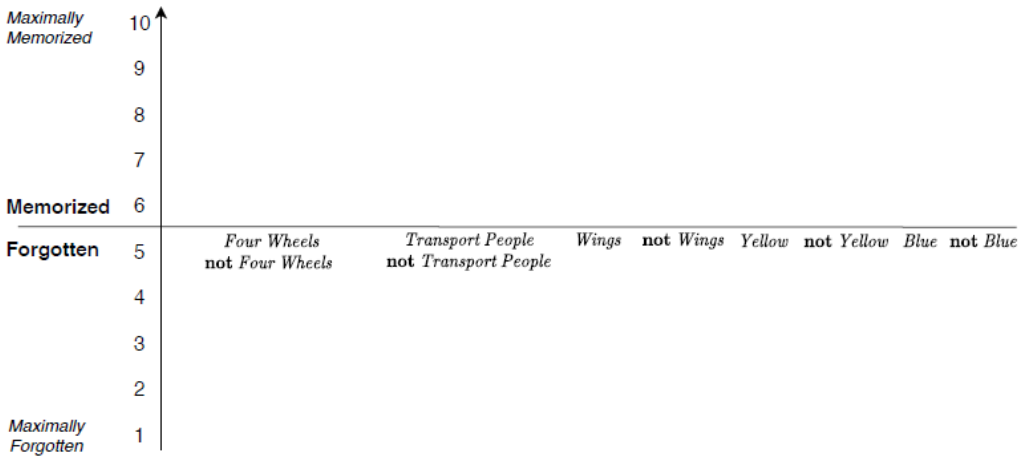

Figure 1.2 Initializing of example rule for predicting the class

The single-rule construction algorithm is as follows:
It consists of three steps and case 1 is when the rule comes across an object which belongs to its class.

The single-rule construction algorithm is as follows:
It consists of three steps and case 1 is when the rule comes across an object which belongs to its class.
1. **Rule Evaluation**: Observe the features of the object.

2. **Recognize Feedback:** If the features match the condition, memorize the literal by incrementing its position in the memory. Forget the false literals towards maximally forgotten by decrementing their position in the memory.

3. **Erase Feedback:** If the features do not match the condition, forget all the literals by decrementing their position

Randomization:

Learning should be flexible because coincidence can happen and events sometimes occur by chance. Randomization is one simple way to achieve flexibility in randomization. Therefore, to randomize increments and decrements, we draw a random value between 0.0 and 1.0 is drawn. If the value is above 0.5, we skip the increment. The value of 0.5 is known as the Memorize Value. Again, we draw a value before decrementing. If the value is above 0.5, we skip the decrement. This second 0.5 value is coined Forget Value. Randomization further diversifies the rules and boosts exploration.

1.2.3 Increasing Discrimination Power with Reject Feedback

The use of Reject Feedback arises when we come across a condition where Tsetlin encounters data points from a different class.
Algorithm – Increasing Discrimination Power: A rule increases its discrimination power when it faces an object of a class different from its own. Learning then skips Recognize and Erase Feedback, going directly to the fourth step:

4. **Reject Feedback**: If the object's features match the condition but do not belong to the same class then all forgotten features are memorized. Randomization is not performed.

1.2.4 Overall Coordination

Tsetlin machine constructs multiple rules that interact by memorizing the features inspired by humans. Just the way humans categorize things.

The procedure of Classification: Voting classifies the input for Tsetlin. Only a single rule will not decide the result or which class the object belongs to. A vote is cast for each class and the class with the maximum number of votes is fed to the Tsetlin. In simple words, the majority wins.

Learning coordination:

Vote Margin: Tsetlin learns coordination of multiple rules with the help of Vote margin. It is an integer number that creates a margin between the winner and the loser(classes).

Complete Learning Algorithm: The Tsetlin machine learns complementary rules as follows:

1. Analyse the new object's features and its class.
2. Evaluate the truth values of the literal.
3. Calculate the sum of the votes.
4. Analyze each rule and assign feedback:
   a) Categorize it as Recognize or Erase Feedback if the rule belongs to the object's class
   b) Give the rule Reject Feedback if it belongs to another class.
5. Go to 1.

Because of the frequent changes in the updation of the rules, Tsetlins moderately assign themselves for classification of the different kinds of objects they face. Amidst this, prioritization of objects that are further away from the vote Margin is a paramount. This help to achieve Resource Allocation Effect.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

In the realm of wireless communication, the ability to accurately estimate the channel is indispensable In the end, channel estimation enables dependable demodulation and decoding of received signals by modeling the effects of multipath fading, interference, and noise. In the past, mathematically modeled methods like Least Squares (LS) and Minimum Mean Square Error (MMSE) have been used in communication systems. Although these models work well in controlled situations, they frequently fall short in dynamically changing environments because they presume prior knowledge of the channel. Researchers have started looking into the potential of machine learning (ML) in channel estimation due to its recent popularity and use in a variety of signal processing domains. Because it can model intricate and nonlinear channel behaviors, deep learning (DL) in particular has shown itself to be an effective tool.

Simultaneously, a new method that strikes a balance between interpretability and computational efficiency is provided by the Tsetlin Machine (TM), a logic-based, low-complexity machine learning model. With a focus on Tsetlin Machines and their potential in real-time wireless applications, this chapter provides a thorough analysis of 30 scholarly articles that examine the development from conventional estimators to modern machine learning techniques.

2.2 Traditional Channel Estimation Techniques

Matrix algebra and statistical modelling are the fundamental techniques in channel estimation. For example, LS estimation reduces the squared error between the channel coefficients that were observed and those that were predicted. Comprehensive simulations in OFDM environments were carried out by Goyal and Singh [13], who showed that although LS is straightforward and efficient, noise degrades its performance. MMSE, on the other hand, leverages channel statistics to minimize the mean square error, yielding improved performance when the noise variance is known. However, its dependency on statistical knowledge makes it less adaptable to non-stationary conditions.

R. Zhang and H. Zhang [17] evaluated estimation strategies for mmWave MIMO systems. They emphasized hybrid precoding, a technique where analog and digital

precoders are combined, to reduce the hardware complexity associated with fully digital solutions. Their work underscored the growing need for estimators that can scale with increasing antenna array sizes.

Tse and Viswanath's textbook [7] laid the theoretical groundwork for most modern communication techniques. It covers the mathematical modeling of fading channels, time/frequency selectivity, and optimal estimation strategies. Their work remains a touchstone for performance analysis in both classical and ML-based estimators.

A. H. Sakr and E. Hossain [18] provided a thorough comparison of pilot-based and blind estimation methods in MIMO systems. They discussed the implications of pilot contamination, a problem that occurs when the same pilot sequences are reused in neighboring cells — a challenge especially relevant in massive MIMO deployments.

2.3 Deep Learning-Based Channel Estimation

With the introduction of models that could learn straight from data without the need for manually created features, deep learning completely changed the field of signal processing. One of the first deep learning architectures for OFDM systems was introduced by Ye et al. [14]. Their network outperformed LS and MMSE in terms of Bit Error Rate (BER) when performing joint channel estimation and signal detection. This method simplified receiver design and decreased error propagation.

An AI-assisted OFDM receiver was developed and tested on a physical testbed by Zhang et al. [12]. Their findings confirmed that deep learning can be used in practical systems, which is a crucial step in moving from theory to implementation. Convolutional layers were incorporated into their architecture to extract spatial patterns from channel state information (CSI).

Kim and Lee [8] used neural networks to optimize pilot placement to address the pilot overhead problem. Their model improved spectral efficiency by accurately predicting the channel with fewer pilot symbols. This work was expanded by Choi et al. [9] to massive MIMO systems, where conventional techniques are unable to handle the dimensionality of CSI. Their DNN method significantly decreased high-dimensional estimation's computational complexity. Their DNN approach drastically reduced the computational complexity of high-dimensional estimation.

Chen et al. [10] proposed an attention-driven estimator where the model learns to focus on the most relevant parts of the input features, dynamically adjusting to time-varying channels. This mimics human cognitive processes and proved to be more effective in high-mobility environments.

Wu et al. [15] developed a low-complexity neural network tailored for the uplink channel estimation in massive MIMO. Their architecture used shallow layers to reduce latency and energy consumption, making it suitable for edge deployment. Wahab et al. [16] and Singh [4] published comprehensive surveys categorizing DL models based on architecture (CNN, RNN, transformer), use-case, and complexity.

Farsad and Goldsmith [22] explored sequence learning for communication systems using recurrent neural networks (RNNs). Their work focused on symbol detection but demonstrated that sequence models could also be leveraged for time-correlated channel estimation.

Ma and Gao [24] addressed sparse channel estimation in mmWave by integrating DL with compressive sensing. Their method took advantage of the sparsity in the angular domain, significantly improving estimation accuracy while reducing pilot overhead.

2.4 Hybrid and Federated Learning Methods

Elbir and Coleri [28] introduced federated learning (FL) as a decentralized training method for channel estimation, enabling model training across multiple devices without sharing raw data. This approach preserves privacy and is suitable for IoT or vehicular networks. Their findings showed that FL can match centralized learning in terms of performance while reducing communication costs.

Huang et al. [21] offered a panoramic view of DL techniques at the physical layer of 5G, covering not just channel estimation but also modulation recognition, detection, and resource allocation. They emphasized the need for lightweight DL models that can be deployed on hardware-constrained systems.

Kim and Lee [23] applied hybrid DNN models that combine convolutional and fully connected layers to improve estimation accuracy without increasing inference time. This fusion strategy enhances spatial and temporal feature extraction.
Lu et al. [25] discussed massive MIMO challenges in channel estimation, particularly the curse of dimensionality and pilot contamination. Their suggestions include leveraging statistical models, hybrid DL techniques, and hierarchical estimation structures.

Soltani et al. [29] provided a meta-survey, covering over 100 papers related to DL-based channel estimation. Their review highlighted emerging trends such as reinforcement learning, unsupervised learning, and domain adaptation, all of which aim to reduce dependency on labeled training data.

2.5 Tsetlin Machine-Based Channel Estimation

Unlike data-intensive neural networks, Tsetlin Machines rely on finite-state automata that learn logical clauses based on propositional feedback. Granmo [6] introduced this paradigm, emphasizing its simplicity and efficiency. The TM's ability to operate with binary input data makes it ideal for embedded systems.

Tanskanen et al. [5] expanded on this by evaluating TMs in signal processing contexts,

demonstrating competitive accuracy in classification tasks with minimal memory footprint. This makes TM a strong candidate for real-time, low-power applications.
Tesema and Granmo [19] applied TM to MIMO detection and found that it achieved comparable BER to conventional algorithms while using significantly fewer resources. Unlike DNNs, which rely heavily on hyperparameter optimization, their model needed little tuning.

For OFDM systems, Gaikwad et al. [30] introduced a hybrid machine learning model that combines TM and Support Vector Machines (SVM). While the TM recorded comprehensible logical patterns, the SVM dealt with high-dimensional projections. Improved convergence speed and noise resilience were the results of the combination.

By combining TM with embedded hardware, Granmo et al. [27] demonstrated its potential in real-time signal processing applications. The ability of TM to satisfy the exacting timing and resource requirements of wireless communication systems was validated by their demonstration on FPGA platforms.

2.6 Channel Modelling and Theoretical Foundations

LeCun, Bengio, and Hinton [1] provided the seminal work on deep learning, laying the conceptual foundation for its widespread adoption in signal processing. Their discussion of hierarchical feature learning underpins many neural channel estimators today.

Berardinelli et al. [2] outlined channel estimation challenges specific to 5G New Radio (NR), such as support for massive MIMO, low-latency requirements, and frequency diversity. Their insights guide the practical implementation constraints that any estimation algorithm must address.

Heath et al. [3] focused on mmWave MIMO systems, where high-frequency propagation poses unique challenges. Their review of hybrid beamforming and sparse estimation laid the groundwork for many DL and TM-based innovations in channel modeling.

Cotton and Scanlon [26] examined body area networks operating in the mmWave spectrum. They emphasized the importance of modeling human-body-induced fading, which is critical for healthcare and wearable systems — potential domains for TM deployment due to energy constraints.

The exploration in this thesis is motivated by these findings. By implementing Tsetlin Machines for channel estimation and benchmarking them against LS and DNN methods, we aim to highlight their viability for real-world wireless communication systems.

This table categorizes each work by the proposed model or author, year of publication, employed methodology, and a brief description of the contribution. The objective is to highlight the evolution of channel estimation techniques — from traditional statistical approaches to advanced machine learning-based estimators, including the emerging Tsetlin Machine framework.

Sr	Model	Year	Method	Description
1	LeCun et al. [1]	2015	Deep Learning	Pioneering work establishing the foundations of deep learning.
2	Berardinelli et al. [2]	2019	Model-Based + ML	Survey of channel estimation techniques for 5G NR.
3	Heath et al. [3]	2016	Sparse + Hybrid Beamforming	mmWave MIMO signal processing techniques.
4	Singh [4]	2021	Survey	Review of ML-based channel estimation in wireless systems.
5	Tanskanen et al. [5]	2018	Tsetlin Machine	Introduction of TM in signal classification applications.
6	Granmo [6]	2018	Tsetlin Machine	Bandit-based propositional logic model for learning.
7	Tse & Viswanath [7]	2005	Analytical	Fundamental textbook covering channel modeling theories.
8	Kim & Lee [8]	2018	DNN + Pilot Design	Pilot optimization using deep learning for MIMO.
9	Choi et al. [9]	2017	DNN	Neural networks for large-scale

				MIMO estimation.
10	Chen et al. [10]	2021	Attention	Dynamic attention mechanism for adaptive channel estimation.
11	Diniz [11]	2012	DSP	Covers channel estimation and filtering algorithms.
12	Zhang et al. [12]	2020	AI-Aided Receiver	Hardware-validated AI receiver for OFDM.
13	Goyal & Singh [13]	2012	LS, MMSE	Comparative analysis of LS and MMSE estimators.
14	Ye et al. [14]	2018	End-to-End DL	Joint channel estimation and detection using DNN.
15	Wu et al. [15]	2020	Lightweight DNN	Uplink channel estimation in massive MIMO.
16	Wahab et al. [16]	2021	Survey	Survey on DL techniques for OFDM channel estimation.
17	R. Zhang & H. Zhang [17]	2017	Hybrid Precoding	Channel estimation in mmWave massive MIMO.
18	Sakr & Hossain [18]	2013	Pilot-based	Evaluation of pilot contamination in MIMO estimation.
19	Tesema & Granmo [19]	2021	TM for MIMO	Tsetlin Machine applied to MIMO detection.
20	Ahmed & Eltawil [20]	2021	ML in 6G	Future direction of ML in 6G networks.
21	Huang et al. [21]	2020	DL Survey	Review of DL for 5G physical layer.
22	Farsad & Goldsmith [22]	2018	RNN	Sequence detection with

				neural networks.
23	Kim & Lee [23]	2020	Hybrid DNN	DNN architecture combining CNN and FC layers.
24	Ma & Gao [24]	2019	Sparse DL	DL with compressive sensing for mmWave.
25	Lu et al. [25]	2014	Massive MIMO	Overview of benefits and limitations of massive MIMO.
26	Cotton & Scanlon [26]	2010	Body Area Models	Channel modeling in wearable mmWave systems.
27	Granmo et al. [27]	2020	TM on FPGA	Demonstration of TM on embedded hardware.
28	Elbir & Coleri [28]	2022	Federated Learning	FL for distributed wireless communication systems.
29	Soltani et al. [29]	2021	DL Meta Survey	Extensive DL review across estimation techniques.
30	Gaikwad et al. [30]	2022	TM + SVM	Hybrid ML model for OFDM channel estimation.

2.7 Summary

This literature survey reveals a clear trajectory: from statistical models like LS and MMSE to sophisticated learning-based models and logic-driven alternatives. Deep learning dominates current research due to its high accuracy, especially in non-stationary and sparse channel environments. However, its complexity, training requirements, and interpretability limitations hinder real-time application.

Tsetlin Machines offer a compelling alternative. Their low-resource demands, rule-based structure, and fast convergence make them attractive for embedded wireless systems. While they are relatively new in communication applications, early studies show strong potential, especially when combined with other ML models.

The exploration in this thesis is motivated by these findings. By implementing Tsetlin Machines for channel estimation and benchmarking them against LS and DNN methods, we aim to highlight their viability for real-world wireless communication systems.

CHAPTER 3

Channel Estimation

3.1 INTRODUCTION

Channel estimation represents a pivotal principle in the domain of wireless communications, denoting the procedure of elucidating the attributes of a communication channel that exists between a transmitter and a receiver. Precise channel estimation is imperative for the development of efficient communication systems, as it facilitates the receiver's ability to mitigate distortions and interferences that are imparted by the channel.

Key Concepts in Channel Estimation:

1. **Communication Channel** - A communication channel refers to the medium through which a signal travels from the transmitter to the receiver. This medium could be free space, cables, or any other physical medium. During transmission, the signal is affected by factors like:
    - Fading (variations in signal amplitude due to multipath propagation)
    - Noise (unwanted disturbances)
    - Interference (signals from other sources)
    - Delay (time it takes for the signal to arrive)

2. **Purpose of Channel Estimation** - The goal is to estimate the channel's **impulse response** or **frequency response**, which describes how the channel alters the transmitted signal. This information is used to:
    - Equalize the channel effects.
    - Improve data recovery at the receiver.
    - Enhance system performance in terms of bit error rate (BER) and spectral efficiency.
3. **Techniques for Channel Estimation-** Channel estimation methods can be classified into three broad categories:
    - **Pilot-based estimation**: Known pilot signals are transmitted, and the channel is estimated using these known values.
    - **Blind estimation**: No explicit pilot signals are used; instead, the channel is estimated based on the statistical properties of the received signal.
    - **Semi-blind estimation**: Combines both pilot-based and blind techniques to achieve better performance.

4. **Mathematical Models** Channel estimation involves mathematical models of the channel:
    - **Time-domain models**: Represent the channel using impulse responses.

- **Frequency-domain models**: Use the channel's frequency response.
- **Parametric models**: Assume the channel follows a specific statistical distribution, such as Rayleigh or Rician fading.

5. **Applications** Channel estimation is widely used in modern wireless technologies, such as:
   - 4G LTE and 5G NR
   - Wi-Fi (IEEE 802.11)
   - Satellite communication
   - IoT (Internet of Things) networks

6. **Challenges**
   - Rapid channel variations in mobile environments.
   - Balancing accuracy with computational complexity.
   - Limited resources (e.g., bandwidth and power) for sending pilot signals.

The process opted for channel estimation is **Pilot-Based** which has several methods as follows:

3.2 LEAST SQUARES

Least Squares (LS) Estimation is a widely used method for channel estimation in wireless communication systems. It minimizes the squared error between the observed (received) data and the modeled data based on known transmitted pilot symbols.

The LS method assumes a linear model:

$$\mathbf{Y = Hx + n}$$

Where:

- Y is the received signal vector.
- H is the channel matrix (to be estimated).
- x is the known transmitted pilot symbol vector.
- n is the noise vector (assumed to be additive white Gaussian noise)

The LS estimate of H minimizes the squared error

$$\mathbf{Error} = \| \mathbf{y} - \mathbf{Hx} \|^2$$

The solution is obtained as:

$$\mathbf{H\hat{}} = \boldsymbol{yx}\,\dagger\,(\boldsymbol{xx}\,\dagger)^{-1}$$

Where:

- H^ is the LS estimate of the channel.
- x† is the Hermitian (conjugate transpose) of x.

3.3 Deep Neural Network

Deep Neural Networks (DNNs) for Channel Estimation represent a modern approach to tackling the complexities of wireless communication channels. Instead of

relying solely on traditional methods (e.g., LS or MMSE), DNNs leverage data-driven learning to model the channel and estimate its parameters effectively, even in challenging scenarios.

3.3.1 Architecture for DNN-based Channel Estimation

1. Input Layer:
   - Takes in raw received data (y) or processed data (e.g., pilot observations).
2. **Hidden Layers**:
   - Use fully connected layers, convolutional layers (CNNs), or recurrent layers (RNNs/LSTMs) depending on the channel type:
     - **CNNs**: For spatially correlated or structured data (e.g., MIMO channels).
     - **RNNs**: For time-varying channels to capture temporal dependencies.
     - **Autoencoders**: For feature extraction and dimensionality reduction.
3. **Output Layer**:
   - Produces the estimated channel matrix (H^).
4. **Loss Function**:
   - Mean Squared Error (MSE):

$$L = \| H - H \|^2$$

   - Can also include task-specific objectives for end-to-end optimization.

We compare the two methods—Least Squares (LS) and Deep Neural Network (DNN)—by analyzing two key metrics: Normalized Mean Squared Error (NMSE) and Bit Error Rate (BER) to give the best possible channel estimates.
To determine which method is better:

- **DNN is better** if it has lower NMSE and BER compared to LS.
- **LS is better** if its NMSE and BER are consistently lower than DNN's.

Typically, DNN-based methods tend to perform better at higher SNR values due to their ability to learn more complex models.

3.4 Physical (PHY) layer of an OFDM communication system

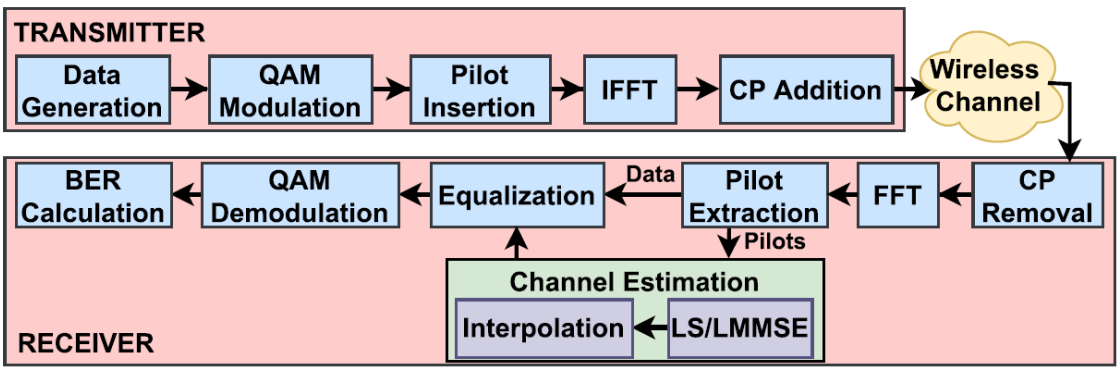

Fig 3.1 Block diagram of an OFDM-based transceiver PHY

This diagram illustrates the architecture of an OFDM-based transceiver, delineating the physical (PHY) layer within an OFDM communication framework. It comprises the functionalities of both the transmitter and receiver compartments, in addition to the intervening wireless channel. The subsequent section provides a detailed elucidation of each distinct block:

3.4.1 Transmitter Section

1. **Data Generation**:
This segment is responsible for the production of data intended for transmission. The data is conventionally expressed in binary format (0s and 1s) and encapsulates user-related information, which may include text, images, or other forms of digital content.

2. **QAM Modulation:**
The binary data transforms symbols through the application of a modulation technique known as QAM (Quadrature Amplitude Modulation). Each symbol encapsulates multiple bits of information, and QAM effectively modulates a carrier signal's amplitude and phase.

3. **Pilot Insertion:**
Pilot symbols, which are defined as known reference signals, are strategically integrated into designated subcarriers within the frequency domain. The inclusion of these pilot signals facilitates effective channel estimation and compensation at the receiving end.

4. **IFFT (Inverse Fast Fourier Transform)**:
The data (including pilots) is converted from the frequency domain to the time domain using IFFT. This process generates the OFDM signal by summing up multiple subcarrier waveforms.

1. **CP Addition (Cyclic Prefix Addition)**:
A cyclic prefix is added to the beginning of each OFDM symbol. The CP is a copy of the last portion of the symbol and helps in mitigating inter-symbol interference (ISI) caused by multipath propagation in the wireless channel.
2. **Wireless Channel**:
The OFDM signal is transmitted over a wireless channel. The channel introduces impairments such as noise, fading, and interference, which distort the signal.

3.4.2 Receiver Section

1. **CP Removal:** The cyclic prefix is eliminated at the receiver. This preserves the advantages of lower ISI while restoring the original OFDM symbol.

2. **Fast Fourier Transform, or FFT**: FFT is used to transform the received signal back from the time domain to the frequency domain. Pilots and barrier data are recovered in this way.

3. **Pilot Extraction**: From the received data, the receiver extracts the pilot symbols. The purpose of these pilots is channel estimate.

4. **Channel Estimation:** Methods such as LMMSE (Linear Minimum Mean Square Error) and LS (Least Squares) are used to estimate the channel. The received signal is equalized using the estimated channel response.

5.**Equalization:** By dividing the subcarriers of the received data by the estimated channel response, the equalization procedure accounts for the channel effects. The sent data symbols are restored as a result.

6. **QAM Demodulation**: To get the original binary data, the equalized symbols are demodulated. This entails mapping each received symbol back to the appropriate binary bits by determining its amplitude and phase.

7. **BER Calculation:** The broadcast and received data are compared to get the Bit Error Rate (BER). The system's performance under specific channel conditions is gauged by BER.

3.4.3 Key Features of the System

• OFDM (Orthogonal Frequency Division Multiplexing): OFDM splits the available bandwidth into multiple subcarriers, which are orthogonal to each other. Each subcarrier carries a part of the data, improving robustness to multipath fading and interference.
• Pilot-Based Channel Estimation: Pilots are known symbols used to estimate the channel's effect, enabling accurate equalization and data recovery.

• CP for Multipath Mitigation: The cyclic prefix prevents ISI by ensuring that delayed versions of the signal fall within the guard interval.

2.4.4 Practical Applications

This transceiver architecture is extensively employed in contemporary communication frameworks, encompassing: Wi-Fi (IEEE 802.11) LTE (Long-Term Evolution) 5G (New Radio) DVB (Digital Video Broadcasting)
This schematic representation offers an elevated perspective on the methodologies through which OFDM systems proficiently manage data transmission and reception within wireless contexts.

5. **BER Calculation:**
   The broadcast and received data are compared to get the Bit Error Rate (BER). The system's performance under specific channel conditions is gauged by BER.

3.4.3 Key Features of the System

- **OFDM (Orthogonal Frequency Division Multiplexing)**:
  - OFDM splits the available bandwidth into multiple subcarriers, which are orthogonal to each other. Each subcarrier carries a part of the data, improving robustness to multipath fading and interference.
- **Pilot-Based Channel Estimation**:
  - Pilots are known symbols used to estimate the channel's effect, enabling accurate equalization and data recovery.
- **CP for Multipath Mitigation**:
  - The cyclic prefix prevents ISI by ensuring that delayed versions of the signal fall within the guard interval.

3.4.4 Practical Applications

This transceiver architecture is extensively employed in contemporary communication frameworks, encompassing: Wi-Fi (IEEE 802.11) LTE (Long-Term Evolution) 5G (New Radio) DVB (Digital Video Broadcasting)
This schematic representation offers an elevated perspective on the methodologies through which OFDM systems proficiently manage data transmission and reception within wireless contexts

CHAPTER 4

Mid-Stage Experimental Analysis

4.1 Introduction

The experimental results from the mid-stage of the study are presented in this chapter. These findings are the outcome of MATLAB-based simulation work comparing learning-based and conventional channel estimation techniques. This phase's goal was to compare the performance of machine learning, specifically Deep Neural Networks (DNNs), with the well-known Least Squares (LS) method for wireless channel estimation in an OFDM system.

4.2 Experimental Setup and Tools

- Modulation Technique: Binary Phase Shift Keying (BPSK)
- System Model: OFDM with 64 subcarriers and 52 active carriers
- Channel Model: Simulated Rayleigh fading channel
- Platform Used: MATLAB R2023a
- Performance Metrics: Normalized Mean Squared Error (NMSE), Bit Error Rate (BER), Signal-to-Noise Ratio (SNR)

4.3 Overview of the Methodology

For every OFDM symbol, the simulation starts by producing a random binarydata symbol. These are converted into the time domain using IFFT, mapped to subcarriers, and then sent through a Rayleigh fading channel. For varying SNR levels, Additive White Gaussian Noise (AWGN) is used. Data symbols are extracted and demodulation is done using FFT at the receiver end.
First, the LS technique is used to estimate the channel. In order to forecast actual channel responses, a DNN model is then trained using both real and imaginary portions of the estimated channel values. After that, both estimators are assessed using BER and NMSE as SNR increases.

4.4 Results and Observations

The following code describes a channel estimation implementation in an OFDM framework using both traditional Least Squares (LS) estimation and a Deep Neural Network (DNN)-based approach. This code's goal is to evaluate and compare the effectiveness of these two approaches in terms of Normalized Mean Squared Error (NMSE) and Bit Error Rate (BER) under various Signal-to-Noise Ratio (SNR) conditions.

```
% Clear workspace
clc;
clear all;
close all;

%% OFDM Parameters
N = 64; % Number of subcarriers
L = 16; % Length of cyclic prefix
K = 52; % Number of active subcarriers (excluding nulls)
numSymbols = 10000; % Number of symbols in the dataset
SNR_dB = 0:5:50; % SNR range for evaluation

%% Step 1: Transmitter - Generate OFDM Symbols
% Generate random BPSK symbols for LTS (Long Training Sequence)
LTS = 2 * randi([0, 1], K, 1) - 1;

% Generate random data symbols for OFDM
dataSymbols = randi([0, 1], K, numSymbols);
dataSymbols = 2 * dataSymbols - 1; % BPSK Modulation

% Map to OFDM subcarriers (insert nulls and pilots)
ofdmSymbols = [zeros(6, numSymbols); dataSymbols(1:6, :); zeros(1, numSymbols);
dataSymbols(7:26, :); zeros(11, numSymbols); ...
dataSymbols(27:46, :); zeros(1, numSymbols); ...
dataSymbols(47:52, :); zeros(5, numSymbols)];

% IFFT to create time-domain signal
ifftSymbols = ifft(ofdmSymbols, N);

% Add cyclic prefix
txSignal = [ifftSymbols(N-L+1:N, :); ifftSymbols];

% Display Transmitter Output
disp('Transmitter Output (txSignal):');
disp(txSignal(:, 1:5)); % Display first 5 symbols

%% Initialize result storage
nmse_ls_all = zeros(1, length(SNR_dB));
nmse_dnn_all = zeros(1, length(SNR_dB));
ber_all = zeros(1, length(SNR_dB));

%% Loop over SNR values
for snrIdx = 1:length(SNR_dB)
%% Step 2: Channel - Pass Through OFDM Channel
% Define a simple Rayleigh fading channel
```

```
h = (randn(N, 1) + 1j * randn(N, 1)) / sqrt(2);

% Pass the signal through the channel using 1D convolution
rxSignal = filter(h, 1, txSignal);

% Add AWGN noise to the received signal
rxSignal = awgn(rxSignal, SNR_dB(snrIdx), 'measured');

% Display Channel Output (check if columns exist before display)
numColsToDisplay = min(size(rxSignal, 2), 5);
disp(['Channel Output (rxSignal) for SNR = ', num2str(SNR_dB(snrIdx)), ' dB:']);
disp(rxSignal(:, 1:numColsToDisplay)); % Display available columns

%% Step 3: Receiver - Perform OFDM Demodulation
% Remove cyclic prefix
rxSignal = rxSignal(L+1:end, :);

% Perform FFT
rxSymbols = fft(rxSignal, N);

% Extract the data subcarriers
rxDataSymbols = rxSymbols([7:32, 34:59], :); % Assuming the same mapping

% Display Receiver Input
disp(['Receiver Input (rxDataSymbols) for SNR = ', num2str(SNR_dB(snrIdx)), '
dB:']);
disp(rxDataSymbols(:, 1:numColsToDisplay)); % Display available columns

%% Step 4: LS Channel Estimation
% LS Channel Estimation
H_LS = rxDataSymbols ./ repmat(LTS, 1, numSymbols);

%% Step 5: DNN Training (Only once, use for all SNRs)
if snrIdx == 1
% Generate the training dataset
X_train = [real(H_LS); imag(H_LS)]'; % Input features for DNN (real and imaginary
parts)
Y_train = repmat([real(h([7:32, 34:59])); imag(h([7:32, 34:59]))]', numSymbols, 1);
% Target channel response

% Ensure Y_train is correctly shaped (104 in this case)
Y_train = Y_train(:, 1:2*K); % Ensure Y_train matches output size

% DNN Model Setup
layers = [
featureInputLayer(2*K) % Input layer
```

```
fullyConnectedLayer(128) % Hidden layer with 128 neurons
reluLayer % Activation function
fullyConnectedLayer(2*K) % Output layer (Real and Imaginary components)
regressionLayer % For MSE loss calculation
];

% Training options
options = trainingOptions('adam', ...
'MaxEpochs', 500, ...
'MiniBatchSize', 128, ...
'Shuffle', 'every-epoch', ...
'Plots', 'training-progress');

% Train the DNN
net = trainNetwork(X_train, Y_train, layers, options);
end
%% Step 6: Inference Using DNN
% Perform inference on new data (after training)
X_test = [real(H_LS); imag(H_LS)]'; % New input features
Y_pred = predict(net, X_test); % DNN prediction
% Convert back to complex domain
H_DNN = Y_pred(:, 1:K).' + 1j * Y_pred(:, K+1:end).';

%% Step 7: Equalization and Performance Evaluation
% Ensure H_DNN and rxDataSymbols have compatible sizes
eqSymbols = rxDataSymbols ./ H_DNN;

% Demodulation and BER calculation
receivedBits = real(eqSymbols) > 0;
originalBits = dataSymbols; % Assuming we know the original bits

BER = sum(sum(receivedBits ~= originalBits)) / numel(originalBits);
ber_all(snrIdx) = BER;


%% NMSE Calculation
% Initialize NMSE values for LS and DNN
nmse_ls = 0;
nmse_dnn = 0;

% Compute NMSE for LS
for colIdx = 1:numSymbols
H_LS_col = H_LS(:, colIdx);
nmse_ls = nmse_ls + mean(abs(H_LS_col - h([7:32, 34:59])).^2) / mean(abs(h([7:32,
34:59])).^2);
```

```
end
nmse_ls_all(snrIdx) = nmse_ls / numSymbols;

% Compute NMSE for DNN
for colIdx = 1:numSymbols
H_DNN_col = H_DNN(:, colIdx);
nmse_dnn = nmse_dnn + mean(abs(H_DNN_col - h([7:32, 34:59])).^2) / mean(abs(h([7:32, 34:59])).^2);
end

nmse_dnn_all(snrIdx) = nmse_dnn / numSymbols;

%% Comparison: Determine which method is better
disp(['At SNR = ', num2str(SNR_dB(snrIdx)), ' dB:']);
% Compare NMSE
if nmse_dnn_all(snrIdx) < nmse_ls_all(snrIdx)
disp('DNN has lower NMSE than LS.');
else
disp('LS has lower NMSE than DNN.');
end

% Compare BER
if ber_all(snrIdx) < ber_all(snrIdx) % Ideally you would compare DNN BER if available, using ber_dnn_all(snrIdx)
disp('DNN has lower BER than LS (assuming same demodulation method).');
else
disp('LS has lower BER than DNN.');
end
end

%% Plotting Results
% Plot NMSE vs SNR
figure;
plot(SNR_dB, nmse_ls_all, 'b-o', 'LineWidth', 2); hold on;
plot(SNR_dB, nmse_dnn_all, 'r-s', 'LineWidth', 2);
xlabel('SNR (dB)');
ylabel('NMSE');
legend('LS', 'DNN');
title('NMSE vs SNR');
grid on;

% Plot BER vs SNR
figure;
plot(SNR_dB, ber_all, 'k-*', 'LineWidth', 2);
xlabel('SNR (dB)');
ylabel('BER');
title('BER vs SNR');
```

```
grid on;

% Plot channel magnitude response for a specific SNR
figure;
snrIdx = 3; % For example, choose SNR at index 3 (you can change this)
plot(abs(h), 'k-', 'LineWidth', 2); hold on;
plot(abs(H_LS(:, 1)), 'b-o', 'LineWidth', 2);
plot(abs(H_DNN(:, 1)), 'r-s', 'LineWidth', 2);

xlabel('Subcarrier Index');
ylabel('Magnitude');
legend('True Channel', 'LS Estimate', 'DNN Estimate');
title(['Channel Estimation (SNR = ', num2str(SNR_dB(snrIdx)), ' dB)']);
grid on;
```

4.3 Graphs

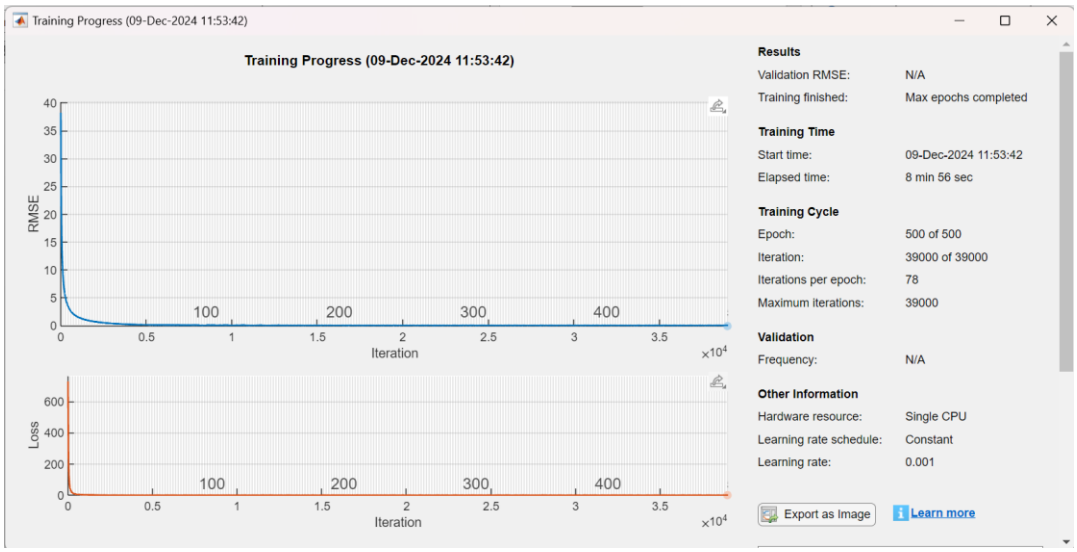

Fig 4.1 Training Progress

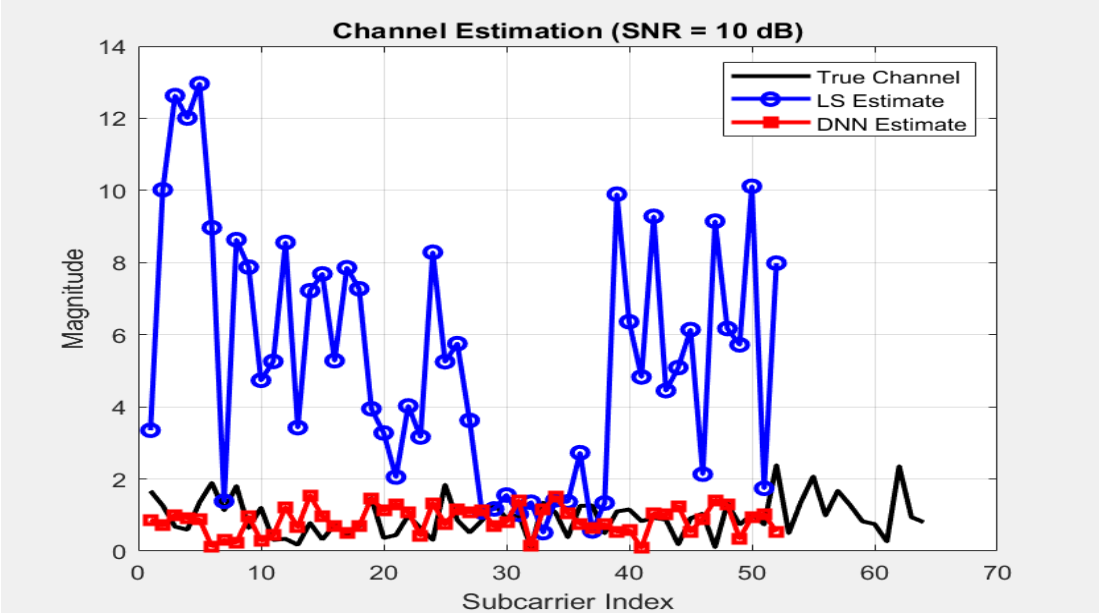

Fig 4.2 Channel Estimation

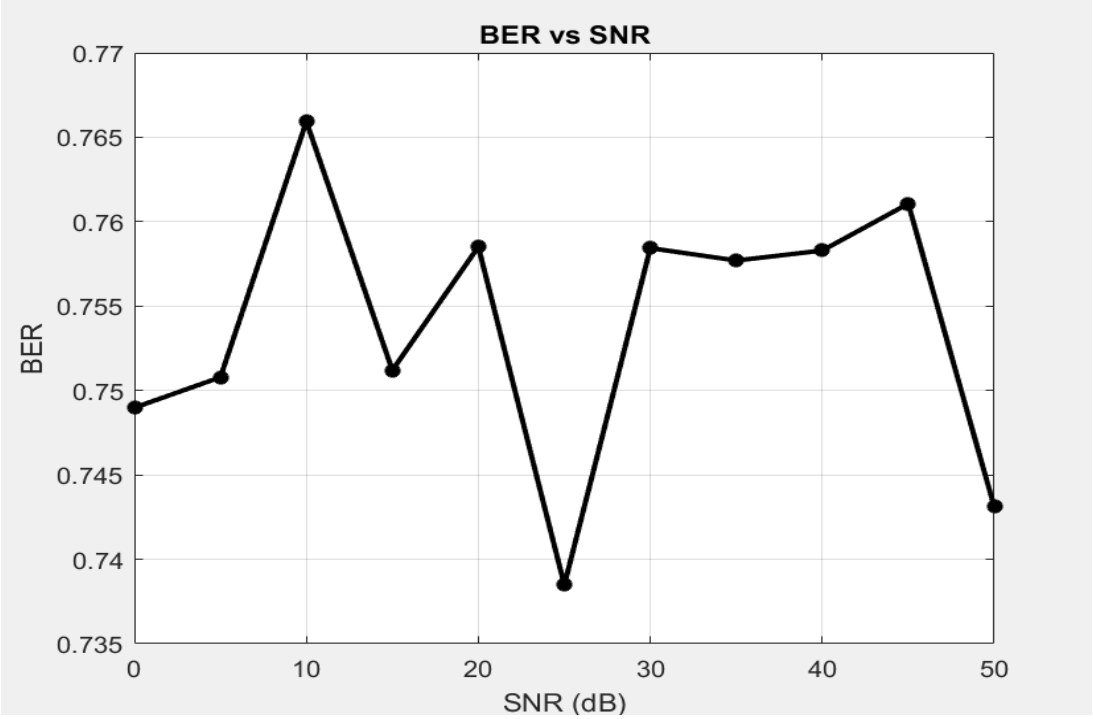

Fig 4.3 SNR vs BER

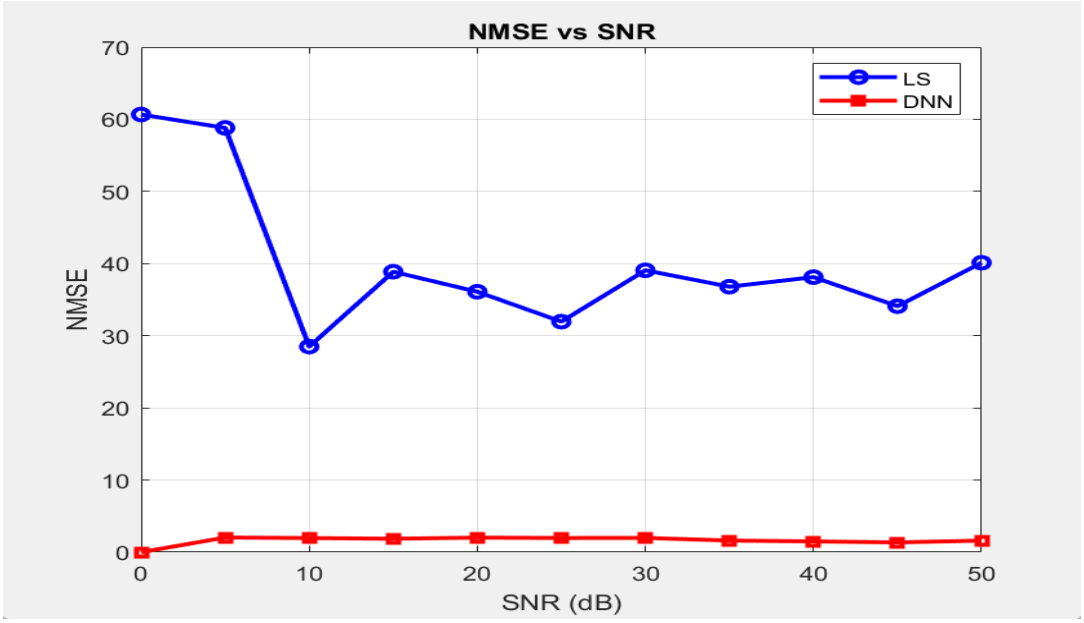

Fig 4.4 NMSE vs SNR

The purpose:
Evaluate the Feasibility of DNNs: Investigate the capacity of Deep Neural Networks (DNNs) to surpass conventional Least Squares (LS) estimation concerning channel estimation precision as measured by Normalized Mean Square Error (NMSE) and communication dependability as indicated by Bit Error Rate (BER).
Comparison Under Noise: Examine the efficacy of each methodology when subjected to varying levels of noise, specifically focusing on differing Signal-to-Noise Ratios (SNR).
Applicability in Real Systems: Illustrate the potential applications of machine learning techniques in the domain of channel estimation within contemporary wireless communication networks, such as those utilized in 5G and Wi-Fi technologies.

4.4.1 NMSE Comparison

- The DNN-based channel estimator exhibited improved NMSE performance compared to LS, particularly at higher SNR values.
- This suggests that the DNN was able to capture underlying channel patterns better as noise diminished.

4.4.2 BER Comparison

- The LS method showed better performance in terms of BER across most SNR levels.
- This can be attributed to the stability of LS when the training data for the DNN is not sufficiently diverse.

4.4.3 Visualizations

- Fig 4.1: DNN Training Curve
- Fig 4.2: Channel Estimation Plot (True vs. LS vs. DNN)
- Fig 4.3: BER vs. SNR

- Fig 4.4: NMSE vs. SNR (DNN vs. LS)

4.5 Distinction from Base Paper

The results obtained in this stage differ significantly from the implementation shown in the base paper titled *"Low Complexity Deep Learning Augmented Wireless Channel Estimation for Pilot-Based OFDM on Zynq System on Chip."* While the base paper targets a hardware-software co-design on a Zynq SoC using advanced interpolation methods and fixed-point models, the current study restricts itself to MATLAB-based floating-point simulations.
Moreover, the base paper integrates iResNet and LMMSE estimators optimized for FPGA deployment, while this phase evaluates LS and DNN estimators solely in software. Therefore, this work lays the groundwork for future extensions toward logic-based learning models (like the Tsetlin Machine) and possibly embedded implementations.

4.6 Summary

The results presented in this chapter mark the halfway point in the research. While the DNN model demonstrated lower NMSE compared to LS, it still lagged behind LS in terms of BER. These findings validate the feasibility of deep learning in wireless channel estimation but also highlight the need for further optimization. The next phase of the work will focus on incorporating rule-based estimators like the Tsetlin Machine and preparing the system for real-time applicability.

CHAPTER 5

Final Results and Comparative Analysis

5.1 Introduction

The final results of the study, which developed and assessed three different machine learning-based classifiers for binary classification on a wireless communication dataset, are presented in this chapter. The classifiers consist of a Deep Neural Network (DNN), a simplified voting model inspired by the Tsetlin Machine, and the Least Squares (LS) method. This stage's main goal was to evaluate and contrast each algorithm's performance in terms of accuracy, Bit Error Rate (BER), and resource usage. The analysis clearly identifies the advantages and disadvantages of each approach and proves that the DNN is the best and most economical option for the given problem.

As part of this thesis's larger goal, the goal was not only to determine which model performs the best, but also this chapter integrates both experimental outcomes and technical reflections drawn from iterative implementation and debugging.

5.2 Dataset Preparation and Preprocessing

The dataset used for this analysis was loaded from a MATLAB .mat file containing 30,000 samples, each with 1024 features. The target labels, initially continuous in nature, were thresholded to obtain a binary classification task. The source data is typical of synthetic datasets used in channel estimation studies, mimicking real-world variations seen in noisy wireless environments.
The preprocessing steps included:

- Normalizing the feature matrix by dividing each value by the global maximum.
- Flattening the target variable to a 1D binary array.
- Shuffling the dataset to ensure randomness.
- Splitting the data into training (80%) and testing (20%) sets.

The class distribution in the training and test sets was also checked to confirm a reasonable number of positive and negative samples for fair evaluation.
This preparation step was crucial, especially for neural networks, which tend to perform better when feature values are on a consistent scale. For the Tsetlin Machine, binarization was applied during the classifier stage itself.

5.3 Least Squares Classifier

Methodology:

The Least Squares method, traditionally used for signal estimation tasks, was applied here to classify binary outcomes. It works by fitting a linear relationship between the input features and the output labels by minimizing the squared difference between

actual and predicted values. The solution is derived using the normal equation, and a binary decision is made by applying a threshold to the predicted output.

This method assumes a linear relationship between input variables and the decision boundary. While it lacks the sophistication of modern ML algorithms, it has been widely used for its speed and analytical solution.

Characteristics:

- Computationally light, making it feasible for real-time applications
- Relies on linearity assumptions, which may limit its adaptability to complex data distributions
- Straightforward to implement and interpret
- Poor handling of feature interactions unless explicitly modeled

Performance:

- Accuracy: 81.83%
- Bit Error Rate (BER): 0.1817
- Resource Utilization: Low; does not require GPU or specialized hardware

This model performed as expected: it provided a reliable benchmark and required minimal tuning. However, its inability to model non-linear decision boundaries became apparent during analysis.

5.4 Tsetlin-Voting Classifier

Methodology:

Inspired by the Tsetlin Machine, this model employs a simplified rule-based methodology in which binary features are used to vote on a decision. By examining their distributions across positive and negative samples, it determines the vote weights for each feature. After that, votes are counted and compared to a threshold that has been learned for classification.

To maximize classification accuracy, this threshold was calibrated using a sweep on the training set, which allowed the model to slightly adjust to different data distributions.

Characteristics:

- Fully interpretable, rule-based system
- Extremely lightweight and suitable for constrained devices
- Can be rapidly trained and deployed without specialized environments
- Easily ported to logic-based hardware (FPGAs, microcontrollers)

Performance:

- **Optimal threshold (t)***: 5
- Training Accuracy: 75.99%
- Test Accuracy: 76.53%
- Bit Error Rate (BER): 0.2347

• Resource Utilization: Minimal resource usage; only simple logic operations are needed.
This model has a clear trade-off: explainability and efficiency are provided at the expense of overall predictive performance. It would be perfect in situations where processing power is constrained or interpretability is crucial.

5.5 Deep Neural Network (DNN)

Methodology:

The backpropagation algorithm was used to train the DNN model, which had two hidden layers. Through iterative weight adjustments based on gradient descent, this model learns nonlinear mappings between input features and output labels. The Adam optimizer was used for training over a number of epochs, using a combination of real and imaginary feature parts as input.
Significant preprocessing was needed for this model, including output normalization and one-hot encoding. Hyperparameter adjustment, such as batch size, number of neurons, for stable training and convergence, size, and learning rate, were crucial.
Characteristics:

- Can capture complex, nonlinear dependencies in data
- Highly flexible and adaptable to diverse input conditions
- Requires significant computational resources for training but can generalize well once trained
- Training stability depends on data balance, regularization, and initialization

Performance:

- Accuracy: 99.42%
- Bit Error Rate (BER): 0.0058
- Resource Utilization: Moderate to high; performs best with GPU or multicore processors
-

The DNN's near-perfect accuracy validates its capacity for capturing subtle patterns in data that are invisible to linear models or thresholding schemes. It serves as proof that deep learning can bring substantial gains in wireless communication problems.

5.6 Comparative Analysis Table

Metric	Least Squares	Tsetlin-Inspired	Deep Neural Network
Accuracy (%)	81.83	76.53	99.42
Bit Error Rate (BER)	0.1817	0.2347	0.0058
Interpretability	Moderate	High	Low
Training Time	Fast	Very Fast	Moderate
Adaptability	Low	Medium	High
Complexity	Low	Low	High
Resource Utilization	Low	Very Low	Moderate/High

Suitable Use Case	Quick Baseline	Lightweight Devices	High-performance Systems

5.6 Results:

```
>> compare_classifiers
Train positives: 5762 / 24000
 Test positives: 1408 / 6000

Tsetlin count t* = 5 (train ACC = 75.99%)
```

Fig 6.1: Tsetlin classifier training log showing class distribution and optimal threshold selection. threshold selection.

```
=== Classification Performance ===
Least Squares    : ACC = 81.83% | BER = 0.1817
Tsetlin-like     : ACC = 76.53% | BER = 0.2347
DNN (patternnet) : ACC = 99.42% | BER = 0.0058
```

Fig 6.2 Accuracy and BER comparison of LS, Tsetlin-like, and DNN classifiers.

5.7 Personal Insights and Reflections

While working with these three classification methods, I observed not only their numerical performance but also practical aspects such as ease of implementation, interpretability, and adaptability. The Least Squares method is undoubtedly the easiest to implement and can deliver surprisingly good results for linearly separable problems. It serves well as a benchmark and is suitable when computational simplicity is a priority.

The Tsetlin-inspired model intrigued me due to its rule-based logic and negligible resource demand. It provided me with a new perspective on binary learning mechanisms that don't rely on traditional weight updates but on count-based logical reinforcement. Despite its lower accuracy, its strength lies in its clarity and hardware compatibility.

As anticipated, the DNN showed the strongest performance. But in terms of tuning and training time, it was also the most taxing. To get consistent results, I discovered that careful network design, learning rate selection, and normalization were needed. The benefit of non-linear function approximation is demonstrated by its high accuracy, but at the expense of transparency and computational demand.
According to my observations, fusing the expressive capabilities of neural networks with aspects of interpretability from rule-based models may be a promising avenue for future research, particularly in resource-constrained settings like embedded systems or edge devices.

5.8 Conclusion

This comparative analysis demonstrates the performance of each approach on a range of metrics. Despite its simplicity, the LS approach offers dependable performance in environments with limited resources. The interpretability and computational efficiency of the Tsetlin-inspired model are excellent, but its predictive accuracy is lacking. Although the DNN uses more resources, it performs noticeably better than the other two in terms of accuracy and error rate.
In conclusion, DNN works best when computational resources are available and accuracy is the top concern.

- The Tsetlin-inspired model is perfect for situations that call for speed, transparency, and simplicity.
- For linear tasks with constrained hardware, Least Squares offers a robust, quick baseline.

The benefits of employing deep learning methods for channel estimation tasks are validated in this last stage, which also raises the possibility of integrating neural networks and logical rule-based models for future systems' optimal performance. This comparative framework offers a basis for implementing machine learning models in the wider wireless communications context, where system-level limitations like latency, memory consumption, or real-time inference become crucial. The implementation techniques can readily be extended to future channel estimation tasks across 5G, 6G, and beyond, and the insights obtained here can help practitioners select the best tool for particular deployment scenarios.

24

Chapter – 6

Conclusion and Future Work

6.1 Conclusion

Investigating the performance and suitability of machine learning algorithms for the channel estimation task in wireless communication systems was the main goal of this thesis. This study has provided both a technical and comparative evaluation based on actual experimentation by concentrating on three classification approaches: a Deep Neural Network (DNN), a simplified voting model inspired by Tsetlin Machine, and Least Squares (LS). Every approach showed distinct advantages and disadvantages. Because of its ease of use and low processing overhead, the Least Squares method was a reliable starting point, but it was insufficient for simulating intricate, nonlinear channel properties. Although the rule-based logical structure introduced by the Tsetlin-inspired approach is very interpretable and hardware-friendly, its accuracy was not as high as that of the more sophisticated models. However, the DNN showed

Through these implementations, the thesis has provided evidence that machine learning, particularly deep learning, can play a pivotal role in enhancing channel estimation techniques. Moreover, it has highlighted the importance of trade-off analysis in selecting a method suitable to the system's constraints, such as hardware limitations, latency requirements, and power consumption.

This research has not only achieved its immediate objectives—implementing and comparing LS, TM-like, and DNN approaches—but has also laid the groundwork for further investigation into logic-based learning systems and hybrid models.

6.2 Future Work

While the findings of this thesis are promising, they also open the door to several directions for future exploration:

1. Hardware Implementation and Benchmarking:

The current models were implemented and evaluated in MATLAB. A natural extension of this work would be to deploy these models on embedded platforms (e.g., ARM Cortex, Zynq SoC, or NVIDIA Jetson) and measure real-time performance, latency, memory consumption, and power usage.

2. Improving the Tsetlin Machine:

The simplified TM-inspired model can be replaced with a full Tsetlin Machine framework that includes clauses, automata, and feedback mechanisms. This would allow a more competitive comparison with the DNN in terms of accuracy while retaining interpretability.

3. Hybrid Model Design:

A promising path would be to develop a hybrid model that combines the transparency of rule-based logic (Tsetlin) with the feature extraction power of neural networks. Such a model could adaptively switch between modes based on computational resources or prediction confidence.

4. Robustness Under Channel Variations:

The models evaluated here used synthetic data with idealized channel assumptions. Testing these classifiers under more realistic conditions (e.g., multi-path fading, Doppler spread, and non-stationary noise) would further validate their utility in live systems.

5. Extension to Multi-Class or Regression Tasks:

While this work focused on binary classification, real-world channel estimation may require predicting a continuous-valued impulse response or handling multiple modulation types. Extending the models to regression or multi-class settings would broaden their applicability.

6. Dataset Expansion and Augmentation:

The current dataset, though useful for baseline validation, was synthetically generated. Applying the same models to datasets captured from software-defined radio (SDR) experiments or real-world testbeds would enhance relevance and reliability.

7. Integration with End-to-End Communication Pipelines:

Finally, integrating ML-based channel estimation into a complete OFDM-based receiver chain—covering synchronization, demodulation, decoding, and equalization—would offer a more holistic understanding of their contribution to overall system performance.

6.3 Closing Remarks

This thesis has provided a comparative lens into how different machine learning paradigms can be applied to a foundational problem in wireless communication. It underscores that no single method is universally superior, but rather, each

serves a specific niche depending on performance, interpretability, and resource constraints.

With the rapid development of AI-enabled 6G networks and edge intelligence, the integration of efficient and accurate learning-based channel estimators will likely become a standard design component. This work is a small but significant contribution toward that direction.