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

Acknowledgements iii

1 Introduction 1
- 1.1 Background and Motivation . . . 2
- 1.2 Problem Statement . . . 2
- 1.3 Research Gap and Contributions . . . 3
- 1.4 Research Questions . . . 3
- 1.5 Scope and Limitations . . . 3
- 1.6 Thesis Structure and Overview . . . 4

2 Literature Review 5
- 2.1 Main Section 1 . . . 5
  - 2.1.1 Subsection 1 . . . 5
  - 2.1.2 Subsection 2 . . . 5
- 2.2 Main Section 2 . . . 5

3 Methodology 7
- 3.1 Sarosh's Perceptron Networks (SPNs) . . . 7
- 3.2 Maximal SPNs . . . 7
  - 3.2.1 Object-based Representation with Partial Inputs and Propagation . . . 7
  - 3.2.2 Sequential Representation with Compounding Input . . . 8
  - 3.2.3 Sequential Representation with Shared Input . . . 8
  - 3.2.4 Block-based Representation with Partial Outputs and Shared Input . . . 8
- 3.3 Free Weights . . . 9
- 3.4 Minimal SPNs . . . 9
- 3.5 Maximal SPNs with Pruning . . . 9

4 Experiments 11
- 4.1 Experimental Setup . . . 11
  - 4.1.0.1 Experimental Environment . . . 11
  - 4.1.1 Datasets . . . 11
    - 4.1.1.1 Image Datasets . . . 12
    - 4.1.1.2 Tabular Datasets . . . 12
    - 4.1.1.3 Language Datasets . . . 12
    - 4.1.1.4 Summary . . . 12
  - 4.1.2 Model Architectures . . . 13
- 4.2 Experimental Procedure . . . 15
  - 4.2.1 Metrics . . . 15

5 Results **17**
5.1 Image Domain Results . . . 17
5.1.1 MNIST Dataset . . . 17
5.1.2 CIFAR 10 Dataset . . . 20
5.2 Tabular Domain Results . . . 23
5.2.1 Titanic Dataset . . . 23
5.2.2 Covertype Dataset . . . 26
5.3 Language Domain Results . . . 29
5.3.1 Newsgroups 20 Dataset . . . 29
5.3.2 IMDB Reviews Dataset . . . 32
5.3.3 Subsection 1 . . . 35
5.3.4 Subsection 2 . . . 35
5.4 Main Section 2 . . . 35

6 Conclusion **37**
6.1 Main Section 1 . . . 37
6.1.1 Subsection 1 . . . 37
6.1.2 Subsection 2 . . . 37
6.2 Main Section 2 . . . 37

A Frequently Asked Questions **39**
A.1 How do I change the colors of links? . . . 39

Bibliography **41**

List of Figures

4.1 MLP Weight Matrix 14
4.2 Free Weights SPN Matrix 14
4.3 Maximal SPN Weight Matrix 14
4.4 Minimal MLP Weight Matrix 15
4.5 Minimal SPN Weight Matrix 15

5.1 base_mlp architecture for MNIST 18
5.2 train_acc vs epochs curve for MNIST 18
5.3 test_acc vs epochs curve for MNIST 19
5.4 base_mlp architecture for CIFAR 10 21
5.5 train_acc vs epochs curve for CIFAR 10 21
5.6 test_acc vs epochs curve for CIFAR 10 22
5.7 base_mlp architecture for Titanic 24
5.8 train_acc vs epochs curve for Titanic 25
5.9 test_acc vs epochs curve for Titanic 26
5.10 base_mlp architecture for Covertype 27
5.11 train_acc vs epochs curve for Covertype 28
5.12 test_acc vs epochs curve for Covertype 29
5.13 base_mlp architecture for Newsgroups 20 30
5.14 train_acc vs epochs curve for Newsgroups 20 31
5.15 test_acc vs epochs curve for Newsgroups 20 32
5.16 base_mlp architecture for IMDB Reviews 33
5.17 train_acc vs epochs curve for IMDB Reviews 34
5.18 test_acc vs epochs curve for IMDB Reviews 35

List of Tables

4.1 Hardware Specifications. 11
4.2 Software Environment. 11
4.3 Dataset Summary. 13

5.1 Cross Model Comparison on the MNIST dataset 19
5.2 Cross Model Comparison on the CIFAR 10 Dataset 22
5.3 Titanic Dataset results 24
5.4 Covertype Dataset results 27
5.5 Newsgroups 20 Dataset results 30
5.6 IMDB Reviews Dataset results 33

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 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

2.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

2.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

2.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

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).

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).

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.

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 $l \ll n$ 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 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.

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

TABLE 4.3: Dataset Summary.

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.

Figures 4.1, 4.2 and 4.3 illustrate the differences between the baseline MLP, Free Weights SPN, and Maximal SPN model architectures while figures 4.4 and 4.5 highlight a direct comparison between minimal MLP and SPN architectures.

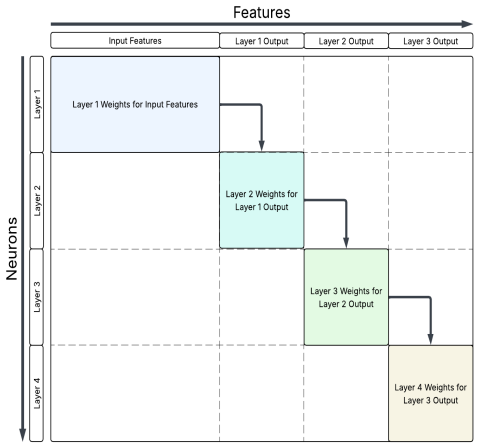

FIGURE 4.1: MLP Weight Matrix

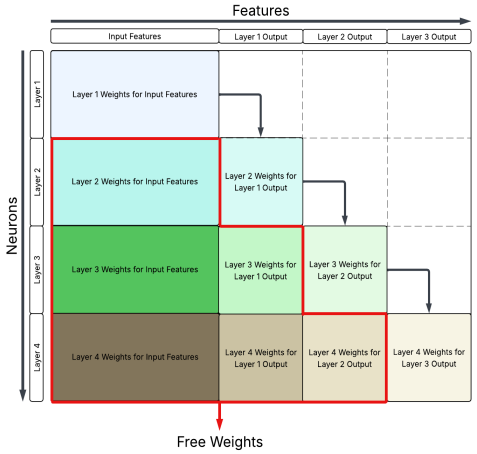

FIGURE 4.2: Free Weights SPN Matrix

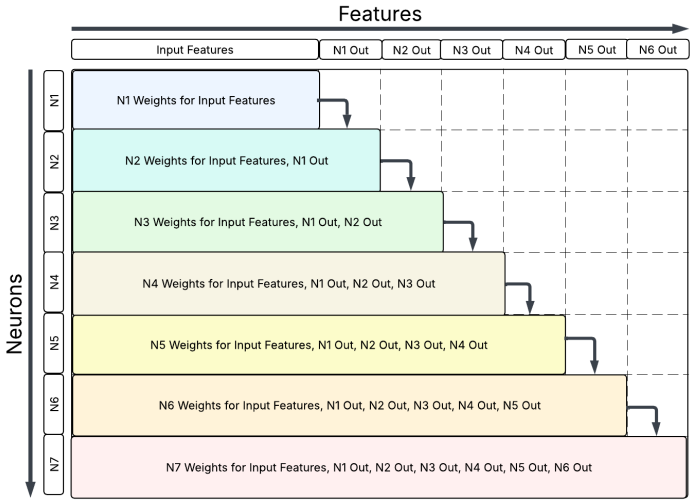

FIGURE 4.3: Maximal SPN Weight Matrix

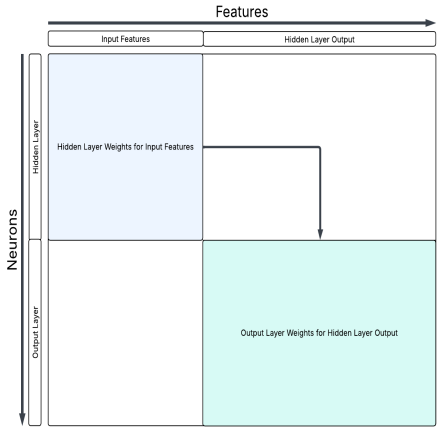

FIGURE 4.4: Minimal MLP Weight Matrix

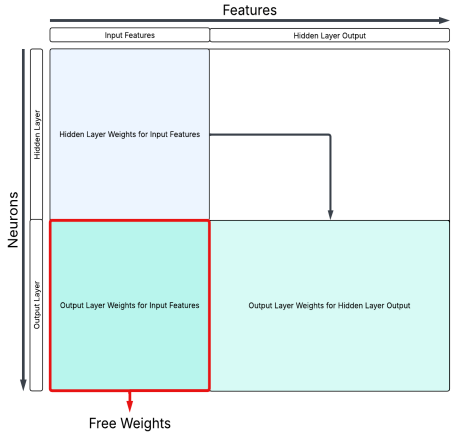

FIGURE 4.5: Minimal SPN Weight Matrix

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

Hellobib [1]

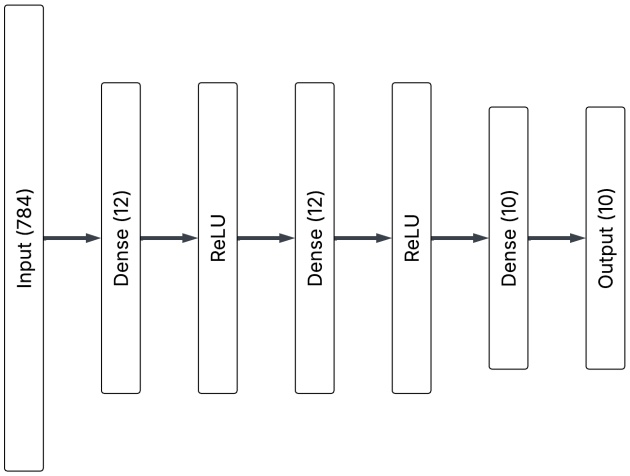

FIGURE 5.1: base_mlp architecture for MNIST

The training and testing curves in Figures 5.2 and 5.3 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.

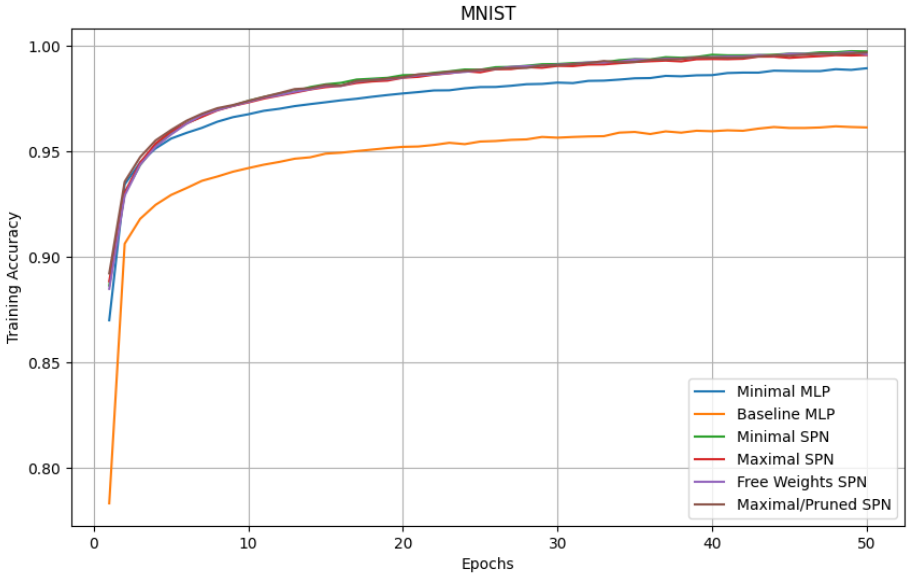

FIGURE 5.2: train_acc vs epochs curve for MNIST

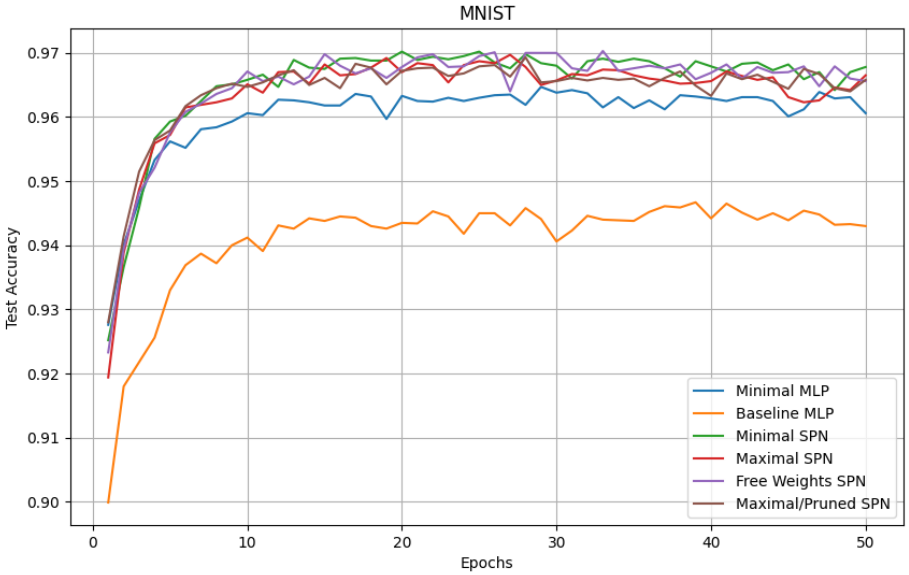

FIGURE 5.3: test_acc vs epochs curve for MNIST

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_spn	26,930	97.02%	18.23	0.053	47.318	1.043
min_mlp	19,090	96.47%	24.85s	0.039	47.074	1.127
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

TABLE 5.1: Cross Model Comparison on the MNIST dataset

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.

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

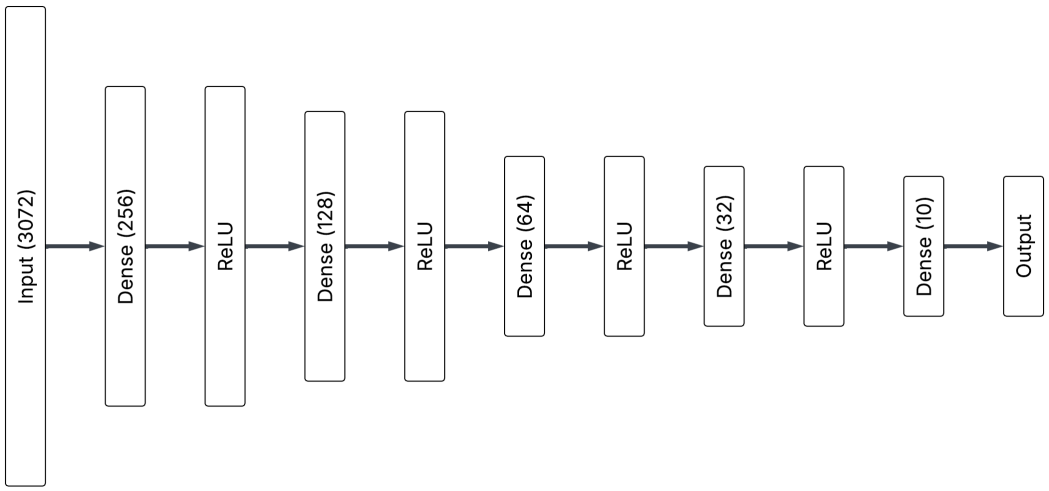

FIGURE 5.4: base_mlp architecture for CIFAR 10

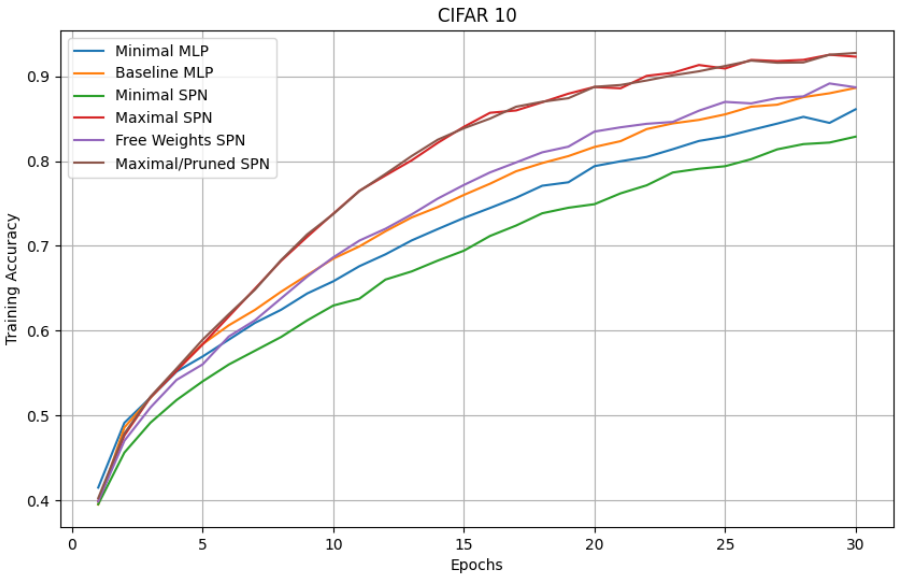

FIGURE 5.5: train_acc vs epochs curve for CIFAR 10

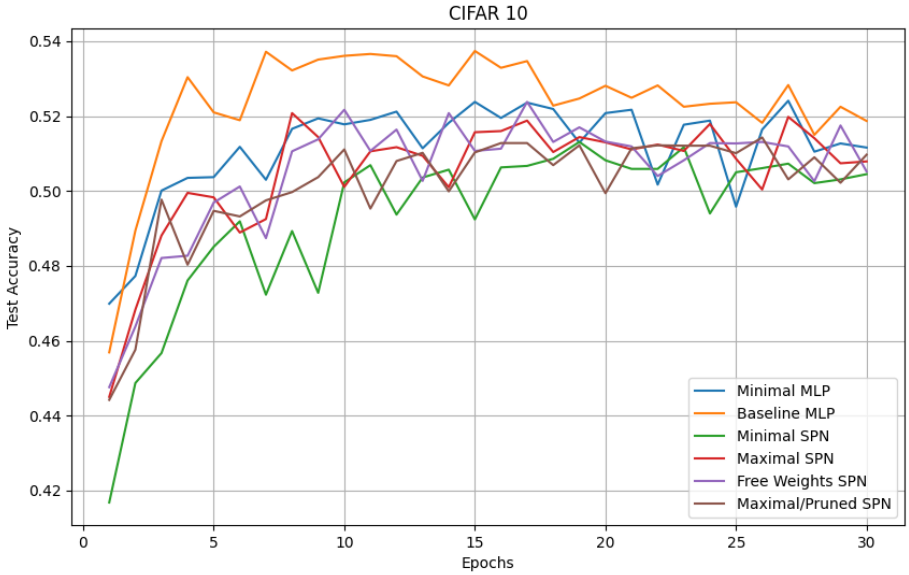

FIGURE 5.6: test_acc vs epochs curve for CIFAR 10

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_spn	1,510,570	51.31%	10.60s	0.048	14.342	0.925
min_mlp	1,479,850	52.41%	12.83s	0.041	14.856	1.100
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

TABLE 5.2: Cross Model Comparison on the CIFAR 10 Dataset

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.

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	86
Mean Epoch Time Before Pruning	62.06s
Mean Epoch Time After Pruning	17.85s
Pruning Time	1520.61s
Pruning Effectiveness	3.48

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.

5.2 Tabular Domain Results

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

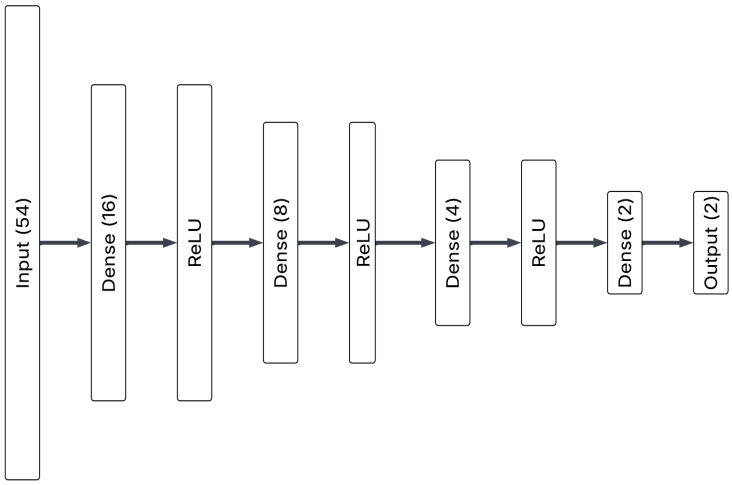

FIGURE 5.7: base_mlp architecture for Titanic

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_spn	296	82.68%	0.02s	34.747	39.941	33.714
min_mlp	282	82.68%	0.09s	9.020	39.802	36.282
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

TABLE 5.3: Titanic Dataset results

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.

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	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 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.

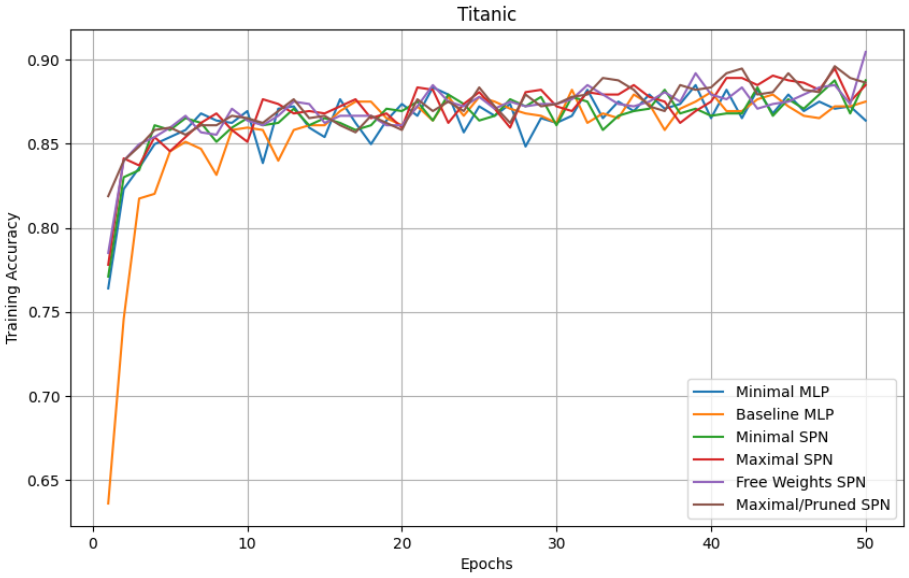

FIGURE 5.8: train_acc vs epochs curve for Titanic

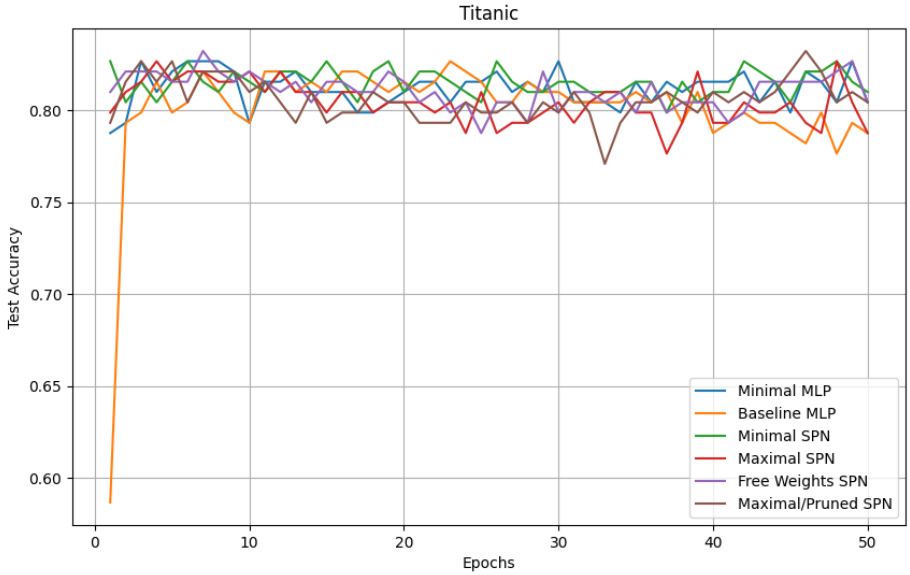

FIGURE 5.9: test_acc vs epochs curve for Titanic

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.

5.2.2 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

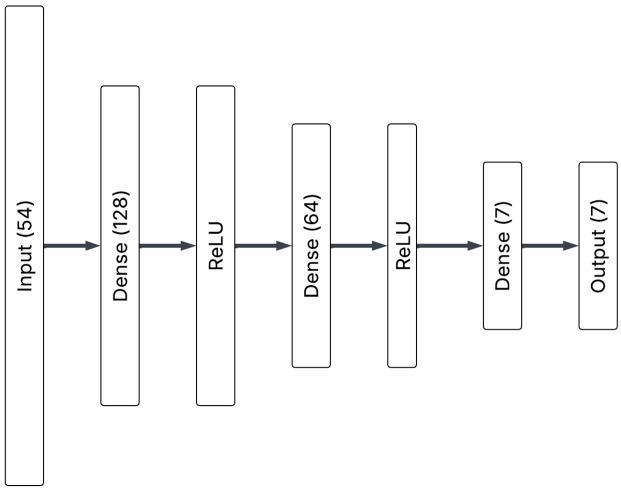

FIGURE 5.10: base_mlp architecture for Covertype

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_spn	12,289	82.19%	50.29s	0.016	118.906	2.354
min_mlp	11,911	81.88%	48.05s	0.017	118.777	2.505
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

TABLE 5.4: Covertype Dataset results

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.

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	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 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.

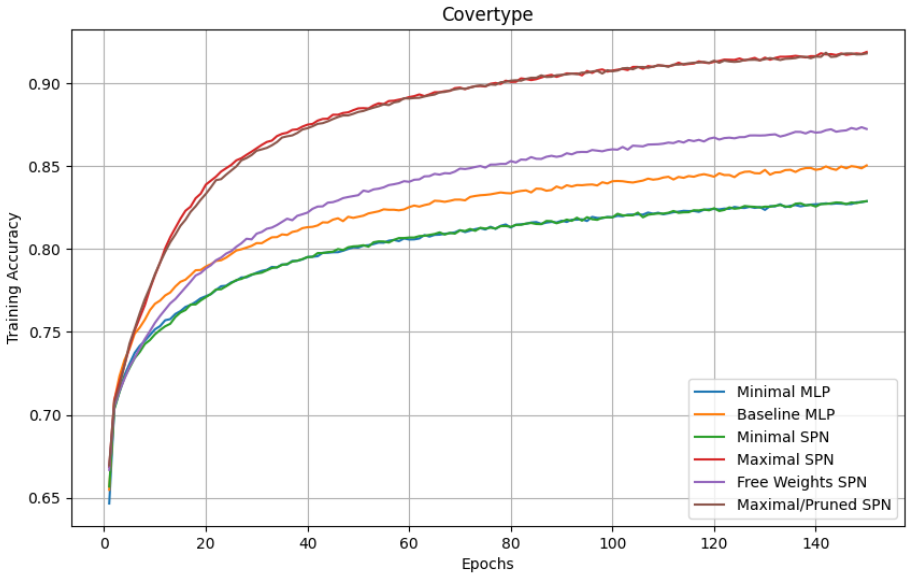

FIGURE 5.11: train_acc vs epochs curve for Covertype

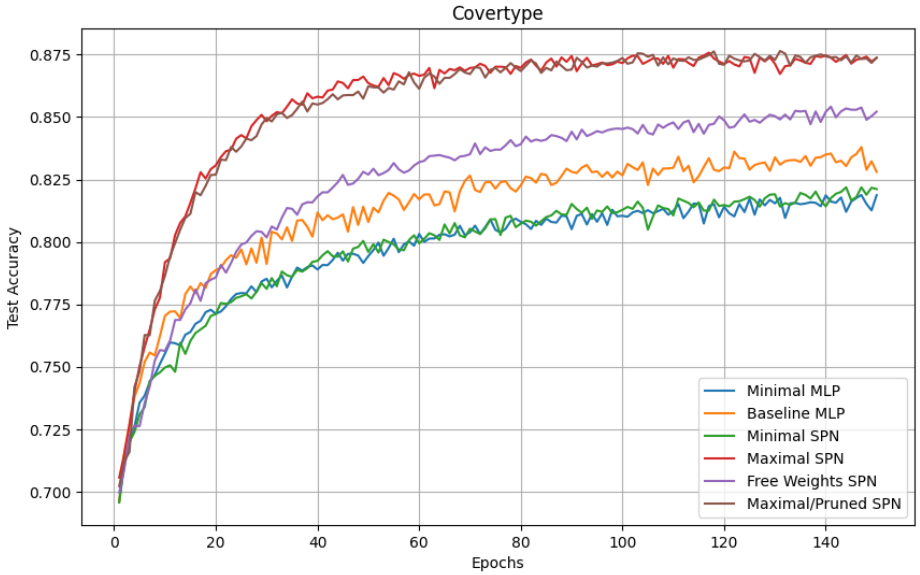

FIGURE 5.12: test_acc vs epochs curve for Covertype

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.

5.3 Language Domain Results

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

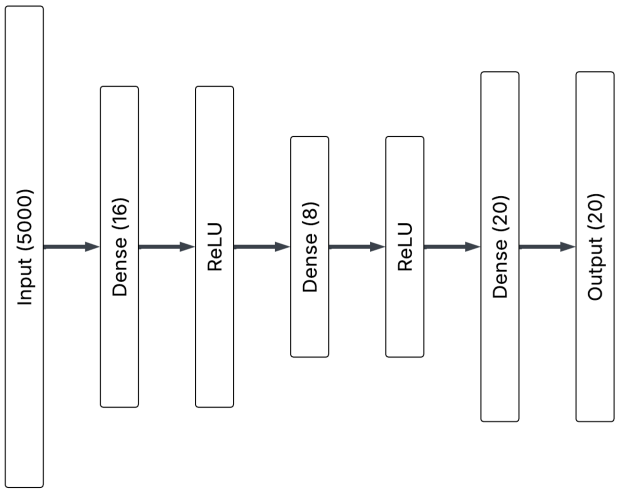

FIGURE 5.13: base_mlp architecture for Newsgroups 20

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_spn	220,524	88.82%	2.14s	0.415	43.105	4.270
min_mlp	120,524	88.33%	0.91s	0.967	42.720	3.937
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

TABLE 5.5: Newsgroups 20 Dataset results

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.

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	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

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.

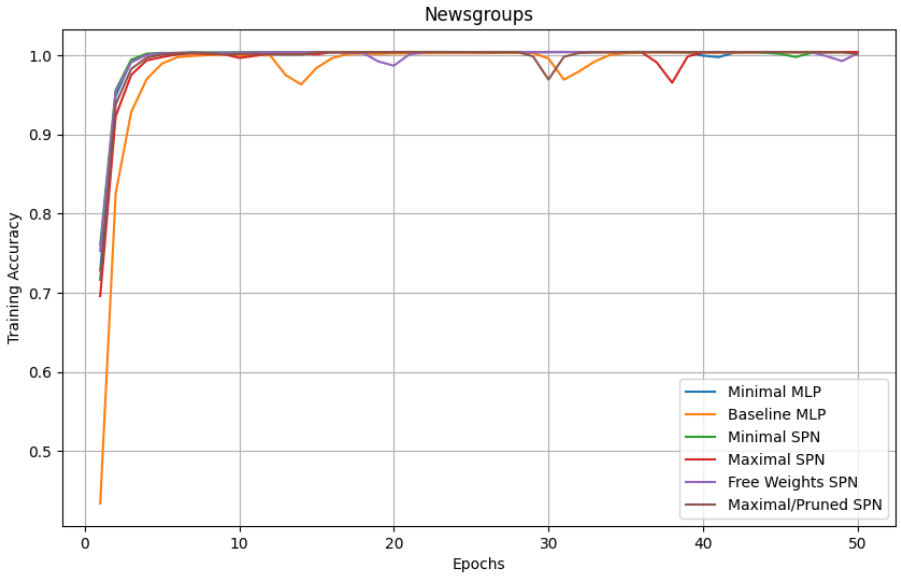

FIGURE 5.14: train_acc vs epochs curve for Newsgroups 20

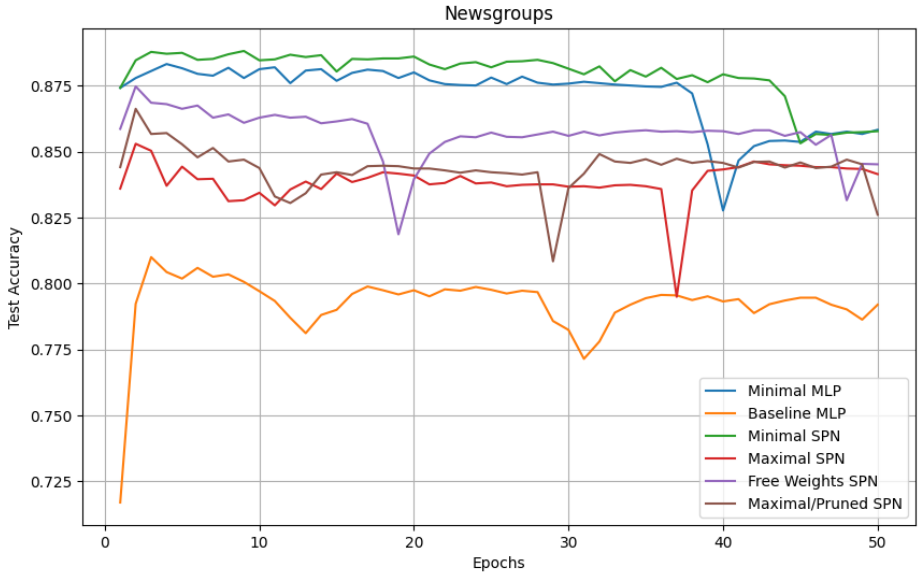

FIGURE 5.15: test_acc vs epochs curve for Newsgroups 20

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.

5.3.2 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

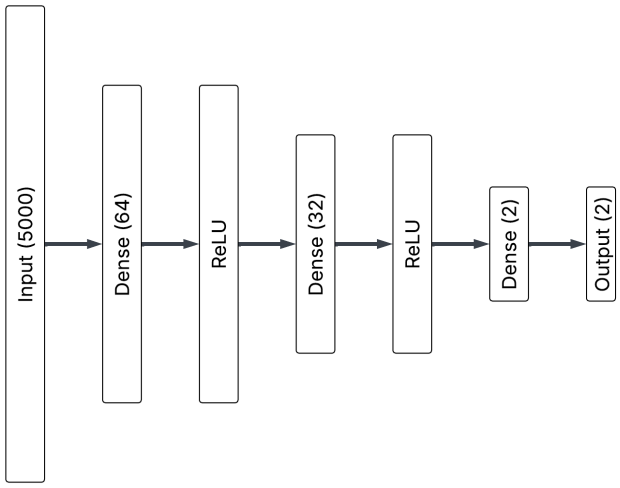

FIGURE 5.16: base_mlp architecture for IMDB Reviews

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_spn	490,290	86.34%	0.90s	0.955	41.455	0.954
min_mlp	480,290	85.99%	1.85s	0.466	41.319	1.009
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

TABLE 5.6: IMDB Reviews Dataset results

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.

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	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

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.

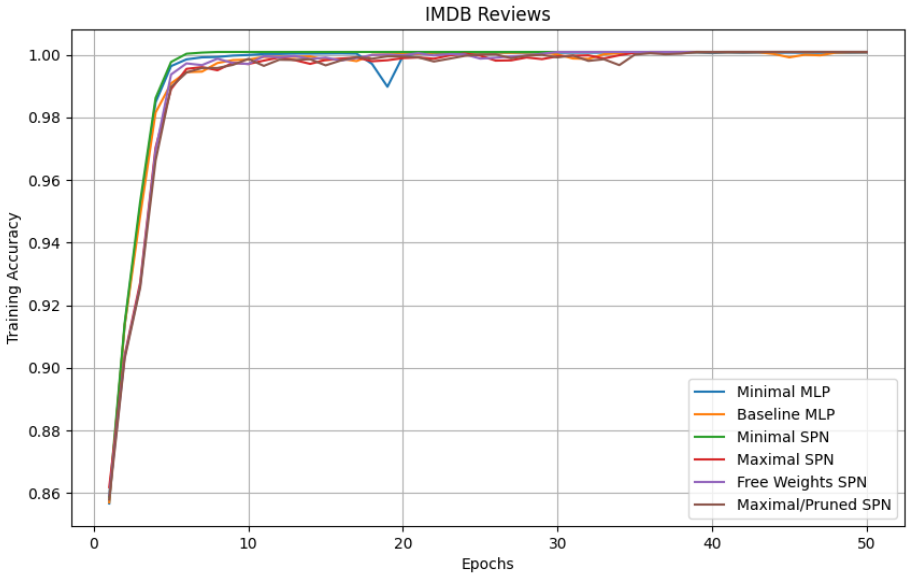

FIGURE 5.17: train_acc vs epochs curve for IMDB Reviews

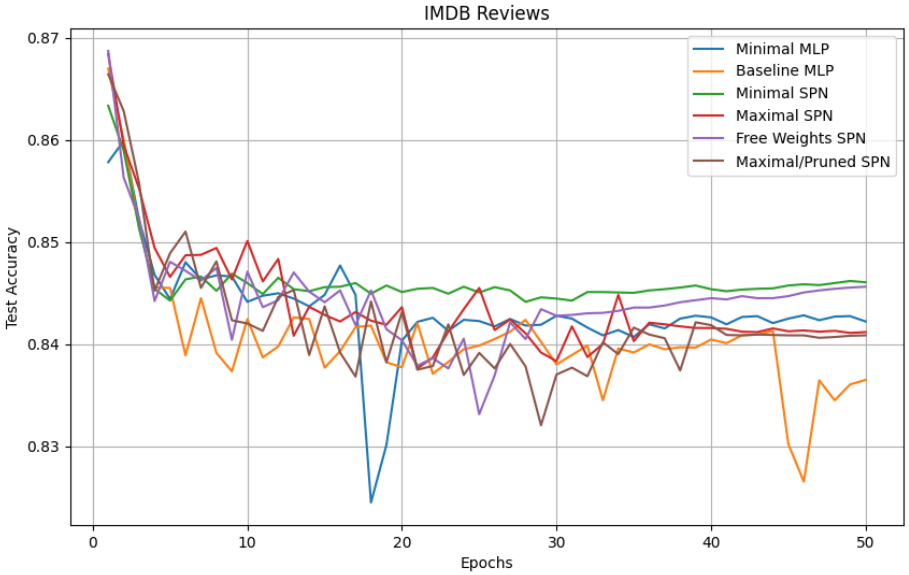

FIGURE 5.18: test_acc vs epochs curve for IMDB Reviews

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.

5.3.3 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

5.3.4 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

5.4 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in.

Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Chapter 6

Conclusion

6.1 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

6.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

6.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

6.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Bibliography

[1] *MNIST on Papers With Code*. https://paperswithcode.com/sota/image-classification-on-mnist. Accessed: 2025-07-13.