A Lower Bound for the Number of Linear Regions of Ternary ReLU Regression Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

With the advancement of deep learning, reducing computational complexity and memory consumption has become a critical challenge, and ternary neural networks (NNs) that restrict parameters to $\{-1, 0, +1\}$ have attracted attention as a promising approach. While ternary NNs demonstrate excellent performance in practical applications such as image recognition and natural language processing, their theoretical understanding remains insufficient. In this paper, we theoretically analyze the expressivity of ternary NNs from the perspective of the number of linear regions. Specifically, we evaluate the number of linear regions of ternary regression NNs with Rectified Linear Unit (ReLU) for activation functions and prove that the number of linear regions increases polynomially with respect to network width and exponentially with respect to depth, similar to standard NNs. Moreover, we show that it suffices to first double the width, then either square the width or double the depth of ternary NNs to achieve a lower bound on the maximum number of linear regions comparable to that of general ReLU regression NNs. This provides a theoretical explanation, in some sense, for the practical success of ternary NNs.

1 Introduction

In recent years, with the rapid development of deep learning, neural networks (NNs) have achieved remarkable results in various fields. However, their large computational and memory consumption has become a serious barrier to applications in mobile devices and edge computing. Particularly, considering implementation in embedded systems that require real-time processing or in environments with limited computational resources, memory and computation reduction of NNs is an urgent issue.

As a promising approach to this problem, methods for discretizing NN parameters have been proposed. Specifically, methods that restrict network weights to binary $\{-1, +1\}$ or ternary $\{-1, 0, +1\}$ values, or quantize the output values of activation functions have been developed (Courbariaux et al., 2016; Li et al., 2022). These methods have achieved performance comparable to conventional continuous-valued NNs in a wide range of tasks including image recognition (Rastegari et al., 2016; Liu et al., 2020), natural language processing (Bai et al., 2021; Wang et al., 2023; Ma et al., 2024), and speech recognition (Xiang et al., 2017), while successfully achieving significant reductions in computational complexity and memory usage. Particularly noteworthy is the surprising fact that these discretized NNs can maintain high performance in practical tasks despite extremely restricting their parameters.

However, theoretical understanding of the fundamental question of why these discretization methods work effectively remains insufficient. The motivation of this research is to theoretically explain the success of ternary NNs in particular. Research providing such theoretical explanations has barely been conducted. Especially, evaluation from the perspective of NN expressivity, particularly evaluation by the number of linear regions, has not yet been performed to the best of our knowledge.

Regarding the number of linear regions of general NNs, several important results have already been reported. In particular, for the expressivity of regression NNs using Rectified Linear Unit (ReLU) for activation functions, it has been shown that the maximum number of linear regions increases polynomially with respect to network width and exponentially with respect to depth (Montúfar et al., 2014). Following this research, various studies have evaluated different quantities related to linear regions (Pascanu et al., 2014; Serra et al., 2018; Hanin & Rolnick, 2019; Esaki et al., 2020). These results provide important theoretical grounds suggesting that deep NNs are inherently superior to shallow NNs in terms of expressivity. Therefore, this paper aims to theoretically characterize the expressivity of ternary NNs by evaluating their number of linear regions. Note that we focus on regression problems using ReLU as the activation function and does not address quantization of activation functions in this paper.

The main contribution of this paper is to show that the maximum number of linear regions of ternary NNs also increases polynomially with respect to width and exponentially with respect to depth, similar to conventional NNs. More specifically, we prove that it suffices to first double the width, then either square the width or double the depth of ternary ReLU regression NNs to obtain a lower bound on the maximum number of linear regions comparable to that for general ReLU regression NNs. Although from the limited perspective of the number of linear regions of piecewise linear functions represented by ReLU NNs, and from the comparison between lower bounds on the maximum number of linear regions of conventional NNs and ternary NNs, these results provide one theoretical explanation for the practical success of ternary NNs.

The rest of this paper is structured as follows. Section 2 introduces notation for explaining this research and previous studies. Section 3 reviews existing research on the number of linear regions of general NNs. Section 4 states the main theorem regarding the number of linear regions of ternary NNs and provides its proof. Section 5 discusses the significance and limitations of the obtained results and concludes this research.

2 Preliminaries

Definition 1 (Neural networks)**.** *Let an arbitrary natural number $L \in \mathbb{N}$ and natural numbers $n_l \in \mathbb{N}$ be given for integers $l = 0, 1, 2, \dots, L$. Then, we call the following function $\boldsymbol{F_\theta} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$, which is expressed as a composition of linear functions*[1] $\boldsymbol{f}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}; \boldsymbol{x} \mapsto \boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}$ *and nonlinear functions* $\boldsymbol{g}^{(l)} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$*, a NN of depth L with width n_l at the l-th layer:*

$$\boldsymbol{F_\theta}(\boldsymbol{x}) = \boldsymbol{f}^{(L)} \circ \boldsymbol{g}^{(L-1)} \circ \boldsymbol{f}^{(L-1)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{x}). \tag{1}$$

Here, n_0 and n_L represent the dimensions of input and output, respectively, and $\boldsymbol{\theta}$ represents all parameters $\{(\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)})\}_{l=1}^{L}$ of the NN. In this paper, we only deal with regression NNs where the final layer is a linear function. The nonlinear function $\boldsymbol{g}^{(l)}$ is called an activation function. In particular, we call regression NNs that use the following ReLU or identity function as the activation function $\boldsymbol{g}^{(l)}$ ReLU Regression NNs in this paper.

$$g_j^{(l)}(\boldsymbol{x}) = \max\{0, x_j\}, \tag{2}$$

where $g_j^{(l)}(\boldsymbol{x})$ and x_j represent the j-th components of $\boldsymbol{g}^{(l)}(\boldsymbol{x})$ and $\boldsymbol{x}$, respectively. Also, we call NNs where the coefficient parameters of the linear function $\boldsymbol{f}^{(l)}$ take only values from $\{1, 0, -1\}$ ternary NNs in this paper.

Limiting NN coefficients to ternary values is also performed in BitNet b1.58 (Ma et al., 2024). While BitNet b1.58 additionally quantizes the output of activation functions, as mentioned in the introduction, this paper does not deal with quantization of activation functions. Evaluating the effect of activation function quantization on NN expressivity is a future research topic.

NNs can be represented as graphs as shown in Fig. 1. In particular, in this paper, we represent that the activation function is ReLU by drawing the bent line inside the node.

Definition 2 (Linear regions (Montúfar et al., 2014))**.** *For a function $\boldsymbol{f} : D \to \mathbb{R}^n$, we say that $U \subset D$ is a linear region of $\boldsymbol{f}$ if the following holds:*

- *The restriction*[2] *of $\boldsymbol{f}$ to U, $\boldsymbol{f}|_U : U \to \mathbb{R}^n$, is a linear function.*

[1]In this paper, linear functions refer to functions that form some hyperplane and do not necessarily pass through the origin. More precisely, such functions are called affine functions, but following the notation of previous research (Montúfar et al., 2014), we call them linear functions in this paper.

[2]A function that has the same input-output mapping as $\boldsymbol{f}$ but is defined only on U.

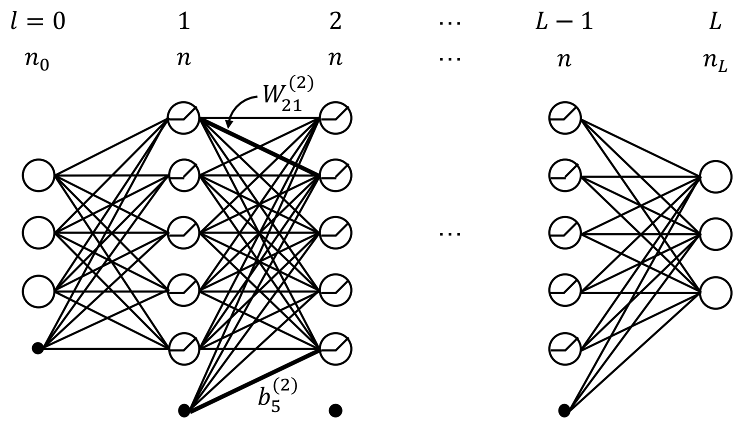

Figure 1: Illustration of NN. The weight of the edge extending from the j-th node of the $(l-1)$-th layer to the i-th node of the l-th layer corresponds to the (i, j) component $W_{ij}^{(l)}$ of the coefficient matrix $\boldsymbol{W}^{(l)}$ of the linear function $\boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}$. The bent line inside the node represents that the activation function is ReLU. Small black dots represent constant terms.

- *For any $V \subset D$ satisfying $V \supsetneq U$, $\boldsymbol{f}|_V$ is not a linear function.*

That is, a locally maximal subset U of D where $\boldsymbol{f}$ becomes a linear function within that region is called a linear region of $\boldsymbol{f}$. Also, we denote the set of all linear regions of $\boldsymbol{f}$ as $\mathcal{L}(\boldsymbol{f})$ in this paper.

Example 1 (Linear regions of absolute value function)**.** *The linear regions of the absolute value function $f(x) = |x|$ are $(-\infty, 0]$ and $[0, \infty)$, so $\mathcal{L}(f) = \{(-\infty, 0], [0, \infty)\}$. Note that linear regions are closed sets, by its definition.*

Here, note that when $\boldsymbol{f} : D \to \mathbb{R}^n$ is a piecewise linear function, the following holds:

$$\bigcup_{U \in \mathcal{L}(\boldsymbol{f})} U = D. \tag{3}$$

3 Previous Studies

Conventionally, it is known that the following holds for the number of linear regions $|\mathcal{L}(\boldsymbol{F})|$ of a ReLU Regression NN $\boldsymbol{F} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with depth L and width n at each layer.

Proposition 1 (A lower bound for the number of linear regions of ReLU Regression NNs (Montúfar et al., 2014))**.** *For a ReLU Regression NN $\boldsymbol{F_\theta} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with depth L and width n at each layer, let $p = \lfloor \frac{n}{n_0} \rfloor$. Then the following holds:*

$$\max_{\boldsymbol{\theta}} |\mathcal{L}(\boldsymbol{F_\theta})| \geq p^{n_0(L-1)}. \tag{4}$$

This suggests that the number of linear regions of NNs increases in polynomial order with respect to network width and in exponential order with respect to depth, and is considered one piece of evidence showing that deepening the depth is more effective than widening the width for enhancing the expressivity of NNs.

The proof of Proposition 1 in (Montúfar et al., 2014) is given by specifically constructing a ReLU Regression NN such that the number of linear regions becomes $p^{n_0(L-1)}$. However, while Montúfar et al. (2014) shows the construction procedure of the ReLU Regression NN, it does not concisely state the mathematical formulas of the linear functions of each layer constructed by this procedure. To make it easier to use in the proof of our theorem described later, we express this using mathematical formulas here. In the ReLU Regression NN constructed by the method of (Montúfar et al., 2014), the n nodes of each intermediate layer are divided into

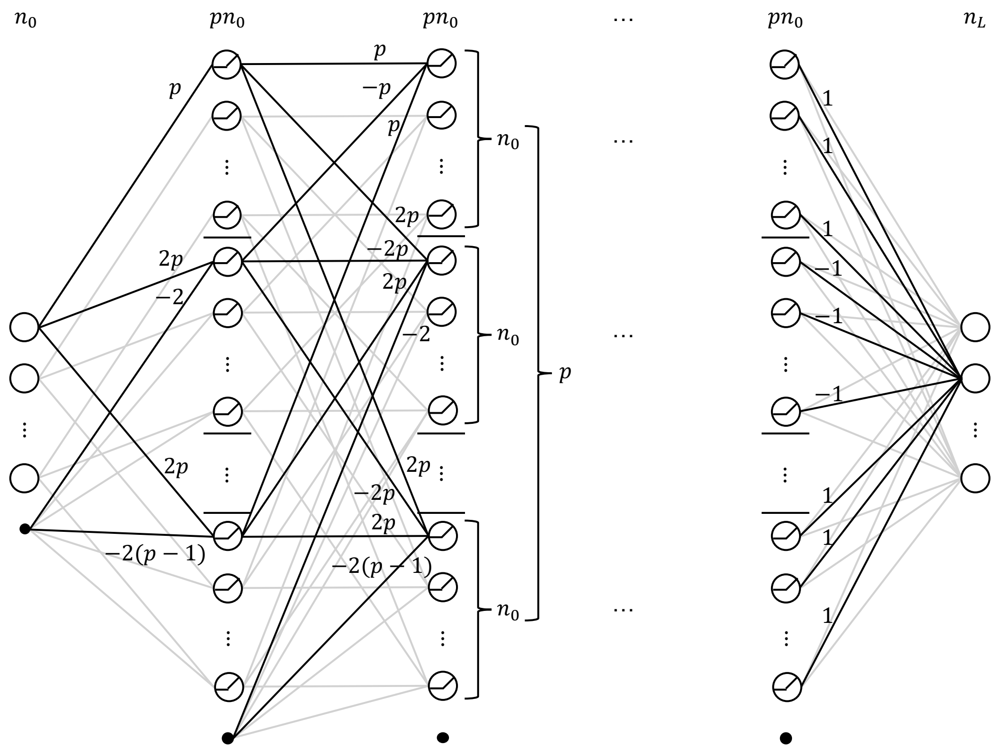

Figure 2: ReLU Regression NN with number of linear regions equal to $p^{n_0(L-1)}$.

p groups of n_0 nodes each, and the linear function $f^{(l)}_{(i-1)n_0+j}(\boldsymbol{x})$ corresponding to the j-th node of the i-th group in the l-th layer is expressed by the following formula (for the remaining $n-pn_0$ nodes, all coefficients are set to 0, so that 0 is identically output).

- **The first layer:** for $l=1$, $i=1,2,\dots,p$ and $j=1,2,\dots,n_0$, the $((i-1)n_0+j)$-th component $f^{(1)}_{(i-1)n_0+j}(\boldsymbol{x})$ of $\boldsymbol{f}^{(1)}(\boldsymbol{x})$ is expressed as follows:

$$f^{(1)}_{(i-1)n_0+j}(\boldsymbol{x}) = \begin{cases} px_j, & i=1, \\ 2px_j - 2(i-1), & 2\le i\le p. \end{cases} \tag{5}$$

- **Intermediate layers:** for $l=2,3,\dots,L-1$, $i=1,2,\dots,p$ and $j=1,2,\dots,n_0$, the $((i-1)n_0+j)$-th component $f^{(l)}_{(i-1)n_0+j}(\boldsymbol{x})$ of $\boldsymbol{f}^{(l)}(\boldsymbol{x})$ is expressed as follows:

$$f^{(l)}_{(i-1)n_0+j}(\boldsymbol{x}) = \begin{cases} p\sum_{k=1}^{p}(-1)^{k-1}x_{(k-1)n_0+j}, & i=1, \\ 2p\sum_{k=1}^{p}(-1)^{k-1}x_{(k-1)n_0+j} - 2(i-1), & 2\le i\le p. \end{cases} \tag{6}$$

- **The last layer:** for $l=L$, $m=1,2,\dots,n_L$, the m-th component $f^{(L)}_m(\boldsymbol{x})$ of $\boldsymbol{f}^{(L)}(\boldsymbol{x})$ is expressed as follows:

$$f^{(L)}_m(\boldsymbol{x}) = \sum_{j=1}^{n_0}\sum_{k=1}^{p}(-1)^{k-1}x_{(k-1)n_0+j}. \tag{7}$$

We can represent these as a graph shown in Fig. 2.

Although it is merely a rewrite of the proof in (Montúfar et al., 2014) using equations equation 5, equation 6, and equation 7, the proof that the number of linear regions of this ReLU Regression NN is indeed $p^{n_0(L-1)}$ is summarized in Appendix.

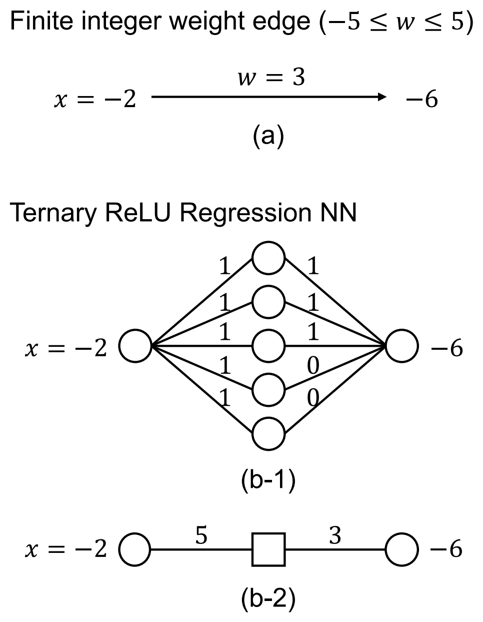
Finite integer weight edge ($-5 \le w \le 5$)
$w = 3$
$x = -2$
-6
(a)
Ternary ReLU Regression NN
1
1
1
1
1
1
1
1
0
0
$x = -2$
-6
(b-1)
$x = -2$
5
3
-6
(b-2)

Figure 3: (a): A certain edge of a bounded integer coefficient NN. Here, the maximum weight is $M = 5$ and the weight in this example is $w = 3$. (b-1): Representation of (a) by a ternary ReLU Regression NN. Here, the activation function of the intermediate layer is the identity function. (b-2): Abbreviated notation of (b-1).

4 Main Results

Before explaining the main results, we show that bounded integer coefficient regression NNs can be represented by ternary ReLU Regression NNs. First, any edge of a bounded integer coefficient NN can be represented by a ternary ReLU Regression NN. Consider a NN where there exists some natural number M such that the coefficients of linear functions are represented by integers in the range from $-M$ to M. Suppose one edge has integer weight w as shown in Fig. 3 (a). That is, suppose this NN partially includes the process of multiplying the input value by w. This process of multiplying the input value by w can be represented in a two-layer ternary ReLU Regression NN with the number of nodes in each layer being $n_0 = 1$, $n_1 = M$, $n_2 = 1$, by setting

$$f_j^{(1)}(x) = x, \quad (j = 1, 2, \ldots, M) \tag{8}$$

$$\boldsymbol{g}^{(1)}(\boldsymbol{y}) = \boldsymbol{y}, \tag{9}$$

$$f^{(2)}(\boldsymbol{z}) = \sum_{j=1}^{|w|} \operatorname{sign}(w) z_j \tag{10}$$

where sign(w) represents the sign of w. This function is also represented as a graph shown in Fig. 3 (b-1). Further, we represent the M nodes in the intermediate layer in this graph by one square node, and write the sum of the weights of edges entering and exiting that node next to the edge, as shown in Fig. 3 (b-2).

Furthermore, when representing not an edge but a bounded integer coefficient NN with a ternary ReLU Regression NN, it is sometimes possible to represent it with fewer nodes than replacing each edge with the aforementioned transformation by combining the nodes of the ternary ReLU Regression NN. For example, Fig. 4 shows ternary ReLU Regression NNs equivalent to a bounded integer coefficient NN (a). In Fig. 4, (b) shows the ternary ReLU Regression NN by trivial transformation, (c-1) shows the ternary ReLU Regression NN by transformation with fewer nodes, and (c-2) shows its abbreviated notation. In Fig. 4 (c-2), edges extend from a square node to multiple nodes, with some numbers attached. Each of these edges represents

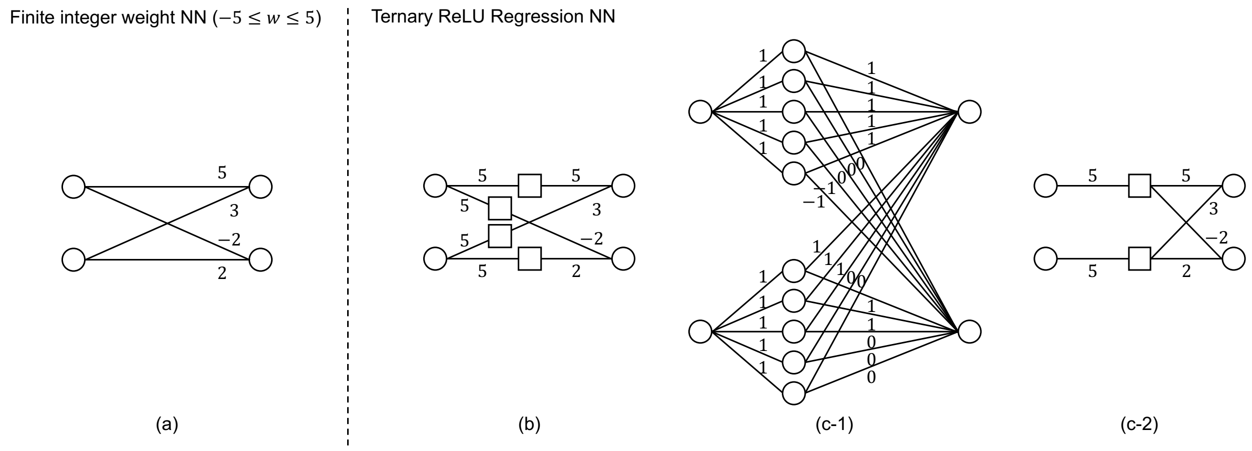
Finite integer weight NN ($-5 \le w \le 5$)
Ternary ReLU Regression NN
(a)
(b)
(c-1)
(c-2)

Figure 4: (a): A bounded integer coefficient NN where weight w satisfies $-5 \le w \le 5$. (b): Representation of (a) by a ternary ReLU Regression NN through trivial transformation. (c-1): Representation of (a) by a ternary ReLU Regression NN through transformation with fewer nodes. (c-2): Abbreviated notation of (c-1).

the M edges extending from the M nodes in (c-1), which correspond to the square node in (c-2), and the number represents the sum of the coefficients of those M edges.

Next, we state the main result of this research and its proof. In the following, for a ReLU Regression NN with depth L and width n, we consider a ternary ReLU Regression NN with the same depth L and width n, and evaluate the lower bound of the maximum number of linear regions that can be represented by adjusting the edge weights. The proof strategy is as follows. We utilize the fact that the coefficients of the linear functions equation 5, equation 6, equation 7 used in the proof of Proposition 1 are bounded integers. We also leverage the relationship between bounded integer coefficient NNs and ternary ReLU Regression NNs mentioned above. Based on these observations, we construct a ternary ReLU Regression NN that represents functions of the same form as equation 5, equation 6, equation 7. The following is the main result of this research.

Theorem 1 (A lower bound of the number of linear regions of ternary NN with ReLU after even layer)**.** *For a ternary NN $\boldsymbol{F_\theta} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ with depth L and width n at each layer, where the activation function of odd-numbered layers is the identity function and the activation function of even-numbered layers is ReLU, let L' be the maximum odd number less than or equal to L and $q = \lfloor \frac{n}{2(n_0+1)} \rfloor$. Then the following holds:*

$$\max_{\boldsymbol{\theta}} |\mathcal{L}(\boldsymbol{F_\theta})| \ge q^{\frac{1}{2} n_0 (L'-1)}. \tag{11}$$

Proof. By constructing the ternary ReLU Regression NN shown in Fig. 5, the function represented by this NN becomes the function obtained by replacing p with $q = \lfloor \frac{n}{2(n_0+1)} \rfloor$ in equation 5, equation 6, equation 7. In the following, we describe the specific construction method of this NN and the mathematical expression of the function represented by it.

Let L' be the maximum odd number less than or equal to L. If L is even, we set the final layer to be an identity transformation, and hereafter we construct a ternary ReLU Regression NN with depth L' and width n at each layer. Setting $q = \lfloor \frac{n}{2(n_0+1)} \rfloor$, we divide the n nodes of the odd layers of this NN into $n_0 + 1$ groups of $2q$ nodes each. In Fig. 5, these $2q$ nodes are represented by one square node. For the remaining $n - 2q(n_0 + 1)$ nodes, all coefficients are set to 0, making them functions that identically output 0. For the nodes of even layers, we create q groups of n_0 nodes each, and for the remaining $n - qn_0$ nodes, all coefficients are set to 0, so that they identically output 0. First, since equation equation 7 already takes only values from $\{1, 0, -1\}$ for weight coefficients, we can use the linear function obtained by replacing p with q in equation equation 7 for the L'-th layer. Next, for $l = 1, 2, \dots, \frac{1}{2}(L' - 1)$, we define the $(2l - 1)$-th layer and the $2l$-th layer as follows:

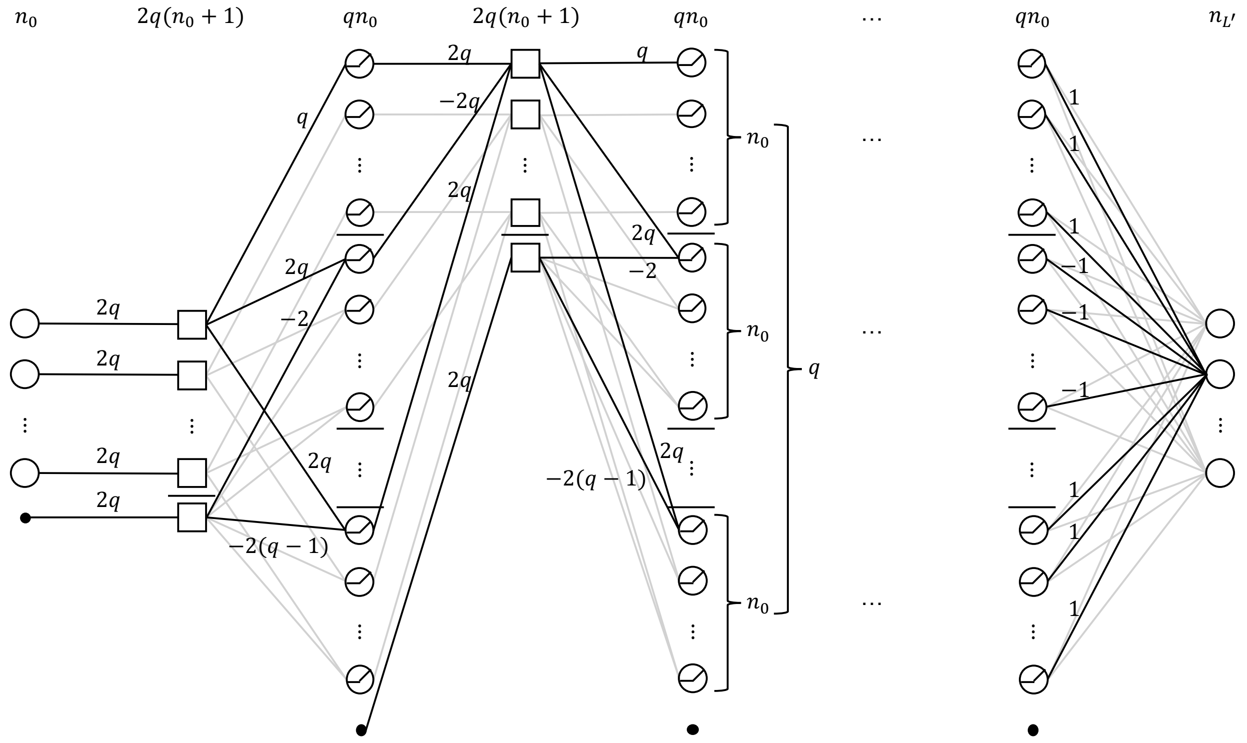

Figure 5: A ternary ReLU Regression NN with number of linear regions equal to $q^{\frac{1}{2}n_0(L'-1)}$.

- For $l = 1$:
  - Definition of $\boldsymbol{f}^{(1)}$: For $j = 1, 2, \ldots, n_0 + 1$ and $k = 1, 2, \ldots, 2q$, define the $(2q(j-1)+k)$-th component $f^{(1)}_{2q(j-1)+k}(\boldsymbol{x})$ of $\boldsymbol{f}^{(1)}(\boldsymbol{x})$ as follows:

$$f^{(1)}_{2q(j-1)+k}(\boldsymbol{x}) = \begin{cases} x_j, & j = 1, 2, \ldots, n_0 \\ 1, & j = n_0 + 1. \end{cases} \tag{12}$$

  - Definition of $\boldsymbol{f}^{(2)}$: For $i = 1, 2, \ldots, q$ and $j = 1, 2, \ldots, n_0$, define the $((i-1)n_0+j)$-th component $f^{(2)}_{(i-1)n_0+j}(\boldsymbol{x})$ of $\boldsymbol{f}^{(2)}(\boldsymbol{x})$ as follows:

$$f^{(2)}_{(i-1)n_0+j}(\boldsymbol{x}) = \begin{cases} \sum_{k=1}^{q} x_{2q(j-1)+k}, & i = 1 \\ \sum_{k=1}^{2q} x_{2q(j-1)+k} + \sum_{k=1}^{2(i-1)}(-1), & i = 2, 3, \ldots, q. \end{cases} \tag{13}$$

- For $l = 2, 3, \ldots, \frac{1}{2}(L'-1)$:
  - Definition of $\boldsymbol{f}^{(2l-1)}$: For $j = 1, 2, \ldots, n_0 + 1$ and $k = 1, 2, \ldots, 2q$, define the $(2q(j-1)+k)$-th component $f^{(2l-1)}_{2q(j-1)+k}(\boldsymbol{x})$ of $\boldsymbol{f}^{(2l-1)}(\boldsymbol{x})$ as follows:

$$f^{(2l-1)}_{2q(j-1)+k}(\boldsymbol{x}) = \begin{cases} \sum_{m=1}^{q}(-1)^{m-1} x_{(m-1)n_0+j}, & j = 1, 2, \ldots, n_0 \\ 1, & j = n_0 + 1. \end{cases} \tag{14}$$

  - Definition of $\boldsymbol{f}^{(2l)}$: For $i = 1, 2, \ldots, q$ and $j = 1, 2, \ldots, n_0$, define the $((i-1)n_0+j)$-th component $f^{(2l)}_{(i-1)n_0+j}(\boldsymbol{x})$ of $\boldsymbol{f}^{(2l)}(\boldsymbol{x})$ as follows:

$$f^{(2l)}_{(i-1)n_0+j}(\boldsymbol{x}) = \begin{cases} \sum_{k=1}^{q} x_{2q(j-1)+k}, & i = 1 \\ \sum_{k=1}^{2q} x_{2q(j-1)+k} + \sum_{k=1}^{2(i-1)}(-1), & i = 2, 3, \ldots, q. \end{cases} \tag{15}$$

For activation functions, we use the identity function for odd layers and ReLU for even layers.

Since the activation function of odd layers is the identity function, when we substitute $\boldsymbol{f}^{(1)}$ into $\boldsymbol{f}^{(2)}$, we can see that $\boldsymbol{f}^{(2)} \circ \boldsymbol{f}^{(1)}$ of the ternary ReLU Regression NN equals the function obtained by replacing p with q in the linear function equation 5 of the first layer of a regular ReLU Regression NN. Also, for $l = 2, 3, \ldots, \frac{1}{2}(L' - 1)$, when we substitute $\boldsymbol{f}^{(2l-1)}$ into $\boldsymbol{f}^{(2l)}$, we can see that $\boldsymbol{f}^{(2l)} \circ \boldsymbol{f}^{(2l-1)}$ of the ternary ReLU Regression NN equals the function obtained by replacing p with q in the linear function equation 6 of the l-th layer of a regular NN. Therefore, the ternary ReLU Regression NN shown here achieves the lower bound of the number of linear regions obtained by replacing p with q and $L-1$ with $\frac{1}{2}(L'-1)$ in the right-hand side of equation equation 4 for a regular ReLU Regression NN. That is, the number of linear regions of this ternary ReLU Regression NN becomes $q^{\frac{1}{2}n_0(L'-1)}$. □

Remark 1. *Comparing equation equation 4 and equation equation 11, roughly speaking, to obtain a lower bound of the maximum number of linear regions comparable to that for general ReLU Regression NNs, it suffices to first double the width, then square the width or double the depth of ternary NNs where the activation function of odd-numbered layers is the identity function and the activation function of even-numbered layers is ReLU.*

5 Conclusion

We theoretically evaluated the expressivity of ternary NNs, which have achieved great practical success in memory and computation reduction of NNs, from the perspective of the number of linear regions. As a result, it was shown that the expressivity of ternary ReLU Regression NNs increases polynomially with respect to network width and exponentially with respect to depth, similar to general ReLU Regression NNs. Furthermore, it was found that it suffices to first double the width, then square the width or double the depth of ternary ReLU Regression NNs to obtain a lower bound on the maximum number of linear regions comparable to that for general ReLU Regression NNs. We believe this provides an explanation of a part of the reason for the practical success of ternary NNs, albeit from the limited perspective of the number of linear regions of piecewise linear functions represented by ReLU NNs. However, in actual applications such as BitNet b1.58, quantization of activation functions is performed for further memory and computation reduction, and our method cannot be directly applied. Theoretical evaluation of the expressivity of such NNs is a future research topic.

References

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT: Pushing the limit of BERT quantization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4334–4348, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.334. URL `https://aclanthology.org/2021.acl-long.334/`.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. URL `https://arxiv.org/abs/1602.02830`.

Yasushi Esaki, Yuta Nakahara, and Toshiyasu Matsushima. Theoretical analysis of the advantage of deepening neural networks. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 479–484, 2020. doi: 10.1109/ICMLA51294.2020.00081.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2596–2604. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/hanin19a.html`.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks, 2022. URL `https://arxiv.org/abs/1605.04711`.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 143–159, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58568-6.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024. URL `https://arxiv.org/abs/2402.17764`.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/fa6f2a469cc4d61a92d96e74617c3d2a-Paper.pdf`.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations, 2014. URL `https://arxiv.org/abs/1312.6098`.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 525–542, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4558–4566. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/serra18b.html`.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models, 2023. URL `https://arxiv.org/abs/2310.11453`.

Xu Xiang, Yanmin Qian, and Kai Yu. Binary deep neural networks for speech recognition. In *Interspeech 2017*, pp. 533–537, 2017. doi: 10.21437/Interspeech.2017-1343.

A Proof of Proposition 1

Before proving Proposition 1, we show the following Lemma.

Lemma 1 (The number of linear regions of a composite function of piecewise linear functions)**.** *Let $\boldsymbol{g} : D_1 \to D_2$ be a piecewise linear function, and suppose that for any linear region $U \in \mathcal{L}(\boldsymbol{g})$ of $\boldsymbol{g}$, $\boldsymbol{g}|_U$ is a bijection from U to D_2. Let $\boldsymbol{f} : D_2 \to \mathbb{R}^n$ be a piecewise linear function. Note that the domain of $\boldsymbol{f}$ coincides with the range of $\boldsymbol{g}$. Then, the following holds for the number of linear regions of the composite function $\boldsymbol{f} \circ \boldsymbol{g}$:*

$$|\mathcal{L}(\boldsymbol{f} \circ \boldsymbol{g})| = |\mathcal{L}(\boldsymbol{f})||\mathcal{L}(\boldsymbol{g})|. \tag{16}$$

Proof of Lemma 1. For any $U \in \mathcal{L}(\boldsymbol{g})$ and any $V \in \mathcal{L}(\boldsymbol{f})$, consider the inverse image of V under $\boldsymbol{g}|_U$, denoted $\boldsymbol{g}|_U^{-1}(V)$. On $\boldsymbol{g}|_U^{-1}(V)$, $\boldsymbol{f} \circ \boldsymbol{g}|_U$ is clearly a linear function. Also, on the set $\boldsymbol{g}|_U^{-1}(V) \cup \{x\}$ obtained by adding any point $x \in U \setminus \boldsymbol{g}|_U^{-1}(V)$ to $\boldsymbol{g}|_U^{-1}(V)$, $\boldsymbol{f} \circ \boldsymbol{g}|_U$ does not become a linear function. This is because, due to the bijectivity of $\boldsymbol{g}|_U$, we have $\boldsymbol{g}|_U(\boldsymbol{g}|_U^{-1}(V) \cup \{x\}) \supsetneq V$, and by the definition of linear region V, $\boldsymbol{f}|_{\boldsymbol{g}|_U(\boldsymbol{g}|_U^{-1}(V)\cup\{x\})}$ does not become a linear function. Therefore, $\boldsymbol{g}|_U^{-1}(V)$ is a linear region of $\boldsymbol{f} \circ \boldsymbol{g}|_U$.

Also, the following holds:

$$\begin{aligned} U &= \boldsymbol{g}|_U^{-1}(D_2) && \because \boldsymbol{g}|_U \text{ is a bijection from } U \text{ to } D_2 && (17)\\ &= \boldsymbol{g}|_U^{-1}\left(\textstyle\bigcup_{V\in\mathcal{L}(\boldsymbol{f})} V\right) && \because \boldsymbol{f} \text{ is a piecewise linear function} && (18)\\ &= \bigcup_{V\in\mathcal{L}(\boldsymbol{f})} \boldsymbol{g}|_U^{-1}(V). && \because \boldsymbol{g}|_U \text{ is a bijection from } U \text{ to } D_2 && (19)\end{aligned}$$

This indicates that U is divided into $|\mathcal{L}(\boldsymbol{f})|$ linear regions $\boldsymbol{g}|_U^{-1}(V)$ of $\boldsymbol{f}\circ\boldsymbol{g}|_U$. Since the same holds for any $U\in\mathcal{L}(\boldsymbol{g})$, we have $|\mathcal{L}(\boldsymbol{f}\circ\boldsymbol{g})| = |\mathcal{L}(\boldsymbol{f})||\mathcal{L}(\boldsymbol{g})|$. □

Next, we prove Proposition 1.

Proof of Proposition 1. First, we define the following three functions $\tilde{\boldsymbol{f}} : [0,1]^{n_0} \to \mathbb{R}^{pn_0}$, $\tilde{\tilde{\boldsymbol{f}}} : \mathbb{R}^{pn_0} \to [0,1]^{n_0}$, and $\tilde{\tilde{\tilde{\boldsymbol{f}}}} : [0,1]^{n_0} \to \mathbb{R}^{n_L}$. Note the domain and range of each function.

- Definition of $\tilde{\boldsymbol{f}} : [0,1]^{n_0} \to \mathbb{R}^{pn_0}$: For any $i = 1,2,\dots,p$ and $j = 1,2,\dots,n_0$, define the $((i-1)n_0+j)$-th component $\tilde{f}_{(i-1)n_0+j}(\boldsymbol{x})$ of $\tilde{\boldsymbol{f}}(\boldsymbol{x})$ as follows:

$$\tilde{f}_{(i-1)n_0+j}(\boldsymbol{x}) = \begin{cases} px_j, & i = 1\\ 2px_j - 2(i-1), & i = 2,3,\dots,p. \end{cases} \tag{20}$$

- Definition of $\tilde{\tilde{\boldsymbol{f}}} : \mathbb{R}^{pn_0} \to [0,1]^{n_0}$: For any $j = 1,2,\dots,n_0$, define the j-th component $\tilde{\tilde{f}}_j(\boldsymbol{x})$ of $\tilde{\tilde{\boldsymbol{f}}}(\boldsymbol{x})$ as follows:

$$\tilde{\tilde{f}}_j(\boldsymbol{x}) = \sum_{k=1}^{p}(-1)^{k-1}x_{(k-1)n_0+j}. \tag{21}$$

- Definition of $\tilde{\tilde{\tilde{\boldsymbol{f}}}} : [0,1]^{n_0} \to \mathbb{R}^{n_L}$: For any $m = 1,2,\dots,n_L$, define the m-th component $\tilde{\tilde{\tilde{f}}}_m(\boldsymbol{x})$ of $\tilde{\tilde{\tilde{\boldsymbol{f}}}}(\boldsymbol{x})$ as follows:

$$\tilde{\tilde{\tilde{f}}}_m(\boldsymbol{x}) = \sum_{j=1}^{n_0} x_j. \tag{22}$$

For simplicity, we restrict the domain of the NN to $[0,1]^{n_0}$. Then, $\boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be expressed using these functions and ReLU $\boldsymbol{g}$ as follows:

$$\begin{aligned} \boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \boldsymbol{f}^{(L)}\circ\boldsymbol{g}^{(L-1)}\circ\boldsymbol{f}^{(L-1)}\circ\cdots\circ\boldsymbol{g}^{(1)}\circ\boldsymbol{f}^{(1)}(\boldsymbol{x}) && (23)\\ &= \tilde{\tilde{\tilde{\boldsymbol{f}}}}\circ\underbrace{\tilde{\tilde{\boldsymbol{f}}}\circ\boldsymbol{g}\circ\tilde{\boldsymbol{f}}}_{\boldsymbol{h}}\circ\underbrace{\tilde{\tilde{\boldsymbol{f}}}\circ\boldsymbol{g}\circ\tilde{\boldsymbol{f}}}_{\boldsymbol{h}}\circ\cdots\circ\underbrace{\tilde{\tilde{\boldsymbol{f}}}\circ\boldsymbol{g}\circ\tilde{\boldsymbol{f}}}_{\boldsymbol{h}} && (24)\\ &= \tilde{\tilde{\tilde{\boldsymbol{f}}}}\circ\boldsymbol{h}\circ\boldsymbol{h}\circ\cdots\circ\boldsymbol{h}. && (25)\end{aligned}$$

Note that $\boldsymbol{h} = \tilde{\tilde{\boldsymbol{f}}}\circ\boldsymbol{g}\circ\tilde{\boldsymbol{f}}$ is a function from $[0,1]^{n_0}$ to $[0,1]^{n_0}$. Also, the j-th component of $\boldsymbol{h}(\boldsymbol{x})$ is expressed as

$$\begin{aligned} h_j(\boldsymbol{x}) = &\max\{0, px_j\} - \max\{0, 2px_j - 2\} +\\ &\cdots + (-1)^{p-1}\max\{0, 2px_j - 2(p-1)\} \end{aligned} \tag{26}$$

and is a one-dimensional piecewise linear function that depends only on x_j, as shown in Fig. 6.

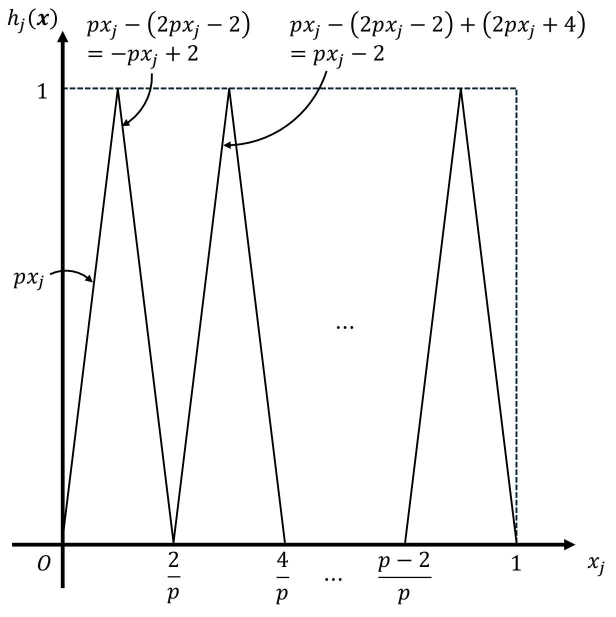

Figure 6: A graph of $h_j(\boldsymbol{x})$

Therefore, for any $t \in \{0, 1, \ldots, p-1\}$, an interval $\left[\frac{t}{p}, \frac{t+1}{p}\right]$ becomes a linear region of h_j, and $h_j|_{\left[\frac{t}{p}, \frac{t+1}{p}\right]}$ becomes a bijection to $[0, 1]$. Since $\boldsymbol{h}$ is a function consisting of h_j as each component, for any $(t_1, t_2, \ldots, t_{n_0}) \in \{0, 1, \ldots, p-1\}^{n_0}$, the Cartesian product $\prod_{j=1}^{n_0} \left[\frac{t_j}{p}, \frac{t_j+1}{p}\right]$ becomes a linear region of $\boldsymbol{h}$, and $\boldsymbol{h}|_{\prod_{j=1}^{n_0}\left[\frac{t_j}{p}, \frac{t_j+1}{p}\right]}$ becomes a bijection to $[0, 1]^{n_0}$.

Here, since $|\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}})| = 1$ and $|\mathcal{L}(\boldsymbol{h})| = |\{0, 1, \ldots, p-1\}^{n_0}| = p^{n_0}$, using Lemma 1, the following holds:

$$|\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}} \circ \boldsymbol{h})| = |\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}})||\mathcal{L}(\boldsymbol{h})| = 1 \cdot p^{n_0} = p^{n_0}. \tag{27}$$

Similarly, the following also holds:

$$|\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}} \circ \boldsymbol{h} \circ \boldsymbol{h})| = |\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}} \circ \boldsymbol{h})||\mathcal{L}(\boldsymbol{h})| = p^{n_0} \cdot p^{n_0} = p^{2n_0}. \tag{28}$$

By repeating this recursively, we obtain

$$|\mathcal{L}(\boldsymbol{F_\theta})| = |\mathcal{L}(\tilde{\tilde{\boldsymbol{f}}} \circ \boldsymbol{h} \circ \boldsymbol{h} \circ \cdots \circ \boldsymbol{h})| = p^{n_0(L-1)} \tag{29}$$

Therefore, Proposition 1 is proven. □