Master's Thesis

Phase Transition in Artificial Neural Networks

Aakash Kumar (20MS209)

Co-supervised by

Prof. Emanuele Natale*

Prof. Pradeep Kumar Mohanty†

Department of Physical Sciences

Indian Institute of Science Education and Research, Kolkata

August 2024 — May 2025

*CNRS researcher (HDR) at Université Côte d'Azur, Nice

†Professor, Department of Physical Sciences, Indian Institute of Science Education and Research, Kolkata

Declaration by the Student

Date: 01 May 2025

I, *Mr. Aakash Kumar*, Registration No. *20MS209*, dated *01 May 2025*, a student of the *Department of Physical Sciences* of the *5 Year BS-MS Dual Degree* programme of IISER Kolkata, hereby declare that this thesis is my own work and, to the best of my knowledge, it neither contains materials previously published or written by any other person, nor has it been submitted for any degree/diploma or any other academic award anywhere before. I have used the originality checking service to prevent inappropriate copying.

I also declare that all copyrighted material incorporated into this thesis is in compliance with the Indian Copyright Act, 1957 (amended in 2012) and that I have received written permission from the copyright owners for my use of their work.

I hereby grant permission to IISER Kolkata to store the thesis in a database which can be accessed by others.

[illegible]

Aakash Kumar

Department of Physical Sciences,
Indian Institute of Science Education and Research Kolkata.

Mohanpur 741246, West Bengal, India.

Certificate from the Supervisor

Date: 01 May 2025

This is to certify that the thesis titled *"Phase Transition in Artificial Neural Networks"* submitted by *Mr. Aakash Kumar*, Registration No. *20MS209*, dated *09 May 2025*, a student of the *Department of Physical Sciences* of the *5 Year BS-MS Dual Degree* programme of IISER Kolkata, is based upon his/her own research work under my supervision. I also certify, to the best of my knowledge, that neither the thesis nor any part of it has been submitted for any degree/diploma or any other academic award anywhere before. In my opinion, the thesis fulfills the requirement for the award of the degree of *Bachelor of Science and Master of Science.*

[illegible]

Prof. Emanuele Natale

CNRS researcher (HDR),
Université Côte d'Azur, COATI Team (INRIA).

2004, route des Lucioles, B.P. 93 F-06902 Sophia Antipolis Cedex, Nice, France.

Acknowledgments

First and foremost, I extend my heartfelt gratitude to my supervisor, *Prof. Emanuele Natale*, for his exceptional guidance and unwavering support throughout this thesis. His insights not only shaped my understanding of the subject but also introduced me to the core ideas essential for this work. Coming from a physics background, I initially thought I would struggle with the mathematical rigor involved. However, under his patient and expert mentorship, the transition felt seamless. I am also deeply grateful to my co-supervisor, *Prof. Pradeep Kumar Mohanty.* While not directly involved in the academic aspects of this work, his support in various other ways has been invaluable throughout my thesis journey. A special thanks to my friends Yash, Dipesh, Pratyay, Hassan, Aritra, Swastik, Rohit and Ankan. Only recently have I come to truly appreciate the importance of good friends, and I am incredibly fortunate to have you all by my side. Your encouragement and support have meant the world to me.

Abstract

Artificial neural networks have demonstrated remarkable utility across various domains; however, their training often demands significant computational resources. The Lottery Ticket Hypothesis suggests that within every fully connected neural network, there exists a smaller subnetwork that can be trained from scratch to achieve similar performance. This thesis builds on recent work related to the Strong Lottery Ticket Hypothesis (SLTH), an even stronger conjecture, which states that sufficiently overparameterized randomly initialized neural networks contain sparse subnetworks that will perform as well as a small trained network on a given dataset—without any training. This has motivated a considerable amount of research trying to prove that a given smaller network can be approximated by pruning a larger network. While previous studies have tried to answer how large a network needs to be to approximate a given target network through pruning, we go further by investigating both the required network size and the precision of its weights. We assume a target network of a given size, whose weights are represented with a certain precision, and a large a network whose weights are represented with more precision than the target, and explore the relationship that must hold between size and precision of the large network, that must hold in order for the larger network to represent the target network. In one of our results, the required network size is almost exact, mostly free of any arbitrary constants unlike other pervious works. Additionally, we show that the upper bound on the parameter count required to approximate a given network matches the lower bound asymptotically for a one layered network by parameter counting argument, hinting at the optimality of the solution.

Contents

Chapter 1

Introduction

Artificial Neural networks have been extremely successful across various domains in the past few decades. A large part of this success has been the increase in computational power and development of efficient training algorithms. This has allowed training of deeper networks, which capture the hidden patterns in a huge corpus of data. However, as the field advances, people plan to train even bigger networks, and hence finding efficient algorithms for training and inference remains an active area of research. Neural networks are highly over parametrized functions. During training, these parameters are adjusted such that the network captures patterns in data, so that it can generalize to data that it has not seen before. One of the compression techniques, known as pruning, involves setting parameters of the network, which are small in magnitude to zero. It has been observed that sometimes, even more than 95% of the parameters can be set to zero without any significant drop in performance. But if one tries to do the reverse, i.e., train a sparse network, it does not works well. Hence it was a big surprise to the community when [FC18] published a conjecture known as the lottery ticket hypothesis, which states that every dense network contains a sparse subnetwork that can be trained from scratch, and performs equally well as the dense network. They also gave experimental validation to their claim. Further works by [Zho+19], [Ram+19], [DK21] and [Wan+19] motivated an even stronger version of this hypothesis, called the Strong Lottery Ticket Hypothesis (SLTH). SLTH states that a sufficiently large neural network contains a subnetwork that will perform well on a dataset without any training. The strong lottery ticket was proved by [Mal+20]. They also provided bounds on how large this network should be to approximate the performance of a network of a given size. These bounds were later improved by [Pen+20], [Bur22]. These results rely on reducing the problem to a bunch of Random Subset Sum (RSS) problems and using results on the RSS problem [Lue98] to obtain bounds on the size of the large network. All the previous theoretical works in the SLTH space assume a large network whose weights are randomly initialized, sampled from some **continuous interval**. Then they try to prune this network in certain ways to approximates a given target network. Our thesis develops on these works. Our analysis heavily relies on the theory of Number Partitioning Problem (NPP). One of the striking features of NPP is that it has a phase transition [Mer98], [BCP01]. Through this thesis, we show how these results on NPP can serve as powerful tools to study SLTH.

1.1 Our Contribution

In this work, we add a new dimension to the strong lottery ticket problem-Weight Quantization. We assume a target network whose weights are sampled from a discrete set like $S_{\delta_1} = \{-1, \ldots, \delta_1, 2\delta_1, \ldots, 1\}$, where $\delta = 10^{-k_1}$ for some $k_1 \in \mathbb{N}$ and a large network whose weights are sampled from say $S_{\delta_2} = \{-1, \ldots, \delta_2, 2\delta_2, \ldots, 1\}$, where $\delta = 10^{-k_2}$, with $k_2 > k_1$. The number δ_i defines the precision of the network. The larger network can be **quantized**, i.e., the precision of it's weights can be decreased (by removing less significant digits from after the decimal place: $0.4326 \rightarrow 0.43$, for example) and it can then be **pruned**. We then ask what relationship must hold between the precision size of the large network such that it can be pruned the given target with high probability. This is more realistic, as computers always represent numbers with finite precision. Moreover, quantization is another method of compressing neural networks, which involves decreasing the precision with which the weights are represented. It has been observed that one can quantize a neural network to a great extent without loosing much performance. We further assume that after certain operation, the precision is set to δ again. For example, in a network, we may assume that output from any neuron of each odd layer is of precision δ. This may happen at different stages in the target and the large network, and may vary from result to result. This assumption was made for theoretical simplicity and we leave it to the future work to deal with a more general case. We adapt constructions from [Pen+20] and [Bur22] but starting in a quantized setting, we reduce the problem to solving a bunch of Number Partitioning Problem (NPP), which is an equivalent problem to RSS. NPP is one of the most important problems in the theory of NP-completeness and is known to exhibit a Phase transition [BCP01]. Using known results on NPP, it is easier to deal with quantized system. Instead of approximating the network output within an error ϵ, we go for exact representation, and generalizing our results to account for errors is another challenge for future works. Informal statements of these results is given.

Theorem (Informal version of Th. 6)**.** *A randomly initialized neural network of precision δ_2, width $\mathcal{O}\left(d\log_2\left(\frac{1}{\delta_2}\right)\right)$ and depth $2l$ of can be pruned to any network, with less precision, of width d and depth l with high probability.*

Theorem (Informal version of Th. 7)**.** *A randomly initialized neural network with width $2d\log_2\left(\frac{1}{\delta_2}\right)$ (except 1st layer whose width is $\mathcal{O}\left(d\log_2\left(\frac{1}{\delta_2}\right)\right)$) and depth $l+1$ can be pruned to any network, with less precision, of width d and depth l with high probability.*

In the latter of these results, note that the required network size of the large network is exact except for it's first layer, free of any undetermined constants unlike other pervious works. We also provide lower bounds on the required parameter count of a network to represent a two layered network. We do this by a parameter counting argument similar of [Pen+20], but in quantized setting. The informal statement is given below.

Theorem (Informal version of Th. 8)**.** *There exists a 2 layered network with d^2 parameters, which cannot be represented by a large neural network of precision δ_2 with high probability unless it has $\Omega\left(d^2\log_2\left(\frac{2}{\delta_2}\right)\right)$ parameters.*

Note that the upper and lower bounds match asymptotically, indicating the bounds are optimal in some sense. In summary, our contribution is 3-folds:

- We provide the relationship must hold between the precision and the size of the large network such that it can be pruned to the given target network with high probability.
- In one of the results, the required size of the large network is exact, free of undetermined constants, except for the first layer.
- Upper and lower bounds on the required size to represent a network match asymptotically, indicating the bounds are optimal in some sense.

Chapter 2

Neural Networks

2.1 What are Neural Networks?

Modern machine learning focuses on developing systems that learn from data by developing generative, predictive, or classification models through the analysis of complex patterns in data. In real world tasks, manually encoding intricate patterns is often infeasible. Modern machine learning addresses this challenge by leveraging highly overparameterized models that learn through loss optimization over given datasets. Machine learning has diverse applications across numerous fields, with notable examples including machine translation, large language models, speech recognition, and image generation. Much of its success stems from a subfield known as deep learning. Emerging in the 1980s, deep learning involves models that distribute computation across multiple layers to capture underlying complicated patterns in data. Neural networks were among the first examples of such learning systems, laying the foundation for many modern advancements in the field. neural networks by inspired by biological neural networks in animal brains both on terms of structure and function. In this chapter we describe what neural networks are and how they work. In this section we describe the construction of neural networks and in the next section we describe Backpropagation, an efficient algorithm to train neural networks. To construct neural networks we first describe a "neuron", which is the fundamental unit of a neural network. Neurons are functions $f : \mathbb{R}^n \to \mathbb{R}$ of the form

$$f(x_1, x_2, ..., x_n) = \sigma\left(\sum_{i=1}^{n} w_i x_i\right), \tag{2.1}$$

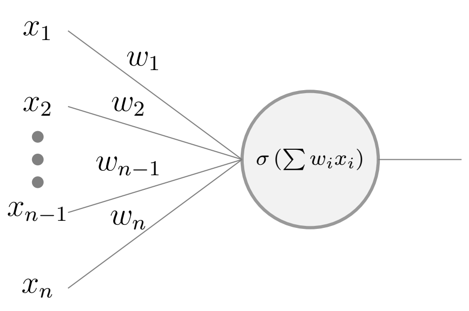

Figure 2.1: Pictorial representation of a neuron.

where $\sigma : \mathbb{R} \to \mathbb{R}$ is known as the non-linear activation function. A neuron is pictorially represented in Figure 2.1. A neuron essentially takes a linear combination of it's n inputs, weighted by w_i's, and then gives the linear combination as an argument to the non linear activation function σ to obtain it's output. The parameters w_i's are known as weights. It is usual to think of these numbers w_i's as numbers associated to the connections (the lines that bring the inputs to the neuron shown in Figure 2.1). We will be using phrases like "weight of the connection" for w_i's frequently. The commonly used non-linear activation functions are the sigmoid function defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

The sigmoid function is plotted in Figure 2.2. As shown in the plot, the sigmoid function compresses its input into the interval $[0, 1]$. It is a monotonically increasing function that approaches 1 as the input becomes very large. Other common choices are $\sigma(x) = \tanh(x)$ and $\sigma(x) = \mathrm{ReLU}(x) = \max\{0, x\}$ are also shown in Figure 2.2. Our model neuron is somewhat

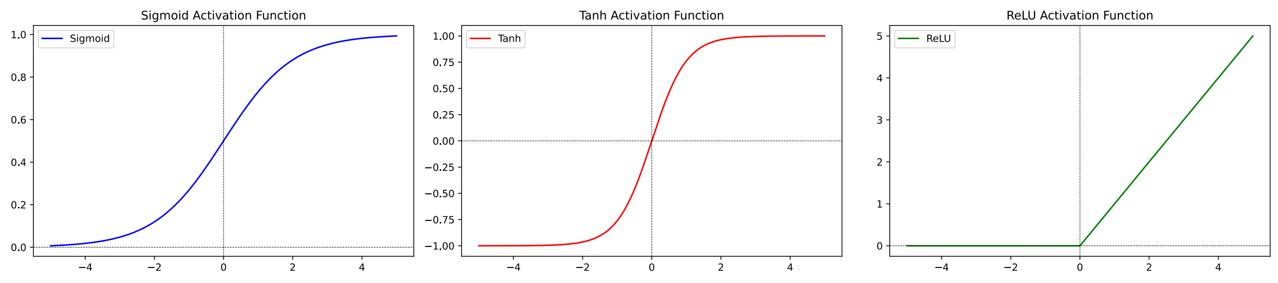

Figure 2.2: Sigmoid, Tanh, and ReLU activation functions.

similar in structure and function to a biological neuron. A biological neuron aggregates signals from other neurons through various connections, each with a strength that can change over time through complex bio-mechanisms. These strengths determine which signals are more important. If the aggregate input exceeds a certain threshold, the neuron "fires," transmitting signals to other neurons. Similarly, our artificial neuron weights incoming signals using w_i's and produces an output based on an increasing function of the aggregate input. Later, we will allow the w_i's to change during a process called training, enabling the neuron to capture meaningful patterns in the data. Having defined a neuron and motivating it's structure, we now construct a network of these neurons, called the feed forward neural network. We stack these neurons in layer and connect them together as shown in Figure 2.3. The neurons in the

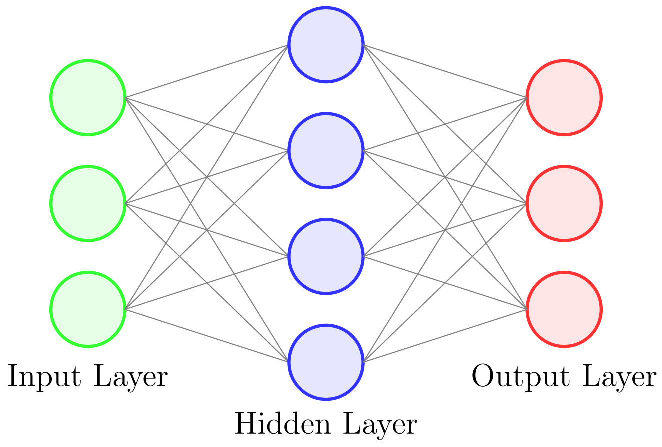

Figure 2.3: A feed-forward neural network.

Input layer take real number inputs and pass them without any change to the neurons in the next layer. A neuron only gives one real number as an output, multiple outputs shown in the figure mean that the neuron a giving the same output to all the neurons in the next layer. The neurons in the next layer, all reviving d_0 inputs, where d_0 is the number of neurons in the input layer, operate according to the rule given in equation 2.1. The neurons then pass outputs to the next layer, and the computation goes on. There can be as many hidden layers in between input and output layers as required. The signal finally reaches the output layer, giving d_ℓ real numbers, (where ℓ is the total number of layers and d_ℓ is the number of neurons in the output layer) which is the output of the network. Note that layers are numbered as $(0, 1, ..., \ell)$. Hence, a neural network defines a function from $\mathbb{R}^{d_0} \to \mathbb{R}^{d_\ell}$, parametrized by the weights of all the neurons. We represent the input $\in \mathbb{R}^{d_0}$ as $\boldsymbol{x} = (x_1, x_2, ...x_{d_0})$. The operation performed between the zeroth layer and the first layer can be represented by $\boldsymbol{W_1 x}$, where $\boldsymbol{W_1}$ is the matrix whose ij^{th} entry is the weight of connection connecting the j^{th} neuron in the zeroth layer and i^{th} neuron in the first layer. Using this compact notation, a neural network can be defined as

Definition 1. *An neural network is a function $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l}$ defined as*

$$f(\mathbf{x}) = \mathbf{W}_l \sigma(\mathbf{W}_{l-1} ... \sigma(\mathbf{W}_1 \mathbf{x})), \tag{2.2}$$

where $\mathbf{W}_i$ has dimension $d_i \times d_{i-1}$, $\mathbf{x} \in \mathbb{R}^{d_0}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the nonlinear activation function and $\mathbf{v} = \sigma(\mathbf{x})$ means $v_i = \sigma(x_i)$.

Having defined the neural network, we now come to the question of what to do with it. Consider some high dimensional data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ where $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$ and $\boldsymbol{y}_i \in \mathbb{R}^{d_\ell}$. Suppose we empirically want to determine the relationship between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. A neural network is a highly parametrized function $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_\ell}$. In that case we would like to figure out the set of weights of the neural network such that it best represents the relationship between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. Here one can also see the significance of non linear activation functions. In the absence of an activation function, the neural network is just a composition of linear functions and hence is a linear function it self. So it can fit only linear function. Adding a non linearity to the design makes it much more powerful. By adding a non linear activation function, the class of functions the neural network can represents increases drastically. Infact it can be shown that a neural network can represent any continuous function, with minimal requirements on what activation function you choose. Given $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, one of the natural things to do is to find the set of weights such that

$$C = \frac{1}{2N} \sum_{i=1}^{N} (f(x_i) - y_i)^2 \tag{2.3}$$

is minimum. C is known as the cost, or the loss function. The specific form in Equation 2.3 is known as the least square loss. This form is a choice, and many other choices of loss functions exist. Given $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, C is a function of all the weights of the network and we seek to minimize C over all possible values of the weights. This is a high dimensional optimization problem can can be quite challenging in general. The process of minimize C over all possible values of the weights is known as *training* in the language of machine learning.

2.2 The Backpropagation Algorithm

In this section we discuss an efficient gradient based algorithm to train a neural network. A neural network of the form given in Equation 2.2 will be used. The weight connecting the i^{th} neuron of the k^{th} layer and j^{th} neuron of $(k-1)^{\text{th}}$ layer will be denoted by w_{ij}^k. The output given by the i^{th} neuron of the k^{th} layer will be denoted by a_i^k. Hence we have

$$a_i^k = \sigma\left(\sum_j w_{ij}^k a_j^{k-1}\right).$$

We will also denote the input received by the i^{th} neuron of the k^{th} layer by

$$z_i^k = \sum_j w_{ij}^k a_j^{k-1}.$$

Let C be the cost function of the network. C need not be of the form 2.3, but we will assume that the cost function can be written as an average of the costs of individual training samples, i.e., $C = \frac{1}{n}\sum_x C_x$. Hence we will compute gradient of C with respect to a single training example and it can then easily be generalized to the entire dataset. We will also assume that C can be written as a function of the outputs from the neural network. Let the dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ be given. We define the error in the i^{th} neuron of the k^{th} layer δ_i^k as

$$\delta_i^k = \frac{\partial C}{\partial z_i^k}.$$

Fist we compute the error of the output layer δ_k^L, where L denotes the last layer. We have

$$\delta_k^L = \frac{\partial C}{\partial z_k^L} = \sum_{i=1}^n \frac{\partial C}{\partial a_i^L}\frac{da_i^L}{dz_k^L},$$

where n is the number of neurons in the last layer. The above equation gives

$$\delta_k^L = \frac{\partial C}{\partial a_k^L}\sigma'(z_k^L). \tag{2.4}$$

Next we compute the error in the ℓ^{th} in terms of $(\ell+1)^{\text{th}}$ layer. Let the number of neurons in the $(\ell+1)^{\text{th}}$ be m. Consider

$$\delta_k^\ell = \frac{\partial C}{\partial z_k^\ell}$$

$$\implies \delta_k^\ell = \sum_{i=1}^{m} \frac{\partial C}{\partial z_k^{\ell+1}} \frac{\partial z_k^{\ell+1}}{\partial z_k^\ell}$$

$$\implies \delta_k^\ell = \sum_{i=1}^{m} \delta_i^{\ell+1} w_{ik}^{\ell+1} \sigma'(z_k^\ell). \tag{2.5}$$

Now we compute $\frac{\partial C}{\partial w_{jk}^\ell}$. We have

$$\frac{\partial C}{\partial w_{jk}^\ell} = \frac{\partial C}{\partial z_k^\ell} \frac{\partial z_k^\ell}{\partial w_{jk}^\ell}$$

$$\implies \frac{\partial C}{\partial w_{jk}^\ell} = \delta_j^\ell a_k^{\ell-1}. \tag{2.6}$$

Equations 2.4, 2.5 and 2.6 are called the equations of Backpropagation. Now our goal is to minimize C with respect to w_{ij}^k for the given dataset, and hence we want to find it's gradient. The gradient is calculated using the equations of Backpropagation. Note that the partial derivative of C with respect to w_{ij}^k is computed layer by layer, starting from the last layer, and hence the name backpropagation. Having calculated ∇C, we can change w_{ij}^k by simply taking small steps in the direction negative gradient, also known as gradient descent. The algorithm goes as follows - First the entire dataset is given as input to the network and the cost is calculated. Then using the equations of backpropagation, the gradient of C

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w_{11}^2} \\ \vdots \\ \frac{\partial C}{\partial w_{nm}^\ell} \end{bmatrix}$$

is calculated. The weights are the updated as

$$w_{ij}^k \to w_{ij}^k - \eta \frac{\partial C}{\partial w_{ij}^k},$$

or in other words, if $\boldsymbol{w}$ denotes vector containing all the weights as elements, then

$$\boldsymbol{w} \to \boldsymbol{w} - \eta \nabla C. \tag{2.7}$$

This process is repeated until convergence, and one round is known as an epoch. Instead

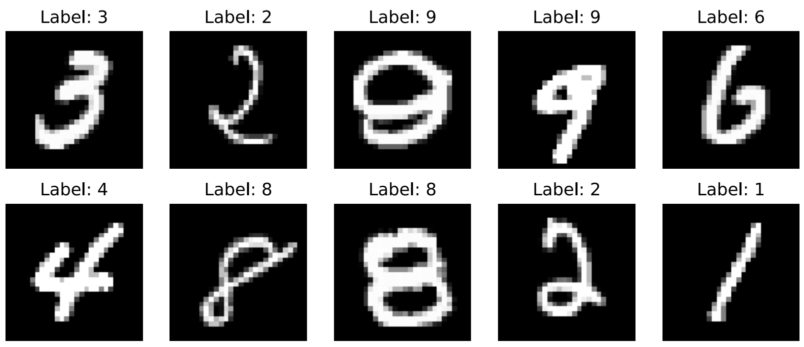

Figure 2.4: MNIST dataset contains of 60,000 training and 10,000 test samples. We used a batch size of 64 for training.

of calculating the loss over the entire dataset, one can also divide the dataset into batches, compute the loss for a batch, update the weights according to 2.7, and then move on to the next batch. This is known as stochastic gradient descent and is usually preferred over gradient descent for practical purposes. Figure 2.4 shows some samples from the MNIST dataset, a collection of images of handwritten images. A neural network was trained to recognize the digits. A fraction of dataset was kept separately and not used for training for testing the performance on previously unseen examples. Figure 2.5 shows the training and test losses as a function of epoch. This shows that neural networks have the capability to capture patterns

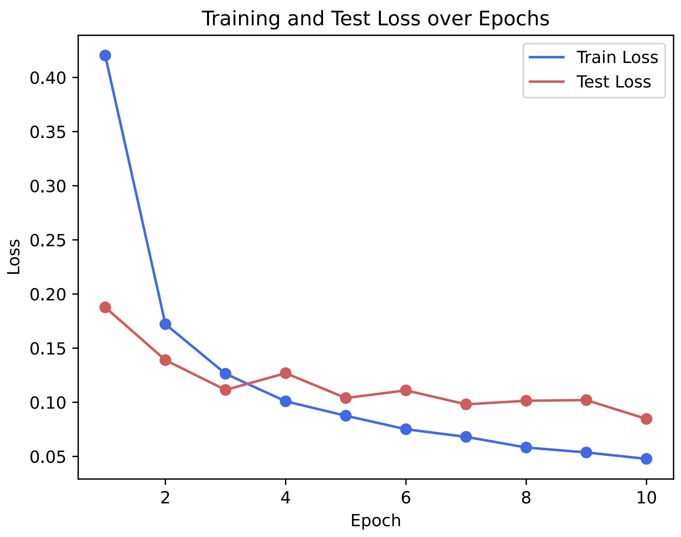

Figure 2.5: A neural network with dimensions $784 \times 128 \times 64 \times 10$ trained on MNIST.

and generalize to previously unseen data.

Chapter 3

The Number Partitioning Problem

3.1 Introduction

Many problems in combinatorics are known to exhibit a phase transition. A combinatorial phase transition is an abrupt change in the qualitative behavior of the problem as an appropriately defined parameter is varied. Well known examples are Erdos and Renyi graphs, k-SAT problem, etc. In this chapter we discuss phase transition in the Number Partitioning Problem (NPP). The problems is to partition n integers uniformly sampled from $\{1, 2, \ldots, M\}$ into two subsets such that the absolute value of the difference of their sums, is minimized. NPP is one of Garey and Johnson's six basic NP-complete problems that lie at the heart of the theory of NP-completeness, [GJ79]. In the early 1990s, there was a debate weather there is a phase transition in NPP or not. Fu modeled the problem as an infinite range Ising spin glass with Mattis-like, antiferromagnetic couplings [Ste89]. He argued that there won't be any phase transition. Then, is was found empirically that there is a phase transition in NPP. The control parameter

$$\kappa = \frac{\log_2 M}{n}$$

was proposed and it was shown empirically that if $\kappa < \kappa_c$, then $\mathcal{O}(2^n)$ number of solutions exist, whereas if $\kappa > \kappa_c$, the number of solutions drop to zero. [Mer98], using the tools from statistical mechanics showed (informally) that there is a phase transition in NPP with $\kappa = 1$. Though informal, Mertens analysis was quite convincing that there is a phase transition in NPP. The fact that there is a phase transition, along with many other results on NPP were proven by [BCP01] formally. These results are central to this thesis, and hence we shall go through both Mertens and Borgs analysis of the NPP in detail in this chapter.

3.2 NPP by Statistical Mechanics

Consider the set of integers $\{X_1, X_2, \ldots, , X_n\}$ where X_i is sampled uniformly from $\{1, 2, \ldots, M\}$. Since we wish to solve NPP, it is natural to consider a system with Hamiltonian

$$\mathcal{H} = \left| \sum_{i=1}^{n} X_i \sigma_i \right|$$

where $\sigma_i \in \{-1, 1\}$.

3.2.1 Calculation of the Partition function

We have the Hamiltonian

$$\mathcal{H} = \left| \sum_{i=1}^{n} X_i \sigma_i \right| .$$

The Canonical partition function (with $\beta = \frac{1}{T}$) is given by

$$\begin{aligned} Z &= \sum_{\sigma_i \in \{-1,1\}} e^{-\beta \mathcal{H}} \\ \implies Z &= \sum_{\sigma_i \in \{-1,1\}} \exp\left(-\beta \left| \sum_{i=1}^{n} X_i \sigma_i \right| \right) \\ \implies Z &= \sum_{\sigma_i \in \{-1,1\}} \int_{-\infty}^{\infty} dx \; e^{-|x|} \; \delta\left(x - \beta \sum_{i=1}^{n} X_i \sigma_i \right) . \end{aligned}$$

Now Fourier expanding the delta function

$$\boxed{\int_{-\infty}^{\infty} dx \; e^{ikx} = 2\pi \delta(k)}$$

$$\implies Z = \sum_{\sigma_i \in \{-1,1\}} \int_{-\infty}^{\infty} dx \; e^{-|x|} \int_{-\infty}^{\infty} \frac{d\tilde{x}}{2\pi} \exp\left(i\tilde{x} \left(x - \beta \sum_{i=1}^{n} X_i \sigma_i \right) \right)$$

$$\Longrightarrow Z = \sum_{\sigma_i \in \{-1,1\}} \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} \frac{d\tilde{x}}{2\pi} \exp\left(-i\tilde{x}\beta \sum_{i=1}^{n} X_i \sigma_i\right) e^{-|x|+ix\tilde{x}}$$

$$\Longrightarrow Z = \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} \frac{d\tilde{x}}{2\pi} \sum_{s_i \in \{-1,1\}} \exp\left(-i\tilde{x}\beta \sum_{i=1}^{n} X_i \sigma_i\right) e^{-|x|+ix\tilde{x}}. \tag{3.2a}$$

Now we simplify the term

$$\begin{aligned} &\sum_{\sigma_i \in \{-1,1\}} \exp\left(-i\tilde{x}\beta \sum_{i=1}^{n} X_i \sigma_i\right) \\ &= \sum_{\sigma_i \in \{-1,1\}} \prod_{i=1}^{n} \exp\left(-i\tilde{x}\beta X_i \sigma_i\right). \end{aligned}$$

Now expand the σ_i summation and use $2\cos(\theta) = e^{i\theta} + e^{-i\theta}$

$$= 2^N \prod_{i=1}^{n} \cos\left(\beta X_i \tilde{x}\right). \tag{3.2b}$$

Now we simplify the term

$$\begin{aligned} &\int_{-\infty}^{\infty} dx\ e^{-|x|+ix\tilde{x}} \\ &= \int_{-\infty}^{0} dx\ e^{x+ix\tilde{x}} + \int_{0}^{\infty} dx\ e^{-x+ix\tilde{x}} \\ &= \int_{-\infty}^{0} dx\ e^{x+ix\tilde{x}} + \int_{0}^{\infty} dx\ e^{-x+ix\tilde{x}} \\ &= \left[\frac{e^{x+ix\tilde{x}}}{1+i\tilde{x}}\right]_{-\infty}^{0} + \left[\frac{e^{-x+ix\tilde{x}}}{-1+i\tilde{x}}\right]_{0}^{\infty} \\ &= \frac{-1}{1+i\tilde{x}} + \frac{1}{-1+i\tilde{x}} \\ &= \frac{1}{i\tilde{x}-1} - \frac{1}{i\tilde{x}+1} = \frac{2}{\tilde{x}^2+1}. \end{aligned} \tag{3.2c}$$

Now put 3.2a and 3.2b in 3.2c we get

$$\begin{aligned} Z &= \int_{-\infty}^{\infty} \frac{d\tilde{x}}{2\pi} 2^n \prod_{i=1}^{n} \cos\left(\beta X_i \tilde{x}\right) \frac{2}{\tilde{x}^2+1} \\ \Longrightarrow Z &= 2^n \int_{-\infty}^{\infty} \frac{d\tilde{x}}{\pi} \prod_{i=1}^{n} \cos\left(\beta X_i \tilde{x}\right) \frac{1}{\tilde{x}^2+1} \end{aligned}$$

Put $\tilde{x} = \tan(y) \implies d\tilde{x} = \sec^2(y)\, dy$,

$$Z = 2^n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\sec^2(y)\, dy}{\pi} \prod_{i=1}^{n} \cos\left(\beta X_i \tan(y)\right) \frac{1}{\sec^2(y)}$$

$$\implies Z = 2^n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dy}{\pi} \prod_{i=1}^{n} \cos\left(\beta X_i \tan(y)\right). \tag{3.2d}$$

Now let

$$G(y) = \frac{1}{n} \sum_{i=1}^{n} \ln \cos(\beta X_i \tan(y)).$$

Hence equation 3.2d becomes

$$Z = 2^n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dy}{\pi} e^{n\, G(y)}. \tag{3.2e}$$

Now we employ saddle point approximation to calculate the partition function Z. Computing the saddle points of $G(y)$:

$$G(y) = \frac{1}{n} \sum_{i=1}^{n} \ln \cos(\beta X_i \tan(y))$$

$$\implies \frac{d}{dy} G(y) = -\frac{1}{n} \sum_{i=1}^{n} \frac{\sin(\beta X_i \tan(y))}{\cos(\beta X_i \tan(y))}\, \beta X_i \sec^2(y)$$

$$\implies \frac{d}{dy} G(y) = -\frac{1}{n} \sum_{i=1}^{n} \tan(\beta X_i \tan(y))\, \beta X_i \sec^2(y).$$

Note that $G(y)$ is zero if $y = \arctan(\frac{\pi}{\beta})k$ for any $k \in \mathbb{Z}$. From now on, we consider a_i's, instead of just being integers, to be an integral multiple of some real number Δa. If $a_i \in \mathbb{Z}$, then $\Delta a = 1$. In that case, $G(y)$ is zero if:

$$y_k = \arctan\left(\frac{\pi}{\beta \Delta a} k\right) \qquad \forall\, k \in \mathbb{Z}.$$

Now let's compute the second derivative of $G(y)$:

$$\frac{d^2}{dy^2} G(y) = -\frac{1}{n} \sum_{i=1}^{n} [\sec^2(\beta X_i \tan(y))\, (\beta a_i \sec^2(y))^2 + \tan(\beta X_i \tan(y))\, 2\beta X_i \sec^2(y) \tan(y)]$$

$$\implies \frac{d^2}{dy^2}G(y_k) = -\frac{1}{n}\sum_{i=1}^{n}\left(\beta X_i \sec^2\left(\arctan\left(\frac{\pi}{\beta\Delta a}k\right)\right)\right)^2$$
$$\implies \frac{d^2}{dy^2}G(y_k) = -\frac{1}{n}\sum_{i=1}^{n}\left(\beta X_i\left(1+\left(\frac{\pi k}{\beta\Delta a}\right)^2\right)\right)^2.$$

Note that we need to find the minima of $-NG(y)$, which corresponds to the maxima of $G(y)$. $G''(y)$ is negative at y_k, confirming that they are the maxima. Now we compute the partition function by applying saddle point approximation on 3.2e.

$$Z = 2^n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dy}{\pi}\exp\left(n\, G(y_0) + N\, G'(y_0)(y-y_0) + \frac{n}{2}G''(y_0)(y-y_0)^2\right),$$

where y_0 is a maxima. We know that at maxima, $G(y_0) = G'(y_0) = 0$. Hence

$$Z = 2^n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dy}{\pi}\exp\left(-\frac{n}{2}|G''(y_0)|\,|y-y_0|^2\right),$$

Now do this for every maxima and extend the domain to $(-\infty,\infty)$ as $y_k = \arctan\left(\frac{\pi k}{\beta\Delta a}\right) \in \left(-\frac{\pi}{2},\frac{\pi}{2}\right)$:

$$Z = 2^n \sum_{k\in\mathbb{Z}} \int_{-\infty}^{\infty} \frac{dy}{\pi}\exp\left(-\frac{n}{2}|G''(y_K)|\,|y-y_K|^2\right)$$
$$\implies Z = \frac{2^n}{\pi}\sum_{k\in\mathbb{Z}}\sqrt{\frac{2\pi}{n\frac{1}{n}\sum_{i=1}^{n}\left(\beta X_i\left(1+\left(\frac{\pi k}{\beta\Delta a}\right)^2\right)\right)^2}}$$
$$\implies Z = 2^n\sqrt{\frac{2}{\pi}}\sum_{k\in\mathbb{Z}}\frac{1}{\sqrt{\sum_{i=1}^{n}\left(\beta X_i\left(1+\left(\frac{\pi k}{\beta\Delta a}\right)^2\right)\right)^2}}$$
$$\implies Z = 2^n\sqrt{\frac{2}{\pi}}\sum_{k\in\mathbb{Z}}\frac{1}{\sqrt{\sum_{i=1}^{n}X_i^2}\;\beta\left(1+\left(\frac{\pi k}{\beta\Delta a}\right)^2\right)}$$
$$\implies Z = 2^n\sqrt{\frac{2}{\pi\sum_{i=1}^{n}X_i}}\sum_{k\in\mathbb{Z}}\frac{(\beta\Delta a)^2}{\beta\left((\beta\Delta a)^2+\pi^2k^2\right)}$$
$$\implies Z = 2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_{i=1}^{n}X_i}}\sum_{k\in\mathbb{Z}}\frac{(\beta\Delta a)}{(\beta\Delta a)^2+\pi^2k^2}.$$

Now using the Identity

$$\boxed{\sum_{n\in\mathbb{Z}} \frac{x}{x^2+n^2\pi^2} = \coth(x)}$$

we get

$$Z = 2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_j X_j^2}}\coth(\beta\Delta a).$$

Calculation of Average Energy and Entropy

The free energy of the system is given by

$$\begin{aligned} F &= -\frac{1}{\beta}ln(Z) \\ \implies F &= -\frac{1}{\beta}\ln\left(2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_j X_j^2}}\coth(\beta\Delta a)\right). \end{aligned}$$

The average energy $\langle E\rangle$ is given by

$$\begin{aligned} \langle E\rangle &= -\frac{\partial}{\partial\beta}\ln(Z) \\ \implies \langle E\rangle &= \Delta a\frac{2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_j X_j^2}}(\coth^2(\beta\Delta a)-1)}{2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_j X_j^2}}\coth(\beta\Delta a)} \\ \implies \langle E\rangle &= \Delta a\frac{\coth^2(\beta\Delta a)-1}{\coth(\beta\Delta a)}. \end{aligned}$$

The entropy can be calculated as

$$\begin{aligned} F &= \langle E\rangle - TS \\ \implies S &= \beta(\langle E\rangle - F) \\ \implies S &= \beta\left(\Delta a\frac{\coth^2(\beta\Delta a)-1}{\coth(\beta\Delta a)} + \frac{1}{\beta}\ln\left(2^n\sqrt{\frac{2\Delta a^2}{\pi\sum_j X_j^2}}\coth(\beta\Delta a)\right)\right) \\ \implies S &= \beta\Delta a\frac{\coth^2(\beta\Delta a)-1}{\coth(\beta\Delta a)} + n\ln 2 + \frac{1}{2}\ln\frac{2\Delta a^2}{\pi\sum_j X_j^2} + \ln(\coth(\beta\Delta a)) \\ \implies S &= n\ln 2 + \frac{1}{2}\ln\frac{2\Delta a^2}{\pi\sum_j X_j^2} + \tilde{S}(\beta\Delta a). \end{aligned}$$

Here $\tilde{S}(\beta\Delta a)$ is the Temperature-dependent part of the entropy. Now we define

$$\kappa_c(n) = 1 - \frac{\ln(\frac{\pi}{6}n)}{2n\ln 2}, \qquad \kappa = \frac{\ln\left(\frac{3}{\Delta a^2}\frac{1}{n}\sum_i X_i^2\right)}{2n\ln 2}.$$

Hence the entropy can be written as

$$S = n\ln 2(\kappa - \kappa_c) + \tilde{S}(\beta\Delta a).$$

3.2.2 Phase transition

For $\kappa < \kappa_c$, and at $T = 0$, entropy is extensive, and the corresponding energy is zero, hence an exponential number of partitions are expected to exist.
For $\kappa_c > \kappa$, we notice that $N\ln 2(\kappa - \kappa_c)$ is negative. But Entropy cannot be smaller than $\ln 2$ for our discrete system, hence the temperature-dependent part $\tilde{S}(\beta\Delta a)$ must contribute. Also notice that $\kappa_c > \kappa$ implies

$$\begin{aligned}
1 - \frac{\ln(\frac{\pi}{6}n)}{2n\ln 2} &> \frac{\ln\left(\frac{3}{\Delta a^2}\frac{1}{n}\sum_i X_i^2\right)}{2n\ln 2} \\
\implies 2n\ln 2 - \ln\left(\frac{\pi}{6}n\right) &> \ln\left(\frac{3}{\Delta a^2}\frac{1}{n}\sum_i X_i^2\right) \\
\implies 2n\ln 2 &> \ln\left(\frac{\pi}{2}\frac{1}{\Delta a^2}\sum_i X_i^2\right) \\
\implies 2^{-n} &> \Delta a\sqrt{\frac{2}{\pi\sum_i X_i^2}}.
\end{aligned}$$

Or in other words, $\Delta a = \mathcal{O}(2^{-N})$. Now we shall see that in this regime contribution from $\tilde{S}(\beta\Delta a)$ cannot be neglected. Consider the expansion of $\tilde{S}(\beta\Delta a)$

$$\tilde{S}(\beta\Delta a) = \ln\left(\frac{1}{\beta\Delta a}\right) + 1 + \mathcal{O}((\beta\Delta a)^2).$$

To deal with this contribution, we introduce an effective zero temperature T_0, beyond which the system cannot be cooled. We can estimate this effective zero from minimum entropy

$$S = n\ln 2(\kappa - \kappa_c) + \tilde{S}(\beta\Delta a) = \ln 2$$

$$\implies \ln 2 \approx n \ln 2(\kappa - \kappa_c) + \ln\left(\frac{1}{\beta_0 \Delta a}\right)$$
$$\implies T_0 = 2\ \Delta a\ 2^{n(\kappa-\kappa_c)}$$
$$\implies T_0 = 2^{-n}\sqrt{2\pi \sum_i X_i^2}.$$

For $\kappa > \kappa_c$, the approximate energy

$$E_0 = T_0 = 2^{-n}\sqrt{2\pi \sum_i X_i^2}.$$

Which is finite, and hence we expect the number of perfect partitions to drop to zero, hence we have a phase transition. Note that as $n \to \infty$, $\kappa_c \to 1$. This analysis, although not formal, is considered one of the major significant developments in the theory of NPP. It was the first to convincingly argue that there is a phase transition in NPP. Now we move in to analysis by [BCP01], which formalized these ideas.

3.3 Number Partitioning Problem: Formal Results

In this section, we describe the formal analysis of Phase transition in NPP by [BCP01]. We start with some definitions and then we state the relevant results.

Definition 2. *Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ be a set of integers sampled uniformly from the set $\{1, 2, 3, \ldots, M\}$. $\mathbb{P}_n$ is the probability measure induced by random variables $\boldsymbol{X}$. The Number Partitioning Problem is defined as the problem of finding a partitioning set $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ with $\sigma_j \in \{-1, 1\}$ such that $|\boldsymbol{\sigma} \cdot \boldsymbol{x}| = \ell$ for some given integer ℓ (called target).*

For a given instance of Number Partitioning Problem, the event that "$\sum_{i=1}^{n} X_i$ is even" will be denoted by $\mathscr{E}_n$, where as the event that the sum is odd will be denoted by $\mathscr{O}_n$.

Definition 3. *Let d_n denote the optimum (minimum) discrepancy of $\mathbf{X}$ over all σ:*

$$d_n = d_n(\mathbf{X}) = \min_{\boldsymbol{\sigma}} (|\boldsymbol{\sigma} \cdot \mathbf{X} - \ell|).$$

A partition is called perfect if $d_n \leq 1$.

It is clear that if $\sum_{i=1}^n X_i$ and ℓ are of same parity (both even or both odd), then a partition is perfect if $d_n = 0$ otherwise a partition is perfect if $d_n = 1$.

Definition 4. *For a number partitioning problem with n integers sampled from $\{1, 2, \ldots, M\}$, κ_n is defined as*

$$\kappa_n = \frac{\log_2 M}{n}.$$

We will also we using a more detailed parametrization $\log_2 M = \kappa_n n$ with

$$\kappa_n = 1 - \frac{\log_2 n}{2n} + \frac{\lambda_n}{n}. \tag{3.1}$$

Definition 5. *Given an instance of Number Partitioning Problem $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ with a set of size n and a target ℓ, $Z_{n,\ell}$ denotes the number of exact solutions to the NPP, i.e.,*

$$Z_{n,\ell} = \sum_{\boldsymbol{\sigma}} \mathbb{I}(|\boldsymbol{\sigma} \cdot \boldsymbol{X}| = \ell).$$

3.3.1 Statement of the Results

Theorem 1. *Let $\log_2 M = \kappa_n n$, and assume the there exists $\lim_{n\to\infty} \kappa_n = \kappa \in [0, \infty)$. Then*

$$\lim_{n\to\infty} \mathbb{P}_n(\exists \textit{ a perfect partition}) = \begin{cases} 1 & \textit{if } \kappa < 1 \\ 0 & \textit{if } \kappa < 1. \end{cases}$$

Theorem 2. *Let $C_0 > 0$ be a finite constant, let $M = M(n)$ be an arbitrary function of n, let*

$$\gamma_n = \frac{1}{M\sqrt{2\pi n c_M}}$$

where

$$c_M = \mathbb{E}\left(\frac{X^2}{M^2}\right) = \frac{1}{3} + \frac{1}{2M} + \frac{1}{6M^2} \ ,$$

and let ℓ and ℓ' be integers. Then,

$$\mathbb{E}[I_{n,\ell}] = \gamma_n \left(\exp\left(-\frac{\ell^2}{2nM^2 c_M} \right) + \mathcal{O}(n^{-1}) \right).$$

Furthermore

$$\mathbb{E}[I_{n,\ell}I_{n,\ell'}] = 2\gamma_n^2 \left(\exp\left(-\frac{\ell^2+(\ell')^2}{2nM^2c_M}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \mathcal{O}\left(\frac{1}{n\gamma_n 2^n}\right)\right) + \frac{\gamma_n}{2^n}\left(\delta_{\ell+\ell',0}+\delta_{\ell-\ell',0}\right)\exp\left(-\frac{\ell^2+(\ell')^2}{2nM^2c_M}\right)$$

if ℓ and ℓ' are of the same parity, i.e., both odd or both even, while $\mathbb{E}[I_{n,\ell}I_{n,\ell'}]=0$ if ℓ and ℓ' are of different parity.

3.3.2 Integral representation and moment estimates

In this section we prove Theorem 2. Estimates of moments of the integral representation of $Z_{n,\ell}$ are the key ingredients in the proving the main results. $Z_{n,\ell}$ (the number of partitions with $|\boldsymbol{\sigma}\cdot\boldsymbol{X}|=\ell$) can be written as

$$Z_{n,\ell} = \sum_{\boldsymbol{\sigma}} \mathbb{I}(|\boldsymbol{\sigma}\cdot\boldsymbol{X}|=\ell)$$

where $\mathbb{I}$ is defined as

$$\mathbb{I}(\boldsymbol{\sigma}\cdot\boldsymbol{X}=\ell) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i(\boldsymbol{\sigma}\cdot\boldsymbol{X}-\boldsymbol{\ell})x}dx$$

and the sum is over all possible configurations of $\boldsymbol{\sigma}$. This gives

$$Z_{n,\ell} = 2^n I_{n,\ell}\times\begin{cases}1 & \text{if } \ell=0\\ 2 & \text{if } \ell>0\end{cases}$$

where $I_{n,\ell}=I_{n,\ell}(X)$ is the random integral given by

$$I_{n,\ell} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(\ell x)\prod_{j=1}^{n}\cos(xX_j)dx\ . \tag{3.2}$$

Proof of Theorem 2. Let M be bounded. We first use Equation 3.2, independence of the X_j, and the Fubini theorem to get

$$\mathbb{E}(I_{n,\ell}) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(\ell x)f^n(x)dx \tag{3.3}$$

and

$$\mathbb{E}(I_{n,\ell}I_{n,\ell'}) = \frac{1}{(2\pi)^2}\int\int_{x_1,\, x_2\, \in\, (-\pi,\pi]} \cos(\ell x)\cos(\ell' x) f^n(x_1,x_2)dx_1dx_2 \tag{3.4}$$

where

$$f(x) := \mathbb{E}(\cos(xX)) = \frac{1}{M}\sum_{j=1}^{M}\cos(jx) = \frac{1}{M}\left[\frac{\sin\big((M+\frac{1}{2})x\big)}{2\sin(x/2)} - \frac{1}{2}\right]$$

and

$$f(x_1,x_2) := \mathbb{E}(\cos(x_1X)\cos(x_2X)) = \frac{1}{2}(f(x_1+x_2)+f(x_1-x_2)).$$

We now need to estimate the above integrals to get the required moment estimates. We use saddle point technique A to estimate the integrals as we care about the limit $n \to \infty$. It is important to note that there is an M in the denominator of both the integrands which can grow with n, however, [BCP01] showed by a careful treatment of error terms that one gets the same result. We only consider the case when M is bounded, as the general case is beyond the scope of this thesis. We start by estimating the first moment 3.3. We have

$$c_M = \frac{1}{3} + \frac{1}{2M} + \frac{1}{6M^2}.$$

Note that

$$c_M M^2 = \frac{M^2}{3} + \frac{M}{2} + \frac{1}{6} = \frac{2M^2+3M+1}{6},$$

so that

$$(M+1)(2M+1) = 2M^2+3M+1 = 6\,c_M M^2.$$

We wish to evaluate, for large n,

$$\mathbb{E}(I_{n,\ell}) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\cos(\ell x)\left\{\frac{1}{M}\left[\frac{\sin\Big((M+\frac{1}{2})x\Big)}{2\sin(x/2)} - \frac{1}{2}\right]\right\}^n dx,$$

We shall use saddle point A method to estimate the integral. Since

$$f(x) = \frac{1}{M}\sum_{j=1}^{M}\cos(jx)$$

its Taylor expansion around $x = 0$ is

$$f(x) = 1 - \frac{(M+1)(2M+1)}{12}\,x^2 + \mathcal{O}(x^4). \tag{3.5}$$

Where we have used

$$\sum_{m=-M}^{M} m^2 = \frac{M(M+1)(2M+1)}{3}.$$

It is convenient to introduce

$$c_M = \frac{(M+1)(2M+1)}{6M^2},$$

so that

$$\frac{(M+1)(2M+1)}{12} = \frac{c_M\,M^2}{2}.$$

Hence, we may write using 3.5

$$f(x) = 1 - \frac{c_M\,M^2}{2}\,x^2 + \mathcal{O}(x^4),$$

where $c_M = 1 + O(1/M)$. Now, raising this expression to the n^{th} power, we have

$$\left(1 - \frac{c_M\,M^2}{2}\,x^2 + \mathcal{O}(x^4)\right)^n.$$

For small x (the dominant region in the saddle point analysis), we write

$$\left(1 - \frac{c_M\,M^2}{2}\,x^2 + \mathcal{O}(x^4)\right)^n = \exp\Big[-\frac{n\,c_M\,M^2}{2}\,x^2\Big]\Big[1 + \mathcal{O}(n\,x^4)\Big].$$

Since the effective integration region is $x = \mathcal{O}\big(n^{-1/2}\big)$, we have $n\,x^4 = \mathcal{O}(1/n)$. In other words, the error in the exponentiation is of order $\mathcal{O}(1/n)$. Thus, for large n the integral is approximated by

$$\mathbb{E}(I_{n,\ell}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \cos(\ell x)\,\exp\Big[-\frac{n\,c_M\,M^2}{2}\,x^2\Big]\Big[1 + O\Big(\frac{1}{n}\Big)\Big]\,dx. \tag{3.6}$$

Since the dominant contribution comes from x near 0, we can extend the integration limits to $\pm\infty$ A (introducing an error that is exponentially small in n):

$$\mathbb{E}(I_{n,\ell}) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \cos(\ell x)\,\exp\Big[-\frac{n\,c_M\,M^2}{2}\,x^2\Big]\,dx\Big[1 + \mathcal{O}\Big(\frac{1}{n}\Big)\Big]. \tag{3.7}$$

Using the standard Gaussian integral

$$\int_{-\infty}^{\infty} \cos(bx) e^{-ax^2} dx = \sqrt{\frac{\pi}{a}} \exp\Big(-\frac{b^2}{4a}\Big),$$

with

$$a = \frac{n\, c_M\, M^2}{2} \quad \text{and} \quad b = \ell,$$

from 3.6 we obtain

$$\int_{-\infty}^{\infty} \cos(\ell x) \exp\Big[-\frac{n\, c_M\, M^2}{2}\, x^2\Big] dx = \sqrt{\frac{2\pi}{n\, c_M\, M^2}} \exp\Big[-\frac{\ell^2}{2\, n\, c_M\, M^2}\Big].$$

Collecting the error terms, we conclude that for large n, from 3.7 we get

$$\mathbb{E}(I_{n,\ell}) = \sqrt{\frac{1}{2\pi\, n\, c_M\, M^2}} \exp\Big[-\frac{\ell^2}{2\, n\, c_M\, M^2}\Big] \left[1 + \mathcal{O}\Big(\frac{1}{n}\Big)\right].$$

Now we estimate the second moment 3.4. For this we will only show the leading term to give the idea, the full proof is out of scope for this thesis. For the full proof, see [BCP01]. We have

$$\mathbb{E}[I_{n,\ell} I_{n,\ell'}] = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \cos(\ell x_1) \cos(\ell' x_2)\, f^n(x_1, x_2)\, dx_1\, dx_2, \tag{3.8}$$

where

$$f(x) = \frac{1}{M} \sum_{j=1}^{M} \cos(jx)$$

and

$$f(x_1, x_2) = \frac{1}{2}\Big[f(x_1 + x_2) + f(x_1 - x_2)\Big]. \tag{3.9}$$

Within the domain of integration, consider the square Q with corners at $(0, \pm\pi)$, $(\pm\pi, 0)$. The coordinate axes partition Q into the 4 isosceles triangles. The integration domain $[-\pi, \pi]^2$ consists of Q and four other triangles. Consider one of the latter triangles, in the first quadrant for example. It has its corners at $(0, \pi), (\pi, 0)$ and (π, π). Clearly, this triangle can be obtained via a parallel translation from the triangle with corners at $(-\pi, 0), (0, -\pi)$ and $(0, 0)$ in the direction of the vector (1,1). In this translation, every point (x_1, x_2) moves to a point (x_1', x_2')

such that

$$x_1', x_2' = x_1 + x_2 + 2\pi \qquad\qquad x_1' - x_2' = x_1 - x_2.$$

Now $f(x)$ is 2π periodic, $f(x_1, x_2) = f(x_1', x_2')$. Now note that when $\ell + \ell'$ is odd, the integral 3.8 is zero, i.e., integral is zero if ℓ and ℓ' are of different parity (one odd and one even). So we only need to consider ℓ and ℓ' of the same parity. Now we start estimating the integral. The idea is to again use saddle point approximation A. Clearly, $f(x_1, x_2)$ has a unique global maxima at $(x_1, x_2) = (0, 0)$ in $[-\pi, \pi]$ hence we expand around $(0, 0)$. For small x we expand the cosine as

$$\cos(jx) = 1 - \frac{(jx)^2}{2} + \mathcal{O}(x^4),$$

so that

$$f(x) = \frac{1}{M}\sum_{j=1}^{M}\cos(jx) = 1 - \frac{x^2}{2M}\sum_{j=1}^{M} j^2 + \mathcal{O}(x^4). \tag{3.10}$$

Since

$$\sum_{j=1}^{M} j^2 = \frac{M(M+1)(2M+1)}{6},$$

we define

$$c_M = \frac{(M+1)(2M+1)}{6M^2} = \frac{1}{3} + \frac{1}{2M} + \frac{1}{6M^2},$$

and hence we get

$$\implies \frac{c_M M^2}{2} = \frac{2M^2 + 3M + 1}{12}.$$

Thus, the expansion 3.10 becomes

$$f(x) = 1 - \frac{M^2 c_M}{2} x^2 + \mathcal{O}(x^4).$$

Similarly, expanding $f(x_1, x_2)$, Equation 3.9, for small x_1, x_2, we have

$$\begin{aligned} f(x_1, x_2) &= \frac{1}{2}\Big[f(x_1 + x_2) + f(x_1 - x_2)\Big] \\ &= \frac{1}{2}\Big(1 - \frac{M^2 c_M}{2}\Big[(x_1^2 + x_2^2) + (x_1^2 - x_2^2)\Big]\Big) + \mathcal{O}(x_1^4, x_2^4). \end{aligned} \tag{3.11}$$

Using

$$(x_1 + x_2)^2 + (x_1 - x_2)^2 = 2(x_1^2 + x_2^2),$$

Equation 3.11 simplifies to

$$f(x_1, x_2) = 1 - \frac{M^2 c_M}{2}(x_1^2 + x_2^2) + \mathcal{O}(x_1^4, x_2^4).$$

Now we want $(f(x_1, x_2))^n$. Note that

$$\log f(x_1, x_2) = -\frac{M^2 c_M}{2}(x_1^2 + x_2^2) + \mathcal{O}(x_1^4, x_2^4).$$

Hence we can write

$$\begin{aligned}(f(x_1, x_2))^n &= \exp\left[n \log(f(x_1, x_2))\right] \\ &= \exp\left[-\frac{nM^2 c_M}{2}(x_1^2 + x_2^2)\right] + \exp\left[n\mathcal{O}(x_1^4, x_2^4)\right].\end{aligned}$$

Because the main contribution comes from a small neighborhood around $(0, 0)$, we may extend the integrals to $\mathbb{R}^2$ without changing the leading asymptotics. Thus, the integral 3.8 becomes

$$\mathbb{E}[I_{n,\ell} I_{n,\ell'}] = \frac{1}{(2\pi)^2} \int_{\mathbb{R}^2} \cos(\ell x_1) \cos(\ell' x_2) \exp\left[-\frac{nM^2 c_M}{2}(x_1^2 + x_2^2)\right] + \exp\left[n\mathcal{O}(x_1^4, x_2^4)\right],$$

The leading order term (say I_0) is given by the separable Gaussian integrals

$$I_0 = \frac{1}{(2\pi)^2} \prod_{j=1}^{2} \int_{-\infty}^{\infty} \cos(\ell_j x) \exp\Big(-\frac{nM^2 c_M}{2} x^2\Big) dx,$$

with $\ell_1 = \ell$ and $\ell_2 = \ell'$. Using

$$\int_{-\infty}^{\infty} \cos(bx) e^{-ax^2} dx = \sqrt{\frac{\pi}{a}} \exp\Big(-\frac{b^2}{4a}\Big),$$

with $a = \frac{nM^2 c_M}{2}$, we obtain

$$\begin{aligned}I_0 &= \frac{1}{(2\pi)^2} \left(\sqrt{\frac{2\pi}{nM^2 c_M}} \exp\Big[-\frac{\ell^2}{2nM^2 c_M}\Big]\right) \left(\sqrt{\frac{2\pi}{nM^2 c_M}} \exp\Big[-\frac{(\ell')^2}{2nM^2 c_M}\Big]\right) \\ &= \frac{1}{2\pi n M^2 c_M} \exp\Big[-\frac{\ell^2 + (\ell')^2}{2nM^2 c_M}\Big].\end{aligned}$$

It is convenient to introduce

$$\gamma_n = \frac{1}{M\sqrt{2\pi n\, c_M}},$$

so that the leading contribution is written as

$$\gamma_n^2 \exp\Big(-\frac{\ell^2 + (\ell')^2}{2nM^2 c_M}\Big).$$

Thus the local saddle-point expansion gives

$$\mathbb{E}[I_{n,\ell} I_{n,\ell'}] = \gamma_n^2 \Big\{ \exp\Big(-\frac{\ell^2 + (\ell')^2}{2nM^2 c_M}\Big) + \mathcal{O}\Big(\frac{1}{n}\Big) \Big\}.$$

A more careful analysis (which is out of scope of this thesis) done by [BCP01] shows that the integral is

$$\mathbb{E}[I_{n,\ell} I_{n,\ell'}] = 2\gamma_n^2 \left(\exp\left(-\frac{\ell^2 + (\ell')^2}{2nM^2 c_M}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \mathcal{O}\left(\frac{1}{n\gamma_n 2^n}\right)\right) + \frac{\gamma_n}{2^n}\left(\delta_{\ell+\ell',0} + \delta_{\ell-\ell',0}\right) \exp\left(-\frac{\ell^2 + (\ell')^2}{2nM^2 c_M}\right).$$

Note that we have only considered bounded M, but in reality, it can grow and saddle point method cannot be applied directly. Fortunately, a careful analysis of error terms by [BCP01] showed that the result still holds, but we will not discuss that as it is out of scope for this thesis. □

3.3.3 Phase transition

Having the moment estimates of $I_{n,\ell}$, we now turn to establishing the existence of Phase transition in NPP formally (Theorem 1). We start with the estimating the probabilities of existence of perfect partitions.

Lemma 1. *Given a Number Partitioning Problem, the probability $\mathbb{P}(Z_{n,\ell} > 0)$ is bounded above and below as*

$$\mathbb{P}(Z_{n,\ell} > 0) \leq \begin{cases} \frac{\rho_n}{2}\left(\exp\left(-\frac{\ell^2}{2nM^2 c_M}\right) + \frac{C_1}{n}\right) & if \;\; \ell = 0 \\ \rho_n\left(\exp\left(-\frac{\ell^2}{2nM^2 c_M}\right) + \frac{C_1}{n}\right) & if \;\; \ell \neq 0 \end{cases}$$

$$\mathbb{P}(Z_{n,\ell} > 0) \geq \frac{1}{2\left(1 + \exp\left(\frac{\ell^2}{nM^2c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)}$$

where ρ_n is defined as $\rho_n = 2^{n+1}\gamma_n$.

Proof. Consider $\ell \neq 0$. From Theorem 2 we have

$$\mathbb{E}[I_{n,\ell}] = \gamma_n \left(\exp\left(-\frac{\ell^2}{2nM^2c_M}\right) + \mathcal{O}(n^{-1})\right).$$

If we multiply by 2^{n+1} we get

$$\mathbb{E}[Z_{n,\ell}] = \rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2c_M}\right) + \mathcal{O}(n^{-1})\right).$$

In other words, there exists a constant $C_1 > 0$ such that

$$\mathbb{E}[Z_{n,\ell}] \leq \rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2c_M}\right) + \frac{C_1}{n}\right). \tag{3.12}$$

Also notice that

$$\mathbb{E}[Z_{n,\ell}] \geq \rho_n \exp\left(-\frac{\ell^2}{2nM^2c_M}\right). \tag{3.13}$$

Furthermore, from Theorem 2 we have

$$\mathbb{E}[I_{n,\ell}I_{n,\ell'}] = 2\gamma_n^2 \left(\exp\left(-\frac{\ell^2 + (\ell')^2}{2nM^2c_M}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \mathcal{O}\left(\frac{1}{n\gamma_n 2^n}\right)\right) + \frac{\gamma_n}{2^n}\left(\delta_{\ell+\ell',0} + \delta_{\ell-\ell',0}\right)\exp\left(-\frac{\ell^2 + (\ell')^2}{2nM^2c_M}\right)$$

If $\ell = \ell'$ we get

$$\mathbb{E}[I_{n,\ell}^2] = 2\gamma_n^2 \left(\exp\left(-\frac{2\ell^2}{2nM^2c_M}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \mathcal{O}\left(\frac{1}{n\gamma_n 2^n}\right)\right) + \frac{\gamma_n}{2^n}\exp\left(-\frac{2\ell^2}{2nM^2c_M}\right)$$

Multiplying by $(2^{n+1})^2$ we get

$$\mathbb{E}[Z_{n,\ell}^2] = 2\rho_n^2 \left(\exp\left(-\frac{2l^2}{2nM^2c_M}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \mathcal{O}\left(\frac{1}{n\rho_n}\right)\right) + 2\rho_n \exp\left(-\frac{2\ell^2}{2nM^2c_M}\right)$$

Or in other words, there exists a constants $C_2 > 0$ and $C_3 > 0$ such that

$$\mathbb{E}[Z_{n,\ell}^2] \leq 2\rho_n^2 \left(\exp\left(-\frac{2\ell^2}{2nM^2c_M}\right) + \left(\frac{C_2}{n}\right) + \left(\frac{C_3}{n\rho_n}\right)\right) + 2\rho_n \exp\left(-\frac{2\ell^2}{2nM^2c_M}\right). \quad (3.14)$$

Now using Markov's inequality B and 3.12 we get

$$\mathbb{P}(Z_{n,\ell} > 0) \leq \rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2c_M}\right) + \frac{C_1}{n}\right).$$

Using Cauchy-Schwartz inequality B, 3.13 and 3.14 we get

$$\mathbb{P}(Z_{n,\ell} > 0) \geq \frac{\rho_n^2 \exp\left(-\frac{2\ell^2}{2nM^2c_M}\right)}{2\rho_n^2\left(\exp\left(-\frac{2\ell^2}{2nM^2c_M}\right) + \left(\frac{C_2}{n}\right) + \left(\frac{C_3}{n\rho_n}\right)\right) + 2\rho_n \exp\left(-\frac{2\ell^2}{2nM^2c_M}\right)}$$
$$\implies \mathbb{P}(Z_{n,\ell} > 0) \geq \frac{1}{2\left(1 + \exp\left(\frac{\ell^2}{nM^2c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)}.$$

The same calculation can be done for $\ell = 0$, the only difference is that $Z_{n,\ell} = 2^n I_{n,\ell}$. □

Now we use 1 to prove Theorem 1. It follows from a simple analysis of probability estimates under the right limits.

Proof of Theorem 1. We need to find the probability of existence of perfect partitions. Let Z_n be the number of perfect partition for zero target. Clearly, on $\mathcal{E}_n$, we have $Z_{n,0} = Z_n$ and $Z_{n,1} = 0$ whereas on $\mathcal{O}_n$, we have $Z_{n,1} = Z_n$ and $Z_{n,0} = 0$. Hence we have

$$\mathbb{P}(Z_n > 0 | \mathcal{E}_n) = \mathbb{P}(Z_{n,0} > 0)\mathbb{P}(\mathcal{E}_n) \qquad \text{and} \qquad \mathbb{P}(Z_n > 0 | \mathcal{O}_n) = \mathbb{P}(Z_{n,1} > 0)\mathbb{P}(\mathcal{O}_n).$$

It can be shown that $\mathbb{P}(\mathcal{E}) = \mathbb{P}(\mathcal{E}) = \frac{1}{2}$ with an error exponentially small in n. Hence we can write

$$\mathbb{P}(Z_n > 0) = \mathbb{P}(Z_{n,0} > 0)\mathbb{P}(\mathcal{E}_n) + \mathbb{P}(Z_{n,1} > 0)\mathbb{P}(\mathcal{O}_n)$$
$$\implies \mathbb{P}(Z_n > 0) = \frac{1}{2}\mathbb{P}(Z_{n,0} > 0) + \frac{1}{2}\mathbb{P}(Z_{n,1} > 0) + \mathcal{O}(2^{-n}).$$

Now using Lemma 1 we can write

$$\mathbb{P}(Z_n > 0) = \frac{1}{1 + \mathcal{O}\left(\frac{1}{n\rho_n}\right) + \mathcal{O}\left(\frac{1}{n}\right) + \frac{1}{\rho_n}}. \tag{3.15}$$

The sensitive parametrization defined in Equation 3.1 corresponds to

$$M = \frac{2^{n+\lambda_n}}{\sqrt{n}}.$$

In this parametrization $\lim_{n\to\infty} \kappa_n < 1$ means $\lim_{n\to\infty} \lambda_n \to -\infty$. Note that in this regime $\rho_n \to \infty$. Now from Equation 3.15, we can see that $\mathbb{P}(Z_n > 0) \to 1$ as $n \to \infty$. On the other hand $\lim_{n\to\infty} \kappa_n > 1$ means $\lim_{n\to\infty} \lambda_n \to \infty$. In this regime $\rho_n \to 0$. Hence again by Lemma 1 we have $\mathbb{P}(Z_{n,\ell} > 0) \leq \rho_n(1 + \mathcal{O}(n^{-1}))$ for $\ell = \mathcal{O}(M)$. Hence we clearly have $\mathbb{P}(Z_n > 0) \to 0$ as $n \to \infty$ $\square$

This proves the existence of phase transition in NPP. This was an extremely celebrated result in theoretical computer science, especially in the study NP-completeness. In this thesis, we have shown, as we shall see in the later chapters, how these results can serve as powerful tools for studying Strong Lottery Ticket Hypothesis.

Chapter 4

Neural Network Pruning and Lottery Tickets

4.1 The Lottery Ticket Hypothesis

As discussed in Chapter 2, neural networks are powerful tools across various domains, but their training requires significant computational resources. In some cases, these networks grow so large that even inference becomes computationally expensive. Consequently, developing efficient algorithms for both training and inference is a crucial area of research in machine learning. One effective approach to improving efficiency is pruning, which reduces the number of parameters in a neural network by systematically removing weights. Remarkably, pruning techniques can reduce the parameter count of trained networks by over 90% without a significant drop in performance. This not only accelerates inference but also minimizes storage requirements, allowing one to train a large network, prune it, and achieve much faster inference with a significantly smaller memory footprint. A widely used pruning method is *magnitude pruning*, where the network is first trained, and then the smallest $p\%$ of the weights—determined by their magnitude—are set to zero. However, training a sparse network from scratch does not perform as well as pruning a fully trained network as illustrated in Figure 4.1. Therefore it was a big surprise when [FC18] proposed the following hypothesis: A dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that—when trained in isolation — reach test accuracy comparable to the original network in a similar number of iterations. If

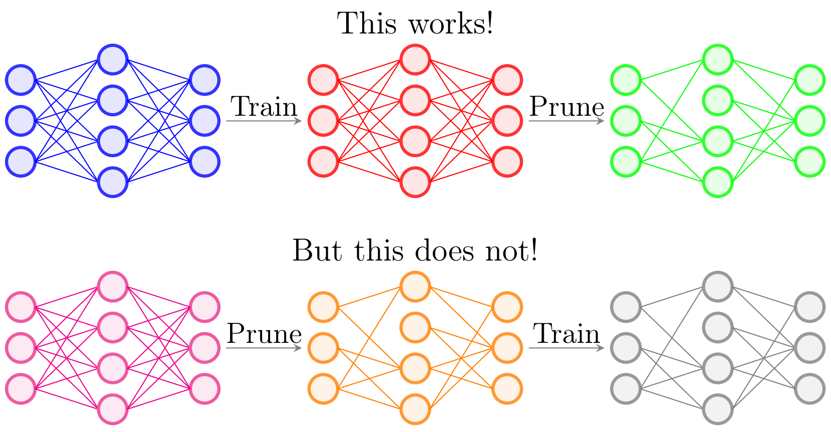

Figure 4.1: Training a sparse network from scratch does not perform as well as pruning a fully trained network.

true, and if one can find a way to find these sparse networks (called Lottery Tickets) hidden in the dense network, it would be great as this would reduce both the training and the inference cost. To support their hypothesis, they also gave an algorithm to identify the Lottery Tickets called the Iterative Magnitude Pruning (IMP) algorithm. The IMP algorithm is illustrated in Figure 4.2 It goes as follows: First, randomly initialize the weights of a network and note this

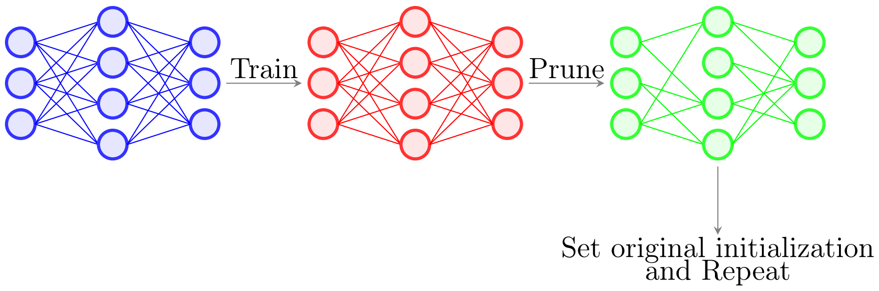

Figure 4.2: The Iterative Magnitude Pruning (IMP) algorithm.

initialization. Then train this network on a given dataset. Now prune a certain amount of the weights of the network based on the magnitude. For example, smallest 5% of the weights are removed. These weights are now freezed and cannot be trained further. For the rest of the weights, set their value to the original initialization and train again on the given dataset. Repeat the process until the performance is not dropping significantly. This procedure can find sparse subnetworks "Lottery tickets", that can be trained from scratch and still obtain results comparable to a dense network. The algorithm basically finds the so called "masks", which are basically a set of weights which are supposed to be set to zero freezed during training. The conjecture was quite celebrated in the field due to its impactful promise. However, as it is evident, it takes multiple rounds of training to find the Lottery, and hence it is not really helping in decreasing in decreasing the training cost. Off course, it decreases the inference cost, that is not the only bane of this this algorithm. Its was also realized that there Lottery tickets are transferable, i.e., once a Lottery ticket is found on a given dataset, it can be used on similar datasets. This is quite powerful as one can spend computational resources on finding the lottery ticket on some dataset, and then it can be used on many similar datasets making most of the training much less expensive. Therefor the Lottery ticket hypothesis became more famous and a lot more experimental work was done. But the problem of proving the hypothesis remains open. It turns out that it is extremely hard to do any kind of theoretical work on a trained network, and hence it is extremely difficult to make progress on this problem.

4.2 The Strong Lottery Ticket Hypothesis

In 2018, [FC18] proposed the Lottery Ticket Hypothesis, which states that every dense network contains a sparse subnetwork that can be trained from scratch, and performs equally well as the dense network. Following this line of work, [Zho+19], [Ram+19] and [Wan+19] found algorithms to find subnetworks within large randomly initialized networks that perform as good on a given task. This motivated the Strong Lottery Ticket Hypothesis (SLTH), which states that sufficiently overparameterized randomly initialized neural networks contain sparse subnetworks that will perform as well as a small trained network on a given dataset without any training. This motivated a lot of formal results proving that a given target network can be approximated by pruning a sufficiently large network. One of the first results were by [Mal+20], where they showed that to approximate a target network of width l and depth d, one needs a network of depth $2l$ and width $\mathcal{O}(d^5 l^2)$. [Pen+20] improved this bound by proving that width $\mathcal{O}(d \log(dl))$ is enough. Another construction was provided by [Bur22], where they showed that a network of width $l+1$ is enough to approximate a network of width l, with a certain compromise on the width. An informal statement of these kinds of results is provided below.

Theorem (Informal statement of previous SLTH results). *With high probability, a random artificial neural network N_S with m parameters can be pruned so that the resulting subnetwork N ϵ-approximates (i.e., approximates up to an error ϵ) any target artificial neural network N_t*

with $\mathcal{O}(m/\log_2(1/\epsilon))$ parameters.

There are also results on the lower bound on the required size of a network in order to approximate a given target network. An informal statement of the result by [Pen+20] on lower bound is given below.

Theorem (Informal statement of Theorem 4)**.** *There exist a to layered net 2-layer neural network with width d which cannot be approximated to error within ϵ by pruning a randomly initialized 2-layer network, unless the random network has width at least $\Omega(d\log(1/\epsilon))$.*

[DK21] showed that within a large network, there exists subnetworks that perform well on a given task, and are resilient to extreme forms of quantization, such as binarization. They also gave a theoretical result related to their claim.

Theorem (Informal, [DK21])**.** *Every fully-connected (FC) target network with ReLU activations of depth l and width n with bounded weights, a random binary FC network with ReLU activations of depth $2l$ and width $\mathcal{O}(ln^{3/2} + ln\log(ln/\delta))$ contains with probability $(1-\delta)$ a binary subnetwork that approximates the target network with error at most ϵ.*

For the convenience of the reader, we now state some of the results mentioned in this section formally.

4.2.1 SLTH: Formal Results

First we define some notation and setup. We use lowercase letters to represent scalars (w) and bold lower-case letters to denote vectors ($\mathbf{v}$). The i-th coordinate of the vector $\mathbf{v}$ is denoted as v_i. Matrices are denoted by bold upper-case letters ($\mathbf{W}$). the ℓ_2 norm of a vector $\boldsymbol{v}$ will be denoted by $\|\boldsymbol{v}\|$. The uniform distribution over the interval $[a, b]$ will be denoted by $U[a, b]$. We use c, C to denote positive absolute constants. We also recall the definition of neural network from Chapter 2.

Definition 6. *An neural network is a function $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l}$ defined as*

$$f(\mathbf{x}) = \mathbf{W}_l\sigma(\mathbf{W}_{l-1}...\sigma(\mathbf{W}_1\mathbf{x})), \tag{4.1}$$

where $\mathbf{W}_i$ *has dimension* $d_i \times d_{i-1}$*,* $\mathbf{x} \in \mathbb{R}^{d_0}$*, and* $\sigma : \mathbb{R} \to \mathbb{R}$ *is the nonlinear activation function and* $\mathbf{v} = \sigma(\mathbf{x})$ *means* $v_i = \sigma(x_i)$*.*

From here on, all neural networks will have ReLU nonlinear activation, i.e., $\sigma(x) = \max(0, x)$. Our goal is to start with a neural network $f(\mathbf{x})$ and a second larger neural network $g(\mathbf{x})$ of the form

$$g(\mathbf{x}) = \mathbf{M}_{2l}\sigma(\mathbf{M}_{2l-1}...\sigma(\mathbf{M}_1\mathbf{x})),$$

and approximating f by obtaining a pruned version of g by eliminating its weights

$$\hat{g}(\mathbf{x}) = (\mathbf{S}_{2l} \odot \mathbf{M}_{2l})\sigma((\mathbf{S}_{2l-1} \odot \mathbf{M}_{2l-1})...\sigma((\mathbf{S}_1 \odot \mathbf{M}_1)\mathbf{x})),$$

where each $\mathbf{S}_i$ is a binary (pruning) matrix, with the same dimension as $\mathbf{M}_i$, and $\odot$ represents element-wise product between matrices.

Theorem 3. *Let* $\mathcal{F}$ *be a neural network of the form defined in Equation 4.1. Consider a* $2l$ *layered randomly initialized* $2l$*-layered neural network*

$$g(\mathbf{x}) = \mathbf{M}_{2l}\sigma(\mathbf{M}_{2l-1}...\sigma(\mathbf{M}_1\mathbf{x})),$$

whose weights are drawn from $U[-1, 1]$*,* $\mathbf{M}_{2i}$ *has dimension*

$$d_i \times Cd_{i-1} \log \frac{d_i d_{i-1} l}{\min\{\epsilon, \delta\}},$$

and $\mathbf{M}_{2i-1}$ *has dimension*

$$Cd_{i-1} \log \frac{d_{i-1} d_i l}{\min\{\epsilon, \delta\}} \times d_{i-1}.$$

Then with probability at least $1 - \delta$*, for every* $f \in \mathcal{F}$*,* $\exists\, \mathbf{S}_i$ *such that*

$$\min_{\boldsymbol{S}_i \in \{0,1\}^{d_i \times d_{i-1}}} \sup_{\|\boldsymbol{x}\| \leq 1} \|f(\boldsymbol{x}) - (\mathbf{S}_{2l} \odot \mathbf{M}_{2l})\sigma((\mathbf{S}_{2l-1} \odot \mathbf{M}_{2l-1})...\sigma((\mathbf{S}_1 \odot \mathbf{M}_1)\mathbf{x}))\| \leq \epsilon.$$

Theorem 3 is basically saying that to approximate a target network of width l and depth d, one needs a network of depth $2l$ width $\mathcal{O}(d \log(dl))$. Now we discuss a lower bound proved by [Pen+20]. Before stating the main result, note that any linear transformation $\mathbf{Wx}$ where $\mathbf{W} \in \mathbb{R}^d \times \mathbb{R}^d$ and $\mathbf{W} \in \mathbb{R}^d$ can be expressed as a 2 layered neural network. Let $\mathcal{F}$ be the class of functions

$$\mathcal{F} := \{h_{\mathbf{W}} : \mathbf{W} \in \mathbb{R}^d \times \mathbb{R}^d\}, \qquad \text{where} \qquad h_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & -\mathbf{I} \end{bmatrix} \sigma\left(\begin{bmatrix} \mathbf{W} \\ -\mathbf{W} \end{bmatrix} \mathbf{x}\right). \tag{4.2}$$

Theorem 4. *Consider a neural network, $g : \mathbb{R}^d \to \mathbb{R}^d$ of the form $g(\mathbf{x}) = \mathbf{M}_l \sigma(\mathbf{M}_{l-1} ... \sigma(\mathbf{M}_1 \mathbf{x}))$, with arbitrary distributions on $\mathbf{M}_i$'s. Let $\mathcal{G}$ be the set of neural networks that can be formed by pruning g. Let $\mathcal{F}$ be defined as in Equation 4.2. If the following statement holds:*

$$\forall\, h \in \mathcal{F},\ \mathbb{P}\left(\exists\, g' \in \mathcal{G} : \sup_{\boldsymbol{x}:\|\boldsymbol{x}\| \leq \mathbf{1}} \|h(\boldsymbol{x}) - g'(\boldsymbol{x})\| < \epsilon\right) \geq \frac{1}{2},$$

then $g(\boldsymbol{x})$ has $\Omega(d^2 \log(1/\epsilon))$ parameters. Further if $l = 2$, then the width of $g(\boldsymbol{x})$ is $\Omega(d \log(1/\epsilon))$.

We now state the theorem by [Bur22] which approximates an l layered network using an $l + 1$ layered network. The architecture of a neural such as in Equation 4.1 is the tuple $\bar{n} = (d_0, d_1, \ldots, d_l)$. We will also use the notation $n_1 = d_0$. We now state the result by [Bur22].

Theorem 5. *Assume $\epsilon, \delta \in (0, 1)$, a target network $f_t(x) : \mathbb{R}^{n_0} \to \mathbb{R}^{d_{L_t}}$ with architecture $\bar{n}_t$, L_t layers and N_t number of non zero parameters, and a source network f_s with architecture $\bar{n}_s$ with $L_s = L_t + 1$ layers are given with all parameters sampled uniformly form $[-1, 1]$. Then, with probability at least $1 - \delta$, f_s contains a subnetwork $f_\epsilon \subset f_s$ so that each output component i is approximated as $\max_{\mathbf{x} \in \mathbb{R}^{n_0}} |f_{t,i}(\mathbf{x}) - f_{\epsilon,i}(\mathbf{x})| \leq \epsilon$ if*

$$n_{s,l+1} \geq C n_{t,l} \log\left(\frac{1}{\min\{\epsilon_{l+1}, \delta/\rho\}}\right)$$

for $l > 1$, where $\rho = C N_t^{\gamma+1} \log(1/\min\{(\min_l \epsilon_l), \delta\})$ and ϵ_l is defined as

$$\epsilon_l := \frac{\epsilon}{n_l L_t}\left[(1 + M_{l-1})\left(1 + \frac{\epsilon}{L_t}\right) \prod_{k=l+1}^{L_t - 1}\left(\|W_t^{(k)}\|_\infty + \frac{\epsilon}{L_t}\right)\right]^{-1}.$$

for any $\gamma > 0$. $W_t^{(k)}$ denotes the weight matrix of the target network at layer k and $M_l :=$ $\sup_{\boldsymbol{x} \in \mathcal{D}} \|\boldsymbol{x}^l\|_1$. Furthermore we require $n_{s,1} \geq C n_{t,1} \log\left(\frac{1}{\min\{\epsilon_{l+1}, \delta/\rho\}}\right)$.

Originally, this result by [Bur22] was proved for arbitrary activation functions. Infact, generalization to arbitrary activation functions was one of the major contributions of this work, except for proving an SLTH result with $l + 1$ layers. However, in this thesis, we are only interested in working with ReLU activation function, and hence we have stated this result for ReLU activation only. In the next chapter, we shall adapt constructions in [Pen+20], [Bur22] and prove SLTH results, but starting in a quantized setting. We will see that the construction by [Bur22] is quite powerful in the quantized setting in our framework. It is using this construction we will be able to get rid of undetermined constants (mostly) and provide the exact size of the network required to represent a given target network.

Chapter 5

Lottery Tickets, Weight Quantization and Phase Transition

5.1 Introduction

In chapter 4, we reviewed some literature on the theoretical work on Strong Lottery Ticket Hypothesis (SLTH). All these results assume a large network whose weights are randomly initialized, sampled from some **continuous interval**. Then they try to prune this network in certain ways to approximates a given target network. However computers always represent numbers with finite precision, and hence weights of a neural network are numbers with finite precision, taken from some discrete set. In this chapter, we ask an important question: **Is the precision of with which the weights are represented is related to weather a given large network can be pruned to some target network**. Moreover, quantization is another method for compression of neural networks. It has been observed that the weights of a trained neural network can be quantized to a great extent without significant decrease in performance. In this chapter, we provide a relationship between the precision of weights, and size of a large network that needs to hold so that it can represent a given target network. We prove results answering this question, which we think lay the foundation of future work in the intersection of pruning and quantization of neural networks. One key difference between our and previous results is that we while previous results go for approximating the output of the given network, we go exact an representation. This was done for theoretical simplicity. Generalizing our results to approximate a given network is a challenge for future work. We assume a target network whose weights are sampled from a discrete set like $S_{\delta_1} = \{-1, \ldots, \delta_1, 2\delta_1, \ldots, 1\}$, where $\delta = 10^{-k_1}$ for some $k_1 \in \mathbb{N}$ and a large network whose weights are sampled from say $S_{\delta_2} = \{-1, \ldots, \delta_2, 2\delta_2, \ldots, 1\}$, where $\delta = 10^{-k_2}$, with $k_2 \geq k_1$. The number δ_i defines the precision of the network. The larger network can be **quantized**, i.e., the precision of it's weights can be decreased (by removing less significant digits from after the decimal place: $0.4326 \to 0.43$, for example) and it can then be **pruned**. We then ask what relationship must hold between the precision size of the large network such that it can be pruned the given target with high probability. We further assume that after certain operation, the precision is set to δ again. For example, in a network, we may assume that output from any neuron of each odd layer is of precision δ. This may happen at different stages in the target and the large network, and may vary from result to result. This assumption was made for theoretical simplicity and we leave it to the future work to deal with a more general case. We adapt constructions from [Pen+20] and [Bur22] but starting in a quantized setting, we reduce the problem to solving a bunch of Number Partitioning Problem (NPP), which is an equivalent problem to RSS. NPP is one of the most important problems in the theory of NP-completeness and is known to exhibit a Phase transition [BCP01]. Leveraging the known results on NPP, it is natural to deal with quantized system. In one of our results, the required network size of the large network is exact except for it's first layer, free of any undetermined constants unlike other pervious works. We also provide lower bounds on the required parameter count of a network to represent a two layered network. We do this by a parameter counting argument similar of [Pen+20], but in quantized setting. The upper and lower bounds have the same form asymptotically, indicating optimality of our results.

Statements of Results

In this section, we formally state our results on SLTH and weight quantization. We start by defining some notation.

Notation

We use lowercase letters to represent scalars, such as w, y and so on. We use bold lower-case letters to denote vectors, such as $\mathbf{v}$. The j^{th} component of a vector $\mathbf{v}$ is denoted as v_i. Matrices are denoted by bold upper-case letters such as $\mathbf{M}$. If a matrix $\mathbf{W}$ has dimension $d_1 \times d_2$, then we say $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$. S_δ denotes the set $\{-1, \dots, \delta, 2\delta, \dots, 1\}$ where $\delta = 10^{-k}$ for some $k \in \mathbb{N}$. A number b is said to be of precision if $b \in S_\delta$. We use $C, C_i,\ i \in \mathbb{N}$ to denote positive absolute constants.

5.1.1 Preliminaries and Setup

Definition 7. *An neural network is a function $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l}$ defined as*

$$f(\mathbf{x}) = \mathbf{W}_l \sigma(\mathbf{W}_{l-1} ... \sigma(\mathbf{W}_1 \mathbf{x})), \tag{5.1}$$

where $\mathbf{W}_i$ has dimension $d_i \times d_{i-1}$, $\mathbf{x} \in \mathbb{R}^{d_0}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the nonlinear activation function and $\mathbf{v} = \sigma(\mathbf{x})$ means $v_i = \sigma(x_i)$.

The elements of $\mathbf{W}$'s are called weights or parameters of the network. From here on, all neural networks will have ReLU nonlinear activation, i.e., $\sigma(x) = \max(0, x)$. Our goal is to start with a neural network $f(\mathbf{x})$ and a second larger neural network $g(\mathbf{x})$ of the form

$$g(\mathbf{x}) = \mathbf{M}_{2l} \sigma(\mathbf{M}_{2l-1} ... \sigma(\mathbf{M}_1 \mathbf{x})),$$

and represent f by obtaining a pruned version of g by eliminating its weights

$$\hat{g}(\mathbf{x}) = (\mathbf{S}_{2l} \odot \mathbf{M}_{2l}) \sigma((\mathbf{M}_{2l-1} \odot \mathbf{M}_{2l-1}) ... \sigma((\mathbf{S}_1 \odot \mathbf{M}_1) \mathbf{x})),$$

where each $\mathbf{S}_i$ is a binary (pruning) matrix, with the same dimension as $\mathbf{M}_i$, and $\odot$ represents element-wise product between matrices. In other words, we need to find $\mathbf{S}_i$ such that $f = \hat{g}$.

5.1.2 Quantized SLTH Results

The main question that we want to answer is the following: given a target neural network, whose weights are of precision δ_1 and a large network of precision δ_2, such that $\delta_1 \geq \delta_2$, what is the relationship between δ_2 and size of the large network such that the bigger network can be pruned to the target network. A key difference here is that we are interested in exactly representing a network and not approximating it within an error ϵ, as it is more natural in the discrete setting. We start with some definitions.

Definition 8. *A δ-quantized neural network is a neural network whose weights are sampled uniformly from $S_\delta = \{-1, \ldots, \delta, 2\delta, \ldots, 1\}$ with $\delta = 10^{-k}$ for some $k \in \mathbb{N}$.*

We also have some mixed precision assumption on the target and the large network. For any target network, the output of a neuron cannot be more precise than the precision of it's weights. Such networks will be called networks of Type I. Then we have two other kinds of networks, Type II and Type III. In a Type II network, the output of every odd layer cannot be more precise than the precision of it's weights. In a Type III network, the output of every neuron cannot be more precise than the precision of it's weights, except for the first layer, where output can be of any precision. Type II and Type III networks will serve as large networks. These assumptions were made for theoretical simplicity, and we leave it to future work to deal with the general case.

Definition 9. *A neural network is said to be of Type I if it is of the form*

$$f(\mathbf{x}) = [\mathbf{W}_l[\sigma(\mathbf{W}_{l-1}...[\sigma(\mathbf{W}_1\mathbf{x})]_\delta)]_\delta]_\delta,$$

of Type II if it is of the form

$$f(\mathbf{x}) = [\mathbf{W}_{2l}\sigma(\mathbf{W}_{2l-1}...[\sigma(\mathbf{W_2}(\sigma(\mathbf{W}_1\mathbf{x})))]_\delta)]_\delta,$$

of Type III if it is of the form

$$f(\mathbf{x}) = [\mathbf{W}_{2l}[\sigma(\mathbf{W}_{2l-1}...[\sigma(\mathbf{W_2}(\sigma(\mathbf{W}_1\mathbf{x})))]_\delta)]_\delta]_\delta.$$

where δ is the precision of elements of $\mathbf{W}_i$'s and $[\cdot]_\delta$ operation makes a number of precision δ

by removing sufficient number of insignificant digits.

Henceforth we denote $[\cdot]_\delta$ by $[\cdot]$ as δ is clear from the input of $[\cdot]$ function. Now we state our first main result, which is analogs to the theorem proved by [Pen+20], but in the quantized setting.

Theorem 6. *Let $\mathcal{F}$ be the class of δ_1 quantized neural networks of Type I. Consider a $2l$ layered randomly initialized δ_2-quantized neural network*

$$g(\mathbf{x}) = \mathbf{M}_{2l}\sigma(\mathbf{M}_{2l-1}...\sigma(\mathbf{M}_1\mathbf{x})),$$

of Type II with $\delta_1 \geq \delta_2$. Say the precision of elements of $\mathbf{M}_i$'s is reduced to δ such that $\delta \leq \delta_1$. If $\mathbf{M}_{2i}$ has dimension

$$d_i \times Cd_{i-1}\log_2\frac{1}{\delta_2^2},$$

and $\mathbf{M}_{2i-1}$ has dimension

$$Cd_{i-1}\log_2\frac{1}{\delta^2} \times d_{i-1}.$$

Then for every $f \in \mathcal{F}$,

$$\exists\, \mathbf{S}_i \in \{0,1\}^{d_i \times d_{i-1}}: \quad (\mathbf{S}_{2l}\odot\mathbf{M}_{2l})\sigma((\mathbf{S}_{2l-1}\odot\mathbf{M}_{2l-1})...\sigma((\mathbf{S}_1\odot\mathbf{M}_1)\mathbf{x})) = f(\mathbf{x}).$$

with probability

$$1 - N_t\,\mathcal{O}\left(\left(\log_2\frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right)$$

where N_t is the total number of parameters in f.

Our next theorem employs construction from [Bur22]. Note the absence of arbitrary constants (except the first layer) in the required size of the large network.

Theorem 7. *Let $\mathcal{F}$ be the class of δ_1 quantized neural networks of Type I. Consider an $l+1$ layered randomly initialized neural network*

$$g(\mathbf{x}) = \mathbf{M}_{l+1}\sigma(\mathbf{M}_l...\sigma(\mathbf{M}_1\mathbf{x})),$$

of Type III whose weights are sampled from $\{-1\ldots,-\delta,\delta,\ldots,1\}$ with $\delta_2 \leq \delta_1$. Say the precision *elements of* $\mathbf{M}_i$*'s is reduced to* δ *such that* $\delta \leq \delta_1$*. If* $\mathbf{M}_1$ *and* $\mathbf{M}_2$ *has dimensions*

$$d_0 \times Cd_0 \log_2 \frac{1}{\delta^2} \qquad \text{and} \qquad 2d_1 \log_2 \frac{1}{\delta^2} \times Cd_0 \log_2 \frac{1}{\delta^2}$$

respectively, $\mathbf{M}_{i+1}$ *has dimension greater than*

$$2d_i \log_2 \frac{1}{\delta^2} \times 2d_{i+1} \log_2 \frac{1}{\delta^2}$$

$\forall\, 2 < i < l-1$ *and* M_{l+1} *has dimension*

$$2 \log_2 \left(\frac{1}{\delta^2}\right) d_{l-1} \times d_l.$$

Then for every $f \in \mathcal{F}$ *we have*

$$\exists\, \mathbf{S}_i \in \{0,1\}^{d_i \times d_i - 1} : \quad (\mathbf{S}_{l+1} \odot \mathbf{M}_{l+1})\sigma((\mathbf{S}_l \odot \mathbf{M}_l)...\sigma((\mathbf{S}_1 \odot \mathbf{M}_1)\mathbf{x})) = f(\mathbf{x}).$$

with probability

$$1 - N_t \; 2\log_2 \left(\frac{1}{\delta^2}\right) \; \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right)$$

where N_t *is the total number of parameters in* f*.*

5.1.3 Lower Bound by Parameter Counting

We now state the result on lower bound, where we show using parameter counting argument that a there exists a two layered δ quantized network with d^2 parameters that cannot be represented by a neural network with unless it has $\Lambda\left(d^2 \log_2\left(\frac{1}{\delta}\right)\right)$ parameters. Note that any linear transformation $\mathbf{Wx}$ where $\mathbf{W} \in \mathbb{R}^d \times \mathbb{R}^d$ and $\mathbf{x} \in \mathbb{R}^d$ can be expressed as a 2 layered neural network. Let $\mathcal{F}$ be the class of functions

$$\mathcal{F} := \{h_{\mathbf{W}} : \mathbf{W} \in \mathbb{R}^d \times \mathbb{R}^d\}, \qquad \text{where} \qquad h_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix} \sigma\left(\begin{bmatrix}\mathbf{W} \\ -\mathbf{W}\end{bmatrix}\mathbf{x}\right). \qquad (5.2)$$

Theorem 8. *Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be a δ quantized neural network of the form*

$$g(\mathbf{x}) = \mathbf{M}_l \sigma(\mathbf{M}_{l-1} ... \sigma(\mathbf{M}_1 \mathbf{x})),$$

where elements of $\mathbf{M}_i$'s are sampled from arbitrary distributions over S_δ. Let α be the total number of non-zero parameters in g. Let $\mathcal{G}$ be the set of all matrices that can be formed by pruning g. Let $\mathcal{F}$ be defined as in Eq. 5.2. Then if

$$\forall\, h \in \mathcal{F},\ \mathbb{P}\left(\exists\, g' \in \mathcal{G} : g' = h\right) \geq p,$$

then we must have

$$\alpha \geq \log_2 p + d^2 \log_2 \left(\frac{2}{\delta} + 1\right).$$

We see that there the lower bound in Theorem 8 and upper bound in Theorem 6 have the same fore asymptotically the same i.e., $Cm \log\left(\frac{1}{\delta}\right)$, where m is the number of parameters in the target. Hence we have showed that atleast for exact representation of network, and for this simple case of two layered network, our solution is optimal in the asymptotic sense. Generalizing this is another challenge for the future work.

5.1.4 Random Subset Sum Problem

The random subset sum problem (RSSP) is the problem of finding a subset of a given set such that the sum of this subset equals a given target t. RSSP is an important tool in proving results on SLTH [Pen+20] [Bur22]. RSSP and NPP are closely related, and hence we can use the results in this section to make statements in RSSP. We shall then use these results on RSSP to prove results on SLTH and quantization.

Definition 10. *Let $\boldsymbol{X} = (X_1, X_2, \dots, X_n)$ be a set of integers sampled uniformly from the set $\{-M, \dots, 1, 2, 3, \dots, M\}$. The Random Subset Sum Problem is defined as the problem of finding an index set $S \subset [n]$ such that $\sum_{i \in S} X_i = t$ for a given integer t, called the target.*

Lemma 2. *Consider a Random subset sum problem on the set $\mathbf{X} = (X_1, X_2, \dots, X_{Cn})$ sampled from an arbitrary distribution with support $\{-M, \dots -1, 1, \dots, M\}$ with a target $t = \mathcal{O}(M)$.*

- *There exists C such that n samples out of the Cn will be uniformly distributed on $\{-M, \dots, -1, 1, \dots, M\}$ with high probability. Let's relabel these uniform samples as $(X_1, X_2, \dots, X_n)$. This defines a new subset sum problem with set $(X_1, X_2, \dots, X_n)$ and target t.*

- *Let $Y_{n,t}$ be the number of possible solutions to this new problem. Then*

$$\mathbb{P}(Y_{n,t} > 0) \leq \begin{cases} \rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2 c_M} \right) + \frac{C_1}{n} \right) & \textit{if } \ell = 0 \\ 2\rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2 c_M} \right) + \frac{C_1}{n} \right) & \textit{if } \ell \neq 0, \end{cases}$$

$$\mathbb{P}(Y_{n,t} > 0) \geq \frac{1}{\left(1 + \exp\left(\frac{\ell^2}{nM^2 c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)},$$

where

$$\ell = \Lambda - 2t, \qquad \Lambda = \sum_{i=1}^{n} X_i,$$
$$\gamma_n = \frac{1}{M\sqrt{2\pi n c_M}}, \qquad c_M = \mathbb{E}\left(\frac{X^2}{M^2}\right) = \frac{1}{3} + \frac{1}{2M} + \frac{1}{6M^2}.$$

Lemma 3. *A random subset sum problem with given set $\mathbf{X} = (X_1, X_2, ..., X_n)$ and target t can be solved iff the number partitioning problem can be solved with the given set $\mathbf{X}$ and target $\Lambda - 2t$ (or $2t - \Lambda$), where $\Lambda = \sum_{i=1}^{n} X_i$.*

Proof. First of all notice that a Number Partitioning Problem on a set of numbers $\mathbf{X} = (X_1, X_2, ..., X_n)$ sampled uniformly from the set $\{-M, .., 1, 2, .., M\}$ can be solved iff the Number Partitioning Problem on a set of numbers $\mathbf{X} = (|X_1|, |X_2|, ..., |X_n|)$ sampled uniformly from the set $\{0, 1, 2, .., M\}$ can be solved for any given target. This is because, first, it is obvious that $\{X_i\}_{i=1}^{n}$ is distributed uniformly over $\{0, 1, 2, .., M\}$, and secondly, the number partitioning problem does not care about the signs of the numbers, a sign can always be absorbed in the σ_i while solving the number partitioning problem.

We have a random subset sum problem with set $\mathbf{X}$ and target t. Assume number partitioning problem can be solved with the given set $\mathbf{X}$ and target $\Lambda - 2t$. Notice that NPP does not care about the sign of the target, as an NPP with target k can be solved iff an NPP with target $-k$ can be solved. WLOG assume there exists two partitions of $\mathbf{X}$, with sums be x and $\Lambda - x$

such that $(\Lambda - x) - x = \Lambda - 2t \implies x = t$. Hence one of the subsets must sum up to t, so the random subset sum problem is solved. It also follows that if this constructed number partitioning problem cannot be solved, then the given random subset sum problem can also not be solved. $\square$

Proof of Lemma 2. The first part of the lemma is obvious from rejection sampling methods (See Appendix C), we will prove the second. Considers the number partitioning problem corresponding to the given random subset sum problem as given by Lemma 3. The target of this number partitioning problem is $\ell = \Lambda - 2t$. Consider $\ell \neq 0$. Note that if Λ is even (event denoted by $\mathscr{E}_n$), then ℓ is also even and if Λ is odd (event denoted by $\mathscr{O}_n$), then ℓ is also odd. The probability that the random subset sum problem can be solved can be written in terms of the probability that the number partitioning problem can be solved

$$\mathbb{P}(Y_{n,t} > 0) = \mathbb{P}(\mathscr{E}_n)\mathbb{P}(Z_{n,\ell} > 0|\mathscr{E}_n) + \mathbb{P}(\mathscr{O}_n)\mathbb{P}(Z_{n,\ell} > 0|\mathscr{O}_n)$$

If ℓ is even, then

$$\mathbb{P}(Z_{n,\ell} > 0) = \mathbb{P}(\mathscr{E}_n)\mathbb{P}(Z_{n,\ell} > 0|\mathscr{E}_n).$$

If ℓ is odd, then

$$\mathbb{P}(Z_{n,\ell} > 0) = \mathbb{P}(\mathscr{O}_n)\mathbb{P}(Z_{n,\ell} > 0|\mathscr{O}_n).$$

But on $\mathscr{E}_n$, ℓ is always even and on $\mathscr{O}_n$, ℓ is always odd. Hence $\mathbb{P}(Y_{n,t} > 0)$ can be written as

$$\mathbb{P}(Y_{n,t} > 0) = 2\mathbb{P}(Z_{n,\ell} > 0).$$

From Lemma 1 it follows that

$$\mathbb{P}(Y_{n,t} > 0) \leq 2\rho_n \left(\exp\left(-\frac{\ell^2}{2nM^2c_M}\right) + \frac{C_1}{n}\right),$$

$$\mathbb{P}(Y_{n,t} > 0) \geq \frac{1}{\left(1 + \exp\left(\frac{\ell^2}{nM^2c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)}.$$

Same can be done for $\ell = 0$. $\square$

Lemma 4. *Let $M = M(n)$ be an arbitrary function of n. Consider a Random subset sum problem on the set $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ sampled uniformly from $\{-M, \ldots, -1, 1, \ldots, M\}$*

with a target $t = \mathcal{O}(M)$. If

$$\kappa_n = \lim_{n\to\infty} \frac{\log_2 M}{n} < 1,$$

then we have

$$\mathbb{P}(Y_{n,t} > 0) = 1 - \mathcal{O}\left(\frac{1}{n^{\frac{1}{7}}}\right).$$

Proof of Lemma 4. We are given that $\lim_{n\to\infty} \kappa_n$ exists and is less than 1. Consider a more sensitive parametrization

$$\kappa_n = 1 - \frac{\log_2 n}{2n} + \frac{\lambda_n}{n} \qquad \text{or} \qquad M = \frac{2^{n+\lambda_n}}{\sqrt{n}}.$$

In this parametrization $\lim_{n\to\infty} \kappa_n < 1$ means $\lim_{n\to\infty} \lambda_n \to -\infty$. Note that in this regime $\rho_n \to \infty$. Now we have

$$\mathbb{P}(Y_{n,t} > 0) \geq \frac{1}{\left(1 + \exp\left(\frac{\ell^2}{nM^2 c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)}$$
$$\implies \mathbb{P}(Y_{n,t} > 0) \geq \frac{1}{\left(1 + \exp\left(\frac{(\Lambda-2t)^2}{nM^2 c_M}\right)\left(\left(\frac{C_2}{n\rho_n}\right) + \left(\frac{C_3}{n}\right)\right) + \frac{1}{\rho_n}\right)}.$$

Now $t = \mathcal{O}(M)$ and assume

$$\Lambda < \frac{1}{\sqrt{3+\beta}} M\sqrt{n\log n}.$$

According to Hoeffding's inequality B, that happens with probability

$$\mathbb{P}\left(\Lambda < \frac{1}{\sqrt{3+\beta}} M\sqrt{n\log n}\right) \geq 1 - \exp\left(\frac{\frac{1}{3+\beta} M^2 n\log n}{4nM^2}\right)$$
$$\implies \mathbb{P}\left(\Lambda < \frac{1}{\sqrt{3+\beta}} M\sqrt{n\log n}\right) \geq 1 - \frac{1}{n^{\frac{1}{2(3+\beta)}}}.$$

Now as $\rho_n \to \infty$, we have

$$\mathbb{P}(Y_{n,t} > 0) \geq \frac{1}{\left(1 + \frac{C_3}{n^{\frac{\beta}{3+\beta}}}\right)}$$
$$\implies \mathbb{P}(Y_{n,t} > 0) \geq 1 - \mathcal{O}\left(\frac{1}{n^{\frac{\beta}{3+\beta}}}\right).$$

Hence the probability (say $\mathbb{P}(E)$) of events $P(Y_{n,t} > 0)$ and $\Lambda < \frac{1}{\sqrt{3+\beta}} M\sqrt{n\log n}$ happening together is given by

$$\mathbb{P}(E) = \left(1 - \mathcal{O}\left(\frac{1}{n^{\frac{\beta}{3+\beta}}}\right)\right)\left(1 - \frac{1}{n^{\frac{1}{2(3+\beta)}}}\right).$$

Note that this probability will converge to 1 fastest if $\beta = \frac{1}{2}$. Hence we choose $\beta = \frac{1}{2}$ and we get

$$\mathbb{P}(E) = \left(1 - \mathcal{O}\left(\frac{1}{n^{\frac{1}{7}}}\right)\right).$$

□

Lemma 4 basically tells us under what condition can a RSSP be solved with high probability. It is important to note here that by **high probability**, we mean probability converges to 1 as the size of the set goes to ∞. Now we shall leverage this result to prove statements on SLTH in quantized setting.

5.2 Proving Results on SLTH and Weight Quantization

5.2.1 The Pensia Construction

Now we provide the proofs of various results stated in Section 5.1.2. We start with Theorem 6. We follow the strategy of [Pen+20] to reduce the problem of pruning (or setting weights to zero) in order to represent a target network to a bunch of random subset sum problems.

Lemma 5 (Representing a Single weight)**.** *Let $g : \mathbb{R} \to \mathbb{R}$ be a randomly initialized δ_2 quantized network of the form $g(x) = [\mathbf{v}^T\sigma(\mathbf{u}x)]$ where $\mathbf{u}, \mathbf{v}, \in \mathbb{R}^{2n}$. Assume $\delta_2 \leq \delta_1$. Say the precision of weights of g is reduced to δ such that $\delta \leq \delta_1$. If $n > C \log \frac{1}{\delta^2}$, then with probability*

$$1 - \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right),$$

we have

$$\forall\, w \in S_{\delta_1}, \quad \exists\, \mathbf{s} \in \{0,1\}^{2n} : [wx] = [(\mathbf{v} \odot \mathbf{s})^T\sigma(\mathbf{u}x)].$$

where $[\cdot]_\delta$ is the operation which reduces the precision of a number to δ.

Proof. Let the precision of g be $\delta \leq \delta_1$. First decompose $wx = \sigma(wx) - \sigma(-wx)$. WLOG say

$w > 0$. Let

$$\mathbf{v} = \begin{bmatrix} \mathbf{b} \\ \mathbf{d} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \mathbf{a} \\ \mathbf{c} \end{bmatrix}, \mathbf{s} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix},$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^n, \mathbf{s}_1, \mathbf{s}_2 \in \{0,1\}^n$. This is shown diagrammatically in Figure 5.1.

Step 1: Let $\mathbf{a}^+ = \max\{0, a\}$ be the vector obtained by pruning all the negative entries of $\mathbf{a}$. Since $w \geq 0$, then for all $x \leq 0$ we have $\sigma(wx) = \mathbf{b}^T\sigma(\mathbf{a}^+ x) = 0$. Moreover, further pruning of $\mathbf{a}^+$ would not affect this equality for $x \leq 0$. Thus we consider $x > 0$ in next two steps. Therefore we get $\sigma(wx) = wx$ and $\mathbf{b}^T\mathbf{a}^+ x = \sum_i b_i a_i^+ x$.

Step 2: Consider the random variables $Z_i = b_i a_i^+$. These are numbers of precision δ^2, sampled from the set $S_\delta \times S_\delta$. Now w, which is a number of precision δ_1, belongs to the set $S_\delta \times S_\delta$, as δ is of the form 10^{-k} for $k \in \mathbb{N}$ and $\delta \leq \delta_1$. Now by Lemma 2 and 4, as long as $n > C \log_2 \frac{1}{\delta^2}$, the subset sum problem with set $\{Z_i\}$ and target w can be solved with probability

$$p = 1 - \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right).$$

Note that solving subset in an integer setting where numbers are sampled from $\{-M, \ldots, M\}$ and numbers of precision δ setting is equivalent with $\delta = \frac{1}{M}$. Hence it follows that with probability p

$$\forall\, w \in S_\delta^+, \quad \exists\, \mathbf{s}_1 \in \{0,1\}^n : w = \mathbf{b}^T\mathbf{s}_1 \odot \mathbf{a}^+.$$

where S_δ^+ denotes positive members of S_δ. The part shown in green in Figure 5.1 hence handles positive inputs.

Step 3: Similar to steps 1 and 2, we can prune negative weights from $\mathbf{c}$ and let the red part shown in figure handle negative inputs. It will follow that

$$\forall\, w \in S_\delta^+, \quad \exists\, \mathbf{s}_2 \in \{0,1\}^n : w = \mathbf{d}^T\mathbf{s}_2 \odot \mathbf{c}^-.$$

with probability p. Hence Lemma 5 follows. $\square$

Lemma 6 (Representing a single Neuron). *Consider a randomly initialized δ_2 quantized neural network $g(\boldsymbol{x}) = [\mathbf{v}^T\sigma(\mathbf{M}\boldsymbol{x})]$ with $\boldsymbol{x} \in \mathbb{R}^d$. Assume $\delta_2 \leq \delta_1$. Say the precision of weights of g is*

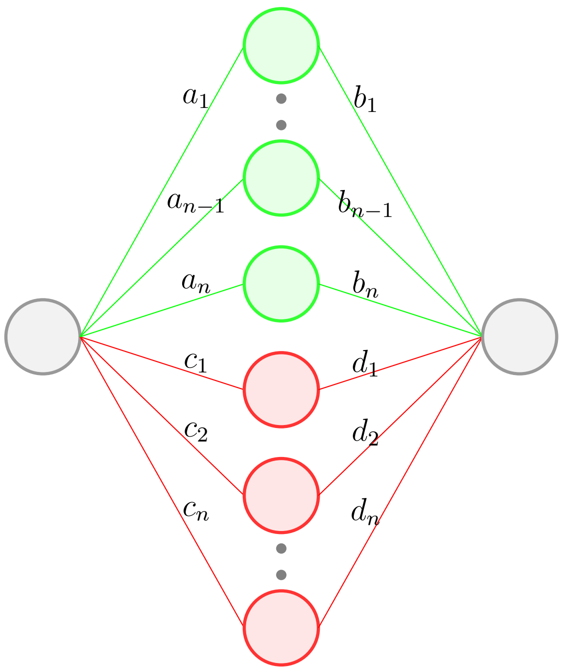

Figure 5.1: Approximating a single weight with ReLU activation

reduced to δ *such that* $\delta \leq \delta_1$. *Let* $\hat{g}(\boldsymbol{x}) = [(\mathbf{s} \odot \mathbf{v})^T \sigma((\mathbf{T} \odot \mathbf{M})\boldsymbol{x})]$ *be the pruned network for a choice of binary vector* $\mathbf{s}$ *and matrix* $\mathbf{T}$. *Let* $f_{\mathbf{w}}(\boldsymbol{x}) = [\mathbf{w}^T \boldsymbol{x}]$ *be a single layered* δ_1 *quantized network. If* $\mathbf{M} \in \mathbb{R}^{Cd \log_2 \frac{1}{\delta^2} \times d}$ *and* $\mathbf{v} \in \mathbb{R}^{Cd \log_2 \frac{1}{\delta^2}}$, *then with probability*

$$1 - d\, \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right),$$

we have

$$\forall \mathbf{w} \in S_\delta^d \ \exists \ \mathbf{s}, \mathbf{T} : \ f_{\mathbf{w}}(\boldsymbol{x}) = \hat{g}(\boldsymbol{x}).$$

Proof. Assume weights of g are of precision δ. We prove the required results by representing each weight of the neuron using 5 (See Figure 5.2).

Step 1: We first prune $\mathbf{M}$ to create a block-diagonal matrix $\mathbf{M}'$. Specifically, we create M by only keep the following non-zero entries:

$$\begin{bmatrix} \mathbf{u}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{u}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_d \end{bmatrix}, \qquad \text{where } \ \mathbf{u}_i \in \mathbb{R}^{C \log_2 \frac{1}{\delta^2}}.$$

We choose the binary matrix $\mathbf{T}$ to be such that $\mathbf{M}' = \mathbf{T} \odot \mathbf{M}$. We also decompose $\mathbf{v}$ and $\mathbf{s}$ as

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_d \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_d \end{bmatrix}, \quad \text{where } \ \mathbf{s}_i, \mathbf{v}_i \in \mathbb{R}^{C \log_2 \frac{1}{\delta^2}}.$$

Step 2: Consider the event

$$E_i : \quad [w_i x_i] = [(\mathbf{v}_i \odot \mathbf{s}_i)^T \sigma(\mathbf{u_i} x_i)].$$

According to Lemma 5, this event happens with probability

$$p = 1 - \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2} \right)^{-\frac{1}{7}} \right).$$

The event (E) in Lemma 6 corresponds with the intersection of these events $E = \cap_{i=1}^{d} E_i$. By taking a union bound B EquationB.4, E happens with a probability $dp - (d-1)$, which is equal to

$$1 - d\, \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2} \right)^{-\frac{1}{7}} \right).$$

The process is illustrated in Figure 5.2. Note that we want $> \log_2(\frac{1}{\delta^2})$ samples to be assured that a RSSP is solved with high probability, but we include that in the constant C. □

Lemma 7 (Representing a single layer)**.** *Consider a randomly initialized δ_2 quantized two layer neural network of the form $g(\mathbf{x}) = [\mathbf{N}\sigma(\mathbf{M}\mathbf{x})]$ with $x \in \mathbb{R}^{d_1}$. Assume $\delta_2 \leq \delta_1$. Say the precision of weights of g is reduced to to δ such that $\delta \leq \delta_1$. Let $\hat{g}(x) = [(\mathbf{S} \odot \mathbf{N})^T \sigma((\mathbf{T} \odot \mathbf{M})\mathbf{x})]$ be the pruned network for a choice of pruning matrices $\mathbf{S}$ and $\mathbf{T}$. Let $f_{\mathbf{W}}(\mathbf{x}) = [\mathbf{W}\mathbf{x}]$ be a single*

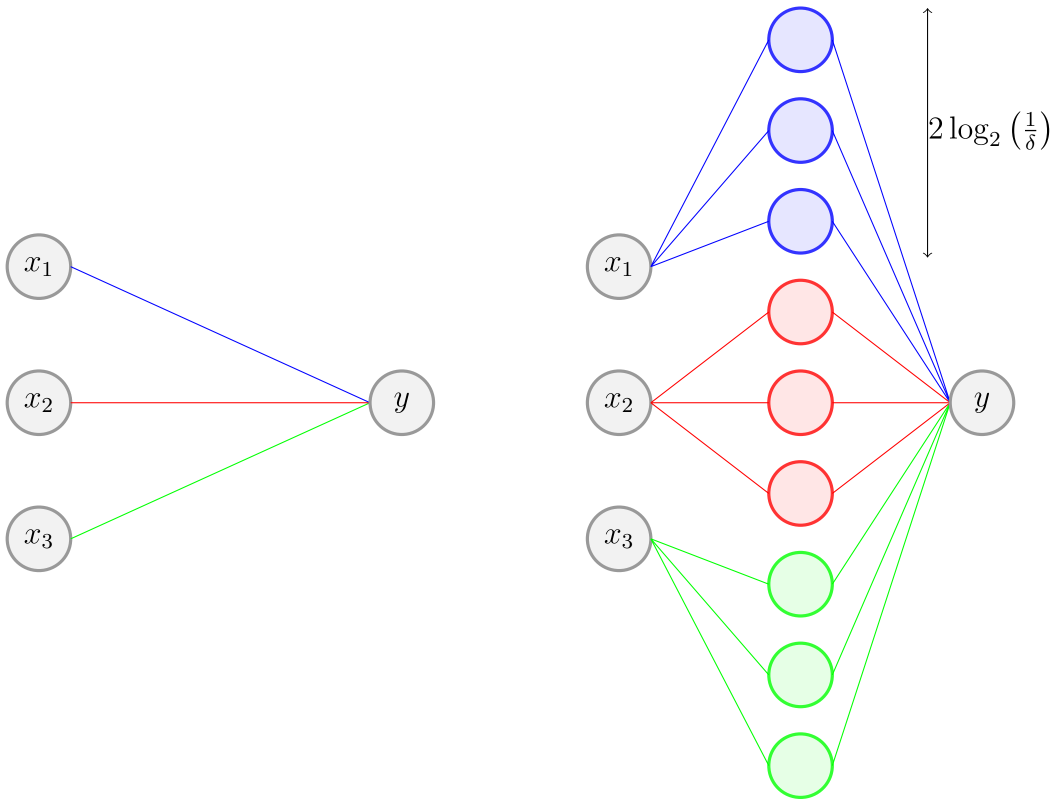

Figure 5.2: Representing a single neuron: Figure on the left shows the target network, where as Figure on the right shows the large network. The colors indicate which part in the target is represented by which part of the source. For example, the red weight on the left is represented by the red subnetwork on the right.

layered network of precision δ_1. *If* $\mathbf{N}$ *has dimension* $d_2 \times Cd_1 \log_2 \frac{1}{\delta^2}$ *and* $\mathbf{M}$ *has dimension* $Cd_1 \log_2 \frac{1}{\delta^2} \times d_1$ *then with probability*

$$1 - d_1 d_2\, \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right),$$

$$\forall\, \mathbf{W} \in S_{\delta_1}^{d_1 \times d_2}\ \ \exists\ \mathbf{S}, \mathbf{T} :\ f_{\mathbf{W}}(\mathbf{x}) = \hat{g}(x).$$

Proof. Assume weights of g are of precision δ. We first prune $\mathbf{M}$ to get a block diagonal matrix $\mathbf{M}'$

$$\mathbf{M}' = \begin{bmatrix} \mathbf{u}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{u}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_{d_1} \end{bmatrix}, \qquad \text{where } \ \mathbf{u}_i \in \mathbb{R}^{C \log_2 \frac{1}{\delta^2}}.$$

Thus, $\mathbf{T}$ is such that $\mathbf{M}' = \mathbf{T} \odot \mathbf{M}$. We also decompose $\mathbf{N}$ and $\mathbf{S}$ as following

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_{1,1}^T & \cdots & \mathbf{s}_{1,d_1}^T \\ \mathbf{s}_{2,1}^T & \cdots & \mathbf{s}_{2,d_1}^T \\ \vdots & \ddots & \vdots \\ \mathbf{s}_{d_2,1}^T & \cdots & \mathbf{s}_{d_2,d_1}^T \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} \mathbf{v}_{1,1}^T & \cdots & \mathbf{v}_{1,d_1}^T \\ \mathbf{v}_{2,1}^T & \cdots & \mathbf{v}_{2,d_1}^T \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{d_2,1}^T & \cdots & \mathbf{v}_{d_2,d_1}^T \end{bmatrix}, \quad \text{where } \mathbf{v}_{i,j}, \mathbf{s}_{i,j} \in \mathbb{R}^{C \log_2 \frac{1}{\delta^2}}.$$

Now note that pruning $\mathbf{u}_i$ and $\mathbf{v}_{i,j}$ (using $\mathbf{s}_{i,j}$) is equivalent to Lemma 6. Hence it's simply an application of Lemma 5 $d_1 d_2$ times. Hence the event in assumption of Lemma 7 occurs with a probability $d_1 d_2 p - (d_1 d_2 - 1)$, by a union bound B, Equation B.4, which is equal to

$$1 - d_1 d_2 \, \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right).$$

The process is illustrated in Figure 5.3 Note that we want $> \log_2(\frac{1}{\delta^2})$ samples to be assured that a RSSP is solved with high probability, but we include that in the constant C. □

Proof of Theorem 6. Now we can see that Theorem 6 can be proved by applying Lemma 7 layer wise, where two layers of the large network represent one layer of the target. Let the total number of parameters in the target network be N_t, i.e.,

$$N_t = \sum_{i=1}^{l-1} d_i d_{i+1}.$$

Then the event in assumption of Theorem 6, by union bound B, Equation B.4, occurs with a probability $N_t p - (N_t - 1)$, where which is equal to

$$1 - N_t \, \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right).$$

□

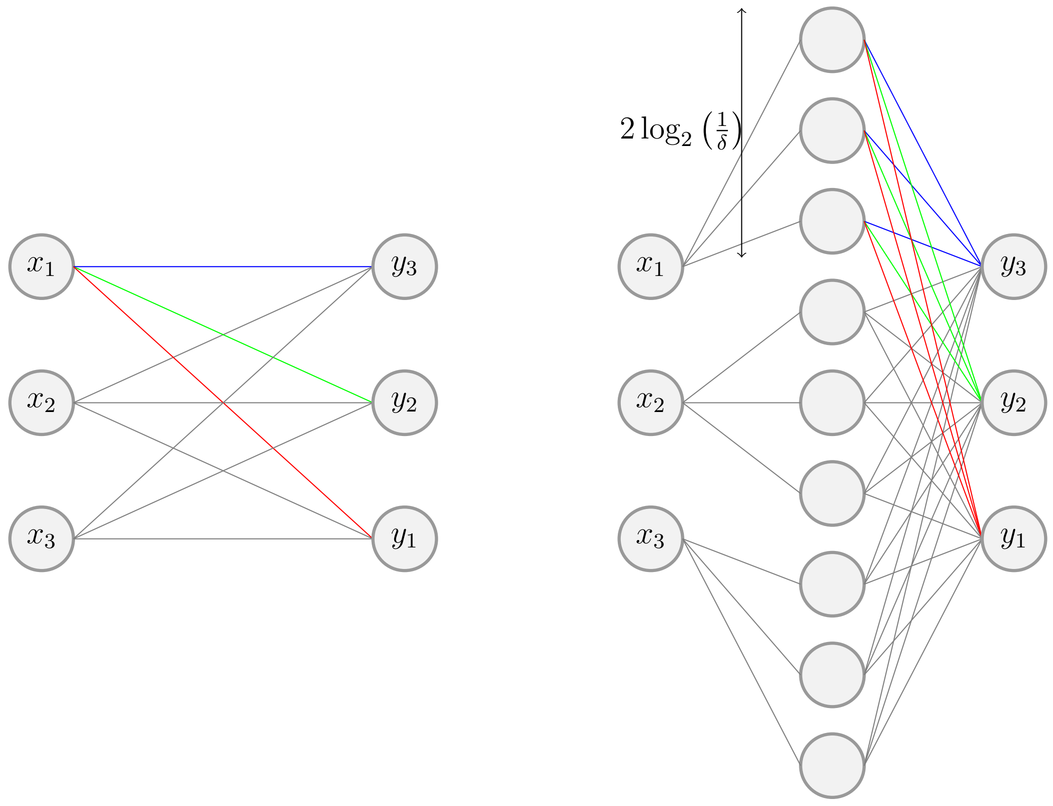

Figure 5.3: Representing a Layer: Figure on the left shows the target network, where as Figure on the right shows the large network. The colors indicate which part in the target is represented by which part of the source. For example, the red weight on the left is represented by the red weights on the right.

5.2.2 The Brukholz Construction

In this section, we adapt construction by [Bur22] to prove Theorem 7. The process is illustrated in Figure 5.4.

Lemma 8. *Consider a randomly initialized δ_2 quantized two layered neural network $g(\mathbf{x}) = \mathbf{N}\sigma(\mathbf{M}\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^{d_1}$, whose weights are sampled uniformly from $\{-1, \ldots, -\delta, \delta, \ldots, 1\}$. Assume $\delta_2 \leq \delta_1$. Say the precision of weights of g is reduced to to δ such that $\delta \leq \delta_1$. Let $\hat{g}(x) = [(\mathbf{S} \odot \mathbf{N})^T \sigma((\mathbf{T} \odot \mathbf{M})\mathbf{x})]$ be the pruned network for a choice of pruning matrices $\mathbf{S}$ and $\mathbf{T}$. Let*

$$f_{\mathbf{W}}(\mathbf{x}) = \begin{bmatrix} [\mathbf{W}\mathbf{x}] \\ [\mathbf{W}\mathbf{x}] \\ \vdots \\ [\mathbf{W}\mathbf{x}] \end{bmatrix}$$

is a single layered network, δ_1 quantized network where $\mathbf{Wx}$ is repeated $2\log_2(\frac{1}{\delta^2})$ times and $\mathbf{W}$ has dimension $d_1 \times d_2$. If $\mathbf{N}$ has dimension $2d_2 \log_2 \frac{1}{\delta^2} \times Cd_1 \log_2 \frac{1}{\delta^2}$ and $\mathbf{M}$ has dimension $Cd_1 \log_2 \frac{1}{\delta^2} \times d_1$ then with probability

$$1 - 2d_1 d_2 \log_2\left(\frac{1}{\delta^2}\right) \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right).$$

we have

$$\forall\, \mathbf{W} \in S_\delta^{d_1 \times d_2} \;\; \exists\; \mathbf{S}, \mathbf{T} : \; f_{\mathbf{W}}(\mathbf{x}) = \hat{g}(x).$$

Proof. Assume weights of g are of precision δ. We first prune $\mathbf{M}$ to get a block diagonal matrix $\mathbf{M}'$

$$\mathbf{M}' = \begin{bmatrix} \mathbf{u}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{u}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{u}_{d_1} \end{bmatrix}, \qquad \text{where } \; \mathbf{u}_i \in \mathbb{R}^{C \log_2 \frac{1}{\delta}}.$$

Thus, $\mathbf{T}$ is such that $\mathbf{M}' = \mathbf{T} \odot \mathbf{M}$. We also decompose $\mathbf{N}$ and $\mathbf{S}$ as following

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \vdots \\ \mathbf{S}_{2\log_2(\frac{1}{\delta^2})} \end{bmatrix} \qquad\qquad \mathbf{N} = \begin{bmatrix} \mathbf{N}_1 \\ \mathbf{N}_2 \\ \vdots \\ \mathbf{N}_{2\log_2(\frac{1}{\delta^2})} \end{bmatrix}$$

where

$$\mathbf{S}_k = \begin{bmatrix} (\mathbf{s}_{1,1}^T)_k & \cdots & (\mathbf{s}_{1,d_1}^T)_k \\ (\mathbf{s}_{2,1}^T)_k & \cdots & (\mathbf{s}_{2,d_1}^T)_k \\ \vdots & \ddots & \vdots \\ (\mathbf{s}_{d_2,1}^T)_k & \cdots & (\mathbf{s}_{d_2,d_1}^T)_k \end{bmatrix}, \qquad \mathbf{N} = \begin{bmatrix} (\mathbf{v}_{1,1}^T)_k & \cdots & (\mathbf{v}_{1,d_1}^T)_k \\ (\mathbf{v}_{2,1}^T)_k & \cdots & (\mathbf{v}_{2,d_1}^T)_k \\ \vdots & \ddots & \vdots \\ (\mathbf{v}_{d_2,1}^T)_k & \cdots & (\mathbf{v}_{d_2,d_1}^T)_k \end{bmatrix},$$

and $(\mathbf{v}_{i,j})_k, (\mathbf{s}_{i,j})_k \in \mathbb{R}^{C \log_2 \frac{1}{\delta^2}}$.

Now note that pruning $\mathbf{u}_i$ and $(\mathbf{v}_{i,j})_k$ (using $(\mathbf{s}_{i,j})_k$) is equivalent to Lemma 6. Hence it's simply an application of Lemma 5 $d_1 d_2 2\log_2\left(\frac{1}{\delta^2}\right)$ times. Hence the event in assumption of Lemma 8 occurs with a probability

$$1 - 2d_1 d_2 \log_2\left(\frac{1}{\delta^2}\right) \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right),$$

using the union bound B, Equation B.4. □

Proof of Theorem 7. Theorem 7 is an application of second step of Lemma 8 for all layers until we reach the last layer, where copying is not required. The process is illustrated in Figure 5.3. The event in the assumption of Theorem 7 happens with probability

$$1 - N_t \, 2\log_2\left(\frac{1}{\delta^2}\right) \mathcal{O}\left(\left(\log_2 \frac{1}{\delta^2}\right)^{-\frac{1}{7}}\right)$$

□

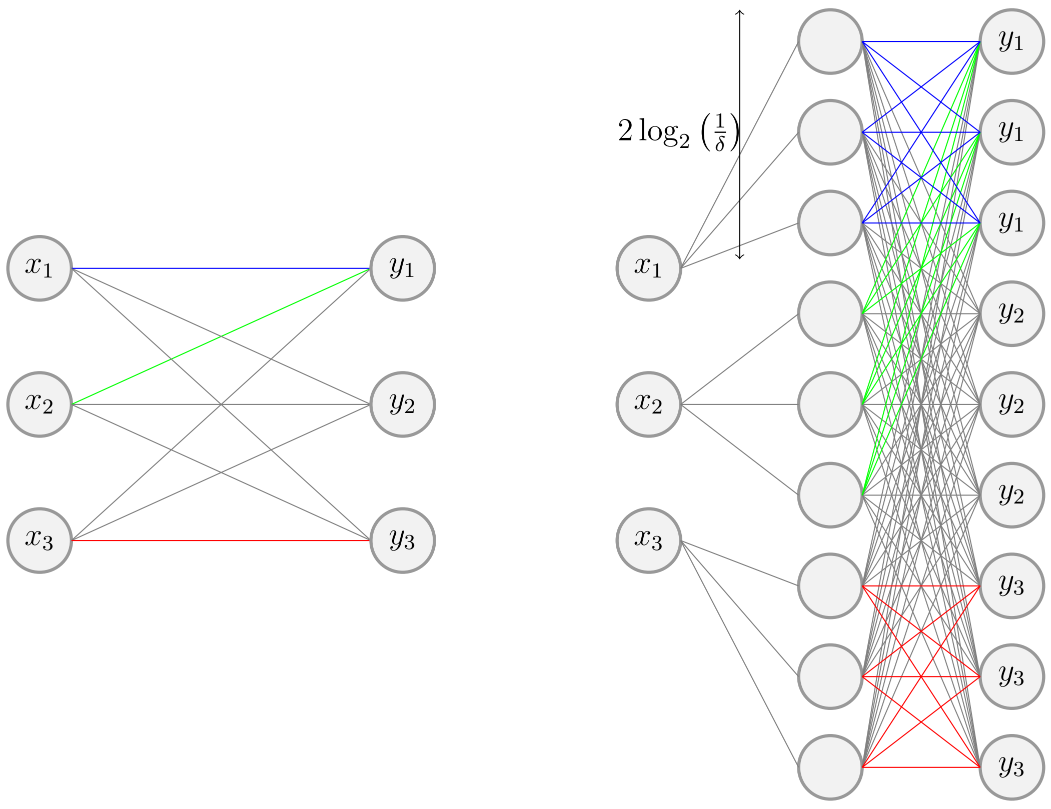

Figure 5.4: The Figure shows representation of first two layers of a network in Theorem 7. Figure on the left shows the target network, where as Figure on the right shows the large network. The colors indicate which part in the target is represented by which part of the source. For example, the red weight on the left is represented by the red weights on the right.

Lower Bound by Parameter Counting

Here we prove Theorem 8 which follows by a parameter counting in the discrete setting.

Proof of Theorem 8. Two matrices represent the same function iff all their elements are the same. Therefore, the number of functions in $\mathcal{F}$ is

$$\left(\frac{2}{\delta}+1\right)^{d^2}.$$

Also, the number of functions in $\mathcal{G}$ is 2^α. Now for the assumption of Theorem 8 to hold, we must have

$$\begin{aligned} 2^\alpha &\geq p\left(\frac{2}{\delta}+1\right)^{d^2} \\ \implies \alpha &\geq \log_2 p + d^2 \log_2\left(\frac{2}{\delta}+1\right). \end{aligned}$$

$\square$

5.3 Future Directions

5.3.1 More on Weight Quantization

The applications of theoretical results on NPP goes well beyond what we have covered in this thesis. For example, if Theorem 2 can be proved for arbitrary distribution rather than uniform distribution, then we can get rid of the undetermined constant in Theorem 6 which would make our bound much better. This requires estimating the moment integrals in Theorem 2 for different distribution, which is a challenge for future works. Also we conjecture the existence of a **phase transition in SLTH**. Given a large network and a target network, there exists a parameter, which depends on the size and the precision of weights of the large network. The probability of approximating/representing the target network shows a sharp jump around a critical value of this parameter in the limit of large networks. Proving this conjecture requires finding the optimal condition for approximating/representing the target network with a given network, which is again a challenge for future works. Finally, one of the criticisms of this work can be - Why go for exact representation, when one only cares about an ϵ approximation. Handling approximation in the discrete setting is difficult, but we think can be done using the analysis of NPP on [BCP01]. We also leave that to future work. Overall, we have shown that theoretical results on NPP have the potential to serve as extremely powerful tools in the analysis of SLTH, and it's a great opportunity for the future works to leverage these results.

5.3.2 The Hopfield Network Version

Hopfield networks are models of associative memory, constructed by John J. Hopfield in early 1980s, for which he received Nobel Prize in Physics in 2024. Hopfield network is simply a fully connected glassy Ising model (also known as Curie–Weiss Models) with the Hamiltonian

$$\mathcal{H} = \sum_{i \neq j} J_{ij} s_i s_j,$$

with $s_i \in \{-1, 1\}$ and $J_{ij} = J_{ji}$. Note that since all the couplings are different (with possibly different signs), this system has very complex energy landscape with many degenerate ground states. The dynamics of this system at a low temperature brings the system into one of the nearest ground states, where it gets stuck for a long time. A Hopfield network is shown in Figure 5.5. Hopfield asked the following question, given a set of m configurations $\{\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(m)}\}$,

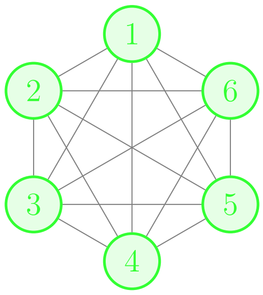

Figure 5.5: A Hopfield network

with $\boldsymbol{\xi}^{(k)} = \{\xi_i^{(k)}\}_{i=1}^{N}$, where N is the number of spins in the system, can we change J_{ij}'s such that the configurations $\{\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(m)}\}$ become the ground state of the system. He found

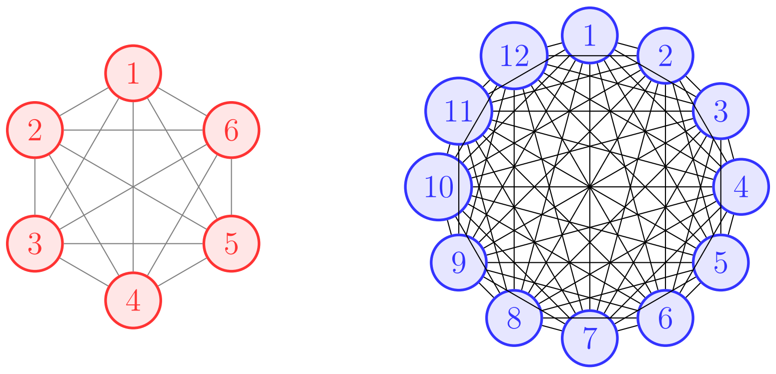

Figure 5.6: Pruning a Hopfield Network

the assignment of the J_{ij}'s which does exactly that:

$$J_{ij} = \sum_{k=1}^{m} \xi_i^{(k)} \xi_j^{(k)}.$$

Hence this is a model of associative memory- the system can remember the states $\boldsymbol{\xi}^k$'s, and recall them when starting from a nearby configuration. We now ask the following question related to Hopfield networks. Consider two Hopfield networks, one much larger than other, as shown in Figure 5.6. All the couplings J_{ij}'s of small network and J'_{ij}'s of the large network are randomly initialized. Say smaller one has N nodes, choose any N nodes of the bigger one. Can the couplings J'_{ij}'s of the large network be pruned (set to zero) such that the chosen N nodes in the bigger network has the same set of ground states as the smaller one? This is basically the Hopfield version of SLTH question. This question is inspired from the pruning process in biological neural networks. It has been observed that in biological brains, a lot of connections are made during the early developmental phase, and then the connections are pruned as the brain develops further. The question we ask is a Direct model of this process, and we leave it to future works to tackle this question.

Chapter 6

Conclusion

6.1 Discrete Random Subset Sum Problem

The theoretical analysis of Random Subset Sum problem (RSSP) is an important area in the field of Computer Science and High dimensional probability. The previous results by [Lue98] state that RSSP on the set $\{X_1, X_2, \ldots X_n\}$ where $X_i \sim U[-1, 1]$ can be solved with high probability if $n > C \log\left(\frac{1}{\epsilon}\right)$, within an error ϵ. This result played a major role in the previous analysis of Strong Lottery Ticket Hypothesis (SLTH) [Pen+20], [Bur22]. Leveraging the analysis by [BCP01], we prove results on RSSP in the discrete setting. Consider RSSP on the set $\{X_1, X_2, \ldots X_n\}$ where X_i's are sampled uniformly from $\{-1, \ldots, \delta \ldots, 1\}$ where $\delta << 1$. We show that with probability converging to 1 as $n \to \infty$, the RSSP can be solved exactly if $n > C \log\left(\frac{1}{\delta}\right)$. This result plays a major role in our analysis of SLTH in the discrete setting.

6.2 SLTH and Weight Quantization

Previous results on SLTH dealt with weights of the neural networks sampled from some continuous interval. However, weights in a computer are always represented with finite precision. To study the effects of quantization, we introduced a quantized formulation of the Strong Lottery Ticket Hypothesis (SLTH), addressing a more realistic setting where neural network weights have finite precision. By reducing the problem to the Number Partitioning Problem (NPP) and leveraging known phase transition results, we established tight upper bounds on the size and precision requirements for a randomly initialized network to represent a given target network exactly through pruning. Our analysis basically provides a relationship between precision of weights and required size of a large network that must hold in order for that network be prunable to a given target network. Notably, one of our constructions achieves exact size requirements (up to the first layer) without arbitrary undetermined constants, which is new. This also shows that the theory of Number Partitioning Problem provides powerful tools to study the theory of SLTH. The asymptotic matching of the forms of upper and lower bounds suggests the optimality of our approach. Our work highlights the fundamental relationship between network overparameterization, weight precision, and the existence of sparse, trainable subnetworks. Future work includes extending these results to more general architectures, relaxing precision assumptions, and analyzing approximate representations. Understanding the trade-offs between quantization, pruning, and expressivity remains a promising direction for both theory and practical network compression.

Appendix A

Saddle Point Approximation

In this chapter, we give a short overview of the Saddle Point Approximation. The Saddle Point Approximation is a powerful tool to get the asymptotic behavior of an Integral which are dominated by the maxima of the integrand, and is frequently used in Statistical Physics. Consider an integral of the form

$$I = \int_a^b dx\, f(x)\, e^{ng(x)}, \tag{A.1}$$

where f and g are real functions and $n > 0$ and g is bounded. The exponential function increases very rapidly, so for large n, the major contribution to the integral only comes from where $g(x)$ is maximum. Say we care about the integral in the $n \to \infty$ limit. The idea is to approximate the integral by the biggest peak of g. Let x_0 be the global maxima of g. Consider the change of variable

$$x = x_0 + \frac{y}{\sqrt{n}}.$$

If g is analytic, then $ng(x)$ it can be expanded around x_0 as

$$ng(x) = ng(x_0) + \frac{1}{2}y^2 g''(x_0) + \frac{y^3 g'''(x_0)}{6\sqrt{n}} + \mathcal{O}(y^4).$$

Note that $g'(x_0) = 0$ as x_0 is a maxima. This is a good approximation of g around its maxima, and for regions far from maxima does not matter as the contribution to the integral is negligible. Hence we have

$$e^{ng(x)} = e^{ng(x_0)} e^{\frac{1}{2}y^2 g''(x_0)} \left(1 + \frac{y^3 g'''(x_0)}{6\sqrt{n}} + \mathcal{O}(y^4)\right). \tag{A.2}$$

We also expand f as

$$f(x) = f(x_0)\left(1 + \frac{yf'(x_0)}{f(x_0)\sqrt{n}} + \frac{y^2 f''(x_0)}{f(x_0)n} + \mathcal{O}(y^3)\right). \tag{A.3}$$

Substituting Equations A.2 and A.3 into Equation A.1 we get

$$I(n) = \frac{f(x_0)e^{ng(x_0)}}{\sqrt{n}} \int_{y_1}^{y_2} dy\, e^{y^2 g''(x_0)/2} \cdot \left(1 + \sum_{k=1}^{\infty} \frac{P_k(y)}{\sqrt{n^k}}\right),$$

where $P_k(y)$ are polynomials in y. It can be shown that $P_k(y)$ are odd polynomials if k is odd and $P_k(y)$ are even polynomials if k is even. Since the integral is completely by it's maxima, we can extend the domain to $(-\infty, \infty)$. Hence we get

$$\begin{aligned} I(n) &\approx \frac{f(x_0)e^{ng(x_0)}}{\sqrt{n}} \int_{-\infty}^{\infty} dy\, e^{y^2 g''(x_0)/2} \cdot \left(1 + \sum_{k=1}^{\infty} \frac{P_k(y)}{\sqrt{n^k}}\right) \\ &= \frac{f(x_0)e^{ng(x_0)}}{\sqrt{n}} \sqrt{\frac{2\pi}{-ng''(x_0)}} \left(1 + \sum_{k=1}^{\infty} \frac{C_{2k}}{n^k}\right), \end{aligned}$$

Where C_{2k}'s are some coefficients. Hence one can write

$$I(n) = f(x_0)e^{ng(x_0)} \sqrt{\frac{2\pi}{-ng''(x_0)}} \left(1 + \mathcal{O}\left(\frac{1}{n}\right)\right).$$

Appendix B

Inequalities

B.1 Markov's Inequality

Theorem 9. *For a non-negative, integer-valued random variable X we have*

$$\mathbb{P}(X > 0) \leq \mathbb{E}[X]. \tag{B.1}$$

B.2 Cauchy-Schwartz inequality

Theorem 10. *If $X > 0$ is a random variable with finite variance, then*

$$\mathbb{P}(X > 0) \geq \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X^2]}. \tag{B.2}$$

B.3 Hoeffding's inequality

Theorem 11. *Let $X_1, X_2, \dots, X_n$ be independent random variables such that $a_i \leq X_i \leq b_i$ almost surely. Consider the sum of these random variables,*

$$S_n = X_1 + X_2 + \cdots + X_n.$$

Then Hoeffding's theorem states that, for all $t > 0$,

$$\mathbb{P}(S_n - \mathbb{E}(s_n) \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{B.3}$$

$$\mathbb{P}(|S_n - \mathbb{E}(s_n)| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

B.4 Union Bound

Theorem 12. *For any events $A_1, A_2, \ldots, A_n$ we have*

$$\mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right) \geq \max\left(0,\ \sum_{i=1}^{n}\mathbb{P}(A_i) - (n-1)\right). \tag{B.4}$$

Appendix C

Rejection Sampling

In this chapter, we describe the Rejection sampling algorithm. It is a simple yet powerful Monte Carlo technique to draw independent samples from a target probability density $f(x)$, using samples from an easier "proposal" density $g(x)$. It is extremely useful in cases, where the known distribution is not normalized. It proceeds in two stages—proposing a candidate, then accepting or rejecting it—so that the retained points are distributed according to f. We require a constant $M \geq 1$ such that

$$f(x) \leq M\, g(x)$$

for all x. Equivalently,

$$M = \sup_x \frac{f(x)}{g(x)}.$$

First we discuss the algorithm, and then we go into the mathematical details to understand how the algorithm works. The algorithm to generate N samples is:

1. Propose $x^* \sim g(x)$.
2. Draw $u \sim \text{Uniform}(0, 1)$, independently.
3. Accept x^* if
   $$u \leq \frac{f(x^*)}{M\, g(x^*)},$$
   otherwise reject and return to step 1.

To see why the accepted samples follow f, consider the joint density of (X^*, U):

$$p(x, u) = g(x) \times 1, \quad 0 \leq u \leq 1.$$

The marginal density of accepted x is

$$p_{\text{accept}}(x) = \int_0^{f(x)/(Mg(x))} g(x)\, du = g(x)\, \frac{f(x)}{M\, g(x)} = \frac{f(x)}{M}.$$

The total acceptance probability is

$$P(\text{accept}) = \int p_{\text{accept}}(x)\, dx = \int \frac{f(x)}{M}\, dx = \frac{1}{M}.$$

Conditioning on acceptance gives

$$p(x \mid \text{accepted}) = \frac{p_{\text{accept}}(x)}{P(\text{accept})} = f(x).$$

Thus each accepted sample is exactly from the target density f. The expected acceptance rate is

$$P(\text{accept}) = \frac{1}{M},$$

so efficiency improves as M approaches 1.

A discrete analogue for a target mass function $f(i)$ on $i \in \{1, \ldots, K\}$ uses a proposal $g(i)$ and a constant $M \geq \max_i f(i)/g(i)$. The steps are:

1. Draw $i^* \sim g$.
2. Draw $u \sim \text{Uniform}(0, 1)$.
3. Accept if
$$u \leq \frac{f(i^*)}{M\, g(i^*)}.$$

The same argument shows the accepted i^* follow the distribution f. Rejection sampling is conceptually simple and provides exact (unbiased) samples from f, but it can be inefficient if M is large or difficult to bound, especially in high dimensions.

Bibliography

[BCP01] Christian Borgs, Jennifer Chayes, and Boris Pittel. "Phase transition and finite-size scaling for the integer partitioning problem". In: *Random Structures amp; Algorithms* 19.3–4 (Oct. 2001), pp. 247–288. URL: http://dx.doi.org/10.1002/rsa.10004.

[Bur22] Rebekka Burkholz. *Most Activation Functions Can Win the Lottery Without Excessive Depth*. 2022. URL: https://arxiv.org/abs/2205.02321.

[DK21] James Diffenderfer and Bhavya Kailkhura. *Multi-Prize Lottery Ticket Hypothesis: Finding Accurate Binary Neural Networks by Pruning A Randomly Weighted Network*. 2021. URL: https://arxiv.org/abs/2103.09377.

[FC18] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: (2018). URL: https://arxiv.org/abs/1803.03635.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*. New York, NY: W.H. Freeman, Apr. 1979.

[Lue98] George S. Lueker. "Exponentially small bounds on the expected optimum of the partition and subset sum problems". In: *Random Structures and Algorithms* 12.1 (Jan. 1998), pp. 51–62. URL: http://dx.doi.org/10.1002/(SICI)1098-2418(199801)12:1%3C51::AID-RSA3%3E3.0.CO;2-S.

[Mal+20] Eran Malach et al. *Proving the Lottery Ticket Hypothesis: Pruning is All You Need*. 2020. URL: https://arxiv.org/abs/2002.00585.

[Mer98] Stephan Mertens. "Phase Transition in the Number Partitioning Problem". In: *Physical Review Letters* 81.20 (Nov. 1998), pp. 4281–4284. URL: http://dx.doi.org/10.1103/PhysRevLett.81.4281.

[Pen+20] Ankit Pensia et al. *Optimal Lottery Tickets via SubsetSum: Logarithmic Over-Parameterization is Sufficient*. 2020. URL: https://arxiv.org/abs/2006.07990.

[Ram+19] Vivek Ramanujan et al. *What's Hidden in a Randomly Weighted Neural Network?* 2019. URL: https://arxiv.org/abs/1911.13299.

[Ste89] Daniel L Stein. *Lectures in the sciences of complexity*. Philadelphia, PA: Westview Press, Sept. 1989.

[Wan+19] Yulong Wang et al. *Pruning from Scratch*. 2019. URL: https://arxiv.org/abs/1909.12579.

[Zho+19] Hattie Zhou et al. *Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask*. 2019. URL: https://arxiv.org/abs/1905.01067.