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 and Micaela Strömborg 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.

I would like to thank my friends Mutahar Aamir, Asad Imtiaz and Somoy Barua for their invaluable feedback during the initial stages of development for this thesis. A special thanks to Irum Muneer for making one of the graphics used in the thesis. I would also like to express my gratitude to Ali Umair, Mira Budenova and Khizar Bin Muzaffar who provided feedback on the writing process of the thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

MLP	**M**ulti-**L**ayer **P**erceptron
SPN	**S**arosh's **P**erceptron **N**etwork
ResNet	**R**esidual **N**etworks
CNN	**C**onvolutional **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks
ViT	**Vision T**ransformers
NAS	**N**eural **A**rchitecture **S**earch
LTH	**L**ottery **T**icket **H**ypothesis

Chapter 1

Introduction

The automated design of neural network architectures has emerged as a key frontier in modern machine learning, driven by the growing complexity of tasks and the scale of available data. Neural Architecture Search (NAS) [7] has enabled researchers to systematically explore vast design spaces, moving beyond manual trial-and-error to discover architectures that strike an optimal balance between performance and efficiency. Despite this progress, many neural network architectures, such as Multi-Layer Perceptrons (MLPs) [23], remain limited by conventional connectivity patterns that restrict information flow to simple, hierarchical pathways.

This thesis aims to challenge and expand this architectural paradigm. It introduces Sarosh's Perceptron Networks (SPNs), a novel approach that breaks free from the rigid layer-by-layer connectivity of traditional MLPs and allows neurons more freedom in forming cross-layer connection that lead to more complex architectures. By allowing for more flexible and expressive patterns of neuron connectivity, SPNs seek to unlock new levels of model capability and generalization. This work investigates whether such architectural freedom can yield meaningful improvements in performance and efficiency, and examines the implications for the future of neural network architecture design and the field of NAS.

1.1 Background and Motivation

Neural Architecture Search (NAS) [7] has emerged as a prominent research area in machine learning, aiming to automate the design process of neural network architectures. NAS systematically explores vast spaces of possible network configurations to discover architectures that optimally balance performance, computational cost, and efficiency for specific tasks. This exploration typically involves methods such as reinforcement learning, evolutionary algorithms, and gradient-based optimization, each strategically navigating a complex, multidimensional search space to find optimal architectural solutions.

At the foundational level of neural network architectures lies the perceptron [22], introduced by Frank Rosenblatt in 1958. As one of the earliest and most influential machine learning models, the perceptron serves as a linear binary classifier, inspired by biological neurons. However, while effective for linearly separable problems, perceptrons inherently struggle to capture complex, non-linear patterns.

To overcome this limitation, Multi-Layer Perceptrons (MLPs) [23] were developed, adding multiple interconnected layers of perceptrons to facilitate learning non-linear relationships. MLPs became capable of effectively training deep architectures through gradient-based optimization. Despite advancements leading to specialized architectures such as Convolutional Neural Networks (CNNs) [15] for image processing and Transformer-based Large Language Models (LLMs) [26] for

natural language tasks, MLPs continue to serve as a fundamental building block within neural network research.

NAS methodologies have been applied extensively to refine MLP architectures, systematically tuning hyperparameters such as neuron count, layer depth, and activation functions. However, traditional MLP architectures have typically constrained their search spaces to hierarchical layer-to-layer connectivity, inherently neglecting more flexible connectivity patterns such as non-sequential cross layer connectivity and internal connections between neurons in the same layer. While architectures like Residual Networks (ResNet) [11] have introduced skip connections to partially mitigate this limitation, the potential of unrestricted neuron connectivity within MLP architectures remains largely unexplored.

1.2 Problem Statement

The SPNs introduced in this thesis provide a novel approach designed to unlock the unexplored connectivity potential within traditional MLP search spaces. SPNs remove the conventional constraints of strict hierarchical connections, enabling more flexible interactions between neurons across the entire network. By facilitating direct neuron-to-neuron connections irrespective of layer boundaries, SPNs aim to enhance neural network expressiveness and improve model performance significantly.

Critically, SPNs seek to expand architectural expressivity without incurring substantial computational overhead or sacrificing training efficiency. This research systematically investigates whether leveraging the expanded search space provided by SPNs can achieve meaningful performance improvements compared to traditional MLPs. By thoroughly exploring the advantages offered by this approach, this thesis aims to provide new insights and methodologies that could significantly impact the design and optimization of neural network architectures.

1.3 Research Gap and Contributions

There is a significant gap in the existing literature regarding the optimization of neural network architectures, particularly concerning neuron connectivity. Traditional MLPs rely on rigid, fixed structures that inherently limit the interactions among neurons across different layers. Although ResNet [11] introduced skip connections to address some of these limitations, it still imposes restrictive conditions, such as requiring consistent dimensionality across residually connected layers or limiting connections primarily to adjacent layers.

The main contributions of this thesis are:

1. **Introduction of Sarosh's Perceptron Networks (SPNs):** A novel neural network architecture framework that removes the connectivity constraints inherent to traditional MLP designs.

2. **Empirical Evaluation:** A comprehensive comparison of SPNs with conventional MLPs to investigate whether enhanced neuron connectivity translates into improved model accuracy, training speed, and efficiency.

3. **Enhanced Training Efficiency:** An investigation into the potential of SPNs to achieve superior performance while mitigating the computational and memory costs typically associated with increasing complexity and parameters in neural network architectures.

1.4 Research Questions

RQ1. Does increasing connection density in Sarosh's Perceptron Networks (SPNs), while keeping the layer count and design identical to MLPs, lead to better model performance?

RQ2. Does removing connectivity restrictions in MLP architectures (allowing arbitrary neuron-to-neuron connections) enhance the network's learning capability or representational power?

RQ3. Can SPNs with fewer layers but higher connection density match or exceed the performance of deeper MLPs, thereby improving throughput efficiency and reducing training times?

RQ4. How do SPNs compare to MLPs in terms of training time, computational efficiency, memory usage, and predictive accuracy across a range of datasets and tasks?

RQ5. Does the increased architectural flexibility of SPNs improve their ability to generalize across diverse datasets and tasks compared to standard MLPs?

RQ6. After maximizing connections in SPNs, can pruning strategies maintain high predictive performance while significantly improving computational efficiency and training time?

1.5 Scope and Limitations

This research focuses on SPN's potential to enhance neural network performance through increased neuron connectivity. The primary scope of this thesis is the empirical evaluation of SPNs in comparison to traditional MLPs, with all experiments limited to supervised classification tasks.

Key limitations include:

1. **Computational Resources:**By design, SPNs introduce a significantly larger number of connections than standard MLPs, resulting in increased computational demands for both training and inference. This research primarily evaluates whether the potential improvements in model performance and expressiveness are sufficient to justify the additional computational cost. However, due to hardware constraints, some larger-scale configurations or extended hyperparameter sweeps may be beyond the feasible scope of experimentation.

2. **Task and Domain Specificity:** The experiments and comparisons in this study are restricted to a selection of well-known benchmark datasets for classification. As such, the observed results may be influenced by the characteristics of these datasets, and the findings may not fully generalize to other domains (such as regression, reinforcement learning, or unsupervised tasks) or to highly complex, real-world data distributions.

3. **Scope of Architectural Exploration:** The present study is limited to comparisons between SPNs and traditional MLPs. It does not extend to specialized neural network architectures such as CNNs, ResNets, Recurrent Neural Networks (RNNs) [16], Vision Transformers (ViTs) [6], or LLMs. As a result, the boundaries of SPN expressiveness and their potential for integration with these

advanced architectures remain unexplored. Further research will be required to evaluate the applicability and benefits of SPNs in the context of these specialized models.

1.6 Thesis Structure and Overview

The remainder of this thesis is organized as follows:

1. Chapter 2 presents a comprehensive literature review, covering traditional Multi-Layer Perceptrons (MLPs), neural architecture search (NAS), ResNets and pruning methods and the Lottery Ticket Hypothesis.
2. Chapter 3 details the methodology used in this research, including the design and implementation of SPNs.
3. Chapter 4 describes the the experimental setup, dataset selection, model selection, training procedures, and evaluation metrics.
4. Chapter 5 presents the results and analysis of empirical comparisons between SPNs and traditional MLPs across multiple benchmark classification datasets.
5. Chapter 6 concludes the thesis by summarizing the key findings, and proposing directions for future research in neural architecture optimization and SPN applications.

Chapter 2

Literature Review

While much of recent deep learning research has focused on developing novel architectures and specialized modules, there has been comparatively little investigation into the role of dense and flexible connectivity within neural networks. Existing work often explores new types of connections or processing blocks, such as attention mechanisms [26], MLP Mixers [25], and Convolutional blocks [15], but rarely addresses the fundamental question of how increasing connection density itself could impact learning capacity and model performance.

This chapter begins with an overview of the field of Neural Architecture Search (NAS), discussing its objectives, prevalent techniques, and its influence on neural network optimization. The review then covers the classical MLP framework, pruning methods—including the influential Lottery Ticket Hypothesis—and other architectural innovations relevant to the development and motivation of Sarosh's Perceptron Networks (SPNs). Together, these topics provide the theoretical and practical context for the proposed research and its contributions.

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) [7]. NAS has significantly contributed to the advancement of deep learning by automating architecture design, traditionally a manually intensive process.

At its core, NAS operates by defining a *search space*, a set of possible neural network components and connections from which candidate architectures can be constructed. The effectiveness of NAS heavily depends on how this search space is formulated, as it determines the diversity and expressiveness of potential solutions. To explore this search space, NAS utilizes various *search strategies* to systematically generate, evaluate, and refine architectures based on their performance as shown in Figure 2.1. By iteratively searching and evaluating within this space, NAS aims to efficiently identify high-performing architectures that meet specific accuracy and efficiency criteria [27, 7].

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** approaches frame architecture search as a sequential decision-making problem. An RL agent, often implemented as a

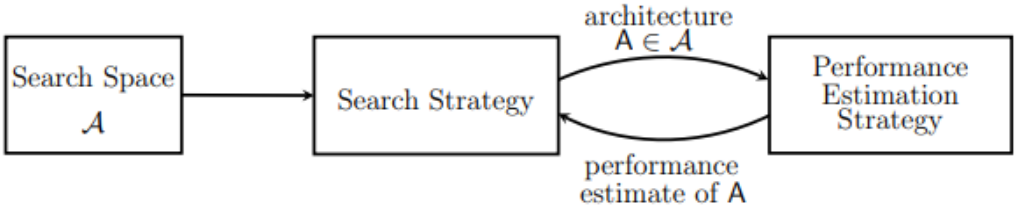

FIGURE 2.1: An abstract illustration of NAS [7] Search Methods.

recurrent neural network (RNN) controller, generates candidate architectures and receives reward signals based on their performance after training. The controller's policy is updated to favor architectures yielding higher rewards. The seminal work by Zoph and Le [31] introduced this paradigm, achieving impressive results on image classification benchmarks. Further extensions have incorporated advanced RL algorithms such as Proximal Policy Optimization (PPO) and Q-learning [2, 20]. While RL-based NAS provides great flexibility and search space exploration, it is computationally expensive due to the need to train and evaluate many candidate architectures.

- **Evolutionary Algorithm-based NAS** methods draw inspiration from biological evolution, maintaining a population of candidate architectures that evolve over generations through operations such as mutation, crossover, and selection. Regularized Evolution [21] demonstrated that EA-based NAS can yield high-performing and diverse architectures, even outperforming some RL-based methods in certain settings. Techniques like Aging Evolution and genetic programming [18, 28] have been used to maintain population diversity and avoid premature convergence. EA-based approaches are known for robust global search capabilities, but often require significant computational resources.

- **Gradient-based NAS** methods, such as DARTS: Differentiable Architecture Search [17], convert the discrete NAS problem into a continuous one, enabling efficient search via gradient descent. Architectures are parameterized in a way that allows gradients to flow through architectural choices, greatly reducing the search cost compared to RL and EA approaches. DARTS and its successors [3, 4] are popular for their scalability and efficiency, though they can be sensitive to hyperparameter choices and initialization, sometimes resulting in suboptimal or unstable architectures [30].

Recent advances also include hybrid NAS approaches that combine multiple techniques (e.g., RL with gradient-based fine-tuning), meta-learning strategies, and hardware-aware NAS for deployment on edge devices [7, 27, 24]. Furthermore, open-source frameworks such as Auto-Keras [14], ENAS [20], and NAS-Bench-101 [29] have made NAS research more accessible and reproducible.

In the context of NAS, this thesis focuses on expanding the architectural search space for neural networks. As discussed in Chapter 3, SPNs extend traditional MLPs by introducing additional connections, thereby increasing the diversity and flexibility of possible network topologies. By enabling more granular and unconstrained connectivity patterns, SPNs broaden the set of architectures that NAS can explore. This expanded search space provides NAS algorithms with greater expressive power, offering the potential to discover more effective and high-performing models that may not be achievable within the constraints of standard MLP architectures.

TABLE 2.1: Comparison of NAS Techniques

Technique	Advantages	Disadvantages
RL-based	Flexible search, can handle large, discrete spaces	Very computationally expensive; requires training many architectures
EA-based	Robust global search, maintains diversity	Extensive evaluations required; can be slow to converge
Gradient-based	Highly efficient, scalable, low search cost	Sensitive to initialization, may get stuck in local minima, can suffer from instability
Hybrid/Meta-learning	Combines strengths of various approaches, can leverage prior knowledge	Increased implementation complexity; may inherit disadvantages of base methods

For a comprehensive survey of NAS techniques, see [7, 27]. For practical implementations, refer to frameworks and benchmarks in [14, 20, 29].

2.2 Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs) are classical feedforward artificial neural networks composed of multiple layers of nodes (neurons). Each neuron employs a non-linear activation function enabling the network to learn complex patterns through supervised learning techniques, such as backpropagation [23]. MLPs have traditionally served as baseline models due to their simplicity and general applicability across diverse tasks, including classification and regression problems.

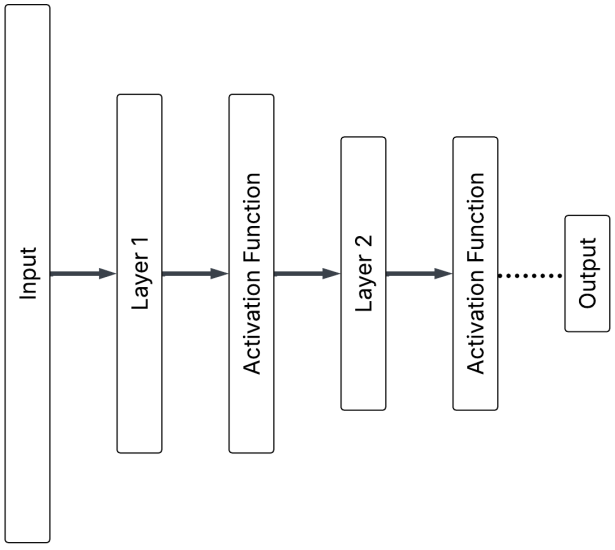

FIGURE 2.2: An example MLP Architecture.

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 Residual Networks (ResNets)

Residual Networks (ResNets), introduced by He et al. [11], represent a significant breakthrough in deep learning by enabling the effective training of very deep neural networks. ResNets address the vanishing gradient problem, a phenomenon where gradients become exceedingly small as they are backpropagated through many layers, hindering learning in deep architectures, by introducing residual connections. A residual connection, or skip connection, allows the input of a layer (or a set of layers) to be added directly to the output, thereby forming a shortcut path for the gradient during backpropagation.

Formally, for a given input $\mathbf{x}$ and a residual function $F(\mathbf{x})$, the output of a residual block is $\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$. Importantly, these connections do not introduce new trainable weights for the shortcut path; instead, they simply sum the input and the output of the residual block, providing a form of informed augmentation rather than increasing connectivity in the traditional sense. For residual addition to be valid, the dimensions of the input and output must match; when they do not, projection shortcuts with 1×1 convolutions are used to adjust dimensions.

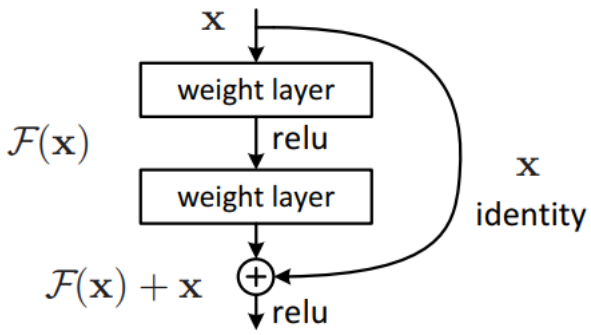

FIGURE 2.3: Example of a residual block in ResNets [11].

Furthermore, residual connections are typically local, linking only layers that are one or two steps apart, while more distant neurons remain disconnected. Thus, although ResNets represent a major advance in making deep learning practical and effective, they do not fundamentally expand the overall connectivity space of the network, but rather improve information and gradient flow through carefully structured local augmentations.

2.4 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 [10]. 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 [10].

Structured pruning removes entire channels or layers, maintaining regular structure beneficial for hardware acceleration [12].

Learning-based pruning, such as iterative magnitude pruning, incrementally prunes the network while retraining it to recover accuracy [10].

2.5 Lottery Ticket Hypothesis

The Lottery Ticket Hypothesis, proposed by Frankle and Carbin, posits that dense, randomly-initialized neural networks contain smaller subnetworks (winning tickets often as small as less than 10% the size of the original network) which, when trained in isolation, can match or exceed the test accuracy of the original network [9].

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.

2.6 Summary

This literature review highlights the advancements in neural architecture optimization through NAS, the foundational structure and limitations of MLPs and ResNets, 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 this thesis.

Chapter 3

Methodology

3.1 Sarosh's Perceptron Networks (SPNs)

This chapter introduces the theoretical framework for Sarosh's Perceptron Networks (SPNs), 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 (since cycles would cause deadlocks). 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 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. Figure 3.1 illustrates an arbitrary eample of such a network.

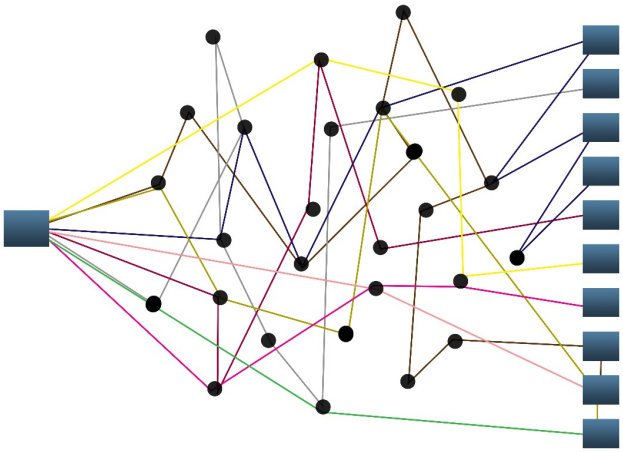

FIGURE 3.1: An Example SPN Network.

3.1.1 Connection vs Neuron

Before defining connectivity, it is important to clarify what is meant by a *connection*. As illustrated in Figure 3.2, a perceptron (used interchangeably with "neuron" throughout this thesis) receives multiple inputs, each of which is assigned a learnable weight. The perceptron computes a weighted sum of its inputs, adds a learnable bias term, and passes the result through an activation function. In this thesis, a **connection** is defined as any instance where an input transmits its value to a given perceptron. Each connection introduces a new learnable weight parameter for that

perceptron, thereby increasing the overall parameter count of the network by one per connection.

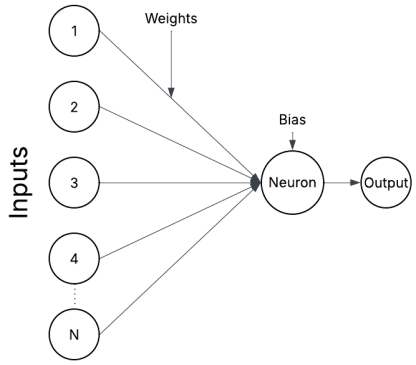

FIGURE 3.2: Architecture of a single Perceptron.

Accordingly, a Multi-Layer Perceptron is defined as a network composed of multiple layers of perceptrons. Thus, SPNs focuses on increasing the connection density of MLP models, while retaining the original number of neurons.

3.1.2 Motivation for SPNs

As illustrated in Figure 3.3, a typical MLP architecture features sequential connectivity: every neuron in a given layer is fully connected to all neurons in the subsequent layer. However, this conventional design omits two important types of connections.

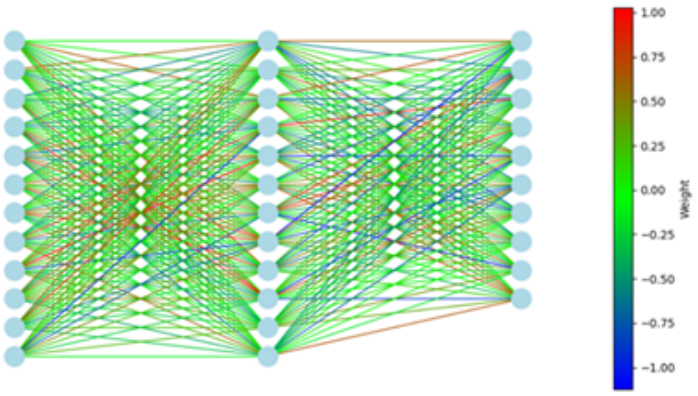

FIGURE 3.3: Low level illustration of an MLP.

1. **Cross-layer connections** (shown in Figure 3.4): These connections would allow a layer to link to any preceding layer, not just the one immediately behind it.

2. **Intra-layer connections** (depicted in Figure 3.5): These would enable neurons within the same layer to connect to each other, a feature absent in standard MLPs.

Alternatively, the weights of an MLP can be represented as a single, large weight matrix, where the x-axis corresponds to the input features and the y-axis to the neurons containing these weights, as illustrated in Figure 3.6. In the illustration, every layer is assumed to compose of only two neurons.

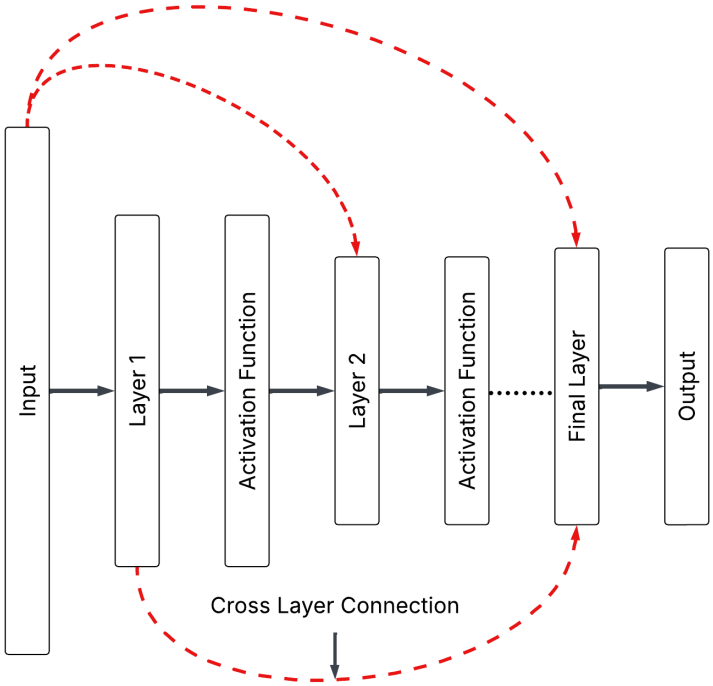

FIGURE 3.4: Missing cross layer connections in MLP.

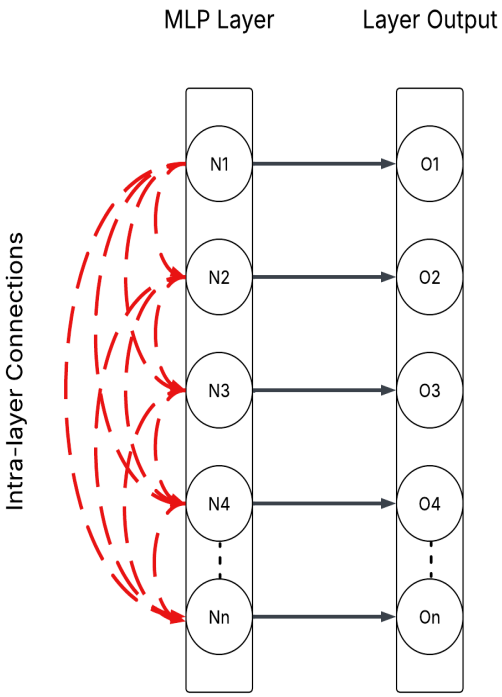

FIGURE 3.5: Missing Intra-layer connections in a single MLP layer.

In the representation in Figure 3.6, the nonzero weights appear in diagonal arragned blocks, forming a staircase-like pattern that is separated along both axes. On the right side of the boxes, there is a theoretical boundary of connections. If any connections cross this boundary, it would form a cycle within the network where a neuron would be connected to itself (dependent on its own output). Thus, this

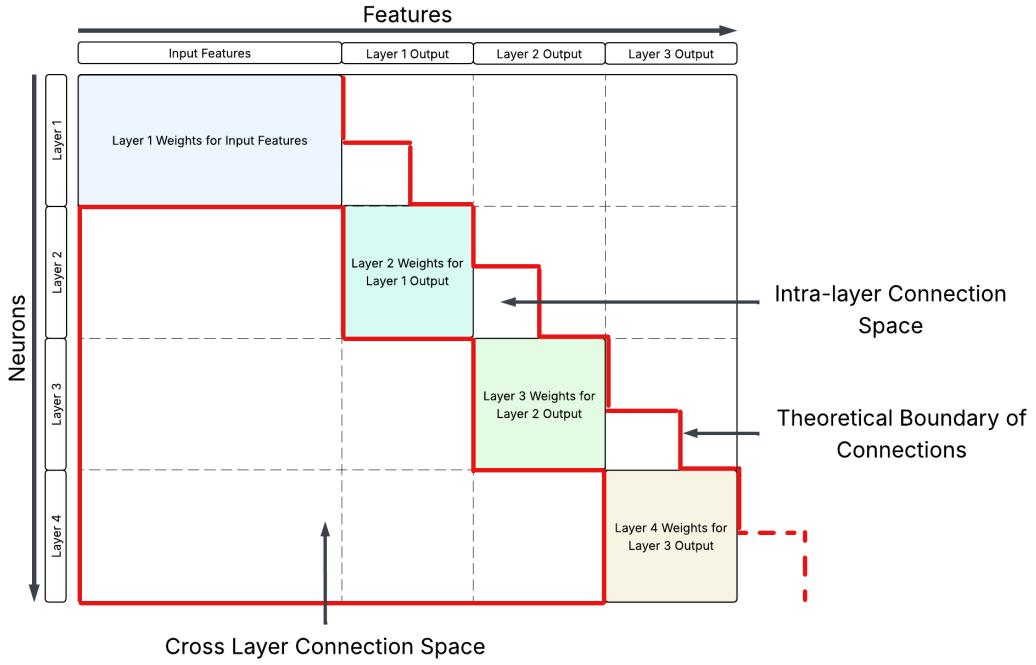

FIGURE 3.6: Cross Layer Connection Space and Intra-layer connection space with the theoretical connection boundary in the combined weight matrix of a traditiona MLP, assuming each layer has two neurons.

boundary can never be crossed.

This visualization also makes it evident that both the left and right regions outside the blocks are completely empty. The empty region on the left corresponds to the absence of cross-layer connections, while the empty region on the right reflects the lack of intra-layer connections within the same layer.

Examining the weight matrix further, we notice that the right side of the matrix effectively determines the model's number of layers. If we begin populating this region by introducing intra-layer connections, we break the independence that typically defines neurons within the same layer. Adding this dependency thus splits the original layer into multiple distinct layers. Intuitively, this aligns with the conventional understanding that neurons within a layer should not interact directly.

Adding weights to either the left or right of the existent MLP blocks transforms the classic MLP structure into a Sarosh's Perceptron Network.

3.1.3 Free Weight SPNs

In the MLP weight matrix shown in Figure 3.3, adding connections to the left side of the matrix does not increase the model's layer count. This is crucial as the number of layers in a model determines how many times the input is processed before producing an output, and thus directly relates to the model's throughput. Therefore, incorporating weights on the left side of the matrix maximizes the possible connections within the given layer structure without impacting throughput. Therefore, these additional connections are termed "Free Weights", as they can be added without any cost to the model's throughput.

Incorporating free weights into traditional MLPs is analogous to concatenating the output of a layer with its input before passing it to the next layer. This approach

increases the space complexity of the MLP by $\mathcal{O}(l^2)$, while maintaining a time complexity of $\mathcal{O}(l)$, where l is the number of layers. A key advantage of this method is that subsequent layers can learn from both the original input and features extracted by previous layers, allowing the network to capture more complex and nuanced patterns.

An MLP enhanced with Free Weights is then referred to as a Free Weights SPN.

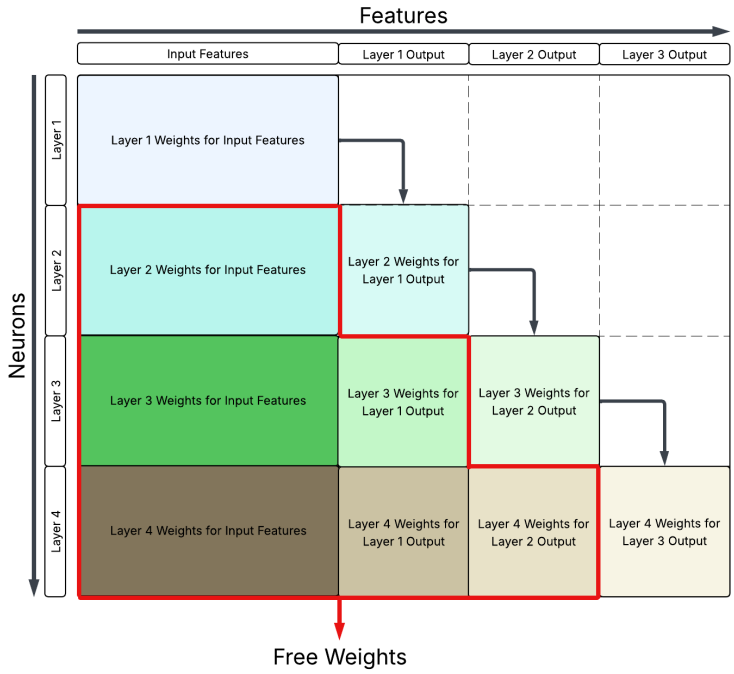

FIGURE 3.7: Illustration of a traditional MLP with Free Weights.

3.1.4 Maximal SPNs

Additionally, by adding weights to the right side of the diagonal blocks, the weight matrix becomes a fully filled staircase pattern, with each "step" corresponding to a single neuron. This configuration not only maximizes the number of possible connections but also increases the layer count to match the total number of neurons, thereby minimizing the model's throughput. Such a model, which achieves both maximal connectivity and maximal depth, is referred to as a Maximal SPN.

- Time complexity of Maximal SPNs: $\mathcal{O}(n)$, since every neuron becomes its own layer
- Space complexity: Using I as the number of features in the input data and n as the number of neurons.

$$
\begin{aligned}
&= I + (I+1) + (I+2) + \cdots + (I+n-1) \\
&= I \cdot (1 + 2 + \cdots + (n-1)) \\
&= I \cdot \frac{n(n-1)}{2} \\
&= \mathcal{O}(I \cdot n^2)
\end{aligned}
$$

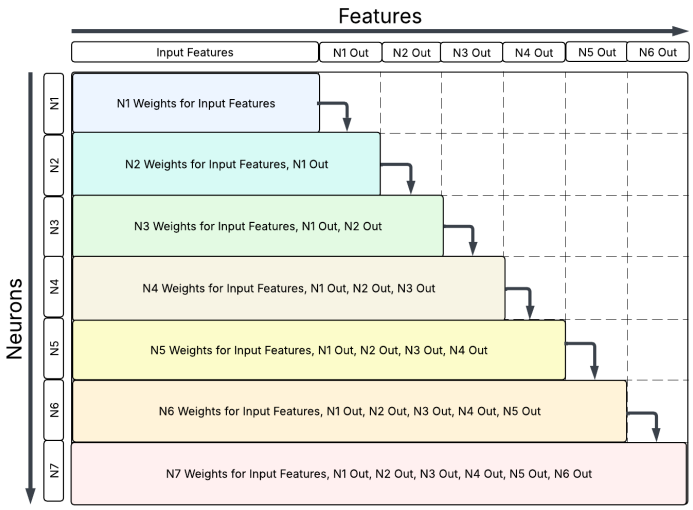

FIGURE 3.8: Illustration of a Maximal SPN.

3.1.5 Pruned Maximal SPNs

Since maximal SPNs can have an excessive number of layers and consequently, poor throughput, one potential solution is **pruning**. The Lottery Ticket Hypothesis posits that over 90% of weights in a network can be removed while still achieving performance comparable to the original network, suggesting the existence of a subnetwork within the initial network that is equally trainable [9].

Applying this principle, if we prune a maximal SPN, zeroing out selected weights on the rightmost edges of the staircase weight matrix, we effectively reduce dependencies between certain neurons as illustrated in Figure 3.9. This reduction allows for the merging of independent neurons into shared layers, thus reintroducing parallelism to the network while maintaining its high expressiveness and capacity for learning complex data patterns.

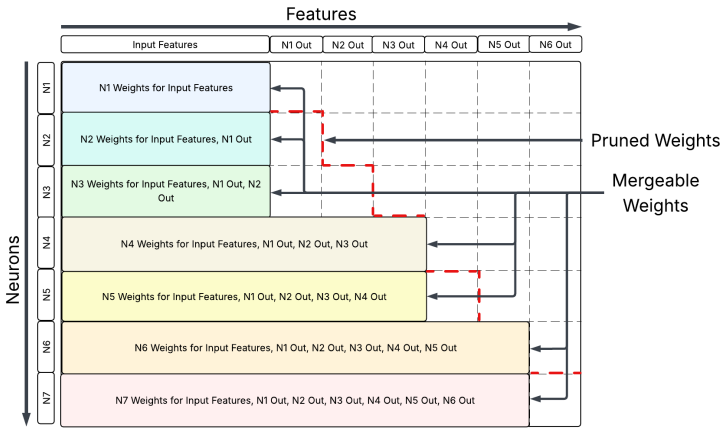

FIGURE 3.9: Pruning a Maximal SPN.

The pruning strategy adopted in the current SPN framework is inspired by the Lottery Ticket Hypothesis and follows these steps:

1. Train the original network to completion.

2. Prune a fraction of the weights with the lowest magnitudes.
3. Reset the remaining weights to their initial values.
4. Repeat this process iteratively for several cycles.

This iterative process is repeated for a user-defined number of rounds, with the optimal number typically determined through empirical testing or validation. In each round, a fraction of the model's lowest-magnitude weights are pruned, and the initial weights yielding the best performance are retained. This continues until up to 99% of the weights have been removed. At the end of the process, the best-performing set of initial weights is used to construct the final pruned model.

At the conclusion of this procedure, neurons that have become independent as a result of pruning are merged to reduce the overall layer count and improve throughput. It is important to note that merging is performed only after the entire pruning process has finished. Preliminary experiments revealed that merging neurons during pruning can alter the network's architecture in ways that negatively affect subsequent pruning steps, making the overall process less effective than waiting until pruning is complete.

3.1.6 Minimal SPNs

At the other end of the spectrum, if our goal is to minimize the number of layers while maximizing connectivity, we arrive at the concept of minimal SPNs. This involves organizing the network with as few layers as possible and enriching them with free weights.

Theoretically, the most minimal architecture would be a single perceptron, or for multiclass classification, a single layer with as many neurons as there are output classes (o). However, for this analysis, we assume a fixed total number of neurons (n) and seek to arrange them into layers to minimize throughput. This leads to two possible scenarios:

1. When the total number of neurons equals the output size ($n = o$): Only a single output layer of size o is possible, with no opportunity for cross-layer connections.
2. When the total number of neurons exceeds the output size ($n > o$): The network can be organized into two layers: an output layer of size o and a hidden layer with the remaining $n - o$ neurons. This configuration yields a minimal MLP, as illustrated in Figure 3.10. By introducing free weights, we transform it into a minimal SPN, depicted in Figure 3.11.

- Time complexity: $\mathcal{O}(1)$, since there can always be a maximum of two layers, the time complexity is constant.
- Space complexity: Using I as the number of features in the input data and n as the number of neurons.

$$
\begin{aligned}
&= I \cdot (n) + n \cdot o \\
&= \mathcal{O}(I \cdot n)
\end{aligned}
$$

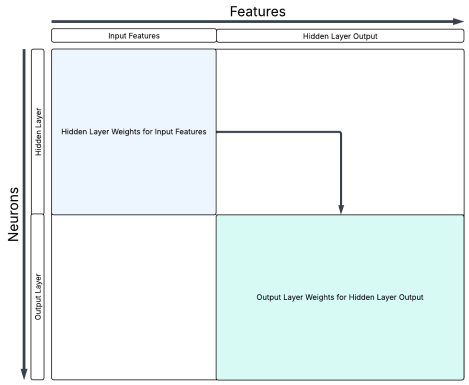

FIGURE 3.10: Minimal MLP.

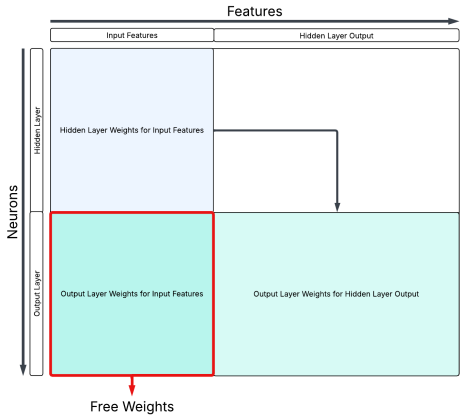

FIGURE 3.11: Minimal SPN.

3.2 Implementation Approach

In practice, there are three potential ways of designing the forward-propagation and backpropagation algorithms for SPNs. For all these approaches, the time and space complexity of implementing Maximal SPNs will be considered as they represent the worst case scenario.

3.2.1 Object-based Representation with Partial Inputs

One potential method to achieve the SPN framework is to have neurons, as independent objects, 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.

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: $\mathcal{O}(n^2)$.
- Space complexity: $\mathcal{O}(I \cdot n^2)$.

3.2.2 Sequential Representation

In this approach, we describe the SPN using a list of weights, where every element in the list corresponds to the weights of a single neuron, exactly how it was illustrated in Figure 3.8.

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 neuron in the list and proceeding to the last. For each step, 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: $\mathcal{O}(n)$.
- Space complexity: $\mathcal{O}(I \cdot n^2)$.

3.2.3 Block-based Representation

When considering the weight matrix of a Maximal SPN, we can draw vertical lines at the edge of each neuron's weight vector, forming vertical blocks within the matrix. The largest block, which is the first block, contains the weights corresponding to the input features, and subsequent blocks contain the weights for the output of each neuron as showin in Figure 3.12.

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: $\mathcal{O}(n)$.
- Space complexity: $\mathcal{O}(I \cdot n^2)$.

3.2.4 Summary

The performance difference between the sequential and block-based approaches is minimal, as both exhibit the same overall time complexity. Their relative efficiency primarily depends on the underlying hardware. The block-based approach frontloads the computational burden by handling the most demanding calculations first, while the sequential approach distributes the workload more evenly across all steps. Depending on the hardware and the specific model architecture, one approach may be preferable over the other.

In this thesis, the sequential method was chosen for its balanced workload distribution; with larger models, the block-based approach could potentially overwhelm hardware resources at the outset, resulting in slower performance. Ultimately, a hybrid strategy that combines elements of both approaches may offer the greatest efficiency.

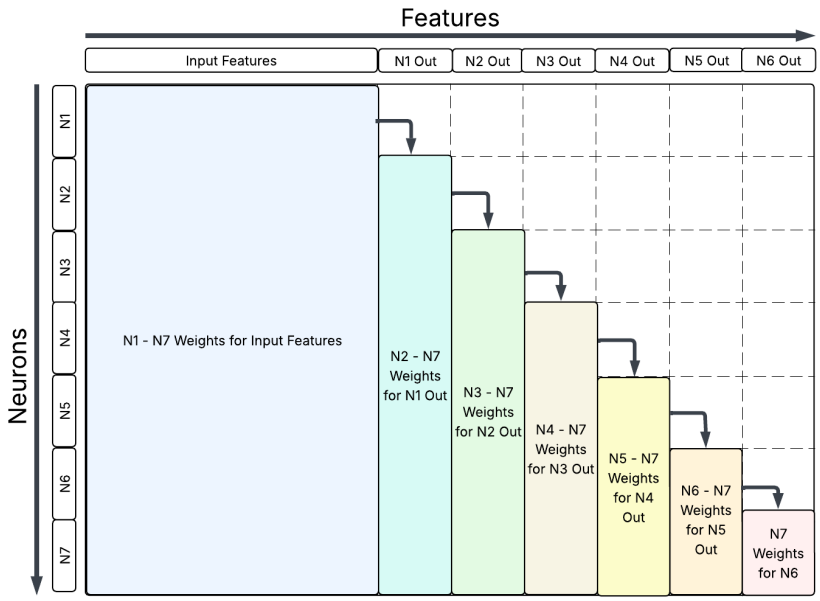

FIGURE 3.12: Illustration of a block based Maximal SPN.

TABLE 3.1: Comparison of time and space complexities for the implementation approaches.

Approach	**Time Complexity**	**Space Complexity**
Object Based	$\mathcal{O}(n^2)$	$\mathcal{O}(I \cdot n^2)$
Sequential	$\mathcal{O}(n)$	$\mathcal{O}(I \cdot n^2)$
Block Based	$\mathcal{O}(n)$	$\mathcal{O}(I \cdot n^2)$

Overall, this thesis aims to investigate whether introducing cross-layer connections, intra-layer connections, or both can enable MLPs to achieve a deeper understanding of the data and/or improve their computational efficiency. The experimental setup and the corresponding results for that investigation are presented in the subsequent chapters.

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.

4.1.1 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

4.1.2 Datasets

To address RQ5., 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 multiclass classification tasks.

4.1.2.1 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.

4.1.2.2 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.

4.1.2.3 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.

4.1.2.4 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 multiclass classification. This selection allows for a comprehensive evaluation of SPNs and MLPs across binary and multiclass 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

4.1.3 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 RQ1., enabling a direct examination of the effects of increased connection density.

3. **Minimal SPN**: To address RQ3., 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 RQ1..

5. **Maximal SPN**: are designed to maximize all connections between perceptrons for a given number of neurons. These models aim to address RQ2. 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 RQ6. 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 training 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.
$$\text{Training Efficiency} = \frac{\text{Best Test Accuracy}}{\text{Time to Best Test Accuracy}}$$

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.
$$\text{Throughput Efficiency} = \frac{\text{Best Test Accuracy}}{\text{Mean Epoch Time}}$$

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 Maximal SPN and the mean epoch time of the Pruned SPN. It reflects how fast the Pruned SPN is compared to the Maximal SPN.

$$\text{Pruning Effectiveness} = \frac{\text{Mean Epoch Time of Max SPN}}{\text{Mean Epoch Time of Pruned SPN}}$$

Together, these metrics enable us to address RQ4. 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

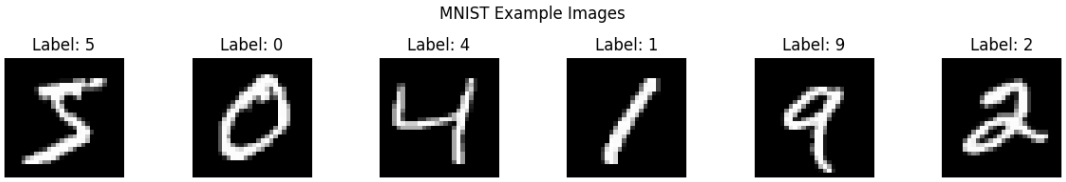

FIGURE 5.1: Example Images in MNIST

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 [19], 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.

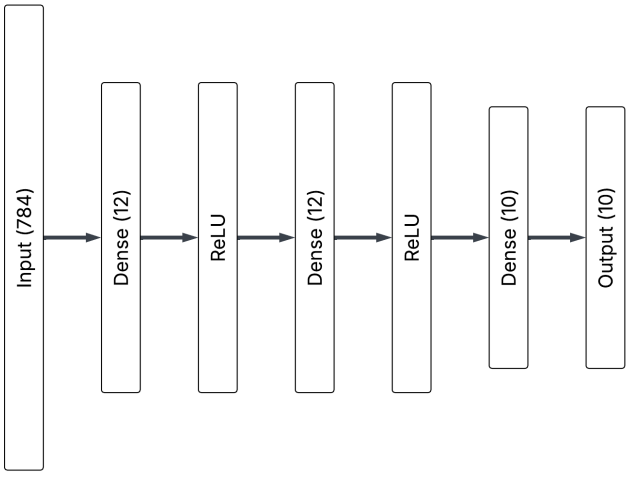

FIGURE 5.2: base_mlp architecture for MNIST

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.

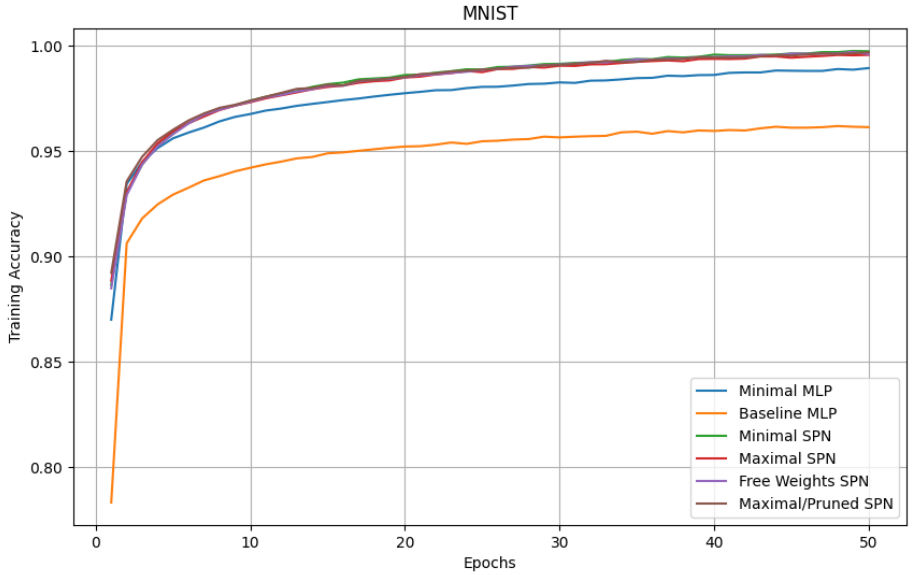

FIGURE 5.3: train_acc vs epochs curve for MNIST

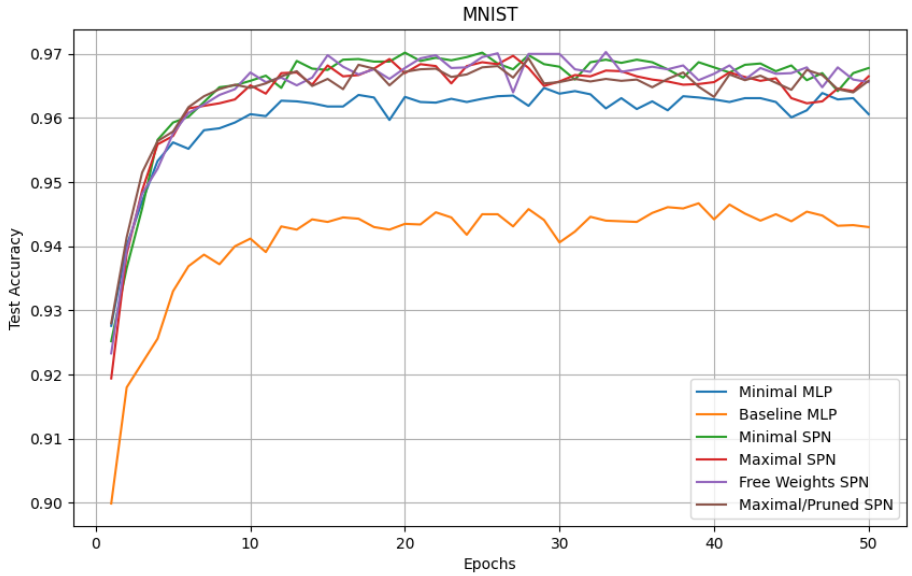

FIGURE 5.4: test_acc vs epochs curve for MNIST

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 [5], 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.

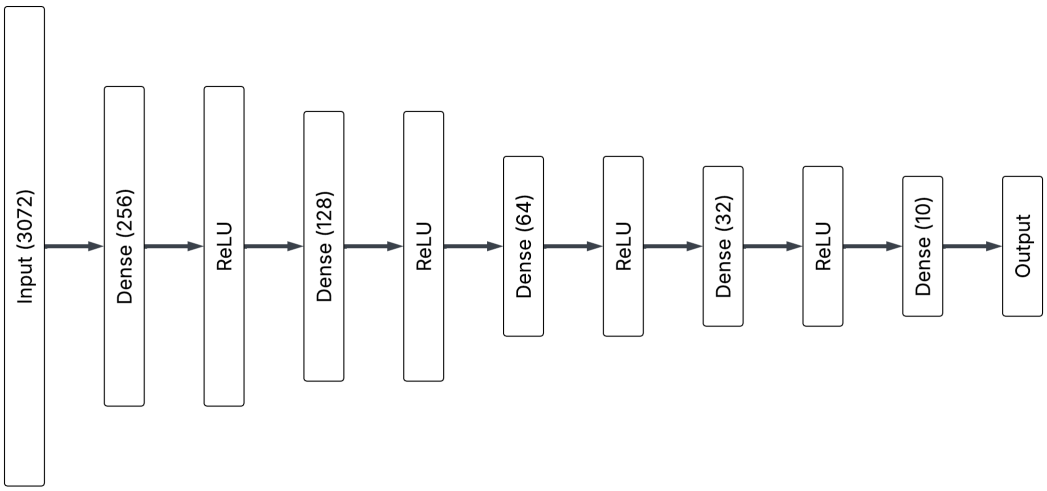

FIGURE 5.6: base_mlp architecture for CIFAR-10

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.

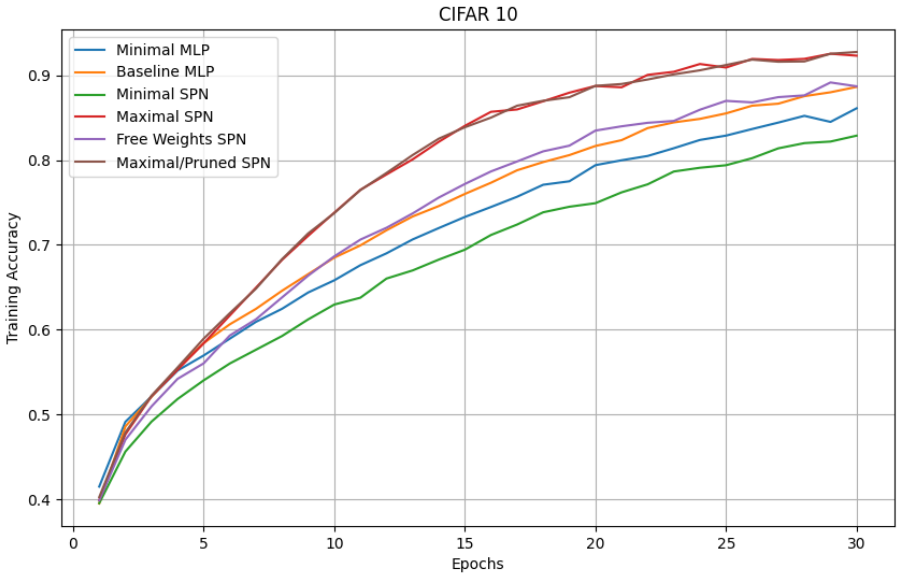

FIGURE 5.7: train_acc vs epochs curve for CIFAR-10

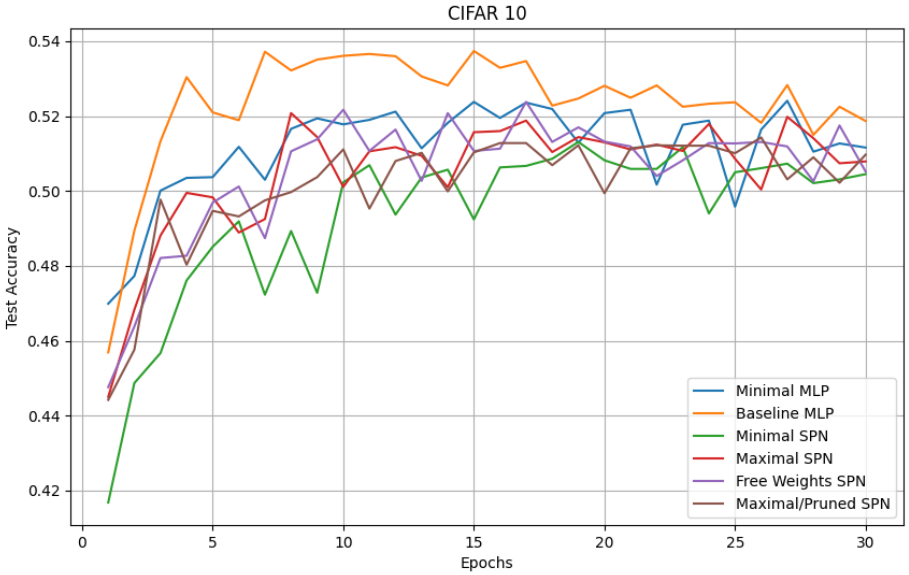

FIGURE 5.8: test_acc vs epochs curve for CIFAR-10

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.

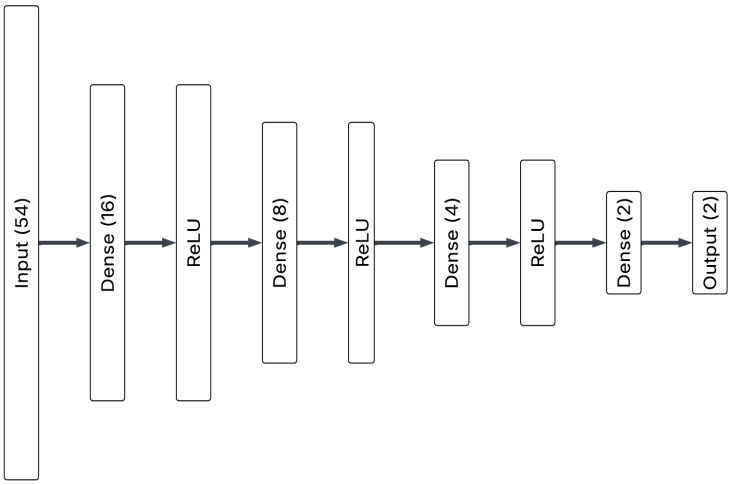

FIGURE 5.9: base_mlp architecture for Titanic

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.

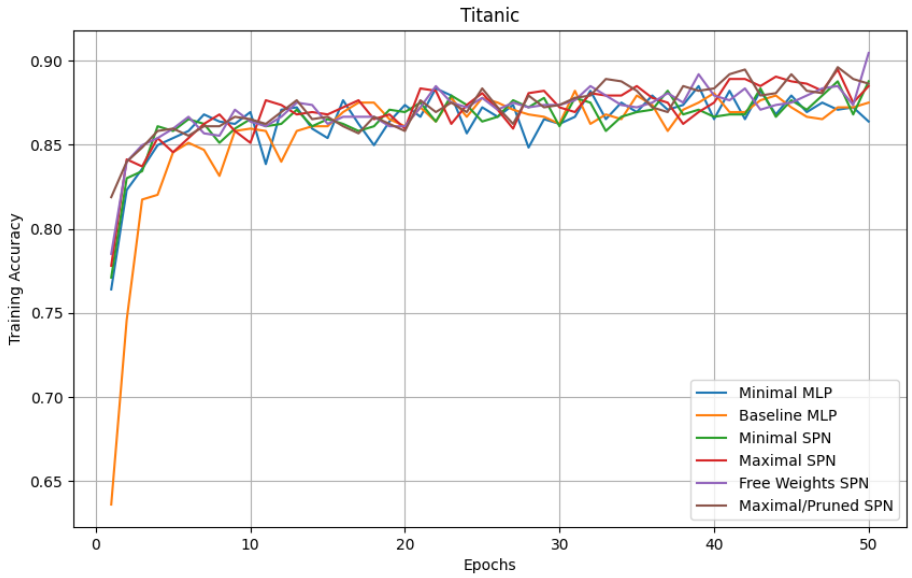

FIGURE 5.10: train_acc vs epochs curve for Titanic

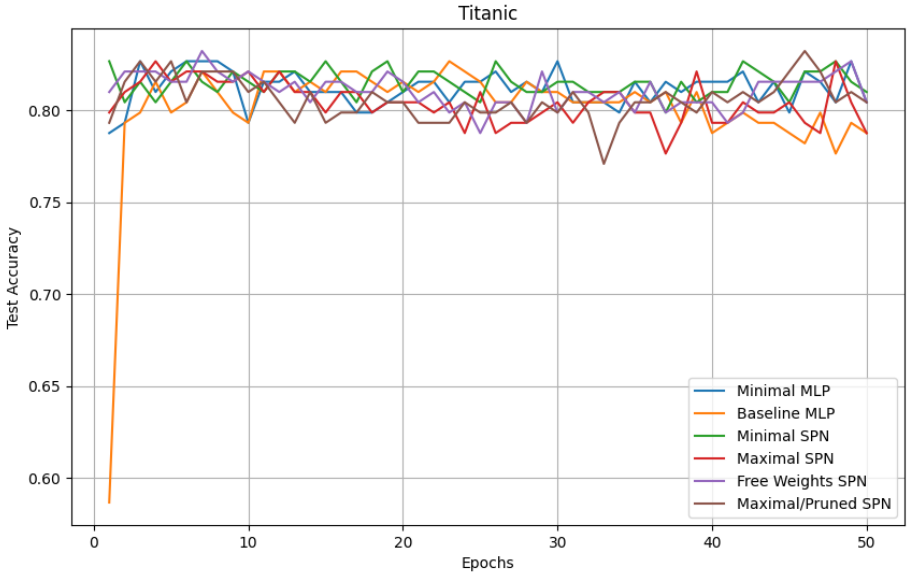

FIGURE 5.11: test_acc vs epochs curve for Titanic

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 [8]. And since the test accuracy was closer to 80%, a three-layer MLP of substantial size was selected for further experiments as there was significant room for observing improvements, if any were made by using SPN techniques.

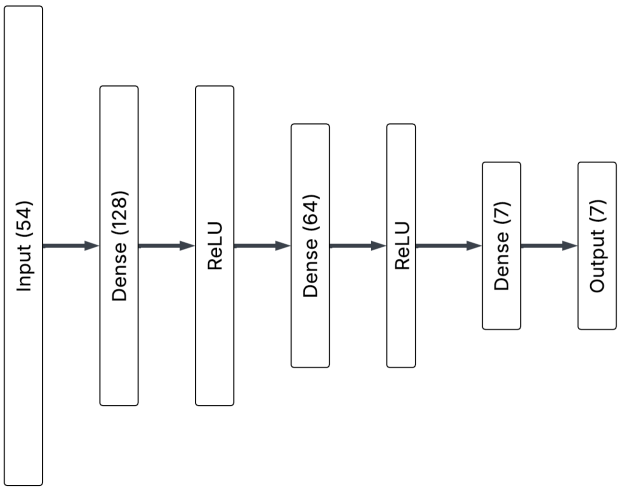

FIGURE 5.12: base_mlp architecture for Covertype

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.

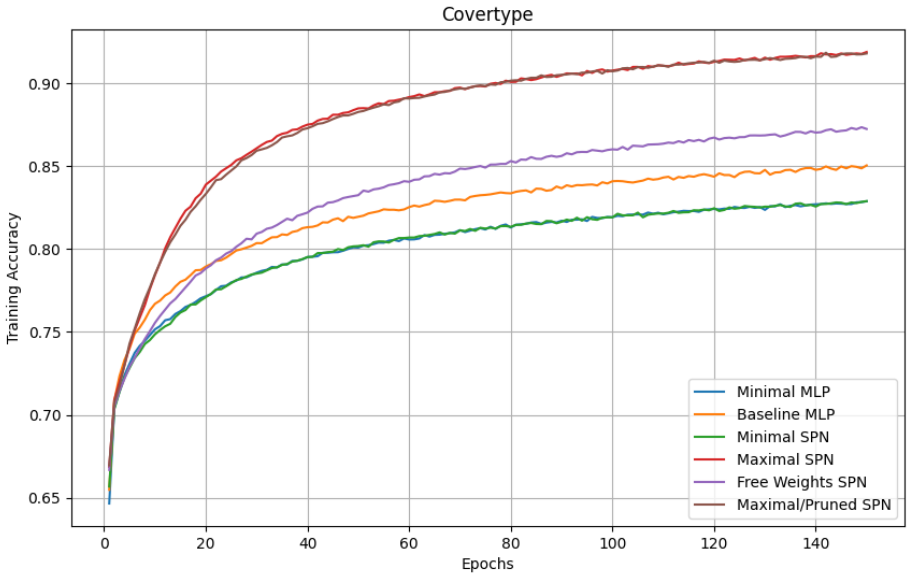

FIGURE 5.13: train_acc vs epochs curve for Covertype

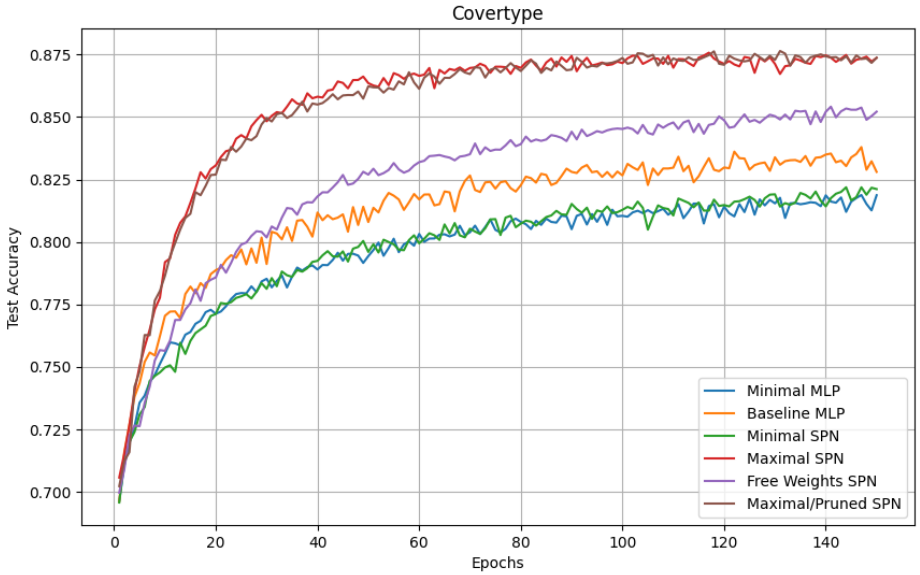

FIGURE 5.14: test_acc vs epochs curve for Covertype

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 [8]. 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.

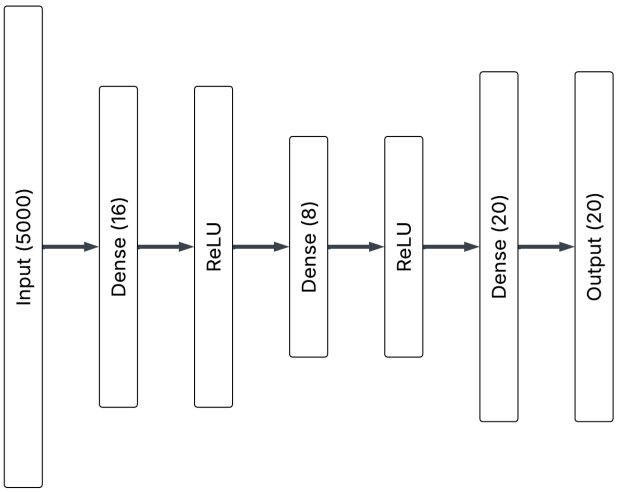

FIGURE 5.15: base_mlp architecture for Newsgroups 20

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.

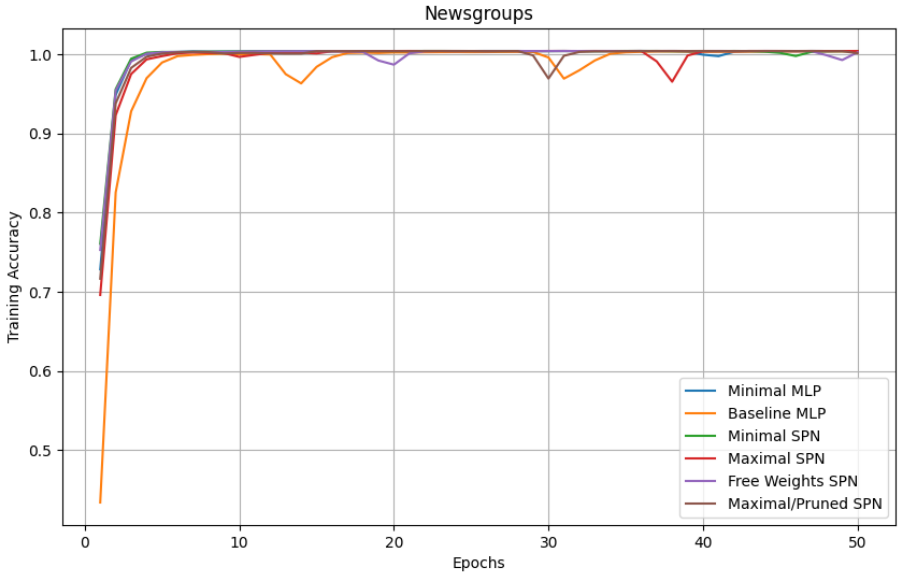

FIGURE 5.16: train_acc vs epochs curve for Newsgroups 20

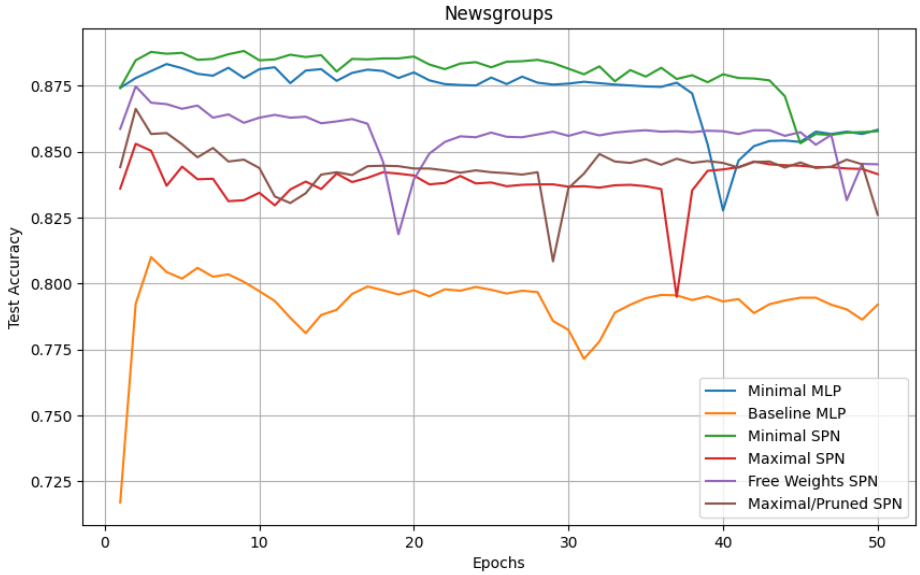

FIGURE 5.17: test_acc vs epochs curve for Newsgroups 20

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.

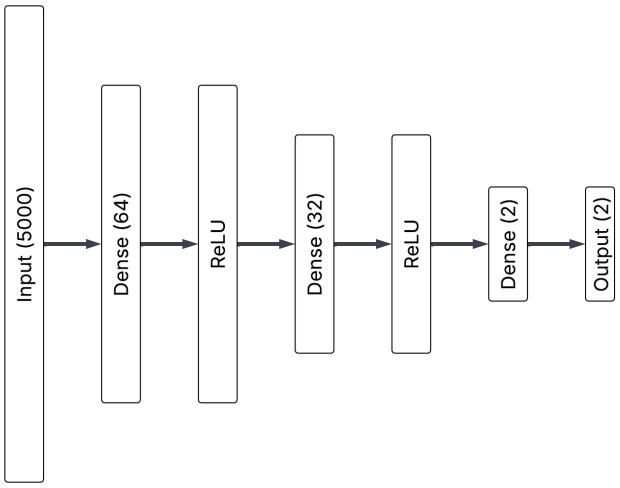

FIGURE 5.18: base_mlp architecture for IMDb Reviews

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.

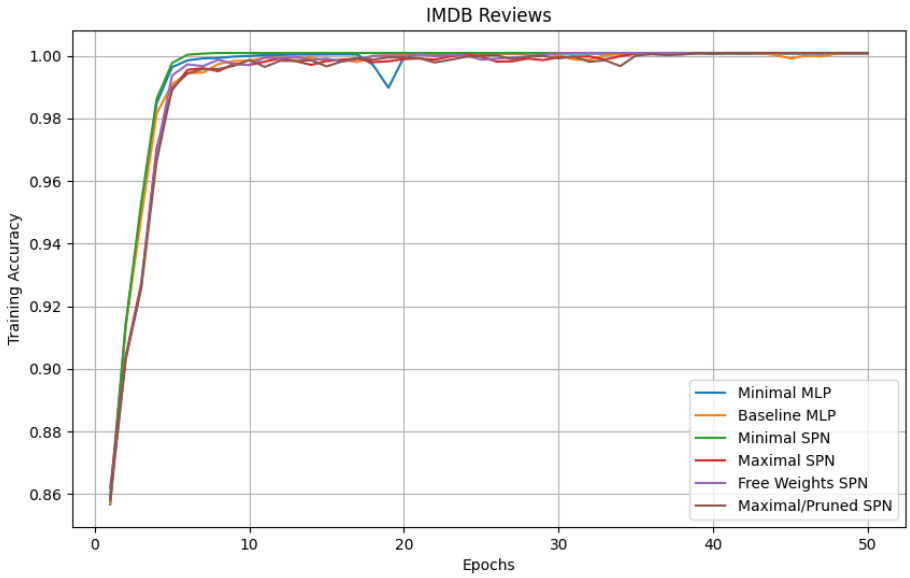

FIGURE 5.19: train_acc vs epochs curve for IMDb Reviews

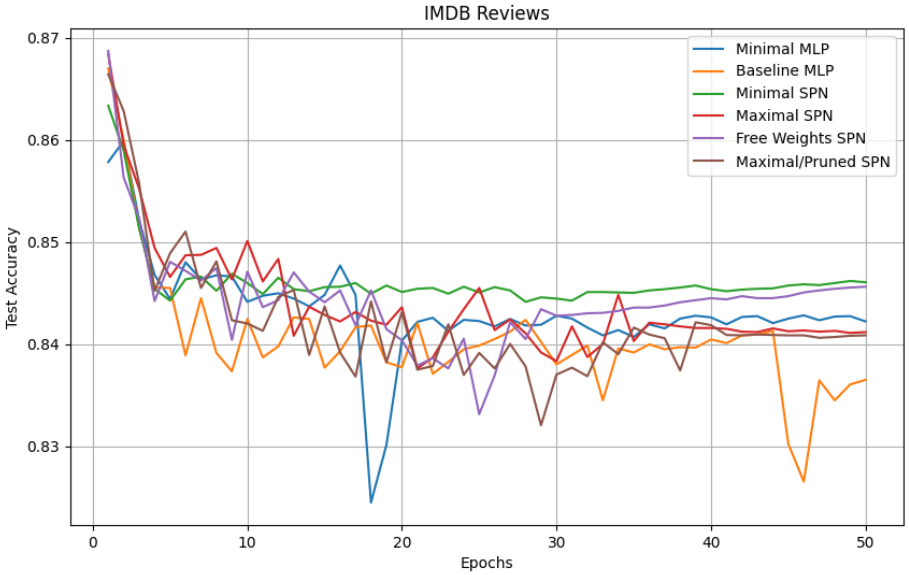

FIGURE 5.20: test_acc vs epochs curve for IMDb Reviews

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 [13]. These results further suggest that perceptron-based models are not well-suited for complex data tasks like sentiment analysis.

5.4 Discussion

This section seeks to answer the research questions posed in the introduction using the findings from the experimental results.

RQ1: Does increasing connection density in SPNs, while keeping the layer count and design identical to MLPs, lead to better model performance?

Across the majority of experiments, especially for the MNIST image dataset and both tabular datasets, SPNs with increased connection density clearly outperformed traditional MLPs with the same number of layers and neurons. In Table 5.1 and Table 5.4, fw_spn consistently achieved higher test accuracy than base_mlp, confirming that enhanced connectivity can yield better model performance. This effect was strongest in structured datasets where feature relationships are important and more easily captured with additional internal connections.

RQ2: Does removing connectivity restrictions in MLP architectures (allowing arbitrary neuron-to-neuron connections) enhance the network's learning capability or representational power?

The results affirm that lifting connectivity restrictions enhances both representational capacity and learning ability. In both simple and complex tabular datasets, SPNs consistently demonstrated better performance, as indicated by higher best test accuracy and area under curve efficiency. However, in highly complex domains such as CIFAR-10 and IMDb Reviews, the representational advantage of SPNs was limited by the fundamental capabilities of perceptron-based models, which struggle to capture spatial or semantic relationships lost during input flattening or vectorization. Thus, while enhanced connectivity is powerful, its effectiveness is domain-dependent.

RQ3: Can SPNs with fewer layers but higher connection density match or exceed the performance of deeper MLPs, thereby improving throughput efficiency and reducing training times?

Experiments with minimal models show that shallow, densely connected networks can often match or surpass deeper MLPs, especially on tabular and certain text

datasets (e.g., Newsgroups 20). For instance, Table 5.5 demonstrates that min_spn achieved both the highest test accuracy and best efficiency scores, while maintaining a slight edge over min_mlp in test accuracy across most tasks. This supports the view that increasing connection density can compensate for network depth, resulting in more efficient architectures without sacrificing performance.

RQ4: How do SPNs compare to MLPs in terms of training time, computational efficiency, memory usage, and predictive accuracy across a range of datasets and tasks?

SPNs generally require more memory due to their larger parameter count, as seen in all summary tables. However, their training efficiency and throughput efficiency were often on par with, or better than, MLPs, especially in the tabular and simple datasets. Notably, pruned SPNs demonstrated that computational efficiency can be recovered through targeted pruning, reducing both training time and memory usage while preserving accuracy (see Tables 5.1, 5.4). On complex image and language tasks, neither MLPs nor SPNs could match the specialized models (CNNs, ViTs, Transformers), and the efficiency benefits of SPNs were less pronounced.

RQ5: Does the increased architectural flexibility of SPNs improve their ability to generalize across diverse datasets and tasks compared to standard MLPs?

The experimental results show that SPNs' flexibility improves generalization in many settings, particularly for tabular and structured datasets. The ability to learn from more complex feature interactions enables SPNs to adapt to different data distributions, resulting in more robust performance across tasks. However, for highly unstructured or semantically complex tasks, such as image classification in CIFAR-10 or sentiment analysis in IMDb Reviews, the generalization advantage of SPNs diminishes, and specialized architectures remain superior.

RQ6: After maximizing connections in SPNs, can pruning strategies maintain high predictive performance while significantly improving computational efficiency and training time?

Pruning strategies applied to maximal SPNs were largely successful in reducing computational overhead without substantially sacrificing predictive accuracy. For example, in the MNIST and Covertype experiments, pruned SPNs maintained or even slightly improved peak test accuracy while reducing mean epoch time (see corresponding metrics tables). However, the practical benefit was sometimes offset by the additional time required for the pruning process itself, indicating that more efficient pruning algorithms could further enhance the value of this approach.

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)[15], Vision Transformers (ViTs)[6], Transformer-based language models[26], and Recurrent Neural Networks (RNNs)[16].

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. Due to time constraints, experiments were conducted with a single random seed. Future work will include statistical significance testing across multiple runs.

2. 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.

3. 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.

4. Applying alternative NAS strategies, such as reinforcement learning, could further optimize maximal SPNs by intelligently selecting and refining connectivity patterns, rather than relying solely on pruning. Exploring RL-based NAS may yield architectures that retain the expressive power of maximal SPNs while achieving greater efficiency and adaptability.

5. Integration with specialized architectures: While SPNs demonstrate clear improvements over traditional MLPs, their integration with current state-of-the-art architectures such as ResNets [11], MLP Mixer Architectures[25], CNNs[15], RNNs[16], and Transformer-based models[26] remains unexplored. Replacing perceptron-based components in these architectures with SPNs could potentially enhance these advanced models.

6. 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.

7. Hardware-level improvements could be explored by developing hybrid implementations that combine block-based and sequential processing strategies, leveraging the strengths of both approaches. Such hybrid methods may better utilize available hardware resources, potentially enabling more efficient training and inference for large and densely connected SPN architectures.

Overall, SPNs provide a valuable framework for exploring flexible connectivity in neural networks, particularly for small scale or tabular data applications.

Bibliography

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

[2] Bowen Baker et al. "Designing neural network architectures using reinforcement learning". In: *arXiv preprint arXiv:1611.02167* (2016).

[3] Han Cai, Ligeng Zhu, and Song Han. "Proxylessnas: Direct neural architecture search on target task and hardware". In: *arXiv preprint arXiv:1812.00332* (2018).

[4] Xin Chen et al. "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1294–1303.

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

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

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

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

[9] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *arXiv preprint arXiv:1803.03635* (2018).

[10] Song Han et al. "Learning both weights and connections for efficient neural network". In: *Advances in neural information processing systems* 28 (2015).

[11] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[12] 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*. 2017, pp. 1389–1397.

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

[14] Haifeng Jin, Qingquan Song, and Xia Hu. "Auto-keras: An efficient neural architecture search system". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 1946–1956.

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

[16] 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).

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search". In: *arXiv preprint arXiv:1806.09055* (2018).

[18] Hanxiao Liu et al. "Hierarchical representations for efficient architecture search". In: *arXiv preprint arXiv:1711.00436* (2017).

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

[20] Hieu Pham et al. "Efficient neural architecture search via parameters sharing". In: *International conference on machine learning*. PMLR. 2018, pp. 4095–4104.

[21] Esteban Real et al. "Regularized evolution for image classifier architecture search". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4780–4789.

[22] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

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

[24] Mingxing Tan et al. "Mnasnet: Platform-aware neural architecture search for mobile". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2820–2828.

[25] 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.

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

[27] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. "A survey on neural architecture search". In: *arXiv preprint arXiv:1905.01392* (2019).

[28] Lingxi Xie and Alan Yuille. "Genetic cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1379–1388.

[29] Chris Ying et al. "Nas-bench-101: Towards reproducible neural architecture search". In: *International conference on machine learning*. PMLR. 2019, pp. 7105–7114.

[30] Arber Zela et al. "Understanding and robustifying differentiable architecture search". In: *arXiv preprint arXiv:1909.09656* (2019).

[31] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (2016).