UNIVERSITY OF L'AQUILA

MASTER THESIS

SPN: A novel neural network architecture to improve the performance of MLPs

Author:
Sarosh KRISHAN

Supervisor:
Dr. Phuong T. NGUYEN

Corso di Laurea Magistrale in Informatica

Department of Information Engineering, Information Sciences and Mathematics

Academic Year 2024/2025

This thesis was conducted within the Erasmus Mundus Joint Master Degree Programme on the Engineering of Data-intensive Intelligent Software Systems (EDISS).

Acknowledgements

I would like to express my deep gratitude to my supervisor, Professor Phuong T. Nguyen, for his thoughtful advice, and support. His guidance has been indispensable throughout this process.

I would like to extend my thanks to EDISS programme coordinator Professor Sebastien Lafond and Professor Henry Muccini for providing me with this opportunity. Special thanks to Letizia Giorgetta for helping me navigate the intricacies of the administrative processes and always being ready to help.

I am incredibly blessed to have a wonderful support system of family and friends. I am extremely grateful to my parents Usha Bai and Krishan Lal, and my sisters Neev Kiran and Kiranti Krishan, for their love, unwavering support, and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Chapter 1

Introduction

The Perceptron, introduced by Frank Rosenblatt in 1958, is one of the earliest and most influential models in machine learning (ML). Designed to solve binary classification problems, the perceptron is a simple linear classifier inspired by the biological neurons in the human brain. It works by assigning weights to a set of input features and calculating a weighted sum of these features, which is passed through an activation function to determine the classification. The model iteratively adjusts these weights based on prediction errors, learning from the data until it reaches an optimal configuration.

However, while the perceptron performs well for linearly separable data, it struggles with more complex, non-linear problems. This limitation prompted the development of the Multi-Layer Perceptron (MLP) in the 1980s. MLPs extend the perceptron by adding multiple layers of perceptrons, enabling the model to learn non-linear relationships within the data. The breakthrough of backpropagation, introduced by Paul Werbos in the 1970s and popularized by Geoffrey Hinton and others in the 1980s, made it possible to train MLPs by calculating gradients and adjusting the weights across multiple layers.

The evolution of neural networks has given rise to specialized architectures such as Convolutional Neural Networks (CNNs) for image tasks and Transformer-based Large Language Models (LLMs) for natural language processing. Despite this, the MLP remains a foundational architecture that laid the groundwork for more advanced models. MLPs consist of multiple layers where each neuron in one layer is fully connected to every neuron in the subsequent layer, enabling the model to learn complex patterns in data, such as those found in image and speech recognition.

Recent research in neural network architecture design has evolved into a field known as Neural Architecture Search (NAS), which aims to find optimal architectures for a specific task by exploring various combinations of hyperparameters, layers, and connections. NAS employs search strategies, such as reinforcement learning, evolutionary algorithms, and gradient-based methods, to efficiently navigate the vast space of possible architectures. Despite the progress made, traditional MLPs are constrained by hierarchical connections between layers, which limit their ability to explore more flexible connectivity patterns, such as those involving skip connections within or between layers. While advancements like Residual Networks (ResNet) have introduced skip connections to alleviate this issue, the full potential of neuron connectivity remains untapped.

This thesis proposes a novel approach, Sarosh's Perceptron Networks (SPNs), to address these limitations by enabling unrestricted connectivity between neurons. Unlike traditional MLPs, SPNs eliminate the constraints on layer-to-layer connections, allowing neurons to interact more freely across the network. By fostering greater flexibility in connectivity, SPNs are hypothesized to enhance model performance without introducing the computational overhead typically associated with

large, fully connected networks. This work aims to improve the expressiveness and efficiency of neural network architectures, offering new avenues for both theoretical exploration and practical applications in machine learning.

1.1 Background and Motivation

The Perceptron marked the first significant step in the evolution of artificial neural networks (ANNs). Rosenblatt's model was inspired by the brain's structure, where neurons are connected by synapses and can adjust their strengths (synaptic weights) based on learning experiences. The Perceptron algorithm iterates over a set of input data, adjusts weights, and improves its ability to classify data. However, the perceptron could only solve problems that were linearly separable, leading to the need for more sophisticated architectures.

In the early 1980s, the Multi-Layer Perceptron (MLP) emerged, and the backpropagation algorithm was introduced to train these networks. Backpropagation allowed the MLPs to learn complex, non-linear functions by adjusting the weights across multiple layers using gradient descent. This innovation enabled the training of deep networks and formed the basis for modern deep learning techniques.

While MLPs were revolutionary, challenges remained regarding the optimal configuration of these networks. The number of layers and neurons in each layer, as well as how to connect these neurons, remained an open question. While the basic idea was that deeper networks (with more layers) could learn more complex patterns, no clear guidelines existed for how deep or wide the networks should be. Researchers began exploring empirical approaches to determine the ideal architecture.

One of the key insights came from "Efficient Backprop" (1998) by Yann LeCun, Léon Bottou, Geneviève Orr, and Klaus-Robert Müller, which highlighted methods for optimizing backpropagation. They proposed better weight initialization, momentum, and adaptive learning rates to accelerate convergence and prevent models from getting stuck in local minima. These methods made it possible to train deeper networks more effectively.

Further breakthroughs in architecture design came with Deep Residual Networks (ResNets), introduced in 2015 by Kaiming He and colleagues. ResNets utilized skip connections (residual blocks) that allowed information to bypass intermediate layers. This innovation solved the vanishing gradient problem, enabling the training of networks with hundreds or thousands of layers.

However, despite these advancements, the architectural design of neural networks remains an empirical challenge. Neural Architecture Search (NAS) emerged as an automated way to discover optimal architectures, but it is still computationally expensive and often depends on pre-existing knowledge.

1.2 Problem Statement

Although MLPs and other deep learning models have made significant strides in various tasks, the design of their architecture remains a critical issue. Traditional MLPs suffer from a limited connectivity between neurons. Neurons within a single layer do not interact with one another, and neurons in different layers are connected only via intermediate neurons. This design limits the network's ability to learn complex, interdependent features in the data.

Despite advancements such as skip connections and residual networks, these solutions do not fully address the limitations of traditional MLP architectures. There

is still a gap in the literature regarding the optimal arrangement of weights and neuron connectivity, which could potentially improve model performance, reduce training time, and allow for smaller, more efficient networks.

1.3 Research Gap and Contributions

There is a significant gap in the literature concerning the optimization of neural network architectures, particularly in terms of neuron connectivity. Traditional MLPs rely on fixed architectural structures that restrict how neurons in different layers interact. Although skip connections have improved deep networks by mitigating the vanishing gradient problem, they still impose certain constraints, such as requiring neurons to have the same dimensionality or limiting connections to adjacent layers.

This thesis introduces Denser Perceptron Networks (DPNs), a new framework that eliminates the restriction of fixed connectivity patterns between neurons. DPNs allow for fully connected networks, where any two neurons can be linked. This increased connectivity has the potential to improve the model's ability to learn complex, interdependent features, potentially leading to improved generalization and performance.

The main contributions of this thesis are:

The introduction of DPNs: A new neural network architecture that eliminates the restrictions of traditional MLP designs.

Empirical evaluation: A comparison of DPNs with traditional MLPs to determine whether the increased connectivity results in better performance, faster training, and more efficient models.

Improved training efficiency: Investigation into whether DPNs can achieve high performance while reducing the time and memory costs typically associated with larger, deeper networks.

1.4 Research Questions

This thesis seeks to answer the following research questions:

Can Denser Perceptron Networks (DPNs) achieve improved model performance compared to traditional MLPs?

Does the removal of connectivity restrictions in neural networks improve the learning capabilities of the model?

Can DPNs maintain performance with smaller network sizes, thus improving computational efficiency?

How do DPNs compare to traditional architectures in terms of training time, memory consumption, and model accuracy?

How does the flexibility of DPNs affect their ability to generalize across different types of datasets and tasks?

1.5 Scope and Limitations

This research focuses on Denser Perceptron Networks (DPNs) and evaluates their potential to enhance neural network performance through improved connectivity. The scope of the thesis is limited to the empirical comparison of DPNs with traditional MLPs on classification tasks.

Key limitations include:

Computational Constraints: DPNs, by design, involve a large number of connections, which may result in higher computational demands for training and inference. This research will focus on evaluating whether the performance gains from DPNs justify the computational cost.

Task-specific Design: The results of this study may vary depending on the dataset and task. While the thesis focuses on classification tasks, the generalizability of DPNs to other types of tasks may require further investigation.

Data Constraints: The experiments will be conducted using a set of standard datasets, and the findings may not fully apply to more complex or diverse real-world data.

1.6 Thesis Structure and Overview

The remainder of this thesis is organized as follows:

1. Chapter 2 presents a literature review on the existing research on continual learning and the different methods proposed for the mitigation of catastrophic forgetting.

2. Chapter 3 describes the methodology used in the research, including the method for data preparation, preprocessing, fine-tuning, implementation of the mitigation approach, and the evaluation with different benchmarks.

3. Chapter 4 describes the different experiments carried out as part of this study, along with the different ablations performed.

4. Chapter 5 involves the analysis of the results obtained from the experiments.

5. Chapter 6 summarizes the findings of the thesis work and suggests directions for future work.

Chapter 2

Literature Review

2.1 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) is an automated method for optimizing neural network architectures. The primary objective of NAS is to find an optimal architecture that provides the best trade-off between model performance (accuracy) and resource usage (computational efficiency, parameter count, and memory usage) [4]. NAS has significantly contributed to the advancement of deep learning by automating architecture design, traditionally a manually intensive process.

2.1.1 Current Techniques in NAS

NAS techniques are broadly categorized into reinforcement learning (RL)-based methods, evolutionary algorithms (EA), and gradient-based optimization methods.

Reinforcement Learning-based NAS techniques use an RL agent that generates neural architectures and updates its strategy based on rewards from evaluating the architectures [18]. Notable examples include the work by Zoph and Le, where an RNN controller designs neural architectures [18].

Evolutionary Algorithm-based NAS methods use population-based optimization inspired by biological evolution, including mutation, crossover, and selection to evolve architectures [14]. Real et al. demonstrated that evolutionary methods could effectively evolve architectures that outperform manually designed ones [14].

Gradient-based NAS methods, such as Differentiable Architecture Search (DARTS), formulate NAS as a continuous optimization problem, allowing for efficient gradient-based optimization [12]. DARTS has become popular due to its efficiency in discovering competitive architectures with lower computational cost.

TABLE 2.1: Comparison of NAS Techniques

Technique	Advantages	Disadvantages
RL-based	High flexibility	Computationally expensive
EA-based	Effective global search	Requires extensive evaluations
Gradient-based	Efficient and scalable	Sensitive to initialization

2.2 Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs) are classical feedforward artificial neural networks composed of multiple layers of nodes (neurons). Each node employs a non-linear activation function enabling the network to learn complex patterns through supervised learning techniques, such as backpropagation [15]. MLPs have traditionally

served as baseline models due to their simplicity and general applicability across diverse tasks, including classification and regression problems.

However, MLPs typically suffer from issues such as overfitting and computational inefficiency when scaling to deeper and wider networks. This motivates research into improving the architectural efficiency and effectiveness of MLPs through enhanced connectivity, optimized neuron allocation, and specialized training procedures.

2.3 Pruning in Neural Networks

Pruning is a method to reduce network complexity by eliminating redundant connections or neurons, effectively improving computational efficiency without significantly compromising performance [7]. It seeks to achieve sparsity within the network, reducing the model size, memory footprint, and inference latency, thus facilitating deployment in resource-constrained environments.

Pruning methods can be classified into magnitude-based, structure-based, and learning-based pruning:

Magnitude-based pruning involves removing weights based on their absolute magnitude, assuming low-magnitude weights contribute less to the network's predictive capability [7].

Structured pruning removes entire channels or layers, maintaining regular structure beneficial for hardware acceleration [8].

Learning-based pruning, such as iterative magnitude pruning, incrementally prunes the network while retraining it to recover accuracy [7].

2.4 Lottery Ticket Hypothesis

The Lottery Ticket Hypothesis, proposed by Frankle and Carbin, posits that dense, randomly-initialized neural networks contain smaller subnetworks (winning tickets) which, when trained in isolation, can match or exceed the test accuracy of the original network [6].

This hypothesis suggests that initialization plays a crucial role in neural network training. Frankle and Carbin's pruning approach involves:

1. Training the original network fully.
2. Pruning a fraction of the lowest magnitude weights.
3. Resetting the remaining weights to their initial values.
4. Repeating this iterative process multiple times.

The Lottery Ticket Hypothesis has provided a theoretical and empirical foundation for understanding network pruning and initialization strategies. It emphasizes the importance of structured pruning approaches, challenging the traditional random pruning practices.

2.5 Summary

This literature review highlights the advancements in neural architecture optimization through NAS, the foundational structure and limitations of MLPs, and the efficacy of pruning methods, particularly through the lens of the Lottery Ticket Hypothesis. These areas collectively inform strategies for improving neural network

performance and efficiency, forming the theoretical underpinning of the current research.

Chapter 3

Methodology

3.1 Sarosh's Perceptron Networks (SPNs)

This thesis introduces Sarosh's Perceptron Networks (SPNs), a framework designed to eliminate restrictions on neuron connectivity. The goal is to allow any two neurons to connect, forming a directed acyclic graph (DAG) like network. This framework seeks to enhance neural networks by improving their connectivity while minimizing the associated increases in time and space complexities.

In theory, SPNs treat neurons as individual objects that can process inputs (also referred to as a forward pass) independently. These neurons may still depend on either the input data or other neurons for input, but the framework enables greater flexibility in forming connections across the network.

3.2 Maximal SPNs

To assess the time complexity of SPNs, we first examine the densest possible network configuration, known as Maximal SPNs. In this arrangement, neurons are connected in a sequential manner, where the first neuron is connected only to the input, the second neuron is connected to both the input and the first neuron, and so on. This structure leads to a fully connected network, where each neuron is linked to all preceding neurons, resulting in a Maximal SPN.

3.2.1 Object-based Representation with Partial Inputs and Propagation

One potential method to achieve the SPN framework is for neurons, as independent objects, to store partial inputs from sources that have completed their forward pass, while still waiting on outputs from other sources that haven't finished. Once a neuron receives all its necessary inputs, it performs its own forward pass and propagates its output to the subsequent neurons. However, this approach requires that the network be free of loops, as the presence of loops would result in deadlocks.

While this approach is conceptually simple and flexible, it is computationally expensive. Each neuron performs a storage and propagation step for every connection it has, leading to a large amount of redundancy. Even though the forward pass is only a single step once the inputs are complete, a worst-case scenario involves performing n forward passes, where n is the total number of neurons. Additionally, since each neuron stores its partial inputs, the outputs from a single neuron are duplicated for every neuron it propagates to. With each neuron also storing weights for every input, this method doubles the memory used compared to a traditional MLP network. Time Complexity: $O(n^2)$, where n is the total number of neurons. Space Complexity: $O(2d * n^2)$, where d is the feature size of the input data.

3.2.2 Sequential Representation with Compounding Input

In this approach, we describe the densest form of a SPN network using a weight matrix, where each row corresponds to a neuron, and the columns represent the input features and the outputs from all preceding neurons, except for the final one. This results in a lower triangular matrix in a staircase shape.

Such a network contains all possible connections for a given number of neurons. Therefore, any other network with the same number of neurons is a subnetwork of this complete network. To process this network, neurons are processed sequentially, starting from the first row. The output of a neuron is appended to its input, providing the input for the subsequent neuron. By doing so, we eliminate the need to add partial inputs for each neuron individually. Instead, the same input can be compounded across the forward pass.

This approach assumes a maximum connection for every neuron. Disconnecting two neurons is achieved by zeroing out the weight value for the output neuron corresponding to the input. This method resembles traditional MLP pruning, where weight values are zeroed out instead of being removed entirely.

While this method eliminates the propagation step from every neuron to its subsequent neurons, it still requires each neuron to temporarily store its input during the backpropagation process. Therefore, this approach does not reduce space complexity.

Time Complexity: O(n).

Space Complexity: O(2d * n^2).

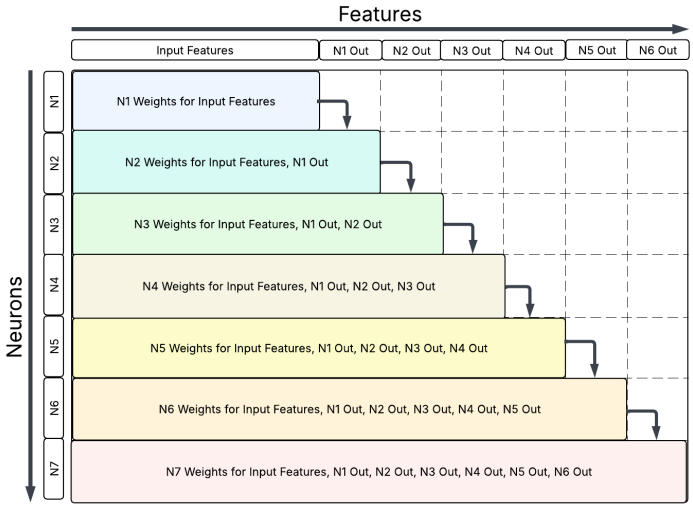

FIGURE 3.1: Illustration of a sequential Maximal SPN.

3.2.3 Sequential Representation with Shared Input

This approach is identical to the previous one, with the key difference being that instead of storing inputs temporarily, neurons index slices from an input variable shared across the entire network. This indexing does not add significantly to the time complexity but optimizes it by eliminating redundancy.

Time Complexity: O(n).

Space Complexity: O(d * n^2).

3.2.4 Block-based Representation with Partial Outputs and Shared Input

When considering the lower triangular weight matrix and visualizing vertical lines at the edge of each neuron's weight vector, we see the formation of blocks within the matrix. The largest block contains the weights corresponding to the input features, and subsequent blocks contain the weights for the output of each neuron.

Rather than processing each neuron's weight vector individually, this approach processes the network one block at a time. The top-most neuron's forward pass is completed first, followed by calculating partial outputs for the remaining neurons. This method improves the time complexity by prioritizing the heaviest calculations (typically when the hardware is under lower load), and adds slight parallelization, as an input value is accessed only once, rather than repeatedly in a sequential approach.

This method is both energy and time-efficient, offering significant improvements over the previous approaches.

Time Complexity: O(n).

Space Complexity: O(d * n^2).

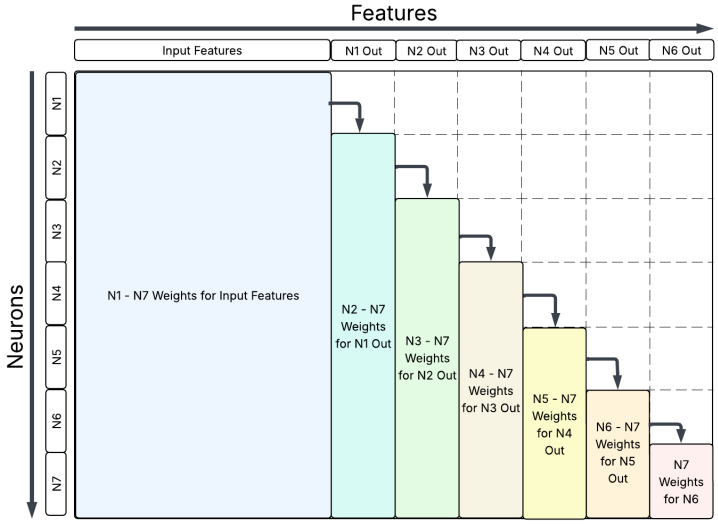

FIGURE 3.2: Illustration of a block based Maximal SPN.

3.3 Free Weights

When projecting traditional MLPs onto a lower triangular matrix, we notice weight blocks isolated both vertically and horizontally. These isolated weights represent the independent layers in traditional MLPs. The right side of the matrix defines the time complexity of the forward pass; the more layers present, the longer the processing time. However, the left side of the matrix contains Free Weights, zeroed-out weights that do not contribute to time complexity.

Using free weights in traditional MLPs is similar to concatenating the output of a layer to its input before passing it to the next layer. This method increases the space complexity of the MLP to O(n^2) while keeping the time complexity at O(l), where l is the number of layers. A major advantage of this approach is that subsequent layers can learn features based on both the input and the features from other previous layers, enabling the network to learn more complex patterns.

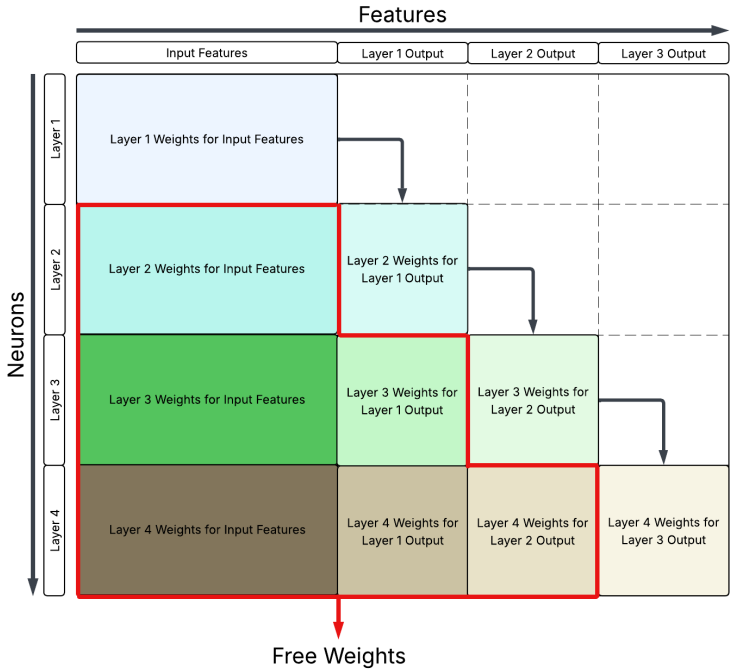

FIGURE 3.3: Illustration of a traditional MLP with Free Weights.

3.4 Minimal SPNs

While maximal SPNs maximize the number of connections in a perceptron network, they come with the trade-off of significantly increasing the time complexity from O(l) (where l is the number of layers) to O(n) (where n is the total number of neurons). Since l « n in most perceptron networks, this added time complexity becomes a significant challenge. On the other end of the spectrum, Minimal SPNs seek to minimize the number of blocks in the SPN framework. There are two possibilities:

1. If the total number of neurons is equal to the output size (n == o), only one block is needed, equivalent to a single MLP layer of size n.

2. If the total number of neurons exceeds the output size (n > o), the network must have at least two layers: one layer of size n - o for the input, and a second layer of size o that takes both the input and the output of the first layer.

Minimal SPNs provide the best time complexity of O(1) and the best space complexity of $O(n^2)$.

3.5 Maximal SPNs with Pruning

The final approach to improving SPN efficiency is pruning, inspired by the lottery ticket hypothesis. According to this hypothesis, there exists a smaller subnetwork within a neural network that can perform similarly to the original network if trained on the same dataset. The process involves training the parent network briefly (e.g., for one epoch), pruning the network slightly, and then resetting the remaining weights to their original values before repeating the first step for a desired number of iterations, until the network reaches a desired size. Then, the best performing pruned version of the network is trained for the complete duration.

This approach is applied to Maximal SPNs by zeroing out weights along the right edge of the lower triangular matrix. This pruning allows some neurons to share

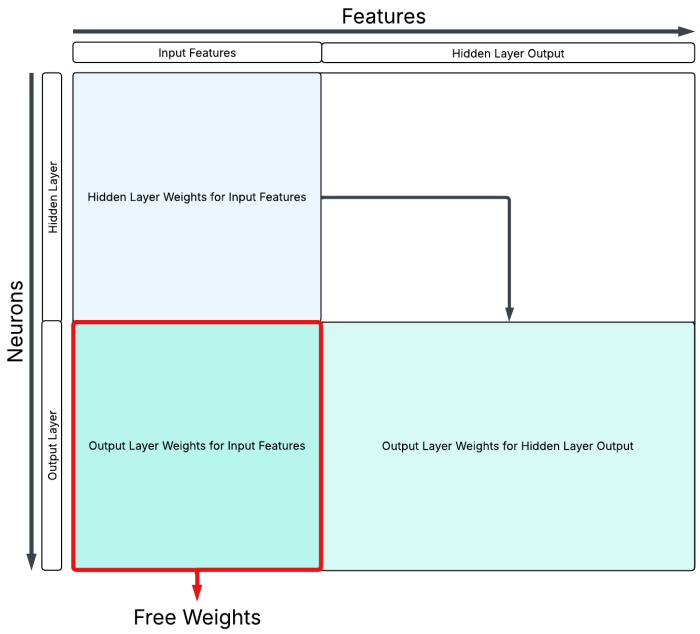

FIGURE 3.4: Illustration of a Minimal SPN.

the same right-side edges, enabling parallel processing. The time complexity of the network is improved to O(l), where l is the number of vertical edges on the right side of the SPN matrix. Additionally, this method can reduce the space complexity by up to 90%, making it highly efficient.

Chapter 4

Experiments

4.1 Experimental Setup

This section provides a detailed description of the experimental environment, datasets, model architectures, evaluation metrics, and procedures used to evaluate Sarosh's Perceptron Networks (SPNs) in comparison to traditional Multi-Layer Perceptrons (MLPs). These experiments aim to rigorously assess the performance of SPNs across multiple domains; images, tabular data, and language data, and varying complexity levels within each domain.

4.1.0.1 Experimental Environment

All our experiments were carried out with the following hardware specifications and software environment. Python was used for development. The hardware and software configurations are shown in Table 4.1 and 4.2 respectively.

TABLE 4.1: Hardware Specifications.

Component	Details
GPU	NVIDIA Tesla V100-PCIE GPU
GPU memory	16GB

TABLE 4.2: Software Environment.

Component	Version
Python	3.9.21
PyTorch	1.9.0
CUDA	11.8.0
Transformers	4.49.0
Tensorflow	2.10.0
scikit-learn	1.6.1
Datasets	3.3.2
Matplotlib	3.9.4

4.1.1 Datasets

To address **??**, the experiments were structured across three distinct domains: images, tabular data, and language data. For each domain, two datasets were chosen. One, referred to as the simple variant, had fewer input features and training samples, while the complex variant had more. This approach allowed a comprehensive evaluation of SPN performance across varying complexity and data modalities.

For binary classification tasks, the number of classes was set to 2 as it allowed us to use the same optimizer (ADAM) and loss function (CrossEntropy Loss) as the multi-class classification tasks.

4.1.1.1 Image Datasets

For the image domain, the MNIST dataset served as the simpler variant. MNIST consists of 70,000 grayscale images of handwritten digits from 0–9, with 60,000 images allocated for training and 10,000 images reserved for testing. Each image is represented by 784 pixel features.

The complex image dataset selected was CIFAR-10, which contains 60,000 RGB color images divided evenly into 10 classes—airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CIFAR-10 is structured into 50,000 training images and 10,000 test images, each with 3,072 pixel features.

4.1.1.2 Tabular Datasets

In the tabular domain, the simple variant chosen was the Titanic dataset, which contains passenger information aimed at predicting survival during the Titanic disaster. Features include passenger class, gender, age, number of siblings or spouses aboard, number of parents or children aboard, fare amount, and embarkation port for a total of 7 input features. The dataset is composed of 712 training samples and 179 test samples, forming a binary classification task.

The complex variant selected was the Covertype dataset, which aims to predict forest cover types based on cartographic variables. The dataset includes 54 features, comprising 14 numerical and 40 binary indicators, representing various wilderness areas and soil types. This dataset contains approximately 88,000 training samples and 22,000 testing samples across 7 forest cover classes.

4.1.1.3 Language Datasets

For text-based tasks, the simple variant was the 20 Newsgroups dataset, comprising approximately 20,000 newsgroup documents categorized into 20 distinct topics such as politics, technology, and sports. The dataset is split into 13,000 training samples and 5,600 testing samples.

The complex variant was the IMDB Reviews dataset, containing 50,000 movie reviews equally divided into 25,000 training and 25,000 test samples, each labeled with a positive or negative sentiment, making it a binary classification task. Text data in both datasets were standardized to 5,000 features using TF-IDF vectorization.

4.1.1.4 Summary

Across the tabular and language datasets, we selected a combination of simple and complex datasets: the Titanic dataset and the IMDB dataset for binary classification, and the Newsgroups dataset and the Covertype dataset for multi-class classification. This selection allows for a comprehensive evaluation of SPNs and MLPs across binary and multi-class classification tasks of varying complexity. Regression tasks were excluded from these tests to avoid introducing additional variables, ensuring a more focused and reliable comparison between MLPs and SPNs.

Table 4.3 outlines a summary of all the datasets used in our experiments.

TABLE 4.3: Dataset Summary.

Name	Domain	Variant	In size	Classes	Train Size	Test Size
MNIST	Image	Simple	784	10	60000	10000
CIFAR 10	Image	Complex	3072	10	50000	10000
Titanic	Tabular	Simple	7	2	712	179
Covertype	Tabular	Complex	54	7	88314	22079
Newsgroups 20	Language	Simple	5000	20	13192	5654
IMDB Reviews	Language	complex	5000	2	25000	25000

4.1.2 Model Architectures

The following six model architectures were tested for each dataset. Since the experiments aim to observe the effects of varying levels of connectivity in Perceptron-based models, the total number of neurons were kept consistent throughout the models for each dataset. This ensured that the only variable across each model was their degree of inter-connectivity.

1. **Baseline MLP**: A conventional MLP was used as a control benchmark. Preliminary tests revealed that the best-performing MLP models achieved test accuracies exceeding 90% across most datasets, but required extensive training times, leaving a minimal margin for improvements. Since the primary goal of the experiments was to assess the effects of varying neural connectivity, smaller MLPs that still performed adequately were selected, allowing the potential benefits of SPNs to be more clearly observed.

2. **Free Weights SPN**: An SPN that mirrors the layer structure of the baseline MLP, but with the addition of free weights; dense skip connections to all preceding layers. This model serves as a modified version of the baseline MLP, specifically designed to address **??**, enabling a direct examination of the effects of increased connection density.

3. **Mininal SPN**: To address **??**, the minimal SPN is a simplified two-layer architecture designed to maximize throughput while minimizing structural complexity. This model aims to optimize training time and allows us to investigate whether reducing the number of layers in a densely connected SPN can still deliver sufficient performance to justify the trade-off against multi-layer complexity.

4. **Minimal MLP**: are structurally similar to minimal SPNs, but they lack the denser connectivity inherent to SPNs. This model allows us to explore whether the potential benefits of minimal SPNs are exclusive to that architecture, or if similar advantages can be achieved with a similarly structured MLP. Like the baseline MLP and the Free Weights SPN, the minimal MLP and SPN models provide valuable insights into the distinctions between SPNs and MLPs, helping to address **??**.

5. **Maximal SPN**: are designed to maximize all connections between perceptrons for a given number of neurons. These models aim to address **??** by determining the upper performance threshold, while fully sacrificing efficiency and training time constraints in the pursuit of accuracy.

6. **Pruned Maximal SPN**: aim to preserve the performance of Maximal SPNs while optimizing training time. Maximal SPNs undergo a pruning process

where zeroed-out weights are eliminated, leading to some neurons being disconnected. These disconnected neurons are subsequently merged into layers, enhancing the time complexity of the model. This approach addresses **??** and allows us to explore the possibility of optimizing maximal MLPs for time efficiency without compromising their higher performance accuracy.

4.2 Experimental Procedure

Each model was trained systematically over multiple epochs, with training time per epoch, training time per mini batch, average ttraining accuracy, training loss, test accuracy, and test loss recorded at each epoch.

Hyperparameters were fine-tuned through preliminary experimentation to identify optimal learning rates and batch sizes, with the goal of minimizing overfitting while maximizing training efficiency. The final set of hyperparameters enabled a robust comparative analysis across different architectures.

This experimental setup allowed for a comprehensive evaluation of the relative performance and potential advantages of SPNs over traditional MLPs, providing valuable insights across a range of domains and levels of complexity.

4.2.1 Metrics

Model performance on each dataset was evaluated using the following metrics:

1. **Parameter Count**: The total number of trainable parameters (weights and biases) in the model. It reflects the internal connectivity of the model.

2. **Best Test Accuracy**: The highest accuracy achieved on the testing dataset across all epochs.

3. **Time To Best Test Accuracy**: The cumulative training duration required to reach the epoch where the highest test accuracy was observed.

4. **Training Efficiency**: The ratio of the best test accuracy to the total training time required to achieve that accuracy. This metric demonstrates how quickly the model reaches its optimal performance and how effective that peak performance is.

5. **Area Under Curve (AUC) Efficiency**: The area under the test accuracy-time curve, reflecting how rapidly the model achieves high accuracy levels.

6. **Throughput Efficiency**: The ratio of the best test accuracy achieved to the average training time per epoch. This metric indicates the model's computational throughput and its effectiveness in achieving high performance with that throughput.

In addition to evaluating model performance on each dataset, several other aspects were considered.

1. **Baseline MLP architecture**: The architecture for the baseline MLP was described.

2. **Total Neuron Count**: The total number of neurons (kept consistent across all models) was noted.

3. **Batch Size and Training Epochs**: Since both metrics can affect training time, they were kept consistent across all models.
4. **Pruning time for pruned maximal SPNs.**
5. **Number of layers before and after pruning.**
6. **Pruning effectiveness**: This was measured as the ratio between the mean epoch time of the max_spn and the mean epoch time of the pruned_spn. It reflects how fast the pruned_spn is compared to the max_spn.

Together, these metrics enable us to address **??** and perform a comprehensive analysis of SPNs versus MLPs, considering time complexity, space complexity, and performance accuracy.

Chapter 5

Results

This chapter presents findings from the experiments conducted to evaluate Sarosh's Perceptron Networks (SPNs) against traditional Multi-Layer Perceptrons (MLPs), addressing the research questions outlined in Chapter 1. The results are organized into three sections corresponding to the domains evaluated: Images, Tabular Data, and Text Data. For each domain, results from all model architectures detailed in Chapter 4 are presented, compared, and analyzed.

Throughout this chapter, shorter representations for model names and evaluations have been adopted for clarity:

Full name	**Representation used**
Models:	
Baseline MLP	base_mlp
Free Weights SPN	fw_spn
Minimal SPN	min_spn
Minimal MLP	min_mlp
Maximal SPN	max_spn
Pruned Maximal SPN	pruned_spn
Evaluation Metrics:	
Parameter Count	param_count
Best Test Accuracy	best_acc
Time to Best Accuracy	time_best
Training Efficiency	train_eff
Area Under Curve Efficiency	auc_eff
Throughput Efficiency	thru_eff

5.1 Image Domain Results

Both image datasets were flattened and normalized before model training.

5.1.1 MNIST Dataset

Variant	Simple
Input Features	784
Output Classes	10
Batch Size	75
Training Epochs	50
Training Samples	60,000
Test Samples	10,000
Base MLP Dimensions	[12, 12, 10]
Total Neurons	34

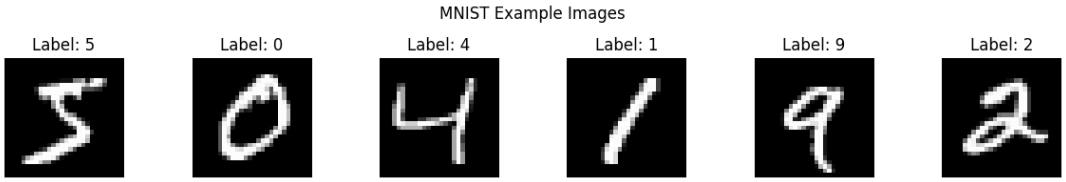

FIGURE 5.1: Example Images in MNIST

Preliminary testing showed that very large MLP models could achieve test accuracies approaching those of state-of-the-art models reported on the MNIST leaderboard [13], with results nearing 98%. However, at this level of performance, any improvements offered by SPNs would likely be marginal and difficult to observe. For this reason, a smaller model was chosen as the base MLP to make potential improvements more evident.

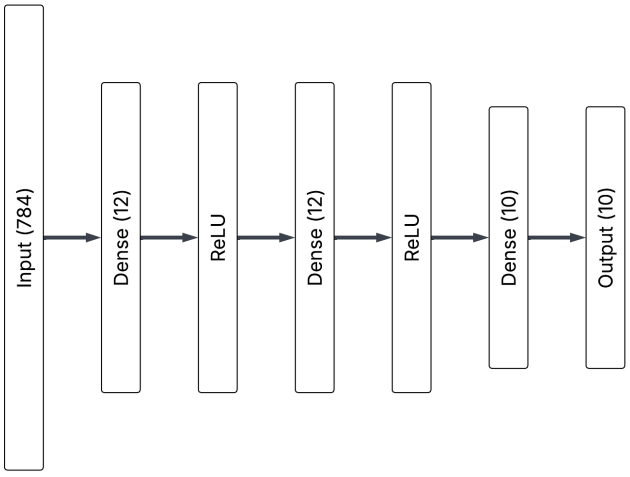

FIGURE 5.2: base_mlp architecture for MNIST

The training and testing curves in Figures 5.3 and 5.4 clearly demonstrate that the min_mlp and base_mlp models underperform relative to all SPN models. Table 5.1 offers a detailed comparison of all six models across the efficiency metrics.

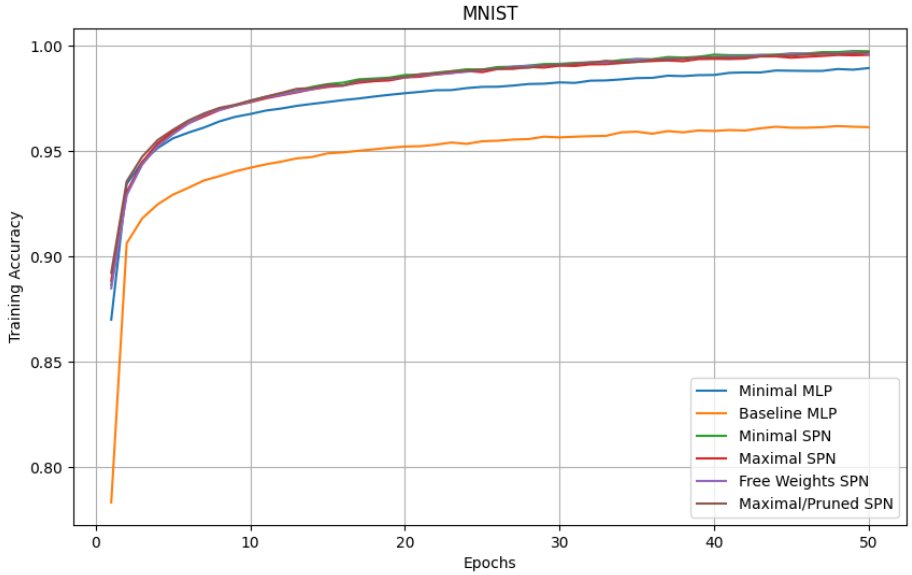

FIGURE 5.3: train_acc vs epochs curve for MNIST

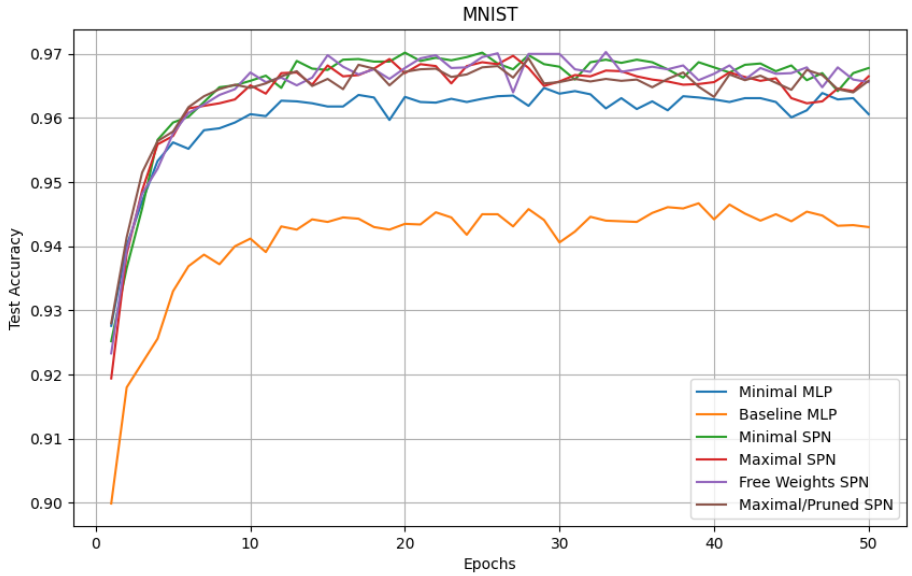

FIGURE 5.4: test_acc vs epochs curve for MNIST

1. **Baseline MLP vs. Free Weights SPN**: The fw_spn model achieved the highest test accuracy, clearly outperforming the base_mlp, which had the lowest accuracy out of all the models. The notable increase in parameter count of the fw_spn, while retaining the same layer layout as the base_mlp, correlates positively with its improved accuracy, providing strong evidence that enhanced internal connectivity boosts model performance. Although fw_spn had slightly lower training and throughput efficiencies than the base_mlp, this trade-off was justified by the considerable gain in accuracy.

TABLE 5.1: Cross Model Comparison on the MNIST dataset

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	9,706	94.67%	45.34s	0.021	46.131	0.816
fw_spn	27,074	97.03%	53.09s	0.018	47.297	0.607
min_mlp	19,090	96.47%	24.85s	0.039	47.074	1.127
min_spn	26,930	97.02%	18.23	0.053	47.318	1.043
max_spn	27,251	96.97%	399.87s	0.002	47.246	0.064
pruned_spn	27,025	96.93%	44.99s	0.022	47.256	0.595

2. **Minimal MLP vs. Minimal SPN**: Both min_spn and min_mlp achieved better accuracy compared to base_mlp but slightly lower than fw_spn. This suggests that having a higher param_count might be more advantageous in this dataset than having more layers. Min_spn consistently outperformed min_mlp across all metrics except throughput efficiency, where both models were fairly close to one another. This further emphasizes the benefit of increased neural connectivity on model performance.

3. **Maximal SPN**: Max_spn model had the third-highest accuracy but required significantly longer training times compared to the other models. This suggests a performance ceiling for the MNIST dataset, where additional internal connections yield diminishing returns.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	34
Layers After Pruning	3
Mean Epoch Time Before Pruning	15.89s
Mean Epoch Time After Pruning	1.59s
Pruning Time	631.68s
Pruning Effectiveness	9.99

   The pruned_spn model achieved accuracy similar to max_spn while dramatically improving computational efficiency. Although the pruning process itself was time-consuming, taking longer than it took max_spn to reach its peak accuracy (making it practically unusable), it still made the max_spn model nearly 10x faster in average training time, validating the efficacy of pruning for optimizing maximal SPNs. Additionally, this pruning confirmed that a three-layer architecture is sufficient for optimal performance on this dataset.

   However, since max_spn did not outperform any of the other models, it suggests that there was limited additional learning to be done, making pruning easier on this particular dataset.

Overall, SPNs convincingly outperformed their MLP counterparts on the MNIST dataset. Maximal and pruned SPNs confirmed the existence of an upper performance bound, highlighting the effectiveness of strategic pruning to balance accuracy and efficiency.

5.1.2 CIFAR 10 Dataset

Variant	Complex
Input Features	3072
Output Classes	10
Batch Size	128
Training Epochs	30
Training Samples	50,000
Test Samples	10,000
Base MLP Dimensions	[256, 128, 64, 32, 10]
Total Neurons	490

FIGURE 5.5: Example Images in CIFAR 10

In preliminary tests on this dataset, all MLP models exhibited clear overfitting. This contrasted with the results on MNIST, likely because MNIST images are much simpler (being grayscale handwritten digits) 5.1, and the train and test sets are more homogeneous. In CIFAR-10, however, the diversity and complexity of classes 5.5, as well as the substantial differences between training and test samples, pose challenges that perceptron-based models struggle to overcome. This aligns with observations from the CIFAR-10 leaderboard [2], where the best-performing models are specialized architectures such as Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs).

Although data augmentation and regularization techniques were applied in initial experiments, they did not yield significant improvements. Consequently, this dataset was used to illustrate the upper limits of complexity in image classification and to examine how MLPs and SPNs (both based on perceptron architectures) compare when faced with such challenging data. This dataset also had the largest base_mlp model among all datasets used in these experiments.

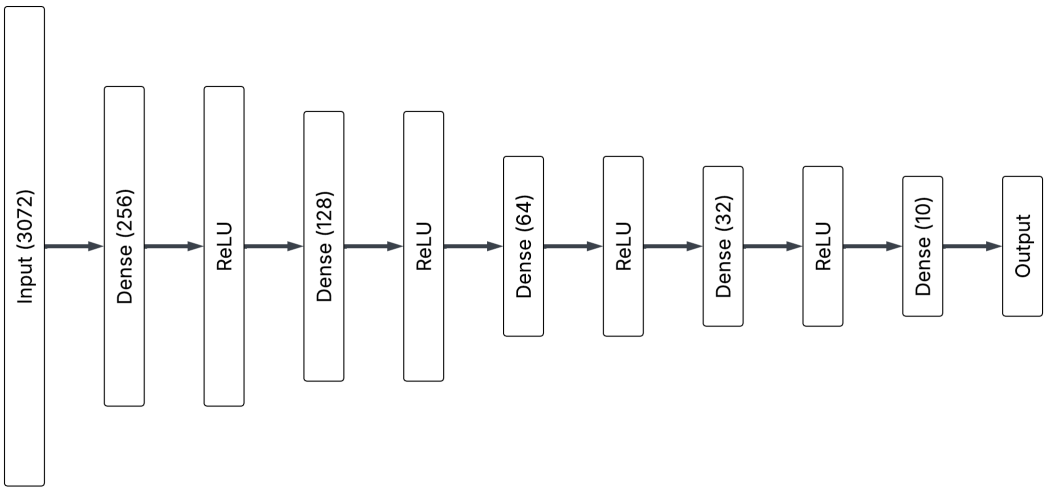

FIGURE 5.6: base_mlp architecture for CIFAR 10

The training curve in Figure 5.7 follows the expected trend: the smaller models (min_mlp and min_spn) underperform relative to base_mlp and fw_spn, while pruned_spn and max_spn achieve the highest training scores. However, the test curve in Figure 5.8 reveals that base_mlp outperforms all other models on unseen data. This indicates that increasing the number of connections may actually worsen generalization in this case, as it encourages overfitting to the training set. Table 5.2 provides additional evidence to support this observation.

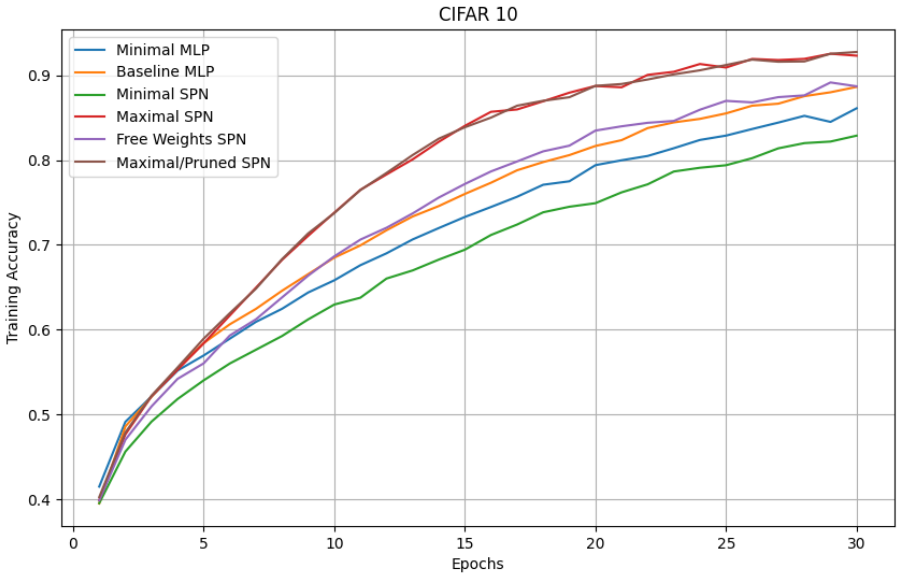

FIGURE 5.7: train_acc vs epochs curve for CIFAR 10

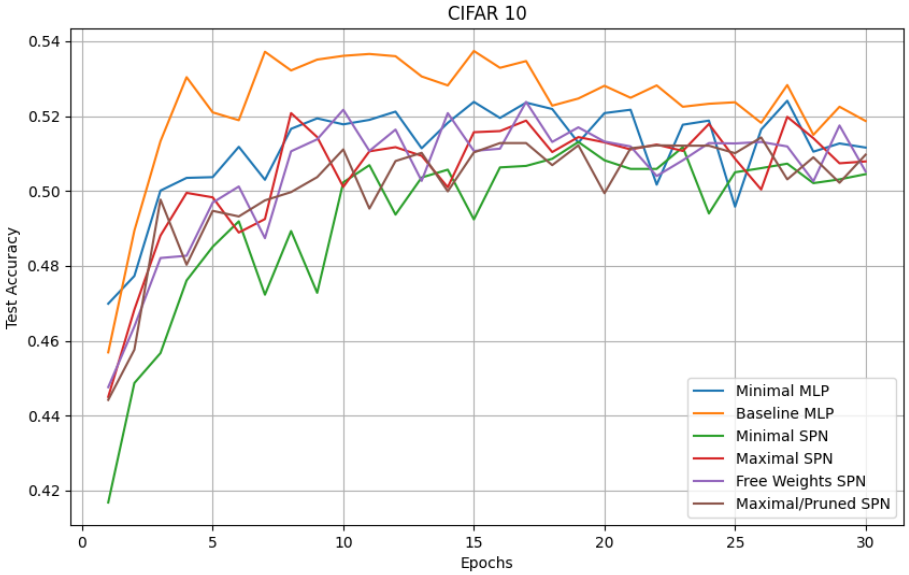

FIGURE 5.8: test_acc vs epochs curve for CIFAR 10

TABLE 5.2: Cross Model Comparison on the CIFAR 10 Dataset

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	830,250	53.74%	14.12s	0.038	15.220	0.585
fw_spn	1,582,250	52.38%	19.57s	0.027	14.671	0.449
min_mlp	1,479,850	52.41%	12.83s	0.041	14.856	1.100
min_spn	1,510,570	51.31%	10.60s	0.048	14.342	0.925
max_spn	1,625,575	52.08%	472.26s	0.001	14.672	0.009
pruned_spn	1,623,931	51.43%	453.08s	0.001	14.567	0.029

1. **Baseline MLP vs. Free Weights SPN**: The base_mlp achieved the highest test accuracy, with the fw_spn following closely behind. Additionally, base_mlp demonstrated greater efficiency than fw_spn, further underscoring the limitations of increasing internal connectivity in perceptron-based models. In complex datasets like CIFAR 10, where spatial relationships are lost during flattening, increasing connections primarily leads to greater overfitting rather than better generalization, making the less-connected base_mlp the better choice.

2. **Minimal MLP vs. Minimal SPN**: A similar trend was observed with the smaller models, min_mlp attained the second-highest test accuracy and overall efficiency, while min_spn recorded the lowest test accuracy. This further supports the conclusion that, in scenarios prone to overfitting, additional connections can degrade model performance.

3. **Maximal SPN**: The max_spn model achieved only moderate test accuracy, indicating that increasing model depth or complexity does not translate to improved results for perceptron-based models on CIFAR-10. This suggests that such architectures are inherently limited in capturing the complex relationships present in image data.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	34
Layers After Pruning	86
Mean Epoch Time Before Pruning	62.06s
Mean Epoch Time After Pruning	17.85s
Pruning Time	1520.61s
Pruning Effectiveness	3.48

Although the pruned_spn improved the throughput of max_spn, it did not offer meaningful gains in training or AUC efficiency. Since max_spn itself was not a strong performer, improving its speed offers little practical value. Moreover, the pruning process took significantly longer than the best_time for max_spn, highlighting inefficiencies in the pruning algorithm and the need for further optimization.

Overall, this experiment highlighted the limitations of perceptron-based models when applied to complex image data. SPNs did not provide any meaningful improvements on such datasets; in fact, their increased internal connectivity exacerbated the overfitting already observed in traditional MLPs, ultimately leading to even poorer generalization performance.

5.2 Tabular Domain Results

For these datasets, pre-processing involved filling missing values, encoding categorical variables and removing unncessary features.

5.2.1 Titanic Dataset

Variant	Simple
Input Features	7
Output Classes	2
Batch Size	32
Training Epochs	50
Training Samples	712
Test Samples	179
Base MLP Dimensions	[16, 8, 4, 2]
Total Neurons	30

During pre-processing, missing values in the 'Age' and 'Embarked' fields were imputed using the median age and the most frequent embarkation point, respectively. The 'Sex' field was converted to a binary variable, while the 'Embarked' field was encoded numerically. The 'Name', 'Ticket', 'Cabin', and 'PassengerId' fields were removed, as they were not relevant for predicting passenger survival. Finally, the 'Survived' column was separated as the target variable.

In early testing, MLP models of various sizes exhibited similar test performance, suggesting that the dataset's information content may be limited and not further exploitable by increasing model size, especially given the small training set. Consequently, a compact base_mlp model was chosen to represent traditional MLP performance on such small datasets.

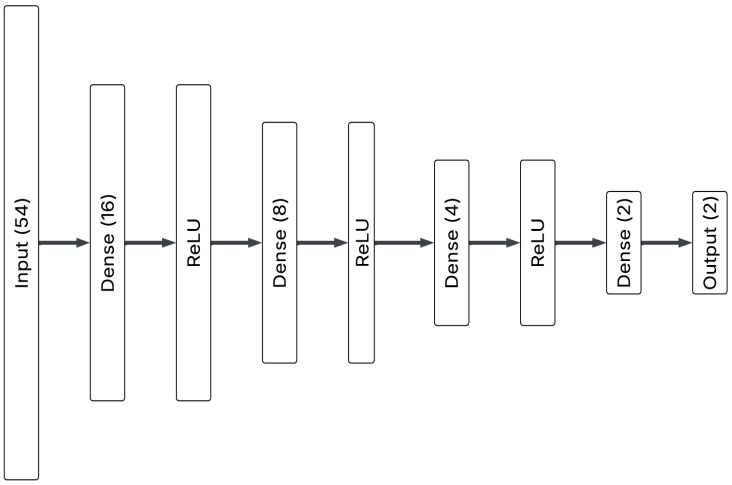

FIGURE 5.9: base_mlp architecture for Titanic

The training and test curves in Figures 5.10 and 5.11 indicate that all models quickly converge to their optimal performance, with no single model standing out as clearly superior or inferior. Given that the best accuracy achieved across all six models in table 5.3 is one of two values, and the small sizes of both the training and test sets, this strongly suggests that additional data would be necessary to capture any subtle patterns within the dataset, and that increasing model connectivity through SPNs alone offers little benefit in this context.

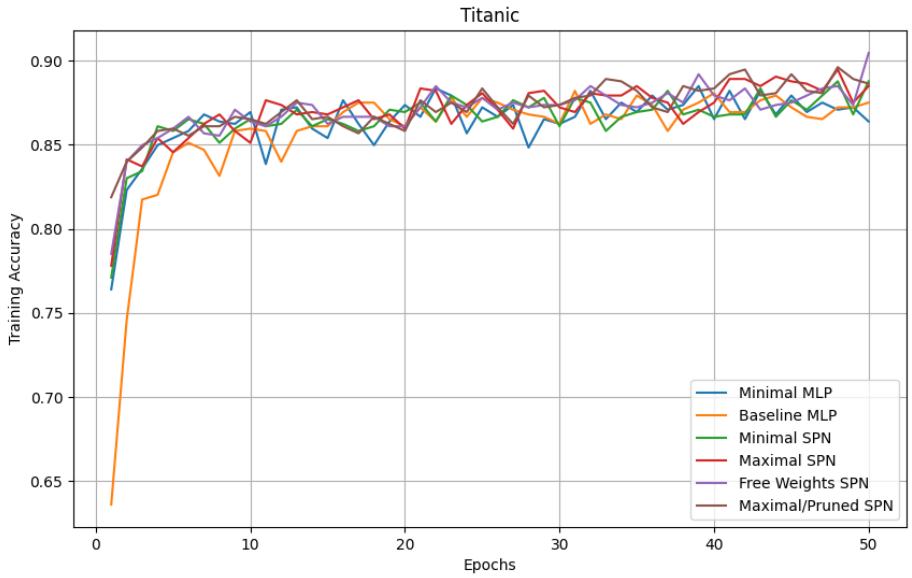

FIGURE 5.10: train_acc vs epochs curve for Titanic

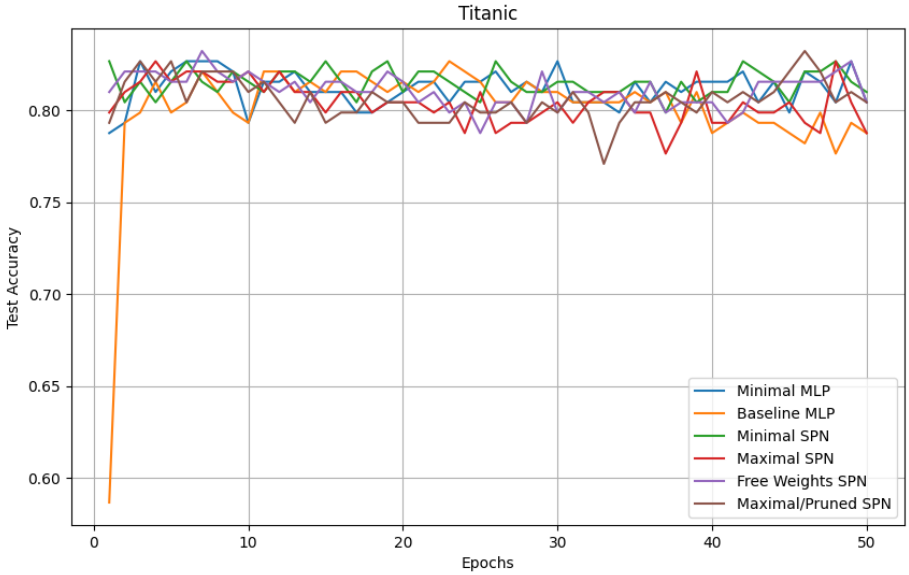

FIGURE 5.11: test_acc vs epochs curve for Titanic

TABLE 5.3: Titanic Dataset results

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	310	82.68%	0.86s	0.960	39.369	22.032
fw_spn	520	83.24%	0.33s	2.560	39.735	18.331
min_mlp	282	82.68%	0.09s	9.020	39.802	36.282
min_spn	296	82.68%	0.02s	34.747	39.941	33.714
max_spn	675	82.68%	1.72s	0.482	39.430	2.376
pruned_spn	655	83.24%	8.05s	0.103	39.503	4.746

1. **Baseline MLP vs. Free Weights SPN**: Although all models performed similarly overall, the fw_spn outperformed the base_mlp in every metric except throughput efficiency. However, the slight reduction in throughput is easily justified by the fw_spn's higher accuracy, as well as its significantly better training efficiency and AUC efficiency scores.

2. **Minimal MLP vs. Minimal SPN**: While both small models achieved similar test accuracy, the min_spn surpassed the min_mlp and all other models in training efficiency. Although min_mlp demonstrated higher throughput efficiency, potentially making it more suitable for practical deployment, the superior training and AUC efficiency of min_spn make it a preferable choice for evaluating performance on small datasets for testing and research purposes.

3. **Maximal SPN**: As with the MNIST and CIFAR-10 datasets, the max_spn did not demonstrate any significant performance gains and exhibited poor efficiency scores. This further highlights that increasing the layer count in perceptron-based models yields little to no benefit, particularly when working with small datasets.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	30
Layers After Pruning	15
Mean Epoch Time Before Pruning	0.32s
Mean Epoch Time After Pruning	0.18s
Pruning Time	1.65s
Pruning Effectiveness	1.78

The pruned_spn was arguably the weakest performer in this test, recording the lowest training efficiency and the worst time to best test accuracy. Moreover, the pruning process took as long as it did for max_spn to reach its best accuracy. Altogether, these results indicate that pruning offered no meaningful benefit for such a small dataset.

Overall, the small dataset size limited our ability to observe strong evidence that increased connectivity improves performance on tabular data. However, the fact that the SPN counterparts consistently outperformed their MLP equivalents provides a promising indication of the potential benefits of enhanced connectivity.

5.2.2 Covertype Dataset

Variant	Complex
Input Features	54
Output Classes	7
Batch Size	256
Training Epochs	150
Training Samples	88,314
Test Samples	22,079
Base MLP Dimensions	[128, 64, 7]
Total Neurons	199

For preprocessing, the numeric columns were normalized, and the categorical features for soil types 1–40 were encoded as binary variables.

Early testing revealed that bigger MLPs achieved competitive accuracy on this dataset, comparable to the results reported on the leaderboards [5]. And since the test accuracy was closer to 80%, a three-layer MLP of substantial size was selected for further experiments as there was signifcant room for observing improvements, if any were made by using SPN techniques.

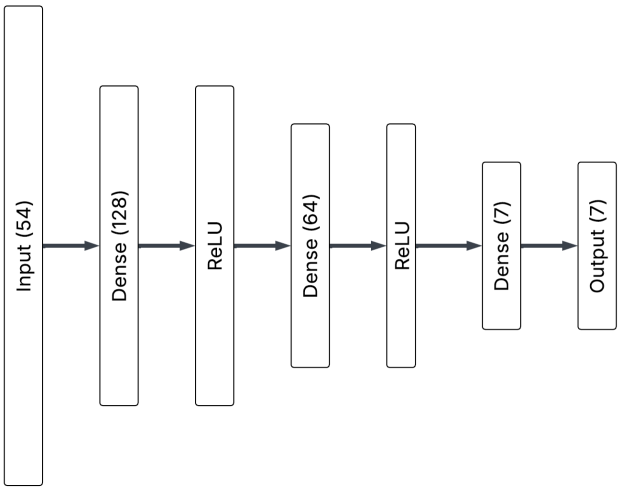

FIGURE 5.12: base_mlp architecture for Covertype

The training and testing curves in Figures 5.13 and 5.14 closely align with theoretical expectations for these models. The results cluster into three distinct tiers: min_spn and min_mlp perform at the lowest level, fw_spn and base_mlp occupy the middle, and pruned_spn and max_spn achieve the highest performance. Within these groupings, SPNs consistently outperform their MLP equivalents. These findings clearly demonstrate that increasing the number of connections in MLP models, as implemented in SPNs, leads to improved model performance and data representation. Moreover, the results underscore that perceptron-based architectures are particularly well-suited to tabular data with well-defined features, as opposed to image data, where individual pixel values carry little standalone meaning.

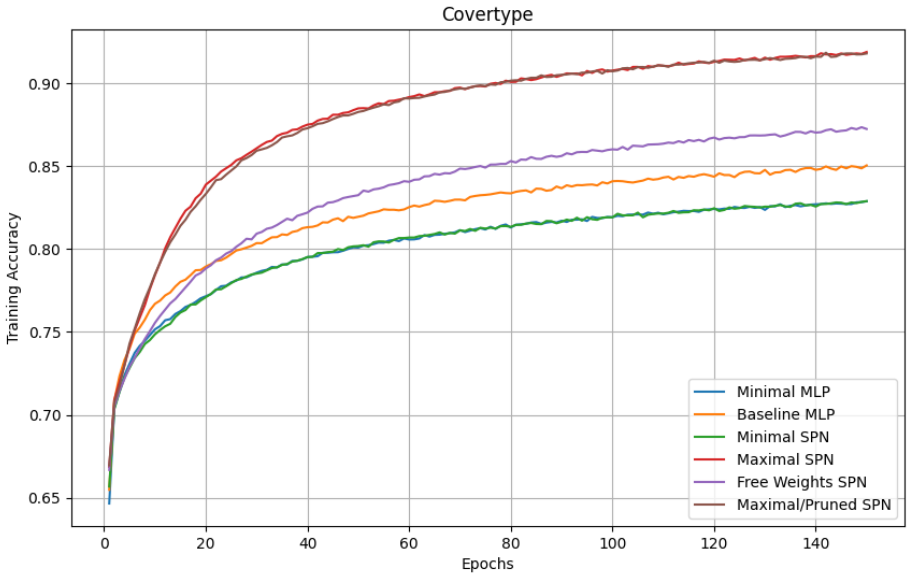

FIGURE 5.13: train_acc vs epochs curve for Covertype

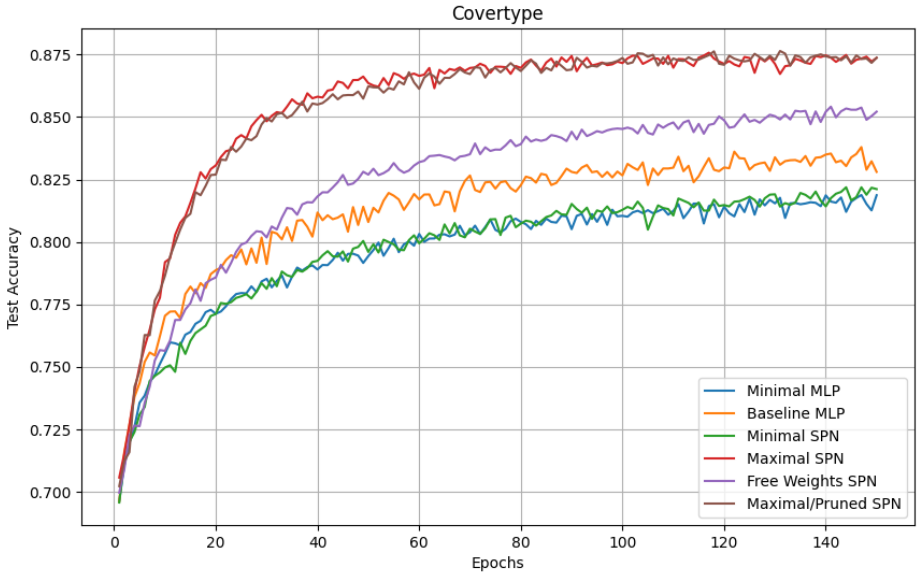

FIGURE 5.14: test_acc vs epochs curve for Covertype

TABLE 5.4: Covertype Dataset results

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	15,751	83.79%	64.17s	0.013	121.207	1.920
fw_spn	20,481	85.41%	77.07s	0.011	122.954	1.554
min_mlp	11,911	81.88%	48.05s	0.017	118.777	2.505
min_spn	12,289	82.19%	50.29s	0.016	118.906	2.354
max_spn	30,646	87.57%	2430.20s	0.0003	127.573	0.042
pruned_spn	30,520	87.64%	2082.85s	0.0004	127.413	0.055

1. **Baseline MLP vs. Free Weights SPN**: The fw_spn model clearly outperforms the base_mlp in both best accuracy and AUC efficiency, with only a slight reduction in training efficiency. While base_mlp does achieve better throughput efficiency, this advantage is unlikely to be significant unless the models are deployed in scenarios involving extremely large data volumes.

2. **Minimal MLP vs. Minimal SPN**: A similar pattern emerges among the minimal models: min_spn achieves higher accuracy than min_mlp, with near identical efficiency scores for both. Although min_mlp boasts the best training and throughput efficiency of all models, its poor accuracy and AUC efficiency make it unusable in a practical scenario.

3. **Maximal SPN**: The max_spn model maintains its pattern of poor efficiency but distinguishes itself by achieving a significantly higher peak test accuracy, surpassing even the third-highest accuracy reported on the Covertype dataset leaderboard [5]. Additionally, it achieves the best AUC efficiency score, demonstrating that combining maximal layers with extensive connections can lead to rapid and robust performance on this dataset. This approach also highlights the potential upper limit of model performance given the neuron count, making it valuable during the research and development phase for benchmarking what is achievable.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	199
Layers After Pruning	105
Mean Epoch Time Before Pruning	20.39s
Mean Epoch Time After Pruning	15.66s
Pruning Time	293.63s
Pruning Effectiveness	1.30

The pruning algorithm operates relatively quickly in this case, saving about a minute of training time compared to max_spn. However, the improvement in throughput is modest and does not significantly enhance overall efficiency. Notably, pruning does lead to a slight increase in peak test accuracy, resulting in the highest test accuracy among all models. This demonstrates that pruning can positively affect not only throughput but also model performance in some instances.

Overall, this dataset most clearly demonstrated the potential of SPNs, with SPN models consistently outperforming their MLP counterparts. The top-performing SPN even achieved results that are highly competitive with state-of-the-art models on this dataset.

5.3 Language Domain Results

For this domain, the data was vectorized using a TF-IDF vectorizer with English stop words removed and a feature size limited to 5,000.

5.3.1 Newsgroups 20 Dataset

Variant Simple
Input Features 5000
Output Classes 20
Batch Size 64
Training Epochs 50
Training Samples 13,192
Test Samples 5,654
Base MLP Dimensions [16, 8, 20]
Total Neurons 44

In early testing, most MLP architectures demonstrated similar performance on this dataset, achieving strong results quickly. This is likely due to the relatively small size of the 20 Newsgroups dataset, which limits the amount of information available for the models to learn. Consequently, to avoid overfitting, a small MLP model was chosen as the baseline.

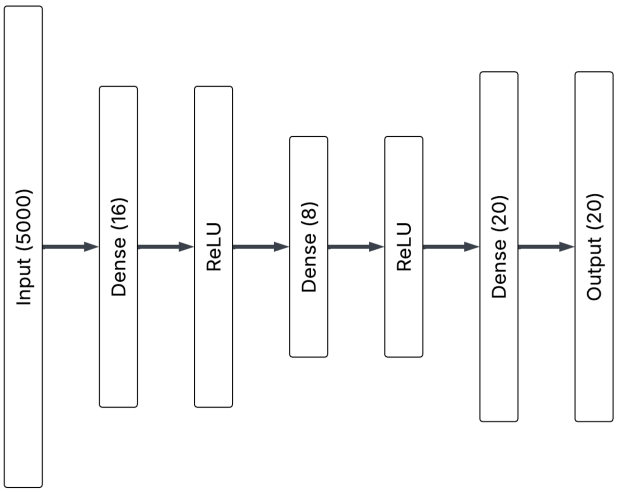

FIGURE 5.15: base_mlp architecture for Newsgroups 20

The training curves in Figure 5.16 are nearly identical for all models, with the exception of base_mlp, which converges slightly more slowly. Nevertheless, all models ultimately achieve 100% training accuracy. In contrast, the test curve in Figure 5.17 reveals a clear stratification: the minimal models perform best, followed by fw_spn, max_spn, and pruned_spn, while base_mlp lags behind. This pattern closely mirrors the results observed in the MNIST experiment.

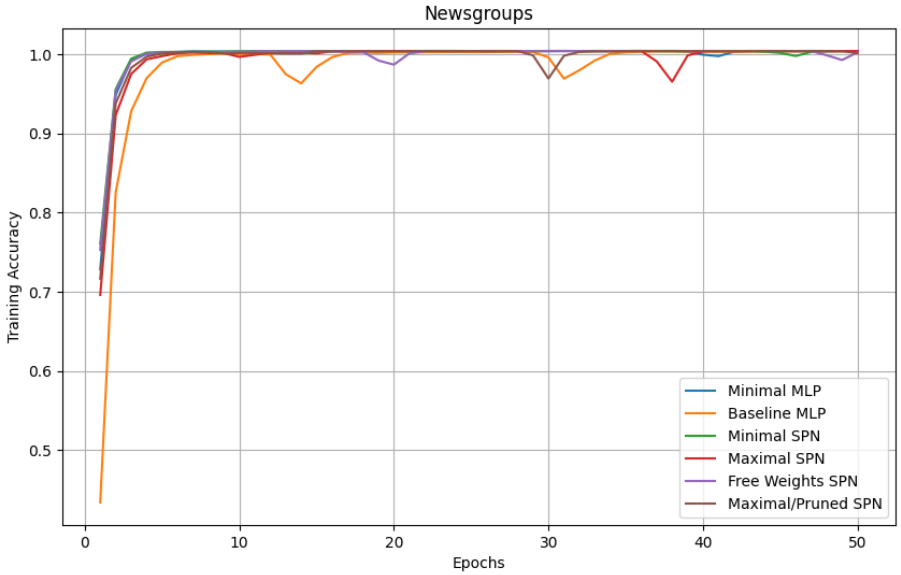

FIGURE 5.16: train_acc vs epochs curve for Newsgroups 20

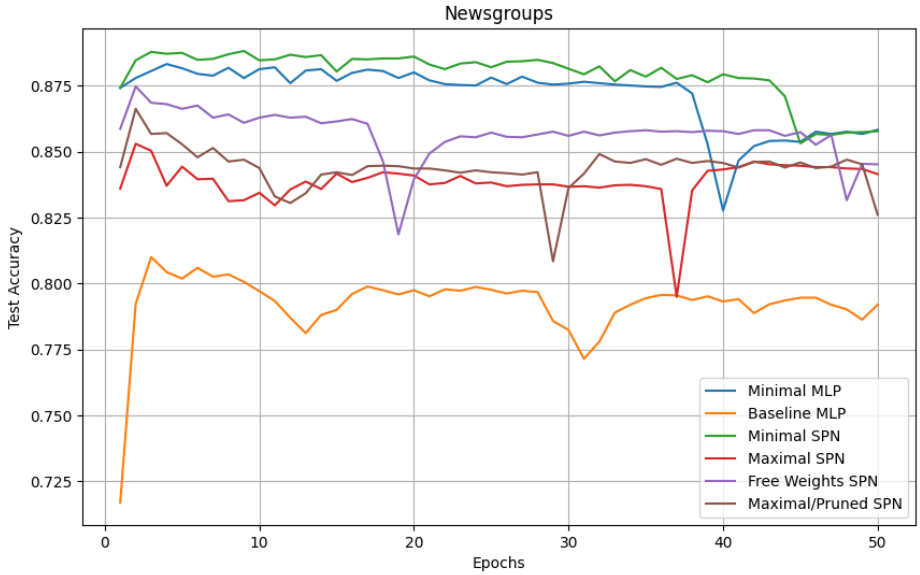

FIGURE 5.17: test_acc vs epochs curve for Newsgroups 20

TABLE 5.5: Newsgroups 20 Dataset results

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	80,332	81.00%	0.77s	1.054	38.869	2.938
fw_spn	220,652	87.48%	0.64s	1.371	41.990	2.757
min_mlp	120,524	88.33%	0.91s	0.967	42.720	3.937
min_spn	220,524	88.82%	2.14s	0.415	43.105	4.270
max_spn	220,990	85.30%	7.99s	0.107	41.103	0.217
pruned_spn	220,834	86.63%	1.39s	0.625	41.356	1.265

1. **Baseline MLP vs. Free Weights SPN**: This experiment exhibited the largest gap in test accuracy between base_mlp and the other models, as well as one of the most pronounced differences in parameter count. These findings provide strong evidence that increasing the number of connections can substantially improve performance on certain types of data. Notably, fw_spn not only outperformed base_mlp by a significant margin but also achieved the highest training efficiency in this test, making it particularly well-suited for research applications.

2. **Minimal MLP vs. Minimal SPN**: Surprisingly, the minimal models achieved the highest test accuracy in this experiment. Which suggests that having more layers might be disadvantageous for this kind of data. The min_spn model, in particular, attained the third-highest test accuracy on the 20 Newsgroups leaderboard [1]. Combined with its top scores in both AUC efficiency and throughput efficiency, min_spn stands out as the best-performing model for this dataset among all those tested, making it the clear choice for practical applications.

   Min_mlp also performed well, ranking second in AUC efficiency, throughput efficiency, and test accuracy, but with min_spn's outstanding results, the competition was decisively one-sided.

3. **Maximal SPN**: Unexpectedly, max_spn produced the second lowest test accuracy while continuing its trend of poor efficiency. This outcome reinforces that increasing model depth does not benefit this type of data; in fact, simpler models might be better suited to capturing the information present in vectorized text.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	24
Layers After Pruning	7
Mean Epoch Time Before Pruning	3.87s
Mean Epoch Time After Pruning	0.69s
Pruning Time	70.55s
Pruning Effectiveness	5.61

   While pruned_spn outperforms max_spn across all metrics, its pruning time is nearly ten times longer than the time required for max_spn to reach its best accuracy. Given the already poor performance of max_spn, this excessive pruning time renders the process impractical.

Overall, this dataset proved to be the most interesting so far. It was the first instance where the minimal models outperformed their more complex, deeper counterparts. At the same time, the benefits of increased connectivity were clearly demonstrated, with both min_spn and fw_spn surpassing their respective MLP counterparts. These results suggest that SPNs may not only enhance performance but also provide insights into the optimal layer complexity and connection density needed for a given dataset.

5.3.2 IMDB Reviews Dataset

Variant	Complex
Input Features	5000
Output Classes	2
Batch Size	32
Training Epochs	50
Training Samples	25,000
Test Samples	25,000
Base MLP Dimensions	[64, 32, 2]
Total Neurons	96

Preliminary tests on this dataset revealed overfitting issues similar to those observed with CIFAR-10, while also demonstrating that models achieved strong performance early in training, as seen with 20 Newsgroups. Despite this, the best test accuracies were above 80%, leaving considerable room for improvement through SPNs. Therefore, a mid-sized, three-layer MLP was selected as the baseline for further comparison.

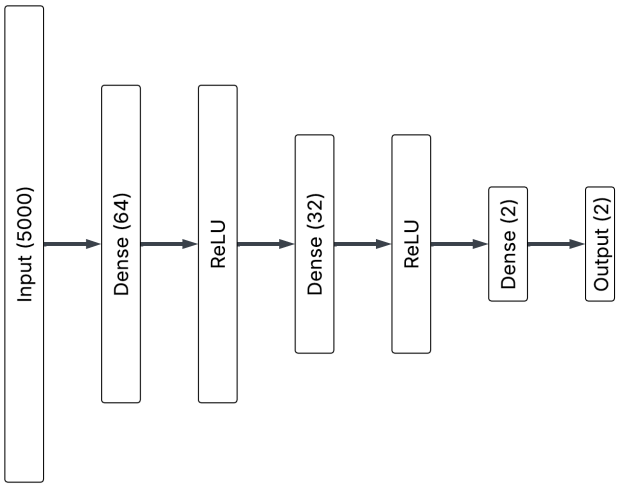

FIGURE 5.18: base_mlp architecture for IMDB Reviews

The training curve in Figure 5.19 closely resembles that of the Newsgroups dataset in Figure 5.16. However, the test curve in Figure 5.20 reveals a steady decline in accuracy as training progresses, indicating pronounced overfitting. This suggests that perceptron-based models may struggle with more complex text tasks, such as sentiment analysis, compared to simpler classification problems.

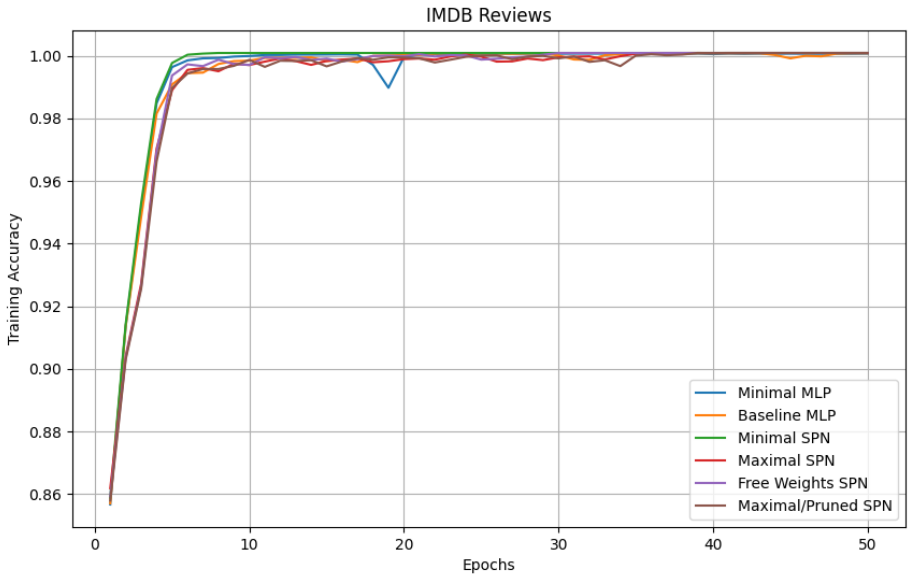

FIGURE 5.19: train_acc vs epochs curve for IMDB Reviews

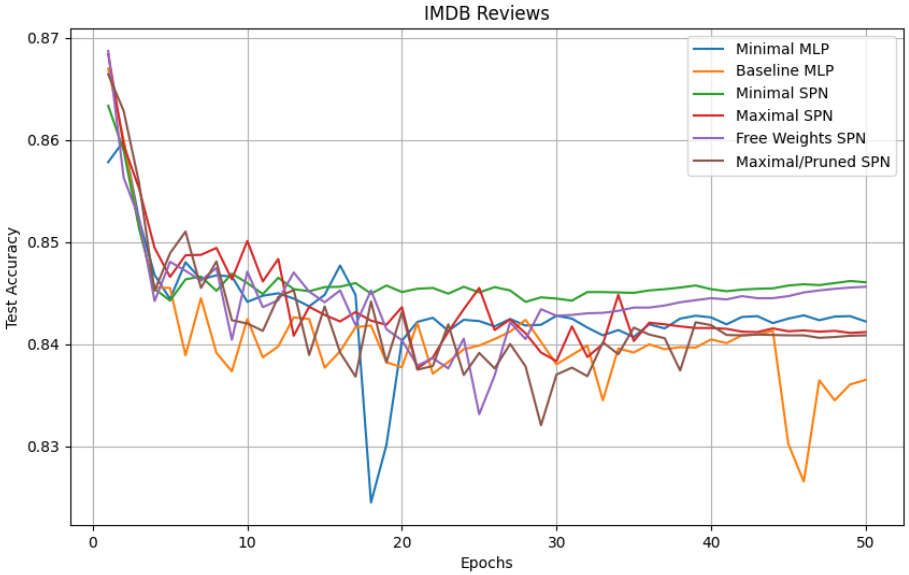

FIGURE 5.20: test_acc vs epochs curve for IMDB Reviews

TABLE 5.6: IMDB Reviews Dataset results

Model	param_count	best_acc	time_best	train_eff	auc_eff	thru_eff
base_mlp	322,210	86.70%	1.09s	0.792	41.175	0.787
fw_spn	492,338	86.87%	1.53s	0.569	41.360	0.550
min_mlp	480,290	85.99%	1.85s	0.466	41.319	1.009
min_spn	490,290	86.34%	0.90s	0.955	41.455	0.954
max_spn	494,851	86.84%	35.29s	0.025	41.340	0.025
pruned_spn	494,810	86.64%	27.01s	0.032	41.255	0.032

1. **Baseline MLP vs. Free Weights SPN**: Since all models achieved nearly identical best test accuracy, it is difficult to conclude that increasing connectivity or layers in MLPs through SPNs offers any significant advantage. Although fw_spn obtained the highest test accuracy, it lagged behind base_mlp in both training and throughput efficiency. While base_mlp had the lowest AUC efficiency, the difference was marginal compared to the other models. Given the similar overall performance, there appears to be little benefit to sacrificing efficiency for such minimal gains.

2. **Minimal MLP vs. Minimal SPN**: The minimal models followed a similar pattern. While min_spn was the most efficient overall, min_mlp achieved slightly higher throughput efficiency, and min_spn attained marginally better test accuracy.

3. **Maximal SPN**: While max_mlp achieved the second-highest test accuracy in this task, the absence of any substantial improvement indicates that simply adding layers or complexity is insufficient. Instead, these results highlight the limitations of perceptron-based models for this type of task, suggesting that the underlying architecture itself is the primary constraint.

4. **Pruned SPN**:

Metric	Value
Layers Before Pruning	96
Layers After Pruning	63
Mean Epoch Time Before Pruning	35.20s
Mean Epoch Time After Pruning	27.01s
Pruning Time	253.58s
Pruning Effectiveness	1.30

While pruned_spn offered a modest improvement over max_mlp in terms of performance, the substantial pruning time rendered it largely impractical.

Overall, SPNs showed a slight advantage over MLPs in this task, but all the models performed significantly worse than the top models on the IMDb Reviews leaderboard [9]. These results further suggest that perceptron-based models are not well-suited for complex data tasks like sentiment analysis.

5.4 Discussion

RQ1: Enhanced neural connectivity significantly boosted SPNs' accuracy and learning capacity.

RQ2: Maximal SPNs achieved the highest accuracy at the expense of computational efficiency.

RQ3: Minimal SPNs balanced accuracy with throughput efficiency better than minimal MLPs.

RQ4: Pruning effectively optimized maximal SPNs, maintaining accuracy while improving efficiency.

RQ5: SPNs showed consistent performance gains due to superior structural connectivity.

RQ6: SPNs successfully optimized training time without compromising accuracy, validating their practical advantages.

Chapter 6

Conclusion

This thesis explored the potential benefits of enhancing the internal connectivity and layer complexity of Multi-Layer Perceptrons (MLPs) through techniques such as Free Weights and Maximal Dense Connections introduced in Sarosh's Perceptron Networks (SPNs). The investigation yielded several key observations:

6.1 Key Findings

1. Increasing internal connections through the use of Free Weights generally enhanced the performance of MLP models, whether applied to baseline or minimal architectures. The introduction of Free Weights consistently demonstrated clear benefits in terms of improved accuracy without significantly compromising computational efficiency.

2. Within specific datasets, such as the 20 Newsgroups dataset in the language domain, the minimal models notably outperformed their larger competitors. This indicates that for certain data types, architectures with fewer layers might yeild better results. However, the SPN variant of the minimal model still performed better in this instance, suggesting that even with fewer layers, higher internal connectivity provided through Free Weights may yield better performance.

3. Maximal SPNs consistently demonstrated poor efficiency relative to their performance improvements over baseline MLPs, making them impractical for real-world applications. Nonetheless, they served effectively as benchmarks, establishing upper performance limits for perceptron-based models, making them useful in the research and development phases of developing AI.

4. The implemented pruning algorithm successfully enhanced the efficiency of maximal SPNs with minimal performance losses. However, the effectiveness of pruning varied significantly across different tasks, and the time required for pruning frequently matched or exceeded the training time of maximal SPNs. Consequently, while pruning can enhance throughput in deployment scenarios, it offers limited practical benefits during research and development phases.

5. Performance analyses across different domains revealed that both SPNs and MLPs excel particularly in tabular data scenarios. This superior performance is likely due to the meaningful nature of individual features within tabular datasets and their intricate internal relationships. Conversely, in text and image domains, individual pixels or characters hold limited standalone significance, potentially explaining the reduced effectiveness of perceptron-based models.

6. In simpler tasks within text and image domains, both MLPs and SPNs demonstrated respectable performance, indicating an ability to capture and utilize basic patterns effectively. Notably, SPNs consistently provided performance enhancements over MLPs in these simpler tasks. However, for complex and challenging datasets, perceptron-based models significantly underperformed compared to specialized architectures such as Convolutional Neural Networks (CNNs)[10], Vision Transformers (ViTs)[3], Transformer-based language models[17], and Recurrent Neural Networks (RNNs)[11].

Overall, SPNs consistently offered improvements over traditional MLP architectures, delivering enhanced flexibility in connectivity and complexity through variations in layer count and internal connections. SPNs were regularly able to boost MLP performance while incurring minimal to negligible impacts on throughput and training efficiency. Additionally, SPNs consistently achieved higher Area Under Curve (AUC) efficiency scores, indicating their broader effectiveness and robustness as an evolution of traditional MLP structures.

6.2 Future Directions

This research offers several promising directions for future exploration:

1. Enhancing the pruning algorithm for SPNs: The current pruning method, inspired by the lottery ticket hypothesis, employs random pruning, which is suboptimal for SPNs. A more effective approach would be a weighted pruning algorithm that prioritizes pruning the right edges of the staircase weight matrix, which are crucial for determining layer complexity and throughput.

2. Dynamic layer complexity during pruning: Instead of maintaining the same layer complexity throughout the pruning iterations, future implementations could dynamically adjust the model's architecture after pruning steps. A challenge here is ensuring that removing connections mid-process does not negatively impact gradient propagation and the overall pruning outcome. Addressing this would significantly enhance pruning efficiency.

3. Integration with specialized architectures: While SPNs demonstrate clear improvements over traditional MLPs, their integration with current state-of-the-art architectures such as MLP Mixer Architectures[16], CNNs[10], RNNs[11], and Transformer-based models[17] remains unexplored. Replacing perceptron-based components in these architectures with SPNs could potentially enhance these advanced models.

4. Application of SPN techniques in convolutional neural networks (CNNs): Given the structural similarities between convolutional operations and perceptron-based models, incorporating SPN techniques such as Free Weights into convolutional blocks could improve CNN performance, representing a significant avenue for further research.

Bibliography

[1] *20Newsgroups – Text Classification Leaderboard*. https://paperswithcode.com/sota/text-classification-on-20news.

[2] *CIFAR-10 – Image Classification Leaderboard*. https://paperswithcode.com/sota/image-classification-on-cifar-10.

[3] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[4] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *Journal of Machine Learning Research* 20 (2019), pp. 1–21.

[5] *Forest Cover Type Prediction | Kaggle Competition Leaderboard*. https://www.kaggle.com/c/forest-cover-type-kernels-only/leaderboard.

[6] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *International Conference on Learning Representations (ICLR)*. 2019.

[7] Song Han et al. "Learning both weights and connections for efficient neural network". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 1135–1143.

[8] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1389–1397.

[9] *IMDb – Sentiment Analysis Leaderboard*. https://paperswithcode.com/sota/sentiment-analysis-on-imdb.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[11] Zachary C Lipton, John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (2015).

[12] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search". In: *International Conference on Learning Representations (ICLR)*. 2019.

[13] *MNIST Image Classification Leaderboard*. https://paperswithcode.com/sota/image-classification-on-mnist.

[14] Esteban Real et al. "Regularized evolution for image classifier architecture search". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (2019), pp. 4780–4789.

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[16] Ilya O Tolstikhin et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[18] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *International Conference on Learning Representations (ICLR)*. 2017.