UNIVERSITY OF CALIFORNIA, MERCED

Stochastic Methods for Machine Learning and their Applications

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy
Proposal for PhD qualifying exam in

Electrical Engineering and Computer Science

by

Azar Alizadeh

Committee members:

Professor Mukesh Singhal, Chair
Professor Reza Ehsani
Professor Xiaoyi Lu

2024

Copyright Notice

The Dissertation of Azar Alizadeh is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Reza Ehsani

Xiaoyi Lu

Mukesh Singhal (Chair)

University of California, Merced

2024

Dedication

This thesis is dedicated, first and foremost, to my parents Shamsi Hashemnejhad and Ali Alizadeh, whose unwavering support, love, and sacrifices have made this journey possible. Your belief in me and your guidance through every step of life has been my foundation.

To my beloved daughter Diana Naserkhaki, who fills my life with joy and purpose and whose presence has been a constant source of inspiration and strength. You were by my side during the toughest moments, and together, we have grown and learned, shaping our paths forward.

To my siblings, Arezou, Mohammad, and Azadeh Alizadeh, for being my closest allies and for standing by me in every challenge, always encouraging me and giving me a sense of belonging.

This work is also dedicated to my advisors, mentors, and colleagues, whose guidance and insights have shaped my research and scholarly growth. Your wisdom and support have been instrumental in my academic journey.

Finally, I dedicate this thesis to all those whose names may not be mentioned but whose contributions, large and small, have left an indelible mark on my life and academic endeavors. I deeply appreciate your belief in me and your encouragement along the way.

Epigraph

The growth of knowledge depends entirely upon disagreement.

— *Karl Popper*

Contents

List of Figures **ix**

List of Tables **xi**

List of Algorithms **xii**

Acknowledgment **xiii**

Vita and Publication **xiv**

Abstract **xvii**

1 Introduction and Background **1**

1.1 Introduction 2
1.2 Supervised Learning 2
1.3 Unsupervised Learning 14
1.4 Stochastic Processes in Machine Learning 18
1.5 Problem Statement 21
1.6 Motivation and Objectives 23
1.7 Dissertation Outline 24

2 Stochastic Induction of Decision Trees with Application to Learning Haar Trees **26**

2.1 Introduction 27
2.2 Problem Statement and Basic Idea 30
2.3 Preliminaries 32
2.3.1 Induction of a decision tree 32
2.4 Proposed Method 33
2.5 Analysis of the Proposed Algorithm 34

2.5.1 Computational Complexity 34
2.5.2 Relation to Objective Function 36
2.6 Experiments, Results, and Discussion 39
2.6.1 Haar Tree 47
2.6.2 Haar Features 47
2.6.3 Haar Tree Results 61
2.7 Chapter Summary 65
2.8 Limitation and Future Work 65

3 A Novel Approach For Synthetic Reduced Nearest-Neighbor Leveraging Neural Networks 66
3.1 Introduction 67
3.2 The Proposed Methods 69
3.2.1 Convergence to a local maximum 71
3.2.2 Pruning 72
3.3 Experimental Results 72
3.3.1 Datasets 72
3.3.2 Experimental Setup 73
3.3.3 Results 74
3.4 Chapter Summary 86

4 Enhancing Synthetic Reduced Nearest-Neighbor with Two-Layer Neural Networks: A Step Forward in Image Classification 87
4.1 Introduction 88
4.2 The Proposed Methods 90
4.2.1 Model Architecture and Learning Objective 90
4.2.2 Rationale Behind the Architectural Choices 94
4.2.3 Selection Process 94
4.2.4 Resource Efficiency Benefits 94
4.2.5 Interpretability and Adaptability 95
4.2.6 Contributions and Advantages 95
4.3 Experimental Results 98
4.3.1 Datasets 98
4.3.2 Experimental Setup 99
4.3.3 Results 100
4.3.4 Complexity Analysis of TLN-SRNN 105

4.4 Chapater Summary 106

5 Enhancing Irrigation Efficiency with a Unified Stochastic Decision Tree Model: Predictive Analysis of Stem Water Potential in Almond and Pistachio Orchards 108

5.1 Introduction 109
5.2 Stochastic Decision Trees 113
5.3 Material And Method 114
5.4 SDT setup 117
5.5 Experimental Results 118
5.6 Chapter Summary 121

6 Concluding Remarks and Directions for Future Research 122

6.1 Summary of Contributions 122
6.2 Directions for future research 124

Bibliography 125

List of Figures

2.1 Training and Testing Error ratio versus computational complexity for MNIST dataset. 40
2.2 Training and Testing Error ratio versus computational complexity for FASHIONMNIST dataset. 41
2.3 Training and Testing Error ratio versus computational complexity for COVERTYPE dataset. 42
2.4 Training and Testing Error ratio versus computational complexity for ISOLET dataset. 43
2.5 Training and Testing Error ratio versus computational complexity for SATIMAGE dataset. 44
2.6 Training and Testing Error ratio versus computational complexity for SIGNMNIST dataset. 45
2.7 Haar Filters 48
2.8 Impurity reduction at each iteration for (a) 1 versus 0, (b) 1 versus 7. 49
2.9 Impurity reduction at each iteration for (a) 5 versus 8, (b) all classes. 50
2.10 Feature importance (FI) of features by SDT at each iteration for the problem of 0 versus 1. Brighter pixels show a higher value of FI, and darker pixels show a lower value of FI. 52
2.11 Feature importance of features by SDT at each iteration for the problem of 1 versus 7. Brighter pixels show a higher value of FI, and darker pixels show a lower value of FI. 53
2.12 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on MNIST 55
2.13 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on FASHIONMNIST 56
2.14 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on SIGNMNIST 57

2.15 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on ISOLET 58
2.16 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on SATIMAGE 59
2.17 Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on COVERTYPE 60
2.18 Inference complexity versus error ratio over train and test set for different models on MNIST. 63
2.19 Inference complexity versus error ratio over train and test set for different models on FashionMNIST. 64

3.1 Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for MNIST. 75
3.2 Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for FASHION MNIST. 76
3.3 Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for SIGN MNIST. 78
3.4 Training (a) and Testing (b) response of Neural-SRNN with different numbers of neural networks for each class. 80
3.5 Different configurations of Neural net structures. (a) One neural net per class, (b) Twenty Neural nets per class. 83
3.6 Feature importance of features by Neural-SRNN for (a)-(b) class 1 and class 7 in MNIST dataset and (c)-(d) class pullover and shirt from FASHION-MNIST. 85

4.1 TLN-SRNN accuracy on the train set (a) and test set (b) 101
4.2 TLN-SRNN Training time vs. N-SRNN Training time. 103
4.3 Learning figures of the TLN-SRNN model on MNIST, Fashion MNIST, and Sign MNIST. 106

5.1 Prediction accuracy for different scenarios in Trainset and Testset . . 119
5.2 Train & Test set accuracy for all scenarios on SVM, KNN, RF, & SDT 120

List of Tables

2.1 The used datasets Information 46

4.1 Layer-wise description of Shallow Net & MiniCNNs 91
4.2 Performance comparison of models across different datasets 104

5.1 Dataset scenarios 118
5.2 Comparison of ML Models on Precision, Recall & F1 Score on SC1 . 120

List of Algorithms

1 Nearest neighbor algorithm 68

Acknowledgment

I express my deepest gratitude to my advisor, Professor Mukesh Singhal, for his unwavering support and invaluable guidance throughout my doctoral studies. His mentorship has powerfully shaped this thesis and my academic and professional growth. I feel incredibly fortunate to have had him as my advisor; His influence will continue to guide me in my future endeavors.

I am deeply grateful to my co-advisor, Professor Reza Ehsani, for his unwavering support, mentorship, and invaluable guidance throughout this journey. His constant advice has been essential for both my research and my personal growth. I am incredibly fortunate to have had him as my co-advisor, and his belief in me was the driving force behind my decision to pursue a PhD in the United States.

I want to express my appreciation to my committee member, Dr. Xiaoyi Lu, for his thoughtful feedback and suggestions that have enhanced the quality of this work.

I also owe a debt of gratitude to my colleagues and friends, especially Dr. Aditya Ranganath, Dr. Pooya Tavallali, and Dr. Vahid Behzadan, for their collaborative efforts, enriching discussions, and unwavering companionship during this journey.

I am deeply grateful to the University of California, Merced, for the financial support and resources provided during my doctoral studies, which allowed me to focus on my research and pursue my academic goals.

I am also profoundly grateful to Dr. Azadeh Alizadeh and Dr. Abbas Bozorgirad for supporting me while pursuing my PhD. Your belief in me has been my anchor during the most challenging times. I give special thanks to Dr. Elnaz Akbari, who may not be my blood sister, but is genuinely a soul sister and has been a constant source of strength and support.

Finally, I would like to extend my heartfelt thanks to my family for their endless love, support, and encouragement throughout this long journey. I am also grateful to everyone who has contributed to this journey. Their encouragement and belief in me have meant more than words can express. Thank you for your unwavering support.

VITA

- **University of California, Merced**. Merced, CA, USA. (2018 - 2024).
  Ph.D. in Electrical Engineering and Computer Sciences.

- **Azad University**. Iran. (2009 - 2012).
  M.Sc. Industrial Engineering (Improvement & System Management).

- **Azad University**. Iran. (1999 - 2005).
  B.Sc. Computer Engineering (Software Design).

PUBLICATIONS

Conference Proceedings

- **Azar Alizadeh**, M. Farajijalal, Z. Rezvani, A. Toudeshki, and R. Ehsani, "Developing a data-driven model for predicting water stress in pistachio trees", in International Congress on Agricultural Mechanization and Energy in Agriculture, Springer Nature Switzerland Cham, 2023, pp. 186–196.

- **Azar Alizadeh**, M. Mortazavi, M. Farajijalal, A. Toudeshki, R. Ehsani, and M. Singhal, "Enhancing irrigation efficiency with a unified stochastic decision tree model: Predictive analysis of stem water potential in almond and pistachio orchards", in The 26th International Conference on Artificial Intelligence (ICAI'24: July 22-25, 2024; Las Vegas, USA), 2024.

- **Azar Alizadeh** and M. Singhal, "Enhancing synthetic reduced nearest-neighbor with two-layer neural networks: A step forward in image classification", in 2024 IEEE Workshop on Signal Processing Systems, 2024.

- **Azar Alizadeh**, M. Singhal, V. Behzadan, P. Tavallali, and A. Ranganath, "Stochastic induction of decision trees with application to learning haar trees", in 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), 2022.

- **Azar Alizadeh**, P. Tavallali, V. Behzadan, A. Ranganath, and M. Singhal, "A novel approach for synthetic reduced nearest-neighbor leveraging neural networks", in 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2022, pp. 831–836.

- **Azar Alizadeh**, P. Tavallali, M. R. Khosravi, and M. Singhal, "Survey on recent active learning methods for deep learning", in Advances in Parallel & Distributed Processing, and Applications: Proceedings from PDPTA'20, CSC'20, MSV'20, and GCC'20, Springer International Publishing, 2021, pp. 609–617.

- L. Pandey, **Azar Alizadeh**, and A. S. Arif, "Enabling predictive number entry and editing on touchscreen-based mobile devices", in Proceedings of the 2020 Conference on Human Information Interaction and Retrieval, 2020, pp. 13–22.

- P. Tavallali, V. Behzadan, **Azar Alizadeh**, A. Ranganath, and M. Singhal, "Adversarial label-poisoning attacks and defense for general multi-class models based on synthetic reduced nearest neighbor", in 2022 IEEE International Conference on Image Processing (ICIP), IEEE, 2022, pp. 3717–3722.

Journal Articles

- L. Afsah-Hejri, E. Akbari, A. Toudeshki, T. Homayouni, **Azar Alizadeh**, and R. Ehsani, "Terahertz spectroscopy and imaging: A review on agricultural applications", Computers and Electronics in Agriculture, vol. 177, 2020.

- H. Ahmadi, M. Nilashi, L. Shahmoradi, O. Ibrahim, F. Sadoughi, M. Alizadeh, and **Azar Alizadeh**, "The moderating effect of hospital size on inter and intra-organizational factors of hospital information system adoption", Technological Forecasting and Social Change, 2018.

- E. Akbari, M. Mir, M. V. Vasiljeva, **Azar Alizadeh**, and M. Nilashi, "A computational model of neural learning to predict graphene based isfet", Journal of Electronic Materials, vol. 48, pp. 4647–4652, 2019.

- E. Akbari, R. Moradi, A. Afroozeh, **Azar Alizadeh**, and M. Nilashi, "A new approach for prediction of graphene based isfet using regression tree and neural network", Superlattices and Microstructures, vol. 130, pp. 241–248, 2019.

- E. Akbari, M. Nilashi, **Azar Alizadeh**, and Z. Buntat, "Soft computing techniques in prediction gas sensor based 2d material", Organic Electronics, vol. 62, pp. 181–188, 2018.

- **Azar Alizadeh**, F. Mosalanezhad, A. Afroozeh, E. Akbari, and Z. Buntat, "Support vector regression and neural networks analytical models for gas sensor based on molybdenum disulfide", Microsystem Technologies, vol. 25, pp. 115–119, 2019.

- M. Nilashi, O. Ibrahim, E. Yadegaridehkordi, S. Samad, E. Akbari, and **Azar Alizadeh**, "Travelers decision making using online review in social network sites: A case on tripadvisor", Journal of computational science, vol. 28, pp. 168–179, 2018.

- M. Nilashi, S. Samad, E. Yadegaridehkordi, **Azar Alizadeh**, and E. Akbari, "Early detection of diabetic retinopathy using ensemble learning approach.", Journal of Soft Computing & Decision Support Systems, vol. 6, no. 2, 2019.

- S. M. Sajadi, **Azar Alizadeh**, M. Zandieh, and F. Tavan, "Robust and stable flexible job shop scheduling with random machine breakdowns: Multi-objectives genetic algorithm approach", International journal of mathematics in operational research, vol. 14, no. 2, pp. 268–289, 2019.

ABSTRACT OF THE DISSERTATION

Stochastic Methods for Machine Learning and their Applications

by

Azar Alizadeh

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Merced, 2024

Dr. Mukesh Singhal, Chair

In the fast-advancing domains of artificial intelligence and machine learning, the need for models capable of efficiently handling large, complex datasets is inevitable. Traditional methods such as decision trees and nearest-neighbor algorithms often struggle with computational complexity and scalability when applied to high-dimensional data, especially in domains like agriculture, where large amounts of real-time data are generated from sensor networks. These challenges require the development of more efficient learning algorithms that can reduce computational overhead while maintaining predictive accuracy.

To address these challenges, in this research, we propose a novel Stochastic Decision Tree (SDT) model that introduces randomness into the tree induction process, significantly reducing computational complexity while maintaining or improving accuracy. The proposed method leverages stochasticity to prioritize important features and efficiently process large datasets. The proposed Neural-SRNN model addresses class imbalance issues by using a specialized loss function (Focal Loss) to focus on hard-to-classify instances. We demonstrate the effectiveness of this approach through empirical evaluations of standard datasets.

In addition, we extend the application of stochastic methods by developing a Neural-Synthetic Reduced Nearest Neighbor (Neural-SRNN) algorithm, which integrates neural networks into the SRNN framework to improve performance and interpretability in high-dimensional classification tasks. This method combines the flexibility of neural networks with the computational advantages of the SRNN model, achieving superior performance on image classification benchmarks such as MNIST and Fashion-MNIST with an expectation-maximization approach.

To preserve the strength of the proposed Neural-SRNN model and improve ef-

ficiency and scalability, we proposed a two-layer Neural-SRNN model that builds on the Synthetic Reduced Nearest-Neighbor (SRNN) architecture. This model uses a modular approach, employing Mini-Convolutional Neural Networks (Mini-CNNs) in the first layer to perform class-specific feature extraction, followed by a shallow neural network for final classification. This two-layer architecture allows for efficient parallel processing, significantly reducing computational overhead while maintaining high accuracy. Moreover, it improves generalization and prevents overfitting on complex datasets such as SignMNIST and FashionMNIST. The two-layer Neural-SRNN model demonstrates superior accuracy and computational efficiency, demonstrating its scalability and adaptability to complex, high-dimensional data.

The dissertation further explores the application of the Stochastic Decision tree technique to real-world agricultural problems, particularly in optimizing water usage in almond and pistachio orchards. By predicting stem water potential using aerial and ground sensor data, the SDT model improves irrigation efficiency and contributes to the development of sustainable agricultural practices. This work advances machine learning by presenting novel and efficient stochastic algorithms and demonstrating their practical applicability in critical areas such as agriculture.

Chapter 1

Introduction and Background

1.1 Introduction

Machine learning (ML) and deep learning are rapidly growing areas of artificial intelligence (AI), which focus on developing algorithms capable of recognizing patterns in data, making decisions or predictions based on these patterns, and learning from the data. Unlike traditional computer programs that follow explicit instructions, ML models are designed to learn from data and perform tasks autonomously LeCun et al. (2015), Goodfellow et al. (2016). These models generate hypotheses, reason, and make decisions without requiring exhaustive rules for every scenario, imitating human-like decision-making processes.

Machine learning allows computers to generalize past experiences or data. Rather than memorizing problem examples, ML systems learn lessons from prior data to tackle new unseen problems. This ability to generalize is the core strength of machine learning, allowing systems to process new information, generate forecasts, and improve decision-making Mitchell and Mitchell (1997).

Machine learning algorithms can be divided into several major categories based on Alpaydin (2020), the most notable being supervised and unsupervised learning. Supervised and unsupervised learning have succeeded significantly in various fields, including image and speech recognition, healthcare, predictive analytics, and natural language processing.

1.2 Supervised Learning

Supervised learning is one of the most fundamental approaches in machine learning. This approach trains a model on a labeled dataset where each data point (input) is associated with a corresponding output (label). The objective of supervised learning is to find a function that maps the inputs X to the outputs Y as accurately as possible so the model can predict the correct outputs for new unseen data Bishop (2006).

Supervised learning can be divided into two categories based on the task it focuses on, which are **Classification** and **Regression** Bishop (2006).

Classification refers to a supervised learning approach where the target variable Y is a discrete label or category. The objective of a classification model is to learn a relationship from input data X to one of the predefined categories Y. This approach is extensively applied in practical scenarios, including spam filtering, image recogni-

tion, medical diagnosis, and speech recognition Hastie et al. (2009), Goodfellow et al. (2016).

In classification tasks, algorithms aim to minimize the classification error, which is the proportion of incorrect predictions made by the model. Standard metrics used to evaluate classification models include accuracy, precision, recall, F1-score, and the area under the ROC curve. Common models for classification include:

- **Decision Trees**: These algorithms split the data into branches based on feature values, assigning class labels at the leaf nodes Quinlan (1986a).
- **k-Nearest Neighbors (KNN)**: A distance-based algorithm that assigns the label based on the majority vote among the k nearest neighbors Cover and Hart (1967a).
- **Support Vector Machines (SVMs)**: These algorithms find a hyperplane that best separates the data into different classes Bishop (2006).

Regression is a supervised learning approach where the output variable Y is continuous rather than categorical. Regression aims to predict a continuous value based on the input features X. Regression models are frequently used in financial forecasting, resource allocation, environmental modeling, and agricultural yield prediction Hastie et al. (2009), Bishop (2006).

In regression tasks, the model's goal is to minimize the error between the predicted and actual values of the output variable. A common evaluation metric for regression models is the Mean Squared Error (MSE), which measures the average squared difference between predicted values and true values:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2$$

where Y_i is the true value and $\hat{Y}_i$ is the predicted value for the i-th data point.

Some common regression algorithms include:

- **Linear Regression**: This algorithm models the relationship between the input and output variables by fitting a linear equation to the data. It assumes a linear relationship between input features and output Bishop (2006).
- **Decision Trees for Regression**: Like their classification counterparts, the regression decision trees split data into subsets, with the final output being a continuous value Quinlan (1986a).

- **k-Nearest Neighbors for Regression**: KNN can also be used for regression by averaging the outputs of the k-nearest neighbors to the input Hastie et al. (2009).

Supervised learning models require substantial labeled datasets, which can be costly and time-intensive to acquire Goodfellow et al. (2016), and training on large datasets can also be computationally demanding LeCun et al. (2015). However, supervised learning models can be over-fitted when dealing with limited datasets or noisy data Bishop (2006). Nevertheless, when supervised learning models are trained with an ample amount of labeled data, they tend to generalize well to unseen datasets. Algorithms like decision trees are typically straightforward to understand, and with the appropriate model choice and enough labeled data, supervised learning models can achieve significant accuracy Mitchell and Mitchell (1997).

Supervised learning algorithms are highly varied; they range from very simple to the most complex systems, such as decision trees, K-Nearest Neighbors, and deep neural networks. Each algorithm has its own specifications, strengths, and weaknesses, including which problems it can or can't be applied to and how effectively it can do so. Therefore, the choice of algorithm is a critical factor in determining how well the underlying structure of the data is captured and applied to new tasks. In exploring supervised learning algorithms, we need to understand how they work based on guiding principles. The following sections will detail the Decision Trees, K-nearest neighbors, and Deep Neural Networks, including their advantages, disadvantages, and the mathematical formulations that define the learning mechanisms.

Decision Trees

A decision tree is a non-parametric supervised learning algorithm widely used for both classification and regression. It is structured as a flowchart, where each internal node represents a decision based on a feature, each branch represents the outcome of the decision, and each leaf node represents the final output or prediction Quinlan (1986a).

The process of constructing a decision tree involves recursively splitting the dataset into subsets based on the value of a feature. In classification, metrics like Gini Impurity and Information Gain are used to measure the quality of a split, while in regression, Mean Squared Error (MSE) is often used Hastie et al. (2009).

The Gini Impurity $G(t)$ for a node t is given by:

$$G(t) = 1 - \sum_{i=1}^{C} p_i^2 \tag{1.1}$$

where p_i represents the proportion of instances belonging to class i at node t, and C denotes the total number of classes. The Gini Impurity calculates the chance of incorrectly classifying a randomly selected element if it were labeled randomly based on the subset's label distribution Quinlan (1986a).

On the other hand, Information Gain is grounded on a discrete but essential principle, **entropy**, which informs us about the disorder of the data or its impurity and the amount of entropy associated with a specific dataset. The entropy H of a dataset T is defined as:

$$H(T) = -\sum_{i=1}^{C} p_i \log_2(p_i) \tag{1.2}$$

The Information Gain for a feature X is then given by:

$$IG(T, X) = H(T) - \sum_{v \in \text{Values}(X)} \frac{|T_v|}{|T|} H(T_v) \tag{1.3}$$

where T_v is the subset of T for which feature X has value v Quinlan (1986a).

Decision trees are easy to interpret as the decision-making process is visually representable. They can be used for both categorical and numerical data without any need for pre-processing. Moreover, unlike distance-based algorithms, such as KNN, which require feature scaling because the distance between data points can be distorted if features are on different scales, decision trees do not require feature scaling since they make decisions based on thresholds. These thresholds are not influenced by the absolute values of the features. This characteristic makes decision trees simpler to use with datasets where features have different scales, as there is no need to apply additional pre-processing like normalization or standardization Loh (2011), Breiman et al. (1984).

On the other hand, decision trees tend to overfit, especially in terms of inducing a deep tree, and when the model becomes too complex, Quinlan (1986a). Decision trees can also be unstable since small changes in the data can significantly change the tree's structure. Moreover, decision trees can be biased towards classes with a larger group of samples in the dataset Hastie et al. (2009).

Literature Review and Related Work

Decision trees have been widely studied since their inception, providing a robust and interpretable approach to both classification and regression tasks. Their popularity comes from their intuitive structure, ease of implementation, and ability to handle both numerical and categorical data.

The concept of decision trees was first introduced by **Quinlan** in 1986 with the **ID3 algorithm** Quinlan (1986a). This algorithm was based on the use of entropy and information gain to split nodes. The subsequent development of **C4.5**, which extended ID3 to handle continuous attributes and missing values, solidified decision trees as a popular machine learning method Quinlan (1993). **CART (Classification and Regression Trees)**, introduced by **Breiman et al.** in 1984, became another foundational algorithm for decision trees, supporting both classification and regression tasks using Gini impurity and mean squared error for node splitting Breiman et al. (1984).

Decision Trees in Agriculture: Water Management

One of the emerging areas of decision tree applications is in **agriculture**, particularly in **water resource management**. Efficient water usage is crucial in modern farming, especially given the increasing concerns over water scarcity and the need for sustainable agricultural practices. Decision trees provide an effective solution for predicting and managing water needs, irrigation schedules, and optimizing water usage.

For example, **Araya et al.** Araya et al. (2016) used decision trees to classify **crop water stress** and predict water requirements based on environmental and crop-specific factors. Their study demonstrated that decision trees could help farmers make informed irrigation decisions by analyzing weather patterns, soil moisture levels, and crop growth stages.

Similarly, **Tsoumakas and Vlahavas** Tsoumakas and Vlahavas (2002) applied decision trees to predict **water consumption** for various crops in Greece. They used historical weather data, crop types, and soil characteristics to train decision trees that helped optimize irrigation schedules. This method improved water efficiency by ensuring that crops received the appropriate amount of water at the right time.

In another study, **Aqeel-ur-Rehman et al.** Aqeel-ur Rehman et al. (2009) used decision trees for **precision irrigation** by predicting soil moisture levels based on

real-time sensor data. This approach allowed for more accurate predictions of when and how much water should be applied, reducing water wastage and increasing crop yield. The flexibility of decision trees in handling both continuous and categorical data made them an ideal choice for managing the various factors involved in water management.

In a related study, **Al-Ghobari and Mohammad** Al-Ghobari and Mohammad (2011) used decision trees to develop a **decision support system (DSS)** for optimizing water use in irrigation. The system provided real-time recommendations for irrigation based on soil, weather, and crop data, significantly improving water-use efficiency. Their findings suggest that decision trees can contribute to sustainable water management practices in agriculture.

In addition to irrigation, decision trees have also been applied to forecast **drought conditions**. **Pereira et al.** Pereira et al. (2002) applied decision trees to predict drought risks in Portugal based on climate data and historical patterns. This allowed farmers to anticipate water shortages and plan their water usage accordingly, preventing crop loss during droughts.

The versatility of decision trees in managing various environmental and crop-related factors has been demonstrated in other agricultural water management contexts. For instance, **Singh et al.** Singh et al. (2015) developed a **decision tree-based model** to predict water demand for different crops based on climatic conditions and soil type in India. Their model helped farmers optimize their water usage during different growing seasons, resulting in better crop yields and more sustainable water consumption.

Despite the success of decision trees in water management, the models must be adapted to local environmental conditions. **Fan et al.** Fan et al. (2019) highlighted that decision trees, when combined with other techniques such as **remote sensing** and **machine learning ensembles**, can improve water-use efficiency in large-scale farming operations by providing region-specific irrigation recommendations.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple but powerful instance-based learning algorithm used for both classification and regression tasks Cover and Hart (1967a). Unlike many other algorithms, KNN does not build an explicit model during the training phase. Instead, it memorizes the entire training data set and makes predictions based on the k closest data points (neighbors) to a new input.

In a classification problem, KNN predicts the most common label among the k nearest neighbors. For regression, KNN predicts the average output of the k nearest neighbors. Mathematically, for a new instance x, the prediction in classification is:

$$\hat{y} = \text{mode}\{y_i : x_i \in \mathcal{N}_k(x)\} \tag{1.4}$$

and for regression:

$$\hat{y} = \frac{1}{k} \sum_{x_i \in \mathcal{N}_k(x)} y_i \tag{1.5}$$

The K-Nearest Neighbors (KNN) is highly plausible, simple, intuitive, and easy to implement and grasp. There is no training phase requirement for it (apart from storing the dataset); therefore, it is a lazy learning algorithm. KNN is a highly flexible method that can be applied to classification and regression tasks. It is efficient in many cases where data distribution is characterized by a decision boundary with high-level irregularity, and it naturally adapts to non-linear relationships. KNN is a non-parametric model that does not assume that the data comes from any particular distribution or fits a specific functional form. It does not try to "learn" a global model of the data; instead, it simply stores the training data and bases its predictions on the local neighborhood of the new data point Hastie et al. (2009).

K-Nearest Neighbors (KNN) is computationally demanding in terms of both memory and time, particularly for large datasets, as it requires storing and comparing all data points even after making predictions Cover and Hart (1967a). This makes the choice of k (the number of neighbors) crucial. Additionally, KNN's performance tends to degrade in high-dimensional spaces due to the curse of dimensionality. Overall, the scale of the data significantly affects how KNN functions. Features with larger ranges can dominate the distance calculation, leading to biased predictions. Therefore, it is essential to normalize the features correctly to avoid bias toward features with larger scales Bishop (2006).

Literature Review and Related Work

KNN has been applied in several domains due to its versatility and simplicity. For instance, **Hand et al.** Hand and Vinciotti (2001) utilized KNN in **biomedical classification**, particularly for cancer diagnosis using microarray data. The simplicity of KNN allowed it to handle high-dimensional biomedical datasets, though the curse of

dimensionality posed challenges, as discussed later.

In **finance**, KNN has been applied to predict stock market trends. **Liao and Wang** Liao and Wang (2010) showed that KNN could be used to forecast financial markets by analyzing historical price trends and using neighboring data points for predictions. In such applications, the flexibility of KNN in adapting to non-linear relationships between variables made it suitable for highly volatile and dynamic data.

KNN has also been successfully used in **image recognition**. For example, **Zhang et al.** Zhang and Luo (2006a) applied KNN to the **MNIST digit classification** dataset, demonstrating its ability to classify handwritten digits. However, they noted that the algorithm's performance degraded as the number of features increased, illustrating KNN's sensitivity to the curse of dimensionality.

Several improvements have been proposed to address the **computational inefficiencies** and **curse of dimensionality** associated with KNN. One common issue with KNN is that it requires storing and searching through the entire training dataset for each prediction, which can be computationally expensive for large datasets.

To address this, **Ball and Tree-based structures** have been introduced to optimize nearest neighbor search. **Omohundro** Omohundro (1989) developed **Ball Trees** to reduce the time complexity of finding nearest neighbors by partitioning the dataset into hierarchical clusters. This approach significantly improves the search time, especially for large datasets.

KD-Trees are another popular data structure used to optimize KNN search, particularly in low-dimensional spaces. **Bentley** Bentley (1975) introduced **KD-Trees**, which divide the data into hyperplanes for efficient neighbor searching. However, KD-Trees perform poorly in high-dimensional spaces due to the curse of dimensionality.

Another notable improvement to KNN is the use of **distance-weighted KNN**. **Dudani** Dudani (1976) proposed assigning weights to neighbors based on their distance from the query point, giving more influence to closer neighbors in the prediction. This helps improve the algorithm's accuracy in cases where the nearest neighbors are not equally informative.

A major challenge for KNN is the **curse of dimensionality**, which refers to the phenomenon where the distance between data points becomes less meaningful as the number of dimensions increases. **Bellman** Bellman (1966) first described this issue in the context of optimization and decision theory, and it remains a significant challenge for KNN.

High-dimensional data often results in neighbors that are equidistant from the

query point, leading to poor classification or regression performance. One solution to this problem is the application of **dimensionality reduction techniques** such as **Principal Component Analysis (PCA)** and **Linear Discriminant Analysis (LDA)**. **Bennett and Mangasarian** Bennett and Mangasarian (1994) used PCA to reduce the dimensionality of the feature space before applying KNN, significantly improving its performance.

Feature selection is another approach to combating the curse of dimensionality. **Guyon and Elisseeff** Guyon and Elisseeff (2003) reviewed various methods for selecting the most informative features in high-dimensional data. Their work emphasized that by eliminating irrelevant or redundant features, the performance of KNN could be enhanced in tasks such as image recognition and text classification.

Several studies have explored hybrid models that combine KNN with other machine learning algorithms, particularly **neural networks**. The motivation behind this combination is to leverage the **non-parametric** nature of KNN and the **feature extraction** capability of neural networks.

For instance, **Zhang et al.** Zhang and Luo (2006a) combined KNN with a neural network for **digit recognition**, where the neural network was used to extract features from the image data, and KNN performed the final classification based on these features used CNNsdel improved performance compared to using either KNN or neural networks alone, highlighting the advantages of using KNN for decision-making in feature space defined by neural networks.

Similarly, **Wu et al.** Wu et al. (2019) proposed a model that integrated **Convolutional Neural Networks (CNNs)** with KNN for image classification. In theTheirach, CNNs were used for feature extraction from raw images, and KNN was employed as the final classifier. This hybrid model achieved state-of-the-art results on several image datasets, demonstrating the complementary strengths of CNNs and KNN.

The primary advantages of KNN include its simplicity, versatility, and ability to adapt to non-linear relationships in data. However, the algorithm suffers from limitations related to high-dimensional data, computational complexity, and sensitivity to the choice of k. Various improvements, such as distance weighting, Ball Trees, KD-Trees, and hybrid models with neural networks, have been proposed to address these issues. Despite these challenges, KNN remains a popular algorithm for classification and regression tasks across a wide range of applications.

Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs) are powerful supervised learning models inspired by the structure of the human brain. DNNs comprise multiple layers and are designed to perform a long sequence of interoperator nodes or neurons. The input in the first layer will be converted into a higher-level abstract representation. From this point on, the model visualizes the relations developed that are not recognizable by the human eye. In turn, the model learns which inputs can generate the most probable outputs. DNNs have greatly improved dealing with high-dimensional information in datasets, such as picture recognition, speech recognition, and machine text understanding. Goodfellow et al. (2016).

A DNN comprises an input layer, multiple hidden layers, and one output layer. Each neuron in the layers of this network is connected to every other one via the next layer. The synapse strengths, represented by the links in the sense of the connecting body's segments, are continuously adjusted during the learning process to reduce the prediction error.

Mathematically, the operation of a single neuron can be described as follows. Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ be the input vector to a neuron, $\mathbf{w} = [w_1, w_2, \ldots, w_n]^\top$ be the corresponding weight vector, and b be the bias term. The neuron computes a weighted sum of its inputs, adds the bias, and passes the result through a non-linear activation function $\sigma(\cdot)$ to produce the output:

$$z = \mathbf{w}^\top \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b \tag{1.6}$$

$$y = \sigma(z) \tag{1.7}$$

Common activation functions $\sigma(\cdot)$ include the sigmoid function, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU). The output of each neuron in a layer is the input to the neurons in the subsequent layer. The output layer produces the network's result, which, depending on the task, can be a classification label, a regression value, or any other prediction type.

The training process of a DNN involves minimizing a loss function $L(\mathbf{y}, \hat{\mathbf{y}})$, where $\mathbf{y}$ is the true output and $\hat{\mathbf{y}}$ is the predicted output. The loss function measures the difference between the predicted and actual values. For example, in a classification task, the cross-entropy loss is commonly used.

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{1.8}$$

where C is the number of classes, y_i is the true label (usually encoded as a one-hot vector), and $\hat{y}_i$ is the predicted probability for class i.

To optimize the network's parameters (weights and biases), gradient-based optimization algorithms such as stochastic gradient descent (SGD) are employed. The gradients of the loss function with respect to the network's parameters are computed using backpropagation, a process that applies the chain rule of calculus to propagate errors backward from the output layer to the input layer, updating the weights along the way:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\mathbf{y}, \hat{\mathbf{y}}) \tag{1.9}$$

where θ_t represents the network's parameters at iteration t, and η is the learning rate.

One of the most significant strengths of the deep neural network is its high flexibility to model complex nonlinear functions with high accuracy. However, at the same time, they also include large computational resources, an enormous amount of labeled data, and the careful tuning of hyperparameters such as the learning rate, the number of layers, and the number of neurons per layer. In addition, DNNs exhibit the phenomenon of overfitting, particularly in cases of deep architecture and insufficient data. Problems such as dropout, batch normalization, and data augmentation are regular methods to deal with overfitting and increasing DNN efficiency. In a nutshell, Deep Neural Networks are a very useful method in the field of machine learning, capable of solving a wide range of complex tasks. However, they also bring about training complexity, computational costs, and the risk of overfitting LeCun et al. (2015), Goodfellow et al. (2016), Bishop (2006).

Literature Review and Related Work

Deep Neural Networks (DNNs) are a cornerstone of modern artificial intelligence, enabling breakthroughs in areas such as **image recognition**, **speech processing**, and **natural language understanding**. NNs are highly expressive models capable of learning complex representations from data. Their performance on tasks such as **speech recognition** and **natural language processing (NLP)** has significantly surpassed traditional methods. **Krizhevsky et al.** demonstrated the power of deep

learning with their **AlexNet** architecture, which won the ImageNet competition in 2012 and sparked widespread adoption of deep learning for computer vision tasks Krizhevsky et al. (2012).

While DNNs have demonstrated exceptional performance in many areas, combining them with other machine learning models can further enhance their capabilities by exploiting the strengths of different approaches. Hybrid models combine the **feature extraction power** of DNNs with the **decision-making capabilities** of traditional machine learning algorithms, offering improvements in performance, robustness, and interoperability Zhou (2012), Kontschieder et al. (2015), Roy and Kar (2021).

One of the most common combinations is between DNNs and **Support Vector Machines (SVMs)**. SVMs are effective in classification tasks where maximizing the margin between classes can lead to better generalization, while DNNs excel in feature extraction from complex datasets. By combining the two, the model can learn more powerful representations while ensuring that classification is based on maximized margins between classes Ding et al. (2014).

Another prominent hybrid approach combines DNNs with **Decision Trees** or **Random Forests**. Decision trees are highly interpretable and perform well on small datasets, while DNNs excel with large-scale, high-dimensional data. The combination of these models seeks to exploit the **interpretability** and **low-dimensional efficiency** of decision trees while retaining the **feature learning** capabilities of DNNs Kontschieder et al. (2015).

Several studies have explored the combination of k-Nearest Neighbors (KNN) with Neural Networks to leverage the strengths of both approaches. **Zhang et al. (2006)** proposed a hybrid model that combined a Neural Network with KNN for handwritten digit recognition. In this approach, the Neural Network was used to extract high-level features from images, and KNN was applied in the feature space for classification based on proximity. This method improved classification accuracy by combining the feature extraction capabilities of the Neural Network with the instance-based decision-making of KNN Zhang and Luo (2006b).

Wu et al. (2019) introduced a hybrid CNN-KNN model for image classification. The Convolutional Neural Network (CNN) component was responsible for deep feature extraction, and KNN was used in the final classification stage to refine the decision-making process. Their experiments on datasets such as MNIST and CIFAR-10 showed that combining CNN's deep learning capabilities with KNN's local classification significantly enhanced performance, particularly in cases where fine

decision boundaries were required Wu et al. (2019). These studies demonstrate that integrating KNN with Neural Networks can improve performance by exploiting both the generalization ability of Neural Networks and the local decision-making of KNN.

1.3 Unsupervised Learning

Unsupervised learning is a machine learning paradigm in which the algorithm is trained on unlabeled data, which means that the model does not have access to explicit input-output pairs during the training process. The objective of unsupervised learning is to discover hidden patterns or structures within the data, such as clusters, anomalies, or correlations Murphy (2012). This approach is especially useful in scenarios where labeled data are difficult or expensive to obtain and is widely used in applications ranging from data exploration to anomaly detection.

Unlike supervised learning, unsupervised learning focuses on understanding the inherent structure of the data without predefined labels. It aims to infer the natural distribution or groupings within the dataset. Two of the most common tasks in unsupervised learning are **Clustering** and **Dimensionality Reduction**.

Clustering involves grouping similar data points into clusters based on their features. The goal is to partition the dataset into distinct subsets (clusters), such that data points within the same cluster are more similar to each other than to those in other clusters Hastie et al. (2009). This technique is frequently used in customer segmentation, market analysis, image segmentation, and bioinformatics. Some popular clustering algorithms include **K-Means Clustering**, **Hierarchical Clustering**, **Gaussian Mixture Models (GMMs)** Lloyd (1982a), Johnson (1967), Bishop (2006).

Dimensionality reduction is one of the key tasks in unsupervised learning. It aims to reduce the number of input features in a dataset while preserving as much information as possible. This is particularly useful for high-dimensional datasets, where reducing the feature space can improve computational efficiency and visualization. Common techniques include **Principal Component Analysis (PCA)**, **t-Distributed Stochastic Neighbor Embedding (t-SNE)** Jolliffe (2002), van der Maaten and Hinton (2008)

One notable algorithm often used in the context of unsupervised learning is the **Expectation-Maximization (EM)** algorithm. It is particularly useful in probabilistic models that involve latent variables or missing data. EM iteratively refines

the estimates of model parameters by alternating between an expectation step (E-step) that computes the expected value of the latent variables, and a maximization step (M-step) that updates the model parameters to maximize the likelihood of the observed data Dempster et al. (1977). This flexibility makes EM widely applicable in fields like clustering (e.g., Gaussian Mixture Models) and image segmentation Bishop (2006).

Unsupervised learning is invaluable for tasks where labeled data is unavailable, making it especially useful in exploratory data analysis and large-scale datasets Murphy (2012). These models are adept at discovering hidden structures, such as clusters or patterns, without prior knowledge of the data. This flexibility allows unsupervised learning methods to excel in tasks like data compression, anomaly detection, and recommendation systems, where identifying underlying patterns is critical Hastie et al. (2009). By not relying on labels, these methods open up a range of possibilities for working with complex, high-dimensional data Bishop (2006).

Nonetheless, unsupervised learning presents various challenges. A major issue is the absence of evaluation metrics, since there is no labeled data to verify model performance, complicating the assessment of accuracy Bishop (2006). Additionally, these models can be quite sensitive to initial conditions, as algorithms such as K-Means and Expectation-Maximization (EM) depend heavily on starting points, influencing both convergence and results Lloyd (1982a), Dempster et al. (1977). Despite these hurdles, unsupervised learning remains crucial for processing unstructured data, providing robust tools for exploration and generalization in numerous domains Hastie et al. (2009).

The Expectation-Maximization (EM) Algorithm

The expectation maximization (EM) algorithm is a computational method used to find maximum-likelihood estimates of parameters in probabilistic models, mainly concerning the cases of latent variables and incomplete data. The expectation maximization (EM) algorithm is one of the most popular methods for such types of problems, where the likelihood cannot be computed directly due to the hidden states, which are the latent variables Dempster et al. (1977).

The EM algorithm works iteratively and involves two steps Neal and Hinton (1998):

1. **Expectation Step (E-Step)**: In this step, the algorithm computes the ex-

pected value of the complete-data log-likelihood, which includes both the observed data X and the hidden variables Z. This is done based on the current estimate of the model parameters $\theta^{(t)}$. Specifically, the expected value of the log-likelihood is computed with respect to the conditional distribution of the hidden variables Z given the observed data X and the current parameters $\theta^{(t)}$. This expectation can be written as:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{Z|X,\theta^{(t)}} \left[\log p(X, Z|\theta)\right] \tag{1.10}$$

Here, $\mathbb{E}_{Z|X,\theta^{(t)}}$ denotes the expected value over the hidden variables Z conditioned on the observed data X and the current parameters $\theta^{(t)}$ Dempster et al. (1977).

2. **Maximization Step (M-Step)**: In this step, the algorithm updates the parameter estimates by maximizing the expected complete-data log-likelihood $Q(\theta|\theta^{(t)})$ with respect to θ. Mathematically, this is represented as:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)}) \tag{1.11}$$

This maximization step produces updated parameter values $\theta^{(t+1)}$, which are then used in the next iteration of the algorithm Neal and Hinton (1998).

The process repeats until convergence, which occurs when changes in parameter estimates θ between successive iterations become negligibly small Dempster et al. (1977), Neal and Hinton (1998).

The EM algorithm is extremely effective in certain scenarios that are common in clustering using Gaussian Mixture Models (GMMs) or in scenarios where the parameters of Hidden Markov Models (HMMs) are being estimated. Although this algorithm achieves a local maximum of the likelihood function, it may fail to find the largest one globally. The selection of the initial parameter values can play a dominant role in the convergence and, eventually, the solution obtained.

The EM algorithm is particularly useful in scenarios involving missing data or latent variables, which is one of the key advantages of this approach, especially in real-world problems where missing values are common. EM is flexible and can be applied to various probabilistic models. Moreover, EM guarantees an improvement in

the likelihood function with each iteration, eventually converging to a local maximum Dempster et al. (1977).

Although the Expectation-Maximization (EM) algorithm offers several advantages, it also has certain drawbacks. Depending on the initial parameter values, EM may converge to local optima rather than the global maximum. Additionally, the algorithm can suffer from slow convergence and be computationally intensive, often requiring many iterations to reach convergence Dempster et al. (1977).

Literature Review and Related Work

Several studies have employed the **Expectation-Maximization (EM)** algorithm across diverse applications. **Dempster et al. (1977)** Dempster et al. (1977) introduced the EM algorithm, which has since been widely adopted for problems involving missing or incomplete data. Following this, **McLachlan and Krishnan (1997)** McLachlan and Krishnan (1997) explored its use in **Gaussian Mixture Models (GMMs)**, providing a detailed extension of the EM algorithm for clustering tasks. Their work highlighted EM's effectiveness in soft clustering, where data points can belong to multiple clusters probabilistically.

In **speech recognition**, **Rabiner (1989)** Rabiner (1989) applied the EM-based **Baum-Welch algorithm** for training **Hidden Markov Models (HMMs)**. This work has had a profound influence on sequence prediction tasks, setting the foundation for modern speech-to-text systems. **Shepp and Vardi (1982)** Shepp and Vardi (1982) pioneered the use of EM in **tomographic image reconstruction**, specifically in **Positron Emission Tomography (PET)**. Their method improved the accuracy of reconstructing images from incomplete or noisy projection data, which is critical in medical imaging.

In the field of **Natural Language Processing (NLP)**, **Brown et al. (1993)** used the EM algorithm in the development of the **IBM Models** for **statistical machine translation**, focusing on estimating word alignments between two languages. This method laid the groundwork for many advancements in machine translation. Additionally, **Blei et al. (2003)** Blei et al. (2003) applied EM to **Latent Dirichlet Allocation (LDA)**, enabling the discovery of hidden topics within large corpora of text. This work has been pivotal in text mining and topic modeling.

Several variants of the EM algorithm have also been proposed to address its limitations. **Neal and Hinton (1998)** Neal and Hinton (1998) introduced modifications like **Generalized EM (GEM)** and **Stochastic EM (SEM)**, aimed at improv-

ing computational efficiency and convergence properties. These variants have been applied to large-scale problems such as **social network analysis** and **big data clustering**. **Jordan et al. (1999)** Jordan et al. (1999) proposed the **Variational EM (VEM)** algorithm, an extension that approximates the posterior distribution of latent variables, making it suitable for more complex probabilistic models, such as **Bayesian neural networks**.

1.4 Stochastic Processes in Machine Learning

Stochastic processes are integral to addressing real-world problems where either the environment is inherently random or the solution must employ stochastic techniques. Many machine learning algorithms, such as decision trees and nearest neighbors, can incorporate stochasticity by introducing randomness during the learning process. Stochastic methods play a vital role in handling large datasets to reduce computational expenses and managing limited or noisy data to avoid overfitting Hastie et al. (2009), Goodfellow et al. (2016). By introducing randomness, these methods improve machine learning models' robustness and generalization capabilities, making them more adaptable to unseen data Bottou (2010a), Breiman (2001).

Using stochastic methods in machine learning is a statistical approach that incorporates randomness into the learning process. This randomness helps manage uncertainty and variability in both data and algorithms. The stochastic approaches can generally be grouped based on how they introduce randomness at various stages. For instance, **Stochastic Gradient Descent (SGD)** is one of the most widely used stochastic methods. In SGD, model parameters are updated iteratively using a randomly selected subset of data (often called a mini-batch) rather than the entire dataset. This introduces variability into the optimization process, which can help the algorithm escape local minima and speed up convergence, making it a popular choice for large-scale optimization Robbins and Monro (1951), Bottou (2010a).

Another group of methods involves ensemble learning techniques, such as **Random Forests** and **Bootstrap Aggregating (Bagging)**. These methods introduce randomness during the training phase by using different random subsets of the training data or by selecting different random subsets of features Breiman (2001). In **Random Forests**, for example, each tree in the ensemble is trained on a bootstrap sample (a random sample with replacement) of the training data, and a random subset of features is considered at each split. This randomness helps to decorrelate the

models, reducing variance and improving the generalization of the ensemble Friedman (2001), Hastie et al. (2009).

Stochastic methods are essential for building scalable, robust, and efficient machine learning models. By introducing flexibility and randomness, these approaches can effectively handle the inherent complexities of real-world data, helping models generalize beyond the training data. One of the key advantages of stochastic methods, such as mini-batch Stochastic Gradient Descent (SGD), is faster convergence compared to traditional full-batch gradient descent, which is especially valuable when dealing with large datasets Bottou (2010b). Additionally, the introduction of randomness reduces overfitting by allowing the model to escape local minima and explore a broader range of possible solutions, enhancing its ability to generalize Goodfellow et al. (2016), Breiman (2001).

Stochastic methods also offer significant scalability, making them ideal for large-scale machine learning problems. The inherent randomness allows models to process smaller batches of data simultaneously, significantly reducing computational costs while still achieving effective learning. However, the stochastic nature of these algorithms presents challenges. Stochastic updates can sometimes introduce noise into the learning process, making it harder to fine-tune the model and achieve precise optimization. Furthermore, these algorithms tend to be sensitive to hyperparameters such as learning rate and batch size, which require careful tuning to avoid convergence problems Bottou (2010b).

Despite their strengths, stochastic methods can converge to suboptimal solutions if not properly managed. The randomness that helps models avoid overfitting can also lead to noisy updates, potentially hindering the model's performance. Ensuring that the learning process remains stable requires careful tuning of the hyperparameters, and improper settings can lead to suboptimal convergence. However, with proper calibration, stochastic methods remain a cornerstone of modern machine learning due to their adaptability, efficiency, and ability to handle complex data environments.

Literature Review and Related Work

The stochastic approach has been applied in various ways to handle massive datasets. For example, Abdufattokhov et al. Abdufattokhov et al. (2019) applied a stochastic Gaussian process to manage large-scale data in big data environments. Their approach leverages the uncertainty and probabilistic nature of Gaussian processes, making them more scalable for extensive datasets. However, the computational

complexity of Gaussian processes, even in stochastic settings, remains a limiting factor for extremely large datasets.

Naik and Patel Naik and Patel (2021) presented a stochastic approach for automating the collection of training data from remote sensing images. Their method introduced stochastic elements into the data collection process, allowing for more efficient data gathering in unpredictable environments, such as natural scenes. However, the precision of the collected data is still subject to the inherent noise and unpredictability of the remote sensing process, suggesting the need for more robust filtering and noise-reduction techniques.

Nguyen et al. Nguyen et al. (2017) introduced the SARAH algorithm, a variance-reducing stochastic recursive gradient algorithm, which combines several existing optimization techniques to achieve better performance. SARAH reduces variance in gradient estimates, improving the stability and convergence of stochastic optimization. While the algorithm demonstrates promise in various applications, more work is required to integrate SARAH with other machine learning algorithms and models, particularly those that operate in real-time or require fast adaptation to changing data.

In another notable contribution, Ranganath et al. Ranganath et al. (2014) introduced stochastic variational inference (SVI) for large-scale Bayesian models. This approach allows for scalable inference in models that would otherwise be computationally infeasible to train, such as in complex graphical models. While SVI significantly improves the scalability of Bayesian inference, it still requires careful tuning and approximation methods, which can introduce biases into the results.

Liu et al. Liu et al. (2020) conducted a survey on stochastic computing for neural networks, emphasizing the trade-off between hardware efficiency and model accuracy. Their work showed that stochastic computing could reduce the hardware costs of neural networks while maintaining a reasonable level of accuracy. However, this approach introduces additional sources of error and variance, which need to be carefully managed to ensure that the model's performance does not degrade beyond acceptable limits.

Alaghi and Hayes Alaghi and Hayes (2013) reviewed stochastic computing techniques, particularly in the context of neural networks and hardware-efficient machine learning models. Their work highlights the potential for stochastic computing to drastically reduce power consumption and hardware resources, but also points out that the loss of precision and increased error rates are significant challenges that need

further attention.

Decision trees have also been explored in the context of stochastic approaches. Gouk et al. Gouk et al. (2019) applied stochastic methods to gradient-based decision trees, showing improvements in model flexibility and accuracy. However, the reliance on gradient information makes these approaches less suitable for highly non-differentiable or discrete problems.

Earlier work by Hespos and Strassmann Hespos and Strassmann (1965) introduced stochastic decision trees that combine the strengths of risk analysis and conventional decision tree methods. They provided a method to handle uncertainty in decision-making processes, particularly in fields such as finance and risk management. However, their approach was limited by the computational complexity of handling large, branching decision trees with many stochastic elements.

Hazen Hazen (1992) extended the stochastic tree framework to medical decision-making problems, demonstrating that rollback techniques could be used to determine optimal decisions. While this work laid a foundation for applying stochastic methods to decision trees, it remains primarily theoretical, with limited applications in modern machine learning.

1.5 Problem Statement

The rapid growth in data volume and complexity has revealed significant limitations in the efficiency of traditional machine-learning algorithms. For example, in agriculture, where sensor networks, satellite imagery, and other data sources generate vast, high-dimensional datasets, the scalability and robustness of these algorithms are critical. Among these, decision trees and nearest-neighbor algorithms—despite their popularity—face specific, well-documented challenges in handling large-scale, high-dimensional data.

Decision trees, widely valued for their interpretability and flexibility, encounter severe computational bottlenecks when applied to big data. The computational complexity of tree induction, which scales with $\mathcal{O}(DN \log N)$, becomes unmanageable as both the size of the dataset (N) and the number of dimensions (D) increase. While there have been advancements in improving the efficiency of tree-based algorithms, current methods largely fall short of delivering effective solutions that reduce computational costs without compromising accuracy or model performance. This limitation is particularly problematic in agricultural contexts, where the need for real-time

decision-making and large-scale deployment is essential.

In addition to decision trees, nearest-neighbor algorithms struggle with the notorious *"curse of dimensionality."* As the number of features (dimensions) increases, the concept of proximity—critical to nearest-neighbor methods—becomes less meaningful, significantly degrading performance. While data structures like KD-Trees, R-Trees, and clustering methods like k-means offer partial solutions, they do not fully address the scalability and optimization issues inherent in these models, especially in environments where data is both high-dimensional and noisy. In fields such as agriculture, where data is often multi-dimensional and subject to high variability due to environmental factors, these limitations pose significant barriers to effective application.

These challenges underscore the urgent need for developing new, scalable machine learning approaches that can overcome the computational and interpretative difficulties of both decision trees and nearest-neighbor methods. The limitations of traditional methods, particularly in the agricultural domain, motivate the exploration of stochastic methods that introduce randomness into the learning process, thus reducing computational overhead while preserving model accuracy and interpretability.

This dissertation aims to tackle these critical issues by proposing novel stochastic decision tree and synthetic reduced nearest-neighbor (SRNN) models that address the computational and scalability challenges of existing approaches. Specifically, the work focuses on improving the robustness and efficiency of these algorithms in high-dimensional, real-world datasets, with a particular emphasis on agricultural applications. By advancing the scalability of decision trees and enhancing the performance of nearest-neighbor algorithms in noisy, high-dimensional environments, this research contributes to both the theoretical development of machine learning methods and their practical applicability in precision agriculture, where predictive accuracy and computational efficiency are paramount. The summary of key challenges addressed are as follows:

1. **High Computational Demand of Decision Trees**: The computational complexity of tree induction, which grows with both the number of features (D) and the dataset size (N), poses scalability issues that hinder the effectiveness of decision trees in large-scale and real-time applications. This dissertation develops methods to handle this scalability issue that arises as a result of dealing with a large number of features and large data size.

2. **Curse of Dimensionality in Nearest-Neighbor Methods**: As dimensionality increases, the concept of proximity loses meaning, undermining the performance of nearest-neighbor algorithms. Current partial solutions like KD-Trees and clustering methods cannot fully address the demands of high-dimensional, noisy data environments. This dissertation presents solutions for solving this issue that arises from noisy and highly dimensional data.

3. **Need for Scalable, Efficient, and Interpretable Models in Precision Agriculture**: Agricultural data often exhibits high variability due to environmental factors, and traditional methods struggle to deliver both real-time responsiveness and computational efficiency in such contexts. This dissertation develops methods to deliver both real-time response and computational efficiency for agricultural data.

The focus of this research, therefore, is on developing new stochastic methods that address these challenges by enhancing computational efficiency and interpretability without compromising on model performance.

1.6 Motivation and Objectives

This research is motivated by the need to address critical gaps in the current machine learning methodologies, particularly with respect to scalability, computational efficiency, and the ability to handle high-dimensional datasets in real-world applications such as agriculture. While effective in controlled or smaller-scale environments, existing methods lack the robustness required for large-scale agricultural datasets that are often noisy and dynamic. The motivation comes from three primary areas:

- **Handling Large Datasets:** As data grows in both size and complexity, traditional decision trees and nearest-neighbor algorithms struggle with high computational costs. Despite some efforts to improve efficiency, there has been limited success in reducing the complexity of tree induction and nearest-neighbor searches in high-dimensional spaces. This research aims to tackle this issue by introducing stochastic methods into decision trees and synthetic nearest neighbors, leveraging randomness to improve scalability and reduce computational overhead.

- **Improving Theoretical Foundations:** Theoretical understanding of the limitations of nearest neighbor algorithms in high-dimensional environments and the under-explored potential of non-gradient-based stochastic decision trees motivates this work. By extending the Synthetic Reduced Nearest Neighbor (SRNN) approach through neural network integration and stochastic optimization techniques, this research seeks to improve the inference capabilities and computational efficiency of KNN models. Additionally, it introduces a novel stochastic approach to decision tree induction that addresses the scalability issues inherent in high-dimensional datasets.

- **Agricultural Applications:** Efficient water management in agriculture is critical given the sector's high demand for freshwater resources. This research focuses on developing intelligent, data-driven systems that can predict crop water needs using stochastic methods applied to large-scale sensor data. By combining stochastic methods' scalability with decision trees' interpretability and the precision of nearest neighbor algorithms, this work aims to contribute to sustainable agricultural practices through more efficient resource management.

These objectives reflect the need for more adaptable and scalable machine learning methods capable of addressing the specific challenges posed by large, high-dimensional datasets, particularly in sectors like agriculture, where the impact of such advancements can be substantial.

1.7 Dissertation Outline

The rest of the thesis is organized as follows:

In Chapter 2, we introduce the foundational concepts and notations necessary for understanding the proposed stochastic methods. This chapter presents an in-depth exploration of the Stochastic Decision Tree (SDT) model, which applies stochastic processes to enhance the learning of Haar trees. We thoroughly examine the structure and mechanics of decision trees and demonstrate how stochasticity can reduce computational complexity while maintaining or improving accuracy. This chapter also includes experimental results validating the efficacy of the proposed method on several datasets.

Chapter 3 presents a novel approach to synthetic reduced nearest-neighbor (SRNN) classification, leveraging neural networks to improve performance and in-

terpretability in high-dimensional data settings. We address the limitations of traditional nearest-neighbor algorithms, particularly the challenges posed by the curse of dimensionality, by integrating neural networks into the SRNN framework. This chapter details the architecture of the proposed method, its theoretical foundations, and the results of applying the approach to benchmark datasets.

In Chapter 4, we extend the SRNN framework by proposing an advanced Two-Layer Neural Network model. This chapter explores how the introduction of a modular, two-layer architecture enhances the generalization capabilities of SRNN models, especially in the context of image classification tasks. The chapter details the architectural choices, the learning process, and the computational benefits achieved through parallelization and modularization. We present experimental results demonstrating the superior performance of this model in reducing error rates while maintaining computational efficiency on high-dimensional image datasets like MNIST and Fashion-MNIST.

Chapter 5 focuses on applying stochastic decision trees in real-world agricultural problems. Specifically, we propose a predictive analysis model that utilizes stochastic decision trees to estimate stem water potential in almond and pistachio orchards. This chapter provides a comprehensive discussion of how the SDT model can optimize water usage in agricultural practices, contributing to sustainable water management. We analyze the experimental results using aerial and ground sensor data, highlighting the model's ability to make accurate predictions and improve irrigation efficiency.

Finally, Chapter 6 provides a summary of the key contributions and findings of this dissertation. We highlight the advancements made in both the theoretical development of stochastic methods for decision trees and nearest-neighbor models, as well as their practical applications in high-dimensional data environments, particularly in the context of agriculture. The chapter concludes with a discussion of potential avenues for future research and improvements to the methods proposed in this work.

Chapter 2

Stochastic Induction of Decision Trees with Application to Learning Haar Trees

For every supervised learning task, decision trees are a convenient and well-established solution. Decision trees are utilized in a variety of applications, including medical imaging and computer vision. Decision trees are trained by greedily splitting leaf nodes into a split and two leaf nodes until a halting requirement is met. Finding the best feature and threshold that minimizes a criterion is the technique for dividing a node. An exhaustive search algorithm is generally used to solve the criterion minimization problem. This exhaustive search strategy, however, is expensive, especially when the number of samples and features is large. To address this issue, a novel stochastic technique for criteria minimization is presented in this chapter. Asymptotically, the suggested approach is several orders of magnitude faster than a traditional exhaustive search. It is also demonstrated that the proposed method minimizes an upper bound for the criterion. The algorithm is tested against various state-of-the-art decision tree learning approaches, including the baseline non-stochastic approach. Despite being non-deterministic, the suggested method surpasses every existing decision tree learning strategy (including online and fast). It performs as well as the baseline algorithm in terms of accuracy and computational cost. We use the proposed technique to learn a Haar tree over the MNIST dataset, which contains more than $\$200,000$ features and $\$60,000$ samples, for empirical evaluation. This tree achieved a test accuracy of 94% on the MNIST dataset, surpassing any other known axis-aligned tree by 4%. Its performance is comparable to that of oblique trees while offering substantial improvements in both inference and training times.

2.1 Introduction

Decision trees are among the most well-established machine learning models in the scientific community, and they have been widely employed in a variety of applications such as computer vision and medical imaging Criminisi and Shotton (2013), Freund et al. (1999), Breiman (2001), Tavallali et al. (2019). Decision trees are widely applied as a sub-procedure in ensemble models such as boosting and random forests. Training a decision tree entails iteratively splitting a leaf node into one split and two leaf nodes Breiman et al. (1984), Gehrke et al. (1999). The split node is made up of a decision stump that sends samples to either its left or right child. A leaf node is divided by choosing the best feature and threshold that minimizes a given criterion.

Decision trees can be broadly categorized into two types: classification trees and regression trees, each serving different purposes in machine learning tasks. Classification trees are designed to categorize input data into predefined classes. For instance, a classification decision tree might be used to determine whether an email is spam or not spam. This type of tree is built by selecting splits that maximize the homogeneity of the categories within the resulting nodes. Common criteria for selecting the best split in classification trees include the Gini index and cross-entropy.

The Gini index is a measure of node impurity, with lower values indicating purer nodes. It is calculated as follows:

$$Gini(t) = 1 - \sum_{i=1}^{k} p_i^2 \tag{2.1}$$

where p_i is the probability of a particular class at node t, and k is the total number of classes. The Gini index's goal is to find the split that results in the lowest impurity, thereby creating nodes that are as homogeneous as possible.

Cross-entropy, also known as information gain, is another criterion used to evaluate splits. It measures the difference between the true class distribution and the predicted distribution at a node:

$$Cross - Entropy(t) = -\sum_{i=1}^{k} p_i \log(p_i) \tag{2.2}$$

Like the Gini index, cross-entropy seeks to minimize impurity, but it does so by maximizing the reduction in uncertainty after a split.

On the other hand, regression trees are used to predict continuous outcomes, such

as house prices, based on features like square footage and location. The objective of a regression tree is to minimize the variance of the target variable within each node after a split.

The criterion for splitting nodes is an impurity function, such as the Gini index Breiman et al. (1984), cross-entropy Quinlan (1986b), uniform piece-wise constant densities Ram and Gray (2011), Gaussian differential (continuous) entropy Criminisi and Shotton (2013), and others Yang and Wong (2014), Liu and Wong (2014), Sheikhi and Babamir (2018, 2016). The best feature and threshold are chosen using an incremental algorithm that assesses every potential feature and threshold combination Breiman et al. (1984). However, this approach becomes computationally expensive if the dataset is large or high dimensional. The computational complexity of solving the criterion minimization problem is $\mathcal{O}(DN \log N)$, where D and N represent the dimensionality and the number of samples, respectively. Because D and N are multiplied in the computational complexity expression, the computational complexity increases dramatically. In this study, we address the aforementioned issue by employing the splitting technique in an iterative and stochastic manner, removing less important features at each iteration.

Many proposals for speeding up the training operations of various learning algorithms may be found in the literature. Pre-processing the dataset by lowering the sample size or applying some feature selection technique is a popular approach Dash and Liu (1997), Jović et al. (2015), John et al. (1994). Pre-training feature selection may result in a less computationally expensive learning procedure Jović et al. (2015), Kohavi and Sommerfield (1995), Koller and Sahami (1996), Korn et al. (2001).

Filter, wrapper, and embedded techniques are the three feature selection approaches. The filter approaches involve picking features based on a desired criterion Jović et al. (2015). Filter techniques often employ statistical metrics such as minimum Redundancy Maximum Relevance (mRmR) Tang et al. (2014), correlation Yu and Liu (2003), chi-square Moh'd A Mesleh (2007), information gain Bhattacharyya and Kalita (2013), gain ratio Witten and Frank (2002), and so on. The wrapper approaches involve jointly selecting features and minimizing the loss function Bhattacharyya and Kalita (2013), Bradley and Mangasarian (1998), Maldonado et al. (2014), Hastie et al. (2009). However, because features are not guaranteed to contribute to the lower loss, feature selection as a preprocessing strategy may not result in an accurate final model Jović et al. (2015). Wrappers, on the other hand, are only applicable to quick modeling algorithms or greedy search methods, such as linear

SVM, Naive Bayes, and Extreme Learning Machines Liu et al. (2014), Cortizo and Giraldez (2006), Benoît et al. (2013). Embedded techniques employ a filter approach to choose a subset of features, followed by a wrapper approach to select the best candidate feature Guyon and Elisseeff (2003), Das (2001). As embedded methods, several decision tree algorithms such as CART, C4.5, and random forest are often utilized Breiman et al. (1984), Quinlan (2014), Sandri and Zuccolotto (2006). Embedded methods typically produce highly accurate results by combining filters and wrappers to improve accuracy.

Using feature selection as a preprocessing method reduces the computational cost of the training procedure. However, it is unsatisfactory in terms of the training's objective function and inefficient for big datasets because it does not reduce computation over the number of samples Rodriguez-Lujan et al. (2010).

Online learning is a well-known method for dealing with huge datasets Utgoff (1989). Several studies examine integrating online learning with decision trees, including Utgoff (1989), Schlimmer and Fisher (1986) and Wang et al. (2003). Such algorithms are simple yet fast and usually make few statistical assumptions. Online learning aims to update and train the model from a data stream Shalev-Shwartz et al. (2011). The model optimally gets updated at each time step regarding the most recently collected samples. Online learning is more computationally efficient than offline learning, which uses the full trainset for training Shalev-Shwartz et al. (2011). However, upon accessing a new batch of data, these techniques expand the size of the tree.

Despite the widespread use and application of decision trees, few techniques have focused on lowering the high computing cost of forming a tree Criminisi and Shotton (2013), Yates et al. (2018), Loh and Nowozin (2013). However, there is a lack of research when it comes to dealing with the high computational complexity of inducing the decision tree and examining potential improvements. None of the prior studies propose the computational cost of their algorithm, nor do they provide any analysis of the usefulness of features chosen at the split node. To fulfill this void, we evaluated SDT's computational complexity as well as its efficiency in locating near-optimal features at the split node. The authors of Zheng et al. (1998) proposed randomly selecting a subset of features at the split node and then determining the best feature among the set utilizing all samples present at the node. However, this method's optimality is not guaranteed due to its random nature.

2.2 Problem Statement and Basic Idea

The high computational cost derives directly from the exhaustive search algorithm being used to solve the split criterion. This approach becomes particularly computationally intensive when the dataset is huge and multidimensional. This is due to the computational cost of solving the splitting criterion, which is $\mathcal{O}(DN \log N)$ Hoare (1961), Breiman et al. (1984).

To address the challenges mentioned above, we offer a precise and fast Stochastic Decision Tree (SDT) induction approach that efficiently optimizes the splitting criterion. The suggested method begins at a node with an empty set S_t and all D features. The method iteratively picks a small random collection of samples from S_j (in order of $2^{-C} \times |S_j|$) and dismisses half of the less important features concerning S_t. The set and number of samples at the jth node are denoted by S_j and $|S_j|$, respectively.

SDT monotonically minimizes an upper bound of the splitting criterion for the best feature and threshold obtained at each iteration. Overall, the algorithm descends steeply toward the criterion function's minimum. The outcomes of the experiments section also indicate this. Mathematically, we demonstrate that the algorithm prioritizes more distinct features. Essentially, the more distinct a feature, the more likely it is to be chosen in the final iteration.

The algorithm for inducing a tree of depth Δ has a computational cost of $\mathcal{O}(\frac{\Delta DN \log \frac{N}{2^C}}{2^C})$, where C is a hyperparameter that we set to 10 in our experiments. SDT's computational cost is several orders of magnitude less than that of the original approach (with $\mathcal{O}(\Delta DN \log N)$), and its optimum implementation ($\mathcal{O}(\Delta DN + DN \log N)$). Our method is faster because it focuses on reducing the computational complexity caused by a large number of samples and features. SDT has a numerical advantage over other implementations when the sample and feature sizes are large (ND is very large).

Stochastic decision trees are a variation of traditional decision trees that incorporate randomness into the tree-building process to reduce computational complexity and enhance model generalization. Unlike traditional decision trees that exhaustively search for the best feature and split at each node, stochastic decision trees introduce randomness by either selecting a random subset of features or using a random threshold for splits. This randomness helps to avoid overfitting and reduces the computational cost, particularly in large datasets.

The concept of introducing randomness into decision trees gained popularity with

the development of ensemble methods like Random Forests Breiman (2001), where each tree in the forest is built on a randomly selected subset of features and data samples. This approach reduces the model's variance and improves its robustness by ensuring that no single tree dominates the decision-making process.

Recent research has explored various aspects of stochastic decision trees. For instance, Scornet et al. Scornet (2016) analyzed the consistency and convergence rates of Random Forests, demonstrating that stochastic elements can lead to both computationally efficient and statistically robust models. Fan et al. Fan et al. (2003) studied the asymptotic properties of stochastic decision trees, highlighting their potential for dealing with high-dimensional data and complex decision boundaries.

In this study, we build on these ideas and propose a Stochastic Decision Tree (SDT) induction method that efficiently optimizes the splitting criterion while maintaining a low computational cost. Our approach iteratively refines the set of features and samples considered at each node, thereby focusing computational resources on the most informative parts of the dataset. This iterative refinement leads to a significant reduction in the overall computational complexity, making SDT particularly suitable for large-scale applications.

In experiments, we found the proposed method achieved the same or higher accuracy as the original greedy algorithm Breiman et al. (1984), while outperforming all other relevant state-of-the-art algorithms by several orders of magnitude in terms of accuracy and complexity. We used our technique to learn a Haar tree over MNIST for empirical evaluation, and Haar Viola and Jones (2001) filters were used to extract $200,000$ features from the dataset. While its inference time was much lower, the Haar tree could reach competitive accuracy to more complex trees such as oblique and optimal oblique trees. In theory, an oblique tree requires $\mathcal{O}(D\Delta)$ operations to produce a prediction, whereas a Haar tree only requires $\mathcal{O}(D + \Delta)$ operations. In terms of size, the Haar tree requires $\mathcal{O}(2^{\Delta})$ space, whereas the oblique tree requires $\mathcal{O}(D2^{\Delta})$ space.

2.3 Preliminaries

2.3.1 Induction of a decision tree

The objective function that every decision tree tends to minimize is:

$$L_\alpha(T) = L(T) + \alpha \times leaves(T), \qquad \alpha > 0, \tag{2.3}$$

where, $L(T)$ represents the tree's (T) loss function over a train set, $leaves(.)$ is the number of leaves in a tree, and $\alpha > 0$ shows the coefficient of the cost complexity. Following the tree's construction, pruning is carried out to minimize the cost function $L\alpha(T)$ over the validation set. Finding the optimum value of the cost function is NP-complete Laurent and Rivest (1976). As a result, state-of-the-art tree-inducing algorithms involve greedily splitting a leaf node into a split and two leaf nodes. Assume a dataset S with N pairs of observations $\{(x_i, y_i)\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^D$ and $y_i \in \{1, 2, 3, ..., M\}$. At the j^{th} split node, the function $f_j(x)$ consists of a decision stump $f_j(x) = sign(x^p - th)$, where superscript p and th indicate the p^{th} feature and the threshold, respectively. During the growth phase of a decision tree, p and th are optimally found in terms of specified criterion Breiman et al. (1984). Typically, the tree is expanded until a stopping point or maximum depth is reached. A split node is constructed by minimizing the desired criterion over the samples of each child of the node.

$$\begin{aligned} \min_{f_j} \quad & \sum_{f_j=\{-1,1\}} \frac{|S_j^{f_j}|}{|S_j|} H(S_j^{f_j}) \\ s.t \quad & f_j(x) = sign(x^p - th) \end{aligned} \tag{2.4}$$

where, $S_j^{f_j}$ presents the set of samples at the left child ($f_j = -1$) or right child ($f_j = 1$) of a node j. S_j shows the set of samples present at the j^{th} node, and $|.|$ returns the number of samples at its input set. H is an entropy function. The optimum value of problem (2.4) can be found in $\mathcal{O}(DN_j logN_j)$ where D and N_j are features and number of samples at node j ($|S_j|$), respectively. This is accomplished by sorting all samples along each feature and finding the optimum threshold for each feature graduallyBreiman et al. (1984).

2.4 Proposed Method

Solving (2.4) has a considerably high computational complexity, particularly when the number of features and samples is big. In this section, we suggest a method that minimizes the problem (2.4) by several orders of magnitude quicker than the original approach Breiman et al. (1984).

Basic ideas: The problem (2.4) includes finding the appropriate feature to optimally divide the space into two halves. State-of-the-art algorithms use the exhaustive search described in Section 2.3.1 to select the optimal feature Breiman et al. (1984). In this subsection, we present a low-cost stochastic exhaustive search approach for solving (2.4). The concept is based on 3 ideas.

Idea #1 Each feature can be roughly ranked based on the objective function of (2.4) over a small randomly selected subset of the samples.

Idea # 2 A bigger group of samples is required for a precise ranking of the features.

Idea # 3 The good news is that only the most competitive features necessitate high precision. As a result of merging Ideas #1-3, the algorithm, at each iteration, consists of removing less significant attributes and randomly selecting more samples from the dataset. This technique progressively improves the ranking precision for immensely important features. This approach is known as the stochastic induction of decision tree (SDT).

The algorithm Considering the above-mentioned ideas, we are describing the proposed algorithm to solve criterion problem (2.4). The process starts by an empty set S_{j^t}, where $S_{j^c} = S_j$, and set of features $D^t = \{p\}_{p=1}^{D}$. p is the p^{th} feature. The algorithm is as follows:

Step 1 Randomly select and remove $\frac{N_j}{2^C}$ samples from S_{j^c} and add them to set S_{j^t}. C is a user-defined hyperparameter.

Step 2 $\forall p \in D^t$, at k^{th} iteration, find feature importance of p^{th} feature (FI_p^k) by solving

$$\begin{aligned} (FI_p^k)^{-1} = \min_{th} \quad & \sum_{f_j=\{-1,1\}} \frac{|S_{j^t}^{f_j}|}{|S_{j^t}|} H(S_{j^t}^{f_j}) \\ s.t \quad & f_j(x) = sign(x^p - th) \end{aligned} \tag{2.5}$$

where, $S_{j^t}^{f_j}$ is the set of samples from S_{j^t} directed to the left child ($f_j = -1$) or the right child $f_j = 1$. FI_p^k shows the feature importance of p^{th} feature at k^{th} iteration. It should be noted that the optimization occurs just over th in (2.5) and $(FI_p^k)^{-1}$ is equal to the optimum of the objective function in (2.5). Problem in (2.5) can be addressed efficiently in $\mathcal{O}(|S_{j^t}|log(|S_{j^t}|))$ Breiman et al. (1984).

Step 3 Based on FI_p^k, sort all the features in D^t and discard half of the features in D^t with lowest FI.

Step 4 For α times iterate over steps 1-3 . α is a user-defined hyperparameter.

Step 5 For only the remaining features in D^t over the S_j, solve problem (2.4) Practically, depending on the dataset, we choose C to be about 10 and α to be around $7 - 10$. Intuitively, C and α are the hyper-parameters that control the algorithm's computational complexity. C specifies the number of samples to be utilized in each iteration, while α specifies the number of features to be utilized in the final iteration. The computational complexity drops further as C and α are increased. However, there is a concern that the model will lose accuracy with excessively high C and α values. To reduce computational complexity, we retain the samples in S_{j^t} sorted for the remaining features (D^t) from the last iterations in Step 2. Only the new samples from Step 1 must be sorted and merged with S_{j^t}. The merging will take at most $|S_{j^t}|$. This process will preserve the computational complexity of step 2 on $\mathcal{O}(2D\frac{N_j}{2^C}log(\frac{N_j}{2^C}))$ for all α steps (assuming $\alpha \to \infty$).

2.5 Analysis of the Proposed Algorithm

This section provides the theoretical properties of the algorithm and their analysis.

2.5.1 Computational Complexity

The computational complexity of splitting a node and induction of a tree are analyzed in this section. Assume that the samples of S_{j^t} from past iterations are **NOT** kept sorted. Following the theorem, we will demonstrate how keeping samples sorted affects node training complexity and storage overhead Alizadeh et al. (2022a).

Theorem 1 (node training complexity)**.** *The computational complexity of splitting the node using SDT is* $\mathcal{O}(4DN_0logN_0)$*, where* $N_0 = \frac{N_j}{2^C}$*.*

Proof. At each iteration, N_0 samples are added to the set S_{j^t}, and half of the features are discarded. To solve (2.5), samples along each remaining feature have to be sorted,

which takes $\mathcal{O}(|S_{j^t}|logS_{j^t})$. At the k^{th} iteration, $|S_{j^t}|$ is $(k+1)N_0$. Therefore, the total complexity of steps 1-4 is as follows:

$$\sum_{k=0}^{\alpha} \frac{D}{2^k}(k+1)N_0 log(k+1)N_0 \tag{2.6}$$

The summation in (2.6) can further be simplified to:

$$\sum_{k=0}^{\alpha} \frac{D}{2^k}(k+1)N_0(logN_0 + log(k+1)) \leq DN_0(logN_0 + log(\alpha+1)) \sum_{k=0}^{\alpha} \frac{k+1}{2^k} \tag{2.7}$$

Using the formula for the geometrical series of $\sum_{k=0}^{\infty} \frac{k+1}{2^k} = \sum_{k=0}^{\infty} \frac{1}{2^k} + \sum_{k=1}^{\infty} \frac{1}{2^k} + \sum_{k=2}^{\infty} \frac{1}{2^k} + ... = 2 + 1 + \frac{1}{2} + \frac{1}{4} + ... = 4$ in the right hand side of inequality of (2.7), we have:

$$DN_0(logN_0 + log(\alpha+1)) \sum_{k=0}^{\alpha} \frac{k+1}{2^k} \leq 4DN_0(logN_0 + log(\alpha+1)) \tag{2.8}$$

Asymptotically speaking, the computational complexity is being driven by the term $4DN_0 logN_0$. Note that $log(\alpha+1)$ is relatively very small compared to $logN_0$ (Numerically speaking, N_0 is in order of $100 - 200$ while α is in order of $7 - 10$). Therefore, the computational complexity asymptotically becomes $\mathcal{O}(4DN_0 logN_0)$. □

The computational complexity of the state-of-art algorithm for solving equation (2.4) is $\mathcal{O}(DN_j logN_j)$, which is 2^C times larger than the proposed stochastic approach.

Effect of keeping the S_{j^t} sorted: Effect of keeping the S_{j^t} sorted is that the term $k+1$ is removed from the formula (2.6). Therefore, the computational complexity changes to $\mathcal{O}(2DN_0 logN_0)$. However, it will increase the space usage slightly because $((k+1)N_0$ new variables need to be saved to keep the label of samples along each feature).

Assuming a maximum depth of Δ for the tree, the computational complexity of inducing the whole tree will be $\mathcal{O}(\frac{\Delta 4DNlog\frac{N}{2^C}}{2^C})$. It is simple to demonstrate by utilizing the fact that each node's offspring receives a separate set of samples from its parent and the fact that total samples at the same depth are at most N. This complexity is less than the complexity of the baseline technique Breiman et al. (1984) by orders of magnitude (2^C).

2.5.2 Relation to Objective Function

This section presents the evaluation of the value of the objective function in (2.4) using the best combination of feature and threshold found by the SDT at each iteration over S_{j^t}. For simplicity purposes, in this section, the objective function of (2.4) is represented by $L(S_j)$. Consequently, $L(S_{j^t})$ represents error over the set S_{j^t}. In this section, we assume the used criterion is misclassification error $H(S_j^{f_j}) = 1 - p_{m^*}^{f_j}$, where $p_m^{f_j}$ and m^* represent the ratio of class m in the f_j child and majority class in the partition, respectively. All the theorems and analyses can be readily extended to Gini-index and Cross-entropy Breiman et al. (1984) by applying their corresponding differences. The differences and how to apply them are explained after the next theorem.

Theorem 2 (Monotonic decrease of an upper bound)**.** *The algorithm will monotonically decrease an upper bound over the objective function of* (2.4)*.*

Proof. At each iteration k, the following equality holds:

$$|S_j|L^k(S_j) = |S_{j^t}^k|L^k(S_{j^t}^k) + |S_{j^c}^k|L^k(S_{j^c}^k) \tag{2.9}$$

where, the superscript k for sets and $L^k(.)$ represents the set of samples at the k^{th} iteration and the value of error with optimum parameters for $S_{j^t}^k$ over the input set, respectively. The loss $L(.)$ is always bounded from above (e.g., for misclassification, it is 1) by a constant value L_{max}. By replacing $L^k(S_{j^c}^k)$ with L_{max} in (2.9), for iteration k we have

$$|S_j|L^k(S_j) \leq |S_{j^t}^k|L^k(S_{j^t}^k) + |S_{j^c}^k|L_{max} = |S_j|L_{upper}^k(S_j) \tag{2.10}$$

At each iteration, the first term in (2.10) is minimized. Therefore, the new optimum parameters at each iteration provide lower error to the first term of (2.10) than the optimum parameters of previous iteration which is $|S_{j^t}^{k+1}|L^k(S_{j^t}^{k+1}) \geq |S_{j^t}^{k+1}|L^{k+1}(S_{j^t}^{k+1})$. Note that the superscript of the loss L is different for both sides of the inequality. Also, $|S_{j^c}^{k+1}| < |S_{j^c}^k|$. Therefore,

$$\frac{|S_{j^t}^{k+1}|L^{k+1}(S_{j^t}^{k+1}) + |S_{j^c}^{k+1}|L_{max}}{|S_j|} = L_{upper}^{k+1}(S_j) \leq L_{upper}^k(S_j) \tag{2.11}$$

□

For the case of Gini-index or Cross-entropy, formula (2.9) should be replaced with $L^k(S_j) \leq L^k(S^k_{j^t}) + L^k(S^k_{j^c})$. The reason for this change is that the algorithm finds optimum over $S^k_{j^t}$ and $min(L^k(S^k_{j^t}), L^k(S^k_{j^c})) \leq L^k(S_j) \leq max(L^k(S^k_{j^t}), L^k(S^k_{j^c}))$.

Average Decrease The Theorem (2) is based on worst case scenario. The proof of Theorem (2) assumes that the samples selected at each iteration do not have any correlation with the rest of the samples. In reality, this is not necessarily correct, and a set of S_{j^t} can partially represent the whole trainset S_j, and as the size of S_{j^t}, increases through iterations, the FI_p is measured more correctly. In our experiments, we observed a very steep decrease in $L^k(S_{j^t})$ after each iteration, and after several iterations, it almost converged to the loss of the best possible feature and threshold.

Theorem 3 (Probability of discarding a feature)**.** *Assume a binary classification dataset where samples are generated from some distribution along each feature, and features are independent. We assume the samples of both classes have varying degrees of overlap along each feature. The probability of a sample being generated from the overlap area is P^p_o for p^{th} feature. The probability of discarding a feature depends on P^p_o.*

Proof. Picking N_0 samples randomly from an infinitely large dataset (N_j) is equivalent to direct sampling from the distribution itself. Selecting samples is the same as flipping a coin with a probability of P^p_o to be 1 and $1 - P^p_o$ to be 0. The expectation of observing 1 is P^p_o, and essentially, on average, we expect to observe $P^p_o \times N_0$ samples from the overlap area, which is the region along a particular feature where the samples from two different classes are not clearly distinguishable from each other. The probability of deviating from the average case and observing FI^k_p being worse than the true FI_p (please note that FI corresponds to the goodness of a feature) over the whole trainset is equivalent to the probability of observing **at least** 1 extra (observing $P^p_0 \times N_0 + 1$ from overlap area) sample from the overlap area. Therefore, $P(FI^k_p < FI_p) = 1 - (1 - P^k_o)^{N_0(1-P^k_o)}$ (or 1− probability of observing no extra sample from overlap area), which is an upper bound to the probability of discarding a feature. □

Since samples with lower overlap area (lower P^p_o) are more distinctive, consequently, such features have greater FI^k_p, and the probability of overlap area P^p_o is inversely related to FI^k_p. According to the Theorem 3, features with greater true FI_p are more likely to stay in iterations on average. Assume a feature that totally

separates samples from both classes ($P_o^p = 0$); any random group of samples will represent this differentiation because there is no overlap. Such a feature will accomplish $FI_p^k = \infty$. Therefore, it will remain in the D^t.

Theorem 4 (Consistency of SDT)**.** *The SDT is consistent and can learn the function that generated the classes. This theorem is a result of theorem 20.1 and Lemma 20.1 in Devroye et al. (2013).*

Proof. The consistency of the Stochastic Decision Tree (SDT) can be established by leveraging the results from Theorem 20.1 and Lemma 20.1 in Devroye et al. (2013). Theorem 20.1 shows that the k-spacing classifier is consistent under the condition that $k \to \infty$ and $k/n \to 0$ as the number of samples, n increases. This implies that the expected classification error of the classifier converges to the Bayes risk L^* as the sample size grows indefinitely.

In the context of SDT, the partitioning of the input space is analogous to the k-spacing partitioning in Theorem 20.1, where the partitions are formed based on a small subset of the data. Lemma 20.1 further supports this by showing that for any binary tree classifier constructed on k regions, the number of data points in each region grows sufficiently large as n increases, provided that $k/n \to 0$. This ensures that the SDT can accurately estimate the class distributions within each region.

By satisfying these conditions, the SDT effectively captures the underlying structure of the data, allowing it to learn the true function that generated the classes. Therefore, the SDT is consistent, as it asymptotically approaches the optimal decision rule, thereby proving Theorem 4.

□

Theorem 5 (Decrease of the cost)**.** *The objective function of* (2.4) *decreases as new nodes are added.*

Proof. Assuming the data is IID (Independently and Identically Distributed), the objective function in equation (3.4) is bounded below by 0, since impurity measures like entropy or Gini index are non-negative. At each node in the decision tree, the algorithm selects a partition that minimizes the objective function by finding the optimal feature and threshold. This process ensures that the impurity of the child nodes is lower than or equal to that of the parent node, resulting in a decrease in the overall objective function. As new nodes are added, the objective function decreases progressively, thereby reducing the overall error. This confirms the validity of Theorem 5 □

It is worth mentioning that both the theorems of 4 and 5 are true for original decision trees (such as CART) under some mild assumptions, and here we show that SDT follows them too.

2.6 Experiments, Results, and Discussion

In this section, we give experimental results to demonstrate the effectiveness of our proposed approach. The SDT is compared to numerous other cutting-edge algorithms, as well as the baseline tree induction. The Figures 2.1 through Figure 2.6 show the error ratio of the train set and test set versus the computational complexity of each considered algorithm on different datasets mentioned in Table 2.1. The studied existing methods include adaptive re-sampling Iyengar et al. (2000), feature selection as a pre-training procedure Tang et al. (2014), Moh'd A Mesleh (2007), C4.5 Quinlan (2014), and CART Breiman et al. (1984). The studied adaptive re-sampling approach consists of training a tree from scratch iteratively with respect to a small collection of samples and increasing the set with incorrectly classified samples at each iteration by the tree. We used adaptive re-sampling in two different scenarios. In one configuration, we used adaptive re-sampling of Iyengar et al. (2000) until the algorithm achieved higher or comparable accuracy to SDT (Adaptive re-sampling 1). We performed the same processes in the second setup as Iyengar et al. (2000) (Adaptive re-sampling 2). We used ChiSquare and mRmR Tang et al. (2014), Moh'd A Mesleh (2007) as a pre-training setup for feature selection. The pre-training feature selections were used to choose 1%, 50%, and 90% of the features.

The C4.5 was trained using the same configuration as Quinlan (2014). The CART algorithm Breiman et al. (1984) is represented by the decision tree curve. All of the trees were trained for depths ranging from 5 to 12. The number at each point on the curve in Figure 2.1 through Figure 2.6 represents the depth of the tree. The error ratio is displayed on the vertical axis, and the computational complexity of the trained model is displayed on the horizontal axis. We set $N_0 = 20$ for all experiments for SDT. The SDT iterating terminates when only around 0.5% of the total features remain in D^t.

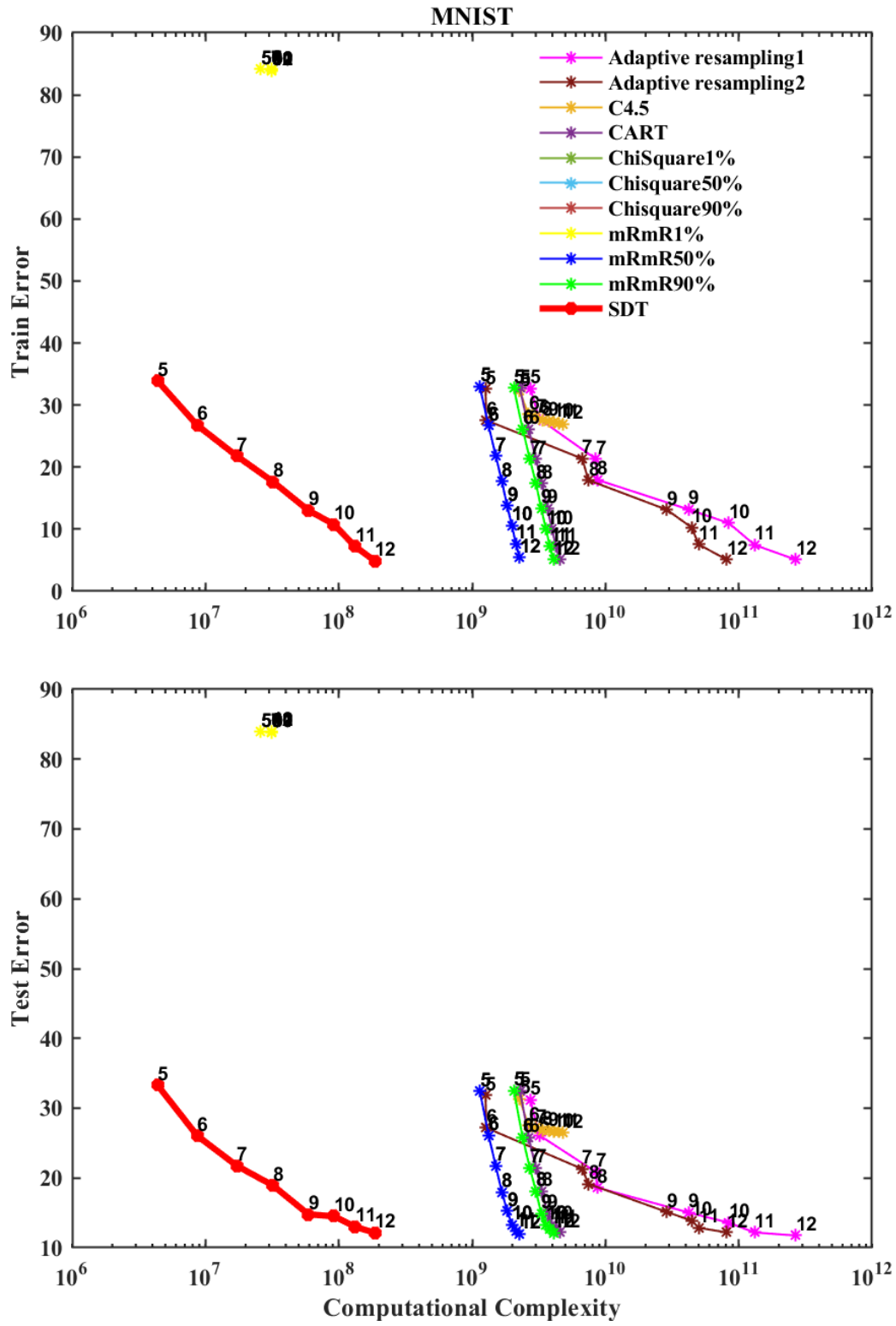


Figure 2.1: Training and Testing Error ratio versus computational complexity for MNIST dataset.

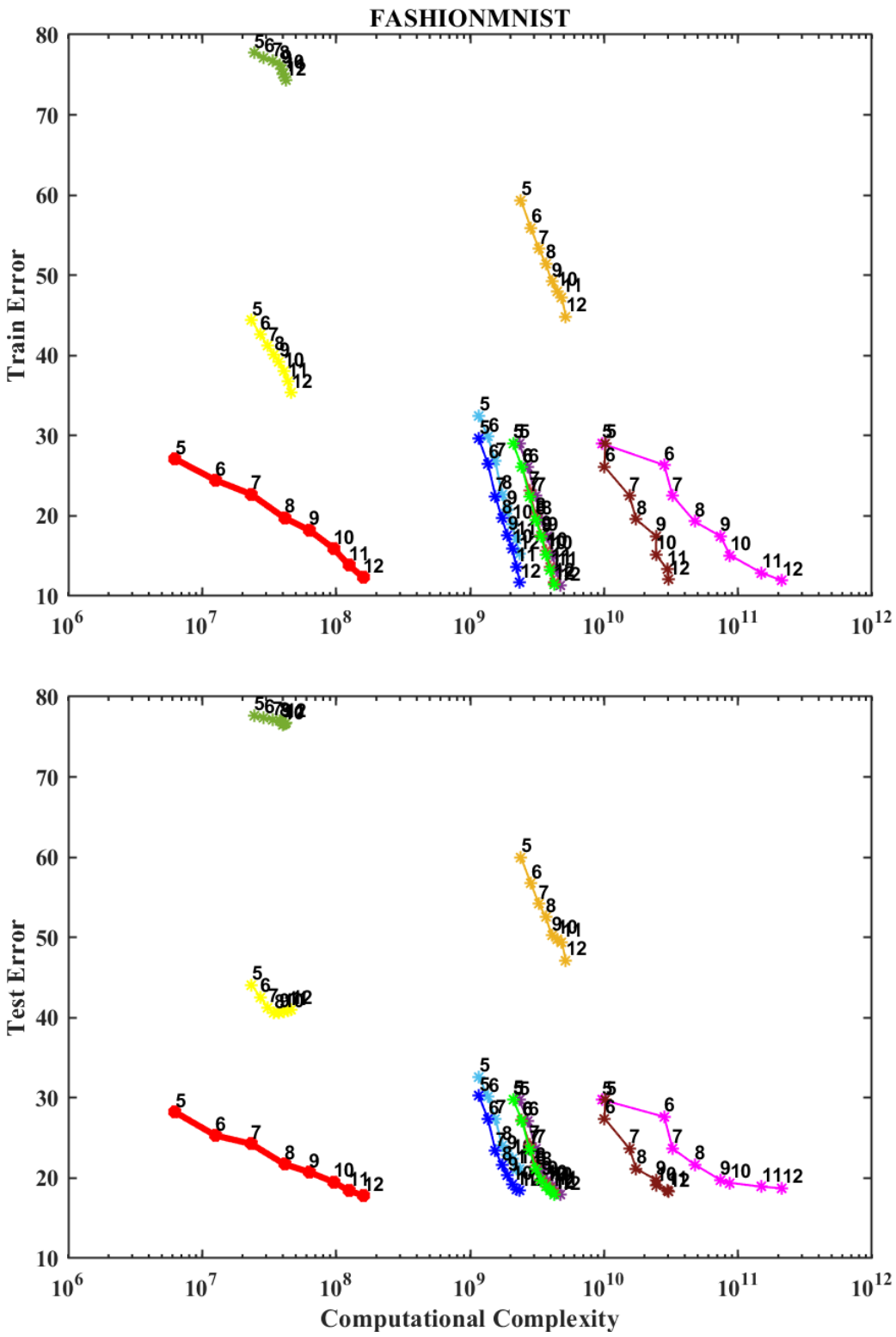


Figure 2.2: Training and Testing Error ratio versus computational complexity for FASHIONMNIST dataset.

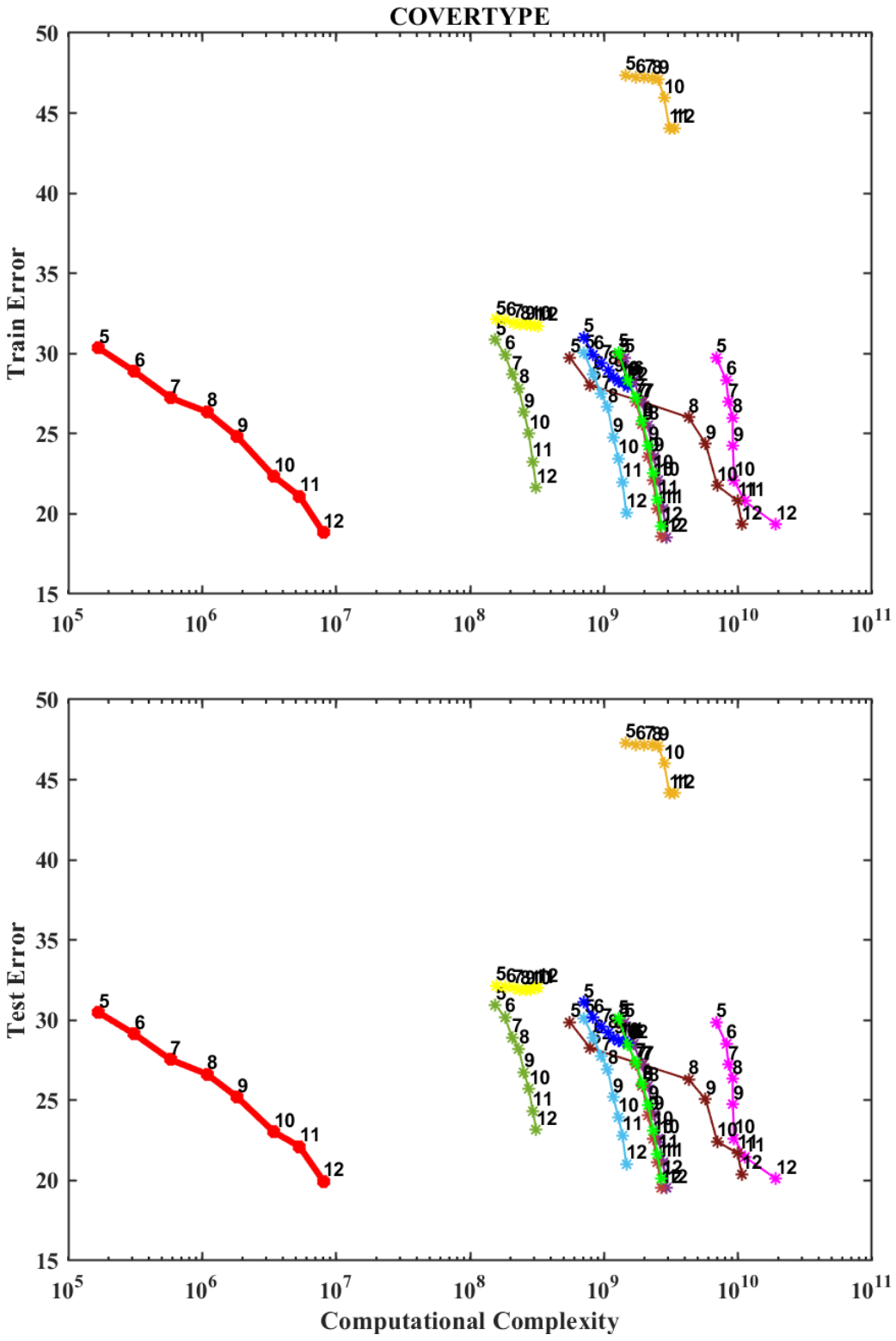


Figure 2.3: Training and Testing Error ratio versus computational complexity for COVERTYPE dataset.

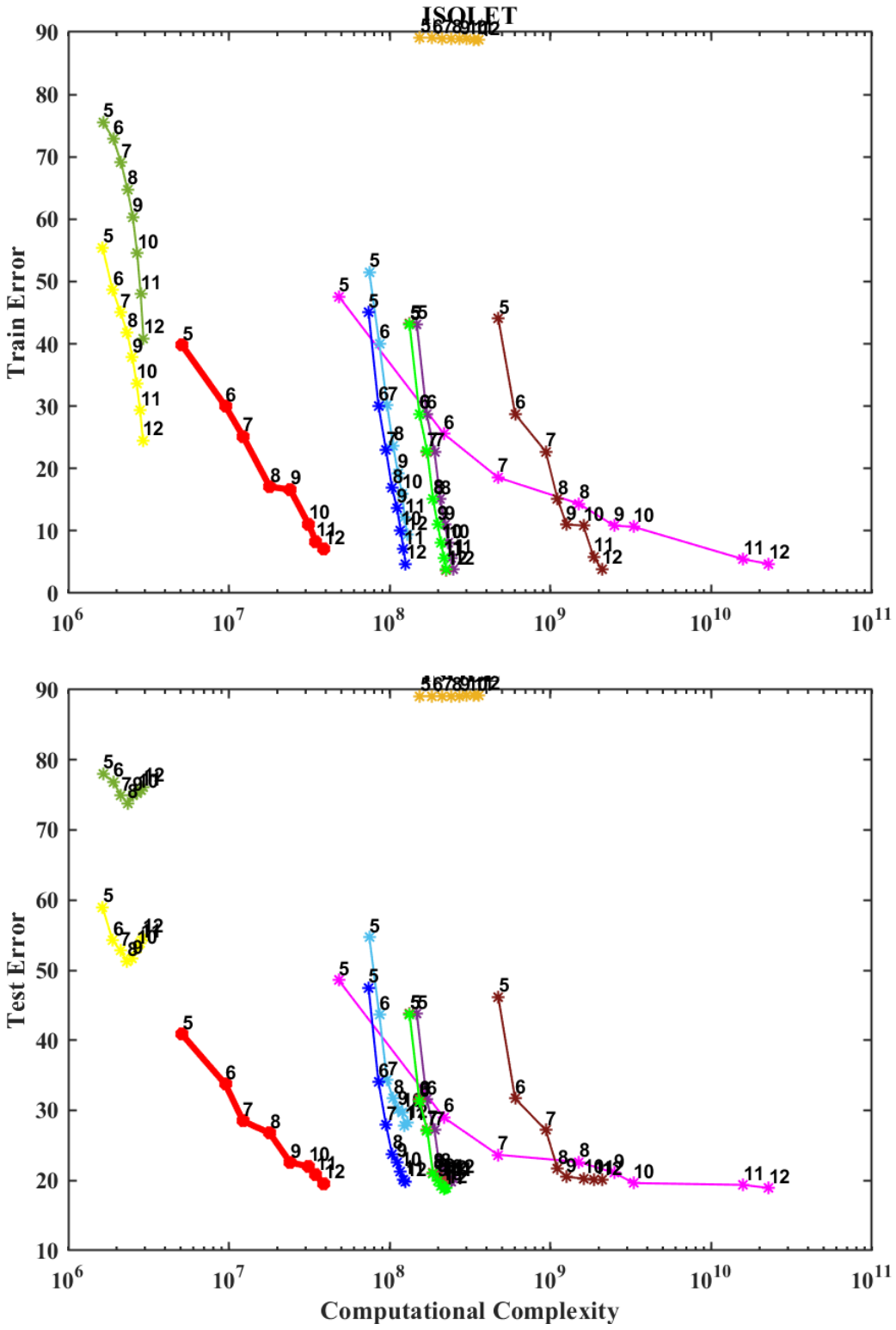


Figure 2.4: Training and Testing Error ratio versus computational complexity for ISOLET dataset.

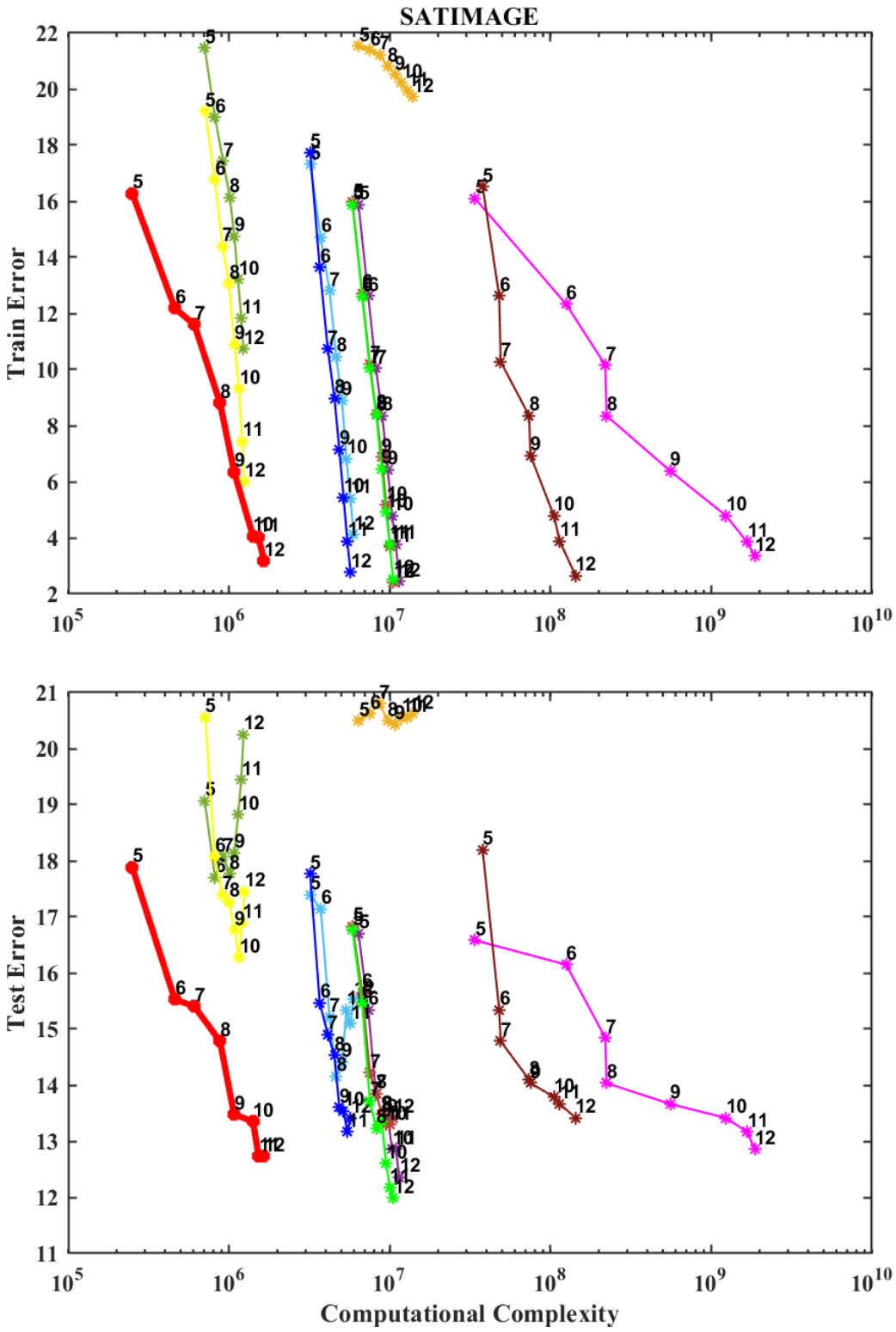


Figure 2.5: Training and Testing Error ratio versus computational complexity for SATIMAGE dataset.

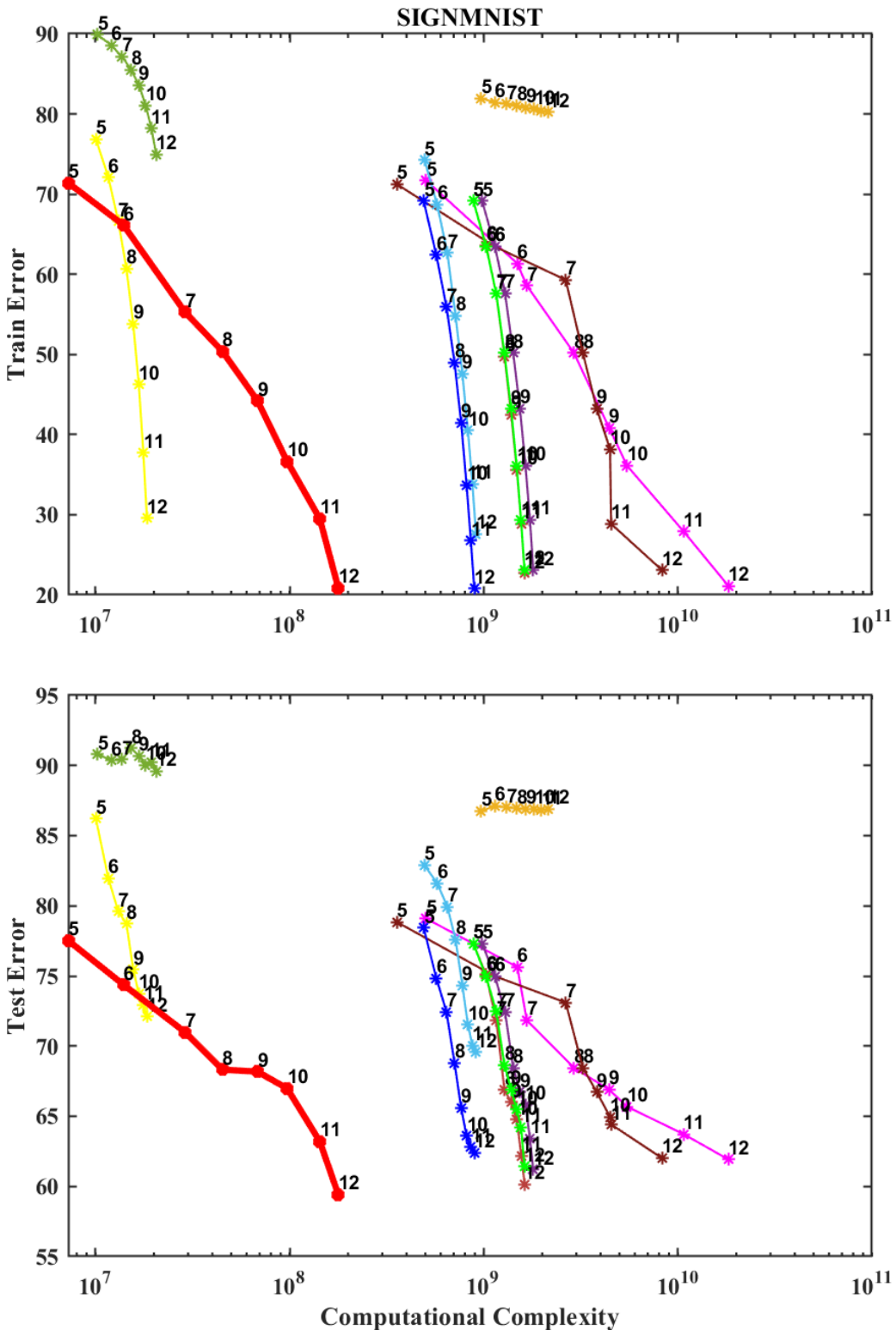


Figure 2.6: Training and Testing Error ratio versus computational complexity for SIGNMNIST dataset.

Table 2.1: The used datasets Information

Index	Name	Size		Description
		Train	**Test**	
1	MNIST	60000 × 784	10000 × 784	A large database of handwritten digits 0 to 9 that is serving as a basis for classification algorithms
2	FASHIONMNIST	60000 × 784	10000 × 784	A dataset of Zalando's article images containing 10 different classes of gray images of T-shirt/top Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot
3	SIGNMNIST	27455 × 784	7172 × 784	The American Sign Language letter database of hand gestures represents a multi-class problem with 24 classes of letters (excluding J and Z, which require motion)
4	ISOLET	5238 × 617	1559 × 617	ISOLET (Isolated Letter Speech Recognition) involves the name of each letter of the alphabet spoken twice by 52 speakers, suitable for a noisy, perceptual task
5	SATIMAGE	4825 × 36	1610 × 36	The database consists of multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, with classification associated with the central pixel in each neighborhood
6	COVERTYPE	435759 × 54	145253 × 54	This dataset contains tree observations from four areas of the Roosevelt National Forest in Colorado, containing cartographic variables

SDT outscored competing rapid tree training techniques in both accuracy and computational complexity, as shown in 2.1 and 2.2. SDT's training complexity is several orders of magnitude smaller (10^{2-3}). We should remark that SDT has the same accuracy as CART while using less computing cost. As a result, SDT accomplished its goal of reducing training complexity while preserving accuracy.

2.6.1 Haar Tree

Haar-like features are well-known in image processing literature and have been widely employed for face detection Viola and Jones (2001), Jones and Viola (2003), Demirkir and Sankur (2004), Mita et al. (2005), Pham and Cham (2007), object detection Lienhart and Maydt (2002), Park and Hwang (2014), Chen et al. (2007), Choi (2012) and ensembles Moghimi et al. (2018), Kwak et al. (2013), Larios et al. (2010). The Haar-like features are created by convolving various sizes of the Haar filters with the image. The next section contains more detailed information about Haar's features. However, because Haar-like features extract a large number of features, training a model with these features can be difficult and expensive. SDT is meant to handle such a difficult task, and we used it to train trees over Haar-like features. We call such a tree a "Haar tree." To extract additional features, we applied Haar-like features to MNIST and FMNIST. The size of each Haar filter region is $(3n, 3m)$ for $m, n = 1, 2, ..., 8$. 121000 features were retrieved from images of 28×28.

2.6.2 Haar Features

Figure 2.7 shows the exact haar features utilized to generate new features. The sizes of row pixels and column pixels are written next to each filter. For each filter, different filters with every conceivable row and column pixel combination are produced and convolved with the image to extract new features.

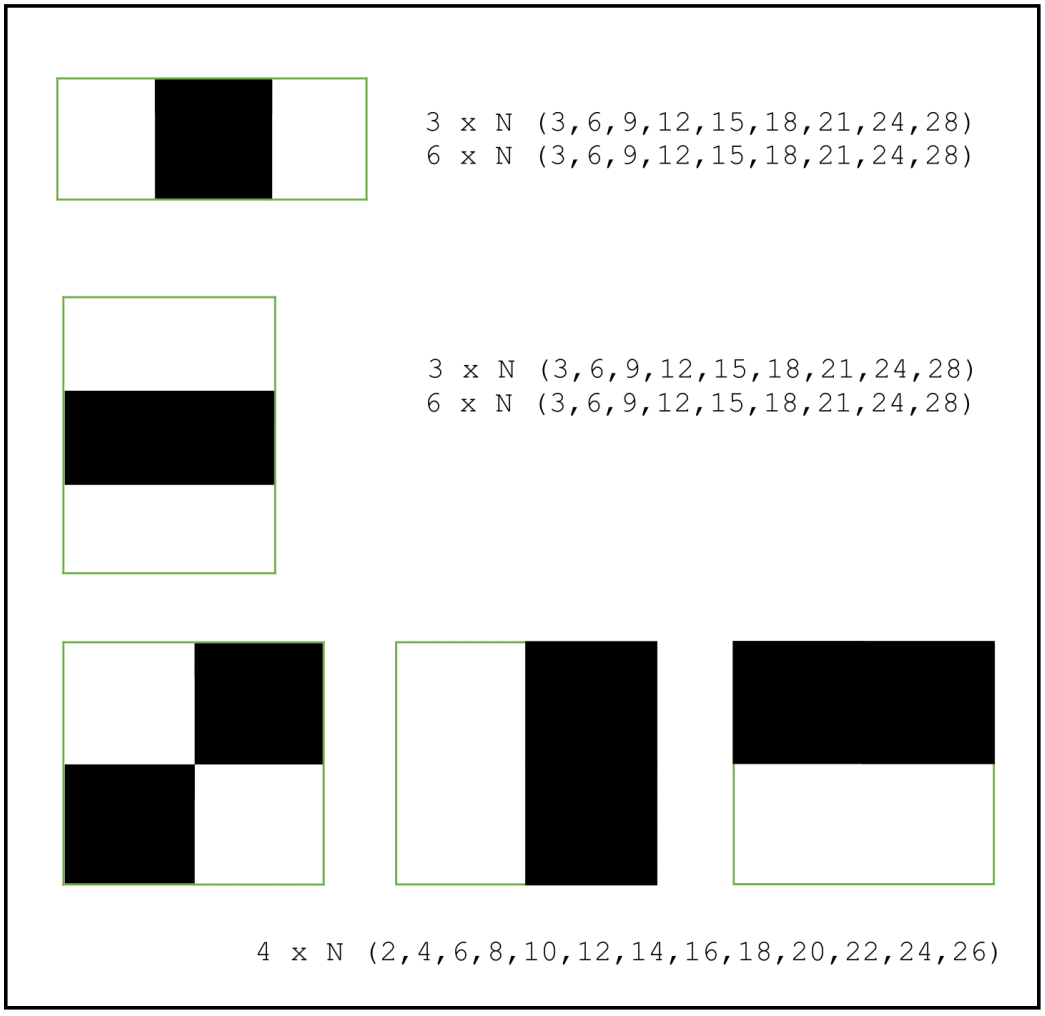
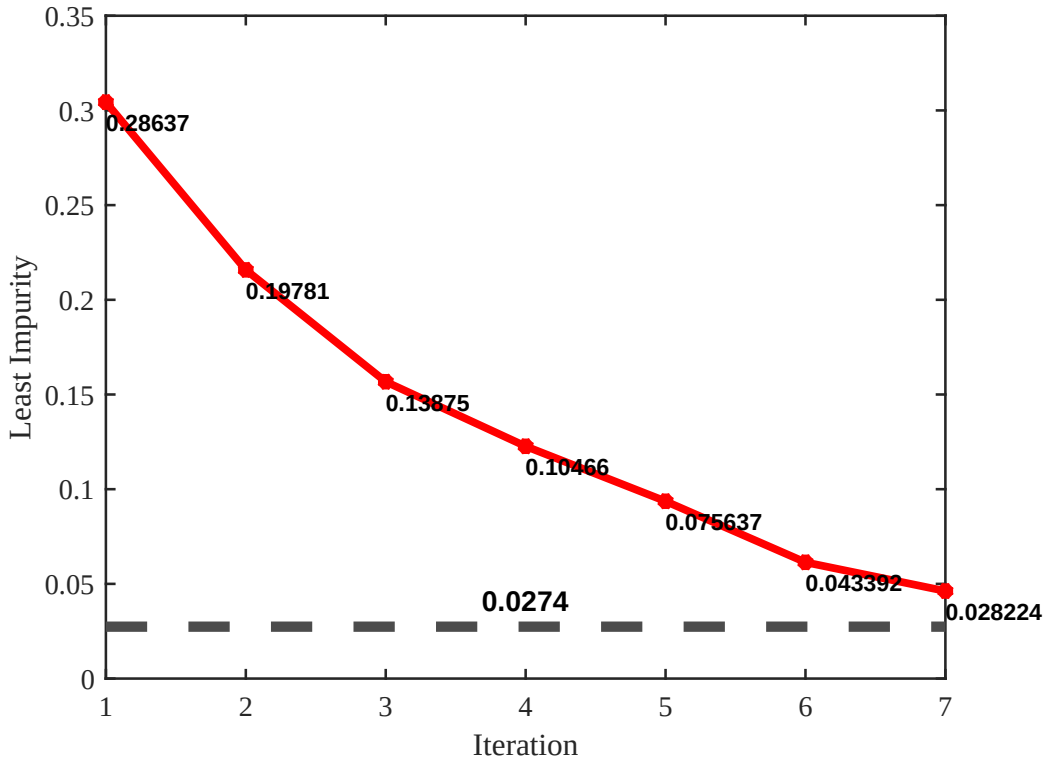


Figure 2.7: Haar Filters

Figure 2.8 and Figure 2.9 present the Impurity reduction over different iterations of SDT for MNIST. In the context of decision trees, the term "reduction in impurity" refers to the decrease in the measure of how mixed the classes are within a node (as defined by the criterion in formula 2.4). The horizontal axis shows the iteration number of SDT for a single node. The vertical axis shows the impurity. The dashed line represents the impurity of the best feature and threshold. The points on the curves represent the exact impurity of the best split found by the algorithm at that iteration. The text above each plot shows the classes used for creating the split node. As can be seen, at each iteration, the loss function has decreased drastically and eventually has converged to the true best feature and threshold.

(a)



(b)

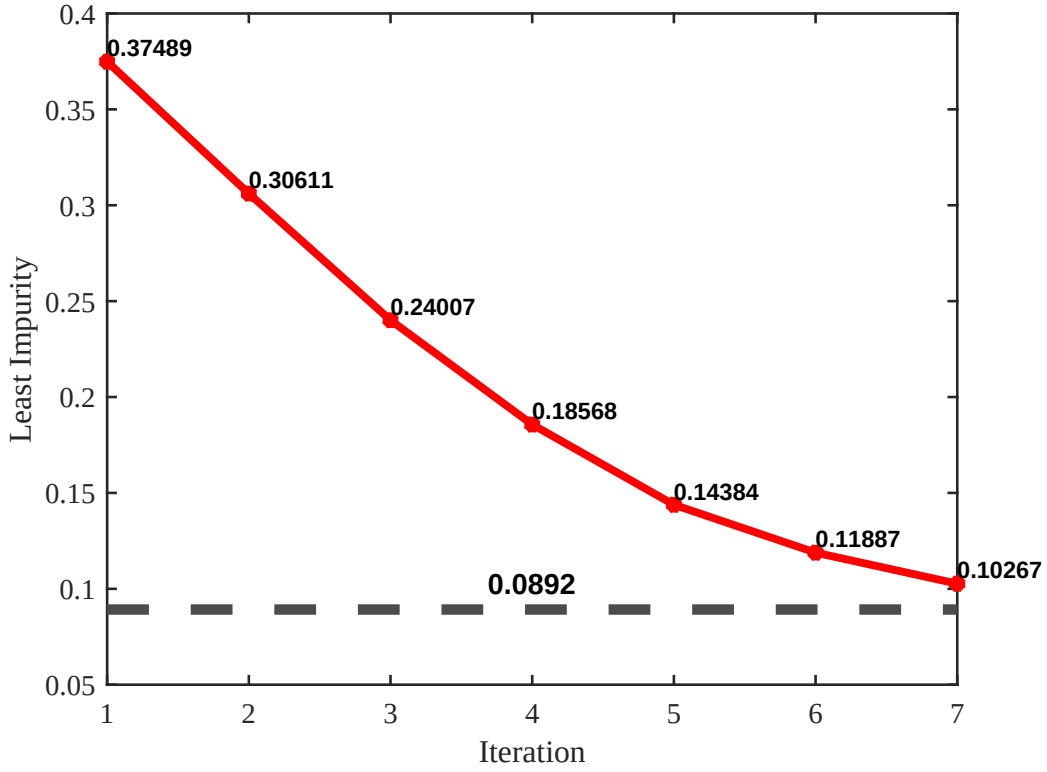
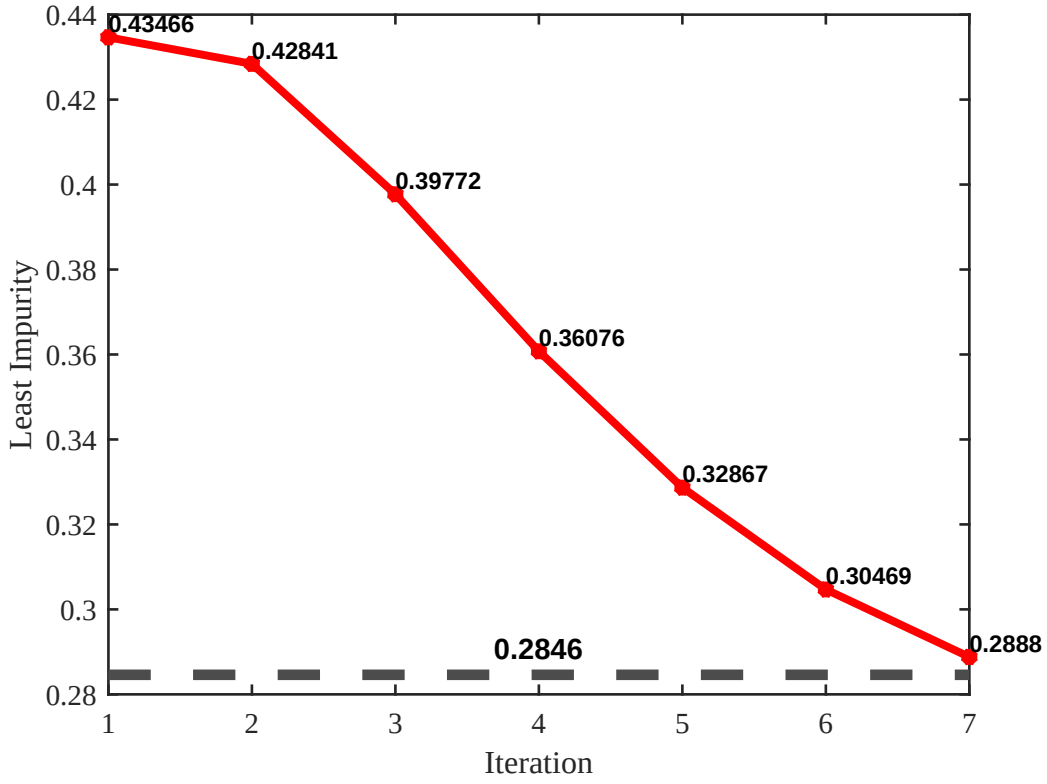


Figure 2.8: Impurity reduction at each iteration for (a) 1 versus 0, (b) 1 versus 7.

(a)



(b)

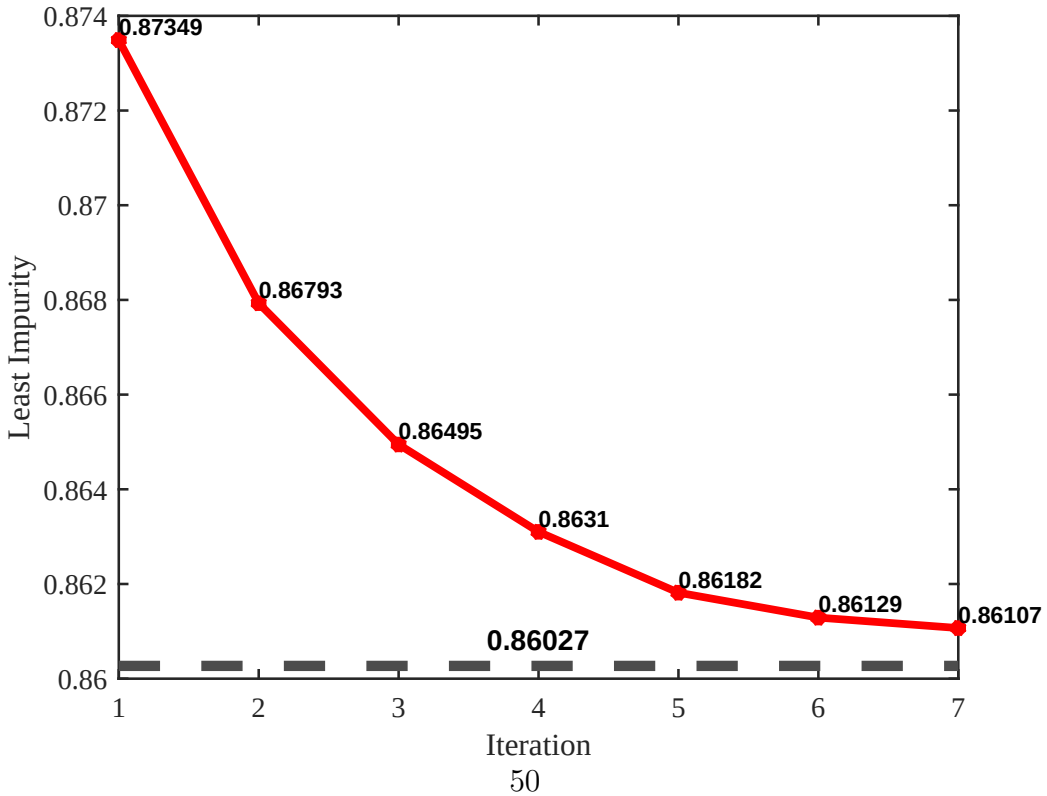


Figure 2.9: Impurity reduction at each iteration for (a) 5 versus 8, (b) all classes.

Figures 2.10 and 2.11 present the feature importance at each iteration of inducing a single split node for the two class pairs (0 vs. 1 and 1 vs. 7) from the MNIST dataset. In these figures, feature importance quantifies the contribution of each feature to the predictive power of the model, indicating how influential a particular feature is in determining the split decision at each iteration of the Stochastic Decision Tree (SDT). The feature importances were normalized to a range between 0 and 1, where brighter pixels represent higher importance values, and darker pixels represent lower values.

As the iterations progress, the figures illustrate how SDT effectively prioritizes more informative features while gradually reducing the weight assigned to less relevant ones. This process is visualized through the increased brightness of certain regions in the plots, corresponding to features that are consistently selected as critical for decision-making. The clear differentiation in the feature importance patterns across iterations demonstrates that SDT is capable of honing in on the most discriminative features for each specific class pair, thereby improving the accuracy and efficiency of the model.

Moreover, these figures serve as visual proof that SDT's iterative process not only identifies but also emphasizes the most compelling features for classification while discarding those that contribute minimally to the decision boundary. This selective emphasis on influential features ensures that the model becomes more robust and interpretable as it converges to an optimal set of features, highlighting SDT's ability to balance computational efficiency with predictive accuracy.

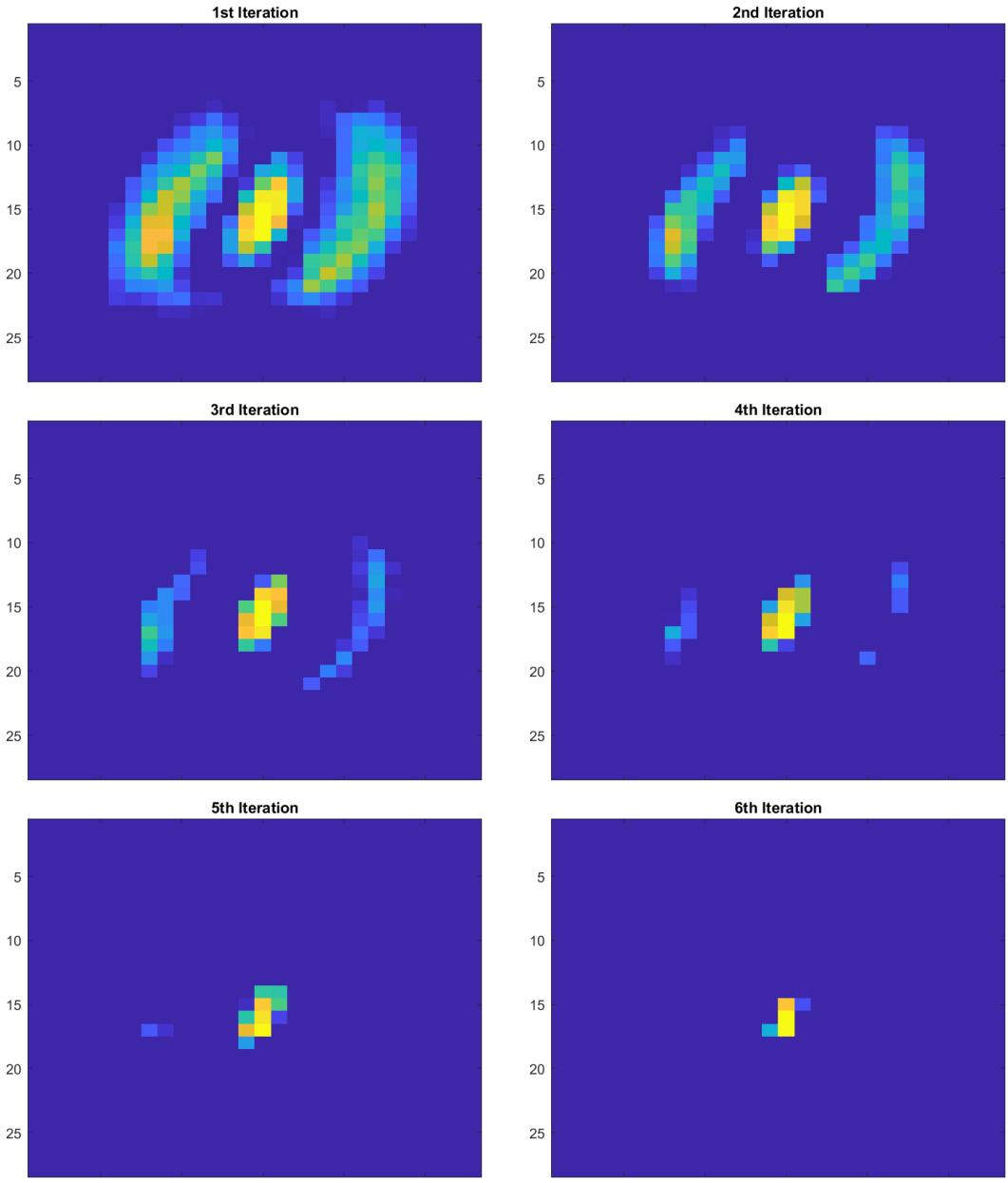


Figure 2.10: Feature importance (FI) of features by SDT at each iteration for the problem of 0 versus 1. Brighter pixels show a higher value of FI, and darker pixels show a lower value of FI.

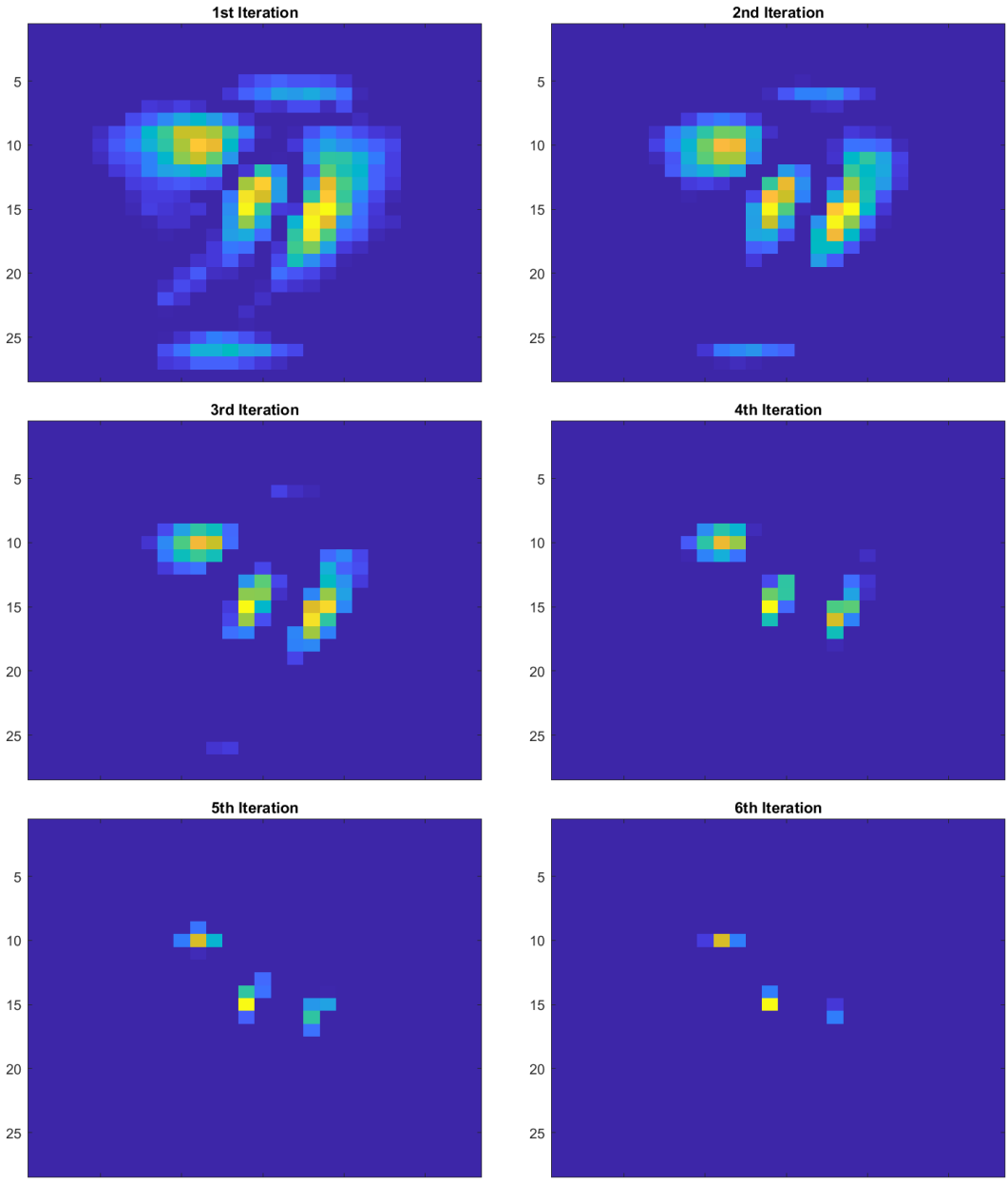


Figure 2.11: Feature importance of features by SDT at each iteration for the problem of 1 versus 7. Brighter pixels show a higher value of FI, and darker pixels show a lower value of FI.

The computational complexity versus error ratio for both training and testing is presented in Figures 2.12 through 2.17. These figures analyze the performance of the Stochastic Decision Tree (SDT) model across different datasets by exploring the effect of varying N_0, the hyperparameter that determines the subset size and influences the final tree structure.

Each curve in these figures is color-coded according to the N_0 value, which is defined in the plot's legend. The numbers along the curves indicate the depth of the tree at various points, providing insight into how the tree's complexity evolves during training. These visualizations highlight how computational complexity influences error rates and show the interaction between tree depth and N_0 in optimizing model performance. The patterns observed across different datasets underscore the SDT's ability to balance computational efficiency with predictive accuracy by fine-tuning N_0 and tree depth.

The results demonstrate that SDT consistently achieves low error rates across a variety of datasets while maintaining computational efficiency. As N_0 increases, the overall error—both training and testing—tends to decrease, though at the cost of higher computational complexity. This trend is particularly evident in datasets like MNIST and FashionMNIST (Figures 2.12 and 2.13), where increasing N_0 results in deeper trees that improve prediction accuracy, though with an accompanying rise in computational burden. However, SDT excels at finding an optimal balance, where modest increases in computational cost lead to significant improvements in accuracy, outperforming simpler models at comparable complexity levels.

The tree depth annotations on the curves offer additional insights into the relationship between tree depth, complexity, and accuracy. For instance, deeper trees improve performance on high-dimensional datasets like ISOLET (Figure 2.16), but gains diminish beyond a certain point, raising concerns about overfitting. This flexibility in tree depth allows SDT to adjust model complexity based on dataset characteristics, thereby preventing overfitting while still capturing essential patterns in the data.

Overall, these figures illustrate that SDT is highly adaptable across datasets with varying characteristics. The balance between error minimization and computational complexity can be fine-tuned by selecting the appropriate N_0 value, depending on the specific needs of the application. This adaptability makes SDT suitable for both small-scale problems, where computational efficiency is critical, and large-scale problems, where predictive accuracy is paramount.

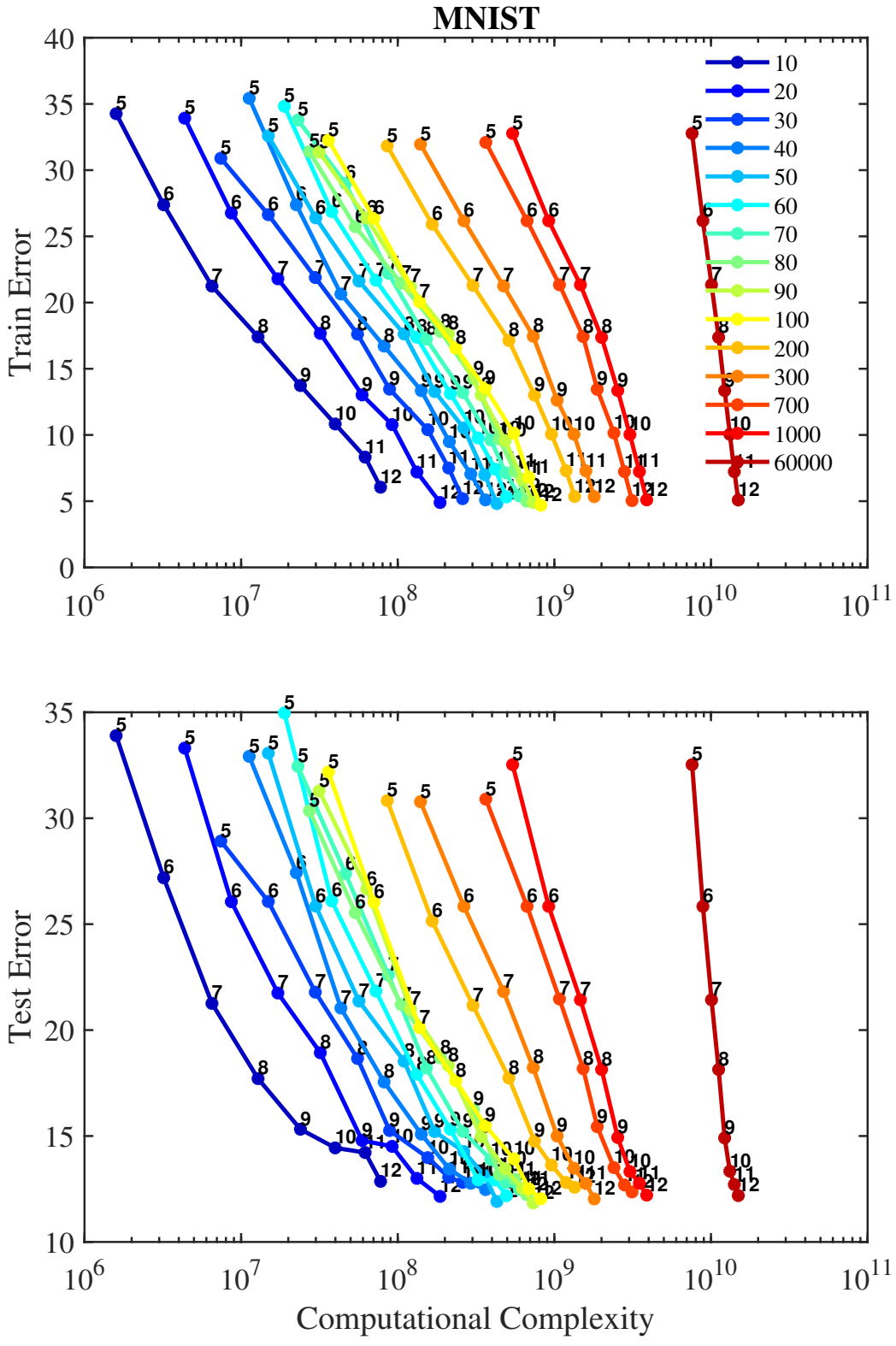


Figure 2.12: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on MNIST

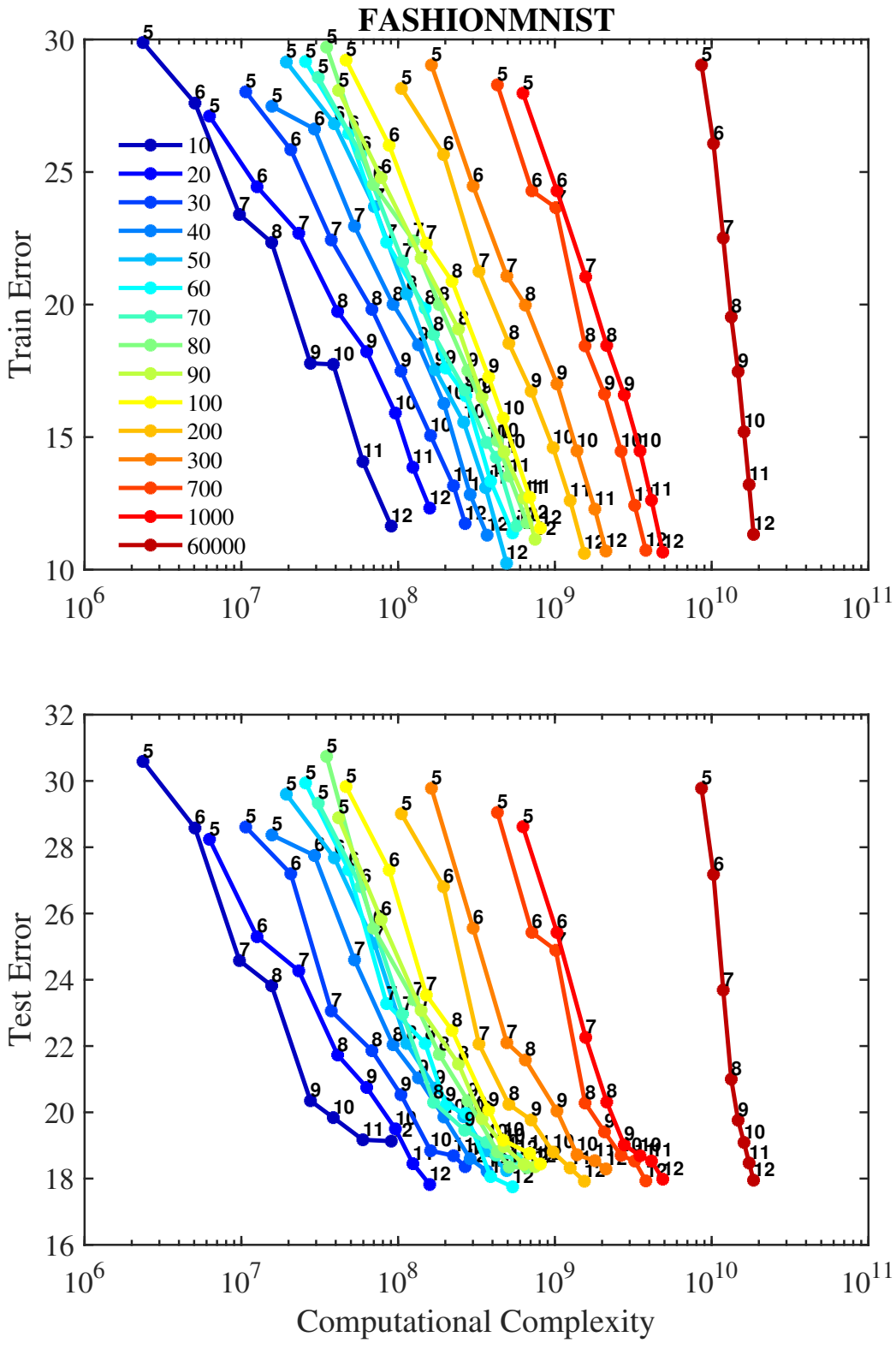


Figure 2.13: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on FASHIONMNIST

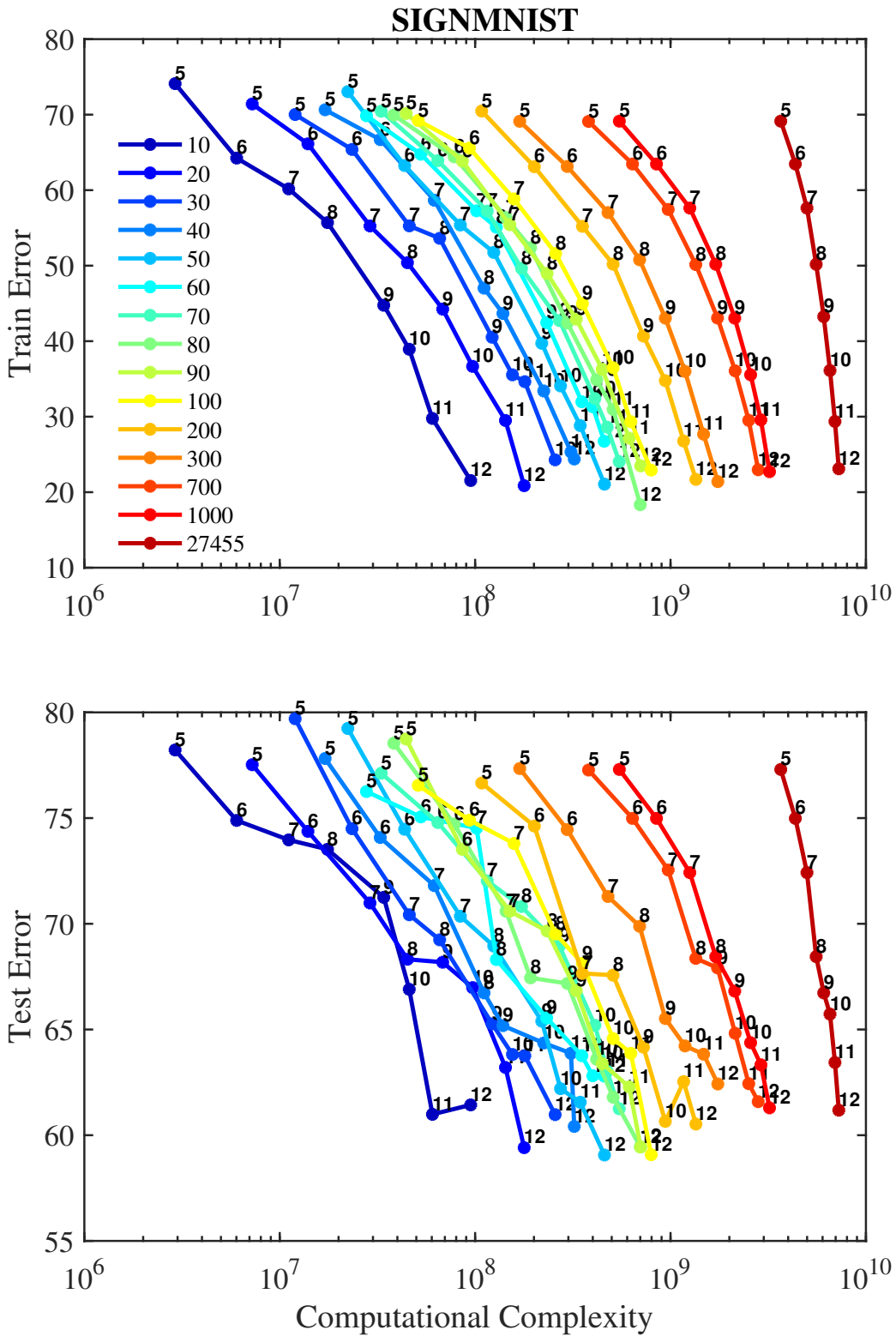


Figure 2.14: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on SIGNMNIST

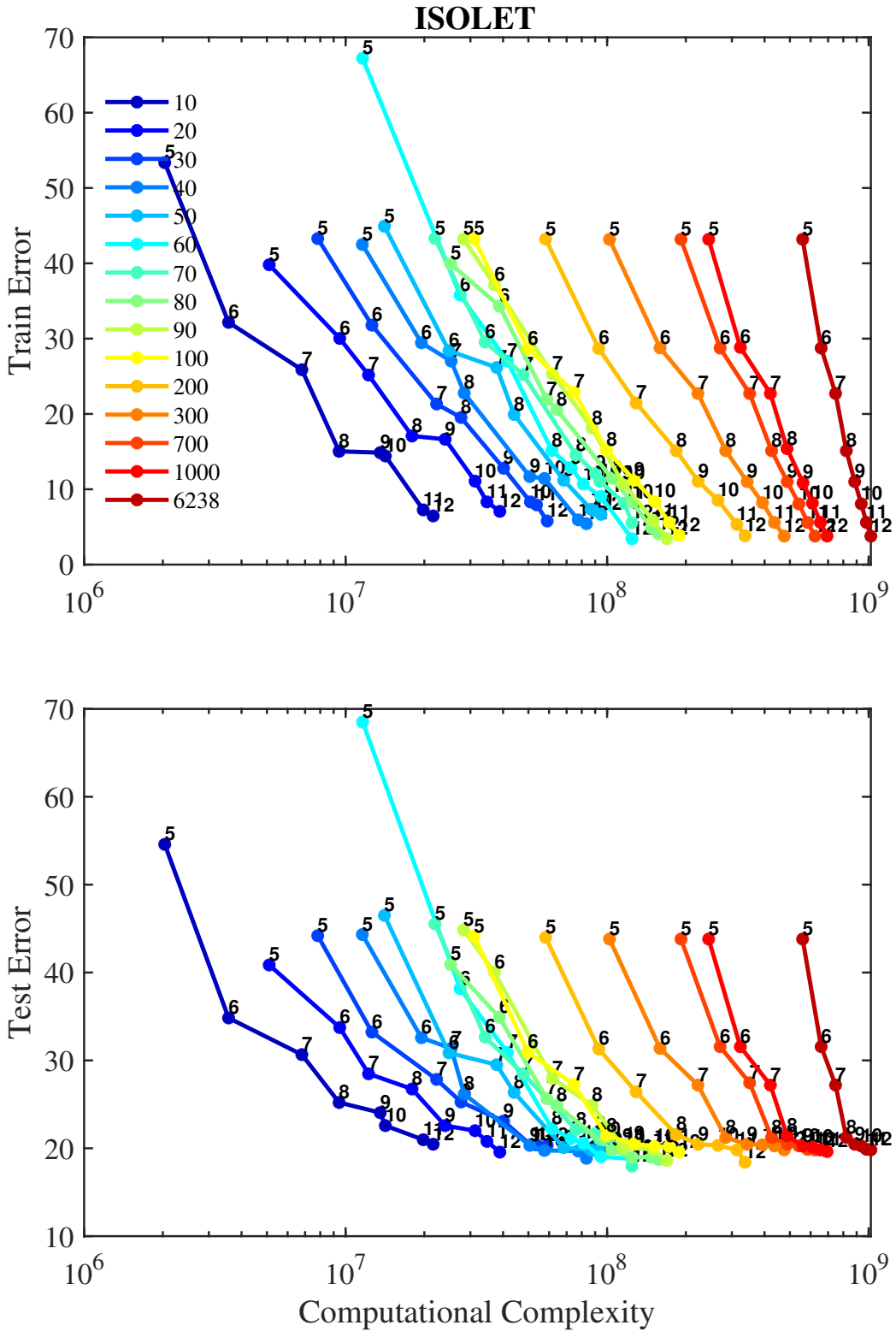


Figure 2.15: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on ISOLET

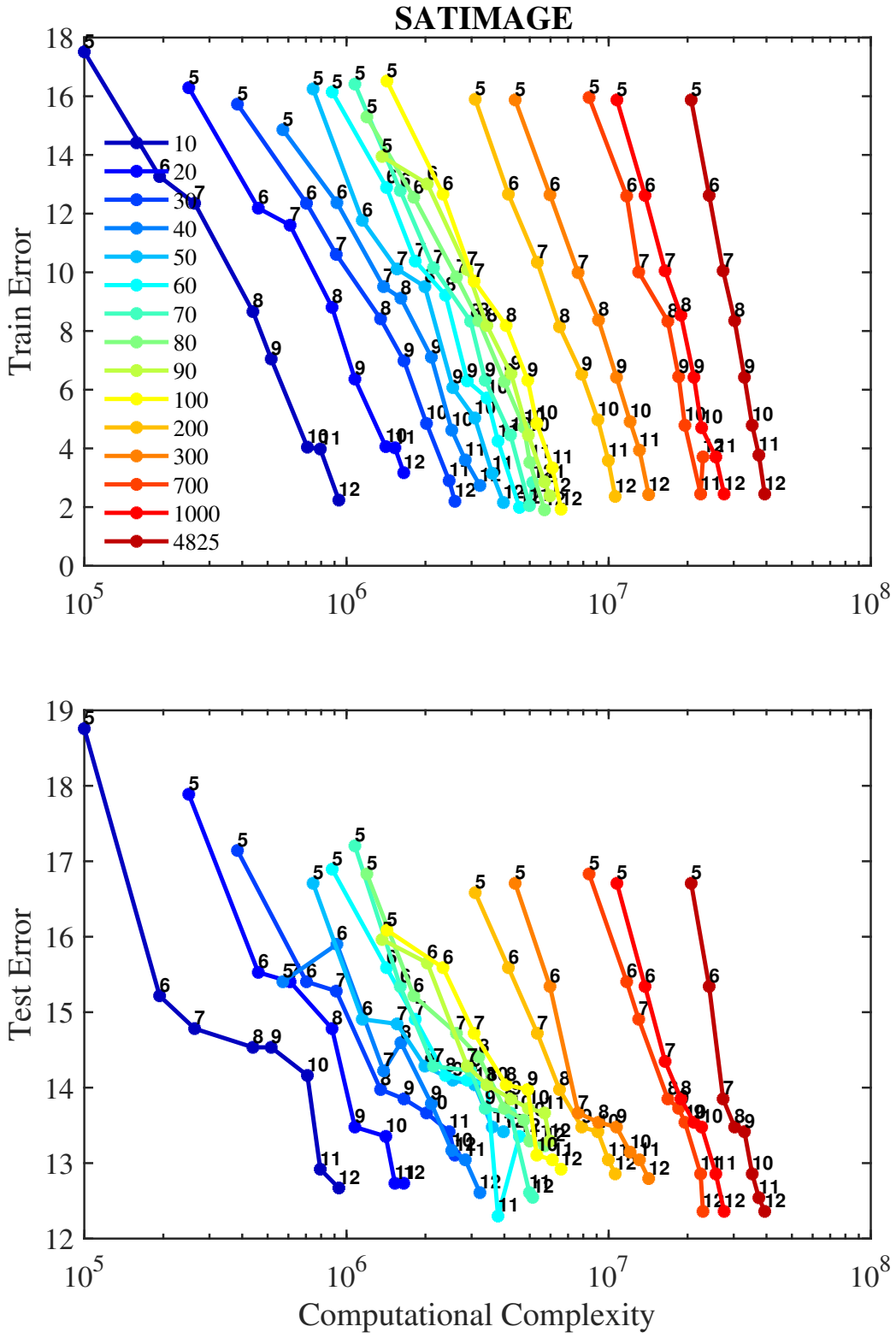


Figure 2.16: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on SATIMAGE

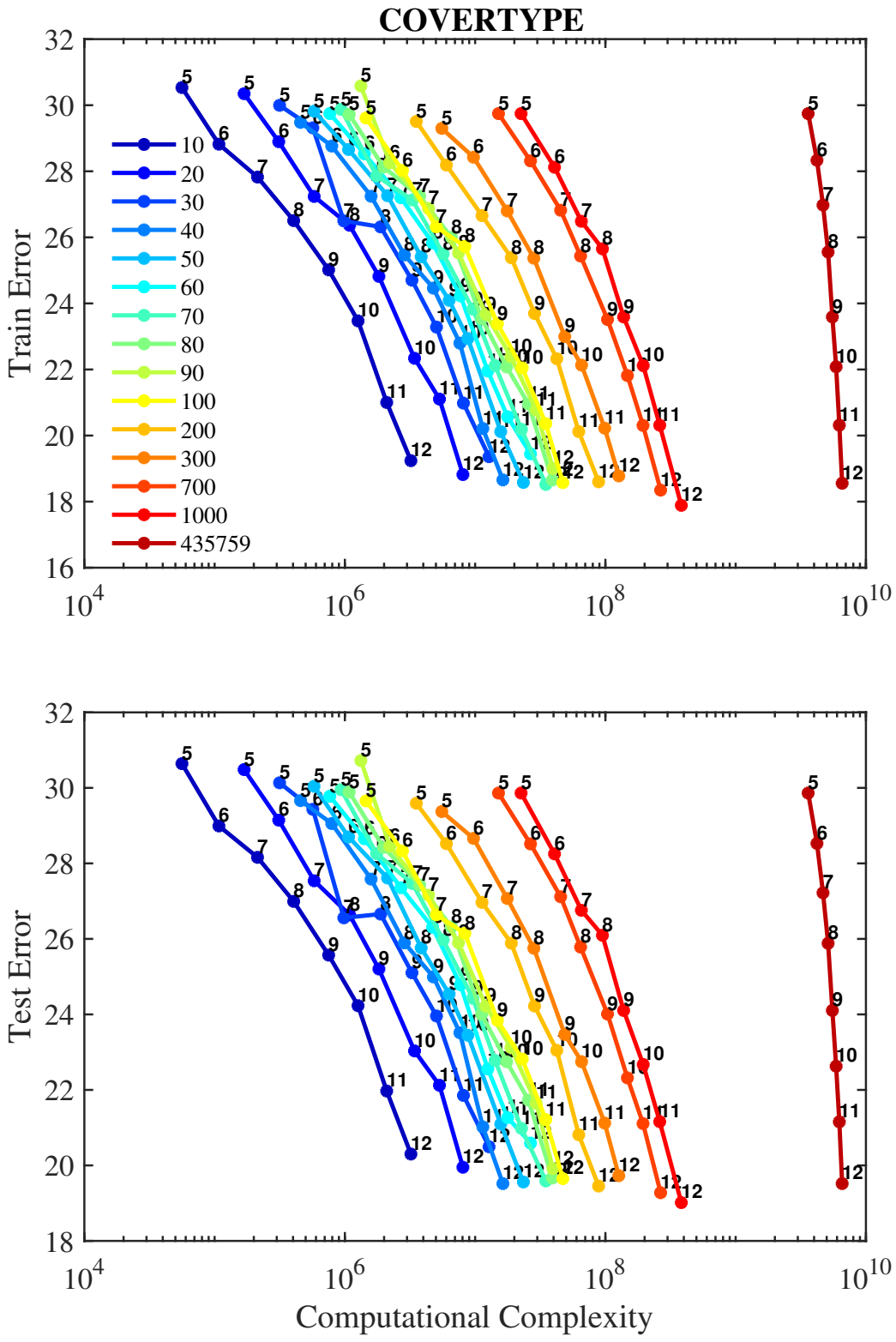


Figure 2.17: Test Error versus Computational Complexity for Different Values of Hyperparameter N_0 on COVERTYPE

2.6.3 Haar Tree Results

For each model, the relationship between inference complexity and error ratio is illustrated in Figure 2.18 for the MNIST dataset and Figure 2.19 for the FashionMNIST dataset. In these figures, Haar trees are compared against oblique trees Heath et al. (1993), axis-aligned trees Breiman et al. (1984), RBF-SVM, KNN, and k-means. The RBF-SVM model was constructed by training an SVM on Radial Basis Functions (RBFs), where the centers of the RBFs were determined using the k-means clustering algorithm. The width of the RBFs was optimized using a validation set, with the training data split into 80% training and 20% validation. The k-means model was further refined by training a k-means algorithm and then using the resulting centroids as a nearest neighbor model, similar to the approach in Tavallali et al. (2020a). The label of each centroid was determined by the majority label of the samples assigned to that centroid.

Our results demonstrate that Haar Trees achieve superior accuracy compared to other tree-based models and offer comparable or improved accuracy relative to other nearest neighbor-based models. Notably, Haar Trees represent the first instance of axis-aligned trees achieving an accuracy greater than 90% on the MNIST dataset, reaching up to 93%. This performance is significant as traditional axis-aligned trees have historically not exceeded the 90% accuracy threshold on MNIST. Furthermore, the induction of a Haar Tree is asymptotically efficient, with a complexity of $\mathcal{O}(D+\Delta)$, due to the use of integral images to compute features at each node efficiently Viola and Jones (2001). In contrast, the inference complexities for oblique trees (OC1), k-means, and RBF-SVM are $\mathcal{O}(D\Delta)$, $\mathcal{O}(DK)$, and $\mathcal{O}(D(K+1))$ respectively. These findings highlight that Haar Trees not only deliver faster inference times compared to other models but also achieve lower test errors, underscoring their effectiveness in high-performance classification tasks.

Furthermore, the Stochastic Decision Tree (SDT) implementation of the Haar Tree leverages stochasticity during the tree induction process, which allows for better handling of complex, high-dimensional datasets by introducing controlled randomness at each split. This stochastic approach enables the model to explore a more comprehensive solution space compared to traditional deterministic methods, reducing overfitting and improving generalization. In Figures 2.18 and 2.19, SDT's superior balance between inference complexity and error ratio is clearly demonstrated, as it consistently outperforms other models, including oblique trees and RBF-SVM, in both

accuracy and efficiency. The stochastic nature of SDT, combined with the computational advantages of Haar features, ensures both robust performance and efficiency, particularly in scenarios where model accuracy and inference speed are critical.

These results further highlight the adaptability of SDT across different datasets, as its performance remains consistent when transitioning from MNIST to Fashion-MNIST, underscoring the model's capacity to generalize well across distinct image classification tasks. The combination of stochastic induction and efficient feature extraction makes SDT particularly well-suited for large-scale applications, where balancing computational complexity and accuracy is paramount. This adaptability across varying data characteristics establishes SDT as a powerful tool for both real-time inference tasks and more complex high-dimensional problems, further enhancing its appeal for a range of practical applications.

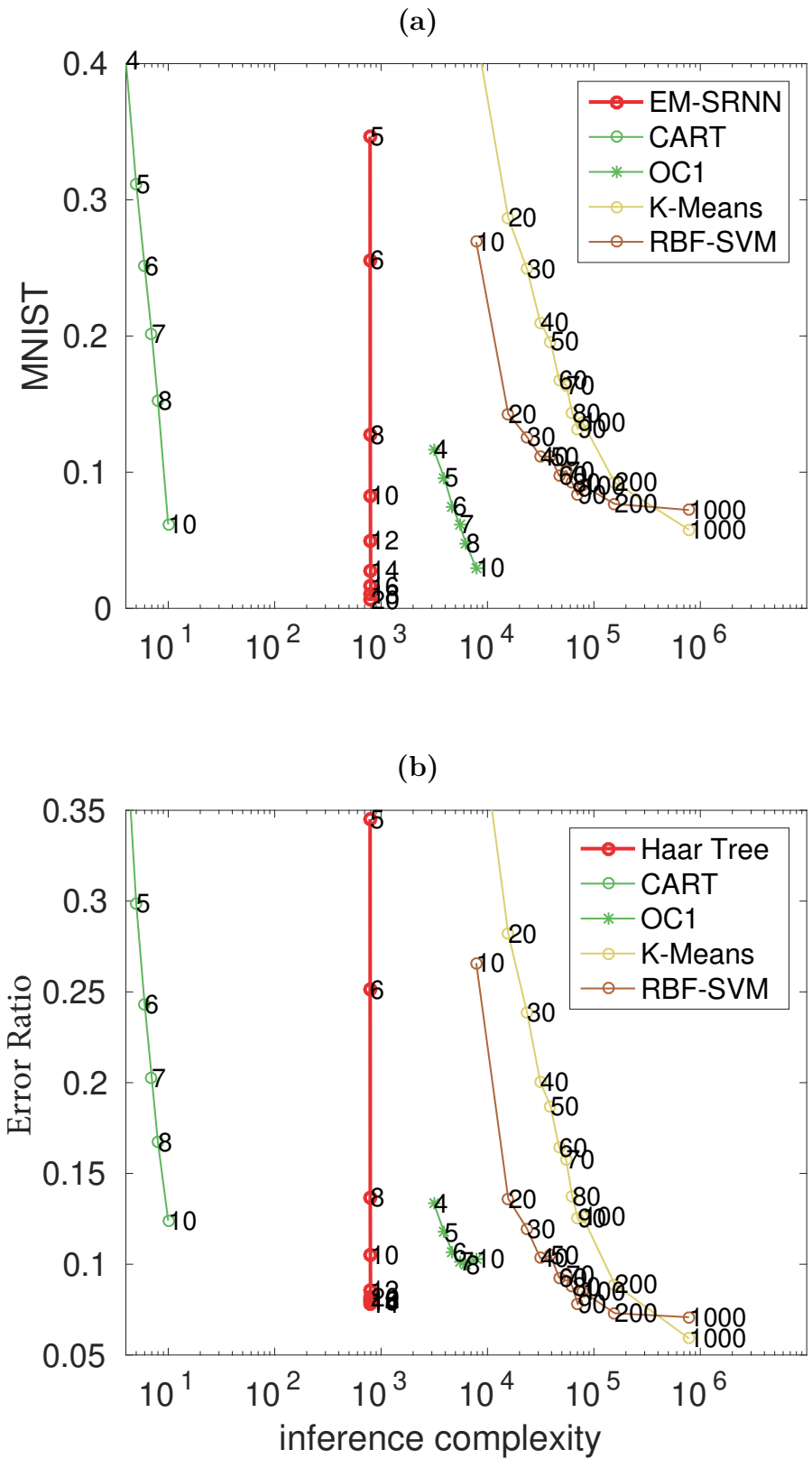


Figure 2.18: Inference complexity versus error ratio over train and test set for different models on MNIST.

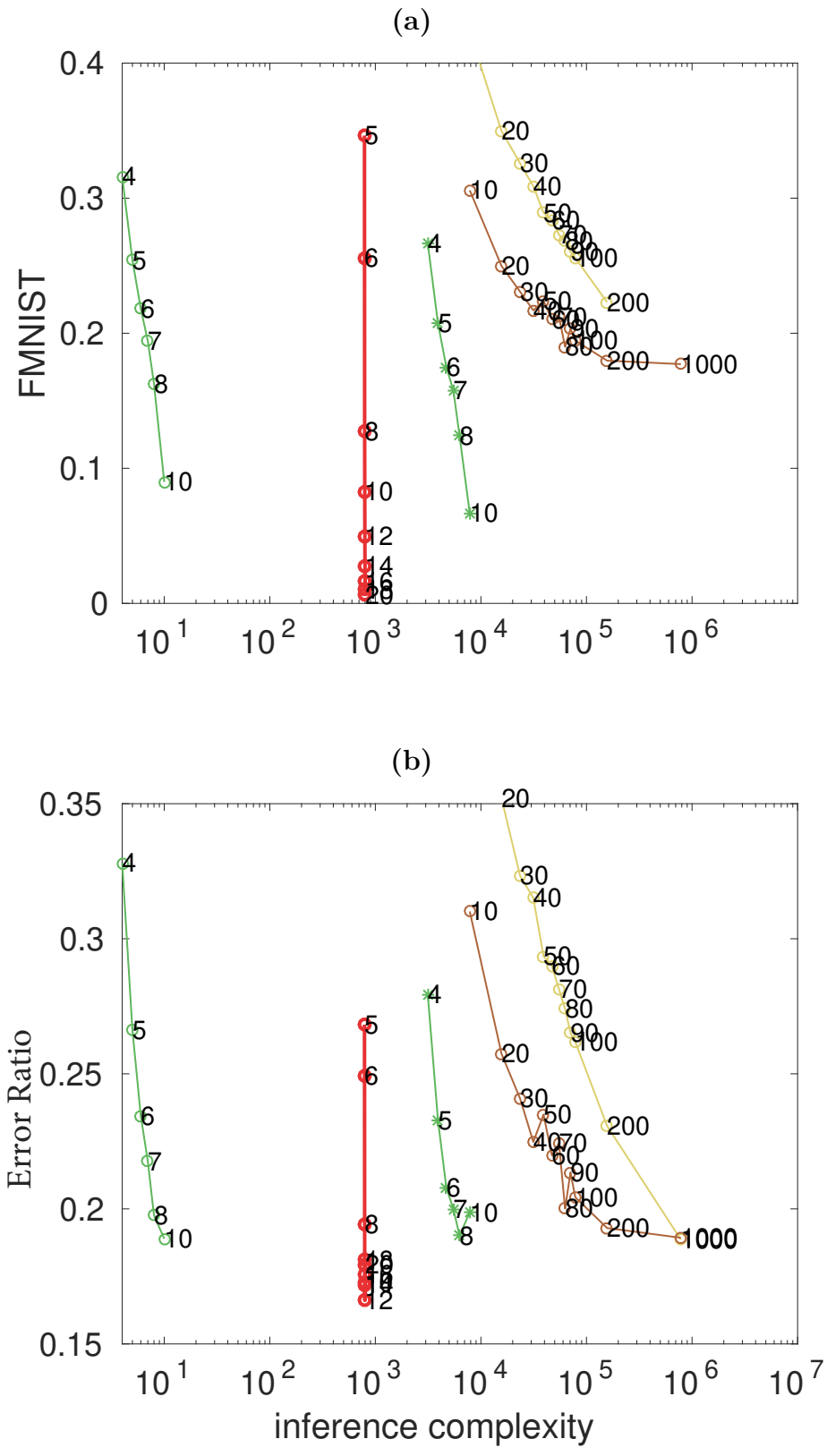


Figure 2.19: Inference complexity versus error ratio over train and test set for different models on FashionMNIST.

2.7 Chapter Summary

Decision trees are widely used and convenient methods for supervised learning problems. The exhaustive search approach used in a decision tree algorithm to determine the ideal feature and threshold that minimizes a given criterion for splitting a node is relatively expensive, especially when the number of samples and features is enormous. This issue has not been adequately addressed in the past. We addressed the issue in this study by offering a novel stochastic technique for inducing a tree. The SDT is the first stochastic algorithm developed for the greedy approach to tree induction. Asymptotically, the SDT reduces the complexity of decision tree induction by several orders of magnitude. The experimental results illustrated that the algorithm minimizes the criterion problem's upper bound.

Despite being non-deterministic, experimental results suggest that, compared to various other similar state-of-the-art decision tree approaches, SDT can perform better than online and fast tree induction algorithms. Furthermore, SDT has the potential to achieve the same accuracy and computational cost as baseline algorithms. The results of applying SDT to datasets of different sizes show that SDT can train a tree several orders of magnitude faster than other approaches. SDT is used to extend the Haar tree across the MNIST dataset, which contains over 200,000 features and 60,000 samples, for empirical evaluation. The 94% accuracy achieved by the Haar tree is not only 4% more than other complex machine learning algorithms such as oblique trees and nearest neighbor-based models but also dramatically reduces inference and training duration.

2.8 Limitation and Future Work

This study presents the first stochastic technique for decision tree induction. As a result, the model and proposed method suffer from the same restrictions as a decision tree. However, in the context of decision tree and ensemble models, it has the potential to improve the state-of-the-art because the approach enables faster and more computationally efficient decision tree and forest model generation. The method essentially paves the way for the implementation of tree-based convolution filters.

Chapter 3

A Novel Approach For Synthetic Reduced Nearest-Neighbor Leveraging Neural Networks

Synthetic Reduced Nearest Neighbor is a nearest neighbor model that is restricted to synthetic samples (i.e., prototypes). The main concept of such models consists of approaches for improving the interpretability and optimization of Synthetic Reduced Nearest Neighbor models using expectation maximization. Inspired by the potential of this paradigm, using a neural network, we offer a novel Expectation Maximization strategy for Synthetic Reduced Nearest Neighbors. The performance of our proposed method is compared to a random forest and ensemble models as classical state-of-the-art machine learning methods. The empirical results show the benefits of utilizing neural networks instead of an expectation maximization technique.

3.1 Introduction

Clustering a dataset is a key task in both supervised and unsupervised learning. In the context of classification, identifying the sub-clusters of each class is especially interesting since it enables the finding of sample variations within each class. Prototype nearest neighbor, also known as Synthetic Reduced Nearest Neighbor (SRNN), is a simplified variant of such a model in which each prototype represents a sub-cluster. Prototypes are samples generated by the learning algorithm, allowing the interpretation process to be completed by locating the prototype that is closest to the input. In the case of handwritten digits, for example, each sub-cluster is represented by a prototype, which is a synthetically produced sample in the space of the cluster's inputs.

Prototype nearest neighbor models have been examined in a variety of scenarios in recent years, ranging from adversarial robustness Saralajew et al. (2020)to few-shot learning Allen et al. (2019). However, the current state of the art leaves considerable gaps in alternate model structures. The current literature, in particular, fails to effectively address the issues inherent in optimizing models with multiple representations of clusters and measures of similarity.

Nearest neighbor algorithms are one of the most primitive types of machine learning Lloyd (1982b), Cover and Hart (1967b), Gates (1972). Their popularity is widely linked to their simplicity of use and competitive performance in a variety of machine-learning applications. These algorithms are frequently employed in collaboration with a distance metric, such as the Manhattan distance or the Euclidean distance, to determine the degree of similarity between dissimilar observations. Assume $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \Re^d$ are the observations and N is the number of samples, and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$, where $y_i \in \Re^l$ are the corresponding labels and m is the number of classes, a set of centroids $\mathbf{C} = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_k\}$, where $\mathbf{c}_i \in \Re^d$ and k is the number of centroids (nearest neighbors), are generated. Algorithm 1 represents a more extensive explanation of the nearest neighbor method using weighted average distance metric.

Despite their simplicity and efficiency, nearest neighbor algorithms frequently suffer from the "curse of dimensionality" - that is by increasing the number of observations in higher dimensional problems, the perceptive concept of proximity or similarity may no longer be descriptively meaningful Sammut and Webb (2017). To reduce the complexity of inference, various data structures such as R-Tree, KD-tree, or similar

Algorithm 1 Nearest neighbor algorithm

Compute D = $\|\mathbf{x}_j - \mathbf{c}_i\|$ for $j \in N, i \in k$
Given some norm $\|.\|$ on $\Re^d$ and a point $\mathbf{c_i}$, reorder to yield
$\|\mathbf{x}_0 - \mathbf{c}_i\| \leq \|\mathbf{x}_1 - \mathbf{c}_i\| \leq \ldots \leq \|\mathbf{x}_N - \mathbf{c}_i\|$
Find the heuristically optimal number of centroids (nearest neighbors) "k"
Calculate an inverse distance weighted average with the k-nearest multivariate neighbors

spatial access trees are utilized De Berg (2000), Beygelzimer et al. (2006), Mathy et al. (2015). Recent research has advocated using Expectation Maximization (EM) techniques like the k-means clustering algorithms to improve the prediction capabilities of nearest neighbor models Tavallali et al. (2020b). These models are easier to understand and have linear computation time complexities. The expectation-Maximization (EM) algorithm is a well-known approach that estimates the maximum likelihood in the presence of missing data. Using an iterative approach, the EM algorithm alternates between two modes. Em algorithm comprises two interconnected steps: **E-step**: An estimation step that estimates the dataset's latent or missing variables. **M-step**: A step in which the model's parameters are optimized in the presence of data Singh (2005), Gupta et al. (2011). Em algorithm techniques are well-known and have been frequently used to train machine learning models Tavallali et al. (2021), Carreira-Perpinán and Tavallali (2018), Dempster et al. (1977), Jordan and Jacobs (1994), Lloyd (1982a).

The strengths of neural networks in terms of classification performance and dimensionality reduction are well-known Niranjani and Selvam (2020), Hinton and Salakhutdinov (2006). Their outstanding performance in a range of classification and regression tasks is due to their cascaded structure, as well as their non-linear and non-convex architecture, as cited in Krizhevsky et al. (2012), He et al. (2016). Typically, neural networks are trained by optimizing an objective function of the form

$$\min_{\Theta \in \Re^n} f(x; \Theta), \tag{3.1}$$

where $\Theta \in \Re^n$ are the network parameters and f represents the empirical loss of estimation. The loss is "backpropagated" through the network parameters (see Rumelhart et al. (1986)) to minimize this empirical loss.

To increase this K-nearest neighbor's inference capabilities by taking advantage of neural networks, we offer a unique expectation maximization strategy that we named

Neural-SRNN.

3.2 The Proposed Methods

Assume we have a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_n\}$, where $\mathbf{x}_i \in \Re^d$ is an input smaple from the dataset. The dataset conataines m classes with labels $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_m\}$.The Neural-SRNN model consists of learning k neural networks for each class. The model makes predictions by selecting a class according to the neural net's output; that is, the model selects the class that corresponds to the neural network that has the highest probability. In other words:

$$\hat{\mathbf{y}} = \text{Class}\left(\arg\max_{\substack{i\in\mathbf{M}\\ j\in\mathbf{J}}} \hat{\mathbf{z}}_{ij}(\mathbf{x})\right), \tag{3.2}$$

where $\mathbf{M} = \{1 \dots m\}$, $\mathbf{J} = \{1 \dots k\}$ and $\hat{\mathbf{z}}_{ij}$ are the output from the i^{th} neural network in class i for input $\mathbf{x}$. Each input $\mathbf{x}_i$ is fed to each neural network j. The "Class" function in eq. (3.2) returns the class i.

Training the neural-SRNN can be modeled as an expectation-maximization problem with a binary loss as the objective function, stated as follows:

$$\mathcal{I}(\hat{\mathbf{y}}, \mathbf{y}) = \begin{cases} 0 & \hat{\mathbf{y}} = \mathbf{y} \\ 1 & \hat{\mathbf{y}} \neq \mathbf{y} \end{cases}$$

This becomes an NP-hard problem and infeasible to solve. Due to its NP-hard nature, a technique would be to replace it with a substitute objective function to make the problem feasible. To train a linear classifier, logistic regression solvers most frequently employ this method.

We explore the challenge of learning a neural SRNN with the appropriate maximum likelihood function and provide a technique based on the K-means EM algorithm. As a result, the challenge of learning a neural SRNN is defined as follows:

$$\begin{aligned} \max_{\mathbf{z}_{c_{ji}}, \Theta_{c,j}} & \sum_{i=1}^{N}\sum_{c=1}^{C}\sum_{j=1}^{K} p_{c_i}\mathbf{z}_{c_{ji}} \log(\hat{\mathbf{z}}_{c_j}(\mathbf{x}_i)) + \\ & (1 - p_{c_i})(1 - \mathbf{z}_{cji}) \log(1 - \hat{\mathbf{z}}_{c_j}(\mathbf{x}_i)) \\ \text{subject to} & \sum_{c,j} \mathbf{z}_{c_{ji}} = 1, \end{aligned} \tag{3.3}$$

where p_{ci} presents the c^{th} element of one-hot encoded vector $\mathbf{p}_i$ which is the one-hot vector of $\mathbf{y}_i$ where all elements of $\mathbf{p}_i$ are 0 except the y_i^{th} element, which is 1. The variable $z_{c_{ji}}$ represents the assignment of a sample to its corresponding neural network. Each sample can be assigned to only one neural network. Thus, $\mathbf{z}_{cji} \in \{0, 1\}$ represents the $\mathbf{z}_{c_{ji}}$ where the $\mathbf{z}_{ji}$ is a one-hot vector (for class c), Θ_j represents the parameters of c^{th} neural network. It is important to note that the objective function in eq. (3.3) connotes that a sample cannot be given to a neural network of a different class, as this would cause the objective to be reduced. We propose a two-step algorithm for tackling the optimization problem in eq. (3.3); (1) in the **assignment step**, the assignments are optimized, and (2) in the **update step**, the neural networks are optimized. A detailed definition of the two steps is as follows:

Assignment step involves assigning training samples to their nearest centroid, and each centroid is labeled ideally. In other words, the assignment step is optimizing the problem over z_{cji} for all $c \in \mathbf{C} = \{1 \dots C\}$ and $j \in \{1 \dots K\}$, while fixing the neural network parameters (Θ_{cj}). For class c, the problem at this step can be written as follows:

$$\begin{aligned} \max_{z_{c_{ji}}, \Theta_{c_j}} & \sum_{j=1} z_{c_{ji}} \log(\hat{\mathbf{z}}_{c_j}(\mathbf{x}_i)) \\ & \text{subject to} \sum_j z_{c_{ji}} = 1 \end{aligned} \tag{3.4}$$

Since our interest in solving the problem only for c^{th} class, hence, for simplicity purposes, we removed the terms that reflect the class of samples from eq. (3.4). The key term that determines the value of $z_{c_{ji}}$ is $z_{c_{ji}} \log(\hat{\mathbf{z}}_{c_j}(\mathbf{x}_i))$, where z_{cji} can only be 1 for the j^* neural network to maximize eq. (3.4). Here j^* is given by:

$$j^* = \arg \max_j \hat{\mathbf{z}}_{c_j}(\mathbf{x}_i) \tag{3.5}$$

Update step corrects the assignments and updates the parameters of the neural networks. This step is actually a binary classification problem in which our goal is to maximize $\hat{\mathbf{z}}_{c_j}(\mathbf{x}_i)$ for samples that are assigned to the cj^{th} neural network and concurrently minimizing the same term for all other samples that are not assigned to c_j^{th} neural network. The problem for c_j neural network is as follows:

$$\max_{\Theta_{cj}} \sum_{i=1}^{N} p_{c_i} z_{c_{ji}} \log(\hat{\mathbf{z}}(\mathbf{x}_i)) + (1 - p_{c_i})(1 - z_{c_{ji}}) \log(1 - \hat{\mathbf{z}}_{c_j}(\mathbf{x}_i)) \tag{3.6}$$

Observations of the same class are sampled uniformly and assigned to each neural network of the same class and neural networks in the **Initialization** phase. After that, the networks are trained on these uniformly sampled images.

3.2.1 Convergence to a local maximum

Every step of the method improves the function in eq. (3.3). As a result of iterating across both phases, the goal function grows. Thus, eq. (3.3) grows until no fresh sample assignment results in an increase, at which time only the second step influences the objective function.

Proof. The claim that every step of the proposed method improves the function in equation (4.2) hinges on the fundamental principles of the expectation-maximization (EM) algorithm, which alternates between maximizing the expected value of the log-likelihood function (the "E-step") and maximizing the likelihood itself with respect to the parameters (the "M-step"). In the context of the Neural-SRNN model, equation (4.2) represents an objective function that combines the assignment of samples to their respective neural networks (via $z_{c_{ji}}$) and the optimization of the neural network parameters (Θ_{c_j}).

During each iteration, the assignment step maximizes the contribution of each sample to the log-likelihood function by selecting the neural network j^* that maximizes the output $z_{c_j}(x_i)$, as given by equation (4.4). This step guarantees that the objective function in equation (4.2) does not decrease because it optimally reassigns samples to the neural networks that best represent them. Subsequently, the update step refines the neural network parameters to further maximize the likelihood of the assigned samples, ensuring a monotonic increase or plateau in the objective function. Given that the EM algorithm is designed to converge to a local maximum, the iterative process of alternating between these two steps ensures that equation (4.2) continues to increase or remains stationary until convergence is achieved. At convergence, no further sample reassignment can lead to an increase in the objective function, implying that only the update step has a potential impact on the final value of the objective function. Therefore, the process inherently leads to the maximization of equation (4.2) until a local maximum is reached, thereby proving the claim.

□

3.2.2 Pruning

In practice, we've discovered that the algorithm prefers to prune neural networks by not allocating any samples to these neural networks while concurrently increasing the objective function. This is especially true when the number of neural networks and their complexity are considerably high. This is potentially a sort of regularization in which the algorithm is regularized to learn manifolds of data that fit the design of the prototype neural network.

3.3 Experimental Results

To assess the proposed method's performance, we compare its performance in the classification problem to popular approaches such as ensemble models Tavallali et al. (2019) and Random forest Breiman (2001). The classification challenge was chosen because of its importance as well as the availability of reproducible model architectures.

3.3.1 Datasets

To demonstrate the efficacy of the proposed strategy in dealing with varied datasets, three different datasets were chosen: **MNIST** Deng (2012), **Fashion-MNIST** Xiao et al. (2017) and **sign-MNIST** Huang et al. (2015).

- **MNIST**: is a large database of handwritten digits 0 to 9 and is widely used for classification algorithms. The number of features is 784. The train and test set contains 60000 and 10000 samples, respectively.
- **Fashion-MNIST**: is a dataset consisting of grayscale images from 10 different classes such as T-shirts/tops, Pullovers, Coats, and so on. The number of features is 784. The train and test set contains 60000 and 10000 samples, respectively.
- **Sign-MNIST**: the American Sign Language letter database consists of 24 classes for each letter of the alphabet, excluding J and Z. The number of features is 784. The train and test set contains 27455 and 7172 samples, respectively.

3.3.2 Experimental Setup

We used seven alternative architectures to investigate the effect of different neural network structures on the results:

Shallow networks: Three distinct single convolutional layers shallow nets are used, each with a filter size of 1, 2, and 3 with a kernel size of 3. All three structures have one Dense layer.

3 layer networks: Two distinct networks, each with three convolutional layers, filter sizes of 2,3 and 4, and a kernel of 5,3,2. This structure is analyzed both with and without dropout. Both of these structures have one Dense layer.

LeNet 5: This neural network architecture LeCun et al. (1998) is made up of two convolutional layers and two Dense layers.

Modified-LeNet 5: The convolutional layer of the original LeNet 5 structure is replaced with Separable Conv2D. The filter and kernel sizes remain unchanged.

We also examine configurations with various numbers of neural networks assigned to each class ($n \in \{2, 4, 8, 10, 20\}$). We also trained the neural networks over the sparse-categorical focal loss objective function to address the unbalanced character of these datasets. Each model was trained and tested over the course of 20 iterations. The Adam optimizer Kingma and Ba (2014) was used for training, with a learning rate of 0.01 and a decay rate of $\alpha = 0.98$.

Also, the results of the Neural SRNN model with different numbers of centroids were compared to other state-of-the-art algorithms, which are as follows:

Random Forest Model: Random forest is a popular model that consists of numerous separate decision trees that work together to form an ensemble Parmar et al. (2018). In this paper, the scikit-learn's Pedregosa et al. (2011) built-in "RandomForest" function is used.

Ensemble Model: Ensemble Neural Networks apply several neural networks simultaneously to train over a dataset. We chose two distinct architectures for the ensemble model to compare it to our proposed model: Lenet-5 and a shallow network. The number of networks is proportional to the number of classes in the dataset. The ensemble model's neural networks are trained on 70% of randomly selected samples from the original training set. The test set is then distributed to each member, and the ensemble error rate is calculated as the average error rate of all component members.

3.3.3 Results

The results of the experiments are presented in this section to highlight the advantages of the proposed strategy. The Neural-SRNN with LeNet5 structure is compared to a number of other cutting-edge methods. The train and test accuracy resulting from various Neural-SRNN model settings and baseline models for three data sets is illustrated in Figures 3.1, 3.2, and 3.3.

For the MNIST dataset, the figures show that the Neural-SRNN models generally achieve higher accuracy compared to baseline methods such as Random Forest and simpler ensemble models. The accuracy tends to increase with the complexity of the Neural-SRNN configurations, particularly with models like NSRNN-10 and NSRNN-20, which maintain high accuracy during both training and testing phases. The small gap between train and test accuracy for these models suggests they are well-regularized and effectively generalized to new data. In contrast, the simpler models exhibit larger discrepancies between training and testing accuracy, indicating potential overfitting or an inability to fully capture the underlying patterns in the MNIST data.

For the Fashion-MNIST dataset, which is more complex than MNIST, a similar pattern emerges. Neural-SRNN models continue to outperform the baseline methods, with NSRNN-10 and NSRNN-20 configurations achieving the highest accuracy. The overall accuracy is slightly lower than that observed for MNIST, reflecting the increased difficulty of the Fashion-MNIST dataset. The ensemble methods and Random Forest show significantly lower accuracy, particularly during testing, which highlights their limitations in handling the more intricate patterns present in this dataset. The Neural-SRNN models, particularly with a higher number of recurrent units, demonstrate their ability to capture and generalize from complex data, resulting in consistently high accuracy across both training and testing phases.

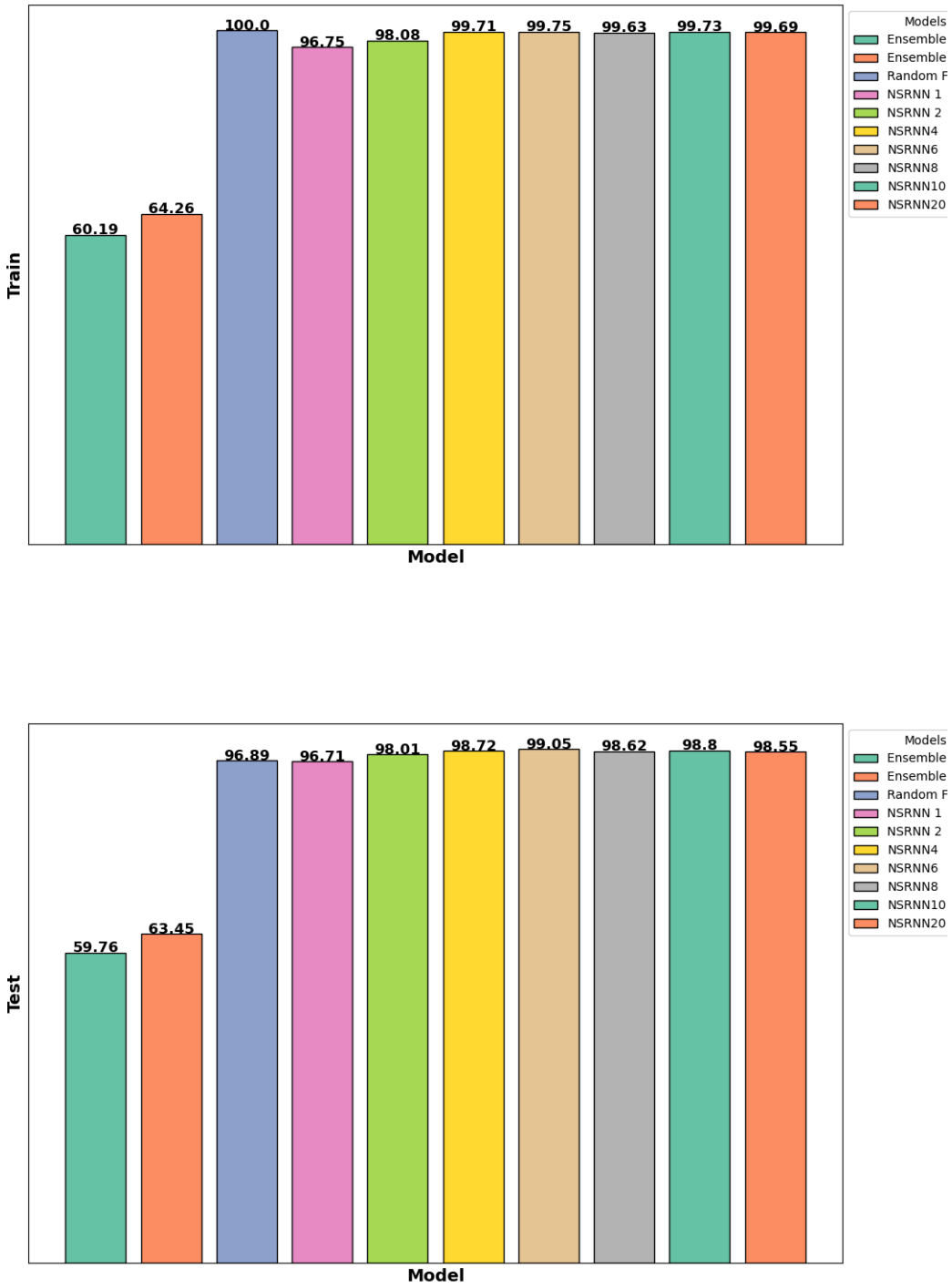


Figure 3.1: Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for MNIST.

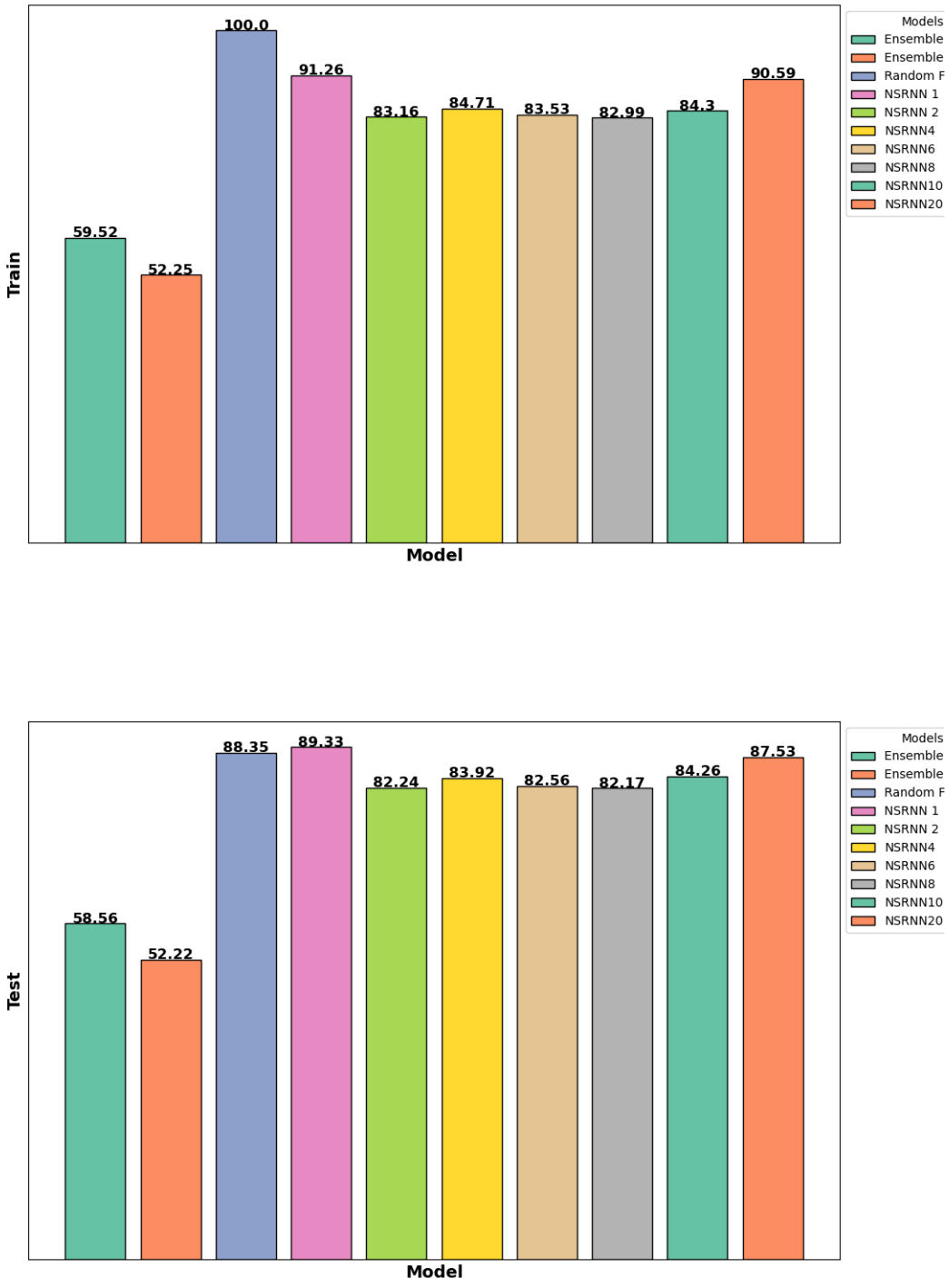


Figure 3.2: Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for FASHION MNIST.

The Sign-MNIST dataset, known for its challenging nature due to the complexity of sign language recognition, further underscores the strengths of the Neural-SRNN architecture. The figures reveal that Neural-SRNN models, especially those with 10 or more recurrent units, achieve high accuracy, far surpassing the baseline models. The testing accuracy for these models remains close to the training accuracy, indicating that the models are not only capable of learning complex patterns but also of applying this knowledge to new data effectively. On the other hand, the simpler models, including Random Forest and basic ensemble methods, struggle to achieve comparable accuracy, particularly in the test phase, highlighting their difficulties in adapting to the nuances of sign language data.

Across all three datasets, the Neural-SRNN models consistently deliver higher accuracy, particularly as the model complexity increases with the number of recurrent units. This trend highlights the ability of Neural-SRNN to handle datasets with varying levels of complexity by effectively capturing and generalizing from the patterns within the data. The performance of the baseline models, while adequate for simpler tasks, clearly diminishes as the complexity of the dataset increases, demonstrating the necessity of more sophisticated architectures like Neural-SRNN for achieving high accuracy in challenging classification tasks.

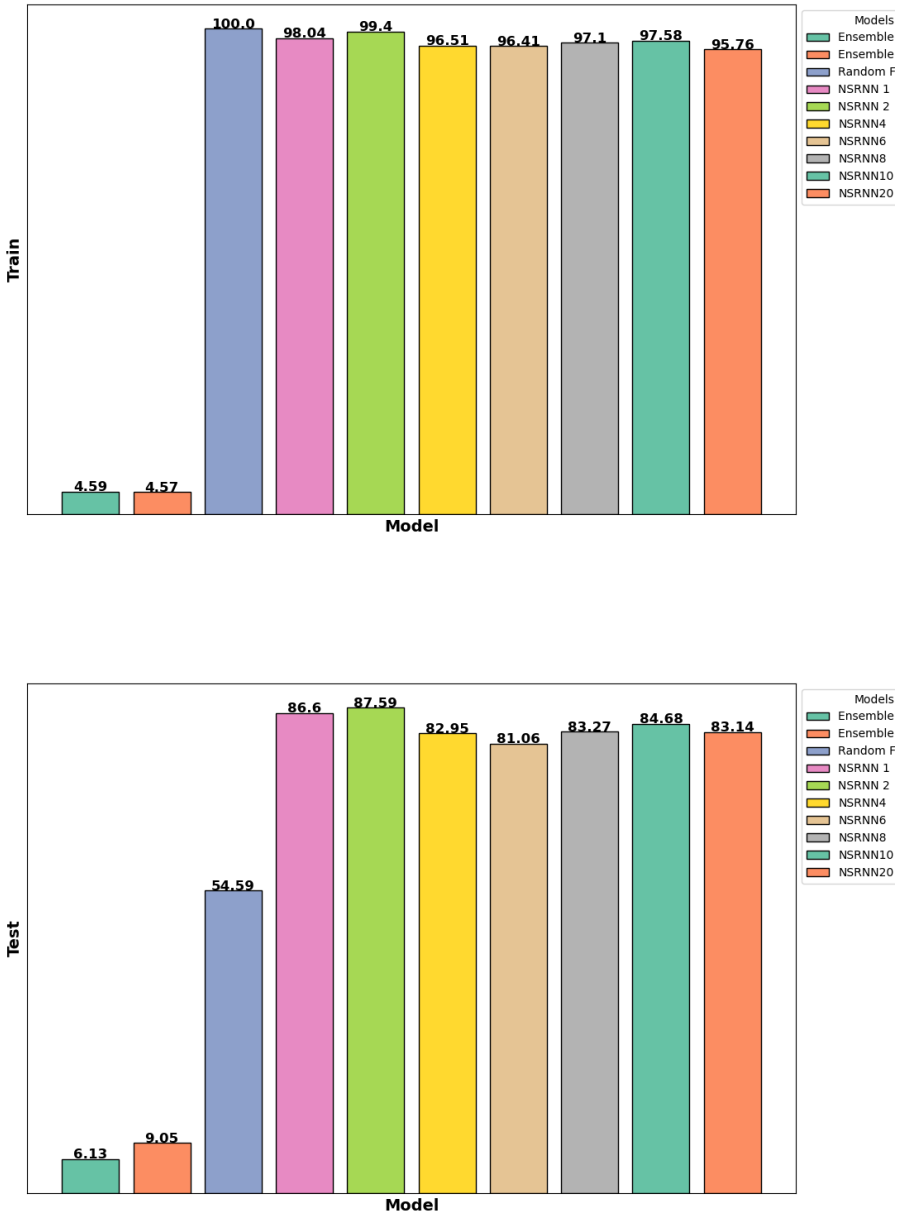


Figure 3.3: Neural-SRNN vs. other state-of-the-art algorithms on Train and Test set for SIGN MNIST.

Figure 3.4 shows the error ratio for test and train sets versus execution iteration for the MNIST, Fashion-MNIST, and Sign-MNIST datasets, providing valuable insight into the performance of Neural-SRNN with varying numbers of centroids (neural networks per class). The results demonstrate that initially, as the number of centroids (which can be thought of as models representing subgroups within each class) increases, the error rate on the MNIST dataset decreases. This means that by using more centroids, the model can more accurately capture the variations within each class, leading to better classification performance. However, this improvement has its limits. When the number of centroids is increased beyond a certain threshold—specifically when the number exceeds 20—the error rate starts to increase again. This suggests that while more centroids allow for finer granularity in representing each class beyond a certain point, adding more centroids introduces complexity that the model may not handle well, leading to overfitting or other issues that degrade performance.

The balance between model expressiveness and overfitting can explain the observed trend. Initially, adding more centroids allows the model to better capture the diversity within each class, as each centroid can represent a distinct variation or subgroup. This leads to a reduction in the error rate because the model can make more precise distinctions between different inputs.

However, as the number of centroids continues to increase, the model becomes increasingly complex. With too many centroids, the model might start to overfit the training data, meaning it becomes too specialized in distinguishing the training examples at the expense of generalization to new data. This overfitting manifests as an increase in the error rate on the test set. Additionally, managing a larger number of centroids can increase the computational burden and may introduce instability in the training process, contributing further to the rise in error rate.

Thus, while increasing the number of centroids initially benefits model performance by improving its representational capacity, there is a tipping point where the added complexity outweighs these benefits, leading to diminished returns and even negative impacts on accuracy.

The Fashion-MNIST and Sign-MNIST datasets, which are more complex than MNIST because of the greater variability and subtler distinctions between classes, show that adding more centroids generally improves performance, but the relationship is nuanced. For these datasets, fewer centroids initially result in higher error rates, indicating that the models struggle to capture the complexity of the data.

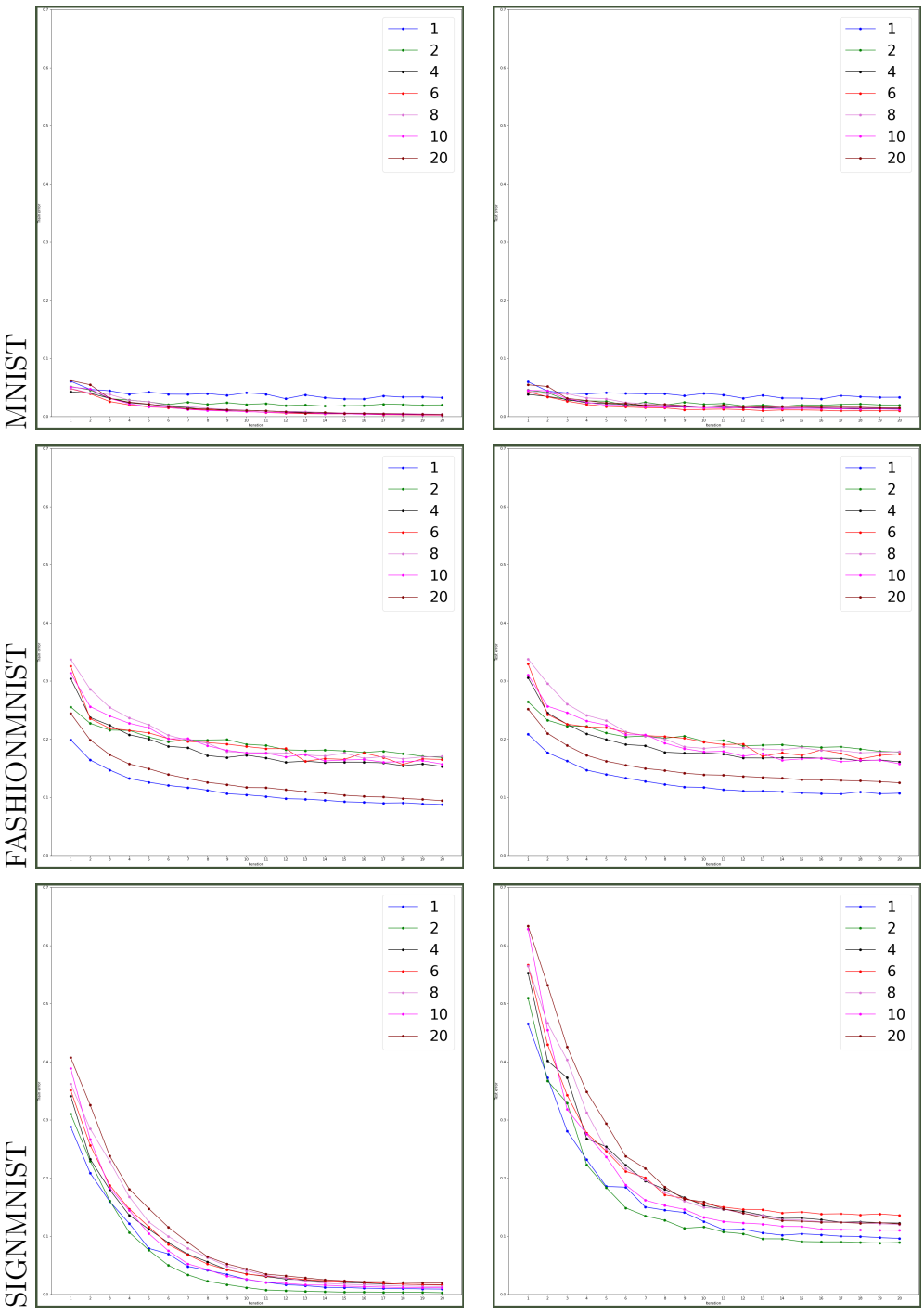


Figure 3.4: Training (a) and Testing (b) response of Neural-SRNN with different numbers of neural networks for each class.

As the number of centroids increases, the model becomes better at capturing the intricate variations within each class, leading to improved performance. However, there is still a point where further increasing the number of centroids leads to diminishing returns or even overfitting, particularly beyond a certain threshold, such as 20 centroids per class. This suggests that while a more complex model is necessary for these datasets, there's still a limit to how much additional complexity is beneficial.

These observations highlight the need for careful tuning of model complexity in Neural-SRNN. While more centroids can help capture the complexity of more challenging datasets like Fashion-MNIST and Sign-MNIST, it's essential to find a balance. Too few centroids might not fully leverage the model's potential, while too many can lead to overfitting, reducing the model's ability to generalize effectively.

This research also studied the effect of neural net structure on the final results. The results of this experiment on (a) MNIST, (b) Fashion-MNIST, and (c) Sign-MNIST are illustrated in Figure 3.5. The results demonstrate that the LeNet-5 structure, particularly when incorporating both standard convolutional layers and separable convolutional layers, consistently outperforms other tested architectures across all datasets. This superior performance is evident in the lower test error rates achieved by LeNet-5 compared to other configurations, especially as the number of iterations increases. The robustness of the LeNet-5 structure is attributed to its ability to effectively capture spatial hierarchies within the data, which is critical for tasks such as image classification.

Further analysis reveals that in shallower network structures, the size of the convolutional filters plays a significant role in prediction accuracy. Increasing the filter size within these structures leads to a reduction in the error ratio, indicating that larger filters are more effective at capturing relevant features in the data. This is particularly noticeable in the early iterations, where models with larger filters converge more rapidly and achieve lower error rates than those with smaller filters.

The impact of dropout is also evaluated, particularly in a three-layer system where the filter sizes are held constant. The findings suggest that incorporating dropout reduces the error ratio, particularly for models with fewer centroids. Dropout serves as a regularization technique, preventing the model from overfitting the training data by randomly omitting units during the training process. This effect is more pronounced in simpler models, where the risk of overfitting is higher. However, as the number of centroids increases, the performance gains from dropout diminish, and the results from both dropout and non-dropout architectures converge, indicating that

the model's complexity becomes sufficient to generalize effectively without additional regularization.

Across all datasets, the results underscore the importance of selecting an appropriate network structure for the specific characteristics of the data. Although the LeNet-5 structure offers a strong baseline, the effectiveness of convolutional filter sizes and dropout as additional tuning parameters suggests that model performance can be further optimized by carefully adjusting these factors. This highlights the need for a tailored approach in neural network design, where both the architecture and hyperparameters are fine-tuned to the demands of the dataset at hand.

The figures provided in Figure 3.6 offer a detailed visualization of how the Neural-SRNN model identifies and emphasizes critical features for specific classes within the MNIST and Fashion-MNIST datasets. Neural-SRNN operates on the principle of breaking down a dataset into multiple classes and then further into sub-clusters or centroids, with each centroid represented by a distinct neural network. This approach allows the model to capture a wide range of variations within each class more effectively, which is particularly useful in datasets with complex and subtle intra-class differences.

In the MNIST dataset, for classes like **"1"** and **"7"**, the Neural-SRNN model demonstrates its ability to focus on the most distinguishing features of each digit. The model's architecture allows it to allocate different neural networks to emphasize varying aspects of the input image. For instance, in the case of the digit **"1"**, the model highlights the vertical stroke, a defining characteristic of this digit. In contrast, for the digit **"7,"**, the model emphasizes the horizontal bar and diagonal stroke, which are crucial for distinguishing **"7"** from other digits. This indicates that Neural-SRNN effectively uses its multiple centroids to capture the critical aspects that differentiate one digit from another, leveraging the model's ability to generalize across different variations of the same digit.

Similarly, in the Fashion-MNIST dataset, which involves more complex images such as clothing items, Neural-SRNN's architecture shows its strength in identifying subtle differences between similar classes. For example, when distinguishing between a **"pullover"** and a **"shirt"**, the model uses different neural networks to focus on features like the neckline, sleeves, or the texture of the fabric. The activation patterns seen in the figures illustrate how Neural-SRNN allocates different centroids to capture these specific features, which are essential for accurate classification in this dataset. The model's ability to break down the complex task of image classification

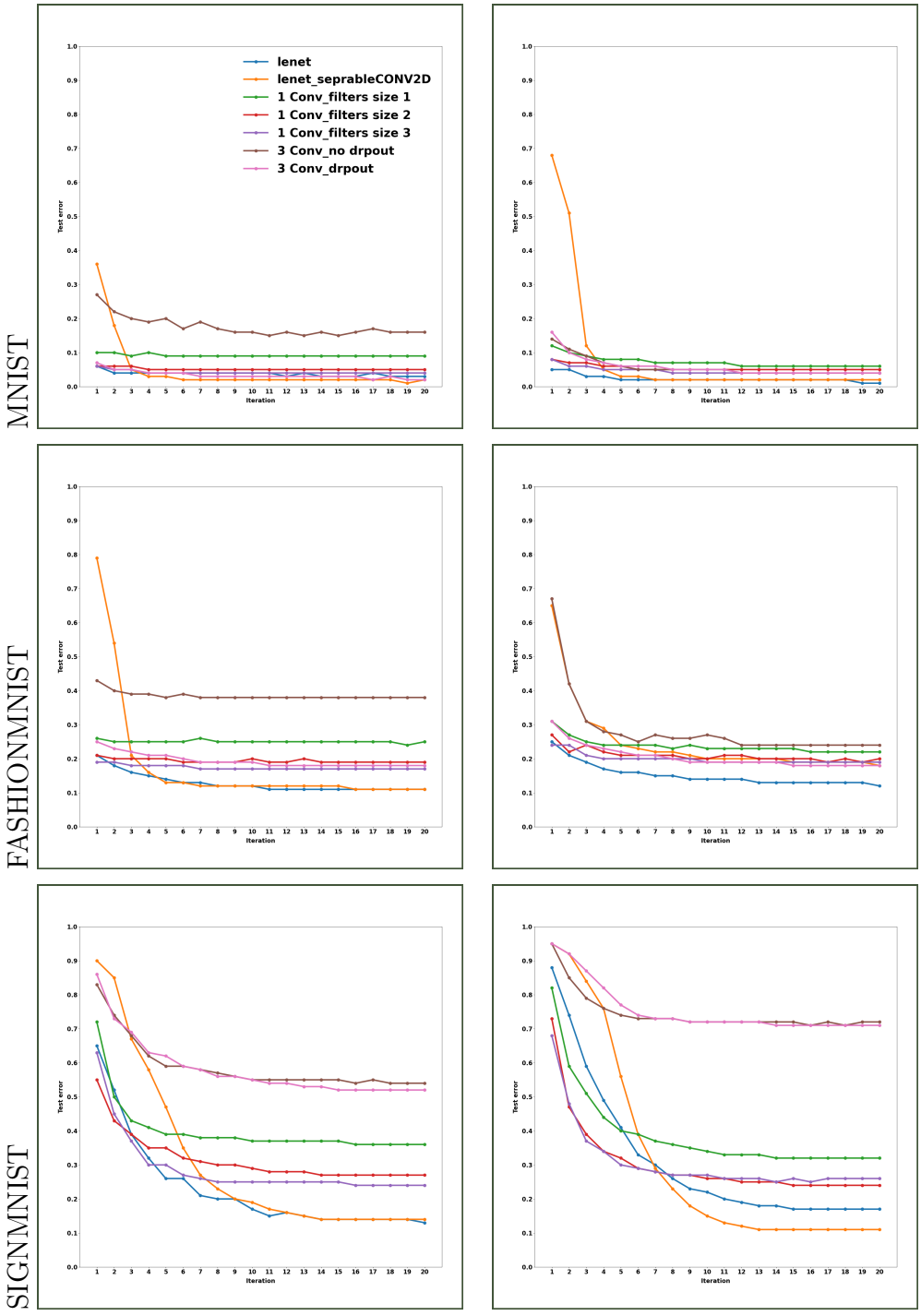


Figure 3.5: Different configurations of Neural net structures. (a) One neural net per class, (b) Twenty Neural nets per class.

into smaller, more manageable sub-tasks allows it to maintain high accuracy even in datasets with fine-grained distinctions.

This figure effectively visualizes the core advantage of Neural-SRNN and its capacity to decompose complex classification tasks into sub-tasks that separate neural networks within the model and can be individually handled. This decomposition enables the model to focus on specific features that are most relevant for each class, thus improving its overall performance. The clear delineation of important features in these figures provides a tangible demonstration of how Neural-SRNN's multi-centroid approach translates into more precise and interpretable classification outcomes.

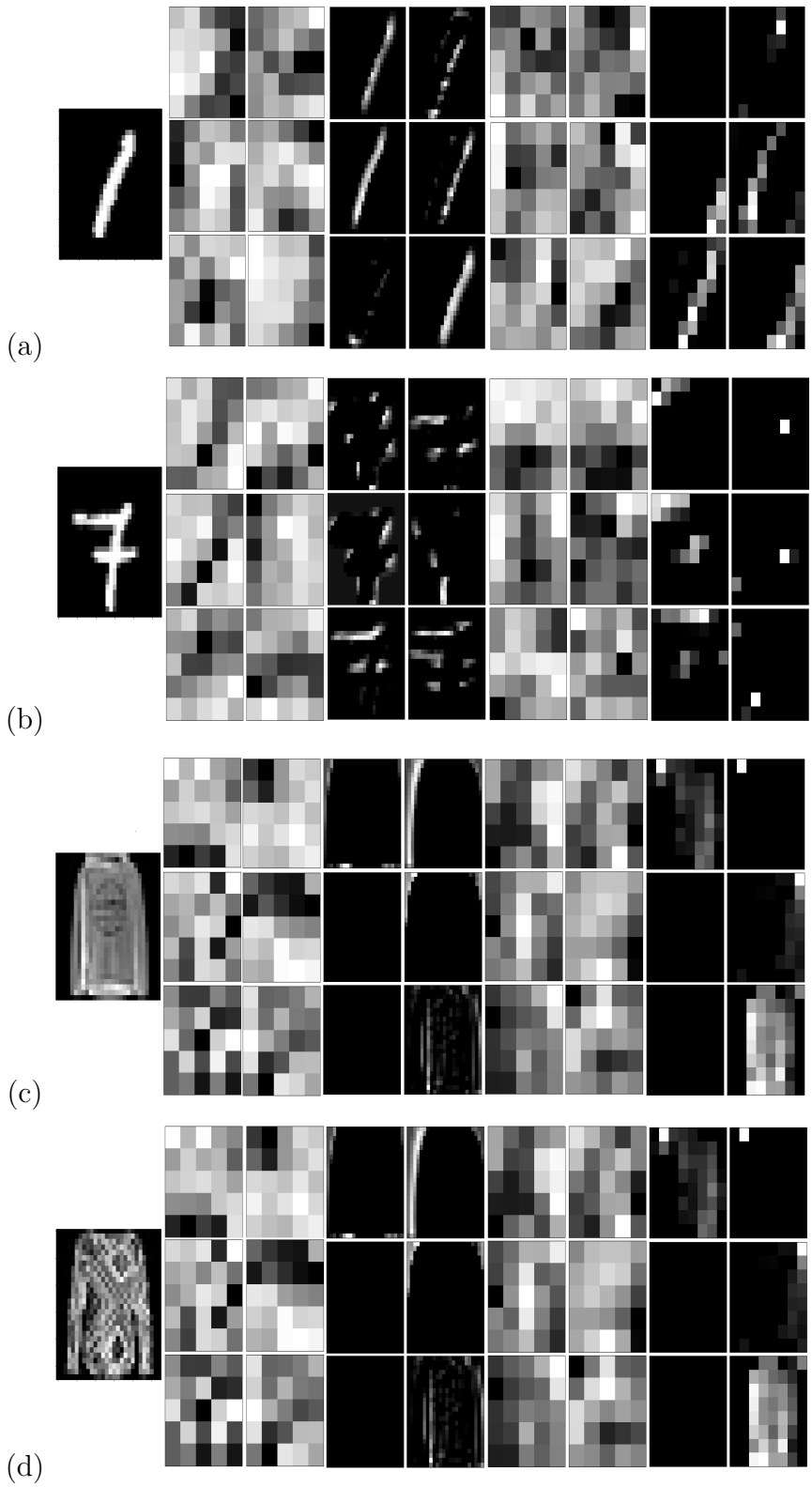


Figure 3.6: Feature importance of features by Neural-SRNN for (a)-(b) class 1 and class 7 in MNIST dataset and (c)-(d) class pullover and shirt from FASHION-MNIST.

3.4 Chapter Summary

In this chapter, we introduced a novel Neural Synthetic Reduced Nearest Neighbor (Neural-SRNN) approach designed to optimize the binary loss function through an innovative combination of k-means-like Expectation-Maximization (EM) algorithms and neural networks. The proposed method operates by dividing the dataset into multiple classes and further into sub-clusters, where each sub-cluster, or centroid, is represented by a distinct neural network. This allows the model to capture a wide range of variations within each class more effectively, which is particularly useful for datasets with complex intra-class differences.

The training process for Neural-SRNN involves two key steps: the Assignment step, where samples are assigned to the neural network that provides the highest probability, and the Maximization step, where each neural network is individually optimized in a binary classification manner to distinguish between the assigned samples and those assigned to other networks. This approach ensures that the model learns to focus on the most distinguishing features of each class.

Experimental results demonstrate that Neural-SRNN achieves superior or comparable performance to state-of-the-art models, such as Random Forests and ensemble models, across various datasets, including MNIST, Fashion-MNIST, and Sign-MNIST. The performance improvements are particularly evident in more complex datasets, where Neural-SRNN effectively captures subtle variations and maintains high accuracy. Furthermore, the chapter explores the impact of different neural network structures on model performance, highlighting the robustness of architectures like LeNet-5, especially when enhanced with separable convolutional layers and dropout techniques.

Overall, the Neural-SRNN approach showcases the significant potential for improving classification accuracy in complex datasets, offering a flexible and powerful tool for machine learning applications.

Chapter 4

Enhancing Synthetic Reduced Nearest-Neighbor with Two-Layer Neural Networks: A Step Forward in Image Classification

Synthetic Reduced Nearest Neighbor (SRNN) models, which operate exclusively on synthetic samples or prototypes, represent a significant advancement in nearest-neighbor algorithms. This innovation enhances model interpretability and optimization through specialized techniques. This chapter introduces the Two-Layer Neural-SRNN (TLN-SRNN) model for classification tasks, diverging from traditional Expectation Maximization (EM) methodologies. The TLN-SRNN significantly improves efficiency and scalability, outperforming traditional methods in both speed and accuracy. Our empirical findings demonstrate the model's rapid convergence and robust performance across diverse datasets, marking it as a notable innovation in machine learning.

4.1 Introduction

In the supervised and unsupervised learning modes, the process of segregating the clusters of the dataset is the primary assignment. It is mainly used in classification, which serves the purpose of subdividing the classes with the samples showing variability between them. Model accuracy is a function of the adoption of this granularity. As a result, models are capable of recognizing and adapting to the inherent differences while still being able to review large amounts of datasets. Classification tasks are fundamental to a myriad of machine learning applications, e.g., image recognition, language processing, and medical diagnosis. At the back of classification lies the clustering of datasets, which aims to expose latent patterns and categorize data points into distinct classes Chen et al. (2020).

The Synthetic Reduced Nearest Neighbor (SRNN) model joins a key simplification of clustering techniques. In the SRNN model, each prototype is the sub-cluster created by the learning algorithm. The use of this approach improves the user's understanding of the model as the prototype closest to the input signal is the cluster of input it represents. For example, in handwritten digit recognition, prototypes can catch style differences within categories consisting of the same digit, so they are appropriate and interpretable Saralajew et al. (2020).

The recent research of the Prototype Nearest Neighbor models has shown the diverse applications it can take, from even enhancing the adversarial robustness to making few-shot learning techniques more fine-tuned Allen et al. (2019). In spite of the development in these areas, mainstream technologies have yet to overcome the lack of model structures, which would be more efficient in the management of multiple cluster representations and similarity metrics. Such things illustrate the urgency of the introduction of better optimization strategies into SRNN models.

The popularity of nearest neighbor algorithms in machine learning is largely due to their simplicity and effectiveness across a wide range of applications. These models typically rely on distance metrics, such as Manhattan or Euclidean distances, to measure similarity between data points. However, as data patterns become more complex and clustering problems increase in difficulty, these traditional distance-based methods are often insufficient for capturing the intricacies of high-dimensional data De Berg (2000), Lloyd (1982b).

One of the fundamental challenges faced by nearest-neighbor algorithms is the "curse of dimensionality." In high-dimensional spaces, the concept of proximity be-

comes less meaningful, as the traditional distance measures used in these algorithms fail to adequately distinguish between similar and dissimilar points. Advanced data structures such as R-Trees, KD-Trees, and various spatial access methods have been developed to solve this problem Sammut and Webb (2017), Krizhevsky et al. (2012), He et al. (2016). Although these data structures do help in dealing with the inefficiencies caused by high-dimensional data, it is essential to bring in more enhancements to overcome the issues.

In reaction to the mentioned problems, different solutions were suggested by including techniques from EM in the nearest neighbor models, e.g., it is possible to use EM in the k-mean clustering. EM, which is a very sophisticated dual-step iterative algorithm, is designed especially to estimate latent variables (E-step) and optimize the model parameters (M-step) Singh (2005), Gupta et al. (2011). Apart from linear regression and analysis of the number of dimensions, neural networks have become a major success in classification and feature layer reduction. For instance, multi-stage Deep Learning (DL) methods have been nicely employed to restore signals from noisy measurements, which testifies to the robustness of the neural network in processing complex datasets Niranjani and Selvam (2020), Hinton and Salakhutdinov (2006).

The convergence of SRNN models and the neural networks is discussed in Alizadeh et al. (2022b) highlights the novel approach of the Neural-SRNN model, which uses several networks per class to boost the model's precision. In this approach, the classification process is handled by k networks for each of the m classes in the dataset $\mathbf{X}$. As for each of the m classes in the dataset, every data sample $\mathbf{x}_i$ in $\mathbf{X}$ is placed on these networks to be processed with the class identified as the one the network gives the highest output probability to. The neural-SRNN model, through its expectation-minimization-like problem during training, included the use of a two-step process: the first step is the assignment step, which assigns each sample to the most probable class, and the second step is the update step, which refines the network parameters for better accuracy. This hybrid approach is the integration of the strength of neural networks with specific class training that aims to effectively deal with the intricacies of multi-class classification. Nevertheless, it is possible that the method might overfit, especially as more centroids appear.

The Two-Layer Neural-SRNN (TLN-SRNN) model, the new method that is meant to overcome the time-consuming training of neural networks, is resolved through this dissertation. The TLN-SRNN model integrates dual-layer with SRNN, allowing the machine to interpret and compute efficiently. This design enables to both increase

the pace at which the training process happens and at the same time, deals with the complexity of the data, thus, the model demonstrates being able to not get overfitted and the adaptation to different sets.

A particularly innovative feature of the TLN-SRNN model is its extraordinary accuracy while significantly reducing the computational overhead, which is usually related to neural network training. The TLN-SRNN, which is a model with a streamlined architecture and simultaneously utilizes the best features of both the neural network model and SRNN, makes a strong breakthrough in the machine learning field. This model is set to provide significant insight into various applications, especially those that revolve around the implementation of large-scale and high-dimensional data, such as image recognition and natural language processing.

The TLN-SRNN - a novel model synthesized in this dissertation - is an answer to the complaints of traditional nearest neighbor and neural network models, which are observably good at classification-like tasks. The dual-layer architecture as an add-on to the model, in addition to making it the most scalable and adaptable one, also makes it the dominant one in solving modern machine-learning problems aggressively.

4.2 The Proposed Methods

TLN-SRNN operates on a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ represents an input sample. The data set is categorized into m classes, with the corresponding labels $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$. The TLN-SRNN model is distinctively structured in two layers, with a single Mini-CNN trained for each class in the first layer and an integrative shallow neural network in the second layer. The architecture begins with m Mini-CNNs, each designated to a distinct class. Each Mini-CNN processes the input $\mathbf{x}_i$ and outputs a probability vector, which indicates the probability of the input being associated with its respective class. Subsequently, a shallow neural network comes into play. This network takes the concatenated outputs from all the Mini-CNNs and carries out the final classification decision.

4.2.1 Model Architecture and Learning Objective

The TLN-SRNN model's architecture embodies a harmonious combination of depth and simplicity, integrating Mini-CNNs with a Shallow Network. Each Mini-CNN, comprising two convolutional layers equipped with Leaky ReLU activation

(negative slope of 0.01), is pivotal in maintaining an effective gradient flow for deep learning. These layers are adept at extracting class-specific features and uncovering intricate patterns within the data. An average pooling layer follows, efficiently reducing the feature dimensions while preserving vital information and leading to two fully connected layers for refined feature processing. In contrast, the Shallow Network adopts a straightforward design with two linear layers, augmented with leaky ReLU activation and a dropout mechanism. This configuration effectively synthesizes the extracted features, culminating in the model's final decision-making process. For more details on the architecture, please refer to Table 4.1.

Table 4.1: Layer-wise description of Shallow Net & MiniCNNs

Model	**Layer**	**Type**
Shallow Network	Input	-
	Fully Connected	Linear
	Dropout	-
	Fully Connected	Linear
	Output	-
MiniCNN	Input	-
	Convolution	Conv2d
	Convolution	Conv2d
	Pooling	AvgPool2d
	Fully Connected	Linear
	Fully Connected	Linear
	Output	-

To address dataset diversity, particularly with class imbalances, the TLN-SRNN model incorporates the advanced Focal Loss function. Evolving from standard Cross-Entropy Loss, Focal Loss introduces a modulation factor $(1 - p_t)^\gamma$, where p_t signifies the predicted probability for the actual class and γ in focal loss acts as a focusing parameter. It essentially modifies the standard cross-entropy loss, reducing the loss assigned to well-classified examples, thereby allowing the model to concentrate more on difficult cases. γ is generally a non-negative value (≥ 0), with typical values

ranging from 1 to 5. As γ increases, the model's focus on challenging examples is intensified. This factor recalibrates the loss, reducing contributions from easily classifiable examples and focusing on more complex, typically minority-class, cases. Consequently, Focal Loss not only counteracts class imbalances but also fosters a more balanced and effective learning paradigm. This approach ensures robust performance across varied class distributions, which is crucial for practical applications.

The TLN-SRNN model incorporates a dual-layered architecture for prediction and learning. The prediction function is given by:

$$\hat{\mathbf{y}}_{\text{TLN-SRNN}} = \text{ShallowNet}\left(\bigoplus_{i=1}^{m} \hat{\mathbf{P}}_i(\mathbf{x})\right), \tag{4.1}$$

where $\bigoplus$ denotes concatenation of output vectors from each Mini-CNN, $\hat{\mathbf{P}}_i(\mathbf{x})$ is the output probability vector of the i^{th} Mini-CNN for input $\mathbf{x}$, and ShallowNet$(\cdot)$ represents the shallow neural network function.

The prediction function of the TLN-SRNN model, as defined in Equation 4.1, illustrates how the model processes input data to generate class predictions. The model achieves this by first utilizing multiple Mini-CNNs (One per class), each responsible for extracting class-specific features from the input data. The output from each Mini-CNN is a probability vector, denoted as $\hat{\mathbf{P}}i(\mathbf{x})$, which represents the likelihood of the input belonging to various classes. These probability vectors are then concatenated using the operation $\bigoplus_{i=1}^{m}$, combining the insights from all Mini-CNNs into a comprehensive feature vector. This concatenated vector is subsequently fed into the ShallowNet- a shallow neural network-, which synthesizes the combined information and produces the final prediction $\hat{\mathbf{y}}_{\text{TLN-SRNN}}$. This architecture allows the model to leverage the diverse and detailed feature representations captured by the Mini-CNNs, leading to more accurate and robust classification outcomes.

The learning objective of the TLN-SRNN model, as expressed in Equation 4.2, serves as the cornerstone of the model's dual-layer architecture. This objective formalizes the training process as a minimization problem, reflecting the foundational principle of supervised learning: reducing the discrepancy between predicted and actual outcomes. The goal is to optimize the parameters of both the Mini-CNNs, denoted by Θ, and the ShallowNet, denoted by Φ, to minimize the loss function $\mathcal{L}(\cdot)$. The loss function measures the difference between the predicted class labels and the actual labels across all training samples. By summing the loss over the entire training dataset, the model is guided to adjust its parameters to reduce this discrepancy,

thereby improving its predictive accuracy.

Specifically, the function $f^j_{\text{cnn}}(\mathbf{x}_i; \Theta_j)$ represents the output of the j^{th} Mini-CNN when processing the input $\mathbf{x}_i$. The outputs from all Mini-CNNs are then concatenated and passed to the ShallowNet for final classification. Through the joint optimization of the parameters Θ and Φ, the TLN-SRNN model effectively integrates the features extracted by the Mini-CNNs, ensuring that these features contribute meaningfully to the final prediction. This holistic approach to learning enables the model to achieve accurate predictions while being well-suited to handle the complexities of diverse datasets.

$$\min_{\Theta,\Phi} \sum_{i=1}^{n} \mathcal{L}\left(\text{ShallowNet}\left(\bigoplus_{j=1}^{m} f^j_{\text{cnn}}(\mathbf{x}_i; \Theta_j)\right), \mathbf{y}_i; \Phi\right), \tag{4.2}$$

In this formulation:

- $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_m\}$ represents the set of parameters for each Mini-CNNs in the first layer. These parameters are crucial for capturing the nuanced features specific to each class. By optimizing Θ, the model learns to extract the most informative features from the input data $\mathbf{x}_i$ that indicate each particular class.

- Φ embodies the parameters of the ShallowNet, the second layer's neural network. The optimization of Φ is pivotal for effectively synthesizing the individual insights gained from the Mini-CNNs. This process involves integrating the class-specific probabilities and making a cohesive decision about the input's class.

- $\mathcal{L}(\cdot)$ denotes the loss function, a measure of the difference between the actual label $\mathbf{y}_i$ and the predicted label output by the model. This function quantifies the accuracy of the model's predictions and guides the training process. A well-chosen loss function aligns the model's predictions with the true labels, driving the learning towards higher accuracy.

- The operation $\bigoplus_{j=1}^{m} f^j_{\text{cnn}}(\mathbf{x}_i; \Theta_j)$ concatenates the outputs of all Mini-CNNs, each processing the input $\mathbf{x}_i$. This concatenated output forms a comprehensive feature representation of the input, reflecting its relation to all classes.

The minimization in Eq. (4.2) is performed over the combined parameter space of both Mini-CNNs and ShallowNet. Through this joint optimization, the TLN-SRNN model learns to extract relevant features for each class and effectively incorporates

these features to make accurate class predictions. This holistic approach to learning ensures that the features extracted by the Mini-CNNs are directly applicable and beneficial for the final classification task performed by the ShallowNet.

4.2.2 Rationale Behind the Architectural Choices

The motivation for adopting a single mini Convolutional Neural Network (mini CNN) per class, as opposed to multiple neural networks per class, stems from our aim to enhance model interpretability while reducing computational complexity. The approach proposed in [17] involved using multiple neural networks per class, which, although effective in improving classification accuracy, leads to a substantial increase in computational and memory overhead. In contrast, our two-layer neural-SRNN (TLN-SRNN) model employs a streamlined architecture to achieve similar accuracy with reduced resource consumption.

4.2.3 Selection Process

The selection of the mini CNN architecture was a rigorous process involving extensive experimentation and hyperparameter tuning. We used Optuna, a hyperparameter optimization framework, to systematically explore various combinations of shallow architectures. This included varying the number of convolutional layers, the type of activation functions, and the pooling strategies. The critical criteria for selection were maintaining high classification accuracy while keeping the computational costs low. The chosen architecture, which consists of two convolutional layers followed by average pooling and fully connected layers, demonstrated the best performance across all evaluated metrics.

4.2.4 Resource Efficiency Benefits

One of the primary advantages of using a single mini CNN per class is the significant reduction in resource costs. The streamlined architecture reduces the number of parameters and accelerates the training process. This is particularly beneficial in scenarios where computational resources are limited or rapid training is essential. Our experimental results, detailed in the supplementary materials, highlight these benefits. For instance, the TLN-SRNN model's training time is notably lower than the multi-model approach [17] without compromising the accuracy.

4.2.5 Interpretability and Adaptability

In addition to resource efficiency, our approach enhances model interpretability. Using a single mini-CNN per class makes the decision-making process more transparent, allowing for better understanding and trust in the model's predictions. This is particularly important in applications where interpretability is crucial. Furthermore, the modular nature of our architecture allows for easy adaptation to different datasets. Should the dataset change, the same selection and validation process can be applied to fine-tune the architecture, ensuring its suitability for the new data characteristics.

The Two-Layer Neural-SRNN (TLN-SRNN) model enhances interpretability by strategically structuring its architecture to allow for more transparent and understandable decision-making processes. The use of a single Mini-CNN per class simplifies the model's complexity, making it easier to trace how specific inputs lead to particular outputs. Each Mini-CNN is dedicated to a single class, which makes the features learned by the model directly relevant to that class, thereby simplifying the understanding of the decision-making process. The shallow network that integrates the outputs from these Mini-CNNs further enhances interpretability by maintaining a simple and transparent architecture.

Moreover, the modularity of the TLN-SRNN allows for adaptability. The architecture can be fine-tuned for different datasets, ensuring that the model remains both interpretable and effective even as the data characteristics change. This adaptability, combined with the model's inherent interpretability, makes the TLN-SRNN particularly valuable in applications where both accuracy and transparency are critical.

4.2.6 Contributions and Advantages

The Two-Layer Neural-Synthetic Reduced Nearest-Neighbor (TLN-SRNN) model introduced in this work marks a novel and significant advancement in the domain of image classification. Its unique dual-layered architecture, combining individual Mini-CNNs per class with an integrative shallow neural network, not only enhances the interpretability of the classification process but also improves scalability and efficiency. Unlike traditional methods, the TLN-SRNN model strategically separates class-specific feature extraction from the final decision-making process. This separation results in several key advantages:

- **(i) Improved Efficiency:** The TLN-SRNN model demonstrates a remarkable increase in training efficiency, showcasing nearly an order of magnitude

faster operation than the baseline Neural-SRNN model without compromising accuracy. This significant improvement in efficiency is largely attributed to the architectural design, which optimizes the use of computational resources by streamlining the training process. Each Mini-CNN within the TLN-SRNN model is focused on a specific class, reducing the complexity typically associated with training deep, monolithic networks. By compartmentalizing the learning process into smaller, more manageable tasks, the model is able to process data more quickly and effectively. Additionally, the shallow network used for final decision-making further reduces the computational burden, ensuring that the model can achieve high accuracy with a fraction of the computational resources required by more complex models. This efficiency not only accelerates the training phase but also makes the TLN-SRNN model more scalable, allowing it to handle larger datasets and more complex problems without a proportional increase in computational cost.

- **(ii) Enhanced Interpretability:** The TLN-SRNN model's architectural design significantly contributes to its interpretability, which is crucial in domains where understanding the rationale behind model predictions is vital, such as healthcare or finance. By employing a single Mini-CNN per class, the model isolates the feature extraction process for each class, making it easier to trace how specific input features influence the final classification decision. This class-specific feature extraction allows practitioners to understand which features are most influential for each class, thereby providing transparency in the decision-making process. Furthermore, the integration of these class-specific features through a shallow neural network enhances interpretability by maintaining a straightforward decision pathway. This simplicity contrasts with deeper and more complex models where the decision process can become opaque. The clear, modular structure of TLN-SRNN not only simplifies debugging and model tuning but also enables more effective communication of how the model works to stakeholders, which is essential for trust and adoption in critical applications.

- **(iii) Robustness to Class Imbalance:** Class imbalance is a common challenge in many real-world datasets, where some classes are significantly underrepresented compared to others. This imbalance can lead to biased models that favor the majority class, resulting in poor generalization to minority classes. The TLN-SRNN model addresses this issue through the adoption of the Focal

Loss function, which modifies the standard cross-entropy loss to focus more on hard-to-classify examples, particularly those from minority classes. By incorporating a modulation factor $(1 - p_t)^\gamma$ into the loss function, where p_t is the predicted probability for the actual class and γ is a focusing parameter, the model down-weights the loss contribution of well-classified examples. This encourages the model to pay more attention to challenging cases, effectively mitigating the impact of class imbalance. The result is a more balanced learning process that enhances the model's ability to perform well across all classes, including those with fewer examples. This robustness makes TLN-SRNN particularly suitable for applications where class distribution is skewed, ensuring that the model remains effective and fair in its predictions.

- **(iv) Robustness to Overfitting:** The robustness of the TLN-SRNN model against overfitting can be attributed to several key factors embedded within its architecture and training methodology. The TLN-SRNN model employs a modular structure, wherein each Mini-CNN is dedicated to processing class-specific features, followed by a shallow network that synthesizes these features for final classification. This modular approach helps to prevent overfitting by reducing the complexity typically associated with deep, monolithic networks. By focusing each Mini-CNN on a specific class, the model can better capture relevant features without being overwhelmed by the noise or irrelevant patterns that often lead to overfitting in more complex architectures.

  Furthermore, the use of Focal Loss within the TLN-SRNN model plays a crucial role in enhancing its robustness. Focal Loss is designed to focus training on harder-to-classify examples, particularly those from minority classes. This focus prevents the model from becoming overly confident in easy-to-classify samples, a common issue that can lead to overfitting. By dynamically adjusting the importance of each sample during training, Focal Loss ensures that the model remains attentive to challenging cases, thereby promoting better generalization to unseen data.

  The TLN-SRNN model's architectural design also supports robustness through its dual-layer structure, which balances depth and simplicity. The shallow network that follows the Mini-CNNs provides an additional layer of abstraction, helping to generalize the features learned by the Mini-CNNs without introducing excessive complexity. This balance is critical in avoiding overfitting, as it

allows the model to learn effectively from the training data while maintaining the capacity to generalize to new, unseen data.

The TLN-SRNN model effectively avoids overfitting by leveraging its modular design, the strategic use of Focal Loss, and a well-balanced architecture that prioritizes both feature specificity and generalization. These components collectively enhance the model's robustness, ensuring it delivers consistent and reliable performance across various datasets. This robustness is particularly evident in the model's sustained accuracy on complex datasets like Fashion MNIST and Sign Language MNIST, where overfitting often challenges less sophisticated models.

- **(v) Adaptability and Flexibility:** The modular design of the TLN-SRNN model offers significant adaptability and flexibility, making it highly versatile across different datasets and application domains. Each Mini-CNN within the model can be independently tuned or replaced, allowing for targeted adjustments that optimize performance for specific datasets. This modularity is especially advantageous when dealing with datasets that have unique characteristics or when transferring the model to new domains. The architecture's flexibility also extends to its ability to accommodate changes in the dataset, such as the addition of new classes or the availability of more data, without requiring a complete redesign of the model. This adaptability ensures that TLN-SRNN can evolve alongside the data, maintaining high performance even as the underlying data distribution changes. This feature is particularly beneficial in dynamic environments where datasets are constantly evolving, enabling the model to remain relevant and effective over time.

These innovative features underscore the TLN-SRNN model's potential as a significant step forward in machine learning for image classification, combining speed, accuracy, robustness, and interpretability in one cohesive framework.

4.3 Experimental Results

4.3.1 Datasets

To demonstrate the efficacy of the proposed strategy in dealing with varied datasets. To train all of the models, three different datasets were chosen: **MNIST**

Deng (2012), **Fashion-MNIST** Xiao et al. (2017) and **sign-MNIST** Huang et al. (2015).

- **MNIST**: is a large database of handwritten digits 0 to 9 and is widely used for classification algorithms. The number of features is 784. The train and test set contains 60000 and 10000 samples, respectively.
- **Fashion-MNIST**: is a dataset consisting of grayscale images from 10 different classes such as T-shirts/tops, Pullovers, Coats, and so on. The number of features is 784. The train and test set contains 60000 and 10000 samples, respectively.
- **Sign-MNIST**: the American Sign Language letter database consists of 24 classes for each letter of the alphabet, excluding J and Z. The number of features is 784. The train and test set contains 27455 and 7172 samples, respectively.

4.3.2 Experimental Setup

To rigorously evaluate the TLN-SRNN model's efficiency, especially in classification tasks, we conducted empirical comparisons with Neural-SRNN Alizadeh et al. (2022b). The choice of classification as our focal point is informed by its pivotal role in machine learning and the existence of established, reproducible architectures in this domain.

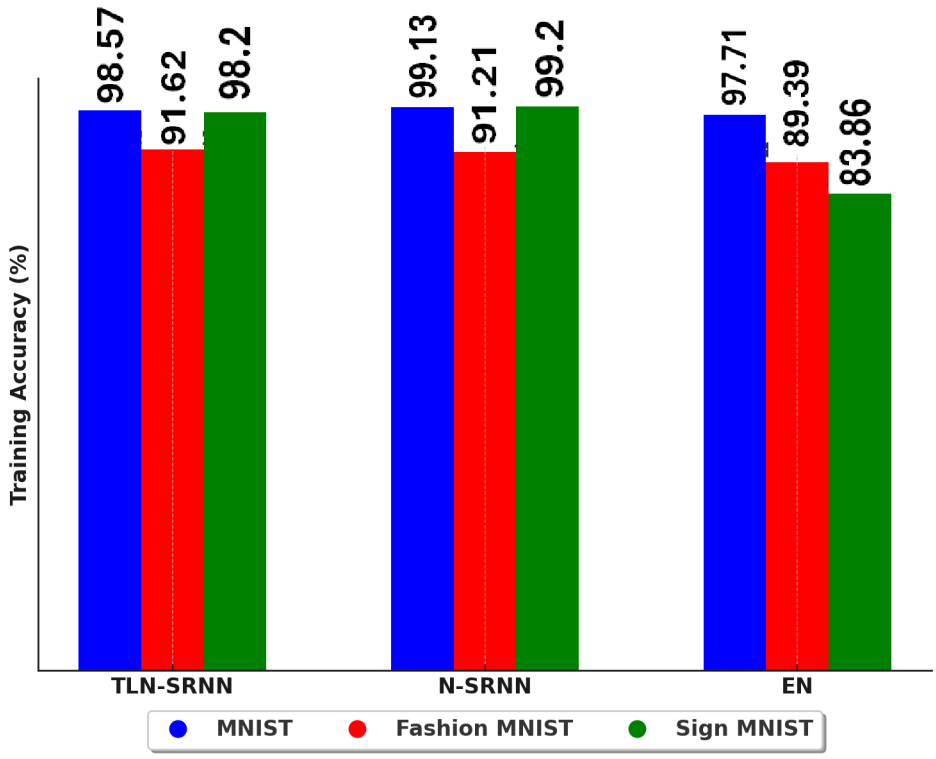
In our experimental analysis, the TLN-SRNN model showcased its robust architecture, integrating Mini-CNNs for intricate feature extraction and a Shallow Network for streamlined decision-making. The Mini-CNNs, with their deep feature analysis capabilities, were complemented by the efficiency of the Shallow Network, effectively balancing the feature extraction process with classification tasks. The TLN-SRNN's architecture, which combines depth (through Mini-CNNs) and simplicity (through the Shallow Network), enables it to learn complex patterns more effectively than the ensemble of shallow networks. A critical component in enhancing the model's performance, particularly in unbalanced dataset scenarios, was the adoption of Focal Loss. The use of Focal Loss in TLN-SRNN helps in focusing on harder-to-classify examples, which can be particularly beneficial in datasets with class imbalances. This is evidenced by its performance on the Fashion MNIST dataset.

In this research, Optuna Akiba et al. (2019) played a pivotal role in hyperparameter tuning for the TLN-SRNN model, leveraging its Bayesian optimization to navigate high-dimensional spaces efficiently. This targeted tuning was applied to the utilized datasets, ensuring a model finely attuned to each dataset's unique characteristics. The iterative refinement of hyperparameters, informed by Optuna's insights during the training loop, led to tailored parameters that significantly boosted the model's classification performance on each dataset. To effectively demonstrate the efficacy of the proposed TLN-SRNN method, it is systematically benchmarked against various established architectures frequently cited in contemporary literature. Uniformity is maintained across all models regarding batch sizes and the number of epochs, further reinforcing the comparative evaluation's validity. This methodical approach guarantees that any observed performance differentials are attributable to the respective models' inherent merits rather than external variances in their configuration or training regimen.

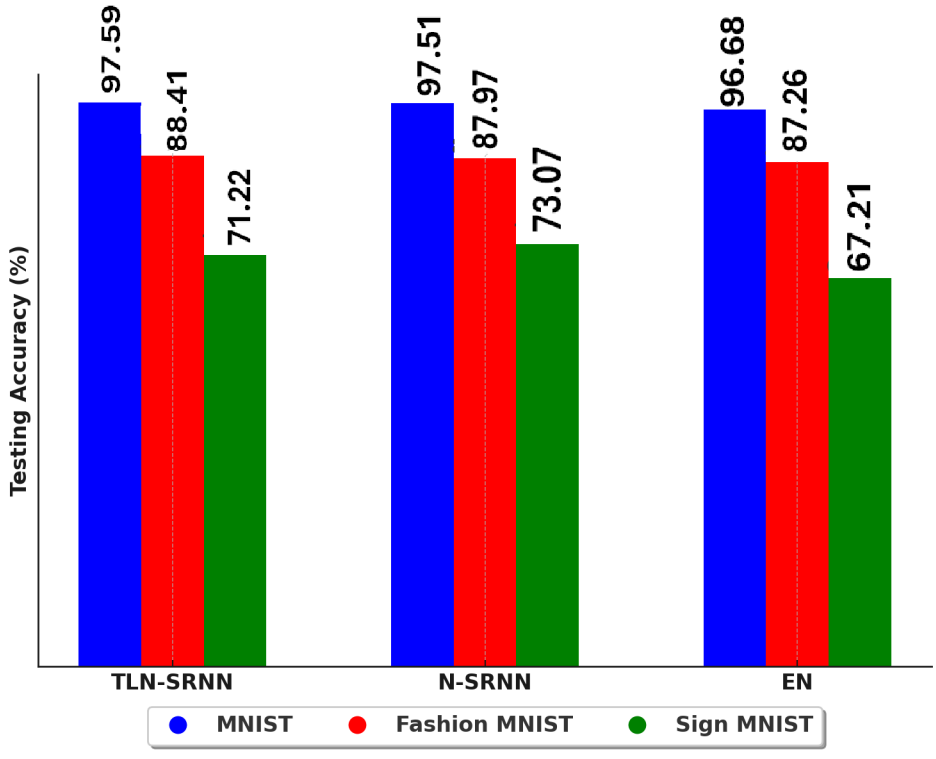
Ensemble Model employs a trio of shallow neural networks. We used the same shallow network structure used in TLN-SRNN to make the results more comparable. Moreover, all the hyperparameters are exactly as TLN-SRNN. These networks are independently trained on distinct subsets, each constituting one-third of the randomly selected samples from the original training set. The ensemble first elicits predictions from each constituent model on the training dataset. These individual predictions are then collectively synthesized, with the final ensemble prediction for each data instance being determined by an aggregation method, specifically by summing the predicted probabilities across all classes and selecting the class with the maximal cumulative probability. This approach is designed to augment the overall predictive accuracy, capitalizing on the diverse insights and strengths of the individual models, thus yielding a more comprehensive and reliable understanding of the training data.

4.3.3 Results

As presented in Fig. 4.1(a), the TLN-SRNN model's performance is notably on par with that of the N-SRNN, with a negligible accuracy difference **below** 1%, despite the latter's adoption of the more profound LeNet-5 architecture, which is a more expensive structure in terms of time and resources. This observation underscores the TLN-SRNN's proficiency in extracting discriminative features using a more computationally efficient shallow structure.



(a) Training Accuracy



(b) Testing Accuracy

Figure 4.1: TLN-SRNN accuracy on the train set (a) and test set (b)

Conversely, the Ensemble Model (EN), despite employing a shallow network architecture and multiple learner integration, did not quite match the TLN-SRNN's accuracy. The EN model achieved training accuracies of 97.71%, 89.39%, and 83.86% on MNIST, Fashion MNIST, and Sign MNIST, respectively, which are notably lower than TLN-SRNN's accuracies. This suggests that the depth and sophistication of individual learners, as seen in TLN-SRNN, are pivotal for achieving optimal performance. The subtle yet telling distinctions in accuracy among these models illustrate the delicate balance between network depth and ensemble strategies in achieving high-performance benchmarks. Fig. 4.1(b) illustrates the accuracies for the EN, TLN-SRNN, and N-SRNN models on the MNIST, Fashion-MNIST, and Sign Language MNIST test sets. The TLN-SRNN model demonstrates commendable accuracy, particularly achieving 97.59%. The training duration comparison between TLN-SRNN and N-SRNN models is quantified in Fig. 4.2. The TLN-SRNN's training time is significantly lower across all datasets, notably on MNIST and Sign MNIST, where the N-SRNN's training time is almost six times greater. This efficiency in training time with TLN-SRNN, while maintaining comparable accuracy as observed in previous results, underscores the model's optimized computational design, making it particularly advantageous for scenarios where training efficiency is paramount. Conversely, the N-SRNN's longer training time can be attributed to its more complex architecture, which, despite yielding slight accuracy improvements, may not justify the computational overhead in specific practical applications.

The ensemble model has significantly lower training times across all datasets. This is expected due to its simpler architecture and the fact that it trains on subsets of the data. An in-depth analysis of the predictive performance metrics for N-SRNN and TLN-SRNN models unveils pivotal insights, particularly when considering the intricacies of the Fashion MNIST and Sign MNIST datasets. The N-SRNN model exhibits stellar proficiency during training across all datasets, with precision, recall, and F1 scores nearing the zenith of perfection. However, this trend experiences a decrease in the test phase, with the Sign Language MNIST data set highlighting the most significant dip in the F1 score to 74%. Such a shift from training to testing landscapes suggests a potential overfitting pitfall, where the model may not generalize well to unseen data, especially when it presents with a higher degree of complexity or variability.

On the contrary, the TLN-SRNN model demonstrates an impressive equilibrium between training and testing performance. Although it slightly lags behind N-SRNN

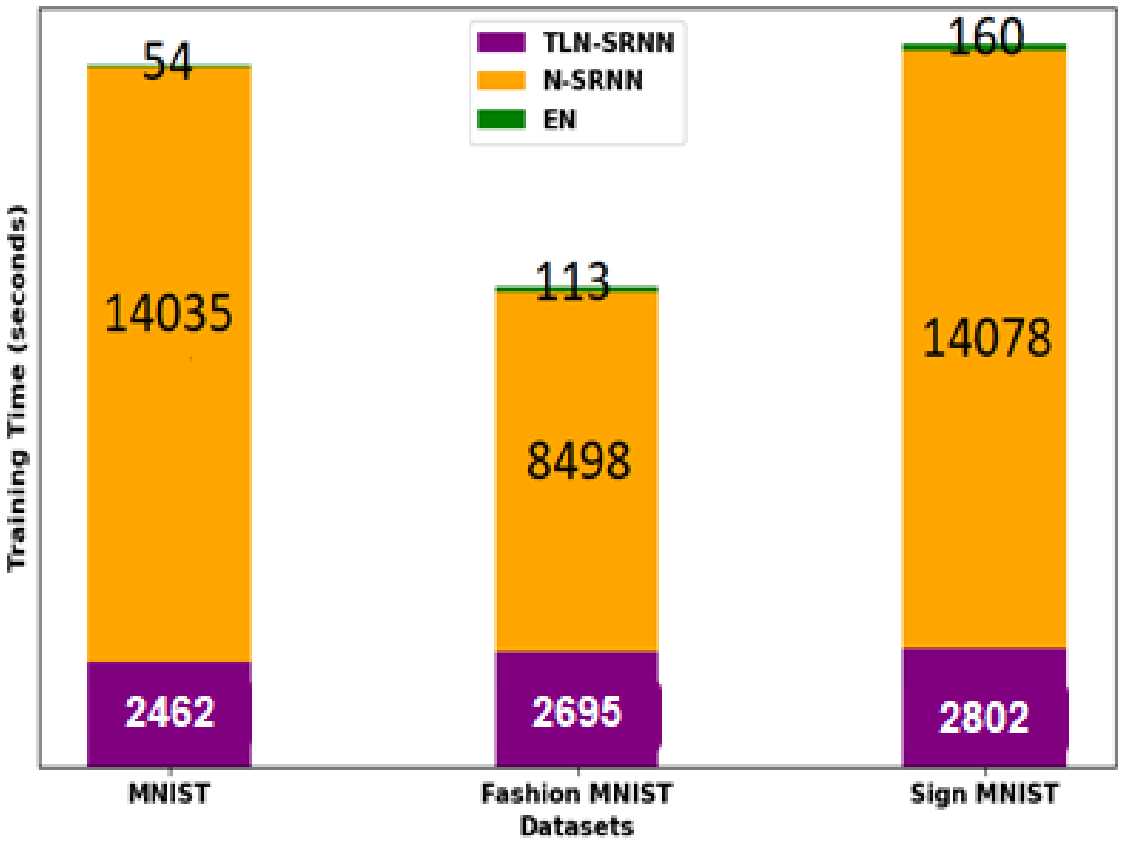


Figure 4.2: TLN-SRNN Training time vs. N-SRNN Training time.

in training metrics, its test performance metrics exhibit less volatility, particularly within the Fashion MNIST and Sign Language MNIST datasets. This reduced volatility in test performance is particularly noteworthy and suggests that the TLN-SRNN model has superior generalization capabilities compared to the N-SRNN. While the TLN-SRNN may not achieve the highest possible training accuracy, this slight lag is more than offset by its ability to perform consistently well on unseen data, as evidenced by its stable performance across different datasets.

This stability is especially apparent in complex datasets like Fashion MNIST and Sign Language MNIST, where the model avoids the sharp declines in performance often observed with overfitting-prone models. The TLN-SRNN's architectural design is a key factor in achieving this balance between training and test performance. By employing class-specific Mini-CNNs in combination with a shallow neural network, the model effectively captures relevant features while maintaining a simpler and more interpretable structure. This modular approach not only reduces the risk of overfitting

but also allows the model to adapt to the inherent complexity and variability present in datasets like Fashion MNIST and Sign Language MNIST. Consequently, the TLN-SRNN delivers reliable and consistent results, making it a strong candidate for real-world applications where stability and generalization are critical.

Table 4.2: Performance comparison of models across different datasets

(a) Train Set Metrics

Dataset	**Model**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
MNIST	N-SRNN	99.13	0.99	0.99	0.99
	TLN-SRNN	98.57	0.98	0.98	0.98
Fashion MNIST	N-SRNN	91.21	0.91	0.91	0.91
	TLN-SRNN	91.62	0.92	0.92	0.92
Sign Language MNIST	N-SRNN	99.20	0.99	0.99	0.99
	TLN-SRNN	98.20	0.98	0.98	0.98

(b) Test Set Metrics

Dataset	**Model**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
MNIST	N-SRNN	97.51	0.98	0.98	0.98
	TLN-SRNN	97.59	0.97	0.97	0.97
Fashion MNIST	N-SRNN	87.97	0.88	0.88	0.88
	TLN-SRNN	88.41	0.88	0.88	0.88
Sign Language MNIST	N-SRNN	73.07	0.78	0.73	0.74
	TLN-SRNN	71.22	0.73	0.71	0.71

The steadiness of TLN-SRNN's precision and recall from training to testing indicates robust consistency, which is crucial for practical deployment where performance predictability is as valuable as the performance itself. Moreover, with test precision and recall metrics for Fashion MNIST slightly surpassing those of the N-SRNN, the TLN-SRNN model emerges not only as a swift learner due to its markedly lower train-

ing times but also as a dependable predictor in diverse conditions, as shown in Table 4.2a and Table 4.2b. This balancing act between agility in learning and steadfastness in prediction underscores the TLN-SRNN model's potential as a viable candidate for real-world applications where time and reliability are of the essence.

Furthermore, the TLN-SRNN consistently achieves higher accuracy on both the training and test sets for the MNIST and Fashion MNIST datasets, indicating its superior capability to capture underlying patterns in the data. The integration of Mini-CNNs with the Shallow Network enables the TLN-SRNN to extract more nuanced features, leading to enhanced performance across both datasets.

The learning figures for the TLN-SRNN model, as presented in Fig. 4.3, provide a visual narrative of the model's training efficacy across diverse datasets. For MNIST, the model's training loss decreases steadily, reflecting effective learning and a capacity for generalization, given the uneventful reduction in loss across epochs. The Fashion MNIST displays a modest yet consistent decline in loss, indicating a slower adaptation to the dataset's complexity. Sign Language MNIST showcases a significant drop in loss despite starting from a higher initial loss, denoting the model's ability to discern intricate patterns within the data. Overall, the TLN-SRNN model's learning figures signify its adeptness at capturing dataset-specific features and improving predictions iteratively. These figures demonstrate the model's versatility and underscore its potential as a reliable tool for complex image recognition tasks in diverse domains.

4.3.4 Complexity Analysis of TLN-SRNN

The TLN-SRNN employs a single Mini-CNN per class, each consisting of two convolutional layers followed by average pooling and fully connected layers. This design efficiently extracts class-specific features while maintaining a manageable number of parameters. Each Mini-CNN comprises 2,012 parameters, while the shallow network contains 1,354 parameters. In contrast, the Neural-SRNN model described in [17] consists of 5,164 parameters. This parameter count is nearly double that of the TLN-SRNN, resulting in a significant reduction in parameters for the TLN-SRNN, thereby enhancing its computational efficiency and speed.

The outputs of the Mini-CNNs are concatenated and fed into a shallow neural network with two dense layers, which synthesize the extracted features to make a final classification decision. The shallow network's simplicity ensures that the additional computational burden is minimal. Compared to traditional deep neural networks,

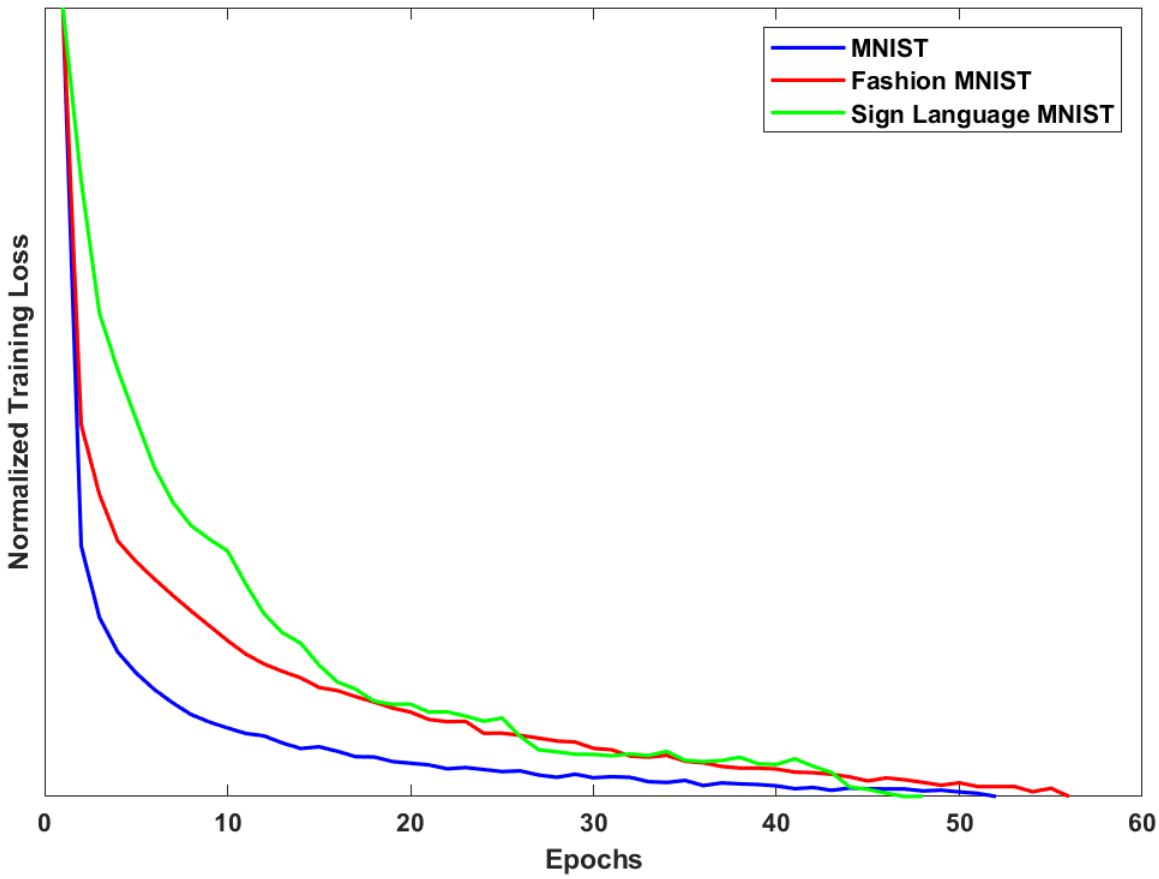


Figure 4.3: Learning figures of the TLN-SRNN model on MNIST, Fashion MNIST, and Sign MNIST.

our approach significantly reduces the number of parameters by limiting the depth and width of each Mini-CNN and using a shallow network for final decision-making. This reduction in parameter count leads to faster training and inference times.

Moreover, our architecture's modular nature, with separate Mini-CNNs for each class, allows for parallel training and inference. This parallelism enhances scalability, making the model suitable for large-scale and high-dimensional datasets.

4.4 Chapater Summary

This study introduced the Two-Layer Neural-Synthetic Reduced Nearest-Neighbor (TLN-SRNN) model, which addresses the limitations of traditional SRNN and ensemble models by enhancing efficiency, scalability, and accuracy. The TLN-SRNN achieves substantial reductions in training times while maintaining high accuracy. Its

modular architecture allows for parallel training and inference, making it suitable for large-scale and high-dimensional datasets.

The model enhances interpretability by employing a single Mini-CNN per class, providing clear insights into the decision-making process. This transparency is valuable for critical applications requiring a deep understanding of model predictions. Additionally, the use of Focal Loss addresses class imbalances, ensuring balanced learning across diverse datasets.

Empirical results demonstrate the TLN-SRNN's superior performance, consistently outperforming the ensemble model in accuracy while significantly reducing training time. The TLN-SRNN's unique dual-layer architecture balances computational efficiency, predictive accuracy, and interpretability, making it practical for scenarios where time and computational resources are critical. Its strength is its robustness and versatility across various image data types, such as fashion items and sign language.

The model's design facilitates efficient feature extraction through Mini-CNNs, followed by a Shallow Network synthesizing these features for final classification. This approach accelerates training and effectively handles complex data, showcasing robustness against overfitting and superior adaptability to diverse datasets.

Targeted hyperparameter tuning with Optuna finely tuned the model for each dataset, significantly boosting classification performance. This systematic approach ensures that performance differentials are due to the respective models' inherent merits rather than external variances in their configuration or training regimen.

In conclusion, the two-layer neural-SRNN model represents a notable innovation in image classification. Its efficient and scalable architecture, combined with enhanced interpretability and robustness, underscores its potential for broad application in machine learning. The TLN-SRNN model is a compelling choice for future research and practical implementations, offering a reliable and efficient solution for complex classification tasks in various domains, from edge computing to large-scale image analysis systems. This combination of speed, precision, and insight positions the TLN-SRNN model as a noteworthy contender for advancements in the field.

Chapter 5

Enhancing Irrigation Efficiency with a Unified Stochastic Decision Tree Model: Predictive Analysis of Stem Water Potential in Almond and Pistachio Orchards

Stem Water Potential (SWP) is the standard method to assess water stress and irrigation scheduling in tree crops. This method is time-consuming and labor-intensive, limiting data collection to only a few trees in the orchard. To find an alternative approach that predicts water stress in every tree in the orchard, we implemented a novel Stochastic Decision Tree (SDT) method, utilizing remote sensing and weather data to predict SWP in almond and pistachio orchards. The input data for our model included various vegetative indices such as NDVI, GNDVI, OSAVI, LCI, and NDRE, as well as local weather parameters, such as temperature (T_a), relative humidity (RH), air pressure (P), Vapor Pressure Deficit (VPD) and the Water Stress Index (WSI). Our results indicate that the SDT model achieves a prediction accuracy of nearly 94%, Outperforming Random Forest (RF), Support Vector Machine (SVM), and the K-nearest neighbor (KNN) algorithms. We investigated various combinations of collected data under different scenarios to improve the impact of sensor-derived data from pistachio and almond orchards and enhance the accuracy of SWP predictions using an SDT model. Our findings suggest that a data-driven model utilizing cost-effectively collected data can predict water stress. The successful development of a universal model improves the accuracy of SWP predictions. Moreover, its adaptability and effectiveness allow it to be utilized for different orchards, making it highly applicable to real-world agricultural scenarios.

5.1 Introduction

Agriculture is the world's largest consumer of fresh water, requiring dependable and resilient fresh-water availability for global food security D'Odorico et al. (2020). As the most prominent user of water resources, the agricultural industry wastes a lot of water due to inefficiency in irrigation. California has $50 billion annual income by producing 66% of all the US growing fruits and nuts. Nut trees are water-intensive, and unlike seasonal crops, they cannot be followed in a dry year. Drought and water shortage are major concerns in California, and Statistics show California's agricultural production's future is uncertain because of water scarcity Medellín-Azuara et al. (2022), Alizadeh et al. (2023a).

Water stress has different impacts on different types of plants, exclusively within photosynthetic limits. Prolonged water stress will severely decrease productivity and plant growth. Therefore, farmers are looking for ways to better utilize available water. Precision irrigation that applies the right amount of water at the right time could potentially be an ultimate approach to saving water; however, some of the needed technologies are currently missingAlizadeh et al. (2023b). One of the requirements of precision irrigation is knowing the exact water status of each tree in the orchard. The deficit irrigation method is another approach to saving water, but this technique relies on ET. But, typically, ET is measured for the whole orchard, not per treeAlizadeh et al. (2018, 2021).

Currently, the Stem Water Potential (SWP) is known as the gold standard method Savchik et al. (2024) for measuring the water status of tree crops and is used in irrigation scheduling methods Ohana-Levi et al. (2022), Carrasco-Benavides et al. (2022), Zhao et al. (2017), Gutiérrez-Gordillo et al. (2021). However, this technique is very labor-intensive and time-consuming. Not only it is practically impossible to collect precise data from all trees in a large orchard, also the collected data is subject to error and depends on the user's observation while reading the pressure chamber device.

Recently, implementing machine learning (ML) techniques in agriculture has shown considerable interest. ML techniques offer boundless potential to solve critical issues in the agricultural sector, such as drought and water stress detection. Many studies have been made on "smart farming or precision farming," which aims to increase both the quality and the quantity of agricultural output by making farming operations more "connected" and "intelligent" Jha et al. (2019), Banerjee et al. (2018),

Smith (2018). ML-based applications estimate evapotranspiration daily, weekly, or monthly, which can enhance the irrigation system efficiency. By utilizing sophisticated algorithms and data-driven models, ML methods enable the analysis of large-scale agricultural datasets to better water management practices, optimize irrigation strategies, and increase crop productivity in response to a changing climate Waleed et al. (2020), Eli-Chukwu (2019).

Traditional drought monitoring and evaluation methods rely on limited, localized data. This makes it challenging to develop effective mitigation strategies. Integrating diverse data sources, such as satellite imagery, meteorological data, soil moisture measurements, and crop health indicators, are novel approaches to resolving this issue, which can be possible by machine learning techniques. By employing ML algorithms, it is possible to accurately predict, monitor, and forecast drought events in real-time, allowing for timely interventions and resource allocation Huang et al. (2019), Xu et al. (2020).

Detection of water stress is another crucial challenge in agricultural management, as water conservation is essential to sustain crop production. ML techniques play a decisive role in this domain by employing sensor technologies, Internet of Things (IoT) devices, and remote sensing collected data to create comprehensive models to monitor soil moisture levels, plant water uptake, and irrigation needs. ML algorithms can provide actionable strategies for optimizing irrigation scheduling, identifying water stress thresholds, and ensuring wise water usage in agricultural systems by analyzing complex datasets and using pattern recognition techniques Sahoo et al. (2020), Wang et al. (2021).

Integrating ML techniques and real agricultural data enables agricultural stakeholders to make better decisions and correct action to reduce the effects of drought and water stress Kamilaris et al. (2018), Minervini et al. (2019). Accurate judgment and monitoring of plant water stress are essential for optimizing irrigation strategiesMinervini et al. (2019), Statista (2024), Alexandratos and Bruinsma (2012), Alizadeh et al. (2018), Postel (2000), Mekonnen and Hoekstra (2016), Alizadeh et al. (2021); therefore, there is a need to find more efficient water stress monitoring methods that can predict the individual needs of each tree precisely and cost-effectively. Water stress detection techniques can be categorized into two main groups: sensor-based and model-based Virnodkar et al. (2020).

- **Sensor-based**

  Sensor-based techniques could be soil-based, which indirectly measures plant water stress, or plant-based, which directly measures plant water stress Sharma et al. (2018), Tanriverdi et al. (2016), Alizadeh et al. (2023a). The soil-based sensors are not very practical for tree crops since the tree roots usually penetrate beyond the range of their measurement and can not be very accurate. Some techniques studied the reactions of plants, including stomatal conductance, leaf water potential, relative water content, stem and fruit diameter, measuring stem water potential as one of the most accurate in-situ methods, and assessments of sap flow to detect plant water stress Turner (1988), Ihuoma and Madramootoo (2017). These techniques are dependable; however, these sensors are usually very costly and can only determine the water stress on the trees on which they are installed. Also, they are often hard and labor-intensive to install and maintain. In addition, since the decision on when to irrigate is made only based on the output from one or two sensors, a faulty sensor could cause significant errors in determining the water status of the whole block.

- **Model-based**

  Model-based techniques use weather data or remotely collected data. Remote sensing methods are non-destructive, not labor- or time-intensive, and work based on vegetation indices, such as the normalized difference water index (NDWI), optimal soil adjusted vegetation index (OSAVI), normalized difference vegetation index (NDVI) Zarco-Tejada et al. (2003), Romero et al. (2018a), Rapaport et al. (2015). Some methods applied Infrared thermometry, CWSI, and VPD Osroosh et al. (2015), Jones (2013), Cohen et al. (2005), Li et al. (2013). Also, some studies applied land surface evapotranspiration (ET) to estimate the water stress level of the crop Glenn et al. (2010), Verstraeten et al. (2008)

Low-cost drones equipped with a multiband camera to collect aerial images from the orchards are commercially available, and many growers have access to them or can get them from service providers at a reasonable cost. These drones can fly over the orchard autonomously, and they come with a software package that quickly stitches the images and produces multiple vegetation indices for the field. The information from vegetation indices and a few other weather-related data could be used to predict water stress using an AI model. This research aims to develop an AI model for

predicting SWP for pistachio and almond trees using a data-driven model that balances low data requirements with high accuracy to improve the timing for irrigation scheduling.

Machine learning and AI algorithms, such as Support Vector Machine (SVM), Random Forest (RF), and Artificial Neural Networks (ANN), are widely used in crop water stress assessmentsVirnodkar et al. (2020). A model from Landsat photos Hassan-Esfahani et al. (2015), local meteorological data and field measurements are trained using Genetic Algorithm (GA) and ML techniques that report field conditions using a soil balancing approach. The model is applied to oats and alfalfa, saving 20% water. Sun et al. (2017) designed a crop water stress detection system. Their model first detected the edge using thermal images from UAVs, then built a Gaussian mixture model for each crop species, and finally calculated the water stress index using the mean value. Many studies applied SVM to address water stress detection issues, such as Yang et al. (2011), Saini and Ghosh (2018), Kaheil et al. (2008), Warner and Nerry (2009). Random forest is another algorithm used in the domain because it prevents overfitting. Poccas et al. (2017) used NIR, WI, and D1 vegetation indices to RF and SVM in the vineyard to detect water stress levels. Moshou et al. (2014) developed a hybrid classification technique that could distinguish between different stress factors in wheat. Loggenberg et al. (2018) applied RF and XGBoost combined with hyperspectral remote sensing to model water stress in the vineyard.

Vegetation indices extracted by Romero et al. (2018b)from aerial multispectral imagery, such as the difference vegetation index, green index (GI), MSAVI, NDVI, NDGI, NDRE, OSAVI, red, green ratio index (RGRI), renormalized difference vegetation index (RDVI), and simple ratio index (SRI). Also, midday stem water potential used ANN to find the correlation between VIs and SWP. An ANN built by Poblete et al. (2017) to predict spatial variability in SWP in a drip-irrigated vineyard. This ANN was widely utilized to determine water stress, while several studies suggest that ML algorithms can provide more accurate estimates under experimental conditions Virnodkar et al. (2020).

Decision trees are among the most well-established and practical machine-learning models for every supervised learning task and have been widely employed in various applications. In the context of agriculture, decision trees have found numerous applications. They can be used to predict crop yield Patil et al. (2020), Kumar et al. (2020), Gupta et al. (2022), weather patterns Gümüşcü et al. (2020), Chauhan and Thakur (2014), Reddy et al. (2020), pest infestations Resti et al. (2022), Tageldin et al. (2020),

and crop water stress detection Huang et al. (2017), Aneley et al. (2023), Genc et al. (2013) and other smart agriculture areas, such as crop disease detection and management, guiding farmers in identifying the most effective treatments based on symptom observations and other relevant parameters. Additionally, decision trees can also be used in agricultural resource management, helping growers make strategic choices to improve overall farm productivity while minimizing environmental impact because their feasibility and ease of implementation make decision trees valuable for precision agriculture, empowering growers to make data-driven decisions for sustainable and more profitable horticultural practices.

Even though water stress detection for some crops is addressed in various studies, a simplified, practical, cost-effective, and non-destructive remote determination method of SWP that end-users can adopt in large-scale orchards for water management is missing for nut trees. A majority of techniques reported in the literature for estimating plant water status used a combination of data collected from Sap flow sensors, soil moisture sensors, and multi-spectral images. Collecting a lot of agricultural data can be challenging and prone to error Osinga et al. (2022). A stochastic decision tree can decrease the imposed computational cost to the system without losing any useful information due to common sub-sampling or feature selection methods. Moreover, uncertainties inherent in real agricultural data and the unpredictable nature of the farming environment make the prediction process very complicated United States Department of Agriculture (2023). A stochastic decision tree can handle this by incorporating randomness into the decision-making process and producing a more robust and reliable prediction. Because of their stochastic properties, decision trees are able to capture and represent complex relationships between different variables, which are often present in large orchard datasets. This enhanced modeling capability can lead to a better perception of the situation and more accurate predictions for optimizing crop yield and resource allocation.

5.2 Stochastic Decision Trees

Decision trees are useful for the decision-making process. Decision trees are trained by greedily splitting the leaf nodes into a split and two leaf nodes until a specified stopping condition is satisfied. The technique for splitting a node consists of determining the best feature and threshold that minimizes a criterion. The criterion minimization problem is solved using an exhaustive search technique. However, this

exhaustive search strategy is quite expensive, especially when the number of samples and features is large. Preprocessing the dataset by reducing the sample size or applying some feature selection techniques are popular approaches, but they might cause overfitting or loss of useful data. A Stochastic Decision Tree (SDT) is offered by Alizadeh et al. (2022a) that efficiently optimizes the splitting criterion. Suppose a dataset S contains N of samples, and D is the dimensionality. The function $f_j(x)$ at the j^{th} split node consists of a decision $f_j(x) = sign(x^p - th)$ where p represents p^{th} feature and th represents the threshold. The suggested method begins at a node with an empty set S_t and all D features. The set and the number of samples at node j_{th} are indicated by S_j and $|S_j|$, respectively. The method iteratively selects a small random collection of samples from S_j (in order of $2^{-C} \times |S_j|$) and dismisses half of the less important features related to S_t. SDT monotonically minimizes an upper bound of the splitting criterion for the best feature and threshold obtained at each iteration. Mathematically, it is demonstrated that the algorithm prioritizes more distinct features. Essentially, the more distinct a feature, the more likely it will be chosen in the final iteration. This method is explicitly and in detail described in 2.

5.3 Material And Method

The study covered two experimental sites within Merced County, California, in the heart of the San Joaquin Valley. This area has a Mediterranean climate with hot, dry summers and mild, wet winters. Specifically, the research was conducted in a 2.5-hectare pistachio orchard and a 3-hectare almond orchard equipped with irrigation systems—a double-line drip system for the pistachio orchard and a macro jet system for the almond orchard. Throughout the summers of 2022 and 2023, data collection was done from June to August, spanning 14 and 18 days, respectively, across the two years. This intensive data collection involved monitoring stem water potential and leaf temperature and capturing aerial multispectral images from 18 pistachio and 17 almond trees strategically selected to reflect the inherent variability within the orchards. With this strategy, we could comprehensively analyze big data and collect environmental and physiological dynamics to deeply understand the critical information about each orchard's health and productivity.

A PMS-615 pressure chamber device (PMS Instrument Company, Albany, OR, USA) measures SWP as the ground truth value at around 1 PM. To reduce the non-conformity in SWP readings across individual trees, three leaves from each sample

tree's lower shaded canopy were chosen for SWP measurements. The mean value of the observed SWP value per tree is calculated for further study. To ensure accurate results, encasing the leaves in bags is critical before doing the SWP measurements. The leaves were packed in aluminum bags for at least 15 minutes before being removed from the tree and tested in the pressure chamber. This method steers the leaf toward equilibrium and mitigates disparities caused by continual photosynthesis and transpiration within the leaves.

The leaf temperature was measured from three independent leaves on each sample tree, while SWP data was collected using an infrared thermometer (model: K-type). We also used weather information such as air temperature, minimum relative humidity, and minimum air pressure from the local weather station installed in the orchards. The radiative flux, ambient air temperature, wind speed, and atmospheric moisture content are just a few of the ambient factors that impact the temperature of the plant canopy, which acts as a proxy for foliar hydration levels. Normalization of the canopy temperature is essential to ensure robustness across diverse environmental conditions. Such normalization necessitates benchmarking against a reference obtained from thermal imaging data. Within this context, the Crop Water Stress Index (CWSI) is instrumental, as it establishes two critical thermal thresholds: T_{dry} and T_{wet}, which correspond to the thermal readings of a leaf under conditions of zero transpiration and maximal transpiration, respectively Zhang et al. (2019a), Ben-Gal et al. (2009). CWSI is calculated as follows:

$$CWSI = \frac{(T_{canopy} - T_{wet})}{(T_{dry} - T_{wet})} \tag{5.1}$$

where T_{canopy} represents the temperature of the canopy, and the Crop Water Stress Index (CWSI) quantifies the hydration status of the plants on a scale from 0 to 1, where 0 corresponds to an adequately watered state and 1 denotes a condition of significant water stress. The parameters T_{dry} and T_{wet}, delineating the upper and lower limits of the index, can be determined by empirical methods or derived from theoretical models. Determining wet-bulb temperature necessitates calculations based on the dry-bulb temperature and relative humidity, which are measured via a calibrated thermometer and an electronic hygrometer. Stull Stull (2011) introduced an empirically derived equation utilizing gene-expression programming to estimate this temperature. The formulation, known as the Stull equation, is expressed as follows:

$$
\begin{aligned}
WBT = T_{\text{air}} \cdot \arctan(0.151977 \cdot \sqrt{RH + 8.313659}) + \\
\arctan(T_{\text{air}} + RH) - \arctan(RH - 1.676331) + \\
0.00391838 \cdot RH^{1.5} \cdot \arctan(0.023101 \cdot RH) \\
- 4.686035;
\end{aligned}
\tag{5.2}
$$

In addition, we calculate the vapor pressure deficit (VPD) as an important factor in determining plants' water needs Elbeltagi et al. (2023), Shekoofa et al. (2016), Yin et al. (2021) and add it to our data set. VPD plays a crucial role in plant physiology by influencing the transpiration rate. Higher VPD values indicate drier air, meaning greater transpiration rates and, consequently, higher demand for water in plants. This increase in transpiration can lead to a significant decrease in SWP as plants lose more water to the atmosphere. By incorporating VPD into our predictive models, we aim to increase the accuracy of SWP predictions, providing more precise irrigation strategies that optimize water usage while minimizing stress on the plants. VPD is the difference between the water vapor pressure in saturated air at a certain temperature and the water vapor pressure in the air at the same temperature Rawson et al. (1977). VPD is determined from the following equation Yin et al. (2021), Grossiord et al. (2020):

$$
\begin{aligned}
\text{VPD} &= e_s - e_a \\
&= 0.6107 \times 10^{(7.5T_{Leaf}/(273.3+T_{Leaf}))} \\
&- \text{RH} \times \Big((0.6107 \times 10^{(7.5T_{air}/(273.3+T_{air}))})/100\Big)
\end{aligned}
\tag{5.3}
$$

where e_s represents the saturated vapor pressure in the stomatal cavity at leaf temperature, e_a is the water vapor pressure of air at ambient temperature, and RH is the relative humidity (%).

Vegetation indices are extracted using aerial images captured using a DJI P4 multispectral agricultural drone with an RGB camera and a five-band multispectral camera. These spectral data were analyzed using software (DJI Terra) to build orthomosaic maps for each orchard. The DJI Terra was used to create a color composite as well as five orthomosaic maps indexed for specific vegetation parameters: NDVI, GNDVI (Green Normalized Difference Vegetation Index), OSAVI (Optimized Soil-Adjusted Vegetation Index), LCI (Leaf Chlorophyll Index), and NDRE (Normalized

Difference Red Edge). The vegetation indices formula can be found in Bannari et al. (1995).

5.4 SDT setup

Stem Water Potential (SWP) is a good way to measure how much water a tree has. The University of California Agricultural Extension (UCANR) has established clear categories for different types of trees Orchards (2024), Agriculture and Resources (2024). Specifically, almond trees were classified from minimal stress in the -6 to -10 bars, -10 to -14 bars as mild to moderate stress, -14 to -18 moderate to high stress, and below -18 was considered as high to severe stress. At the same time, pistachio trees range from non-stressed at -9 to -11 bars, mild to moderately stressed from -11 to -15, and severely stressed below -15 bars. This study examines SWP data for almond and pistachio trees separately and in a combined dataset to facilitate a comprehensive analysis. Cutoff points of -10 bars for binary classification and -10 and -15 bars for a three-class scheme were selected for equitable analysis. Farmers require knowledge of water stress levels rather than precise SWP values in practical agricultural settings. Accordingly, this study approaches the issue as a classification problem, aiming to predict the categorical level of SWP rather than its exact numerical value, which would typically be addressed using a regression model. Due to the small size of the dataset, the SDT hyper-parameters are set as follows:

- In each iteration, we would let the algorithm randomly select and remove 50% of samples from S_{j^c} and add them to set S_{j^t}.
- We considered making decisions based on 0.3 of features in each iteration.
- The dataset was randomly split into 80% trainset and 20% test set for different setups and feature combinations. To compare findings fairly, each scenario was created by removing the desired features from the primary dataset and using a unique training and test set.

This research aims to achieve two different goals. The first was to study the correlation between different factors and the water stress level in the canopy and consequently choose the scenario with the highest accuracy in predicting SWP. Therefore, we studied different scenarios based on different combinations of data collected in the SDT model. The reason for not using a feature selection method was to prevent losing

any information that might increase the SWP prediction's accuracy. Additionally, the structure of SDT incorporates a hidden feature selection into each iteration of tree formation, allowing it to make more precise predictions by utilizing every aspect of the dataset's information. The second goal was to train a general, robust model to predict SWP for almond and pistachio orchards. Different combinations of predictors were studied that include ***Leaf temperature (T_l), Air temperature (T_a), CWSI, NDVI, GNDVI, OSAVI, LCI, NDRE, Relative humidity(RH), Air pressure (P), VPD, Water Stress index which is leaf-air temperature difference (WSI) Zhang et al. (2019b)***. Table 5.1 shows detailed information about all scenarios and dataset sizes. These scenarios were used to solve binary classification problems and three-class classification problems.

Table 5.1: Dataset scenarios

Index	**Dataset size**	**Scenario**
SC1	490 x 13	T_l, RH, P, T_a, WSI, VPD, CWSI, NDVI, GNDVI, OSAVI, LCI, NDRE, SWP
SC2	490 x 8	T_l, RH, P, T_a, WSI, VPD, CWSI, SWP
SC3	490 x 9	WSI, VPD, CWSI, NDVI, GNDVI, OSAVI, LCI, NDRE, SWP
SC4	490 x 4	WSI, VPD, CWSI, SWP
SC5	490 x 5	T_l, WSI, VPD, CWSI, SWP
SC6	490 x 7	T_l, RH, P, T_a, WSI, VPD, SWP
SC7	490 x 12	T_l, RH, P, T_a, VPD, CWSI, NDVI, GNDVI, OSAVI, LCI, NDRE, SWP

5.5 Experimental Results

To identify the key factors influencing SWP prediction, we analyzed seven distinct scenarios as detailed in Section 5.4 for both the binary classification problem approach and the three-class classification problem approach. For consistency across different tests, we limited the depth of the SDT between 2 and 10 for all scenarios. As shown in Fig. 5.1, Scenario SC1 achieves the highest accuracy in the training and testing

phases, underlining its robustness in SWP prediction. The variables involved in SC1 offer a holistic view of environmental and plant physiological parameters, contributing significantly to the model's predictive strength. Similarly, SC7, with a slightly reduced but still extensive variable set, also shows high accuracy, affirming the importance of a broad feature set for effective SWP prediction.

In contrast, scenarios with fewer variables, such as SC4 and SC5, demonstrated lower test accuracies. This reduction suggests that while these scenarios are streamlined, they lack sufficient information to capture all the shades of SWP dynamics effectively. Neglecting critical variables like NDVI and GNDVI, which are key indicators of vegetation health, could explain the diminished predictive capabilities in these simpler scenarios.

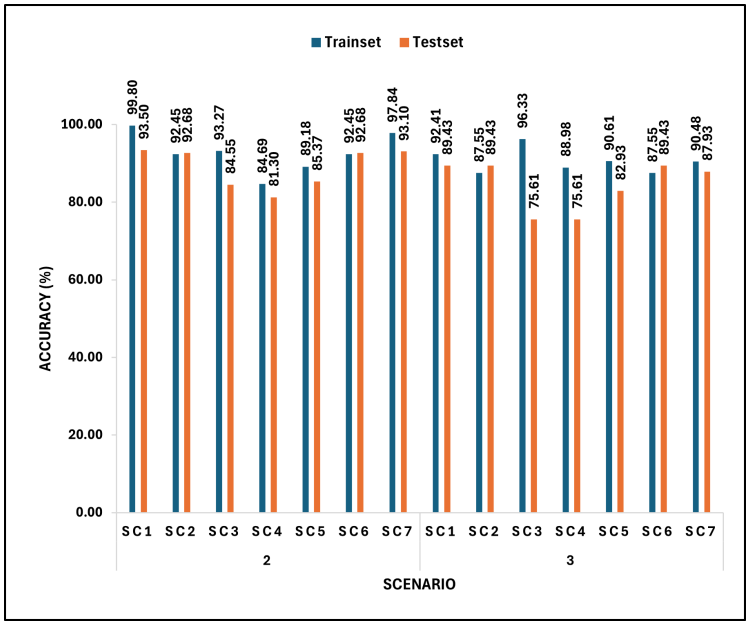


Figure 5.1: Prediction accuracy for different scenarios in Trainset and Testset

To validate the quality of our SDT approach, we conducted a rigorous comparative analysis against three widely recognized machine learning algorithms: SVM employing a linear kernel, KNN utilizing a Euclidean distance metric, and RF configured with 30 trees. Each algorithm was evaluated using a robust 10-fold cross-validation scheme within the training dataset. As illustrated in Figure 5.2, the performance of these models was assessed in two different classification contexts: binary and three-class settings.

In the binary classification scenario, the SDT model demonstrated superior test accuracy, achieving 93.50%, which notably exceeds the performance of SVM at 92.70%,

KNN at 91.90%, and RF at 92.70%. Similarly, in the more complex three-class classification setting, SDT maintained its leading position with an accuracy of 89.43%, compared to 89.40% for SVM, 86.20% for KNN, and 87.80% for RF. This consistent outperformance of SDT in diverse settings highlights its robustness and exceptional ability to generalize from the training to the testing phases. The model's efficacy in handling intricate data structures makes it a preferred option in scenarios where traditional models might falter, underlining its potential as a robust tool in predictive analytics for agricultural applications.

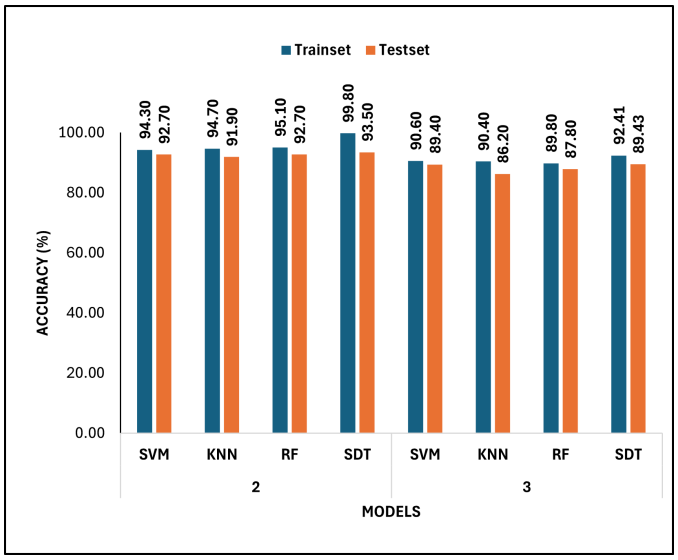


Figure 5.2: Train & Test set accuracy for all scenarios on SVM, KNN, RF, & SDT

Table 5.2: Comparison of ML Models on Precision, Recall & F1 Score on SC1

Model	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
SVM	90.27	90.61	90.26
KNN	90.08	90.41	90.19
RF	89.50	89.80	89.60
SDT(our)	92.05	92.04	92.03

Table 5.2 compares four machine learning models in terms of Precision, Recall, and F1 Score. The SDT model consistently exhibits superior performance across all evaluated metrics, achieving the highest scores with 92.05% in Precision, 92.04% in Recall,

and 92.03% in F1 Score. These results highlight the robustness and effectiveness of SDT in attaining a balanced performance between identifying and retrieving relevant instances. In comparison, the other models— SVM, KNN, and RF —show competitive yet slightly lower metrics. The Random Forest model, in particular, showed a notable drop in all three metrics, indicating a potential trade-off in its precision-recall balance. This comparative analysis highlights the potential of SDT for applications that demand high accuracy and reliability in complex classification tasks.

5.6 Chapter Summary

In this study, we presented a robust stochastic decision tree (SDT) model aimed at enhancing the Precision of water stress estimation in pistachio and almond orchards, thereby aiding in the efficient scheduling of irrigation. Our SDT model reduces the laborious efforts associated with traditional SWP measurements and circumvents the high costs involved with sensor installation and aerial imaging techniques. Through rigorous evaluation, the SDT model demonstrated a notable prediction accuracy of nearly 94%, significantly outperforming established models, such as SVM, KNN, and RF. This superior performance was attributed to the model's ability to handle complex datasets without the necessity for feature selection, thereby preserving the integrity of the data and reducing computational expenses. Notably, the SDT model's stochastic nature contributes to its robustness, making it less susceptible to overfitting and more adaptable to varying data conditions encountered in agricultural applications. In future work, we aim to expand the applicability of our model beyond pistachio and almond orchards to include a variety of crops, thus broadening the scope of our research to encompass a more diverse range of agricultural needs. Further development will focus on integrating real-time data feeds to enhance the model's predictive capabilities and deploying the model in a cloud-based framework to facilitate end-user accessibility. By continuing to refine and adapt our approach, we seek to contribute substantially to precision agriculture, particularly in the domains of water management and drought mitigation.

Chapter 6

Concluding Remarks and Directions for Future Research

6.1 Summary of Contributions

In this dissertation, we have examined various stochastic approaches in machine learning applied to decision trees, algorithms based on nearest neighbors, and agricultural systems. In this chapter, we present a summary of the contributions and directions for future research.

In Chapter 2, we introduced a novel Stochastic Decision Tree (SDT) induction method designed to minimize computational complexity while maintaining high predictive accuracy. A key innovation of this approach is the integration of Haar Trees, leveraging Haar-like features commonly used in image processing to enhance decision-making at each node. By incorporating stochastic processes into the induction of decision trees, the model efficiently optimizes the splitting criterion without relying on traditional exhaustive search methods. The Haar Tree's use of integral images further reduces the computational burden, allowing for rapid computation of features at each node. Our experiments on the MNIST dataset demonstrated that the combination of SDT with Haar Trees achieved a test accuracy of 94%, significantly outperforming traditional axis-aligned decision trees and providing competitive performance compared to more complex oblique trees. This balance between accuracy and efficiency, made possible by the unique combination of stochastic tree induction and Haar-like features, is a key contribution of this chapter, demonstrating the SDT-Haar Tree model's suitability for high-dimensional datasets and resource-constrained environments.

In Chapter 3, we presented a Neural-Synthetic Reduced Nearest-Neighbor

(Neural-SRNN) model, addressing challenges inherent in nearest-neighbor-based methods, such as the curse of dimensionality. This model integrates neural networks into the SRNN framework to enhance interpretability and performance in high-dimensional classification tasks. By leveraging stochastic optimization techniques and introducing a focal loss function, Neural-SRNN handles class imbalance effectively, providing a more robust classification framework. Our experimental results, particularly on datasets such as MNIST and Fashion-MNIST, showed that the Neural-SRNN model outperformed traditional nearest-neighbor algorithms, especially in high-dimensional spaces. The model's use of expectation-maximization allowed for more accurate representation of sub-clusters, improving its classification accuracy and generalization. This method presents a significant advance over traditional nearest-neighbor techniques, demonstrating its potential for scalable, interpretable machine learning in challenging classification tasks.

In Chapter 4, we extended the Neural-SRNN framework by developing a Two-Layer Neural Network architecture aimed at enhancing both classification accuracy and computational efficiency in image classification tasks. The model's architecture includes Mini-Convolutional Neural Networks (Mini-CNNs) in the first layer for class-specific feature extraction, followed by a shallow neural network for final classification. This modular design allows for parallel processing, significantly reducing the computational burden while maintaining high accuracy, particularly on challenging datasets like SignMNIST and FashionMNIST. Additionally, the two-layer approach helps prevent overfitting and demonstrates superior generalization capabilities. Extensive experimental validation confirmed the superiority of this approach over existing models, showing that it delivers a resource-efficient solution for large-scale image classification problems.

In Chapter 5, we applied the SDT model to a real-world agricultural problem, focusing on predictive analysis of stem water potential (SWP) in almond and pistachio orchards. This chapter's primary contribution lies in the development of an efficient irrigation prediction model that integrates both ground and aerial sensor data to optimize water usage. By leveraging vegetative indices and weather data, the SDT model accurately predicts SWP, a crucial indicator of water stress in trees. Our results demonstrated that the SDT model outperformed traditional machine learning methods, such as Random Forests and k-nearest neighbor, in terms of prediction accuracy. Furthermore, the SDT model contributed to sustainable agricultural practices by enabling precise water management at the individual tree level, thereby conserving

water resources. This chapter underscores the versatility of the SDT model and highlights its practical applicability in real-world precision agriculture scenarios, where efficient water management is critical.

6.2 Directions for future research

This dissertation opens several avenues for future research, particularly in extending stochastic decision trees and nearest-neighbor methods. One promising direction is the application of stochastic decision trees to regression tasks, which would support continuous variable predictions critical in agriculture, such as yield forecasting or water demand estimation. Additionally, enhancing the Neural-SRNN framework for multi-modal data integration—such as combining image, sensor, and environmental data—could improve model robustness in complex scenarios. Finally, integrating methods to clarify and interpret model decisions could enhance transparency and accountability, fostering greater trust in model predictions and allowing users to verify outcomes, identify potential biases, and understand the influence of key features on predictions. This would be particularly valuable in precision agriculture, where practical and reliable insights into decision-making processes are essential for resource-intensive applications.

Bibliography

Abdufattokhov, A. A., Yusupov, K. A., and Boyakhmedov, F. I. (2019). Stochastic gaussian process in big data processing. *International Journal of Open Information Technologies*, 7(12):40–44.

Agriculture, U. and Resources, N. (2024). Pistachio short course: Irrigation management. Accessed: 2024-09-01.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Al-Ghobari, H. and Mohammad, F. S. (2011). Development and evaluation of a decision support system for irrigation management in arid regions. *Journal of Irrigation and Drainage Engineering*, 137(10):705–714.

Alaghi, A. and Hayes, J. P. (2013). Survey of stochastic computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2s):1–19.

Alexandratos, N. and Bruinsma, J. (2012). *World agriculture towards 2030/2050: the 2012 revision*. Food and Agriculture Organization of the United Nations (FAO).

Alizadeh, A., Farajijalal, M., Rezvani, Z., Toudeshki, A., and Ehsani, R. (2023a). Developing a data-driven model for predicting water stress in pistachio trees. In *International Congress on Agricultural Mechanization and Energy in Agriculture*, pages 186–196, Cham, Switzerland. Springer Nature.

Alizadeh, A., Farajijalal, M., Rezvani, Z., Toudeshki, A., and Ehsani, R. (2023b). Developing a data-driven model for predicting water stress in pistachio trees. In *International Congress on Agricultural Mechanization and Energy in Agriculture*, pages 186–196. Springer.

Alizadeh, A., Singhal, M., Behzadan, V., Tavallali, P., and Ranganath, A. (2022a). Stochastic induction of decision trees with application to learning haar trees. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 825–830. IEEE.

Alizadeh, A., Tavallali, P., Behzadan, V., Ranganath, A., and Singhal, M. (2022b). A novel approach for synthetic reduced nearest-neighbor leveraging neural networks. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 831–836. IEEE.

Alizadeh, A., Toudeshki, A., Ehsani, R., and Migliaccio, K. (2018). Potential sources of errors in estimating plant sap flow using commercial thermal dissipation probes. *Applied Engineering in Agriculture*, 34(6):899–906.

Alizadeh, A., Toudeshki, A., Ehsani, R., Migliaccio, K., and Wang, D. (2021). Detecting tree water stress using a trunk relative water content measurement sensor. *Smart Agricultural Technology*, 1:100003.

Allen, K., Shelhamer, E., Shin, H., and Tenenbaum, J. (2019). Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pages 232–241. PMLR.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Aneley, G. M., Haas, M., and Köhl, K. (2023). Lidar-based phenotyping for drought response and drought tolerance in potato. *Potato Research*, 66(4):1225–1256.

Aqeel-ur Rehman, A. K., Zaman, A., and Shaikh, A. (2009). A decision support system for agriculture using wireless sensor networks. *Journal of Network and Computer Applications*, 32(5):1204–1216.

Araya, A., Hoogenboom, G., Gowda, P. H., Paz, J. O., and Fraisse, C. W. (2016). Assessment of irrigation water productivity using a decision support system in irrigated agriculture. *Agricultural Water Management*, 178:133–145.

Banerjee, G., Sarkar, U., Das, S., and Ghosh, I. (2018). Artificial intelligence in agriculture: A literature survey. *International Journal of Scientific Research in Computer Science Applications and Management Studies*, 7(3):1–6.

Bannari, A., Morin, D., Bonn, F., and Huete, A. (1995). A review of vegetation indices. *Remote Sensing Reviews*, 13(1–2):95–120.

Bellman, R. (1966). Dynamic programming. *science*, 153(3731):34–37.

Ben-Gal, A., Agam, N., Alchanatis, V., Cohen, Y., Yermiyahu, U., Zipori, I., Presnov, E., Sprintsin, M., and Dag, A. (2009). Evaluating water stress in irrigated olives: correlation of soil water status, tree water status, and thermal imagery. *Irrigation Science*, 27:367–376.

Bennett, K. P. and Mangasarian, O. L. (1994). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34.

Benoît, F., Van Heeswijk, M., Miche, Y., Verleysen, M., and Lendasse, A. (2013). Feature selection for nonlinear models with extreme learning machines. *Neurocomputing*, 102:111–124.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, volume 18, pages 509–517.

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104.

Bhattacharyya, D. K. and Kalita, J. K. (2013). *Network anomaly detection: A machine learning perspective*. Crc Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Bottou, L. (2010a). Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010*, pages 177–186.

Bottou, L. (2010b). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT*, pages 177–186. Springer.

Bradley, P. S. and Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group.

Carrasco-Benavides, M., Viejo, C. G., Tongson, E., Baffico-Hernández, A., Ávila-Sánchez, C., Mora, M., and Fuentes, S. (2022). Water status estimation of cherry trees using infrared thermal imagery coupled with supervised machine learning modeling. *Computers and Electronics in Agriculture*, 200:107256.

Carreira-Perpinán, M. A. and Tavallali, P. (2018). Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in neural information processing systems*, 31.

Chauhan, D. and Thakur, J. (2014). Data mining techniques for weather prediction: A review. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2(8):2184–2189.

Chen, Q., Georganas, N. D., and Petriu, E. M. (2007). Real-time vision-based hand gesture recognition using haar-like features. In *2007 IEEE instrumentation & measurement technology conference IMTC 2007*, pages 1–6. IEEE.

Chen, R.-C., Dewi, C., Huang, S.-W., and Caraka, R. (2020). Selecting critical features for data classification based on machine learning methods. *Journal of Big Data*, 7(1):1–20.

Choi, J. (2012). Realtime on-road vehicle detection with optical flows and haar-like feature detectors. *Department of Computer Science University of Illinois at Urbana-Champaign*.

Cohen, Y., Alchanatis, V., Meron, M., Saranga, Y., and Tsipris, J. (2005). Estimation of leaf water potential by thermal imagery and spatial analysis. *Journal of Experimental Botany*, 56(417):1843–1852.

Cortizo, J. C. and Giraldez, I. (2006). Multi criteria wrapper improvements to naive bayes learning. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 419–427. Springer.

Cover, T. and Hart, P. (1967a). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Cover, T. and Hart, P. (1967b). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Criminisi, A. and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *Icml*, volume 1, pages 74–81.

Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent data analysis*, 1(3):131–156.

De Berg, M. (2000). *Computational geometry: algorithms and applications*. Springer Science & Business Media.

Demirkir, C. and Sankur, B. (2004). Face detection using boosted tree classifier stages. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, 2004.*, pages 575–578. IEEE.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142.

Devroye, L., Györfi, L., and Lugosi, G. (2013). *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media.

Ding, W., Taylor, G., and Jin, Y. (2014). Combining cnn with svm for object recognition. *Neurocomputing*, 150:174–183.

D'Odorico, P., Chiarelli, D. D., Rosa, L., Bini, A., Zilberman, D., and Rulli, M. C. (2020). The global value of water in agriculture. *Proceedings of the National Academy of Sciences*, 117(36):21985–21993.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327.

Elbeltagi, A., Srivastava, A., Deng, J., Li, Z., Raza, A., Khadke, L., Yu, Z., and El-Rawy, M. (2023). Forecasting vapor pressure deficit for agricultural water management using machine learning in semi-arid environments. *Agricultural Water Management*, 283:108302.

Eli-Chukwu, N. C. (2019). Applications of artificial intelligence in agriculture: A review. *Engineering, Technology & Applied Science Research*, 9(4).

Fan, J., Jiang, L., and Zhang, W. (2019). Decision tree model for irrigation scheduling and optimization based on remote sensing data. *Computers and Electronics in Agriculture*, 161:148–160.

Fan, J., Yao, Q., and Tong, H. (2003). Asymptotic theory for stochastic decision trees. *Journal of the American Statistical Association*, 98(462):545–556.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

Gates, G. W. (1972). The reduced nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theory*, 18:431–433.

Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W.-Y. (1999). Boat—optimistic decision tree construction. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of Data*, pages 169–180.

Genc, L., Inalpulat, M., Kizil, U., Mirik, M., Smith, S. E., and Mendes, M. (2013). Determination of water stress with spectral reflectance on sweet corn (zea mays l.) using classification tree (ct) analysis. *Zemdirbyste-Agriculture*, 100(1):81–90.

Glenn, E. P., Nagler, P. L., and Huete, A. R. (2010). Vegetation index methods for estimating evapotranspiration by remote sensing. *Surveys in Geophysics*, 31(6):531–555.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Gouk, H., Frank, E., and Pfahringer, B. (2019). Stochastic gradient trees. *Journal of Machine Learning Research*, 20(57):1–25.

Grossiord, C., Buckley, T. N., Cernusak, L. A., Novick, K. A., Poulter, B., Siegwolf, R. T., Sperry, J. S., and McDowell, N. G. (2020). Plant responses to rising vapor pressure deficit. *New Phytologist*, 226(6):1550–1566.

Gümüşcü, A., Tenekeci, M. E., and Bilgili, A. V. (2020). Estimation of wheat planting date using machine learning algorithms based on available climate data. *Sustainable Computing: Informatics and Systems*, 28:100308.

Gupta, M., Krishna, B. S., Kavyashree, B., Narapureddy, H. R., Surapaneni, N., and Varma, K. (2022). Crop yield prediction techniques using machine learning algorithms. In *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, pages 1–7. IEEE.

Gupta, M. R., Chen, Y., et al. (2011). Theory and use of the em algorithm. *Foundations and Trends® in Signal Processing*, 4(3):223–296.

Gutiérrez-Gordillo, S., d. l. G. González-Santiago, J., Trigo-Córdoba, E., Rubio-Casal, A. E., García-Tejero, I. F., and Egea, G. (2021). Monitoring of emerging water stress situations by thermal and vegetation indices in different almond cultivars. *Agronomy*, 11(7).

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Hand, D. J. and Vinciotti, V. (2001). Discriminant analysis when the classes appear asymmetric: the impact of distributional assumptions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):315–330.

Hassan-Esfahani, L., Torres-Rua, A., and McKee, M. (2015). Assessment of optimal irrigation water allocation for pressurized irrigation system using water balance approach, learning machines, and remotely sensed data. *Agricultural Water Management*, 153:42–50.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hazen, G. (1992). Stochastic trees: A new technique for temporal medical decision modeling. *Medical Decision Making*, 12(3):163–178.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heath, D., Kasif, S., and Salzberg, S. (1993). Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007.

Hespos, R. and Strassmann, F. (1965). Stochastic decision trees for the analysis of investment decisions. *Management Science*, 11(10):244–259.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hoare, C. A. R. (1961). Algorithm 64: quicksort. *Communications of the ACM*, 4(7):321.

Huang, G., Liu, Z., Maaten, L. V. D., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, J., Zhou, W., Li, H., and Li, W. (2015). Sign language recognition using 3d convolutional neural networks. In *2015 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE.

Huang, Q., Han, W., Wei, W., and Xu, Z. (2019). A deep learning-based method for estimating crop water stress in precision agriculture using multisource data. *IEEE Access*, 7:118242–118254.

Ihuoma, S. O. and Madramootoo, C. A. (2017). Recent advances in crop water stress detection. *Computers and Electronics in Agriculture*, 141:267–275.

Iyengar, V. S., Apte, C., and Zhang, T. (2000). Active learning using adaptive resampling. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 91–98.

Jha, K., Doshi, A., Patel, P., and Shah, M. (2019). A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, 2:1–12.

John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer.

Jones, H. G. (2013). *Plants and microclimate: A quantitative approach to environmental plant physiology*. Cambridge University Press, Cambridge, United Kingdom.

Jones, M. and Viola, P. (2003). Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3(14):2.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee.

Kaheil, Y. H., Rosero, E., Gill, M. K., McKee, M., and Bastidas, L. A. (2008). Downscaling and forecasting of evapotranspiration using a synthetic model of wavelets and support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 46(9):2692–2707.

Kamilaris, A., Kartakoullis, A., and Prenafeta-Boldú, F. X. (2018). A review on the practice of big data analysis in agriculture. *Computers and Electronics in Agriculture*, 143:23–37.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *Proceedings of ICLR*.

Kohavi, R. and Sommerfield, D. (1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *KDD*, pages 192–197.

Koller, D. and Sahami, M. (1996). Toward optimal feature selection. Technical report, Stanford InfoLab.

Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1475.

Korn, F., Pagel, B.-U., and Faloutsos, C. (2001). On the" dimensionality curse" and the" self-similarity blessing". *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kumar, Y. J. N., Spandana, V., Vaishnavi, V. S., Neha, K., and Devi, V. G. R. R. (2020). Supervised machine learning approach for crop yield prediction in agriculture sector. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 736–741. IEEE.

Kwak, J.-H., Woen, I.-Y., and Lee, C.-H. (2013). Learning algorithm for multiple distribution data using haar-like feature and decision tree. *KIPS Transactions on Software and Data Engineering*, 2(1):43–48.

Larios, N., Soran, B., Shapiro, L. G., Martínez-Muñoz, G., Lin, J., and Dietterich, T. G. (2010). Haar random forest features and svm spatial matching kernel for stonefly species identification. In *2010 20th International Conference on Pattern Recognition*, pages 2624–2627. IEEE.

Laurent, H. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, Z. L., Tang, B. H., Wu, H., Ren, H., Yan, G., Wan, Z., Trigo, I. F., and Sobrino, J. A. (2013). Satellite-derived land surface temperature: Current status and perspectives. *Remote Sensing of Environment*, 131:14–37.

Liao, Z. and Wang, J. (2010). Forecasting models for financial markets: A knn-based approach. *International Journal of Financial Research*, 1(2):48–53.

Lienhart, R. and Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, volume 1, pages I–I. IEEE.

Liu, B. et al. (2020). Survey of stochastic computing for neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):715–735.

Liu, C., Jiang, D., and Yang, W. (2014). Global geometric similarity scheme for feature selection in fault diagnosis. *Expert Systems with Applications*, 41(8):3585–3595.

Liu, L. and Wong, W. H. (2014). *Multivariate density estimation based on adaptive partitioning: Convergence rate, variable selection and spatial adaptation*. Department of Statistics, Stanford University.

Lloyd, S. (1982a). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Lloyd, S. (1982b). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Loggenberg, K., Strever, A., Greyling, B., and Poona, N. (2018). Modelling water stress in a shiraz vineyard using hyperspectral imaging and machine learning. *Remote Sensing*, 10(2):202.

Loh, P.-L. and Nowozin, S. (2013). Faster hoeffding racing: Bernstein races via jackknife estimates. In *International Conference on Algorithmic Learning Theory*, pages 203–217. Springer.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

Maldonado, S., Weber, R., and Famili, F. (2014). Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information sciences*, 286:228–246.

Mathy, C., Derbinsky, N., Bento, J., Rosenthal, J., and Yedidia, J. (2015). The boundary forest algorithm for online supervised and unsupervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

McLachlan, G. J. and Krishnan, T. (1997). *The EM algorithm and extensions*. John Wiley & Sons.

Medellín-Azuara, J., Escriva-Bou, A., Rodríguez-Flores, J. M., Cole, S. A., Abatzoglou, J., Viers, J. H., Santos, N., Summer, D. A., Medina, C., Arévalo, R., et al. (2022). Economic impacts of the 2020–22 drought on california agriculture. *UC Merced*.

Mekonnen, M. M. and Hoekstra, A. Y. (2016). Four billion people facing severe water scarcity. *Science Advances*, 2(2):e1500323.

Minervini, M., Abdelsamea, M. M., and Tsaftaris, S. A. (2019). Image-based plant phenotyping with deep learning: A review. *Trends in Plant Science*, 24(11):961–975.

Mita, T., Kaneko, T., and Hori, O. (2005). Joint haar-like features for face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1619–1626. IEEE.

Mitchell, T. M. and Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.

Moghimi, M. M., Nayeri, M., Pourahmadi, M., and Moghimi, M. K. (2018). Moving vehicle detection using adaboost and haar-like feature in surveillance videos. *arXiv preprint arXiv:1801.01698*.

Moh'd A Mesleh, A. (2007). Chi square feature extraction based svms arabic language text categorization system. *Journal of Computer Science*, 3(6):430–435.

Moshou, D., Pantazi, X. E., Kateris, D., and Gravalos, I. (2014). Water stress detection based on optical multisensor fusion with a least squares support vector machine classifier. *Biosystems Engineering*, 117:15–22.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Naik, S. and Patel, V. (2021). Stochastic approach for remote sensing data collection. *Remote Sensing Applications: Society and Environment*, 22:100540.

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer.

Nguyen, L. M., Liu, J., Scheinberg, K., and Tran, D. T. (2017). Sarah: A stochastic recursive gradient algorithm. *SIAM Journal on Optimization*, 27(2):748–771.

Niranjani, V. and Selvam, N. S. (2020). Overview on deep neural networks: Architecture, application and rising analysis trends. *Business Intelligence for Enterprise Internet of Things*, pages 271–278.

Ohana-Levi, N., Zachs, I., Hagag, N., Shemesh, L., and Netzer, Y. (2022). Grapevine stem water potential estimation based on sensor fusion. *Computers and Electronics in Agriculture*, 198:107016.

Omohundro, S. M. (1989). Efficient algorithms for multidimensional nearest neighbor searching. *International Journal of Supercomputer Applications*, 3(4):99–112.

Orchards, S. V. (2024). Advanced swp interpretation in almond. Accessed: 2024-09-01.

Osinga, S. A., Paudel, D., Mouzakitis, S. A., and Athanasiadis, I. N. (2022). Big data in agriculture: Between opportunity and solution. *Agricultural Systems*, 195:103298.

Osroosh, Y., Peters, R. T., Campbell, C. S., and Zhang, Q. (2015). Automatic irrigation scheduling of apple trees using theoretical crop water stress index with an innovative dynamic threshold. *Computers and Electronics in Agriculture*, 118:193–203.

Park, K.-Y. and Hwang, S.-Y. (2014). An improved haar-like feature for efficient object detection. *Pattern Recognition Letters*, 42:148–153.

Parmar, A., Katariya, R., and Patel, V. (2018). A review on random forest: An ensemble classifier. In *International Conference on Intelligent Data Communication Technologies and Internet of Things*, pages 758–763. Springer.

Patil, P., Panpatil, V., and Kokate, S. (2020). Crop prediction system using machine learning algorithms. *International Research Journal of Engineering and Technology (IRJET)*, 7(2).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pereira, L., Allen, R., and Smith, M. (2002). A decision support system for irrigation scheduling based on soil moisture forecasting. *Agricultural Water Management*, 51(3):173–195.

Pham, M.-T. and Cham, T.-J. (2007). Fast training and selection of haar features using statistics in boosting-based face detection. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE.

Poblete, T., Ortega-Farias, S., Moreno, M. A., and Bardeen, M. (2017). Artificial neural network to predict vine water status spatial variability using multispectral information obtained from an unmanned aerial vehicle (uav). *Sensors*, 17(11):2488.

Poccas, I., Gonçalves, J., Costa, P. M., Gonçalves, I., Pereira, L. S., and Cunha, M. (2017). Hyperspectral-based predictive modelling of grapevine water status in the portuguese douro wine region. *International Journal of Applied Earth Observation and Geoinformation*, 58:177–190.

Postel, S. L. (2000). Entering an era of water scarcity: the challenges ahead. *Ecological Applications*, 10(4):941–948.

Quinlan, J. R. (1986a). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (1986b). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (1993). C4. 5: Programs for machine learning. *Machine Learning*, 16(3).

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ram, P. and Gray, A. G. (2011). Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–635. ACM.

Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference. In *Artificial Intelligence and Statistics (AISTATS)*.

Rapaport, T., Hochberg, U., Shoshany, M., Karnieli, A., and Rachmilevitch, S. (2015). Combining leaf physiology, hyperspectral imaging and partial least squares-regression (pls-r) for grapevine water status assessment. *ISPRS Journal of Photogrammetry and Remote Sensing*, 109:88–97.

Rawson, H. M., Begg, J. E., and Woodward, R. G. (1977). The effect of atmospheric humidity on photosynthesis, transpiration and water use efficiency of leaves of several plant species. *Planta*, 134:5–10.

Reddy, K. S. P., Roopa, Y. M., N., K. R. L., and Nandan, N. S. (2020). Iot based smart agriculture using machine learning. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 130–134. IEEE.

Resti, Y., Irsan, C., Amini, M., Yani, I., and Passarella, R. (2022). Performance improvement of decision tree model using fuzzy membership function for classification of corn plant diseases and pests. *Science and Technology Indonesia*, 7(3):284–290.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Rodriguez-Lujan, I., Elkan, C., Santa Cruz, C., and Huerta, R. (2010). Quadratic programming feature selection. *Journal of Machine Learning Research*.

Romero, M., Luo, Y., Su, B., and Fuentes, S. (2018a). Vineyard water status estimation using multispectral imagery from an uav platform and machine learning algorithms for irrigation scheduling management. *Computers and electronics in agriculture*, 147:109–117.

Romero, M., Luo, Y., Su, B., and Fuentes, S. (2018b). Vineyard water status estimation using multispectral imagery from an uav platform and machine learning algorithms for irrigation scheduling management. *Computers and Electronics in Agriculture*, 147:109–117.

Roy, K. and Kar, S. (2021). Deep learning-based hybrid approach for medical image classification. *Pattern Recognition Letters*, 140:203–212.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sahoo, R. N., Kamilaris, A., Pappas, O., and Kumar, S. (2020). Internet of things for smart irrigation systems: A comprehensive review. *IEEE Internet of Things Journal*, 7(5):4038–4063.

Saini, R. and Ghosh, S. K. (2018). Crop classification on single date sentinel-2 imagery using random forest and support vector machine. In *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, volume XLII-5, pages 683–688. International Society for Photogrammetry and Remote Sensing (ISPRS).

Sammut, C. and Webb, G. I. (2017). *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated.

Sandri, M. and Zuccolotto, P. (2006). Variable selection using random forests. In *Data analysis, classification and the forward search*, pages 263–270. Springer.

Saralajew, S., Holdijk, L., and Villmann, T. (2020). Fast adversarial robustness certification of nearest prototype classifiers for arbitrary seminorms. *Advances in Neural Information Processing Systems*, 33:13635–13650.

Savchik, P., Nocco, M., and Kisekka, I. (2024). Mapping almond stem water potential using machine learning and multispectral imagery. *Irrigation Science*, pages 1–16.

Schlimmer, J. C. and Fisher, D. (1986). A case study of incremental concept induction. In *AAAI*, volume 86, pages 496–501.

Scornet, E. (2016). Random forests and other ensemble methods. *Statistical Science*, 31(3):354–368.

Shalev-Shwartz, S. et al. (2011). Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194.

Sharma, P. K., Kumar, D., Srivastava, H. S., and Patel, P. (2018). Assessment of different methods for soil moisture estimation: A review. *Journal of Remote Sensing and GIS*, 9(1):57–73.

Sheikhi, S. and Babamir, S. M. (2016). A predictive framework for load balancing clustered web servers. *The Journal of Supercomputing*, 72(2):588–611.

Sheikhi, S. and Babamir, S. M. (2018). Using a recurrent artificial neural network for dynamic self-adaptation of cluster-based web-server systems. *Applied Intelligence*, 48(8):2097–2111.

Shekoofa, A., Sinclair, T. R., Messina, C. D., and Cooper, M. (2016). Variation among maize hybrids in response to high vapor pressure deficit at high temperatures. *Crop Science*, 56(1):392–396.

Shepp, L. A. and Vardi, Y. (1982). Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, 1(2):113–122.

Singh, A. (2005). The em algorithm. *Recuperado de: http://www. cs. cmu. edu/˜ awm/15781/assignments/EM. pdf.*

Singh, R., Kumari, V., and Singh, R. (2015). Decision tree-based prediction model for crop water demand and irrigation scheduling. *Journal of Agricultural Science*, 7(4):19–31.

Smith, M. J. (2018). Getting value from artificial intelligence in agriculture. *Animal Production Science*, 60(1):46–54.

Statista (2024). Development of the world population from 10,000bce to 2100. Accessed: 2024-09-01.

Stull, R. (2011). Wet-bulb temperature from relative humidity and air temperature. *Journal of Applied Meteorology and Climatology*, 50(11):2267–2279.

Sun, G., Xie, H., and Sinnott, R. O. (2017). A crop water stress monitoring system utilising a hybrid e-infrastructure. In *Proceedings of the10th international conference on utility and cloud computing*, pages 161–170.

Tageldin, A., Adly, D., Mostafa, H., and Mohammed, H. S. (2020). Applying machine learning technology in the prediction of crop infestation with cotton leafworm in greenhouse. *Biorxiv*, pages 2020–09.

Tang, J., Alelyani, S., and Liu, H. (2014). Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37.

Tanriverdi, C., Degirmenci, H., Gonen, E., and Boyaci, S. (2016). A comparison of the gravimetric and tdr methods in determining the corn plant's soil water content. *Scientific Papers-Series A-Agronomy*, 59:153–158.

Tavallali, P., Tavallali, P., Khosravi, M. R., and Singhal, M. (2020a). Interpretable synthetic reduced nearest neighbor: An expectation maximization approach. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1921–1925. IEEE.

Tavallali, P., Tavallali, P., Khosravi, M. R., and Singhal, M. (2020b). Interpretable synthetic reduced nearest neighbor: An expectation maximization approach. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1921–1925.

Tavallali, P., Tavallali, P., Khosravi, M. R., and Singhal, M. (2021). An em-based optimization of synthetic reduced nearest neighbor model towards multiple modalities representation with human interpretability. *Multimedia Tools and Applications*, pages 1–14.

Tavallali, P., Yazdi, M., and Khosravi, M. R. (2019). Robust cascaded skin detector based on adaboost. *Multimedia Tools and Applications*, 78(2):2599–2620.

Tsoumakas, G. and Vlahavas, I. (2002). Water consumption prediction with machine learning techniques. *Artificial Intelligence Applications and Innovations*, pages 359–366.

Turner, N. C. (1988). Measurement of plant water status by the pressure chamber technique. *Irrigation Science*, 9(4):289–308.

United States Department of Agriculture, E. R. S. (2023). Risk in agriculture. Accessed: 2024-08-27.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine learning*, 4(2):161–186.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Verstraeten, W., Veroustraete, F., and Feyen, J. (2008). Assessment of evapotranspiration and soil moisture content across different scales of observation. *Sensors*, 8(1):70–117.

Viola, P. and Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518.

Virnodkar, S. S., Pachghare, V. K., Patil, V. C., and Jha, S. K. (2020). Remote sensing and machine learning for crop water stress determination in various crops: a critical review. *Springer US*, 21.

Waleed, M., Um, T. W., Kamal, T., Khan, A., and Iqbal, A. (2020). Determining the precise work area of agriculture machinery using internet of things and artificial intelligence. *Applied Sciences*, 10(10):3365.

Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. AcM.

Wang, S., Wang, Y., and Chen, W. (2021). A comprehensive review of deep learning applications in precision agriculture. *Information Fusion*, 73:112–126.

Warner, T. A. and Nerry, F. (2009). Does single broadband or multispectral thermal data add information for classification of visible, near-and shortwave infrared imagery of urban areas? *International Journal of Remote Sensing*, 30(9):2155–2171.

Witten, I. H. and Frank, E. (2002). Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77.

Wu, J., Wang, H., Zhao, X., and Liu, Y. (2019). Cnn-knn: Convolutional neural networks combined with k-nearest neighbors for image classification. *Pattern Recognition Letters*, 125:111–120.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, Y., Gao, J., Zhang, J., Zhang, Y., and Zhang, J. (2020). Deep learning models for predicting crop drought stress using multisource remote sensing and weather data. *Computers and Electronics in Agriculture*, 179:105828.

Yang, C., Everitt, J. H., and Murden, D. (2011). Evaluating high resolution spot 5 satellite imagery for crop identification. *Computers and Electronics in Agriculture*, 75(2):347–354.

Yang, K. and Wong, W. H. (2014). Density estimation via adaptive partition and discrepancy control. *arXiv preprint arXiv:1404.1425*.

Yates, D., Islam, M. Z., and Gao, J. (2018). Spaarc: a fast decision tree algorithm. In *Australasian Conference on Data Mining*, pages 43–55. Springer.

Yin, S., Ibrahim, H., Schnable, P. S., Castellano, M. J., and Dong, L. (2021). A field-deployable, wearable leaf sensor for continuous monitoring of vapor-pressure deficit. *Advanced Materials Technologies*, 6(6):2001246.

Yu, L. and Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863.

Zarco-Tejada, P. J., Rueda, C. A., and Ustin, S. L. (2003). Water content estimation in vegetation with modis reflectance data and model inversion methods. *Remote Sensing of Environment*, 85(1):109–124.

Zhang, H. and Luo, W. (2006a). An improved k-nearest neighbor model for classification. In *8th International Conference on Machine Learning and Applications*, pages 193–198. IEEE.

Zhang, H. and Luo, W. (2006b). An improved k-nearest neighbor model for classification. In *Proceedings of the 8th international conference on machine learning and applications (ICMLA)*, pages 123–128. IEEE.

Zhang, R., Zhou, Y., Yue, Z., Chen, X., Cao, X., Ai, X., Jiang, B., and Xing, Y. (2019a). The leaf-air temperature difference reflects the variation in water status and photosynthesis of sorghum under waterlogged conditions. *PLoS One*, 14(7):e0219209.

Zhang, R., Zhou, Y., Yue, Z., Chen, X., Cao, X., Ai, X., Jiang, B., and Xing, Y. (2019b). The leaf-air temperature difference reflects the variation in water status and photosynthesis of sorghum under waterlogged conditions. *PLoS One*, 14(7):e0219209.

Zhao, T., Stark, B., Chen, Y., Ray, A. L., and Doll, D. (2017). Challenges in water stress quantification using small unmanned aerial system (suas): Lessons from a growing season of almond. *Journal of Intelligent & Robotic Systems*, 88:721–735.

Zheng, Z., Webb, G. I., and Ting, K. M. (1998). Integrating boosting and stochastic attribute selection committees for further improving the performance of decision

tree learning. In *Proceedings Tenth IEEE International Conference on Tools with Artificial Intelligence (Cat. No. 98CH36294)*, pages 216–223. IEEE.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. CRC Press.