

Learning Neural Networks by Neuron Pursuit

Akshay Kumar

*Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN*

kumar511@umn.edu

Jarvis Haupt

*Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN*

jdhaupt@umn.edu

Abstract

The first part of this paper studies the evolution of gradient flow for homogeneous neural networks near a class of saddle points exhibiting a sparsity structure. The choice of these saddle points is motivated from previous works on homogeneous networks, which identified the first saddle point encountered by gradient flow after escaping the origin. It is shown here that, when initialized sufficiently close to such saddle points, gradient flow remains near the saddle point for a sufficiently long time, during which the set of weights with small norm remain small but converge in direction. Furthermore, important empirical observations are made on the behavior of gradient descent *after* escaping these saddle points. The second part of the paper, motivated by these results, introduces a greedy algorithm to train deep neural networks called *Neuron Pursuit* (NP). It is an iterative procedure which alternates between expanding the network by adding neuron(s) with carefully chosen weights, and minimizing the training loss using this augmented network. The efficacy of the proposed algorithm is validated using numerical experiments.

1 Introduction

Deep neural networks trained using gradient-based methods exhibit remarkable generalization performance. Despite this empirical success, our theoretical understanding of why and how these networks perform so well remains limited. A widely held hypothesis is that the *implicit regularization* induced by the training algorithm plays a pivotal role in this success (Soudry et al., 2018). This belief has motivated extensive research into the dynamics of neural network training, yielding several important insights (Jacot et al., 2018; Chizat et al., 2019; Mei et al., 2019; Lyu & Li, 2020; Yang & Hu, 2021). Nevertheless, a comprehensive theoretical understanding of the training dynamics is still lacking.

An important insight emerging from these works is that the scale of initialization dictates two fundamentally different training regimes. In the large initialization regime—referred to as the Neural Tangent Kernel (NTK) regime (Jacot et al., 2018) or the *lazy* regime (Chizat et al., 2019)—the weights of the neural network remain close to the initialization throughout the training, and the training dynamics is linear. This is in stark contrast to the small initialization regime, where the weights change significantly during training and the training dynamics is extremely non-linear. This regime is also known as the *feature learning* regime (Geiger et al., 2020; Yang & Hu, 2021; Mei et al., 2019; Woodworth et al., 2020), since the weights adapt to the underlying features present in the data. While the large initialization regime has been analyzed in considerable depth, our theoretical understanding of the small initialization regime remains relatively limited.

Early investigations into the small initialization regime studied the limiting behavior of gradient-based methods. In diagonal linear networks, gradient flow with vanishing initialization converges to minimum ℓ_1 -norm solution (Woodworth et al., 2020; Vaskevicius et al., 2019). In fully-connected linear networks, small initialization has been empirically observed to bias the training dynamics toward low-rank solutions (Gunasekar et al., 2017; Arora et al., 2019), with rigorous guarantees established for matrix factorization

(Jiang et al., 2023; Chou et al., 2024) and matrix sensing using two-layer networks (Stöger & Soltanolkotabi, 2021; Jin et al., 2023; Xiong et al., 2024; Ma & Fattah, 2024). For small initialization, empirical evidence indicates that non-linear neural networks also seek sparse solutions (Chizat et al., 2019), and theoretical results have been obtained in many cases involving shallow networks such as linearly separable data (Phuong & Lampert, 2021; Lyu et al., 2021; Wang & Ma, 2023), orthogonal or nearly orthogonal data (Boursier et al., 2022; Frei et al., 2023), and learning the XOR function (Glasgow, 2024; Brutzkus & Globerson, 2019). Recent works have also explored the ability of neural networks to learn sparse functions, such as single- or multi-index functions, though these too are largely confined to shallow networks (Bietti et al., 2022; Abbe et al., 2022; Damian et al., 2022; Lee et al., 2024; Dandi et al., 2023). Overall, it is conjectured that deep neural networks trained via gradient descent with small initialization, are implicitly regularized toward sparse or low-complexity solutions. However, our theoretical understanding of this phenomenon remains incomplete.

A more recent line of work has sought to understand the early training dynamics of deep *homogeneous* neural networks in the small initialization regime (Kumar & Haupt, 2024; 2025a; Maennel et al., 2018; Atanasov et al., 2022; Boursier & Flammarion, 2024). These studies show that, for sufficiently small initialization, the weights remain small in norm and near the origin during the early stages of training but converge in direction—a phenomenon referred to as early directional convergence. Moreover, in feed-forward deep homogeneous networks, the weights converge to a direction such that the norm of incoming and outgoing weights of each hidden neuron is proportional. Consequently, if the incoming weights of a hidden neuron are zero, its outgoing weights must also be zero, and vice versa. Empirically, Kumar & Haupt (2025a) observed that many hidden neurons exhibit this behavior, with their incoming and outgoing weights both becoming zero in the early stages of training—resulting in the emergence of a sparsity structure among the weights.

Building on these findings, Kumar & Haupt (2025b) investigate the subsequent phase of training—the gradient flow dynamics of homogeneous neural networks after the weights escape from the origin. They show that, after escaping the origin, the weights get close to a saddle point of the training loss, and they characterize this saddle point. Moreover, for feed-forward homogeneous neural networks, the sparsity structure which emerges among the weights before the escape is preserved throughout this phase. In particular, the set of neurons whose incoming and outgoing weights became zero during the early stages of training, those weights remained zero even after escaping from the origin and until reaching the saddle point.

In the small initialization regime, a complementary line of research has observed an intriguing phenomenon in the trajectory of gradient descent, known as *saddle-to-saddle dynamics* (Jacot et al., 2021; Li et al., 2021). Over the course of training, gradient descent passes through a sequence of saddle points, where the network appears to increase its complexity as it moves from one saddle point to another. This is also reflected in the loss curve, which alternates between long plateaus and sharp drops. This phenomenon has also been referred to as *incremental learning* (Gidel et al., 2019; Gissin et al., 2020; Razin et al., 2022), as the network learns increasingly complex functions in phases. Formal results on this phenomenon has been established in certain settings, such as linear neural networks (Pesme & Flammarion, 2023; Jin et al., 2023; Abbe et al., 2024; Simon et al., 2023) and two-layer nonlinear networks for specific training data (Boursier et al., 2022; Berthier et al., 2023; Wang & Ma, 2023; Abbe et al., 2023). Notably, the work of Kumar & Haupt (2025b) can be interpreted as characterizing the first saddle point encountered by gradient flow after escaping the origin, for homogeneous neural networks. However, establishing saddle-to-saddle dynamics throughout the entire course of training, especially in deeper neural networks, remains an open problem.

1.1 Our Contributions

In the first part of this paper, we study the gradient flow dynamics of homogeneous neural networks near saddle points with a sparsity structure. For feed-forward neural networks, inspired from the work of Kumar & Haupt (2025b), we consider saddle points where a subset of hidden neurons has both incoming and outgoing weights with zero norm. In Theorem 7, we show that when initialized sufficiently close to such saddle points, gradient flow remains near the saddle point for a sufficiently long time. During this period, the subset of weights initialized with small norm remain small in norm but converge in direction. Moreover, they converge to a direction such that the norm of incoming and outgoing weights are proportional. Overall, the training dynamics near these saddle points share many similarities with the training dynamics near the origin, as described previously in Kumar & Haupt (2024; 2025a).

Establishing our results requires overcoming key obstacles absent in prior works. The analyses in Kumar & Haupt (2024; 2025a) establish directional convergence for the *entire* set of weights, by exploiting the homogeneity of the network output with respect to all parameters. In contrast, our setting requires proving directional convergence for only a *subset* of weights, for which the output is *not* homogeneous—a key difference from earlier analyses. While Kumar & Haupt (2024) also considered saddle points with a sparsity structure, they imposed a separability assumption on the network architecture that effectively ensured homogeneity with respect to the relevant weights. Such assumptions do not hold for fully-connected networks, making their techniques inapplicable in our setting. Our key technical contribution is to show that, near the saddle point, the network output can be decomposed into a term homogeneous in the desired subset of weights and another term independent of them. This decomposition, which relies crucially on the sparsity structure, is valid only in the neighborhood of the saddle point and enables us to establish directional convergence.

In Section 4, we present empirical observations on the dynamics of gradient descent after escaping these saddle points. For feed-forward neural networks, we observe that, after escaping the saddle point, the trajectory gets close to another saddle point, consistent with the saddle-to-saddle dynamics hypothesis. Notably, the sparsity structure that emerges among the weights with small norm near the saddle point is preserved, even after escaping it and until reaching the next saddle point. Moreover, this new saddle point exhibits a similar sparsity structure, with incoming and outgoing weights of a subset of hidden neurons being zero. While we cannot rigorously establish these empirical observations, they suggest strong similarities between the dynamics of gradient descent after escaping these saddle point and after escaping the origin, as described in Kumar & Haupt (2025b).

Drawing on these insights, in Section 5, we present a greedy algorithm to train deep neural networks, which we call *Neuron Pursuit* (NP). At a high level, NP is inspired from the saddle-to-saddle dynamics hypothesis and builds the network by moving from one saddle point of the training loss to another. It further leverages the sparsity structure that emerges at these saddle points and after escaping them, as studied in this paper and previous works. Concretely, the algorithm begins by considering a neural network with one neuron in every layer, and then trains it to minimize the training loss via gradient descent using specifically chosen initial weights. It then proceeds iteratively: at each iteration, a neuron is added to the network with specifically chosen incoming and outgoing weights, after which the training loss is minimized via gradient descent using this augmented network. Compared to the traditional back-propagation algorithm, where the set of neurons in neural networks are fixed and it is trained end-to-end, in NP algorithm the size of the neural networks gradually increases as training progresses. Our experiments demonstrate that the NP algorithm is able to successfully learn sparse non-linear functions.

Overall, our work is a step towards demystifying neural networks. The theoretical analyses deepen our understanding of their training dynamics, while the NP algorithm provides an alternative lens for understanding how feature learning unfolds in deep networks. Together, these contributions brings us closer to a principled understanding of mechanisms that drive generalization in deep networks.

2 Background

We use \mathbb{N} to denote the set of natural numbers, and for any $L \in \mathbb{N}$, we let $[L] := \{1, 2, \dots, L\}$. For vectors, $\|\cdot\|_2$ denotes the ℓ_2 -norm. For matrices, $\|\cdot\|_F$ and $\|\cdot\|_2$ denote the Frobenius and spectral norms, respectively. For a matrix \mathbf{W} , $\mathbf{W}[:, j]$ and $\mathbf{W}[j, :]$ denote its j -th column and j -th row, respectively. For a vector \mathbf{p} , its i th entry is denoted by p_i . The d -dimensional unit sphere is denoted by \mathbb{S}^{d-1} . We use \odot to denote elementwise multiplication between vectors or matrices. A KKT point of an optimization problem is called a non-negative (positive, zero) KKT point if the objective value at the KKT point is non-negative (positive, zero).

Homogeneous neural networks. For a neural network \mathcal{H} , $\mathcal{H}(\mathbf{x}; \mathbf{w})$ denotes its output, where $\mathbf{x} \in \mathbb{R}^d$ is the input and $\mathbf{w} \in \mathbb{R}^k$ is a vector containing all the weights. A neural network \mathcal{H} is referred to as *L-(positively) homogeneous* if

$$\mathcal{H}(\mathbf{x}; c\mathbf{w}) = c^L \mathcal{H}(\mathbf{x}; \mathbf{w}), \text{ for all } c \geq 0 \text{ and } \mathbf{w} \in \mathbb{R}^k.$$

Suppose $\{\mathbf{x}_i, y_i\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$ is the training data, and let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathbf{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$. Let $\mathcal{H}(\mathbf{X}; \mathbf{w}) = [\mathcal{H}(\mathbf{x}_1; \mathbf{w}), \dots, \mathcal{H}(\mathbf{x}_n; \mathbf{w})] \in \mathbb{R}^n$ be the vector containing outputs of the neural network, and

$\mathcal{J}(\mathbf{X}; \mathbf{w})$ denotes the Jacobian of $\mathcal{H}(\mathbf{X}; \mathbf{w})$ with respect to \mathbf{w} . Assuming square loss is used for training, the training loss can be written as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (\mathcal{H}(\mathbf{x}_i; \mathbf{w}) - y_i)^2 = \frac{1}{2} \|\mathcal{H}(\mathbf{X}; \mathbf{w}) - \mathbf{y}\|_2^2. \quad (1)$$

Minimizing the above optimization problem using gradient flow gives us the following differential equation:

$$\dot{\mathbf{w}} = -\nabla \mathcal{L}(\mathbf{w}) = -\mathcal{J}(\mathbf{X}; \mathbf{w})^\top (\mathcal{H}(\mathbf{X}; \mathbf{w}) - \mathbf{y}). \quad (2)$$

We will use $\psi(t, \mathbf{w}(0))$ to denote the solution of above differential equation, where $\mathbf{w}(0)$ is the initialization.

Feed-forward neural network. Suppose $L \geq 2$, then the output of an L -layer feed-forward neural network \mathcal{H} is defined as

$$\mathcal{H}(\mathbf{x}; \mathbf{W}_1, \dots, \mathbf{W}_L) = \mathbf{W}_L \sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}) \dots), \quad (3)$$

where $\mathbf{W}_l \in \mathbb{R}^{k_l \times k_{l-1}}$, $k_0 = d$ and $k_L = 1$, and the activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is applied elementwise. Note that, $\mathbf{W}_l[j, :]$ contains the incoming weights to the j -th neuron in the l -th layer, and $\mathbf{W}_{l+1}[:, j]$ contains the outgoing weights from the same neuron. Also, if $\sigma(x) = \max(x, \alpha x)^p$, for some $p \in \mathbb{N}$ and $\alpha \in \mathbb{R}$, then the above neural network is positively homogeneous with respect to its weights.

We next briefly review the results of Kumar & Haupt (2024; 2025a;b), which studies the phenomenon of early directional convergence in homogeneous neural networks and the dynamics of gradient flow after the weights escape from the origin. These works are particularly relevant, as our analysis builds upon and extends them. Moreover, the behavior of gradient flow near and beyond the saddle points studied in this paper, share many similarities with these earlier results.

2.1 Early Directional Convergence

For neural networks with degree of homogeneity two or higher, the origin is a critical point of the training loss in eq. (1). Hence, if initialized near the origin, gradient flow will remain near the origin for some time before eventually escaping it. In Kumar & Haupt (2024; 2025a), the authors analyze the training dynamics while the trajectory remains close to the origin. To better describe their results, we introduce some basic concepts. For a vector \mathbf{z} and neural network \mathcal{H} , the Neural Correlation Function (NCF) is defined as

$$\mathcal{N}_{\mathbf{z}, \mathcal{H}}(\mathbf{u}) = \mathbf{z}^\top \mathcal{H}(\mathbf{X}; \mathbf{u}). \quad (4)$$

The NCF could be viewed as measuring the correlation between the vector \mathbf{z} and the output of the neural network. We use $\tilde{\mathcal{N}}_{\mathbf{z}, \mathcal{H}}(\mathbf{u})$ to denote the corresponding constrained NCF problem which is defined as

$$\tilde{\mathcal{N}}_{\mathbf{z}, \mathcal{H}}(\mathbf{u}) := \max_{\mathbf{u}} \mathcal{N}_{\mathbf{z}, \mathcal{H}}(\mathbf{u}), \text{ s.t. } \|\mathbf{u}\|_2^2 = 1. \quad (5)$$

We omit the dependent variable in the definition of the constrained NCF and NCF when it is clear from context. Next, consider the (positive) gradient flow of the NCF:

$$\dot{\mathbf{u}} = \nabla \mathcal{N}_{\mathbf{z}, \mathcal{H}}(\mathbf{u}). \quad (6)$$

We use $\phi(t, \mathbf{u}(0); \mathcal{N}_{\mathbf{z}, \mathcal{H}})$ to denote the solution of above differential equation, where $\mathbf{u}(0)$ is the initialization. The following lemma from Kumar & Haupt (2024; 2025a) describes the limiting dynamics of the gradient flow of the NCF, showing that it either converges to the origin or goes to infinity and converges in direction.

Lemma 1. *Suppose \mathcal{H} is L -homogeneous, for some $L \geq 2$. For any vector \mathbf{z} and initialization $\mathbf{u}_0 \in \mathbb{R}^k$,*

- either $\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})$ converges to the origin,
- or $\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})$ goes to infinity and $\frac{\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})}{\|\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})\|_2}$ converges in direction to a non-negative KKT point of $\tilde{\mathcal{N}}_{\mathbf{z}, \mathcal{H}}$.

We next define the notion of stable set for a non-negative KKT point of the constrained NCF.

Definition 2.1. *The stable set $\mathcal{S}(\mathbf{u}_*; \mathcal{N}_{\mathbf{z}, \mathcal{H}})$ of a non-negative KKT point \mathbf{u}_* of $\tilde{\mathcal{N}}_{\mathbf{z}, \mathcal{H}}$ is the set of all unit-norm initializations such that gradient flow of the NCF converges in direction to \mathbf{u}_* :*

$$\mathcal{S}(\mathbf{u}_*; \mathcal{N}_{\mathbf{z}, \mathcal{H}}) := \left\{ \mathbf{u}_0 \in \mathbb{S}^{k-1} : \frac{\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})}{\|\phi(t, \mathbf{u}_0; \mathcal{N}_{\mathbf{z}, \mathcal{H}})\|_2} \rightarrow \mathbf{u}_* \right\}$$

The next lemma describes the phenomenon of early directional convergence during the early stages of training (Kumar & Haupt, 2024; 2025a).

Lemma 2. *Suppose \mathbf{w}_0 is a unit-norm vector. For any arbitrarily small $\epsilon > 0$, there exists T such that for all sufficiently small δ we have*

$$\|\psi(t, \delta \mathbf{w}_0)\|_2 = O(\delta), \text{ for all } t \in [0, T/\delta^{L-2}].$$

Furthermore, if $\mathbf{w}_0 \in \mathcal{S}(\mathbf{w}_*; \mathcal{N}_{\mathbf{y}, \mathcal{H}})$, where \mathbf{w}_* is a non-negative KKT point of $\tilde{\mathcal{N}}_{\mathbf{y}, \mathcal{H}}$, then

$$\|\psi(T/\delta^{L-2}, \delta \mathbf{w}_0)\|_2 \geq \delta \eta \text{ and } \frac{\psi(T/\delta^{L-2}, \delta \mathbf{w}_0)^\top \mathbf{w}_*}{\|\psi(T/\delta^{L-2}, \delta \mathbf{w}_0)\|_2} = 1 - O(\epsilon),$$

else, $\|\psi(T/\delta^{L-2}, \delta \mathbf{w}_0)\|_2 = \epsilon \cdot O(\delta)$. Here, η is a positive constant independent of ϵ and δ .

The above lemma describes the evolution of weights under gradient flow with initialization $\delta \mathbf{w}_0$, where $\delta > 0$ is a scalar that controls the scale of initialization. It shows that for small initialization, the weights remain small during the early stages of training. Moreover, if the initial direction \mathbf{w}_0 belongs to the stable set of a non-negative KKT point of $\tilde{\mathcal{N}}_{\mathbf{y}, \mathcal{H}}$, the constrained NCF defined with respect to \mathbf{y} and \mathcal{H} , then the weights approximately converge in direction towards that KKT point. Also, if \mathbf{w}_0 does not belong to the stable set of a non-negative KKT point, then $\phi(t, \mathbf{w}_0; \mathcal{N}_{\mathbf{y}, \mathcal{H}})$ converges to the origin (see Lemma 1). In such cases, instead of directional convergence, the weights approximately become zero, as $\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2 = \epsilon \cdot O(\delta)$, where ϵ and δ are both small. In contrast, in the previous case, $\|\mathbf{w}_z(T/\delta^{L-2})\|_2 \geq \delta \eta$, where η is a constant.

For feed-forward homogeneous neural networks, the next lemma states an important property of *positive* KKT points of the constrained NCF.

Lemma 3. *Let \mathcal{H} be an L -layer feed-forward neural network as in eq. (3), where $\sigma(x) = \max(x, \alpha x)^p$, for some $p \in \mathbb{N}$ and $\alpha \in \mathbb{R}$. Let $(\bar{\mathbf{W}}_1, \dots, \bar{\mathbf{W}}_L)$ be a positive KKT point of*

$$\max_{\mathbf{W}_1, \dots, \mathbf{W}_L} \mathcal{N}_{\mathbf{z}, \mathcal{H}}(\mathbf{W}_1, \dots, \mathbf{W}_L) := \mathbf{z}^\top \mathcal{H}(\mathbf{X}; \mathbf{W}_1, \dots, \mathbf{W}_L), \text{ s.t. } \sum_{i=1}^L \|\mathbf{W}_i\|_F^2 = 1. \quad (7)$$

Then, $\|\bar{\mathbf{W}}_l[j, :]\|_2^2 = p \|\bar{\mathbf{W}}_{l+1}[:, j]\|_2^2$, for all $j \in [k_l]$ and $l \in [L-1]$.

In the above lemma, the condition $\|\bar{\mathbf{W}}_l[j, :]\|_2^2 = p \|\bar{\mathbf{W}}_{l+1}[:, j]\|_2^2$ implies that the norm of each hidden neuron's incoming weights is proportional to the norm of its outgoing weights. Consequently, if the incoming weights of a neuron have zero norm, its outgoing weights must also have zero norm, and vice-versa.

Since gradient flow converges in direction to a KKT point of the constrained NCF in the early stages of training, the weights in the early stages will also satisfy this property. Empirically, the weights usually converge to a KKT point where only a few neurons have non-zero incoming and outgoing weights, leading to the emergence of a sparsity structure among the weights in the early stages of training. In fact, for $p \geq 2$, Kumar & Haupt (2025a) observed that typically only a single neuron in each layer had non-zero incoming and outgoing weights. For $p = 1$, multiple neurons in each layer had non-zero incoming and outgoing weights; however, the rank of all the weight matrices were typically one, and the resulting network output could be expressed using one neuron per layer.

Note that, if the incoming and outgoing weights of a hidden neuron are zero, then the output of that neuron is zero. Such a neuron can be considered inactive, as it does not contribute to the overall output of the network. Hence, in the early stages of training, certain hidden neurons become approximately inactive because the weights converge towards a KKT point of the constrained NCF.

2.2 Gradient Flow Dynamics Beyond the Origin

We now discuss the results of Kumar & Haupt (2025b), which studies the gradient flow dynamics of homogeneous neural networks after escaping the origin. They also characterize the first saddle point encountered by gradient flow after escaping the origin.

Lemma 4. *Suppose \mathcal{H} is an L -homogeneous neural network, for some $L \geq 2$, and $\mathbf{w}_0 \in \mathcal{S}(\mathbf{w}_*; \mathcal{N}_{\mathbf{y}, \mathcal{H}})$, where \mathbf{w}_* is a second-order positive KKT point of $\tilde{\mathcal{N}}_{\mathbf{y}, \mathcal{H}}$. Let $\tilde{T} \in (-\infty, \infty)$ be arbitrarily large, then for all sufficiently small $\delta > 0$,*

$$\|\psi(t + T_\delta, \delta\mathbf{w}_0) - \mathbf{p}(t)\|_2 = O(\delta^\beta), \text{ for all } t \in [-\tilde{T}, \tilde{T}], \quad (8)$$

where $\beta > 0$, T_δ is some function of δ , and $\mathbf{p}(t)$ is defined as:

$$\begin{aligned} \mathbf{p}(t) &:= \lim_{\delta \rightarrow 0} \psi\left(t + \frac{\ln(1/\delta)}{2\mathcal{N}_{\mathbf{y}, \mathcal{H}}(\mathbf{w}_*)}, \delta\mathbf{w}_*\right), \text{ if } L = 2, \text{ and} \\ \mathbf{p}(t) &:= \lim_{\delta \rightarrow 0} \psi\left(t + \frac{1/\delta^{L-2}}{L(L-2)\mathcal{N}_{\mathbf{y}, \mathcal{H}}(\mathbf{w}_*)}, \delta\mathbf{w}_*\right), \text{ if } L > 2. \end{aligned}$$

Furthermore, let $\epsilon > 0$ be arbitrarily small, and let $\mathbf{p}^* = \lim_{t \rightarrow \infty} \mathbf{p}(t)$, where \mathbf{p}^* is a saddle point of the training loss. Then, there exists a T_ϵ such that for all for all sufficiently small $\delta > 0$,

$$\|\psi(T_\epsilon + T_\delta, \delta\mathbf{w}_0) - \mathbf{p}^*\|_2 \leq \epsilon. \quad (9)$$

To interpret this result, recall that $\psi(t, \delta\mathbf{w}_*)$ denotes the gradient flow trajectory initialized at $\delta\mathbf{w}_*$. For small δ , the trajectory remains near the origin initially. Roughly speaking, for $L = 2$, $\psi(t, \delta\mathbf{w}_*)$ escapes from the origin after $\frac{\ln(1/\delta)}{2\mathcal{N}_{\mathbf{y}, \mathcal{H}}(\mathbf{w}_*)}$ time has elapsed; for $L > 2$, this time scales as $\frac{1/\delta^{L-2}}{L(L-2)\mathcal{N}_{\mathbf{y}, \mathcal{H}}(\mathbf{w}_*)}$. Therefore, $\mathbf{p}(t)$ is the limiting path taken by $\psi(t, \delta\mathbf{w}_*)$ after escaping from the origin, as scale of initialization approaches zero.

According to eq. (8), if \mathbf{w}_0 lies in the stable set of \mathbf{w}_* , then the gradient flow trajectory $\psi(t + T_\delta, \delta\mathbf{w}_0)$ remains close to $\mathbf{p}(t)$ for all $t \in [-\tilde{T}, \tilde{T}]$ and for all sufficiently small δ . That is, the trajectory of $\psi(t + T_\delta, \delta\mathbf{w}_0)$ after escaping the origin is same as the limiting trajectory $\mathbf{p}(t)$ for an arbitrarily long time and for sufficiently small initialization. Therefore, using the definition of $\mathbf{p}(t)$, the gradient flow trajectory initialized at $\delta\mathbf{w}_0$ escapes from the origin along the same path as if initialized at $\delta\mathbf{w}_*$, for all small δ . Thus, the escape dynamics are determined by the KKT point \mathbf{w}_* , regardless of the specific choice of \mathbf{w}_0 in its stable set.

Eventually, $\mathbf{p}(t)$ converges to a saddle point \mathbf{p}^* of the training loss, and according to eq. (9), gradient flow $\psi(t, \delta\mathbf{w}_0)$ also gets arbitrarily close to this saddle point at some time. However, the lemma does not assert that gradient flow converges to \mathbf{p}^* ; it may eventually escape from \mathbf{p}^* . The work of Kumar & Haupt (2025b) does not address the behavior of gradient flow near or beyond this saddle point.

For homogeneous feed-forward neural networks, the next lemma describes the sparsity structure present in the saddle point reached by gradient flow after escaping the origin. For brevity, we use $\mathbf{W}_{1:L}$ to denote the weight matrices of an L -layer network, that is, $\mathbf{W}_{1:L} := (\mathbf{W}_1, \dots, \mathbf{W}_L)$.

Lemma 5. *Suppose \mathcal{H} is an L -layer feed-forward neural network as defined in eq. (3), where $L \geq 2$, $\sigma(x) = \max(x, \alpha x)^p$, for $p = 1, \alpha = 1$ or $p \in \mathbb{N}, p \geq 2$ and $\alpha \in \mathbb{R}$. Let $\mathbf{W}_{1:L}^0 \in \mathcal{S}(\overline{\mathbf{W}}_{1:L}; \mathcal{N}_{\mathbf{y}, \mathcal{H}})$, where $\overline{\mathbf{W}}_{1:L}$ is a second-order positive KKT point of $\tilde{\mathcal{N}}_{\mathbf{y}, \mathcal{H}}$. Define subset of the weights \mathbf{w}_z as:*

$$\mathbf{w}_z := \bigcup_{l=1}^{L-1} \left(\{\mathbf{W}_l[j, :] : \|\overline{\mathbf{W}}_l[j, :]\|_2 = 0, j \in [k_l]\} \cup \{\mathbf{W}_{l+1}[:, j] : \|\overline{\mathbf{W}}_{l+1}[:, j]\|_2 = 0, j \in [k_l]\} \right).$$

Let $\tilde{T} \in (-\infty, \infty)$ be arbitrarily large, then for all sufficiently small $\delta > 0$,

$$\|\psi(t + T_\delta, \delta\mathbf{W}_{1:L}^0) - \mathbf{p}(t)\|_2 = O(\delta^\beta), \text{ for all } t \in [-\tilde{T}, \tilde{T}], \quad (10)$$

where $\beta > 0$, T_δ is some function of δ , and $\mathbf{p}(t)$ is defined in the same way as in Lemma 4. Furthermore, let $\epsilon > 0$ be arbitrarily small. Then, there exists a T_ϵ such that for all sufficiently small $\delta > 0$,

$$\|\psi(T_\epsilon + T_\delta, \delta \mathbf{W}_{1:L}^0) - \mathbf{p}^*\|_2 \leq \epsilon, \quad (11)$$

where \mathbf{p}^* is a saddle point of the training loss function and is defined in the way as in Lemma 4. Finally,

$$\|\mathbf{p}_{\mathbf{w}_z}(t)\|_2 = 0, \text{ for all } t \in (-\infty, \infty), \text{ and } \|\psi_{\mathbf{w}_z}(t + T_\delta, \delta \mathbf{W}_{1:L}^0)\|_2 = O(\delta^\beta), \text{ for all } t \in [-\tilde{T}, \tilde{T}], \quad (12)$$

which implies $\|\mathbf{p}_{\mathbf{w}_z}^*\|_2 = 0$ and $\|\psi_{\mathbf{w}_z}(T_\epsilon + T_\delta, \delta \mathbf{W}_{1:L}^0)\|_2 \leq \epsilon$, where $(\cdot)_{\mathbf{w}_z}$ denotes the sub-vector corresponding to weights in \mathbf{w}_z .

In this lemma, $\overline{\mathbf{W}}_{1:L}$ is a positive KKT point of the constrained NCF defined with respect to \mathbf{y} and \mathcal{H} . The set \mathbf{w}_z consists of all rows and columns of the weight matrices where the corresponding rows and columns of $\overline{\mathbf{W}}_{1:L}$ have zero norm. As shown in Lemma 3, this construction ensures that if $\mathbf{W}_{l+1}[:, j] \in \mathbf{w}_z$, then $\mathbf{W}_l[j, :] \in \mathbf{w}_z$, and vice-versa. Thus, the set \mathbf{w}_z will contain the incoming and outgoing weights of certain subset of hidden neurons.

If $\mathbf{W}_{1:L}^0$ lies in the stable set of $\overline{\mathbf{W}}_{1:L}$, then—as in the previous lemma—the gradient flow trajectory stays close to the limiting path $\mathbf{p}(t)$ after escaping the origin, eventually getting close to the saddle point \mathbf{p}^* , as stated in eq. (11). More importantly, eq. (12) states that $\|\mathbf{p}_{\mathbf{w}_z}(t)\|_2 = 0$, implying that along the trajectory of $\mathbf{p}(t)$, the weights belonging to \mathbf{w}_z have zero norm. Consequently, $\|\psi_{\mathbf{w}_z}(t + T_\delta, \delta \mathbf{W}_{1:L}^0)\|_2$ is small, $\|\mathbf{p}_{\mathbf{w}_z}^*\|_2 = 0$ and $\|\psi_{\mathbf{w}_z}(T_\epsilon + T_\delta, \delta \mathbf{W}_{1:L}^0)\|_2$ is small. Thus, the weights belonging to \mathbf{w}_z remain small throughout the time interval during which the gradient flow $\psi(t, \delta \mathbf{W}_{1:L}^0)$ escapes from the origin and gets close to the saddle point \mathbf{p}^* .

To summarize, for homogeneous feed-forward neural networks, Lemma 2 shows that during the early stages of training, weights remain small in norm but converge in direction towards $\overline{\mathbf{W}}_{1:L}$. Since $\overline{\mathbf{W}}_{1:L}$ exhibits a sparsity structure—specifically, the weights belonging to \mathbf{w}_z are zero—the same sparsity structure would also emerge among the weights during the early stages of training. The above lemma further says that this sparsity structure is preserved even after gradient flow escapes from the origin and until it reaches a saddle point, since weights belonging to \mathbf{w}_z remain small during this time interval.

An alternative perspective on this behavior is through the lens of hidden neurons. In the early stages of training, directional convergence toward $\overline{\mathbf{W}}_{1:L}$ causes a subset of hidden neurons to become inactive, as their incoming and outgoing weights become small. Moreover, these neurons remain inactive even after gradient flow escapes from the origin and reaches the next saddle point. As a result, until reaching the first saddle point, the training loss is essentially minimized using only the active neurons, through gradient flow initialized with weights that are small in norm and aligned in direction with a KKT point of the constrained NCF.

Another key takeaway is that gradient flow reaches a saddle point where the incoming and outgoing weights of a certain subset of hidden neurons are zero. This structural property of the saddle point will be crucial in our analysis of gradient flow dynamics near saddle points in homogeneous neural networks.

3 Gradient Flow Dynamics Near Saddle Points

In this section, we study the gradient flow dynamics near saddle points of the training loss for homogeneous neural networks. Our focus is on a specific class of saddle points where a subset of the weights is zero. More precisely, we assume that the weights of the neural network can be divided into two sets, $\mathbf{w} = (\mathbf{w}_n, \mathbf{w}_z)$, such that $(\overline{\mathbf{w}}_n, \mathbf{0})$ is the saddle point of the training loss. Under this setup, the training loss can be expressed as

$$\mathcal{L}(\mathbf{w}_n, \mathbf{w}_z) = \frac{1}{2} \|\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{y}\|_2^2. \quad (13)$$

Minimizing the above optimization problem using gradient flow with initialization near the saddle point $(\overline{\mathbf{w}}_n, \mathbf{0})$ gives us the following differential equation:

$$\dot{\mathbf{w}}_n = -\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)^\top (\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{y}), \mathbf{w}_n(0) = \overline{\mathbf{w}}_n + \delta \mathbf{n} \quad (14)$$

$$\dot{\mathbf{w}}_z = -\mathcal{J}_z(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)^\top (\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{y}), \mathbf{w}_z(0) = \delta \mathbf{z}, \quad (15)$$

where $\delta > 0$ is a scalar that controls how close the initialization is to the saddle point, and $\|\mathbf{n}\|_2 = \|\mathbf{z}\|_2 = 1$. Here, $\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ and $\mathcal{J}_z(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ denote the Jacobian of $\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ with respect to \mathbf{w}_n and \mathbf{w}_z , respectively.

Next, we will make certain assumptions on output of the neural network. These assumptions are motivated from previous works which describe the saddle points encountered by gradient flow during training. To provide intuition for these assumptions and to outline our main results, we begin with an informal analysis of gradient flow dynamics of a homogeneous feed-forward neural networks near certain saddle points.

3.1 An Informal Analysis

Consider a three-layer neural network \mathcal{H} with activation function $\sigma(x) = x^2$ and its output is

$$\mathcal{H}(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) = \mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x})),$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{W}_1 \in \mathbb{R}^{k_1 \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{k_2 \times k_1}$ and $\mathbf{W}_3 \in \mathbb{R}^{1 \times k_2}$. Thus, \mathcal{H} has two hidden layers, with first and second layer containing k_1 and k_2 neurons, respectively. Next, we are going to divide the weights in the following way:

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{N}_1 \\ \mathbf{A}_1 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} \mathbf{N}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & \mathbf{C}_2 \end{bmatrix}, \mathbf{W}_3 = [\mathbf{N}_3 \quad \mathbf{B}_3],$$

where $\mathbf{N}_1 \in \mathbb{R}^{p_1 \times d}$, $\mathbf{A}_1 \in \mathbb{R}^{(k_1 - p_1) \times d}$, $\mathbf{N}_2 \in \mathbb{R}^{p_2 \times p_1}$, $\mathbf{C}_2 \in \mathbb{R}^{(k_2 - p_2) \times (k_1 - p_1)}$, $\mathbf{N}_3 \in \mathbb{R}^{1 \times p_2}$, $\mathbf{B}_3 \in \mathbb{R}^{1 \times (k_2 - p_2)}$, and \mathbf{B}_2 and \mathbf{A}_2 are defined in a consistent way. In the above division, the matrix \mathbf{A}_1 ($\mathbf{B}_2, \mathbf{C}_2$) contains all the incoming (outgoing) weights of the last $k_1 - p_1$ neurons of the first hidden layer. Similarly, \mathbf{B}_3 ($\mathbf{A}_2, \mathbf{C}_2$) contains all the outgoing (incoming) weights of the last $k_2 - p_2$ neurons of the second hidden layer. Let $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$ be a saddle point of the training loss such that all the incoming and outgoing weights of the last $k_1 - p_1$ neurons of the first layer and last $k_2 - p_2$ neurons of the second layer are zero. Therefore,

$$\bar{\mathbf{W}}_1 = \begin{bmatrix} \bar{\mathbf{N}}_1 \\ \mathbf{0} \end{bmatrix}, \bar{\mathbf{W}}_2 = \begin{bmatrix} \bar{\mathbf{N}}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \bar{\mathbf{W}}_3 = [\bar{\mathbf{N}}_3 \quad \mathbf{0}].$$

As discussed in Section 2.2, the choice of this saddle point is motivated from the result of Kumar & Haupt (2025b), which showed that after escaping from the origin, gradient flow reaches a saddle point where the incoming and outgoing weights of a subset of hidden neurons have zero norm. Our goal here is to (informally) analyze the gradient flow dynamics of the training loss when initialized near the above saddle point.

Let \mathbf{w}_z be the subset of the weights containing all the incoming and outgoing weights of the last $k_1 - p_1$ neurons of the first layer and last $k_2 - p_2$ neurons of the second layer, and \mathbf{w}_n contains the remaining weights. With this notation, the full set of network weights can be written as $(\mathbf{w}_n, \mathbf{w}_z)$, and the saddle point $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$ corresponds to the point $(\bar{\mathbf{w}}_n, \mathbf{0})$, for some $\bar{\mathbf{w}}_n$.

Suppose \mathbf{w}_n and \mathbf{w}_z are evolving according to eq. (14) and eq. (15), respectively, where δ is small. Since $(\bar{\mathbf{w}}_n, \mathbf{0})$ is a saddle point, the weights will remain near the saddle point for some time after training begins. Thus, in the initial stages of training, we can assume $\mathbf{w}_n \approx \bar{\mathbf{w}}_n$ and $\mathbf{w}_z \approx \mathbf{0}$. Define $\bar{\mathbf{y}} := \mathbf{y} - \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0})$, then the dynamics of \mathbf{w}_z can approximately be written as

$$\begin{aligned} \dot{\mathbf{w}}_z &\approx -\mathcal{J}_z(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)^\top (\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z) - \mathbf{y}) \approx -\mathcal{J}_z(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)^\top (\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathbf{y}) \\ &= \mathcal{J}_z(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)^\top \bar{\mathbf{y}}. \end{aligned} \tag{16}$$

To understand the behavior of $\mathcal{J}_z(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$, we will simplify $\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$. Since $\sigma(x) = x^2$, we get

$$\begin{aligned}\mathcal{H}(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z) &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \sigma \left(\begin{bmatrix} \bar{\mathbf{N}}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} (\bar{\mathbf{N}}_1 \mathbf{x})^2 \\ (\mathbf{A}_1 \mathbf{x})^2 \end{bmatrix} \right) \\ &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \sigma \left(\begin{bmatrix} \bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2 + \mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2 \\ \mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2 + \mathbf{C}_2(\mathbf{A}_1 \mathbf{x})^2 \end{bmatrix} \right) \\ &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \left[\begin{bmatrix} (\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2 + (\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2)^2 + 2(\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2) \\ (\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2 + (\mathbf{C}_2(\mathbf{A}_1 \mathbf{x})^2)^2 + 2(\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2) \odot (\mathbf{C}_2(\mathbf{A}_1 \mathbf{x})^2) \end{bmatrix} \right] \\ &= \bar{\mathbf{N}}_3(\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2 + 2\bar{\mathbf{N}}_3((\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2)) + \mathbf{B}_3(\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2 \\ &\quad + 2\mathbf{B}_3(\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2) \odot (\mathbf{C}_2(\mathbf{A}_1 \mathbf{x})^2) + \bar{\mathbf{N}}_3(\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2)^2 + \mathbf{B}_3(\mathbf{C}_2(\mathbf{A}_1 \mathbf{x})^2)^2.\end{aligned}$$

In the last term, the expression $\bar{\mathbf{N}}_3(\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2$ is independent of \mathbf{w}_z . The remaining terms, however, do depend on \mathbf{w}_z and are homogeneous functions of \mathbf{w}_z with different degrees of homogeneity. Since we have assumed \mathbf{w}_z to be small, we can just keep the term with lowest degree of homogeneity and ignore the higher order terms. Therefore,

$$\mathcal{H}(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z) \approx \bar{\mathbf{N}}_3(\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2 + 2\bar{\mathbf{N}}_3((\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2)) + \mathbf{B}_3(\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2.$$

Define $\mathcal{H}_1(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z) := 2\bar{\mathbf{N}}_3((\bar{\mathbf{N}}_2(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{B}_2(\mathbf{A}_1 \mathbf{x})^2)) + \mathbf{B}_3(\mathbf{A}_2(\bar{\mathbf{N}}_1 \mathbf{x})^2)^2$, and let $\mathcal{J}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ denote the Jacobian of $\mathcal{H}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ with respect to \mathbf{w}_z . Then, $\mathcal{H}_1(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ is 3-homogeneous in \mathbf{w}_z . Thus, eq. (16) can be written as

$$\dot{\mathbf{w}}_z \approx \mathcal{J}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)^\top \bar{\mathbf{y}}, \mathbf{w}_z(0) = \delta \mathbf{z}. \quad (17)$$

Hence, near the saddle point, the evolution of \mathbf{w}_z is governed by the gradient flow that maximizes a homogeneous function. From Lemma 1, we know that in such a scenario \mathbf{w}_z will converge in direction towards a KKT point of an appropriate constrained NCF. Thus, near the saddle point, the weights belonging to \mathbf{w}_z remain small in norm but converge in direction. We will follow a similar approach towards formally establishing directional convergence among the weights with small magnitude near the saddle points.

Although we have assumed $\sigma(x) = x^2$, similar decomposition holds for $\sigma(x) = x^p$, as shown later. In the case of ReLU-type activation functions such as $\sigma(x) = \max(0, x)^p$, an analogous decomposition requires Taylor approximations of $\sigma(x)$. More specifically, in that case,

$$\begin{aligned}\mathcal{H}(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z) &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \sigma \left(\begin{bmatrix} \bar{\mathbf{N}}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) \\ \sigma(\mathbf{A}_1 \mathbf{x}) \end{bmatrix} \right) \\ &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \sigma \left(\begin{bmatrix} \bar{\mathbf{N}}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \\ \mathbf{A}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \end{bmatrix} \right) \\ &= [\bar{\mathbf{N}}_3 \quad \mathbf{B}_3] \left[\begin{bmatrix} \sigma(\bar{\mathbf{N}}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) \\ \sigma(\mathbf{A}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})) \end{bmatrix} \right] \\ &= \bar{\mathbf{N}}_3 \sigma(\bar{\mathbf{N}}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) + \mathbf{B}_3 \sigma(\mathbf{A}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})) \\ &\approx \bar{\mathbf{N}}_3 \sigma(\bar{\mathbf{N}}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x})) + \bar{\mathbf{N}}_3 \sigma'(\bar{\mathbf{N}}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) + \mathbf{B}_3 \sigma(\mathbf{A}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x})) \\ &\quad + \mathbf{B}_3 \sigma'(\mathbf{A}_2 \sigma(\bar{\mathbf{N}}_1 \mathbf{x})) \odot (\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})),\end{aligned}$$

where the final equality follows from Taylor's approximation of $\sigma(\cdot)$ and \mathbf{w}_z being small. As before, $\mathcal{H}(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ decomposes into a term that only depends on \mathbf{w}_n and other terms that are homogeneous in \mathbf{w}_z , with different degrees of homogeneity. It is important to note, however, that applying Taylor approximations requires assuming sufficient smoothness of $\sigma(\cdot)$ —an assumption we will explicitly state in our results.

3.2 Main Results

In this subsection, we present our main results describing the gradient flow dynamics of homogeneous neural networks near saddle points. We begin by stating an assumption on the output of neural network that plays a central role in our analysis.

Assumption 1. Suppose \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$. Then, for all sufficiently small $\delta > 0$, we have

- (i) $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K)$,
- (ii) $\nabla_{\mathbf{w}_z} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \nabla_{\mathbf{w}_z} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^{K-1})$,
- (iii) $\nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \nabla_{\mathbf{w}_n} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K)$,

where $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is L -homogeneous in \mathbf{w}_z for some $L \geq 2$, $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ has locally Lipschitz gradients in both \mathbf{w}_n and \mathbf{w}_z , and $K > L$.

The first condition of the above assumption states that, when $\|\mathbf{w}_z\|_2 = O(\delta)$, the output of the network can be decomposed into a leading term independent of \mathbf{w}_z , an L -homogeneous term in \mathbf{w}_z , and a higher-order remainder term whose magnitude is of $O(\delta^K)$, where $K > L$. The other two conditions require the gradient of the output with respect to \mathbf{w}_n and \mathbf{w}_z to also behave consistently with the first. This assumption is inspired by the behavior of feed-forward neural networks near certain saddle points, as discussed in Section 3.1. Later, we will show that this assumption is indeed satisfied by feed-forward neural networks when \mathbf{w}_z consists of the incoming and outgoing weights of a subset of hidden neurons, and \mathbf{w}_n contains the rest.

We next state a lemma describing how the scale of initialization affects gradient flow trajectories of homogeneous functions. The proof is in Appendix D.1.

Lemma 6. Suppose $g(\mathbf{s})$ is an L -homogeneous function in \mathbf{s} for some $L \geq 2$. Let \mathbf{s}_0 be a non-zero vector and $\mathbf{s}(t)$ be the solution of the following differential equation:

$$\dot{\mathbf{s}} = \nabla_{\mathbf{s}} g(\mathbf{s}), \mathbf{s}(0) = \mathbf{s}_0.$$

For any scalar $\delta > 0$, let $\mathbf{s}_\delta(t)$ be the solution of the following differential equation:

$$\dot{\mathbf{s}} = \nabla_{\mathbf{s}} g(\mathbf{s}), \mathbf{s}(0) = \delta \mathbf{s}_0.$$

Then,

$$\mathbf{s}(t) = \frac{1}{\delta} \mathbf{s}_\delta \left(\frac{t}{\delta^{L-2}} \right). \quad (18)$$

For gradient flow of homogeneous functions, this result implies that scaling the initialization only leads to the scaling of magnitude and time of the gradient flow trajectory. Consequently, the limiting direction of the gradient flow will not be affected, however, the convergence time will be scaled by $1/\delta^{L-2}$. This fact is directly relevant to us because near the saddle point, as shown in eq. (17), \mathbf{w}_z approximately evolves according to gradient flow of a homogeneous function, but its initialization is scaled by δ . Therefore, \mathbf{w}_z will require $O(1/\delta^{L-2})$ time to converge in direction. However, we arrived at eq. (17) under the assumption that $\mathbf{w}_n(t) \approx \bar{\mathbf{w}}_n$. Therefore, we have to ensure that $\mathbf{w}_n(t) \approx \bar{\mathbf{w}}_n$ holds true for $O(1/\delta^{L-2})$ amount of time. To ensure this, we make the following assumption on the saddle point $(\bar{\mathbf{w}}_n, \mathbf{0})$.

Assumption 2. We assume that $(\bar{\mathbf{w}}_n, \mathbf{0})$ is a saddle point of the training loss in eq. (13) such that $\bar{\mathbf{w}}_n$ is a local minimum of

$$\tilde{\mathcal{L}}(\mathbf{w}_n) := \mathcal{L}(\mathbf{w}_n, \mathbf{0}) = \frac{1}{2} \|\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{0}) - \mathbf{y}\|_2^2,$$

and Lojasiewicz's inequality is satisfied in a neighborhood of $\bar{\mathbf{w}}_n$: there exists $\mu_1, \gamma > 0$ and $\alpha \in (0, \frac{L}{2(L-1)})$ such that

$$\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 \geq \mu_1 (\tilde{\mathcal{L}}(\mathbf{w}_n) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^\alpha, \text{ if } \|\mathbf{w}_n - \bar{\mathbf{w}}_n\|_2 \leq \gamma.$$

Here, $\tilde{\mathcal{L}}(\mathbf{w}_n)$ is defined by fixing $\mathbf{w}_z = \mathbf{0}$ in the training loss $\mathcal{L}(\mathbf{w}_n, \mathbf{w}_z)$. Since $(\bar{\mathbf{w}}_n, \mathbf{0})$ is a saddle point of eq. (13), it follows that $\bar{\mathbf{w}}_n$ is a stationary point of $\tilde{\mathcal{L}}(\mathbf{w}_n)$. By further assuming it is a local minimum of $\tilde{\mathcal{L}}(\mathbf{w}_n)$, we can ensure that $\mathbf{w}_n(t)$ remains close to $\bar{\mathbf{w}}_n$ after training begins. However, we require $\mathbf{w}_n(t)$ to remain close to $\bar{\mathbf{w}}_n$ for $O(1/\delta^{L-2})$ amount of time. This is where the Lojasiewicz's inequality plays a key role. We will use the path length bounds obtained via Lojasiewicz's inequality to ensure $\mathbf{w}_n(t)$ remains close to $\bar{\mathbf{w}}_n$ for the required duration.

Regarding the validity of the above assumption, Lojasiewicz's inequality is known to hold near local minima of real-analytic and subanalytic functions for some $\alpha \in (0, 1)$ (Łojasiewicz, 1993; Bolte et al., 2007). This includes feed-forward neural networks with activation functions such as x^p and $\max(x, 0)^p$. But, we require $\alpha \in (0, \frac{L}{2(L-1)})$, which is a stricter condition for all $L > 2$. We provide a simple instance in Appendix D.2 where the local minimum of a feed-forward homogeneous neural network satisfy Lojasiewicz's inequality with $\alpha = \frac{1}{2}$, validating Assumption 2 in those cases. Establishing the validity of Assumption 2 more generally, or even relaxing it, is an important direction for future work.

It is also worth noting that $\alpha = \frac{1}{2}$, which corresponds to the Polyak-Łojasiewicz (PL) inequality (Polyak, 1963; Karimi et al., 2016), is always contained in $(0, \frac{L}{2(L-1)})$. The PL inequality is widely studied and holds near *global minima* of many optimization problems, including those involving feed-forward networks (Liu et al., 2022; Chatterjee, 2022). Perhaps these results could be useful in establishing the validity of Assumption 2.

We now present the main result of this subsection.

Theorem 7. *Suppose Assumption 1 is satisfied, and $(\bar{\mathbf{w}}_n, \mathbf{0})$ is a saddle point of the training loss in eq. (13) such that Assumption 2 is satisfied. Let $(\mathbf{w}_n(t), \mathbf{w}_z(t))$ evolve according to eq. (14) and eq. (15), respectively. Define $\bar{\mathbf{y}} := \mathbf{y} - \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0})$ and $\bar{\mathcal{H}}_1(\mathbf{x}; \mathbf{w}_z) := \mathcal{H}_1(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$. Then, for any arbitrarily small $\epsilon > 0$, there exists T_ϵ such that for all sufficiently small $\delta > 0$ the following holds:*

$$\|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2 = O(\delta^{\beta_1}) \text{ and } \|\mathbf{w}_z(t)\|_2 = O(\delta), \text{ for all } t \in \left[0, \frac{T_\epsilon}{\delta^{L-2}}\right],$$

where $\beta_1 > 0$. Moreover, if $\mathbf{z} \in \mathcal{S}(\mathbf{z}_*; \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1})$, where \mathbf{z}_* is a non-negative KKT point of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$, then

$$\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2 \geq \delta\eta_1 \text{ and } \frac{\mathbf{z}_*^\top \mathbf{w}_z(T_\epsilon/\delta^{L-2})}{\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2} = 1 - O(\epsilon),$$

else, $\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2 = \epsilon \cdot O(\delta)$. Here, η_1 is a positive constant independent of ϵ and δ .

In the above theorem, $\bar{\mathbf{y}}$ denotes the residual error at the saddle point $(\bar{\mathbf{w}}_n, \mathbf{0})$, and $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ is the NCF defined with respect to $\bar{\mathbf{y}}$ and $\bar{\mathcal{H}}_1(\mathbf{x}; \mathbf{w}_z)$. The theorem states that, under Assumption 1 and Assumption 2, the gradient flow initialized near the saddle point $(\bar{\mathbf{w}}_n, \mathbf{0})$ evolves such that, during the initial stages of training, \mathbf{w}_n remains close to $\bar{\mathbf{w}}_n$ and \mathbf{w}_z remains small. Moreover, if the initial direction of \mathbf{w}_z , denoted by \mathbf{z} , lies in a stable set of a non-negative KKT point of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$, then \mathbf{w}_z approximately converge in direction towards that KKT point. Note that, $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ has the following form:

$$\tilde{\mathcal{N}}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1} := \max_{\|\mathbf{w}_z\|_2^2=1} \bar{\mathbf{y}}^\top \bar{\mathcal{H}}_1(\mathbf{X}; \mathbf{w}_z) = \max_{\|\mathbf{w}_z\|_2^2=1} \bar{\mathbf{y}}^\top \mathcal{H}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z) \quad (19)$$

If \mathbf{z} does not lie in a stable set of a non-negative KKT point, then \mathbf{w}_z approximately becomes zero, as $\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2 = \epsilon \cdot O(\delta)$, where ϵ and δ are both small. In contrast, in the previous case, $\|\mathbf{w}_z(T_\epsilon/\delta^{L-2})\|_2 \geq \delta\eta_1$, where η_1 is a positive constant. This essentially happens because the gradient flow of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ converges to the origin in this case.

Our proof technique is similar to the discussion in Section 3.1. We show that, in the initial stages of training, the dynamics of \mathbf{w}_z stated in eq. (15) is close to the gradient flow dynamics of the NCF $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ with initialization $\delta\mathbf{z}$. According to Lemma 1, the latter dynamics would either converge in direction to a KKT point of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ or converge to the origin. This implies \mathbf{w}_z also either converges in direction to the KKT point or goes towards the origin. The detailed proof is in Appendix B.1.

Remark 1. In Theorem 7, the KKT point to which \mathbf{w}_z converges in direction depends on \mathbf{z} , the initial direction of \mathbf{w}_z . For homogeneous feed-forward neural network, as stated in Lemma 5, after escaping from the origin, the gradient flow reaches a saddle point where certain subset of weights are small. In Theorem 7, \mathbf{w}_z corresponds to this subset of small weights. However, while Lemma 5 characterizes the magnitude of these weights, it does not provide any insight into their direction. On the other hand, Theorem 7 suggests that information about the direction of these small weights is also crucial for a better understanding of the dynamics of gradient flow near such saddle points. Determining this directions appears to be difficult and is an interesting direction for future works.

The above theorem crucially relies on Assumption 1. We now show that Assumption 1 is satisfied for feed-forward neural networks with homogeneous activation functions, under the condition that \mathbf{w}_z contains all the incoming and outgoing weights of a subset of hidden neurons, while \mathbf{w}_n contains the remaining weights.

Lemma 8. Let \mathcal{H} be an L -layer feed-forward neural network as described in eq. (3), for some $L \geq 2$, with activation function $\sigma(x) = \max(x, \alpha x)^p$, where $p \in \mathbb{N}$. Let \mathcal{G}_z denote the subset of hidden neurons containing last $k_l - p_l$ neurons of the l th hidden layer, for all $l \in [L-1]$. Let \mathbf{w}_z be the subset of the weights containing all outgoing and incoming weights of hidden neurons in \mathcal{G}_z , and \mathbf{w}_n contains the remaining weights. More specifically, let

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{N}_1 \\ \mathbf{A}_1 \end{bmatrix}, \mathbf{W}_l = \begin{bmatrix} \mathbf{N}_l & \mathbf{B}_l \\ \mathbf{A}_l & \mathbf{C}_l \end{bmatrix} \text{ for } 2 \leq l \leq L-1, \text{ and } \mathbf{W}_L = \begin{bmatrix} \mathbf{N}_L & \mathbf{B}_L \end{bmatrix},$$

where $\mathbf{N}_1 \in \mathbb{R}^{p_1 \times d}$, $\mathbf{A}_1 \in \mathbb{R}^{(k_1 - p_1) \times d}$, $\mathbf{N}_l \in \mathbb{R}^{p_l \times p_{l-1}}$, $\mathbf{C}_2 \in \mathbb{R}^{(k_1 - p_1) \times (k_{l-1} - p_{l-1})}$, $\mathbf{N}_L \in \mathbb{R}^{1 \times p_{L-1}}$, $\mathbf{B}_L \in \mathbb{R}^{1 \times (k_{L-1} - p_{L-1})}$, and \mathbf{B}_l and \mathbf{A}_l are defined in a consistent way. Then \mathbf{w}_n will contain all the weights belonging to $\{\mathbf{N}_l\}_{l=1}^L$, and \mathbf{w}_z contains the remaining weights.

(i) Suppose $\alpha \neq 1$ and $p \geq 4$. Let \mathbf{w}_n be fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, then for all sufficiently small $\delta > 0$, we have

- (a) $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K)$,
- (b) $\nabla_{\mathbf{w}_z} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \nabla_{\mathbf{w}_z} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^{K-1})$,
- (c) $\nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \nabla_{\mathbf{w}_n} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K)$,

where $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z , and $K > p+1$.

(ii) Suppose $\alpha = 1$ and $p \geq 1$. For any \mathbf{w}_n and \mathbf{w}_z , we have

$$\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \sum_{i=1}^m \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z), \quad (20)$$

for some $m \geq 1$, where $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z . For all $i \geq 2$, $\mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is also homogeneous in \mathbf{w}_z with degree of homogeneity strictly greater than $p+1$. Also, for all $i \geq 1$, $\mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a polynomial in \mathbf{w}_z and \mathbf{w}_n .

Finally, in both of the above cases, $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ can be expressed as:

$$\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sum_{l=1}^{L-2} \nabla_{\mathbf{s}} f_{L,l+2}^{\top} (\mathbf{N}_{l+1} \mathbf{g}_l(\mathbf{x})) \mathbf{B}_{l+1} \sigma(\mathbf{A}_l \mathbf{g}_{l-1}(\mathbf{x})) + \mathbf{B}_L \sigma(\mathbf{A}_{L-1} \mathbf{g}_{L-2}(\mathbf{x})), \quad (21)$$

where, for $1 \leq l \leq L-1$,

$$g_0(\mathbf{x}) = \mathbf{x}, g_l(\mathbf{x}) = \sigma(\mathbf{N}_l g_{l-1}(\mathbf{x})), \text{ and } f_{L,l+1}(\mathbf{s}) = \mathbf{N}_L \sigma(\mathbf{N}_{L-1} \sigma(\cdots \sigma(\mathbf{N}_{l+1} \sigma(\mathbf{s})))) .$$

In the above lemma, \mathcal{G}_z contains the last $k_l - p_l$ neurons of each hidden layer, and the incoming and outgoing weights of these neurons belong to \mathbf{w}_z , while the remaining weights belong to \mathbf{w}_n . In the first case, where $\alpha \neq 1$ and $p \geq 4$, the output of the neural network satisfies the three conditions of Assumption 1, when \mathbf{w}_z is small. Here, we had to assume $p \geq 4$ to ensure the Taylor's approximation of $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and its gradient are sufficiently smooth.

In the second case, where $\alpha = 1$ and $p \geq 1$, the output of the neural network can be decomposed into a leading term that is independent of \mathbf{w}_z , along with additional terms that are homogeneous in \mathbf{w}_z with different degrees of homogeneity. The term with lowest degree of homogeneity is denoted by $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Using the decomposition in eq. (20), we now verify that all three conditions of Assumption 1 are satisfied. Suppose $\|\mathbf{w}_z\|_2 = O(\delta)$. Since $\mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is homogeneous in \mathbf{w}_z with degree of homogeneity strictly greater than $p + 1$, for all $i \geq 2$, it follows that

$$\begin{aligned}\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \sum_{i=2}^m \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K),\end{aligned}$$

for some $K > p + 1$. Next, from Lemma 14, $\nabla_{\mathbf{w}_z} \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is homogeneous in \mathbf{w}_z with degree of homogeneity strictly greater than p , for all $i \geq 2$. Hence,

$$\begin{aligned}\nabla_{\mathbf{w}_z} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \nabla_{\mathbf{w}_z} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \sum_{i=2}^m \nabla_{\mathbf{w}_z} \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \nabla_{\mathbf{w}_z} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^{K-1}).\end{aligned}$$

Finally, from Lemma 15, $\nabla_{\mathbf{w}_n} \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ will be homogeneous in \mathbf{w}_z with same degree of homogeneity as $\mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Therefore, $\|\nabla_{\mathbf{w}_n} \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^K)$, for $K > p + 1$ and $i \geq 2$. Hence,

$$\nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \nabla_{\mathbf{w}_n} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \nabla_{\mathbf{w}_n} \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + O(\delta^K).$$

Thus, all three conditions of Assumption 1 are satisfied.

The expression of $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ in both the cases is stated in eq. (21). In the first case, since $p \geq 4$, $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ has locally Lipschitz gradients with respect to both \mathbf{w}_n and \mathbf{w}_z . In the second case, $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a polynomial with respect to \mathbf{w}_n and \mathbf{w}_z , thus, it has locally Lipschitz gradients with respect to both \mathbf{w}_n and \mathbf{w}_z . The proof of the lemma is in Appendix B.2.

Remark 2. *In Lemma 8, the set \mathbf{w}_z contains the incoming and outgoing weights of the last few neurons in each layer. This choice is made purely for notational simplicity and is without loss of generality. In feed-forward neural networks, neurons within a layer can be arbitrarily reordered without affecting the network's output. Therefore, any subset of neurons could be grouped into \mathbf{w}_z via an appropriate re-indexing.*

What happens if $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = 0$? The proof of Theorem 7 relies on showing that, in the initial stages of training, the dynamics of \mathbf{w}_z stated in eq. (15) is close to the gradient flow dynamics of the NCF $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ with initialization $\delta \mathbf{z}$. But, if $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = 0$, for all \mathbf{w}_z , then the gradient flow of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}$ is not meaningful. We explore this situation further in the Appendix D.3. Our experiments suggest that in such cases, the higher-order remainder terms that arise in the decomposition of the output of the neural network start becoming crucial, and they determine the directional convergence among the weights. This situation arises in deep matrix factorization problem. For certain problems, it also seems to arise when a ReLU-type activation function $(\max(x, 0))^p$ is used. We believe this happens because for ReLU-type activation functions, the function and its gradient both are zero for negative values. We do not encounter this behavior if Leaky ReLU-type activation functions $(\max(x, \alpha x))^p$ are used.

3.3 Additional Discussion

In this subsection, we discuss certain properties of the KKT points of $\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_1}$, when \mathcal{H}_1 is of the form as in eq. (21) and \mathbf{p} is a non-zero vector.

Proportionality of weights at KKT point. We first show that at a positive KKT point of $\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_1}$, the norm of incoming and outgoing weights of hidden neurons belonging to \mathcal{G}_z are proportional.

Lemma 9. *Let \mathcal{H} be an L -layer feed-forward neural network as described in eq. (3), for some $L \geq 2$, with activation function $\sigma(x) = \max(x, \alpha x)^p$, where $p \in \mathbb{N}$ and $p \geq 1$. Suppose its weights are partitioned into two sets \mathbf{w}_n and \mathbf{w}_z as done in Lemma 8, and let \mathcal{H}_1 be as defined in eq. (21). Suppose \mathbf{w}_n is fixed, \mathbf{p} is a non-zero vector, and $\bar{\mathbf{w}}_z$ is a positive KKT point of*

$$\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z) := \max_{\|\mathbf{w}_z\|_2^2=1} \mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z).$$

Then, $\bar{\mathbf{C}}_l = \mathbf{0}$, for all $2 \leq l \leq L-1$, and

$$p \|\bar{\mathbf{B}}_{l+1}[:, j]\|_2^2 = \|\bar{\mathbf{A}}_l[j, :] \|_2^2, \text{ for all } l \in [L-1] \text{ and } 1 \leq j \leq k_l - p_l.$$

Consequently,

$$p \|\bar{\mathbf{W}}_{l+1}[:, j]\|_2^2 = \|\bar{\mathbf{W}}_l[j, :] \|_2^2, \text{ for all } l \in [L-1] \text{ and } p_l + 1 \leq j \leq k_l.$$

Recall that \mathbf{w}_z consists of $\{\mathbf{B}_{l+1}, \mathbf{A}_l\}_{l=1}^{L-1}$ and $\{\mathbf{C}_l\}_{l=2}^{L-1}$, and we denote their values at the KKT point $\bar{\mathbf{w}}_z$ by $\{\bar{\mathbf{B}}_{l+1}, \bar{\mathbf{A}}_l\}_{l=1}^{L-1}$ and $\{\bar{\mathbf{C}}_l\}_{l=2}^{L-1}$. The above lemma shows that at any positive KKT point, the norm of $\bar{\mathbf{B}}_{l+1}[:, j]$ is proportional to $\bar{\mathbf{A}}_l[j, :]$, and $\bar{\mathbf{C}}_l = \mathbf{0}$. Since, for all $l \in [L-1]$ and $p_l + 1 \leq j \leq k_l$, $\bar{\mathbf{W}}_{l+1}[:, j]$ is a concatenation of $\bar{\mathbf{B}}_{l+1}[:, j]$ and $\bar{\mathbf{C}}_{l+1}[:, j]$, and $\bar{\mathbf{W}}_l[j, :]$ is a concatenation of $\bar{\mathbf{A}}_l[j, :]$ and $\bar{\mathbf{C}}_l[j, :]$, we get that $p \|\bar{\mathbf{W}}_{l+1}[:, j]\|_2^2 = \|\bar{\mathbf{W}}_l[j, :] \|_2^2$. The proof is in Appendix B.3.

Recall that in Lemma 8, \mathcal{G}_z denotes the set containing the last $k_l - p_l$ neurons of each hidden layer. The incoming and outgoing weights of neurons in \mathcal{G}_z belong to \mathbf{w}_z . Thus, the above lemma establishes that at a positive KKT point, the norm of incoming and outgoing weights of hidden neurons belonging to \mathcal{G}_z are proportional, implying that if the incoming weight is zero, then outgoing weight would also be zero, and vice-versa. We also observe this behavior in our numerical experiments discussed in Section 3.4.

Note that when $\alpha \neq 1$, Lemma 8 holds for $p \geq 4$, whereas the above lemma holds for $p \geq 1$, which includes ReLU activation. However, for ReLU activation, $f_{L,l}(\cdot)$ is not differentiable everywhere, so $\nabla_{\mathbf{s}} f_{L,l}(\cdot)$ can instead be replaced by any element of the Clarke subdifferential of $f_{L,l}(\cdot)$.

Parallel computation in maximizing NCF. We next show that $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}$ can be written as sum of homogeneous functions, each depending on a distinct and disjoint subset of variables. This decomposition has important implications on the KKT points of $\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_1}$, the constrained NCF corresponding to $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}$.

Recall that $\mathbf{B}_l[:, j]$ and $\mathbf{A}_l[j, :]$ denotes the j -th column of \mathbf{B}_l and j -th row of \mathbf{A}_l , respectively. For any $1 \leq l \leq L-1$, we can write:

$$\mathbf{B}_{l+1} \sigma(\mathbf{A}_l \mathbf{g}_{l-1}(\mathbf{x})) = \sum_{j=1}^{\Delta_l} \mathbf{B}_{l+1}[:, j] \sigma(\mathbf{A}_l[j, :] \mathbf{g}_{l-1}(\mathbf{x})),$$

where $\Delta_l = k_l - p_l$ is the number of neurons in layer l belonging to \mathcal{G}_z . Hence, $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$ can be written as

$$\begin{aligned} \mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z) &= \mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \sum_{l=1}^{L-2} \sum_{j=1}^{\Delta_l} \sum_{i=1}^n p_i \nabla_{\mathbf{s}} f_{L,l+2}^\top(\mathbf{N}_{l+1} \mathbf{g}_l(\mathbf{x}_i)) \mathbf{B}_{l+1}[:, j] \sigma(\mathbf{A}_l[j, :] \mathbf{g}_{l-1}(\mathbf{x}_i)) + \sum_{j=1}^{\Delta_{L-1}} \sum_{i=1}^n p_i \mathbf{B}_L[:, j] \sigma(\mathbf{A}_{L-1}[j, :] g_{L-2}(\mathbf{x}_i)) \\ &= \sum_{l=1}^{L-1} \sum_{j=1}^{\Delta_l} \sum_{i=1}^n p_i \mathcal{H}_{1,l}(\mathbf{x}_i; \mathbf{B}_{l+1}[:, j], \mathbf{A}_l[j, :]) \\ &= \sum_{l=1}^{L-1} \sum_{j=1}^{\Delta_l} \mathbf{p}^\top \mathcal{H}_{1,l}(\mathbf{X}; \mathbf{B}_{l+1}[:, j], \mathbf{A}_l[j, :]) = \sum_{l=1}^{L-1} \sum_{j=1}^{\Delta_l} \mathcal{N}_{\mathbf{p}, \mathcal{H}_{1,l}}(\mathbf{B}_{l+1}[:, j], \mathbf{A}_l[j, :]), \end{aligned}$$

where we define

$$\begin{aligned}\mathcal{H}_{1,l}(\mathbf{x}; \mathbf{b}, \mathbf{a}^\top) &= \nabla_{\mathbf{s}} f_{L,l+2}^\top(\mathbf{N}_{l+1} \mathbf{g}_l(\mathbf{x})) \mathbf{b} \sigma(\mathbf{a}^\top \mathbf{g}_{l-1}(\mathbf{x})), \text{ for all } l \in [L-2], \\ \mathcal{H}_{1,L-1}(\mathbf{x}; \mathbf{b}, \mathbf{a}^\top) &= \mathbf{b} \sigma(\mathbf{a}^\top \mathbf{g}_{L-2}(\mathbf{x})).\end{aligned}$$

Note that each term in the above decomposition is $(p+1)$ -homogeneous in \mathbf{w}_z . Moreover, each individual term is associated with exactly one hidden neuron in \mathcal{G}_z : it depends only on that neuron's incoming and outgoing weights. In other words, the contribution of each neuron in \mathcal{G}_z to the NCF is isolated, and the NCF $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$ separates into neuron-specific, homogeneous components that are mutually independent in terms of the variables they involve. It is also worth noting that the functional form of the terms corresponding to neurons in a particular layer is same but they depend on different set of variables. Hence, the above decomposition of the NCF contains $(L-1)$ distinct functions $\{\mathcal{N}_{\mathbf{p}, \mathcal{H}_{1,l}}\}_{l=1}^{L-1}$ and their multiple copies.

This decomposition and the presence of multiple copies means that maximizing $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$ via gradient flow is equivalent to simultaneously maximizing each of these $(L-1)$ functions via gradient flow, with multiple independent initializations. Since each $\mathcal{N}_{\mathbf{p}, \mathcal{H}_{1,l}}$ is homogeneous, Lemma 1 implies that maximizing $\mathcal{N}_{\mathbf{p}, \mathcal{H}_{1,l}}$ via gradient flow will cause the associated variables to diverge in norm to infinity but converge in direction to a KKT point of $\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_{1,l}}$. However, our interest lies in the limiting direction of all the variables together, which corresponds to a KKT point of $\tilde{\mathcal{N}}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$. The following lemma examines this scenario.

Lemma 10. *Consider maximizing $\sum_{i=1}^m \mathcal{G}_i(\mathbf{w}_i)$ via gradient flow, where each $\mathcal{G}_i(\mathbf{w}_i)$ is a two-homogeneous functions in \mathbf{w}_i , for all $i \in [m]$. Let $(\mathbf{w}_1(t), \mathbf{w}_2(t), \dots, \mathbf{w}_m(t))$ be the corresponding gradient flow trajectories:*

$$\dot{\mathbf{w}}_i = \nabla_{\mathbf{w}_i} \left(\sum_{i=1}^m \mathcal{G}_i(\mathbf{w}_i) \right) = \nabla_{\mathbf{w}_i} \mathcal{G}_i(\mathbf{w}_i), \mathbf{w}_i(0) = \mathbf{w}_{i0}, \quad (22)$$

where $(\mathbf{w}_{10}, \mathbf{w}_{20}, \dots, \mathbf{w}_{m0})$ is the initialization. For all $i \in [m]$, let $\mathbf{w}_{i0} \in \mathcal{S}(\mathbf{w}_i^*; \mathcal{G}_i(\mathbf{w}_i))$, that is,

$$\lim_{t \rightarrow \infty} \frac{\mathbf{w}_i(t)}{\|\mathbf{w}_i(t)\|_2} = \mathbf{w}_i^*,$$

where \mathbf{w}_i^* is a second-order positive KKT point of

$$\tilde{\mathcal{G}}_i(\mathbf{w}_i) := \max_{\|\mathbf{w}_i\|_2^2=1} \mathcal{G}_i(\mathbf{w}_i). \quad (23)$$

Then,

$$\lim_{t \rightarrow \infty} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{j=1}^m \|\mathbf{w}_j(t)\|_2^2}} \begin{cases} > 0, & \text{if } \mathcal{G}_i(\mathbf{w}_i^*) = \max_{j \in [m]} \mathcal{G}_j(\mathbf{w}_j^*) \\ = 0, & \text{if } \mathcal{G}_i(\mathbf{w}_i^*) < \max_{j \in [m]} \mathcal{G}_j(\mathbf{w}_j^*) \end{cases}. \quad (24)$$

The above lemma analyzes the behavior of gradient flow when maximizing a sum of two-homogeneous functions, each depending on a different variable \mathbf{w}_i . Note that, we have not assumed \mathcal{G}_i 's to be distinct. This setup leads to decoupled gradient flows for each \mathbf{w}_i , as described in equation eq. (22). Here, the norm of $\mathbf{w}_i(t)$ diverges and its direction converges to a KKT point of $\tilde{\mathcal{G}}_i(\mathbf{w}_i)$, for all $i \in [m]$. However, when we examine the direction of all the variables combined, eq. (24) reveals that in the limit, only those set of variables corresponding to the most dominant KKT points remain nonzero—that is, those KKT points \mathbf{w}_i^* for which $\mathcal{G}_i(\mathbf{w}_i)$ achieves the largest value among all $\{\mathcal{G}_j(\mathbf{w}_j)\}_{j=1}^m$.

The next lemma considers the case where the degree of homogeneity is greater than two.

Lemma 11. *Consider maximizing $\sum_{i=1}^m \mathcal{G}_i(\mathbf{w}_i)$ via gradient flow, where each $\mathcal{G}_i(\mathbf{w}_i)$ is a L -homogeneous functions in \mathbf{w}_i with $L > 2$. Let $(\mathbf{w}_1(t), \mathbf{w}_2(t), \dots, \mathbf{w}_m(t))$ be the corresponding gradient flow trajectories:*

$$\dot{\mathbf{w}}_i = \nabla_{\mathbf{w}_i} \left(\sum_{i=1}^m \mathcal{G}_i(\mathbf{w}_i) \right) = \nabla_{\mathbf{w}_i} \mathcal{G}_i(\mathbf{w}_i), \mathbf{w}_i(0) = \mathbf{w}_{i0}, \quad (25)$$

where $(\mathbf{w}_{10}, \mathbf{w}_{20}, \dots, \mathbf{w}_{m0})$ is the initialization such that $\mathcal{G}_i(\mathbf{w}_{i0}) > 0$. For all $i \in [m]$, let $\mathbf{w}_{i0} \in \mathcal{S}(\mathbf{w}_i^*; \mathcal{G}_i(\mathbf{w}_i))$, that is, there exists a T_i such that

$$\lim_{t \rightarrow T_i} \|\mathbf{w}_i(t)\|_2 = \infty \text{ and } \lim_{t \rightarrow T_i} \frac{\mathbf{w}_i(t)}{\|\mathbf{w}_i(t)\|_2} = \mathbf{w}_i^*,$$

where \mathbf{w}_i^* is a positive KKT point of

$$\tilde{\mathcal{G}}_i(\mathbf{w}_i) := \max_{\|\mathbf{w}_i\|_2^2=1} \mathcal{G}_i(\mathbf{w}_i). \quad (26)$$

Then,

$$T_i \in \left[\frac{1}{L(L-2)\mathcal{G}_i(\mathbf{w}_i^*)}, \frac{1}{L(L-2)\mathcal{G}_i(\mathbf{w}_{i0})} \right], \text{ for all } i \in [m]. \quad (27)$$

Define $T^* := \min_{i \in [m]} T_i$, then

$$\lim_{t \rightarrow T^*} \sum_{i=1}^m \|\mathbf{w}_i(t)\|_2^2 = \infty \text{ and } \lim_{t \rightarrow T^*} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{j=1}^m \|\mathbf{w}_j(t)\|_2^2}} \begin{cases} > 0, & \text{if } T_i = T^* \\ = 0, & \text{if } T_i > T^* \end{cases}. \quad (28)$$

When the degree of homogeneity is greater than two, the gradient flow solution blows up in finite time. Hence, at the limiting direction of all the variables together, the set of variables that blow up first will remain non-zero, while others will become zero. From eq. (27), we observe that the time required for blow up depends on the value of $\mathcal{G}_i(\mathbf{w}_i^*)$ and $\mathcal{G}_i(\mathbf{w}_{i0})$. Specifically, suppose \mathbf{w}_1^* is the most dominant KKT point, and \mathbf{w}_{10} is sufficiently close to \mathbf{w}_1^* . Then, T_1 will be the smallest and $\mathbf{w}_1(t)$ will blow up first. Therefore, if at least one of the initial weights is near the most dominant KKT point, then at the limiting direction of all the variables together, the set of variables corresponding to the most dominant KKT points will remain nonzero while the rest will become zero. The proof of Lemma 10 and Lemma 11 are in Appendix D.4.

Now, consider maximizing $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$ via gradient flow. If it is two-homogeneous, assume the conditions in Lemma 10 are satisfied. If it is L -homogeneous with $L > 2$, assume the conditions in Lemma 11 hold and that at least one of the initial weights is sufficiently near the most dominant KKT point. Then, at the limiting direction of all the variables together, only those set of variables corresponding to the most dominant KKT points will remain nonzero while the rest will become zero. This behavior highlights a key benefit of over-parameterization. Recall that the decomposition of $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$ only contains $(L-1)$ distinct functions, each having multiple copies. If we over-parameterize by adding more neurons in \mathcal{G}_z , we do not introduce new functions in $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$, but rather more copies of those $(L-1)$ functions. This will imply that each of those $(L-1)$ functions are being maximized with more number of independent initializations, which in turn will boost the odds of selecting a better dominant KKT point.

The conclusion that only the set of variables corresponding to the most dominant KKT points will remain nonzero is not always true and depends on conditions such as those stated in Lemma 10 and Lemma 11. We make some remarks about those conditions. In Lemma 10, we assume that each KKT point \mathbf{w}_i^* is a second-order KKT point. This assumption is used to get a bound on $\|\mathbf{w}_i(t)\|$. This condition seems to be mild, since for many problems gradient descent avoids first-order stationary points (Lee et al., 2016; 2019). Next, along with Lemma 11, we require at least one of the initial weights to be sufficiently near the most dominant KKT point. Suppose the objective is a sum of few distinct homogeneous functions and their multiple copies, similar to $\mathcal{N}_{\mathbf{p}, \mathcal{H}_1}(\mathbf{w}_z)$. Increasing the number of copies will imply that the same set of functions are being maximized with more number of independent initializations. This will increase the chances of at least one of the initial weights being sufficiently near the most dominant KKT point. Thus, over-parameterization can help in satisfying this additional condition. In summary, while the selection of variables based on dominant KKT points is not an absolute certainty, it appears to be quite likely as the conditions required for it does not seem to be overly restrictive in over-parameterized settings.

3.4 Numerical Experiments

We next conduct numerical experiments to validate Theorem 7, with results presented in Figure 1 and Figure 2. In both cases, we train a three layer neural network with ten neurons in each layer—one with square activation function ($\sigma(x) = x^2$) and one with ReLU activation function ($\max(x, 0)$). The weights are initialized near a saddle point where the incoming and outgoing weights of the last nine neurons of each layer is zero; these weights form \mathbf{w}_z and the remaining form \mathbf{w}_n , and the last nine neurons of each layer form \mathcal{G}_z .

Square activation. Figure 1a depicts the evolution of the training loss and the distance of the weights from the saddle point. As expected, the loss does not change much, and the weights remain close to the saddle point, which can also be verified by comparing Figure 1b and Figure 1c. This implies that the weights in \mathbf{w}_z remains relatively small in magnitude. Next, a unit norm vector is a KKT point of the constrained NCF if the vector and the gradient of the NCF at that vector are parallel. From Figure 1d, we observe that \mathbf{w}_z approximately converges in direction to a KKT point of the constrained NCF. Figure 1e and Figure 1f provide a closer look at the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 180000, respectively. At initialization, they are small and random, and at iteration 180000 their norm increases but remains small overall, however, they are structured and exhibit sparsity. Except for the incoming and outgoing weight of the second neuron in the first layer, all the other weights belonging to \mathbf{w}_z have relatively small magnitude. Also, in accordance with Lemma 9, if the incoming weights of a hidden neuron belonging to \mathcal{G}_z have small magnitude, then the outgoing weights are also small, and vice-versa.

ReLU activation. Although our theoretical results do not hold for ReLU activation, in Figure 2 we explore ReLU activation. Figure 2a shows that the loss does not change much and the weights remain close to the saddle point. The latter is also evident from Figure 2b and Figure 2c. Thus, \mathbf{w}_z remains small in magnitude. Figure 2d shows that \mathbf{w}_z converges in direction to a KKT point of the constrained NCF. Figure 2e and Figure 2f depicts the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 6000, respectively. At initialization, they are small and random. At iteration 6000, their norm increases but remains small overall. However, as before, they are structured and exhibit sparsity, where the incoming and outgoing weights of several neurons in the first layer have high magnitude. Yet, they are still consistent with Lemma 9—if the incoming weights of a hidden neuron belonging to \mathcal{G}_z are small, then the outgoing weights are small too, and vice-versa. For instance, 5th and 8th row of \mathbf{W}_1 is small and 5th and 8th column of \mathbf{W}_2 is small as well.

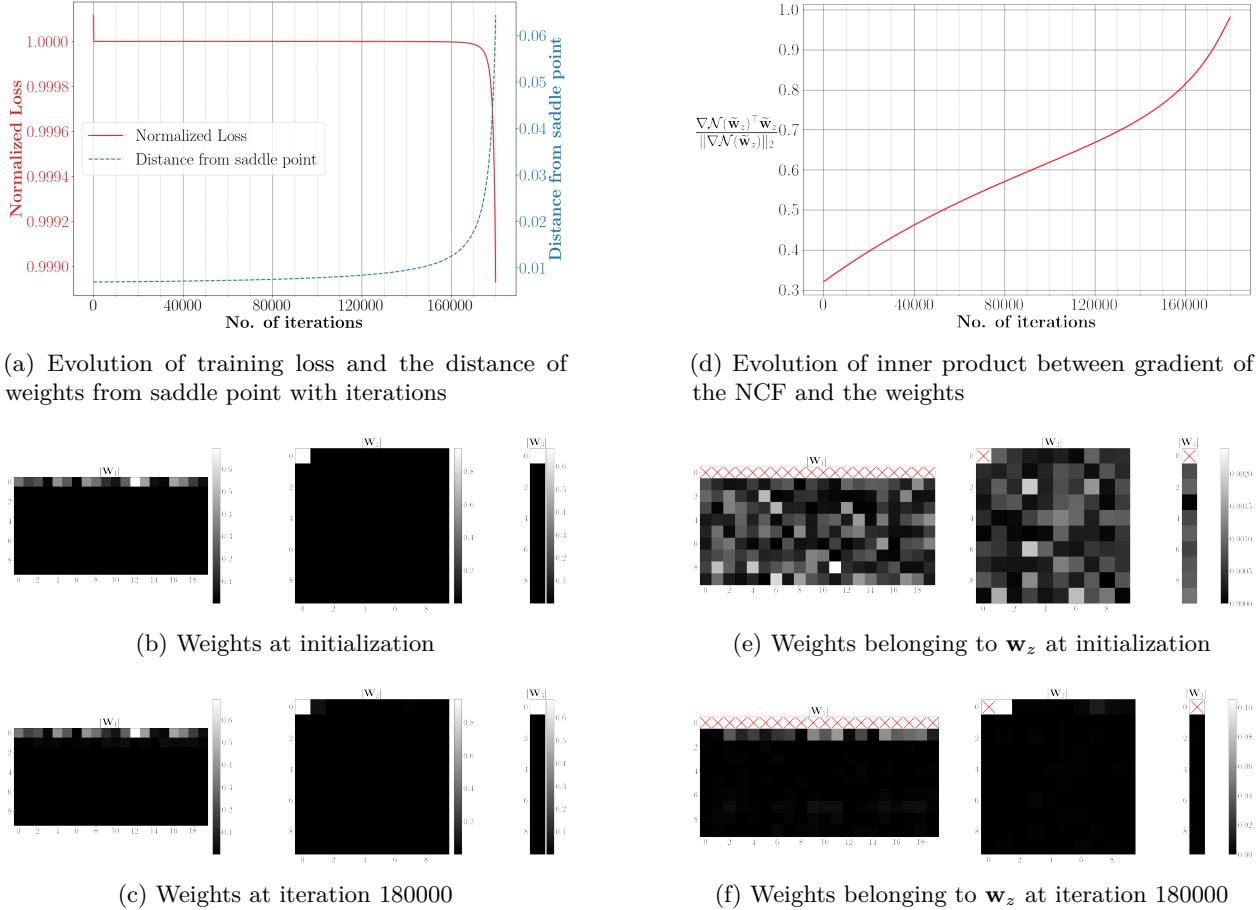
Overall, the empirical behavior is consistent with Theorem 7: in the initial stages, the weights belonging to \mathbf{w}_z remain small in norm but converge in direction to a KKT of the constrained NCF—even in the ReLU case, where the theorem does not strictly apply. Moreover, in accordance with Lemma 9, the norm of incoming and outgoing weights of hidden neurons belonging to \mathcal{G}_z is proportional.

Finally, we want to emphasize on the similarity in the dynamics of gradient flow near the origin and near the saddle point. Near the origin, as discussed in Section 2.1, the weights of a feed-forward neural network remains small in norm but converge in direction, where the norm of incoming and outgoing weights of each hidden neuron is proportional. This in turn leads to emergence of a sparsity structure among the weights, as the incoming and outgoing weights of many neurons become zero, rendering those neurons approximately inactive. Near the saddle point as well, the weights that have small magnitude, which are denoted by \mathbf{w}_z , remain small in norm but converge in direction, where the norm of incoming and outgoing weights of hidden neurons in \mathcal{G}_z is proportional. In this case as well, a sparsity structure emerges among the weights which makes many neurons in \mathcal{G}_z approximately inactive, as illustrated in Figure 1 and Figure 2.

4 Gradient Flow Dynamics Beyond Saddle Points: Empirical Observations

The above section described how the weights evolved while gradient flow remained near the saddle points. In this section, we make important empirical observations about the dynamics of gradient descent after escaping these saddle points. For this, we again consider the experiments illustrated in Figure 1 and Figure 2, and run them for more iterations.

Figure 3 depicts the result of running the experiments described in Figure 1 for more iterations. From Figure 3a we observe that after escaping from the saddle point, the loss rapidly decreases and then eventually



(a) Evolution of training loss and the distance of weights from saddle point with iterations

(d) Evolution of inner product between gradient of the NCF and the weights

(b) Weights at initialization

(e) Weights belonging to \mathbf{w}_z at initialization

(c) Weights at iteration 180000

(f) Weights belonging to \mathbf{w}_z at iteration 180000

Figure 1: (Gradient descent dynamics near saddle point) We train a three-layer neural network using gradient descent whose output is $\mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}))$, where $\sigma(x) = x^2$ (**square activation**), and $\mathbf{W}_3 \in \mathbb{R}^{1 \times 20}$, $\mathbf{W}_2 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_1 \in \mathbb{R}^{10 \times 20}$ are the trainable weights. The weights are initialized near a saddle point where the incoming and outgoing weights of the last nine neurons of each layer is zero. These neurons form \mathcal{G}_z and their incoming and outgoing weights would form \mathbf{w}_z , that is, \mathbf{w}_z contains the last nine rows of \mathbf{W}_1 , the last nine rows and columns of \mathbf{W}_2 and the last nine entries of \mathbf{W}_3 . The remaining weights form \mathbf{w}_n . Panel (a) shows the evolution of the training loss (normalized by the loss at saddle point) and the distance of the weights from the saddle point (normalized by the norm of the weights at saddle point). Panel (b) and (c) depict the weights at initialization and at iteration 180000, respectively. We observe that the training loss does not change much and the weights remain near the saddle point. Also, weights belonging to \mathbf{w}_z remain small in magnitude. Panel (d) shows the evolution of $\nabla \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\tilde{\mathbf{w}}_z)^T \tilde{\mathbf{w}}_z / \|\nabla \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\tilde{\mathbf{w}}_z)\|_2$, where $\tilde{\mathbf{w}}_z := \mathbf{w}_z / \|\mathbf{w}_z\|_2$, which measures how close $\tilde{\mathbf{w}}_z$ is to being a KKT point of the constrained NCF. Panel (d) confirms that \mathbf{w}_z have approximately converged in direction to a KKT point of the constrained NCF. Panel (e) and (f) depicts the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 180000, respectively. More specifically, the weights belonging to \mathbf{w}_n are crossed out and we only plot the weights belonging to \mathbf{w}_z . At initialization, weights belonging to \mathbf{w}_z are small and random. At iteration 180000, they become sparse and structured, while the norm still stays small. In fact, only the incoming and outgoing weights of the second neuron in the first layer have high magnitude, the remaining are relatively much smaller. This behavior is consistent with the result of Lemma 9.

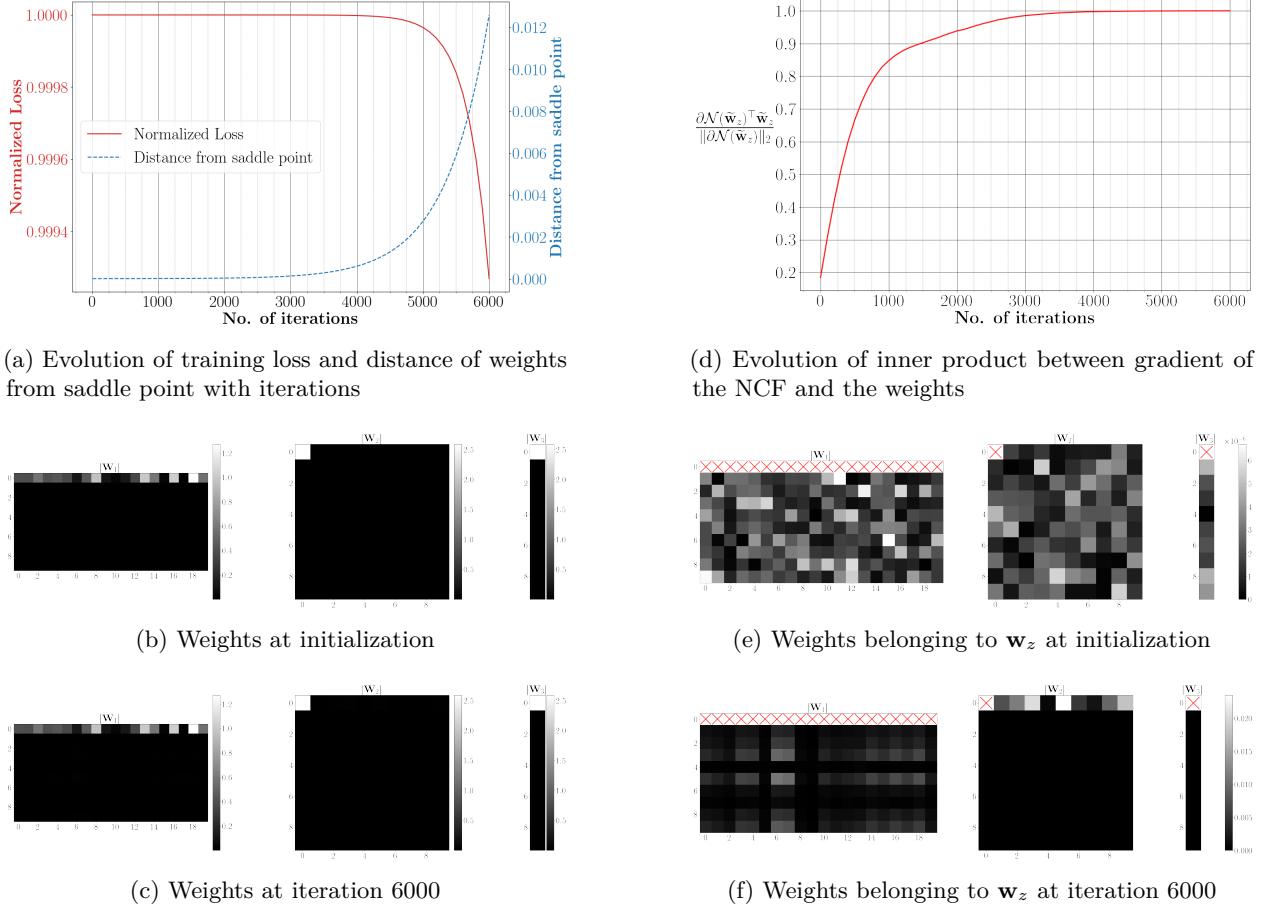


Figure 2: **(Gradient descent dynamics near saddle point)** We train a three-layer neural network using gradient descent whose output is $\mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}))$, where $\sigma(x) = \max(x, 0)$ (**ReLU activation**), and $\mathbf{W}_3 \in \mathbb{R}^{1 \times 20}$, $\mathbf{W}_2 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_1 \in \mathbb{R}^{10 \times 20}$ are the trainable weights. The weights are initialized near a saddle point where the incoming and outgoing weights of the last nine neurons of each layer is zero; these neurons form \mathcal{G}_z and their incoming and outgoing weights would form \mathbf{w}_z , and the remaining weights form \mathbf{w}_n . Panel (a) depicts the evolution of the training loss (normalized by the loss at saddle point) and the distance of the weights from the saddle point (normalized by the norm of the weights at saddle point). Panel (b) and (c) show the weights at initialization and at iteration 6000, respectively. We observe that the training loss does not change much, the weights remain near the saddle point and \mathbf{w}_z remains small. Panel (d) shows the evolution of $\partial \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\tilde{\mathbf{w}}_z)^\top \tilde{\mathbf{w}}_z / \|\partial \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\tilde{\mathbf{w}}_z)\|_2$, where $\tilde{\mathbf{w}}_z := \mathbf{w}_z / \|\mathbf{w}_z\|_2$, which confirms that \mathbf{w}_z has converged in direction to a KKT point of the constrained NCF. Panel (e) and (f) depicts the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 6000, where the weights belonging to \mathbf{w}_n are crossed out. At initialization, weights belonging to \mathbf{w}_z are small and random. At iteration 6000, they become structured and exhibit sparsity, while their norm increases but remains small. Compared to the experiment in Figure 1, the incoming and outgoing weights of multiple neurons in the first layer have high magnitude. However, they are still consistent with Lemma 9, that is, if the incoming weights of a hidden neuron belonging to \mathcal{G}_z are small, then the outgoing weights is small too, and vice-versa.

stagnates, indicating that the weights have reached another saddle point. In Figure 3b, we plot the weights at this new saddle point. Apart from the incoming and outgoing weights of the second neuron in the first layer, all the remaining weights in \mathbf{w}_z remain small. Now, recall from Figure 1 that near the saddle point, a sparsity structure emerges among the weights belonging to \mathbf{w}_z , where only the incoming and outgoing weights of the second neuron in the first layer was non-zero and the remaining weights remained small. This experiment suggests that the sparsity structure, which emerges among \mathbf{w}_z near the saddle point, is preserved even after gradient descent escapes from the saddle point and reaches a new saddle point.

Figure 4 depicts the result of running the experiments described in Figure 2 for more iterations. Overall, we observe a similar behavior to the previous experiment. From Figure 4a, we observe that after escaping from the saddle point, the loss rapidly decreases and then eventually stagnates, indicating that weights have reached a new saddle point. From comparing Figure 2f and Figure 3b, we observe that the sparsity structure, which emerges among \mathbf{w}_z near the saddle point, is preserved even after gradient descent escapes from the saddle point and reaches a new saddle point. For instance, 5th and 8th row of \mathbf{W}_1 were small before gradient descent escapes from the saddle point, and they remain small after gradient descent reaches the new saddle point. The same is true for 5th and 8th column of \mathbf{W}_2 as well. Similarly, all rows of \mathbf{W}_2 , except for the 1st row, were small before gradient descent escapes from the saddle point, and they remain small after gradient descent reaches the new saddle point. The same is true for all entries of \mathbf{W}_3 , except for the 1st entry.

Overall, these experiments suggest a strong similarity between the dynamics of gradient flow after escaping the origin and after escaping from saddle points. As discussed in Section 2.2, gradient flow escapes the origin such that the sparsity structure, which emerged during the early stages of training, is preserved after gradient flow escapes the origin and until it reaches the next saddle point. In other words, the hidden neurons which became inactive during the early stages of training, remain inactive even after gradient flow escapes the origin and until it reaches the next saddle point. Similarly, these experiments suggest that gradient flow escapes the saddle point such that the sparsity structure, which emerged among \mathbf{w}_z while the weights remained near the saddle point, is preserved after gradient flow escapes the saddle point and until it reaches the next saddle point. In other words, the hidden neurons in \mathcal{G}_z which became inactive while the weights were near the saddle point, they remain inactive even after gradient flow escapes the saddle point and until it reaches the next saddle point. Consequently, as the weights escape the saddle point and move towards the next saddle, the training loss is minimized using only the active neurons, including those that became active while the network was near the previous saddle point.

While we are unable to rigorously prove this empirical observation, assuming it holds generally leads to an interesting perspective on the overall training dynamics of neural networks. According to the above experiments, after escaping from a saddle point (say \mathcal{S}_0), gradient descent reaches a new saddle point (say \mathcal{S}_1), where the incoming and outgoing weights of a certain subset of hidden neurons are zero—effectively rendering these neurons inactive. At this new saddle point \mathcal{S}_1 , we can invoke Theorem 7 and Lemma 9. This would imply that near \mathcal{S}_1 , the weights with small magnitude would remain small but converge in direction such that certain neurons are activated while others remain inactive. This will be determined by which KKT point of the constrained NCF does the weights converge to. Then, the weights would escape from \mathcal{S}_1 and reach another saddle point (say \mathcal{S}_2), where the hidden neurons which were inactive near \mathcal{S}_1 will remain inactive until reaching \mathcal{S}_2 . Thus, at the saddle point \mathcal{S}_2 , the incoming and outgoing weights of a certain subset of hidden neurons will have zero norm. Therefore, the entire training dynamics can be viewed as weights moving from one saddle point to another. As they move from one saddle to another, certain new subset of neurons become activated, augmenting the previously active neurons. Moreover, the identity of the newly activated neurons, and their initial weights, is determined by the KKT point of the constrained NCF at the corresponding saddle point. This view of the training dynamics is also consistent with the saddle-to-saddle dynamics hypothesis (Jacot et al., 2021; Li et al., 2021), which argues that gradient descent passes through a sequence of saddle points over the course of training, and the neural network increases its complexity as it moves from one saddle point to another.

Since neurons seem to gradually activate as the training progresses, this observation naturally motivates the design of an algorithm that mimic this behavior by incrementally adding neurons to the network. The goal of the next section is to leverage all these insights to develop a greedy algorithm for training deep neural

networks, wherein neurons are gradually added to the network and then the training loss is minimized using this augmented network.

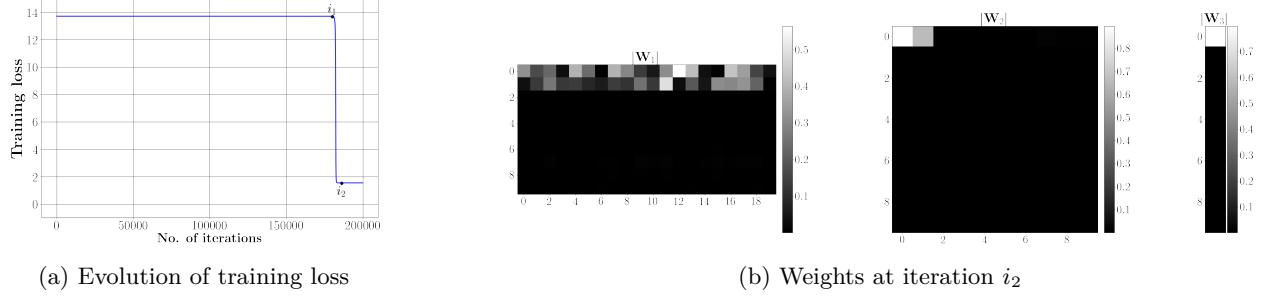


Figure 3: We continue running the experiment illustrated in Figure 1 for more iterations. Panel (a) depicts the evolution of training loss. It shows that after escaping from the saddle point, the loss rapidly decreases before plateauing at a new saddle point. Panel (b) depicts the absolute value of weights at iteration i_2 (marked in Panel (a)), approximately just after reaching the new saddle point. Also $i_1 = 180000$, just before gradient descent escapes from the saddle point. The absolute value of the weights at iteration i_1 is depicted in Figure 1f. Comparing Panel (b) with Figure 1f suggests that the sparsity structure emerging among the weights (specifically \mathbf{w}_z as defined in Figure 1) before escaping the saddle point is preserved until reaching the new saddle point.

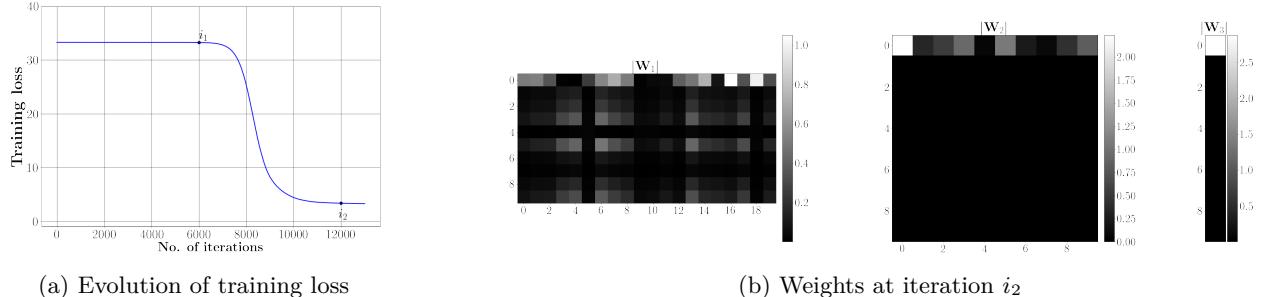


Figure 4: We continue running the experiment illustrated in Figure 2 for more iterations. Panel (a) depicts the evolution of training loss. It shows that after escaping from the saddle point, the loss rapidly decreases before plateauing at a new saddle point. Panel (b) depicts the absolute value of weights at iteration i_2 (marked in Panel (a)), approximately just after reaching the new saddle point. Also $i_1 = 6000$, just before gradient descent escapes from the saddle point. The absolute value of the weights at iteration i_1 is depicted in Figure 2f. Comparing Panel (b) with Figure 2f shows that the sparsity structure emerging among the weights (specifically \mathbf{w}_z as defined in Figure 2) before escaping the saddle point is preserved until reaching the new saddle point.

5 Neuron Pursuit

In this section, we present our algorithm, *Neuron Pursuit* (NP), to train homogeneous feed-forward neural networks. At a high level, the NP algorithm is inspired by the saddle-to-saddle dynamics, and it also leverages insights gained into the emergence of sparsity structure near the saddle points and after escaping them, as discussed above and in previous works. More concretely, it iteratively augments the network by adding neuron(s) selected via the maximization of an appropriate constrained NCF, and then minimizes the training loss via gradient descent using this augmented network. The algorithm is described in Algorithm 1, and we next discuss the steps involved in the algorithm and their motivation.

Algorithm 1 Neuron Pursuit

Require: Training data $\mathcal{D} = \{(\mathbf{X}, \mathbf{y})\} \in \mathbb{R}^{d \times N} \times \mathbb{R}^N$, activation function $\sigma(\cdot)$, small scalar δ , number of iterations E , depth L

- 1: Define $\mathcal{H}(\mathbf{x}; \mathbf{W}_1, \dots, \mathbf{W}_L) = \mathbf{W}_L \sigma(\dots \sigma(\mathbf{W}_1 \mathbf{x}))$, where $\mathbf{W}_1 \in \mathbb{R}^{1 \times d}$ and $\mathbf{W}_l \in \mathbb{R}$, for all $2 \leq l \leq L$, and $\mathcal{L}(\mathbf{W}_1, \dots, \mathbf{W}_L) = \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{W}_1, \dots, \mathbf{W}_L)\|_2^2$.
- 2: Initialize $\mathcal{N}_{\max} \leftarrow -\infty$
- 3: **for** $h = 1$ to H **do** ▷ Maximizing the NCF.
- 4: $(\mathbf{W}_1^*, \dots, \mathbf{W}_L^*) \leftarrow \arg \max \mathbf{y}^\top \mathcal{H}(\mathbf{X}; \mathbf{W}_1, \dots, \mathbf{W}_L)$, s.t. $\sum_{l=1}^L \|\mathbf{W}_l\|_F^2 = 1$.
- 5: $\mathcal{N}^* \leftarrow \mathbf{y}^\top \mathcal{H}(\mathbf{X}; \mathbf{W}_1^*, \dots, \mathbf{W}_L^*)$.
- 6: **if** $\mathcal{N}^* > \mathcal{N}_{\max}$ **then** ▷ Store the KKT point with highest value of the NCF.
- 7: $(\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) \leftarrow (\mathbf{W}_1^*, \dots, \mathbf{W}_L^*)$.
- 8: $\mathcal{N}_{\max} \leftarrow \mathcal{N}^*$
- 9: **end if**
- 10: **end for**
- 11: $(\mathbf{W}_1(0), \dots, \mathbf{W}_L(0)) \leftarrow (\delta \widehat{\mathbf{W}}_1, \dots, \delta \widehat{\mathbf{W}}_L)$ ▷ Small initial weights aligned along the most dominant KKT point.
- 12: **for** $k = 1$ to ∞ **do** ▷ Minimizing training loss using gradient descent.
- 13: $\mathbf{W}_l(k+1) \leftarrow \mathbf{W}_l(k) - \eta \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{W}_1(k), \dots, \mathbf{W}_L(k))$, for all $1 \leq l \leq L$.
- 14: **end for**
- 15: $(\overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L) \leftarrow (\mathbf{W}_1(\infty), \dots, \mathbf{W}_L(\infty))$.
- 16: **for** iteration = 1 to E **do** ▷ Residual error
- 17: $\bar{\mathbf{y}} \leftarrow \mathbf{y} - \mathcal{H}(\mathbf{X}; \overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L)$
- 18: $\mathcal{N}_{\max} = -\infty$
- 19: **for** $h = 1$ to H **do** ▷ Maximizing the NCF.
- 20: **for** $l = 1$ to $L-1$ **do**
- 21: $(\mathbf{a}_l^*, \mathbf{b}_{l+1}^*) \leftarrow \arg \max \bar{\mathbf{y}}^\top \mathcal{H}_{1,l}(\mathbf{X}; \mathbf{b}_{l+1}, \mathbf{a}_l)$, s.t. $\|\mathbf{a}_l\|_2^2 + \|\mathbf{b}_{l+1}\|_2^2 = 1$.
- 22: $\mathcal{N}^* \leftarrow \bar{\mathbf{y}}^\top \mathcal{H}_{1,l}(\mathbf{X}; \mathbf{b}_{l+1}^*, \mathbf{a}_l^*)$.
- 23: **if** $\mathcal{N}^* > \mathcal{N}_{\max}$ **then** ▷ Store the KKT point with highest value of the NCF.
- 24: $(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}) \leftarrow (\mathbf{a}_l^*, \mathbf{b}_{l+1}^*)$.
- 25: $l_* \leftarrow l$, $\mathcal{N}_{\max} \leftarrow \mathcal{N}^*$.
- 26: **end if**
- 27: **end for**
- 28: **end for**
- 29: $\widehat{\mathbf{W}}_{l_*} \leftarrow \begin{bmatrix} \overline{\mathbf{W}}_{l_*} \\ \delta \widehat{\mathbf{a}}^\top \end{bmatrix}$ ▷ The added weights are small and aligned along the most dominant KKT point.
- 30: $\widehat{\mathbf{W}}_{l_*+1} \leftarrow \begin{bmatrix} \overline{\mathbf{W}}_{l_*+1} & \delta \widehat{\mathbf{b}} \end{bmatrix}$
- 31: $\widehat{\mathbf{W}}_l \leftarrow \overline{\mathbf{W}}_l$, for all $l \notin \{l_*, l_*+1\}$
- 32: $(\mathbf{W}_1(0), \dots, \mathbf{W}_L(0)) \leftarrow (\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)$
- 33: **for** $k = 1$ to ∞ **do** ▷ Minimizing training loss using gradient descent.
- 34: $\mathbf{W}_l(k+1) \leftarrow \mathbf{W}_l(k) - \eta \nabla_{\mathbf{W}_l} \mathcal{L}(\mathbf{W}_1(k), \dots, \mathbf{W}_L(k))$, for all $1 \leq l \leq L$.
- 35: **end for**
- 36: $(\overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L) \leftarrow (\mathbf{W}_1(\infty), \dots, \mathbf{W}_L(\infty))$.
- 37: **end for**

The input to the algorithm is the training data, the activation function $\sigma(x)$, which is assumed to be of the form $\max(x, \alpha x)^p$, the depth of the network L and the number of iterations E . It also requires a scalar δ , which is assumed to be small.

The algorithm begins by initializing the neural network \mathcal{H} with depth L and one neuron in every layer. Then, inspired from the work of Kumar & Haupt (2024; 2025b;a), also discussed in Section 2, we minimize the training loss with initial weights small in magnitude and aligned along a KKT point of the constrained NCF. More specifically, in line 3 – 10, the constrained NCF defined with respect to \mathcal{H} and \mathbf{y} is maximized. Although

not specified, we use projected gradient ascent with H different random initializations. The most dominant KKT point, the one which leads to the largest value of the NCF, is stored. In line 12 – 14, the training loss is minimized using gradient descent, where the initial weights have small norm and aligned along the most dominant KKT point. The limiting point of this minimization procedure is stored in $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \dots, \bar{\mathbf{W}}_L)$, which will be a stationary point of the training loss.

Next, $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \dots, \bar{\mathbf{W}}_L)$ can be viewed as a saddle point of the training loss, where only one neuron is active in every layer. In this view, inspired by our discussion in Section 3 and Section 4, we “escape” this saddle point by “activating” certain neurons. More specifically, in line 19 – 28, the constrained NCF with respect to the residual error $\bar{\mathbf{y}}$ and $\mathcal{H}_{1,l}$ is maximized (using projected gradient ascent with H different random initializations), where

$$\begin{aligned}\mathcal{H}_{1,l}(\mathbf{x}; \mathbf{b}_{l+1}, \mathbf{a}_l) &= \nabla_{\mathbf{s}} f_{L,l+2}^{\top} (\bar{\mathbf{W}}_{l+1} \mathbf{g}_l(\mathbf{x})) \mathbf{b}_{l+1} \sigma(\mathbf{a}_l^{\top} \mathbf{g}_{l-1}(\mathbf{x})), \text{ for all } 1 \leq l \leq L-2, \\ \mathcal{H}_{1,L-1}(\mathbf{x}; \mathbf{b}_L, \mathbf{a}_{L-1}) &= \mathbf{b}_L \sigma(\mathbf{a}_{L-1}^{\top} \mathbf{g}_{L-2}(\mathbf{x})),\end{aligned}$$

and, for $1 \leq l \leq L-1$,

$$g_0(\mathbf{x}) = \mathbf{x}, g_l(\mathbf{x}) = \sigma(\bar{\mathbf{W}}_l g_{l-1}(\mathbf{x})), \text{ and } f_{L,l+1}(\mathbf{s}) = \bar{\mathbf{W}}_L \sigma(\bar{\mathbf{W}}_{L-1} \sigma(\dots \sigma(\bar{\mathbf{W}}_{l+1} \sigma(\mathbf{s})))) .$$

The above choice of $\mathcal{H}_{1,l}$ is same as stated in Section 3.3, where only the last neuron in each layer is assumed to be in \mathcal{G}_z , and \mathbf{a}_l ’s and \mathbf{b}_{l+1} ’s can be viewed as the incoming and outgoing weights of those neurons, respectively. Also, maximizing with H different random initializations is inspired from our discussion at the end of Section 3.3: maximizing the constrained NCF is equivalent to maximizing its homogeneous components in parallel with multiple independent initializations, and the weights corresponding to the most dominant KKT point remains non-zero, while others become zero.

In line 29 – 30, the weights corresponding to the most dominant KKT point are added to the network, where the magnitude of the added weights are small. Note that, this procedure is equivalent to adding a neuron with incoming and outgoing weights aligned along the KKT point. Then, in line 33 – 36, the training loss is minimized using gradient descent and the limiting point is stored in $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \dots, \bar{\mathbf{W}}_L)$, which will be a stationary point of the training loss. Now, $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \dots, \bar{\mathbf{W}}_L)$ can again be viewed as a saddle point of the training loss such that certain neurons are active in every layer. Hence, we again “escape” this saddle point by “activating” certain neurons in a similar way. This process is continued for E number of iterations.

Remark 3. *The NP algorithm and the analysis in Section 3 are presented under the assumption that the output of the neural network is a scalar. However, both naturally extend to the vector-valued output setting. For example, in the NP algorithm, the label vector \mathbf{y} can be replaced with a label matrix \mathbf{Y} , where each column corresponds to a multi-dimensional label. Similarly, the NCF can be defined as the matrix inner product between the network’s output and the label matrix. Analogous adjustments can be applied throughout the algorithm and the analysis to accommodate vector-valued outputs.*

Example. For a better understanding of the NP algorithm, we now illustrate its training mechanism with a concrete example. Specifically, we aim to learn the target function $f(\mathbf{x}) = \sigma(\sigma(2x_1 + x_3) - \sigma(x_3 - x_2))$, defined over the binary hypercube $\{\pm 1\}^{20}$, where $\sigma(x) = \max(x, 0)$ is the ReLU activation and x_i denotes the i th coordinate of $\mathbf{x} \in \mathbb{R}^{20}$. To this end, we use a three-layer neural network with ReLU activation function, and train it on a dataset of 500 samples drawn uniformly at random from the hypercube. The NP algorithm is run for two iterations, successfully recovering the target function $f(\mathbf{x})$. We depict the entire training process in Figure 5.

At **Iteration 1**, we begin with a single neuron in every layer. We then maximize the constrained NCF, as described in line 4, using 5 different initializations. The most dominant KKT point, denoted by $(\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_3, \widehat{\mathbf{W}}_3)$, is depicted in Figure 5a(ii). In $\widehat{\mathbf{W}}_1$, the first three entries have larger magnitude than the others, which are small but not negligible. We then minimize the training loss using gradient descent, where the initial weights of the network are small in magnitude and aligned along the most dominant KKT point. The resulting weights are shown in Figure 5a(iii). Notably, the first three entries remain dominant, while the remaining weights shrink further. The function learned at the end of this iteration is plotted in Figure 5a(iv), alongside the ground truth. To do that, we plot the value of the functions at the each vertex of the binary hypercube,

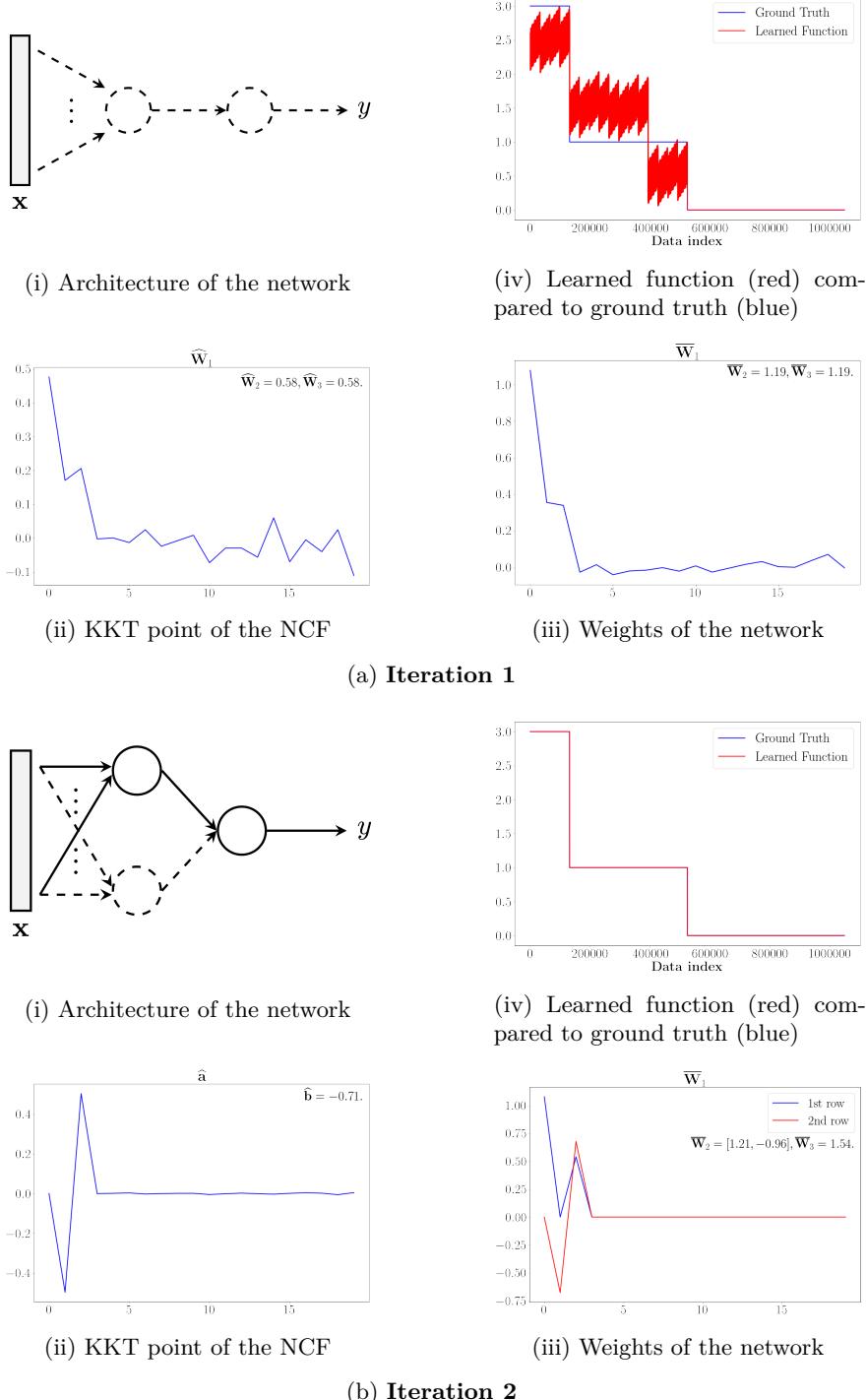


Figure 5: We train a three-layer neural network with ReLU activation $\sigma(x) = \max(x, 0)$ using the NP algorithm to learn the function $f(\mathbf{x}) = \sigma(\sigma(2x_1 + x_3) - \sigma(x_3 - x_2))$ over the binary hypercube $\{\pm 1\}^{20}$, where x_i denotes the i th coordinate of $\mathbf{x} \in \mathbb{R}^{20}$. The learning proceeds iteratively, with the figure illustrating the results over two iterations. For each iteration, we depict: (i) the network architecture during that iteration after adding appropriate neuron(s), (ii) the incoming and outgoing weights of the newly added neuron obtained by maximizing the NCF, (iii) the full network weights after minimizing the training loss, and (iv) the learned function at the end of that iteration compared to the ground truth $f(\mathbf{x})$. Dashed circles and arrows indicate neurons and weights introduced in the current iteration, while solid elements represent those from previous iterations. As shown, the algorithm successfully reconstructs the ground truth function by the end of the second iteration.

ordered lexicographically from $(1, 1, \dots, 1, 1), (1, 1, \dots, 1, -1)$ to $(-1, -1, \dots, -1, -1)$. So, inflection points in the ground truth occur only if x_1 , x_2 , or x_3 changes. The oscillations in the learned function are due to the small but non-zero weights on the coordinates other than the first three.

At **Iteration 2**, we begin by maximizing the constrained NCF with respect to the residual error $\bar{\mathbf{y}}$ and $\mathcal{H}_{1,l}$, as described in line 19 – 28, using 5 different initializations to determine where to add a neuron. The most dominant KKT point $(\hat{\mathbf{a}}, \hat{\mathbf{b}})$ is depicted in Figure 5b(ii), and it corresponds to adding a neuron in the first layer. Accordingly, a neuron is added in the first layer, and its incoming and outgoing weights are small in magnitude but aligned along $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$, respectively, while the remaining weights are same as they were at the end of iteration 1. We next minimize the training loss using gradient descent, and the resulting weights are depicted in Figure 5b(iii). In the first layer, the weights corresponding to coordinates outside x_1, x_2, x_3 shrink to zero, indicating that the network has correctly identified the active components in $f(\mathbf{x})$. Finally, in Figure 5b(iv), we observe that the learned function perfectly matches the ground truth, demonstrating that the NP algorithm has successfully recovered $f(\mathbf{x})$.

5.1 The Descent Property

The NP algorithm attempts to iteratively minimize the training loss. A natural question, then, is whether it can actually reach a local or even a global minimum. Before addressing this, a more fundamental question is does the training loss decrease after each iteration. This is also known as the descent property of an algorithm, and is often the first step towards establishing convergence guarantees. In what follows, we discuss scenarios under which the descent property will be satisfied.

Each iteration of the NP algorithm consists of two stages: neurons are added to the network with small weights aligned along the most dominant KKT point, and then the training loss is minimized using this augmented network via gradient descent. We first study the impact of adding neurons on the training loss, beginning with the first iteration of the NP algorithm.

Lemma 12. *At the end of line 10 of the NP algorithm, suppose $\mathcal{N}_{max} = \mathbf{y}^\top \mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) > 0$; that is, the most dominant KKT point obtained in the first iteration is positive. Then, for all sufficiently small $\delta > 0$,*

$$\mathcal{L}(\mathbf{0}, \dots, \mathbf{0}) > \mathcal{L}(\delta \widehat{\mathbf{W}}_1, \dots, \delta \widehat{\mathbf{W}}_L).$$

Proof. Define $\zeta := \mathbf{y}^\top \mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) > 0$, then, for all sufficiently small $\delta > 0$, we have

$$\begin{aligned} \mathcal{L}(\delta \widehat{\mathbf{W}}_1, \dots, \delta \widehat{\mathbf{W}}_L) &= \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \delta \widehat{\mathbf{W}}_1, \dots, \delta \widehat{\mathbf{W}}_L)\|_2^2 \\ &= \|\mathbf{y} - \delta^L \mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)\|_2^2 \\ &= \|\mathbf{y}\|_2^2 - 2\delta^L \zeta + \delta^{2L} \|\mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)\|_2^2 \\ &\leq \|\mathbf{y}\|_2^2 - \delta^L \zeta < \|\mathbf{y}\|_2^2 = \mathcal{L}(\mathbf{0}, \dots, \mathbf{0}), \end{aligned}$$

the second equality follows from homogeneity of the neural network. The first inequality is true if $\delta^L < \zeta / \|\mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)\|_2^2$. \square

The above lemma shows that if the most dominant KKT point is positive, then initializing the network weights along the most dominant KKT point with sufficiently small norm leads to smaller training loss than at the origin. Importantly, no smoothness assumptions on the activation function are required. The result applies to $\sigma(x) = \max(x, \alpha x)^p$, where $p \in \mathbb{N}$ and $p \geq 1$.

We now turn to addition of neurons in later iterations.

Lemma 13. *At the end of line 28 of the NP algorithm, suppose $\mathcal{N}_{max} = \bar{\mathbf{y}}^\top \mathcal{H}_{1l_*}(\mathbf{X}; \hat{\mathbf{a}}, \hat{\mathbf{b}}) > 0$; that is, the most dominant KKT point is positive. Suppose $\sigma(x) = \max(x, \alpha x)^p$, where if $\alpha = 1$, then $p \geq 1$, otherwise, $p \geq 4$. Then, for all sufficiently small $\delta > 0$,*

$$\mathcal{L}(\bar{\mathbf{W}}_1, \dots, \bar{\mathbf{W}}_L) > \mathcal{L}(\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L),$$

where $(\bar{\mathbf{W}}_1, \dots, \bar{\mathbf{W}}_L)$ is obtained by adding $(\hat{\mathbf{a}}, \hat{\mathbf{b}})$ to $(\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)$ as in line 29-31.

Proof. Since

$$\widehat{\mathbf{W}}_{l_*} = \begin{bmatrix} \overline{\mathbf{W}}_{l_*} \\ \delta \widehat{\mathbf{a}}^\top \end{bmatrix}, \widehat{\mathbf{W}}_{l_*+1} = \begin{bmatrix} \overline{\mathbf{W}}_{l_*+1} & \delta \widehat{\mathbf{b}} \end{bmatrix}, \widehat{\mathbf{W}}_l = \overline{\mathbf{W}}_l \text{ for all } l \notin \{l_*, l_* + 1\},$$

using Lemma 8, for all sufficiently small $\delta > 0$, we get

$$\begin{aligned} \mathcal{H}(\mathbf{x}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) &= \mathcal{H}(\mathbf{x}; \overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L) + \mathcal{H}_{1,l_*}(\mathbf{x}; \delta \widehat{\mathbf{b}}, \delta \widehat{\mathbf{a}}) + O(\delta^K) \\ &= \mathcal{H}(\mathbf{x}; \overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L) + \delta^{p+1} \mathcal{H}_{1,l_*}(\mathbf{x}; \widehat{\mathbf{b}}, \widehat{\mathbf{a}}) + O(\delta^K), \end{aligned}$$

where $K > p+1$. The second equality follows from $(p+1)$ -homogeneity of \mathcal{H}_{1,l_*} . Define $\zeta := \overline{\mathbf{y}}^\top \mathcal{H}_{1,l_*}(\mathbf{X}; \widehat{\mathbf{b}}, \widehat{\mathbf{a}}) > 0$. Using the above equality, we get

$$\begin{aligned} \mathcal{L}(\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) &= \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L)\|_2^2 \\ &= \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L) - \delta^{p+1} \mathcal{H}_{1,l_*}(\mathbf{x}; \widehat{\mathbf{b}}, \widehat{\mathbf{a}}) - O(\delta^K)\|_2^2 \\ &= \|\overline{\mathbf{y}} - \delta^{p+1} \mathcal{H}_{1,l_*}(\mathbf{x}; \widehat{\mathbf{b}}, \widehat{\mathbf{a}}) - O(\delta^K)\|_2^2 \\ &= \|\overline{\mathbf{y}}\|_2^2 - 2\delta^{p+1}\zeta + \delta^{2(p+1)} \|\mathcal{H}_{1,l_*}(\mathbf{x}; \widehat{\mathbf{b}}, \widehat{\mathbf{a}})\|_2^2 + O(\delta^K). \end{aligned}$$

Since $K > p+1$ and $\zeta > 0$, for all sufficiently small $\delta > 0$, we get

$$\mathcal{L}(\widehat{\mathbf{W}}_1, \dots, \widehat{\mathbf{W}}_L) \leq \|\overline{\mathbf{y}}\|_2^2 - \delta^{p+1}\zeta < \|\overline{\mathbf{y}}\|_2^2 = \mathcal{L}(\overline{\mathbf{W}}_1, \dots, \overline{\mathbf{W}}_L),$$

which completes the proof \square

Here as well, if the most dominant KKT point is positive, then adding neurons with sufficiently small incoming and outgoing weights and aligned with the most dominant KKT point leads to a strict decrease in training loss. The extra condition $p \geq 4$ when $\alpha \neq 1$ is required to use Lemma 8.

The above discussion implies that if the most dominant KKT point is positive, then the training loss decreases in the first stage. Next, consider the second stage of each iteration, where the training loss is minimized via gradient descent. It is well known that for sufficiently small step sizes, gradient descent reduces the loss when the objective has a Lipschitz-continuous gradient. However, loss functions involving neural networks do not have globally Lipschitz gradient, and analyzing gradient descent in this setting remains an active area of research. Nevertheless, if the gradient descent updates do in fact reduce the training loss, then combining this with the above discussion of the first stage above shows that the NP algorithm satisfies the descent property: the training loss decreases after each iteration.

5.2 Numerical Experiments

We conduct numerical experiments to evaluate the Neuron Pursuit (NP) algorithm and compare its performance with neural networks trained by gradient descent (GD) with small initialization.

5.2.1 Non-linear Sparse Functions

We attempt to learn non-linear sparse functions, functions which can be represented using a finite number of neurons, over different input distributions. We train deep neural networks using the NP algorithm and gradient descent (GD) with small initialization. For GD, we use 50 neurons per layer and train up to a maximum of 3×10^6 iterations. For the NP algorithm, number of iterations is capped at 31. The performance is evaluated in terms of relative training and test error:

$$\bullet \text{ Training error} := \frac{\|f(\mathbf{X}_{\text{train}}) - \hat{f}(\mathbf{X}_{\text{train}})\|_2}{\|f(\mathbf{X}_{\text{train}})\|_2}, \quad \bullet \text{ Test error} := \frac{\|f(\mathbf{X}_{\text{test}}) - \hat{f}(\mathbf{X}_{\text{test}})\|_2}{\|f(\mathbf{X}_{\text{test}})\|_2},$$

where $\mathbf{X}_{\text{train}}$ and \mathbf{X}_{test} denotes the training and test data, respectively, and $f(\cdot)$ denotes the ground truth function and $\hat{f}(\cdot)$ denotes the learned function. These normalized errors ensure fair comparison across varying training sample sizes. All algorithms are run until the training loss is less than 0.001 or maximum number

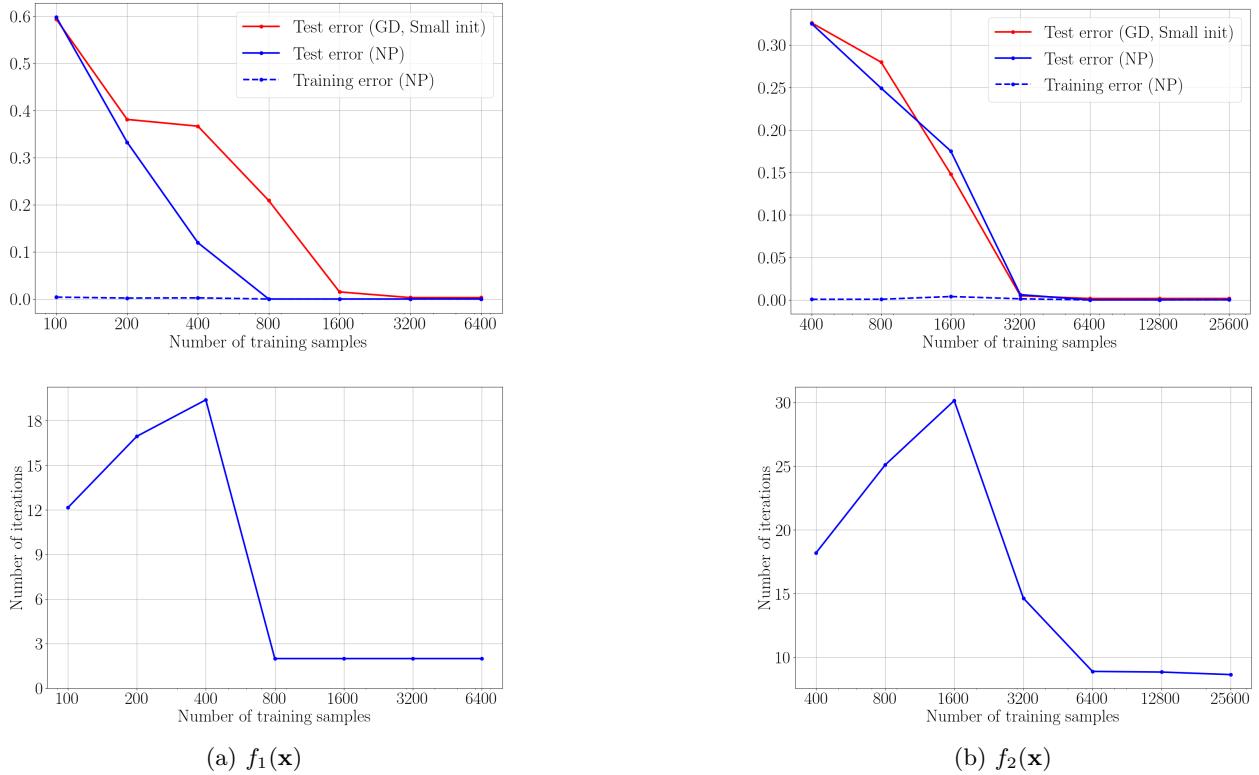


Figure 6: We train three-layer neural networks with activation function $\sigma(x) = \max(x, 0)$ to learn $f_1(\mathbf{x})$ (left) and $f_2(\mathbf{x})$ (right). The network is trained using the NP algorithm for a maximum of 31 iterations, and using gradient descent (GD) with 50 neurons per layer for up to 3×10^6 iterations. For GD, the initialization is small (the weights are sampled uniformly at random from hypersphere of radius 0.01). The top row shows the training and test errors of various algorithms as the number of training samples increases. The bottom row shows the number of iterations required by the NP algorithm to successfully fit the training data. As expected, performance improves for both methods with more training samples. For $f_1(\mathbf{x})$, the NP algorithm achieves small test error with fewer samples. For $f_2(\mathbf{x})$, both algorithms require similar number of samples to achieve small test error.

of iterations are exceeded. The number of training samples varies, while the test set size is fixed at 10^5 . The specific hyperparameters used during training for each algorithm and input distribution is stated in Appendix C.1. The results reported below are averaged over 20 independent runs.

Hypersphere. We train three-layer neural networks with activation function $\sigma(x) = \max(0, x)$ to learn two target functions:

- (i) $f_1(\mathbf{x}) = \sigma(\sigma(2x_1 + x_2) - \sigma(x_3 - x_4))$, and
- (ii) $f_2(\mathbf{x}) = \sigma(\sigma(2x_1 + x_2) - \sigma(x_3 - x_4)) + 5 \sum_{i=1}^2 i \sigma(\sigma(2x_{4i+1} + x_{4i+2}) - \sigma(x_{4i+3} - x_{4i+4}))$,

where x_i denotes the i th coordinate of $\mathbf{x} \in \mathbb{R}^{50}$. The training and test samples are drawn uniformly at random from $\sqrt{d}\mathbb{S}^{d-1}$, where $d = 50$, that is the hypersphere of radius $\sqrt{50}$. Here, $f_1(\mathbf{x})$ is a three-layer neural network with two neurons in the first layer and one neuron in the second layer, and $f_2(\mathbf{x})$ contains six neurons in the first layer and three neurons in the second layer. The above choice of functions is inspired from the notion of *hierarchical* functions studied in Poggio et al. (2017); Dandi et al. (2025).

Figure 6 depicts the test and training errors (top row), and the number of iterations required by the NP algorithm for convergence (bottom row). For $f_1(\mathbf{x})$, the NP algorithm achieves near-zero training error across

all sample sizes. As the number of training samples increases, the test error decreases, and for sufficiently large samples sizes, it becomes nearly zero—indicating that the NP algorithm successfully learns the target function $f_1(\mathbf{x})$. The number of iterations required by the NP algorithm for convergence displays a non-monotonic behavior: it increases as sample size grows, reaches a peak just before the test error transitions from large to small, and then decreases and remains stable. This suggests that in the intermediate sample regime —where test error transitions from large to small —the NP algorithm finds it comparatively harder to fit the training data than low-sample or high-sample regimes.

The overall behavior for $f_2(\mathbf{x})$ is qualitatively similar: the NP algorithm achieves near-zero test error when the number of training samples is large, and the number of iterations peaks just before test error becomes small. However, learning $f_2(\mathbf{x})$ requires significantly more training samples than $f_1(\mathbf{x})$. This is expected, as $f_2(\mathbf{x})$ is built using more neurons and thus more complex.

The figure also depicts the test errors of neural networks trained using gradient descent with small initialization. The training error is small for all number of training samples, and it is not plotted to avoid clutter. The test error decreases upon increasing the number of training samples. For $f_1(\mathbf{x})$, GD performs worse than the NP algorithm, requiring more training samples to achieve small test error. For $f_2(\mathbf{x})$, both algorithms require similar number of training samples to achieve small test error.

Hypercube. We next train three-layer neural networks with activation $\sigma(x) = \max(0.5x, x)$ ¹ to learn two target functions:

$$(i) \ g_1(\mathbf{x}) = x_1x_2 - x_1x_2x_3x_4, \text{ and } (ii) \ g_2(\mathbf{x}) = x_1x_2x_3x_4,$$

where the training and test samples are drawn uniformly at random from the binary hypercube $\{\pm 1\}^{50}$. The learning dynamics of neural networks for such functions has been extensively studied in recent works (Abbe et al., 2023; 2022; Suzuki et al., 2023).

Figure 7 depicts the test and training errors (top row), and the number of iterations required by the NP algorithm to fit the training data (bottom row). For sufficiently large number of samples, the NP algorithm achieves near-zero test error for both $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$. Also, the number of iterations required by the NP algorithm for convergence increases in the beginning and then decreases and stabilizes. Notably, for $g_2(\mathbf{x})$ with 1600 training samples, the NP algorithm fails to achieve small training error in many instances within 31 iterations, which is sufficient number of iterations when learning with more or less samples. This again demonstrates that fitting the training data using the NP algorithm is comparatively harder in the intermediate sample regime than low-sample or high-sample regimes.

Compared to gradient descent, the NP algorithm performs similarly on $g_1(\mathbf{x})$, with both algorithms achieving small test error for similar number of samples. For $g_2(\mathbf{x})$, the NP algorithm requires fewer samples than gradient descent to achieve small test error. It is also worth noting that learning $g_1(\mathbf{x})$ requires fewer samples than $g_2(\mathbf{x})$, for both the algorithms, where, $g_1(\mathbf{x})$ is a sum of second- and fourth-order polynomial and $g_2(\mathbf{x})$ is a fourth-order polynomial. For neural networks trained via gradient descent, this behavior has been attributed to the higher *leap exponent* of $g_2(\mathbf{x})$ (Abbe et al., 2022). It seems neural networks trained via the NP algorithm also exhibit similar behavior.

Gaussian. We train four-layer neural networks with activation function $\sigma(x) = \max(0.5x, x)$ to learn two target functions:

$$(i) \ h_1(\mathbf{x}) = \max(2x_1, x_2), \text{ and } (ii) \ h_2(\mathbf{x}) = \max(x_1, 2x_2) + \max(x_3, 2x_4) - \max(x_1 + x_3, -x_2, -x_4),$$

where the training and test samples are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{50})$. Note that, since maximum of linear functions is piecewise linear, the above functions can be represented using finitely many neurons.

Figure 8 depicts the test and training errors (top row), and the number of iterations required by the NP algorithm to fit the training data (bottom row). For large number of samples, the NP algorithm achieves

¹For ReLU activation, the constrained NCF becomes zero at a certain iteration, when learning $g_1(\mathbf{x})$. As discussed at the end of Section 3.2, for such cases our theoretical results and the NP algorithm are not applicable. We do not face this issue with Leaky ReLU activation.

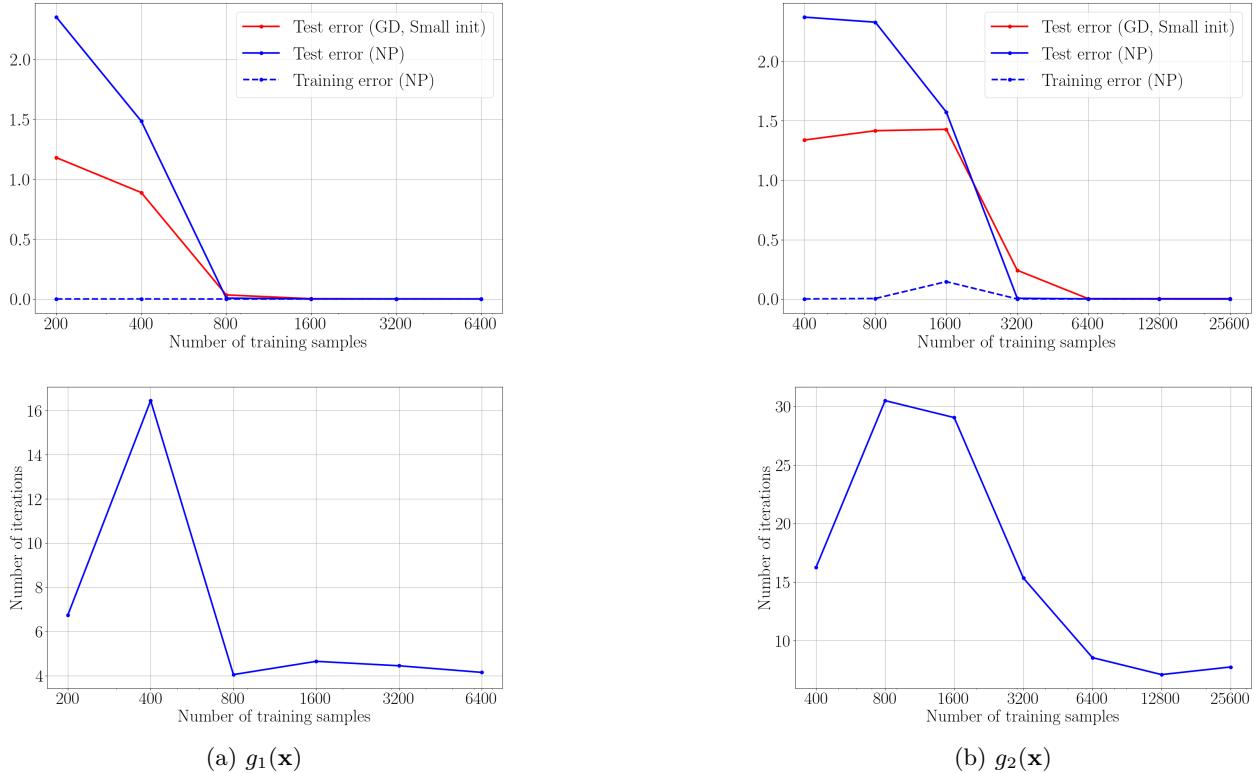


Figure 7: We train three-layer neural networks with activation function $\sigma(x) = \max(x, 0.5x)$ to learn $g_1(\mathbf{x})$ (left) and $g_2(\mathbf{x})$ (right). The network is trained using the NP algorithm for a maximum of 31 iterations, and using gradient descent (GD) with 50 neurons per layer for up to 3×10^6 iterations. For GD, the initialization is small (the weights are sampled uniformly at random from hypersphere of radius 0.01). The top row shows the training and test errors of various algorithms as the number of training samples increases. The bottom row shows the number of iterations required by the NP algorithm to successfully fit the training data. For $g_1(\mathbf{x})$, both algorithms achieve small test error with a similar number of samples. For $g_2(\mathbf{x})$, the NP algorithm achieves small test error with fewer samples.

small test error for both $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$. For $h_1(\mathbf{x})$, GD performs slightly worse than the NP algorithm. However, for $h_2(\mathbf{x})$, the NP algorithm is worse, requiring more training samples to achieve small test error.

As in earlier experiments, the number of iterations required by the NP algorithm for convergence increases in the beginning and then decreases and stabilizes. Also, for $h_2(\mathbf{x})$ with 1600 and 3200 training samples, the NP algorithm fails to achieve small training error in many instances within 31 iterations, which is sufficient number of iterations when learning with more or less samples.

5.2.2 Algorithmic Tasks

We next test efficacy of the proposed method on some algorithmic tasks. Our goal here is to demonstrate that the NP algorithm is able to successfully perform this task.

Modular Addition: We consider the task of learning modular addition using a two-layer neural network with square activation function, $\sigma(x) = x^2$. The goal is to map inputs $a, b \in \{0, 1, \dots, p-1\}$ to the output $(a + b) \bmod p$, where we set $p = 59$. Following prior work (Gromov, 2023; Nanda et al., 2023; Zhong et al., 2023), we formulate this as a p -class classification problem, representing both inputs and outputs as one-hot vectors. For a given input pair (a, b) , the network input is constructed as $[\mathbf{1}_a, \mathbf{1}_b, \mathbf{1}] \in \mathbb{R}^{2p+1}$, where $\mathbf{1}_z$ denotes the one-hot encoding of $z \in \{0, 1, \dots, p-1\}$. The final coordinate is fixed to 1, serving as an explicit bias term for the first layer. The target label is represented as $\mathbf{1}_c$, where $c = (a + b) \bmod p$.

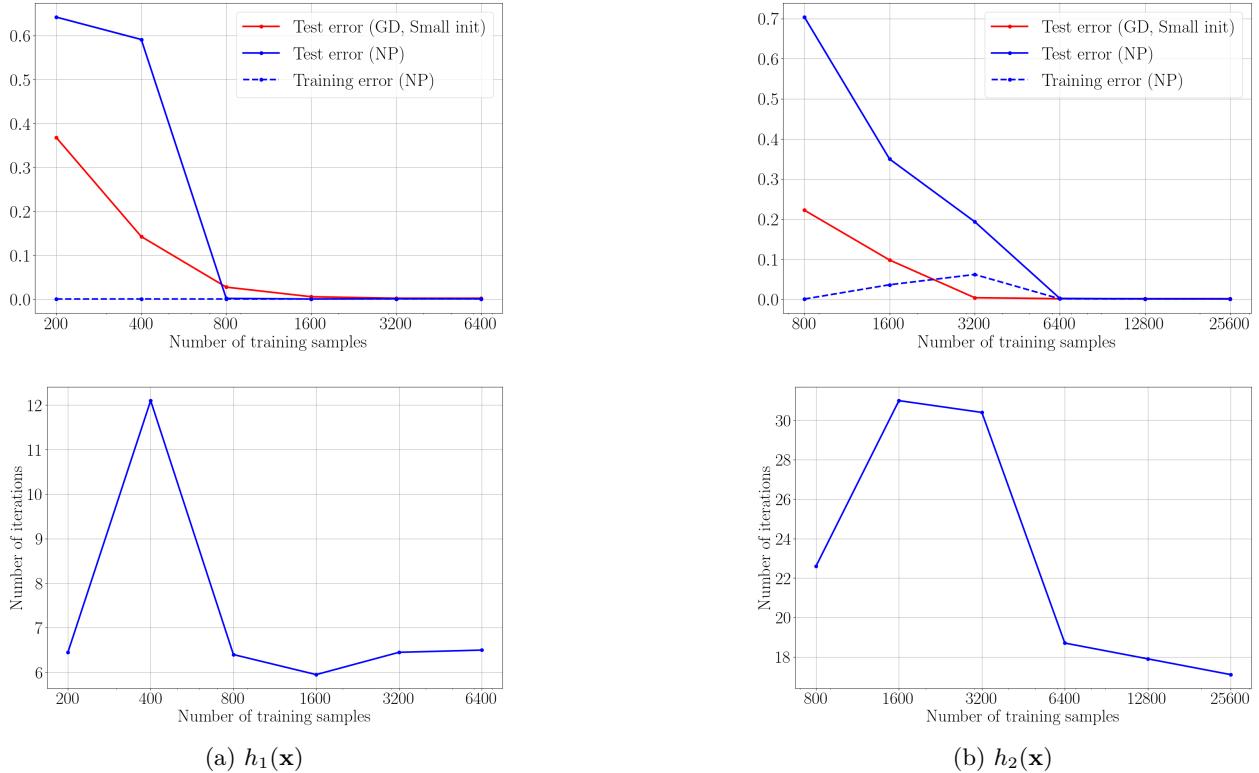
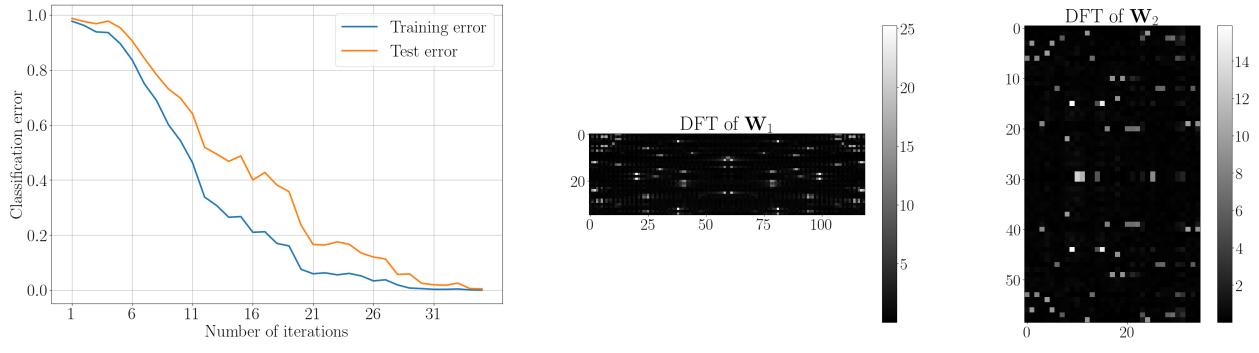


Figure 8: We train four-layer neural networks with activation function $\sigma(x) = \max(x, 0.5x)$ to learn $h_1(\mathbf{x})$ (left) and $h_2(\mathbf{x})$ (right). The network is trained using the NP algorithm for a maximum of 31 iterations, and using gradient descent (GD) with 50 neurons per layer for up to 3×10^6 iterations. For GD, the initialization is small (the weights are sampled uniformly at random from hypersphere of radius 0.1). The top row shows the training and test errors of various algorithms as the number of training samples increases. The bottom row shows the number of iterations required by the NP algorithm to successfully fit the training data. For $h_1(\mathbf{x})$, performance of GD is slightly worse. For $h_2(\mathbf{x})$, the NP algorithm requires more samples than GD to achieve small test error.

We train the model using the squared loss. The training set consists of 1500 input pairs, sampled uniformly at random from the $59 \cdot 59$ possible pairs, with the remainder forming the test set. During prediction, we select the class corresponding to the maximum output coordinate. The performance is measured using classification error on training and test set, which is defined as fraction of samples for which the network’s prediction does not match the target label.

Training is performed using the NP algorithm until the training classification error reaches zero. The experiment is repeated over three independent runs; here, we present the results from one representative run, with the remaining results given in the Appendix C.2.1 along with other specifications of the algorithm. Figure 9a depicts the evolution of training and test classification errors with respect to the iterations. The algorithm converges in 35 iterations, with the test error being small, indicating that the network successfully learns the modular addition function.

For this task, prior works have shown that the learned weights of trained networks exhibit a sinusoidal structure (Gromov, 2023; Nanda et al., 2023; Zhong et al., 2023). In particular, Gromov (2023) constructed a two-layer network with square activation in which the rows of the first-layer weights and the columns of the second-layer weights are sinusoidal. Figure 9b shows the absolute value of the 2D discrete Fourier transform (DFT) of the weights learned by the NP algorithm. We observe that the DFT of each row of the first layer and each column of the second layer concentrates around a certain frequency. Thus, similar to networks trained with gradient descent, the weights learned by the NP algorithm also have a sinusoidal structure.



(a) Evolution of training and test error (b) Absolute value of 2D DFT of the learned weights

Figure 9: We train a two-layer neural network with square activation function via the NP algorithm to learn modular addition. Panel (a) depicts the evolution of training and test classification errors with respect to the iterations. It shows that the learned network is able to achieve small training and test error. Panel (b) depicts the absolute value of 2D DFT of the learned weights. The DFT of each row of the first layer and each column of the second layer is concentrated around a certain frequency, indicating a clear sinusoidal structure among the learned weights.

Pointer Value Retrieval: We next consider the task of Pointer Value Retrieval (PVR), introduced by Zhang et al. (2021). In this task, part of the input acts as a *pointer* that selects a specific location in the input, whose value and its neighbors values determines the output. Formally, for $\mathbf{x} \in \mathbb{R}^n$ and a pointer $p \in \{1, 2, \dots, n\}$, the output is

$$f(p, \mathbf{x}) = \phi(x_p, x_{p+1}, \dots, x_{p+k}),$$

where x_i denotes the i th coordinate of \mathbf{x} and $\phi(\cdot)$ is a scalar-valued function. The complexity of the task increases with number of neighbors and also depends on the choice of $\phi(\cdot)$. In this setup, the network must learn to use the pointer to selectively attend to the relevant coordinates.

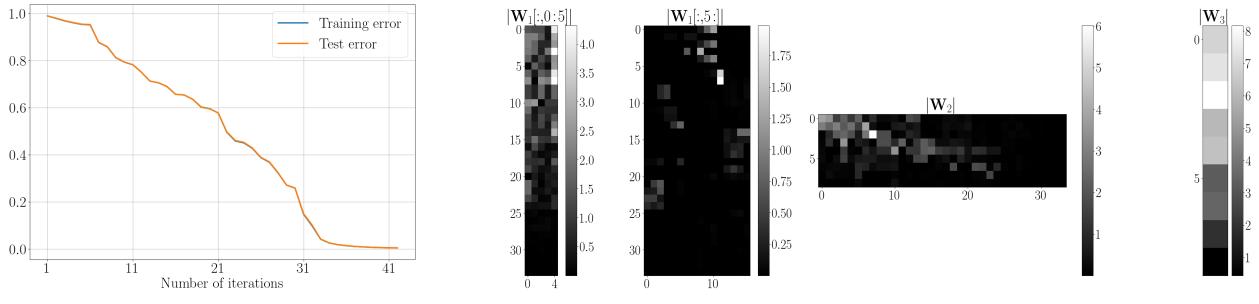
In our experiments, we consider the case where $\mathbf{x} \in \{\pm 1\}^{16}$, the pointer $p \in \{1, 2, \dots, 15\}$, and the output is

$$f(p, \mathbf{x}) = x_p x_{p+1}.$$

We train a three-layer neural network with activation function $\sigma(x) = \max(0, x)$ using the NP algorithm. The input to the network is $[\mathbf{p}, 1, \mathbf{x}] \in \mathbb{R}^{21}$, where $\mathbf{p} \in \mathbb{R}^4$ encodes the pointer in symmetric binary form—for instance, $p = 1$ is encoded as $\mathbf{p} = (-1, -1, -1, -1)$, $p = 2$ as $\mathbf{p} = (-1, -1, -1, 1)$, and so on. The fifth coordinate is fixed at 1, serving as a bias term for the first layer.

Training and test sets each consist of 100,000 samples, where every entry of \mathbf{p} and \mathbf{x} is independently set to 1 or -1 with equal probability. The performance is measured using training and test error, as defined in Section 5.2.1. We use square loss and train until the training error drops below 0.005. The experiment is repeated three times; we report a representative run here, with additional results in the Appendix C.2.2. As shown in Figure 10a, the algorithm converges within 42 iterations, achieving low test error and demonstrating that the network successfully learns the task.

The final network contains 34 neurons in the first layer and 9 in the second. Figure 10b depicts the absolute value of the learned weights. While a complete interpretation of the weights is beyond scope of this work, two notable observations can be made. First, the weights of the first layer associated with \mathbf{x} (the last 16 columns) are sparse, with dominant entries localized within each row. This is consistent with the task's dependence on specific coordinates and their neighbors. Second, the later rows and columns of the weights, which are associated with neurons added at the later stage, are relatively small. This is because the training error is already small by the later stages, and additional neurons and iterations were only needed to reduce the training error below the desired level.



(a) Evolution of training and test error

(b) Absolute value of the learned weights

Figure 10: We train a three-layer neural network with activation function $\sigma(x) = \max(0, x)$ via the NP algorithm to perform the PVR task. Panel (a) depicts the evolution of training and test errors with respect to the iterations, where both errors are almost overlapped. It shows that the learned network achieves small training and test error. Panel (b) depicts the absolute values of the learned weights. The weights of the first layer associated with \mathbf{x} (the last 16 columns) are plotted separately to highlight them better. They are sparse and dominant entries are localized within each row.

5.3 Discussion

The above experiments demonstrate the learning capability of neural networks trained via the NP algorithm. However, in certain instances, we observed a gap between the performance of NP algorithm and gradient descent with small initialization. This suggests that the NP algorithm is not necessarily equivalent to the gradient flow dynamics in the limit of initialization approaching zero. This is more evident if we consider two-layer diagonal linear networks. For such networks, it is known that in the limit of initialization approaching zero, the gradient flow converges to a minimum ℓ_1 -norm solution (Woodworth et al., 2020). However, for diagonal linear networks, NP algorithm is equivalent to the Orthogonal Matching Pursuit (OMP) algorithm (Pati et al., 1993; Tropp, 2004), under certain assumptions, as shown in Appendix D.5. We also empirically validate this claim. The OMP algorithm is a well-known technique to perform sparse recovery, but is different from minimizing the ℓ_1 -norm. This shows that even though the NP algorithm was motivated from the dynamics of gradient flow in the small initialization regime, it is not equivalent to gradient flow dynamics in the limit of initialization approaching zero.

We hypothesize that one of the reasons behind this gap is the way NP algorithm escapes from saddle points. As shown in Theorem 7 and stated in Remark 1, near a saddle point, the initial direction of the weights with small magnitude determine the KKT point to which those weights converge in direction. That, in turn, decides the way gradient flow escapes from that saddle point. Now, for homogeneous feed-forward neural networks, Lemma 5 shows that after escaping from the origin, gradient flow reaches a saddle point where a certain subset of weights have small magnitude; however, it does not characterize the direction of those subset of weights, which is critical for determining the KKT point. Lacking this information, the NP algorithm essentially uses random initial weights to obtain the KKT point. Perhaps, this leads to a different KKT point and a different escape direction from the saddle point. Further investigation along these lines to close the gap between gradient descent and NP algorithm is an interesting future direction.

Despite these differences, the NP algorithm offers a promising tool for studying neural networks. Unlike standard gradient-based methods, it provides a procedure in which neurons are incrementally added as training progresses, naturally favoring low-complexity or sparse solutions. This greedy construction offers a new lens for understanding how neural networks build representations. Developing a rigorous theoretical framework for the NP algorithm could therefore deepen our understanding of feature learning in neural networks. It can also serve as a foundation for future efforts to design similar greedy algorithms for more complex, non-homogeneous architectures.

We conclude this section by mentioning the work of Li et al. (2021), which presents a greedy algorithm to train deep *linear* neural networks. Their method proceeds by incrementally increasing the rank of the weight

matrices and is also inspired from the dynamics of gradient flow in the small initialization regime. However, the NP algorithm can not be extended to deep linear networks. As discussed at the end of Section 3.2 and in Appendix D.3, for deep linear networks, the constrained NCF becomes exactly zero at saddle points; thus, our results on gradient flow dynamics near saddle points are not applicable.

6 Conclusion

In this work, we analyzed the dynamics of gradient flow for homogeneous neural networks near saddle points with a sparsity structure. When initialized sufficiently close to such saddle points, gradient flow remains in their vicinity for a long time, during which weights with small magnitudes stay small and converge in direction. For feed-forward neural networks, the weights converged to a direction such that the norm of the incoming and outgoing weights of hidden neurons were proportional. We then used numerical experiments to describe the behavior of gradient descent after escaping these saddle points. Finally, motivated by these insights, we propose a greedy algorithm to train deep neural networks, called Neuron Pursuit.

Our results open several promising directions for future research. Establishing the validity of Assumption 2 used in Theorem 7, or even relaxing it, is one such direction. Another is to extend our theoretical analysis from square loss to other loss functions such as logistic loss. For logistic loss, the saddle points could be at infinity, making the analysis of dynamics near the saddle points considerably more difficult. It is also important to formally establish the empirical observations in Section 4 about the dynamics of gradient descent after escaping the saddle points.

Another direction is to close the gap between NP algorithm and gradient descent. Also, while the experiments in Section 5.2 demonstrate the learning ability of the NP algorithm, developing a theoretical understanding of this algorithm will be a major step towards demystifying neural networks. Extending the NP algorithm to non-homogeneous architectures such as Residual networks and Transformers is also a promising direction for further exploration.

References

- Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks. In *Proceedings of Thirty Fifth Conference on Learning Theory*, 2022.
- Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. SGD learning on neural networks: Leap complexity and saddle-to-saddle dynamics. In *The Thirty Sixth Annual Conference on Learning Theory*, 2023.
- Emmanuel Abbe, Samy Bengio, Enric Boix-Adsera, Eta Littwin, and Joshua Susskind. Transformers learn through gradual rank increase. *Advances in Neural Information Processing Systems*, 2024.
- Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, 2019.
- Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022.
- Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *arXiv preprint arXiv:2303.00055*, 2023.
- Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *Advances in neural information processing systems*, 35:9768–9783, 2022.
- Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4), 2007.
- Etienne Boursier and Nicolas Flammarion. Early alignment in two-layer networks training is a two-edged sword, 2024. URL <https://arxiv.org/abs/2401.10791>.
- Etienne Boursier, Lucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow ReLU networks for square loss and orthogonal inputs. In *Advances in Neural Information Processing Systems*, 2022.
- Alon Brutzkus and Amir Globerson. Why do larger models generalize better? A theoretical perspective via the XOR problem. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.
- Sourav Chatterjee. Convergence of gradient descent for deep neural networks. *arXiv preprint arXiv:2203.16462*, 2022.
- Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, 2019.
- Hung-Hsu Chou, Carsten Gieshoff, Johannes Maly, and Holger Rauhut. Gradient descent for deep matrix factorization: Dynamics and implicit bias towards low rank. *Applied and Computational Harmonic Analysis*, 2024.
- Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory*, 2022.
- Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. How two-layer neural networks learn, one (giant) step at a time. *arXiv preprint arXiv:2305.18270*, 2023.
- Yatin Dandi, Luca Pesce, Lenka Zdeborová, and Florent Krzakala. The computational advantage of depth: Learning high-dimensional hierarchical functions with gradient descent. *arXiv preprint arXiv:2502.13961*, 2025.
- Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems*, 2018.

-
- Spencer Frei, Gal Vardi, Peter Bartlett, Nathan Srebro, and Wei Hu. Implicit bias in leaky ReLU networks trained on high-dimensional data. In *The Eleventh International Conference on Learning Representations*, 2023.
- Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020.
- Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 2019.
- Daniel Gissin, Shai Shalev-Shwartz, and Amit Daniely. The implicit bias of depth: How incremental learning drives generalization. In *International Conference on Learning Representations*, 2020.
- Margalit Glasgow. SGD finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the XOR problem. In *The Twelfth International Conference on Learning Representations*, 2024.
- Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.
- Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, 2017.
- Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.
- Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.
- Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In *Advances in Neural Information Processing Systems*, 2020.
- Liwei Jiang, Yudong Chen, and Lijun Ding. Algorithmic regularization in model-free overparametrized asymmetric matrix factorization. *SIAM Journal on Mathematics of Data Science*, 2023.
- Jikai Jin, Zhiyuan Li, Kaifeng Lyu, Simon Shaolei Du, and Jason D. Lee. Understanding incremental learning of gradient descent: A fine-grained analysis of matrix sensing. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-Łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases*, 2016.
- Akshay Kumar and Jarvis Haupt. Directional convergence near small initializations and saddles in two-homogeneous neural networks. *Transactions on Machine Learning Research*, 2024.
- Akshay Kumar and Jarvis Haupt. Early directional convergence in deep homogeneous neural networks for small initializations. *Transactions on Machine Learning Research*, 2025a.
- Akshay Kumar and Jarvis Haupt. Towards understanding gradient flow dynamics of homogeneous neural networks beyond the origin, 2025b. URL <https://arxiv.org/abs/2502.15952>.
- Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, 2016.
- Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical programming*, 2019.
- Jason D Lee, Kazusato Oko, Taiji Suzuki, and Denny Wu. Neural network learns low-dimensional polynomials with sgd near the information-theoretic limit. *Advances in Neural Information Processing Systems*, 37: 58716–58756, 2024.

-
- Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*, 2021.
- Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59, 2022.
- Stanislas Łojasiewicz. Sur la géométrie semi- et sous- analytique. *Annales de l'Institut Fourier*, 43(5): 1575–1595, 1993. doi: 10.5802/aif.1384.
- Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2020.
- Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. In *Advances in Neural Information Processing Systems*, 2021.
- Jianhao Ma and Salar Fattah. Convergence of gradient descent with small initialization for unregularized matrix completion. In *Proceedings of Thirty Seventh Conference on Learning Theory*, 2024.
- Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes ReLU network features. *arXiv preprint arXiv:1803.08367*, 2018.
- Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: Dimension-free bounds and kernel limit. In *Conference on learning theory*, 2019.
- Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.
- Yuri Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.
- Y.C. Pati, R. Rezaifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, 1993.
- Scott Pesme and Nicolas Flammarion. Saddle-to-saddle dynamics in diagonal linear networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- Mary Phuong and Christoph H Lampert. The inductive bias of $\text{re}\{\text{lu}\}$ networks on orthogonally separable data. In *International Conference on Learning Representations*, 2021.
- Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5), 2017.
- BT Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.
- Noam Razin, Asaf Maman, and Nadav Cohen. Implicit regularization in hierarchical tensor factorization and deep convolutional neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.
- Lawrence K. Saul. Weight-balancing fixes and flows for deep learning. *Transactions on Machine Learning Research*, 2023.

James B Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *J. Mach. Learn. Res.*, 19(1), January 2018.

Dominik Stöger and Mahdi Soltanolkotabi. Small random initialization is akin to spectral learning: Optimization and generalization guarantees for overparameterized low-rank matrix reconstruction. In *Advances in Neural Information Processing Systems*, 2021.

Taiji Suzuki, Denny Wu, Kazusato Oko, and Atsushi Nitanda. Feature learning via mean-field langevin dynamics: classifying sparse parities and beyond. *Advances in Neural Information Processing Systems*, 2023.

J.A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. *Advances in Neural Information Processing Systems*, 2019.

Mingze Wang and Chao Ma. Understanding multi-phase optimization dynamics and rich nonlinear behaviors of ReLU networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Proceedings of Thirty Third Conference on Learning Theory*, pp. 3635–3673, 2020.

Nuoya Xiong, Lijun Ding, and Simon Shaolei Du. How over-parameterization slows down gradient descent in matrix sensing: The curses of symmetry and initialization. In *The Twelfth International Conference on Learning Representations*, 2024.

Greg Yang and Edward J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Chiyuan Zhang, Maithra Raghu, Jon Kleinberg, and Samy Bengio. Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *arXiv preprint arXiv:2107.12580*, 2021.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in neural information processing systems*, 2023.

A Key Lemmas

The following lemma, also known as Euler’s theorem, states two important properties of homogeneous functions (Lyu & Li, 2020, Theorem B.2).

Lemma 14. *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be locally Lipschitz, differentiable and L -positively homogeneous for some $L > 0$. Then,*

- For any $\mathbf{w} \in \mathbb{R}^d$ and $c \geq 0$, $\nabla f(c\mathbf{w}) = c^{L-1}f(\mathbf{w})$.
- For any $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{w}^\top \nabla f(\mathbf{w}) = Lf(\mathbf{w})$.

The above lemma also holds for certain non-differentiable homogeneous functions that satisfy additional requirements such as admitting a chain rule (Lyu & Li, 2020) or definable under an o-minimal structure Ji & Telgarsky (2020). This includes neural networks with ReLU activation.

The following lemma states an important property of functions that are homogeneous with respect to subset of variables.

Lemma 15. Let $f(\mathbf{w}_1, \mathbf{w}_2)$ be differentiable in $\mathbf{w}_1 \in \mathbb{R}^m$ and L -positively homogeneous with respect to $\mathbf{w}_2 \in \mathbb{R}^n$, for some $L > 0$. Then, $\nabla_{\mathbf{w}_1} f(\mathbf{w}_1, \mathbf{w}_2)$ is also L -positively homogeneous with respect to \mathbf{w}_2 .

Proof. For any $c \geq 0$ and $\mathbf{v} \in \mathbb{R}^m$, we have

$$\begin{aligned}\nabla_{\mathbf{w}_1} f(\mathbf{w}_1, c\mathbf{w}_2)^\top \mathbf{v} &= \lim_{\epsilon \rightarrow 0} \frac{f(\mathbf{w}_1 + \epsilon\mathbf{v}, c\mathbf{w}_2) - f(\mathbf{w}_1, c\mathbf{w}_2)}{\epsilon} \\ &= \lim_{\epsilon \rightarrow 0} \frac{c^L f(\mathbf{w}_1 + \epsilon\mathbf{v}, \mathbf{w}_2) - c^L f(\mathbf{w}_1, \mathbf{w}_2)}{\epsilon} = c^L \nabla_{\mathbf{w}_1} f(\mathbf{w}_1, \mathbf{w}_2)^\top \mathbf{v}.\end{aligned}$$

Since the above equality is true for any \mathbf{v} , we get

$$\nabla_{\mathbf{w}_1} f(\mathbf{w}_1, c\mathbf{w}_2) = c^L \nabla_{\mathbf{w}_1} f(\mathbf{w}_1, \mathbf{w}_2),$$

which completes the proof. \square

The next lemma states the Gronwall's inequality.

Lemma 16. Let α, β, u be real-valued functions defined on an interval $[a, b]$, where β and u are continuous and $\min(\alpha, 0)$ is integrable on every closed and bounded sub-interval of $[a, b]$.

- If β is non-negative and if u satisfies the integral inequality

$$u(t) \leq \alpha(t) + \int_a^t \beta(s)u(s)ds, \forall t \in [a, b],$$

then

$$u(t) \leq \alpha(t) + \int_0^t \alpha(s)\beta(s) \exp\left(\int_s^t \beta(r)dr\right) ds, \forall t \in [a, b].$$

- If, in addition, the function α is non-decreasing, then

$$u(t) \leq \alpha(t) \exp\left(\int_a^t \beta(s)ds\right), \forall t \in [a, b].$$

B Proofs omitted from Section 3

B.1 Proof of Theorem 7

We first state a key lemma used in the proof of Theorem 7. Assuming that the Łojasiewicz inequality holds, we establish that a reverse Łojasiewicz-type inequality also follows. The proof is adapted from a result in Karimi et al. (2016), which shows that the Polyak-Łojasiewicz inequality implies a quadratic growth condition.

Lemma 17. Suppose \mathbf{w}_* is a local minima of $g(\mathbf{w})$ such that Łojasiewicz's inequality is satisfied in a neighborhood of \mathbf{w}_* for some $\alpha \in (0, 1)$, that is, there exists $\mu, \gamma > 0$ such that

$$\|\nabla g(\mathbf{w})\|_2 \geq \mu(g(\mathbf{w}) - g(\mathbf{w}_*))^\alpha, \text{ if } \|\mathbf{w} - \mathbf{w}_*\|_2 \leq \gamma,$$

where $\alpha \in (0, 1)$. Then, there exists a $\mu_1 > 0$, $\gamma_1 \in (0, \gamma)$ such that

$$\|\nabla g(\mathbf{w})\|_2 \leq \mu_1(g(\mathbf{w}) - g(\mathbf{w}_*))^{1-\alpha}, \text{ if } \|\mathbf{w} - \mathbf{w}_*\|_2 \leq \gamma_1.$$

Proof. We use \mathbf{w}_*^γ to define the following set:

$$\mathbf{w}_*^\gamma := \{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_*\|_2 \leq \gamma\}.$$

For $\mathbf{w} \in \mathbf{w}_*^\gamma$, define $f(\mathbf{w}) := (g(\mathbf{w}) - g(\mathbf{w}_*))^{1-\alpha}$, and let $\mathcal{W}_* := \{\mathbf{w} \in \mathbf{w}_*^\gamma : g(\mathbf{w}) = g(\mathbf{w}_*)\}$. Then, for any $\mathbf{w} \in \mathbf{w}_*^\gamma \setminus \mathcal{W}_*$, we have

$$\|\nabla f(\mathbf{w})\|_2 = \left\| \frac{\nabla g(\mathbf{w})}{(g(\mathbf{w}) - g(\mathbf{w}_*))^\alpha} \right\|_2 \geq \mu.$$

Now, choose $\gamma_1 \in (0, \gamma)$ small enough such that $\gamma_1 \leq \gamma/4$ and $f(\mathbf{w})/\mu \leq \gamma/4$, for all $\mathbf{w} \in \mathbf{w}_*^{\gamma_1}$. Next, suppose $\mathbf{w}_0 \in \mathbf{w}_*^{\gamma_1}$ and let $\mathbf{w}(t)$ be the solution of the following differential equation:

$$\dot{\mathbf{w}} = -\nabla f(\mathbf{w}), \mathbf{w}(0) = \mathbf{w}_0.$$

Define

$$T^* = \min_{t \geq 0} \{t : \mathbf{w}(t) \in \mathcal{W}_* \text{ or } \mathbf{w}(t) \notin \mathbf{w}_*^\gamma\}.$$

Thus, T^* denotes the first time when either $\mathbf{w}(t)$ reaches the set \mathcal{W}_* or $\mathbf{w}(t)$ escapes from \mathbf{w}_*^γ . Thus, for all $t \in [0, T^*)$, $\|\nabla f(\mathbf{w}(t))\|_2 \geq \mu$.

Now, let $p(t) = \int_0^t \|\dot{\mathbf{w}}(s)\|_2 ds$. Then, for all $t \in [0, T^*)$, we have

$$\frac{df(\mathbf{w})}{dt} = -\|\nabla f(\mathbf{w}(t))\|_2^2 \leq -\mu \|\nabla f(\mathbf{w}(t))\|_2 = -\mu \|\dot{\mathbf{w}}(t)\|_2 = -\mu p(t).$$

Integrating the above equation from 0 to $t \in [0, T^*)$ we get

$$\mu p(t) \leq f(\mathbf{w}_0) - f(\mathbf{w}(t)) \leq f(\mathbf{w}_0). \quad (29)$$

Since $\|\mathbf{w}(t) - \mathbf{w}_0\|_2 \leq p(t)$, we get

$$\|\mathbf{w}(t) - \mathbf{w}_0\|_2 \leq f(\mathbf{w}_0)/\mu,$$

which implies, for all $t \in [0, T^*)$,

$$\|\mathbf{w}(t) - \mathbf{w}_*\|_2 \leq \|\mathbf{w}(t) - \mathbf{w}_0\|_2 + \|\mathbf{w}_0 - \mathbf{w}_*\|_2 \leq f(\mathbf{w}_0)/\mu + \gamma_1 \leq \gamma/2.$$

The above inequality implies $\mathbf{w}(t) \in \mathbf{w}_*^{\gamma/2}$, for all $t \in [0, T^*)$ and hence, $\mathbf{w}(T^*) \in \mathbf{w}_*^\gamma$.

Next, we derive an upper bound on T^* . For all $t \in [0, T^*)$, we have

$$f(\mathbf{w}(t)) - f(\mathbf{w}_0) = - \int_0^t \|\nabla f(\mathbf{w}(s))\|_2^2 ds \leq -\mu^2 t,$$

which implies

$$\mu^2 t \leq f(\mathbf{w}_0) - f(\mathbf{w}(t)) \leq f(\mathbf{w}_0).$$

Taking $t \rightarrow T^*$, we get $T^* \leq f(\mathbf{w}_0)/\mu^2$. Thus, T^* is finite and since $\mathbf{w}(T^*) \in \mathbf{w}_*^\gamma$, we get $\mathbf{w}(T^*) \in \mathcal{W}_*$. Now, from eq. (29), we know

$$f(\mathbf{w}_0) \geq \mu \|\mathbf{w}(t) - \mathbf{w}_0\|_2, \text{ for all } t \in [0, T^*].$$

Taking $t \rightarrow T^*$ in the above equation gives us

$$f(\mathbf{w}_0) \geq \mu \|\mathbf{w}(T^*) - \mathbf{w}_0\|_2. \quad (30)$$

Since $g(\mathbf{w})$ has locally Lipschitz gradient, there exists a large enough $K > 0$ such that, for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathbf{w}_*^\gamma$, we have

$$\|\nabla g(\mathbf{w}_1) - \nabla g(\mathbf{w}_2)\|_2 \leq K\|\mathbf{w}_1 - \mathbf{w}_2\|_2.$$

Since $\nabla g(\mathbf{w}(T^*)) = \mathbf{0}$, using the above equation in eq. (30) gives us

$$(g(\mathbf{w}_0) - g(\mathbf{w}_*))^{1-\alpha} = f(\mathbf{w}_0) \geq \mu\|\mathbf{w}(T^*) - \mathbf{w}_0\|_2 \geq \frac{\mu\|\nabla g(\mathbf{w}_0)\|_2}{K},$$

which completes the proof since $\mathbf{w}_0 \in \mathbf{w}_*^{\gamma_1}$ and it is arbitrary. \square

Proof of Theorem 7: Since $(\bar{\mathbf{w}}_n, \mathbf{0})$ is a saddle point of the loss function in eq. (13) such that Assumption 2 is satisfied, then there exists $\mu_1, \gamma > 0$ such that

$$\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 \geq \mu_1(\tilde{\mathcal{L}}(\mathbf{w}_n) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^\alpha, \text{ if } \|\mathbf{w}_n - \bar{\mathbf{w}}_n\|_2 \leq \gamma, \quad (31)$$

where $\alpha \in \left(0, \frac{L}{2(L-1)}\right)$.

Without loss of generality, from Lemma 17, we can also assume that there exists $\mu_2 > 0$ such that

$$\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 \leq \mu_2(\tilde{\mathcal{L}}(\mathbf{w}_n) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha}, \text{ if } \|\mathbf{w}_n - \bar{\mathbf{w}}_n\|_2 \leq \gamma. \quad (32)$$

Suppose $\mathbf{u}_1(t)$ is the solution of the following differential equation:

$$\dot{\mathbf{u}} = \nabla_{\mathbf{w}_z} \mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1} = \mathcal{J}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u})^\top \bar{\mathbf{y}}, \mathbf{u}(0) = \mathbf{z},$$

If $\mathbf{z} \in \mathcal{S}(\mathbf{z}_*; \mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1})$, where \mathbf{z}_* is a non-negative KKT point of $\mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1}$, then for any arbitrarily small $\epsilon > 0$ there exists a T_ϵ such that

$$\frac{\mathbf{z}_*^\top \mathbf{u}_1(T_\epsilon)}{\|\mathbf{u}_1(T_\epsilon)\|_2} \geq 1 - \epsilon.$$

We can also assume that there exists a sufficiently large constant B_ϵ and small enough $\eta > 0$ such that $\eta \leq \|\mathbf{u}_1(t)\|_2 \leq B_\epsilon$, for all $t \in [0, T_\epsilon]$.

Else, from Lemma 1, $\mathbf{u}_1(t)$ converges to the origin. In this case, we define $\eta = 1$, $B_\epsilon = \epsilon$, and for any arbitrarily small $\epsilon > 0$ there exists a T_ϵ such that

$$\|\mathbf{u}_1(T_\epsilon)\|_2 \leq B_\epsilon = \epsilon.$$

Next, for $\delta > 0$, let $\mathbf{u}_\delta(t)$ be the solution of the following differential equation:

$$\dot{\mathbf{u}} = \nabla_{\mathbf{w}_z} \mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1} = \mathcal{J}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u})^\top \bar{\mathbf{y}}, \mathbf{u}(0) = \delta \mathbf{z},$$

Then, from Lemma 6, we know $\mathbf{u}_1(t) = \mathbf{u}_\delta(t/\delta^{L-2})/\delta$. Hence, if $\mathbf{z} \in \mathcal{S}(\mathbf{z}_*; \mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1})$, then $\eta\delta \leq \|\mathbf{u}_\delta(t)\|_2 \leq B_\epsilon\delta$, for all $t \in [0, T_\epsilon/\delta^{L-2}]$, and

$$\frac{\mathbf{z}_*^\top \mathbf{u}_\delta(T_\epsilon/\delta^{L-2})}{\|\mathbf{u}_\delta(T_\epsilon/\delta^{L-2})\|_2} \geq 1 - \epsilon.$$

Else, $\|\mathbf{u}_\delta(T_\epsilon/\delta^{L-2})\|_2 \leq B_\epsilon\delta = \epsilon\delta$.

We next define

$$\mathbf{Z}(t) = \max(\|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2^2, \|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2^2/\delta^2).$$

Let $\epsilon \in (0, \gamma)$ be arbitrarily small. We will show that $\mathbf{Z}(t) \leq \epsilon^2$, for all $t \in [0, T_\epsilon/\delta^{L-2}]$ and for all sufficiently small $\delta > 0$. Note that $\mathbf{Z}(0) = \delta^2 < \epsilon^2 \leq \gamma^2$. Define

$$\bar{T}_\delta = \inf_{t \in [0, T_\epsilon/\delta^{L-2}]} \{t : \mathbf{Z}(t) = \epsilon^2\}.$$

Here, \bar{T}_δ indicates the time when $\mathbf{Z}(t)$ becomes ϵ^2 for the first time in the interval $[0, T_\epsilon/\delta^{L-2}]$. Thus, for all $t \in [0, \bar{T}_\delta]$, $\mathbf{Z}(t) \leq \epsilon^2 \leq \gamma^2$. Note that if we can show $\bar{T}_\delta = T_\epsilon/\delta^{L-2}$, then it would imply $\mathbf{Z}(t) \leq \epsilon^2$, for all $t \in [0, T_\epsilon/\delta^{L-2}]$.

Now, for all $t \in [0, \bar{T}_\delta]$,

$$\begin{aligned} \frac{d\mathbf{w}_n}{dt} &= \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))) \\ &= \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})) + \mathbf{q}(t), \end{aligned} \quad (33)$$

where

$$\begin{aligned} \mathbf{q}(t) &= (\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0}))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))) \\ &\quad + \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})^\top (\mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))). \end{aligned}$$

Using the definition of $\tilde{\mathcal{L}}(\mathbf{w}_n)$, we get

$$\frac{d\mathbf{w}_n}{dt} = -\nabla_{\mathbf{w}_n} \tilde{\mathcal{L}}(\mathbf{w}_n) + \mathbf{q}(t). \quad (34)$$

Now, recall that for all $t \in [0, \bar{T}_\delta]$,

$$\|\mathbf{w}_z(t)\|_2 \leq \|\mathbf{u}_\delta(t)\|_2 + \delta\epsilon \leq \delta(B_\epsilon + \epsilon).$$

Therefore, using the third condition in Assumption 1, we get

$$\begin{aligned} &\|(\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0}))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)))\|_2 \\ &\leq \|\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})\|_2 \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 \\ &\leq \|\mathcal{J}_{n1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) + O(\delta^K)\|_2 \|\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 = O(\delta^L), \end{aligned}$$

where $\mathcal{J}_{n1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))$ denotes the Jacobian of $\mathcal{H}_1(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))$ with respect to \mathbf{w}_n . The final equality holds since $\mathcal{J}_{n1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))$ is L -homogeneous in \mathbf{w}_z and $\|\mathbf{w}_z\|_2 = O(\delta)$, and $K > L$.

Next, using the first condition in Assumption 1, we get

$$\begin{aligned} &\|\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})^\top (\mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)))\|_2 \\ &\leq \|\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})\|_2 \|\mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 \\ &\leq \|\mathcal{J}_n(\mathbf{X}; \mathbf{w}_n(t), \mathbf{0})\|_2 \|\mathcal{H}_1(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) + O(\delta^K)\|_2 = O(\delta^L), \end{aligned}$$

where the final equality holds since $\mathcal{H}_1(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))$ is L -homogeneous in \mathbf{w}_z and $\|\mathbf{w}_z\|_2 = O(\delta)$, and $K > L$. Therefore, combining the above two inequalities, we get

$$\|\mathbf{q}(t)\|_2 \leq C\delta^L, \text{ for all } t \in [0, \bar{T}_\delta]$$

where $C > 0$ is a sufficiently large constant. Now, let $p(t)$ denote the length of $\mathbf{w}_n(t)$, that is,

$$p(t) = \int_0^t \|\dot{\mathbf{w}}_n(s)\|_2 ds,$$

then $\dot{p}(t) \leq \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 + \|\mathbf{q}(t)\|_2 \leq \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 + C\delta^L$.

Now, if for some $t \in [0, \bar{T}_\delta]$, $\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) \leq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, then, from eq. (32),

$$\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 \leq \mu_2(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha} = O(\delta^{\frac{L(1-\alpha)}{\alpha}}),$$

which implies

$$\dot{p}(t) \leq \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 + O(\delta^L) = O(\delta^\beta), \quad (35)$$

where $\beta = \min(L, L(1 - \alpha)/\alpha)$.

Next, if $(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n)) \geq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, then using eq. (31), we get $\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 \geq C\delta^L$. Hence,

$$\begin{aligned} \frac{d(\tilde{\mathcal{L}}(\mathbf{w}_n) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))}{dt} &= \nabla \tilde{\mathcal{L}}(\mathbf{w}_n)^\top \dot{\mathbf{w}}_n \\ &= -\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2^2 + \nabla \tilde{\mathcal{L}}(\mathbf{w}_n)^\top \mathbf{q}(t) \\ &\leq -\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2^2 + \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 \|\mathbf{q}(t)\|_2 \\ &\leq -\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2^2 + C\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 \delta^L = \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 (-\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 + C\delta^L) \leq 0. \end{aligned} \quad (36)$$

Thus, if for some $t_0 \in [0, \bar{T}_\delta]$, $(\tilde{\mathcal{L}}(\mathbf{w}_n(t_0)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n)) = (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, then $(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n)) \leq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, for all $t \in [t_0, \bar{T}_\delta]$.

Now, suppose $\tilde{\mathcal{L}}(\mathbf{w}_n(0)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) > (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$ and there exists $T_1 \in (0, \bar{T}_\delta]$ such that $\tilde{\mathcal{L}}(\mathbf{w}_n(T_1)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) = (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$ for the first time. Then, for all $t \in (0, T_1]$, $\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) \geq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, and for all $t \in [T_1, \bar{T}_\delta]$, $\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) \leq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$. Since, for $t \in [0, T_1]$, $\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) \geq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$, we get

$$\begin{aligned} \frac{d(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))}{dt} &\leq \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 (-\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 + C\delta^L) \\ &\leq \mu_1 (\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^\alpha (-\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n)\|_2 + C\delta^L) \\ &\leq \mu_1 (\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^\alpha (-\dot{p}(t) + 2C\delta^L), \end{aligned}$$

where the first inequality follows from eq. (36). The second inequality follows from eq. (31) and since $\|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 \geq C\delta^L$. The last inequality is true since $\dot{p}(t) \leq \|\nabla \tilde{\mathcal{L}}(\mathbf{w}_n(t))\|_2 + C\delta^L$, for all $t \in [0, \bar{T}_\delta]$. Hence,

$$\frac{d(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha}}{dt} \leq \mu_1(1-\alpha)(-\dot{p}(t) + 2C\delta^L).$$

Integrating both sides from 0 to $t \in [0, T_1]$ we get

$$\begin{aligned} \mu_1(1-\alpha)p(t) &\leq (\tilde{\mathcal{L}}(\mathbf{w}_n(0)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha} - (\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha} + 2\mu_1(1-\alpha)C\delta^L t \\ &\leq (\tilde{\mathcal{L}}(\mathbf{w}_n(0)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n))^{1-\alpha} + 2\mu_1(1-\alpha)C\delta^L t \\ &\leq K_1 \|\mathbf{w}_n(0) - \bar{\mathbf{w}}_n\|_2^{1-\alpha} + 2\mu_1(1-\alpha)C_1\delta^L t = K_1\delta^{1-\alpha} + 2\mu_1(1-\alpha)C\delta^L t, \end{aligned}$$

where $K_1 > 0$ is a sufficiently large constant, and the last inequality follows from $\tilde{\mathcal{L}}(\cdot)$ having locally Lipschitz gradient. Hence, for all $t \in [0, T_1]$,

$$\|\mathbf{w}_n(t) - \mathbf{w}_n(0)\|_2 \leq p(t) \leq \frac{K_1\delta^{1-\alpha}}{\mu_1(1-\alpha)} + 2C\delta^L t \leq \frac{K_1\delta^{1-\alpha}}{\mu_1(1-\alpha)} + 2C\delta^2 T_\epsilon = O(\delta^{1-\alpha}).$$

where the last inequality follows since $T_1 \leq \bar{T}_\delta \leq T_\epsilon/\delta^{L-2}$.

Next, for $t \in [T_1, \bar{T}_\delta]$, we have $(\tilde{\mathcal{L}}(\mathbf{w}_n(t)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n)) \leq (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$. Therefore, from eq. (35), we get

$$\dot{p}(t) = O(\delta^\beta).$$

The above equation implies, for all $t \in [T_1, \bar{T}_\delta]$,

$$\|\mathbf{w}_n(t) - \mathbf{w}_n(T_1)\|_2 = \left\| \int_{T_1}^t \dot{\mathbf{w}}_n(s) ds \right\|_2 \leq \int_{T_1}^t \|\dot{\mathbf{w}}_n(s)\|_2 ds = \int_{T_1}^t \dot{p}(s) ds \leq O(\delta^\beta)(t - T_1) = O(\delta^{\beta-L+2}),$$

where the last equality holds since $t - T_1 \leq \bar{T}_\delta \leq T_\epsilon/\delta^{L-2}$. Note that, since $\alpha \in (0, L/(2L-2))$, therefore, $L(1-\alpha)/\alpha - L + 2 > 0$. This implies $\beta > L - 2$.

Now, define $\beta_1 = \min(\beta - L + 2, 1 - \alpha)$, then, for all $t \in [0, \bar{T}_\delta]$,

$$\|\mathbf{w}_n(t) - \mathbf{w}_n(0)\|_2 = O(\delta^{\beta_1}), \text{ and thus, } \|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2 = O(\delta^{\beta_1}). \quad (37)$$

Note that we have assumed $\tilde{\mathcal{L}}(\mathbf{w}_n(0)) - \tilde{\mathcal{L}}(\bar{\mathbf{w}}_n) > (C\delta^L/\mu_1)^{\frac{1}{\alpha}}$. If this assumption is not true, then we can simply choose $T_1 = 0$ and get the above bound in this instance as well.

Next, for all $t \in [0, \bar{T}_\delta]$, we have

$$\begin{aligned} \frac{d\mathbf{w}_z}{dt} &= \mathcal{J}_z(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))) \\ &= \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0})) + \mathbf{r}(t), \end{aligned} \quad (38)$$

where

$$\begin{aligned} \mathbf{r}(t) &= (\mathcal{J}_z(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)))^\top (\mathbf{y} - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))) \\ &\quad + \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))^\top (\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))), \end{aligned}$$

and $\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ denotes the Jacobian of $\mathcal{H}_1(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$ with respect to \mathbf{w}_z . We next show that, for $t \in [0, \bar{T}_\delta]$, $\|\mathbf{r}(t)\|_2 = O(\delta^{L-1+\beta_2})$, for some $\beta_2 > 0$. Note that,

$$\begin{aligned} &\|\mathcal{J}_z(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))\|_2 \\ &\leq \|\mathcal{J}_z(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 + \|\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))\|_2 \\ &= O(\delta^{K-1}) + \delta^{L-1} \|\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t)/\delta) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)/\delta)\|_2 \\ &\leq O(\delta^{K-1}) + K_2 \delta^{L-1} \|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2 = O(\delta^{K-1}) + O(\delta^{L-1+\beta_1}), \end{aligned} \quad (39)$$

where K_2 is a sufficiently large constant. The first term in the first equality follows from the second condition in Assumption 1 and since $\|\mathbf{w}_z(t)\|_2 = O(\delta)$. The second term follows since $\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is $(L-1)$ -homogeneous in \mathbf{w}_z . The second inequality holds since $\mathbf{w}_z(t)/\delta$ is bounded by a constant for all $t \in [0, \bar{T}_\delta]$ and all sufficiently small δ and since $\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is locally Lipschitz in \mathbf{w}_n . The final equality follows from eq. (37). Next,

$$\begin{aligned} &\left\| \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))^\top (\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))) \right\|_2 \\ &\leq \|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))\|_2 \|\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 \\ &= \delta^{L-1} \|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)/\delta)\|_2 \|\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) + \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2 \\ &\leq K_3 \delta^{L-1} (\|\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{0}) - \mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t))\|_2 + \|\mathcal{H}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) - \mathcal{H}(\mathbf{X}; \mathbf{w}_n(t), \mathbf{w}_z(t))\|_2) \\ &\leq K_3 \delta^{L-1} (K_4 \|\mathbf{w}_z(t)\|_2 + K_5 \|\bar{\mathbf{w}}_n - \mathbf{w}_n(t)\|_2) = O(\delta^L) + O(\delta^{L-1+\beta_1}), \end{aligned} \quad (40)$$

where K_3, K_4, K_5 are sufficiently large positive constants. The first equality follows since $\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is $(L-1)$ -homogeneous in \mathbf{w}_z . The second inequality is true since $\mathbf{w}_z(t)/\delta$ is bounded by a constant for all sufficiently small $\delta > 0$ and thus, $\|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)/\delta)\|_2$ is bounded as well, for all $t \in [0, \bar{T}_\delta]$. The final inequality is true since $\mathcal{H}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is locally Lipschitz in \mathbf{w}_n and \mathbf{w}_z . The final inequality follows since $\|\mathbf{w}_z(t)\|_2 = O(\delta)$, for all $t \in [0, \bar{T}_\delta]$, and from eq. (37). Therefore, if we define $\beta_2 = \min(K - L, \beta_1, 1)$, then from eq. (39) and eq. (40), we get

$$\|\mathbf{r}(t)\|_2 = O(\delta^{L-1+\beta_2}), \text{ for all } t \in [0, \bar{T}_\delta]. \quad (41)$$

Next, note that

$$\begin{aligned} &\frac{1}{2} \frac{d\|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2^2}{dt} \\ &= (\mathbf{w}_z(t) - \mathbf{u}_\delta(t))^\top (\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u}_\delta(t)))^\top \bar{\mathbf{y}} + (\mathbf{w}_z(t) - \mathbf{u}_\delta(t))^\top \mathbf{r}(t) \\ &\leq \|\mathbf{w}_z(t) - \mathbf{u}_z(t)\|_2 \|\bar{\mathbf{y}}\|_2 \|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u}_\delta(t))\|_2 + \|\mathbf{w}_z(t) - \mathbf{u}_z(t)\|_2 \|\mathbf{r}(t)\|_2. \end{aligned}$$

The above equation can be simplified to get

$$\begin{aligned} \frac{d\|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2}{dt} &\leq \|\bar{\mathbf{y}}\|_2 \|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u}_\delta(t))\|_2 + \|\mathbf{r}(t)\|_2 \\ &= \delta^{L-1} \|\bar{\mathbf{y}}\|_2 \|\mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{w}_z(t)/\delta) - \mathcal{J}_{z1}(\mathbf{X}; \bar{\mathbf{w}}_n, \mathbf{u}_\delta(t)/\delta)\|_2 + \|\mathbf{r}(t)\|_2 \\ &\leq K_6 \delta^{L-2} \|\bar{\mathbf{y}}\|_2 \|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2 + K_7 \delta^{L-1+\beta_2}, \end{aligned}$$

where K_6, K_7 are sufficiently large positive constants. The first equality follows since $\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is $(L-1)$ -homogeneous in \mathbf{w}_z . The second inequality follows from eq. (41), since $\mathbf{w}_z(t)/\delta$ and $\mathbf{u}_\delta(t)/\delta$ is bounded by a constant for all sufficiently small $\delta > 0$ and $\mathcal{J}_{z1}(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z)$ is locally Lipschitz in \mathbf{w}_z . Now, integrating the above equation from 0 to t and then using Lemma 16 gives us

$$\|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2 \leq K_7 e^{K_6 \delta^{L-2} \|\bar{\mathbf{y}}\|_2 t} \delta^{L-1+\beta_2} t, \text{ for all } t \in [0, \bar{T}_\delta].$$

Since $\bar{T}_\delta \leq T_\epsilon/\delta^{L-2}$, we get

$$\|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2 \leq K_7 T_\epsilon e^{K_6 \|\bar{\mathbf{y}}\|_2 T_\epsilon} \delta^{1+\beta_2} = O(\delta^{1+\beta_2}), \text{ for all } t \in [0, \bar{T}_\delta]. \quad (42)$$

Now, for the sake of contradiction, let $\bar{T}_\delta < T_\epsilon/\delta^{L-2}$. Then, from the definition of \bar{T}_δ , at least one of the following equations must hold

$$\|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2^2 = \epsilon^2 \text{ or } \|\mathbf{w}_z(t) - \mathbf{u}_\delta(t)\|_2^2 / \delta^2 = \epsilon^2.$$

However, from eq. (37) and eq. (42), we observe that neither of the above two equations can hold leading to a contradiction. Hence, $\bar{T}_\delta = T_\epsilon/\delta^{L-2}$. Thus, from eq. (37), eq. (42) and since $\|\mathbf{u}_\delta(t)\|_2 = O(\delta)$, we get

$$\|\mathbf{w}_n(t) - \bar{\mathbf{w}}_n\|_2 = O(\delta^{\beta_1}) \text{ and } \|\mathbf{w}_z(t)\|_2 = O(\delta), \text{ for all } t \in \left[0, \frac{T_\epsilon}{\delta^{L-2}}\right], \text{ where } \beta_1 > 0.$$

Now, define $\bar{T} = T_\epsilon/\delta^{L-2}$. From eq. (42), we know

$$\|\mathbf{w}_z(\bar{T}) - \mathbf{u}_\delta(\bar{T})\|_2 = O(\delta^{1+\beta_2}). \quad (43)$$

Thus, we may write $\mathbf{w}_z(\bar{T}) = \mathbf{u}_\delta(\bar{T}) + \zeta$, where $\|\zeta\|_2 = O(\delta^{1+\beta_2})$. If $\mathbf{z} \in \mathcal{S}(\mathbf{z}_*; \mathcal{N}_{\bar{\mathbf{y}}, \mathcal{H}_1})$, then since $\|\mathbf{u}_\delta(\bar{T})\|_2 \geq \eta\delta$, we have $\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2 \geq \eta\delta/2$, for all sufficiently small δ . Hence,

$$\frac{\mathbf{w}_z(\bar{T})}{\|\mathbf{w}_z(\bar{T})\|_2} = \frac{\mathbf{u}_\delta(\bar{T}) + \zeta}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2},$$

which implies

$$\frac{\mathbf{w}_z(\bar{T})^\top \mathbf{z}_*}{\|\mathbf{w}_z(\bar{T})\|_2} = \frac{\mathbf{u}_\delta(\bar{T})^\top \mathbf{z}_* + \zeta^\top \mathbf{z}_*}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2} = \left(\frac{\mathbf{u}_\delta(\bar{T})^\top \mathbf{z}_*}{\|\mathbf{u}_\delta(\bar{T})\|_2} \right) \frac{\|\mathbf{u}_\delta(\bar{T})\|_2}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2} + \frac{\zeta^\top \mathbf{z}_*}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2}.$$

Now, note that

$$\frac{\|\mathbf{u}_\delta(\bar{T})\|_2}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2} \geq \frac{\|\mathbf{u}_\delta(\bar{T})\|_2}{\|\mathbf{u}_\delta(\bar{T})\|_2 + \|\zeta\|_2} = \frac{1}{1 + \frac{\|\zeta\|_2}{\|\mathbf{u}_\delta(\bar{T})\|_2}} \geq 1 - \frac{\|\zeta\|_2}{\|\mathbf{u}_\delta(\bar{T})\|_2} = 1 - O(\delta^{\beta_2}),$$

and

$$\frac{\zeta^\top \mathbf{z}_*}{\|\mathbf{u}_\delta(\bar{T}) + \zeta\|_2} \geq -O(\delta^{\beta_2}).$$

Hence, for all sufficiently small $\delta > 0$, we get

$$\frac{\mathbf{w}_z(\bar{T})^\top \mathbf{z}_*}{\|\mathbf{w}_z(\bar{T})\|_2} \geq (1 - \epsilon) (1 - O(\delta^{\beta_2})) - O(\delta^{\beta_2}) = 1 - O(\epsilon).$$

Else, $\|\mathbf{u}_\delta(\bar{T})\|_2 \leq \epsilon\delta$, which implies

$$\|\mathbf{w}_z(\bar{T})\|_2 \leq \|\mathbf{u}_\delta(\bar{T})\|_2 + \|\zeta\|_2 = \epsilon O(\delta) + O(\delta^{1+\beta_2}) = \epsilon O(\delta),$$

for all sufficiently small $\delta > 0$. This completes the proof. \square

B.2 Proof of Lemma 8

We first prove the second case, where $\alpha = 1$ and $p \geq 1$.

Case 2 ($\alpha = 1$ and $p \geq 1$): For all $l \in [L - 1]$, define

$$\phi_0 = \mathbf{x}, \mathbf{h}_1 = \mathbf{W}_1 \mathbf{x}, \phi_l = \sigma(\mathbf{h}_l), \mathbf{h}_{l+1} = \mathbf{W}_{l+1} \phi_l.$$

For $1 \leq k \leq l \leq L - 1$, define

$$f_{l,k}(\mathbf{s}) = \sigma(\mathbf{N}_l \sigma(\mathbf{N}_{l-1} \sigma(\dots \sigma(\mathbf{N}_k \sigma(\mathbf{s}))))).$$

For all $2 \leq l \leq L - 1$, we claim that

$$\sigma(\mathbf{h}_l) = \begin{bmatrix} \sigma(\mathbf{N}_l \sigma(\mathbf{N}_{l-1} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}, \quad (44)$$

where $p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ are a vector-valued homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity $p + 1$ and p , respectively. More specifically,

$$\begin{aligned} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l,k+2}^{\top}(\mathbf{N}_{k+1} \mathbf{g}_k(\mathbf{x})) \mathbf{B}_{k+1} \sigma(\mathbf{A}_k \mathbf{g}_{k-1}(\mathbf{x})) + \text{diag}(\sigma'(\mathbf{N}_l \mathbf{g}_{l-1}(\mathbf{x}))) \mathbf{B}_l \sigma(\mathbf{A}_{l-1} \mathbf{g}_{l-2}(\mathbf{x})), \\ s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})). \end{aligned}$$

Furthermore, $q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ are vector-valued functions that are sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity of strictly greater than $p + 1$ and p , respectively.

Now, if the claim stated in eq. (44) is true, then we get

$$\begin{aligned} \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \mathbf{W}_L \sigma(\mathbf{h}_{L-1}) \\ &= [\mathbf{N}_L \quad \mathbf{B}_L] \begin{bmatrix} \sigma(\mathbf{N}_{L-1} \sigma(\mathbf{N}_{L-2} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix} \\ &= \mathbf{N}_L \sigma(\mathbf{N}_{L-1} \sigma(\mathbf{N}_{L-2} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z), \end{aligned}$$

where in the final equality we used $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) = \mathbf{N}_L \sigma(\mathbf{N}_{L-1} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))$. Since \mathbf{N}_L does not belong to \mathbf{w}_z and $p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z , therefore, $\mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z . Also, since $q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity of strictly greater than $p + 1$, $\mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ will be sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity of strictly greater than $p + 1$.

Next, \mathbf{B}_L is 1-homogeneous in \mathbf{w}_z and $s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z , which implies $\mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z . Also, since $r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity of strictly greater than p , $\mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ will be sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity of strictly greater than $p + 1$.

Therefore,

$$\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \sum_{i=1}^m \mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z),$$

where $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z . Also, each of $\{\mathcal{H}_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\}_{i=2}^m$ have degree of homogeneity strictly greater than $p + 1$.

Finally, using the definition of $p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, we get

$$\begin{aligned} & \mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \sum_{k=1}^{L-3} \mathbf{N}_L \nabla_{\mathbf{s}} f_{L-1,k+2}^{\top} (\mathbf{N}_{k+1} \mathbf{g}_k(\mathbf{x})) \mathbf{B}_{k+1} \sigma(\mathbf{A}_k \mathbf{g}_{k-1}(\mathbf{x})) + \mathbf{N}_L \operatorname{diag}(\sigma'(\mathbf{N}_{L-1} \mathbf{g}_{L-2}(\mathbf{x}))) \mathbf{B}_{L-1} \sigma(\mathbf{A}_{L-2} \mathbf{g}_{L-3}(\mathbf{x})) \\ &+ \mathbf{B}_L \sigma(\mathbf{A}_{L-1} g_{L-2}(\mathbf{x})). \end{aligned}$$

Now, note that $f_{L,k}(\mathbf{s}) = \mathbf{N}_L f_{L-1,k}(\mathbf{s})$, for all $2 \leq k \leq L-1$, and $f_{L,L}(\mathbf{s}) = \mathbf{N}_L \sigma(\mathbf{s})$. Hence, $\nabla_{\mathbf{s}} f_{L,k}^{\top}(\mathbf{s}) = \mathbf{N}_L \nabla_{\mathbf{s}} f_{L-1,k}^{\top}(\mathbf{s})$, and $\nabla_{\mathbf{s}} f_{L,L}^{\top}(\mathbf{s}) = \mathbf{N}_L \operatorname{diag}(\sigma'(\mathbf{s}))$. Therefore,

$$\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sum_{k=1}^{L-2} \nabla_{\mathbf{s}} f_{L,k+2}^{\top} (\mathbf{N}_{k+1} \mathbf{g}_k(\mathbf{x})) \mathbf{B}_{k+1} \sigma(\mathbf{A}_k \mathbf{g}_{k-1}(\mathbf{x})) + \mathbf{B}_L \sigma(\mathbf{A}_{L-1} g_{L-2}(\mathbf{x})),$$

which completes the proof.

We next prove the claim stated in eq. (44) using induction. For $l = 2$, note that

$$\begin{aligned} \sigma(\mathbf{h}_2) &= \sigma \left(\begin{bmatrix} \mathbf{N}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & \mathbf{C}_2 \end{bmatrix} \sigma \left(\begin{bmatrix} \mathbf{N}_1 \mathbf{x} \\ \mathbf{A}_1 \mathbf{x} \end{bmatrix} \right) \right) = \begin{bmatrix} \sigma(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) \\ \sigma(\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})) \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^p + p(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-1} \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) + \sum_{k=2}^p \binom{p}{k} (\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-k} \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}))^k \\ (\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^p + \sum_{k=1}^p \binom{p}{k} (\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-k} \odot (\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}))^k \end{bmatrix}, \end{aligned}$$

where the final equality follows since $\sigma(x) = x^p$ and from using Binomial theorem to expand the expressions. If we define $p_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = p(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-1} \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}))$ and $q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sum_{k=2}^p \binom{p}{k} (\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-k} \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}))^k$, then $p_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z and $q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity strictly greater than $p+1$. Next, if we define $s_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = (\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^p$ and $r_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sum_{k=1}^p \binom{p}{k} (\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-k} \odot (\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}))^k$, then $s_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z and $r_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity strictly greater than p . Also, since $\sigma'(x) = px^{p-1}$, we have

$$p_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = p(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))^{p-1} \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) = \operatorname{diag}(\sigma'(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}).$$

Hence, the claim stated in eq. (44) is true for $l = 2$. Now, suppose the claim is true for some $2 < l < L-1$. For brevity, let $\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Then,

$$\begin{aligned} \sigma(\mathbf{h}_{l+1}) &= \sigma(\mathbf{W}_{l+1} \sigma(\mathbf{h}_l)) = \sigma \left(\begin{bmatrix} \mathbf{N}_{l+1} & \mathbf{B}_{l+1} \\ \mathbf{A}_{l+1} & \mathbf{C}_{l+1} \end{bmatrix} \begin{bmatrix} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix} \right) \\ &= \sigma \left(\begin{bmatrix} \mathbf{N}_{l+1} & \mathbf{B}_{l+1} \\ \mathbf{A}_{l+1} & \mathbf{C}_{l+1} \end{bmatrix} \begin{bmatrix} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix} \right) \\ &= \sigma \left(\begin{bmatrix} \mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \mathbf{N}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \mathbf{A}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{C}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix} \right) \end{aligned}$$

Let $\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \mathbf{N}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \mathbf{A}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{C}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Then,

$$\begin{aligned} \sigma(\mathbf{h}_{l+1}) &= \begin{bmatrix} \sigma(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \\ \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \end{bmatrix} \\ &= \begin{bmatrix} ((\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^p + p(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-1} \odot \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \\ ((\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^p + \mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-1} \odot \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \hat{r}_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \end{bmatrix}, \end{aligned}$$

where

$$\hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \sum_{k=2}^p \binom{p}{k} (\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-k} \odot \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)^k$$

and

$$r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \sum_{k=1}^p \binom{p}{k} (\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-k} \odot \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)^k.$$

Now, note that $\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than or equal to $p + 1$. Also, $\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than or equal to p . Hence, $\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than p . This in turn implies $r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than p . Also,

$$\mathbf{s}_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = (\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^p = \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))),$$

and $\mathbf{s}_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z .

Similarly, $\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than or equal to $p + 1$. Hence, $\hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than $p + 1$. Next, $\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ can be simplified to get

$$\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \mathbf{N}_{l+1} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{N}_{l+1} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z).$$

Note that $\mathbf{N}_{l+1} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z . Also, $\mathbf{N}_{l+1} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is sum of homogeneous functions with degree of homogeneity greater than $p + 1$. Hence, if we define

$$\begin{aligned} p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= p(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-1} \odot (\mathbf{N}_{l+1} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)), \\ q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= p(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))))^{p-1} \odot (\mathbf{N}_{l+1} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) + \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z), \end{aligned}$$

then

$$\sigma(\mathbf{h}_{l+1}) = \begin{bmatrix} \sigma(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))) + p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))) + s_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix},$$

where $p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p + 1)$ -homogeneous in \mathbf{w}_z and $q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is a sum of homogeneous polynomial functions in \mathbf{w}_z with degree of homogeneity greater than $p + 1$.

Finally, we simplify $p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. For all $l \in [L - 1]$ and $k \leq l$, we have

$$g_l(\mathbf{x}) = \sigma(\mathbf{N}_l \sigma(\mathbf{N}_{l-1} \dots \sigma(\mathbf{N}_1 \mathbf{x}))) = f_{l,k}(\mathbf{N}_{k-1} g_{k-2}(\mathbf{x})). \quad (45)$$

Since $f_{l+1,k}(\mathbf{s}) = \sigma(\mathbf{N}_{l+1} f_{l,k}(\mathbf{s}))$, we get

$$\nabla_{\mathbf{s}} f_{l+1,k}^{\top}(\mathbf{s}) = \text{diag}(\sigma'(\mathbf{N}_{l+1} f_{l,k}(\mathbf{s}))) \mathbf{N}_{l+1} \nabla_{\mathbf{s}} f_{l,k}^{\top}(\mathbf{s}). \quad (46)$$

Hence,

$$\begin{aligned}
p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= p(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots\sigma(\mathbf{N}_1\mathbf{x}))))^{p-1} \odot (\mathbf{N}_{l+1}p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1}s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \\
&= \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{N}_{l+1} \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) \\
&\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{N}_{l+1} \text{diag}(\sigma'(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})))\mathbf{B}_l\sigma(\mathbf{A}_{l-1}\mathbf{g}_{l-2}(\mathbf{x})) \\
&\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\
&= \sum_{k=1}^{l-2} \text{diag}(\sigma'(\mathbf{N}_{l+1}f_{l,k+2}(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))))\mathbf{N}_{l+1} \nabla_{\mathbf{s}} f_{l,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) \\
&\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x}))))\mathbf{N}_{l+1} \text{diag}(\sigma'(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})))\mathbf{B}_l\sigma(\mathbf{A}_{l-1}\mathbf{g}_{l-2}(\mathbf{x})) \\
&\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\
&= \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l+1,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) + \nabla_{\mathbf{s}} f_{l+1,l+1}^\top(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})) \\
&\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\
&= \sum_{k=1}^{l-1} \nabla_{\mathbf{s}} f_{l+1,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})),
\end{aligned}$$

where the second equality follows from the definition of $p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. The third and fourth equality uses eq. (45) and eq. (46), respectively. This completes the proof for the second case.

We next move towards proving the first case.

Case 1 ($\alpha \neq 1$ and $p \geq 4$): For all $l \in [L-1]$, define

$$\phi_0 = \mathbf{x}, \mathbf{h}_1 = \mathbf{W}_1\mathbf{x}, \phi_l = \sigma(\mathbf{h}_1), \mathbf{h}_{l+1} = \mathbf{W}_{l+1}\phi_l.$$

For $1 \leq k \leq l \leq L-1$, define

$$f_{l,k}(\mathbf{s}) = \sigma(\mathbf{N}_l\sigma(\mathbf{N}_{l-1}\sigma(\dots\sigma(\mathbf{N}_k\sigma(\mathbf{s}))))).$$

For all $2 \leq l \leq L-1$, we claim that

$$\sigma(\mathbf{h}_l) = \begin{bmatrix} \sigma(\mathbf{N}_l\sigma(\mathbf{N}_{l-1}\sigma(\dots\mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}, \quad (47)$$

where $p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ are a vector-valued homogeneous functions in \mathbf{w}_z with degree of homogeneity $p+1$ and p respectively. More specifically,

$$\begin{aligned}
p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) + \text{diag}(\sigma'(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})))\mathbf{B}_l\sigma(\mathbf{A}_{l-1}\mathbf{g}_{l-2}(\mathbf{x})), \\
s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})).
\end{aligned}$$

Furthermore, if $\|\mathbf{w}_z\|_2 = O(\delta)$, then

$$\begin{aligned}
\|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &= O(\delta^{2p+1}), \|\nabla_{\mathbf{w}_z} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}), \|\nabla_{\mathbf{w}_n} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1}), \text{ and} \\
\|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &= O(\delta^{2p}), \|\nabla_{\mathbf{w}_z} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1}), \|\nabla_{\mathbf{w}_n} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}).
\end{aligned}$$

Now, if the claim stated in eq. (47) is true, then we have

$$\begin{aligned}
\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \mathbf{W}_L \sigma(\mathbf{h}_{L-1}) \\
&= [\mathbf{N}_L \quad \mathbf{B}_L] \begin{bmatrix} \sigma(\mathbf{N}_{L-1} \sigma(\mathbf{N}_{L-2} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix} \\
&= \mathbf{N}_L \sigma(\mathbf{N}_{L-1} \sigma(\mathbf{N}_{L-2} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\
&\quad + \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\
&= \mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) + \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\
&\quad + \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z),
\end{aligned}$$

where the final equality follows since $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{0}) = \mathbf{N}_L \sigma(\mathbf{N}_{L-1} \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))$. Now, define $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Since \mathbf{N}_L does not belong to \mathbf{w}_z and $p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z , $\mathbf{N}_L p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z . Also, \mathbf{B}_L is 1-homogeneous in \mathbf{w}_z and $s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z , which implies $\mathbf{B}_L s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z . Hence, $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z .

We next derive bounds on the remainder terms and its gradient. If $\|\mathbf{w}_z\|_2 = O(\delta)$, then

$$\begin{aligned}
\|\mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&\leq \|\mathbf{N}_L\|_2 \|q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_L\|_2 \|r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&= O(\delta^{2p+1}) + O(\delta)O(\delta^{2p}) = O(\delta^{2p+1}),
\end{aligned}$$

where the penultimate equality follows since $\|q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$, $\|r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$ and $\|\mathbf{B}_L\|_2 = O(\delta)$. Next,

$$\begin{aligned}
&\|\nabla_{\mathbf{w}_z} \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \nabla_{\mathbf{w}_z} \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&\leq \|\mathbf{N}_L \nabla_{\mathbf{w}_z} q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} \mathbf{B}_L\|_2 \|r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_L\|_2 \|\nabla_{\mathbf{w}_z} r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&= O(\delta^{2p}) + O(1)O(\delta^{2p}) + O(\delta)O(\delta^{2p-1}) = O(\delta^{2p}),
\end{aligned}$$

where the penultimate equality follows since $\|\nabla_{\mathbf{w}_z} q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, $\|\nabla_{\mathbf{w}_z} r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1})$, $\|r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$ and $\|\mathbf{B}_L\|_2 = O(\delta)$. Also, since \mathbf{B}_L is 1-homogeneous in \mathbf{w}_z , we have $\|\nabla_{\mathbf{w}_z} \mathbf{B}_L\|_2 = O(1)$. Next,

$$\begin{aligned}
&\|\nabla_{\mathbf{w}_n} \mathbf{N}_L q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \nabla_{\mathbf{w}_n} \mathbf{B}_L r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&\leq \|\mathbf{N}_L\|_2 \|q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{N}_L\|_2 \|\nabla_{\mathbf{w}_n} q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_L\|_2 \|\nabla_{\mathbf{w}_n} r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\
&= O(\delta^{2p+1}) + O(\delta^{2p+1}) + O(\delta)O(\delta^{2p}) = O(\delta^{2p+1}),
\end{aligned}$$

where the penultimate equality follows since $\|\nabla_{\mathbf{w}_n} q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$, $\|\nabla_{\mathbf{w}_n} r_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, $\|q_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$ and $\|\mathbf{B}_L\|_2 = O(\delta)$.

Finally, using the definition of $p_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_{L-1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, we get

$$\begin{aligned}
&\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\
&= \sum_{k=1}^{L-3} \mathbf{N}_L \nabla_{\mathbf{s}} f_{L-1, k+2}^{\top} (\mathbf{N}_{k+1} \mathbf{g}_k(\mathbf{x})) \mathbf{B}_{k+1} \sigma(\mathbf{A}_k \mathbf{g}_{k-1}(\mathbf{x})) + \mathbf{N}_L \text{diag}(\sigma'(\mathbf{N}_{L-1} \mathbf{g}_{L-2}(\mathbf{x}))) \mathbf{B}_{L-1} \sigma(\mathbf{A}_{L-2} \mathbf{g}_{L-3}(\mathbf{x})) \\
&\quad + \mathbf{B}_L \sigma(\mathbf{A}_{L-1} \mathbf{g}_{L-2}(\mathbf{x})).
\end{aligned}$$

Now, note that $f_{L,k}(\mathbf{s}) = \mathbf{N}_L f_{L-1,k}(\mathbf{s})$, for all $2 \leq k \leq L-1$, and $f_{L,L}(\mathbf{s}) = \mathbf{N}_L \sigma(\mathbf{s})$. Hence, $\nabla_{\mathbf{s}} f_{L,k}^{\top}(\mathbf{s}) = \mathbf{N}_L \nabla_{\mathbf{s}} f_{L-1,k}^{\top}(\mathbf{s})$, and $\nabla_{\mathbf{s}} f_{L,L}^{\top}(\mathbf{s}) = \mathbf{N}_L \text{diag}(\sigma'(\mathbf{s}))$. Therefore,

$$\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sum_{k=1}^{L-2} \nabla_{\mathbf{s}} f_{L,k+2}^{\top} (\mathbf{N}_{k+1} \mathbf{g}_k(\mathbf{x})) \mathbf{B}_{k+1} \sigma(\mathbf{A}_k \mathbf{g}_{k-1}(\mathbf{x})) + \mathbf{B}_L \sigma(\mathbf{A}_{L-1} \mathbf{g}_{L-2}(\mathbf{x})),$$

which completes the proof.

We next prove the claim stated in eq. (47) using induction. For $l = 2$, note that

$$\sigma(\mathbf{h}_2) = \sigma\left(\begin{bmatrix} \mathbf{N}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & \mathbf{C}_2 \end{bmatrix} \sigma\left(\begin{bmatrix} \mathbf{N}_1 \mathbf{x} \\ \mathbf{A}_1 \mathbf{x} \end{bmatrix}\right)\right) = \sigma\left(\begin{bmatrix} \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \\ \mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \end{bmatrix}\right).$$

From Taylor's theorem, we know

$$\sigma(x+h) = \sigma(x) + \int_0^h \sigma'(x+t)dt, \quad \sigma(x+h) = \sigma(x) + \sigma'(x)h + \int_0^h \sigma''(x+t)(h-t)dt.$$

Hence,

$$\begin{aligned} \sigma(\mathbf{h}_2) &= \sigma\left(\begin{bmatrix} \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \\ \mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}) \end{bmatrix}\right) \\ &= \begin{bmatrix} \sigma(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})) + \sigma'(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})) \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})) + q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \sigma(\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x})) + r_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned} q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \int_0^{\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})} \sigma''(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{t}) \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}) - \mathbf{t}) d\mathbf{t}, \\ r_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \int_0^{\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})} \sigma'(\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}) + \mathbf{t}) dt. \end{aligned}$$

Define $p_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sigma'(\mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})) \odot (\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x}))$ and $s_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sigma(\mathbf{A}_2 \sigma(\mathbf{N}_1 \mathbf{x}))$. Then, $p_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ are $(p+1)$ -homogeneous and p -homogeneous in \mathbf{w}_z , respectively

We next derive bounds on $q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, $r_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and their gradients. Since $\|\mathbf{w}_z\|_2 = O(\delta)$, then $\|\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = O(\delta^{p+1})$, $\|\nabla_{\mathbf{w}_n} \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = 0$, and $\|\nabla_{\mathbf{w}_z} \mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = O(\delta^p)$, where the last equality is true since $\mathbf{B}_2 \sigma(\mathbf{A}_1 \mathbf{x})$ is $(p+1)$ -homogeneous in \mathbf{w}_z . Hence, from Lemma 18, we get

$$\|q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1}), \quad \|\nabla_{\mathbf{w}_z} q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}), \quad \text{and } \|\nabla_{\mathbf{w}_n} q_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1}).$$

We also have $\|\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = O(\delta^{p+1})$, $\|\nabla_{\mathbf{w}_n} \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = 0$, and $\|\nabla_{\mathbf{w}_z} \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x})\|_2 = O(\delta^p)$. Hence, from Lemma 18, we get

$$\|\mathbf{r}_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}), \quad \|\nabla_{\mathbf{w}_z} \mathbf{r}_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1}), \quad \text{and } \|\nabla_{\mathbf{w}_n} \mathbf{r}_2(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}).$$

Therefore, the claim is true for $l = 2$.

Now, suppose the claim is true for some $2 < l < L-1$. For brevity, let $\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Then,

$$\begin{aligned} \sigma(\mathbf{h}_{l+1}) &= \sigma(\mathbf{W}_{l+1} \sigma(\mathbf{h}_l)) = \sigma\left(\begin{bmatrix} \mathbf{N}_{l+1} & \mathbf{B}_{l+1} \\ \mathbf{A}_{l+1} & \mathbf{C}_{l+1} \end{bmatrix} \begin{bmatrix} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}\right) \\ &= \sigma\left(\begin{bmatrix} \mathbf{N}_{l+1} & \mathbf{B}_{l+1} \\ \mathbf{A}_{l+1} & \mathbf{C}_{l+1} \end{bmatrix} \begin{bmatrix} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}\right) \\ &= \sigma\left(\begin{bmatrix} \mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \mathbf{N}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \mathbf{A}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{C}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}\right) \end{aligned}$$

Let $\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \mathbf{N}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) := \mathbf{A}_{l+1} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{C}_{l+1} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Then, using Taylor's theorem,

$$\begin{aligned} \sigma(\mathbf{h}_{l+1}) &= \begin{bmatrix} \sigma(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \\ \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x}))) + \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \end{bmatrix} \\ &= \begin{bmatrix} \sigma(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))) + \sigma'(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))) \odot \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \sigma(\mathbf{N}_1 \mathbf{x})))) + r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned}\hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \int_0^{\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + \mathbf{t}) \odot (\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{t}) d\mathbf{t}, \\ r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \int_0^{\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma'(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + \mathbf{t}) dt.\end{aligned}$$

Define

$$\begin{aligned}p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sigma'(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) \odot (\mathbf{N}_{l+1} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)), \\ q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sigma'(\mathbf{N}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) \odot (\mathbf{N}_{l+1} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) + \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z),\end{aligned}$$

and

$$s_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \sigma(\mathbf{A}_{l+1} \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))).$$

Note that $\mathbf{N}_{l+1} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z , therefore, $p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is and are $(p+1)$ -homogeneous in \mathbf{w}_z . Also, $s_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z .

We next derive bounds on $q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, $r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and their gradients. Since $p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z , $\|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$, $\|\nabla_{\mathbf{w}_z} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, and $\|\nabla_{\mathbf{w}_n} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$, we have

$$\begin{aligned}\|\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1}), \\ \|\nabla_{\mathbf{w}_z} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\nabla_{\mathbf{w}_z} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p), \\ \|\nabla_{\mathbf{w}_n} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\nabla_{\mathbf{w}_n} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_n} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1}),\end{aligned}$$

where the final equality is true since, from Lemma 15, $\nabla_{\mathbf{w}_n} p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ $(p+1)$ -homogeneous in \mathbf{w}_z . Next, since $s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is p -homogeneous in \mathbf{w}_z , $\|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, $\|\nabla_{\mathbf{w}_z} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1})$, and $\|\nabla_{\mathbf{w}_n} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, we have

$$\begin{aligned}\|\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p), \\ \|\nabla_{\mathbf{w}_z} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\nabla_{\mathbf{w}_z} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p-1}), \\ \|\nabla_{\mathbf{w}_n} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\nabla_{\mathbf{w}_n} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_n} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p),\end{aligned}$$

where the final equality is true since, from Lemma 15, $\nabla_{\mathbf{w}_n} s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ p -homogeneous in \mathbf{w}_z . Using the above two set of inequalities, we get

$$\begin{aligned}\|\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{N}_{l+1}\|_2 \|\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta^{p+1}) + O(\delta)O(\delta^p) = O(\delta^{p+1}), \\ \|\nabla_{\mathbf{w}_z} \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{N}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} \mathbf{B}_{l+1}\|_2 \|\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta^p) + O(\delta)O(\delta^{p-1}) + O(1)O(\delta^p) = O(\delta^p), \\ \|\nabla_{\mathbf{w}_n} \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{N}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_n} \mathbf{N}_{l+1}\|_2 \|\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta^{p+1}) + O(\delta^{p+1}) + O(\delta)O(\delta^p) = O(\delta^{p+1}),\end{aligned}$$

and

$$\begin{aligned}\|\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{A}_{l+1}\|_2 \|\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{C}_{l+1}\|_2 \|\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta)O(\delta^{p+1}) + O(\delta)O(\delta^p) = O(\delta^{p+1}), \\ \|\nabla_{\mathbf{w}_z} \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{A}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} \mathbf{A}_{l+1}\|_2 \|\tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &\quad + \|\mathbf{C}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} \mathbf{C}_{l+1}\|_2 \|\tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta)O(\delta^p) + O(1)O(\delta^{p+1}) + O(\delta)O(\delta^{p-1}) + O(1)O(\delta^p) = O(\delta^p), \\ \|\nabla_{\mathbf{w}_n} \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\mathbf{A}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} \tilde{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{C}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} \tilde{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \\ &= O(\delta)O(\delta^{p+1}) + O(\delta)O(\delta^p) = O(\delta^{p+1}).\end{aligned}$$

Now, since $\|\hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$, $\|\nabla_{\mathbf{w}_z} \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p)$, $\|\nabla_{\mathbf{w}_n} \hat{p}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$, from Lemma 18 we get

$$\|\hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+2}), \|\nabla_{\mathbf{w}_z} \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1}), \|\nabla_{\mathbf{w}_n} \hat{q}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+2}).$$

For the sake of brevity, define $\zeta(\mathbf{x}; \mathbf{w}_n) := \sigma'(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots\mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x}))))$. Then,

$$\begin{aligned} \|q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\zeta(\mathbf{x}; \mathbf{w}_n)\|_2 (\|\mathbf{N}_{l+1}\|_2 \|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2) + \|\hat{q}_l(\mathbf{x}; \mathbf{w}_n)\|_2 \\ &= O(\delta^{2p+1}) + O(\delta)O(\delta^{2p}) + O(\delta^{2p+2}) = O(\delta^{2p+1}), \\ \|\nabla_{\mathbf{w}_z} q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\zeta(\mathbf{x}; \mathbf{w}_n)\|_2 \|\mathbf{N}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_z} \hat{q}_l(\mathbf{x}; \mathbf{w}_n)\|_2 \\ &\quad + \|\zeta(\mathbf{x}; \mathbf{w}_n)\|_2 (\|\nabla_{\mathbf{w}_z} \mathbf{B}_{l+1}\|_2 \|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|\nabla_{\mathbf{w}_z} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2) \\ &= O(\delta^{2p}) + O(\delta^{2p+1}) + O(1)O(\delta^{2p}) + O(\delta)O(\delta^{2p-1}) = O(\delta^{2p}), \\ \|\nabla_{\mathbf{w}_n} q_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \|\nabla_{\mathbf{w}_n} \zeta(\mathbf{x}; \mathbf{w}_n)\|_2 (\|\mathbf{N}_{l+1}\|_2 \|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{B}_{l+1}\|_2 \|r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2) \\ &\quad + \|\zeta(\mathbf{x}; \mathbf{w}_n)\|_2 (\|\nabla_{\mathbf{w}_n} \mathbf{N}_{l+1}\|_2 \|q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\mathbf{N}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} q_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2) \\ &\quad + \|\zeta(\mathbf{x}; \mathbf{w}_n)\|_2 \|\mathbf{B}_{l+1}\|_2 \|\nabla_{\mathbf{w}_n} r_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + \|\nabla_{\mathbf{w}_n} \hat{q}_l(\mathbf{x}; \mathbf{w}_n)\|_2 \\ &= O(\delta^{2p+1}) + O(\delta)O(\delta^{2p}) + O(\delta^{2p+1}) + O(\delta^{2p+1}) + O(\delta)O(\delta^{2p}) + O(\delta^{2p+2}) \\ &= O(\delta^{2p+1}). \end{aligned}$$

Next, since $\|\hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$, $\|\nabla_{\mathbf{w}_z} \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p)$, $\|\nabla_{\mathbf{w}_n} \hat{s}_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$, from Lemma 18 we get

$$\|r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}), \|\nabla_{\mathbf{w}_z} r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1}), \|\nabla_{\mathbf{w}_n} r_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}).$$

Finally, we simplify $p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Here as well, for all $l \in [L-1]$ and $k \leq l$, we have

$$g_l(\mathbf{x}) = \sigma(\mathbf{N}_l\sigma(\mathbf{N}_{l-1}\dots\sigma(\mathbf{N}_1\mathbf{x}))) = f_{l,k}(\mathbf{N}_{k-1}g_{k-2}(\mathbf{x})). \quad (48)$$

Since $f_{l+1,k}(\mathbf{s}) = \sigma(\mathbf{N}_{l+1}f_{l,k}(\mathbf{s}))$, we get

$$\nabla_{\mathbf{s}} f_{l+1,k}(\mathbf{s}) = \text{diag}(\sigma'(\mathbf{N}_{l+1}f_{l,k}(\mathbf{s})))\mathbf{N}_{l+1}\nabla_{\mathbf{s}} f_{l,k}(\mathbf{s}). \quad (49)$$

Hence,

$$\begin{aligned} p_{l+1}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \sigma'(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots\sigma(\mathbf{N}_1\mathbf{x})))) \odot (\mathbf{N}_{l+1}p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) + \mathbf{B}_{l+1}s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \\ &= \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{N}_{l+1} \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) \\ &\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{N}_{l+1} \text{diag}(\sigma'(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})))\mathbf{B}_l\sigma(\mathbf{A}_{l-1}\mathbf{g}_{l-2}(\mathbf{x})) \\ &\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\ &= \sum_{k=1}^{l-2} \text{diag}(\sigma'(\mathbf{N}_{l+1}f_{l,k+2}(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))))\mathbf{N}_{l+1}\nabla_{\mathbf{s}} f_{l,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) \\ &\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x}))))\mathbf{N}_{l+1} \text{diag}(\sigma'(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})))\mathbf{B}_l\sigma(\mathbf{A}_{l-1}\mathbf{g}_{l-2}(\mathbf{x})) \\ &\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\ &= \sum_{k=1}^{l-2} \nabla_{\mathbf{s}} f_{l+1,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) + \nabla_{\mathbf{s}} f_{l+1,l+1}^\top(\mathbf{N}_l\mathbf{g}_{l-1}(\mathbf{x})) \\ &\quad + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})) \\ &= \sum_{k=1}^{l-1} \nabla_{\mathbf{s}} f_{l+1,k+2}^\top(\mathbf{N}_{k+1}\mathbf{g}_k(\mathbf{x}))\mathbf{B}_{k+1}\sigma(\mathbf{A}_k\mathbf{g}_{k-1}(\mathbf{x})) + \text{diag}(\sigma'(\mathbf{N}_{l+1}g_l(\mathbf{x})))\mathbf{B}_{l+1}\sigma(\mathbf{A}_l g_{l-1}(\mathbf{x})), \end{aligned}$$

where the second equality follows from the definition of $p_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $s_l(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. The third and fourth equality uses eq. (48) and eq. (49), respectively. This completes the proof for the second case. \square

Lemma 18. Consider the setting of Lemma 8, and suppose \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$. For any $2 \leq l \leq L-1$, let

$$a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + \mathbf{t}) \odot (c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{t}) d\mathbf{t},$$

$$b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) = \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma'(\mathbf{A}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + \mathbf{t}) d\mathbf{t},$$

where $\|c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$, $\|\nabla_{\mathbf{w}_z} c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^p)$ and $\|\nabla_{\mathbf{w}_n} c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$. Then,

- (i) $\|a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+2})$, $\|\nabla_{\mathbf{w}_z} a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+1})$ and $\|\nabla_{\mathbf{w}_n} a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+2})$,
- (ii) $\|b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$, $\|\nabla_{\mathbf{w}_z} b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p-1})$ and $\|\nabla_{\mathbf{w}_n} b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p})$.

Proof. Since \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, there exists a large enough constant $C > 0$ such that

$$\begin{aligned} \|a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \left\| \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + \mathbf{t}) \odot (c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{t}) d\mathbf{t} \right\|_2 \\ &\leq C \left\| \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} (c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - \mathbf{t}) d\mathbf{t} \right\|_2 \\ &= \frac{C}{2} \|c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \odot c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p+2}). \end{aligned}$$

Next, since $\|\mathbf{A}_{l+1}\|_2 = O(\delta)$, $\|c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{p+1})$ and $\sigma'(x)$ is $(p-1)$ -homogeneous, there exists a large enough constant $C > 0$ such that

$$\begin{aligned} \|b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 &\leq \left\| \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma'(\mathbf{A}_{l+1}\sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + \mathbf{t}) d\mathbf{t} \right\|_2 \\ &\leq C\delta^{p-1} \left\| \int_0^{c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} d\mathbf{t} \right\|_2 \\ &= C\delta^{p-1} \|c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 = O(\delta^{2p}). \end{aligned}$$

Using Leibniz's rule, for any $\mathcal{Q}: \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}$ and $m: \mathbb{R}^d \rightarrow \mathbb{R}$, where $\mathcal{Q}(\mathbf{s}, t)$, $\nabla_{\mathbf{s}} \mathcal{Q}(\mathbf{s}, t)$, $m(\mathbf{s})$ and $\nabla_{\mathbf{s}} m(\mathbf{s})$ are continuous in \mathbf{s} , we have

$$\nabla_{\mathbf{s}} \left(\int_0^{m(\mathbf{s})} \mathcal{Q}(\mathbf{s}, t) dt \right) = \mathcal{Q}(\mathbf{s}, m(\mathbf{s})) \nabla_{\mathbf{s}} m(\mathbf{s}) + \int_0^{m(\mathbf{s})} \nabla_{\mathbf{s}} \mathcal{Q}(\mathbf{s}, t) dt.$$

Let $a_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ denote the i th entry of $a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ and $c(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, respectively. Since $p \geq 4$, $\sigma''(x)$ is continuous and has continuous derivatives. Therefore, using Leibniz's rule, we get

$$\begin{aligned} &\nabla_{\mathbf{w}_z} a_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &= \nabla_{\mathbf{w}_z} \left(\int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + t)(c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - t) dt \right) \\ &= \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + t)(c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + t) \nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt \\ &= \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l\sigma(\dots \mathbf{N}_2\sigma(\mathbf{N}_1\mathbf{x})))) + t) \nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt. \end{aligned}$$

Since \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, there exists a large enough constant $C > 0$ such that

$$\|\nabla_{\mathbf{w}_z} a_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \leq C|c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)|\|\nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2,$$

which implies

$$\|\nabla_{\mathbf{w}_z} a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_F = O(\delta^{p+1}\delta^p) = O(\delta^{2p+1}).$$

Let $b_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ denote the i th entry of $b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$. Then,

$$\begin{aligned} \nabla_{\mathbf{w}_z} b_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \nabla_{\mathbf{w}_z} \left(\int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) dt \right) \\ &= \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \nabla_{\mathbf{w}_z} \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) dt. \end{aligned}$$

Since \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, there exists a large enough constant $C > 0$ such that

$$\|\nabla_{\mathbf{w}_z} b_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \leq C\|\mathbf{A}_{l+1}[i, :]\|_2^{p-1}\|\nabla_{\mathbf{w}_z} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + C\|\mathbf{A}_{l+1}[i, :]\|_2^{p-2}|c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)|,$$

which implies

$$\|\nabla_{\mathbf{w}_z} b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_F = O(\delta^{p-1}\delta^p) + O(\delta^{p-2}\delta^{p+1}) = O(\delta^{2p-1}).$$

Next,

$$\begin{aligned} \nabla_{\mathbf{w}_n} a_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \nabla_{\mathbf{w}_n} \left(\int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t)(c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - t) dt \right) \\ &= \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x})))) + t)(c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) - c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) \nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \nabla_{\mathbf{w}_n} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt \\ &= \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) \nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \nabla_{\mathbf{w}_n} \sigma''(\mathbf{N}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) dt. \end{aligned}$$

Since \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, there exists a large enough constant $C > 0$ such that

$$\|\nabla_{\mathbf{w}_n} a_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \leq C|c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)|\|\nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + C|c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)|^2,$$

which implies

$$\|\nabla_{\mathbf{w}_n} a(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_F = O(\delta^{p+1}\delta^{p+1}) + O(\delta^{2p+2}) = O(\delta^{2p+2}).$$

Next,

$$\begin{aligned} \nabla_{\mathbf{w}_n} b_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) &= \nabla_{\mathbf{w}_n} \left(\int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) dt \right) \\ &= \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)) \nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z) \\ &\quad + \int_0^{c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)} \nabla_{\mathbf{w}_n} \sigma'(\mathbf{A}_{l+1}[i, :] \sigma(\mathbf{N}_l \sigma(\dots \mathbf{N}_2 \sigma(\mathbf{N}_1 \mathbf{x}))) + t) dt. \end{aligned}$$

Since \mathbf{w}_n is fixed and $\|\mathbf{w}_z\|_2 = O(\delta)$, there exists a large enough constant $C > 0$ such that

$$\|\nabla_{\mathbf{w}_n} b_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 \leq C \|\mathbf{A}_{l+1}[i, :]\|_2^{p-1} \|\nabla_{\mathbf{w}_n} c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_2 + C \|\mathbf{A}_{l+1}[i, :]\|_2^{p-1} |c_i(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)|,$$

which implies

$$\|\nabla_{\mathbf{w}_n} b(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)\|_F = O(\delta^{p-1} \delta^{p+1}) + O(\delta^{p-1} \delta^{p+1}) = O(\delta^{2p}).$$

□

B.3 Proof of Lemma 9

We will make repeated use of the following lemma to prove Lemma 9.

Lemma 19. Let $h(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n \mathbf{q}_i^\top \mathbf{b} \sigma(\mathbf{a}^\top \mathbf{r}_i)$, where $\mathbf{a} \in \mathbb{R}^{d_1}$, $\mathbf{b} \in \mathbb{R}^{d_2}$ and $\sigma(x) = \max(x, \alpha x)^p$, for some $p \in \mathbb{N}, p \geq 1$ and $\alpha \in \mathbb{R}$. Suppose there exists $\lambda > 0$ and $(\mathbf{a}_*, \mathbf{b}_*)$ such that

$$\nabla_{\mathbf{a}} h(\mathbf{a}_*, \mathbf{b}_*) = \lambda \mathbf{a}_*, \quad \nabla_{\mathbf{b}} h(\mathbf{a}_*, \mathbf{b}_*) = \lambda \mathbf{b}_*.$$

Then, $p \|\mathbf{b}_*\|_2^2 = \|\mathbf{a}_*\|_2^2$.

Proof. Since $h(\mathbf{a}, \mathbf{b})$ is 1-homogeneous in \mathbf{b} and p -homogeneous in \mathbf{a} , we get

$$\begin{aligned} \lambda \mathbf{a}_*^\top \mathbf{a}_* &= \mathbf{a}_*^\top \nabla_{\mathbf{a}} h(\mathbf{a}_*, \mathbf{b}_*) = ph(\mathbf{a}_*, \mathbf{b}_*), \\ \lambda \mathbf{b}_*^\top \mathbf{b}_* &= \mathbf{b}_*^\top \nabla_{\mathbf{b}} h(\mathbf{a}_*, \mathbf{b}_*) = h(\mathbf{a}_*, \mathbf{b}_*). \end{aligned}$$

From the above equation, we get $p \|\mathbf{b}_*\|_2^2 = \|\mathbf{a}_*\|_2^2$. □

Proof of Lemma 9: Since $\bar{\mathbf{w}}_z$ is a KKT point of

$$\max_{\|\mathbf{w}_z\|_2^2=1} \mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z),$$

there exist a scalar λ such that

$$\nabla_{\mathbf{w}_z} (\mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \bar{\mathbf{w}}_z)) = \lambda \bar{\mathbf{w}}_z. \quad (50)$$

Since $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ is $(p+1)$ -homogeneous in \mathbf{w}_z , from (Kumar & Haupt, 2025a, Lemma 11) we get

$$\lambda = (p+1) \mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \bar{\mathbf{w}}_z).$$

Since $\bar{\mathbf{w}}_z$ is a positive KKT point, we have $\mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \bar{\mathbf{w}}_z) > 0$, implying $\lambda > 0$.

Next, as $\{\mathbf{C}_l\}_{l=2}^{L-1}$ does not appear in the expression of $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$, from eq. (50), we get

$$\mathbf{0} = \lambda \bar{\mathbf{C}}_l, \quad \text{for all } 2 \leq l \leq L-1,$$

which implies $\bar{\mathbf{C}}_l = \mathbf{0}$, for all $2 \leq l \leq L-1$. Now, from the discussion in Section 3.3, we know

$$\begin{aligned} \mathbf{p}^\top \mathcal{H}_1(\mathbf{X}; \mathbf{w}_n, \mathbf{w}_z) &= \sum_{l=1}^{L-2} \sum_{j=1}^{\Delta_l} \sum_{i=1}^n p_i \nabla_{\mathbf{s}} f_{L,l+2}^\top (\mathbf{N}_{l+1} \mathbf{g}_l(\mathbf{x}_i)) \mathbf{B}_{l+1}[:, j] \sigma(\mathbf{A}_l[j, :] \mathbf{g}_{l-1}(\mathbf{x}_i)) \\ &\quad + \sum_{j=1}^{\Delta_{L-1}} \sum_{i=1}^n p_i \mathbf{B}_L[:, j] \sigma(\mathbf{A}_{L-1}[j, :] g_{L-2}(\mathbf{x}_i)), \end{aligned}$$

where $\Delta_l = k_l - p_l$, for all $l \in [L-1]$. For any $1 \leq l \leq L-2$ and $1 \leq j \leq \Delta_l$, $(\mathbf{B}_{l+1}[:, j], \mathbf{A}_l[j, :])$ only appears in the following term:

$$\sum_{i=1}^n p_i \nabla_{\mathbf{s}} f_{L,l+2}^\top (\mathbf{N}_{l+1} \mathbf{g}_l(\mathbf{x}_i)) \mathbf{B}_{l+1}[:, j] \sigma(\mathbf{A}_l[j, :] \mathbf{g}_{l-1}(\mathbf{x}_i)).$$

Hence, using the above fact, eq. (50) and Lemma 19, we have

$$p \|\overline{\mathbf{B}}_{l+1}[:, j]\|_2^2 = \|\overline{\mathbf{A}}_l[j, :]\|_2^2.$$

Similarly, for any $1 \leq j \leq \Delta_{L-1}$, $(\mathbf{B}_L[:, j], \mathbf{A}_{L-1}[j, :])$ only appears in the following term:

$$\sum_{i=1}^n p_i \mathbf{B}_L[:, j] \sigma(\mathbf{A}_{L-1}[j, :] g_{L-2}(\mathbf{x}_i)).$$

Hence, again using the above fact, eq. (50) and Lemma 19, we have

$$p \|\overline{\mathbf{B}}_L[:, j]\|_2^2 = \|\overline{\mathbf{A}}_{L-1}[j, :]\|_2^2.$$

This completes the proof. \square

C Experimental Details

This section outlines the implementation and hyperparameter details for the experiments in this paper, all conducted using PyTorch (Paszke et al., 2019).

C.1 Non-linear Sparse Functions

C.1.1 Hypersphere

We train three-layer fully connected neural networks to for both the NP algorithm and gradient descent. The specific hyperparameters for each method are as follows:

Gradient Descent. The network contains 50 neurons per layer and is trained using full-batch gradient descent. The initial weights were sampled from the hypersphere of radius 0.01. Training continued until the training error dropped below 0.001 or until a maximum of 3×10^6 iterations. We evaluated three learning rates: $\{0.05, 0.01, 0.005\}$ for $f_1(\mathbf{x})$, and $\{0.005, 0.002, 0.001\}$ for $f_2(\mathbf{x})$. We report the results corresponding to the lowest test error.

Neuron Pursuit. We run the NP algorithm until the training error drops below 0.001 or until 31 iterations were completed. In each iteration, the constrained NCF is maximized using projected gradient ascent with step-size 0.2 for 2500 iterations, where the initial weights are sampled uniformly from the unit-norm sphere. We use $H = 10$ different random initializations to identify the most dominant KKT point.

The scalar δ in the first iteration was set to 0.25. If the training loss exceeds the loss at the origin, δ is reduced multiplicatively by a factor of 0.8, and this adjustment is repeated up to four times. This procedure is motivated by the discussion in Section 5.1, which shows that for sufficiently small δ , the loss is smaller than at the origin. For subsequent iterations, we follow a similar procedure. We first set $\delta = 0.01 \|\mathbf{w}\|_2$, where $\|\mathbf{w}\|_2$ denotes the norm of the weights at the end of the previous iteration. If resulting loss exceeds the loss at the end of the previous iteration, δ is reduced multiplicatively by a factor of 0.8. We repeat this procedure up to 10 times.

After each neuron addition, we trained the network using full-batch gradient descent for 70,000 iterations. We used two step-sizes: $\{0.005, 0.002\}$ for $f_1(\mathbf{x})$ and $\{0.002, 0.001\}$ for $f_2(\mathbf{x})$. To report the final training and test errors for NP, we applied the following procedure:

- If multiple step-sizes yield test error < 0.005 , report the result corresponding to the fewest number of iterations.

-
- **Otherwise, if multiple step-sizes achieve training error** < 0.001 , report the one with the lowest test error.
 - **Otherwise**, report the one with lowest training error.

The first criterion favors efficient solutions when test error is small. The next two step safeguards against misleading conclusions when the smallest test error does not coincide with smallest training error. This particularly happens in low-sample regimes where the NP algorithm is not able to achieve small test error. Without this safeguard, one might wrongly infer that NP fails to fit the training data, even when a good fit is possible with a different learning rate.

C.1.2 Hypercube

We train three-layer fully connected neural networks to for both the NP algorithm and gradient descent. The specific hyperparameters for each method are as follows:

Gradient Descent. The network contains 50 neurons per layer and is trained using full-batch gradient descent. The initial weights were sampled from the hypersphere of radius 0.01. Training continued until the training error dropped below 0.001 or until a maximum of 3×10^6 iterations. We evaluated three learning rates: $\{0.05, 0.01, 0.005\}$ for $g_1(\mathbf{x})$, and $\{0.02, 0.01, 0.005\}$ for $g_2(\mathbf{x})$. We report the results corresponding to the lowest test error.

Neuron Pursuit. We run the NP algorithm until the training error drops below 0.001 or until 31 iterations were completed. In each iteration, the constrained NCF is maximized using projected gradient ascent with step-size 0.2 for 2500 iterations, where the initial weights are sampled uniformly from the unit-norm sphere. We use $H = 10$ different random initializations to identify the most dominant KKT point.

The scalar δ is chosen in a similar way as in Appendix C.1.1. After each neuron addition, we trained the network using full-batch gradient descent for 70,000 iterations. We use learning rates $\{0.005, 0.002\}$ for $g_1(\mathbf{x})$ and $\{0.002, 0.001\}$ for $g_2(\mathbf{x})$. To report the final training and test errors for NP, we applied the same procedure as in Appendix C.1.1.

C.1.3 Gaussian

We train four-layer fully connected neural networks to for both the NP algorithm and gradient descent. The specific hyperparameters for each method are as follows:

Gradient Descent. The network contains 50 neurons per layer and is trained using full-batch gradient descent. The initial weights were sampled from the hypersphere of radius 0.1. Training continued until the training error dropped below 0.001 or until a maximum of 3×10^6 iterations. We evaluated three learning rates for both $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$: $\{0.02, 0.01, 0.005\}$. We report the results corresponding to the lowest test error.

Neuron Pursuit. We run the NP algorithm until the training error drops below 0.001 or until 31 iterations were completed. In each iteration, the constrained NCF is maximized using projected gradient ascent with step-size 0.2 for 2500 iterations, where the initial weights are sampled uniformly from the unit-norm sphere. We use $H = 10$ different random initializations to identify the most dominant KKT point.

The scalar δ is chosen in a similar way as in Appendix C.1.1, except that in the first iteration δ is first set to 0.5. After each neuron addition, we trained the network using full-batch gradient descent for 100,000 iterations. However, if the training loss gets less than 0.05, the number of iterations is increased to 300,000, because the optimization gets slower. We use learning rates $\{0.005, 0.002\}$ for both $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$. To report the final training and test errors for NP, we applied the same procedure as in Appendix C.1.1.

C.2 Algorithmic Tasks

C.2.1 Modular addition

We train a two-layer neural network with square activation. We run the NP algorithm until the classification error on the training data becomes 0. In each iteration, the constrained NCF is maximized using projected gradient ascent with step-size 2 for 5000 iterations, where the initial weights are sampled uniformly from the unit-norm sphere. We use $H = 10$ different random initializations to identify the most dominant KKT point.

The scalar δ is chosen in a similar way as in Appendix C.1.1, except that from the second iteration onward, we first set $\delta = 0.05\|\mathbf{w}\|_2$. After each neuron addition, we trained the network using full-batch gradient descent for 100,000 iterations with learning rate 10. Figure 11 depicts the evolution of training and test error, along with the absolute value of the 2D DFT of the learned weights, for two additional independent runs.

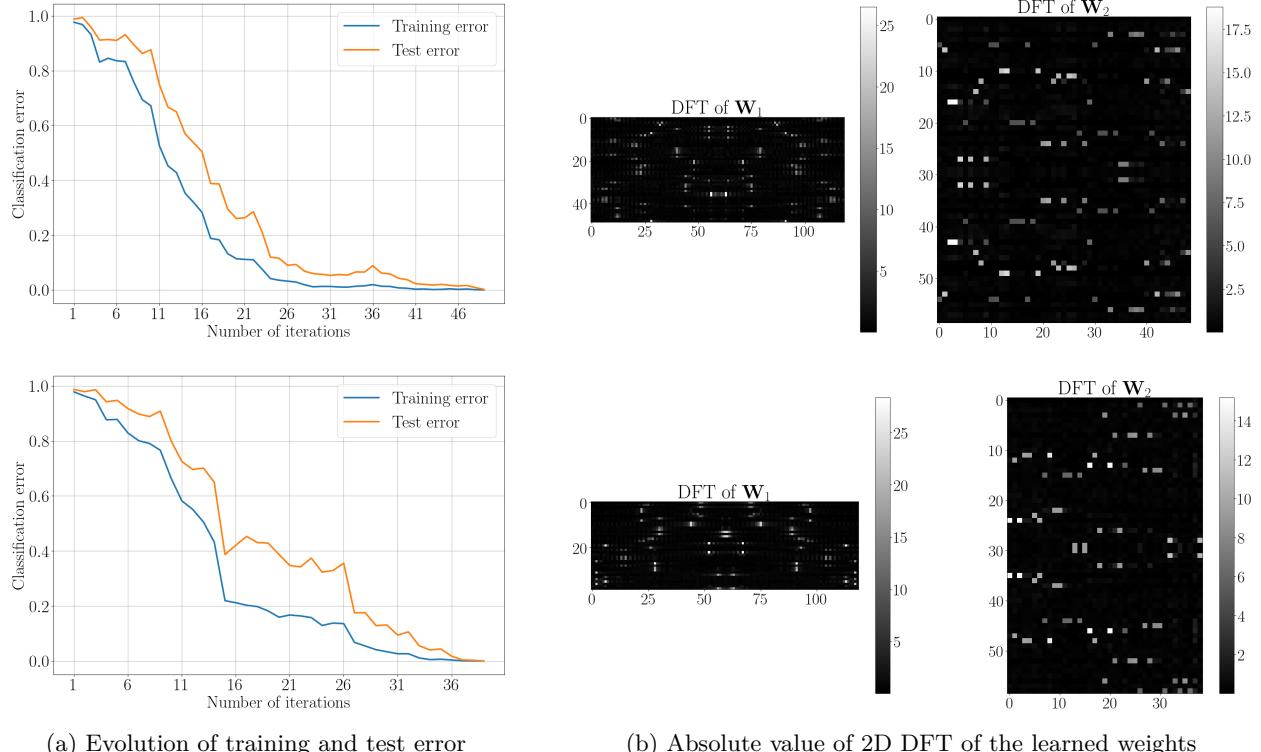


Figure 11: The results from two additional independent runs of learning modular addition using a two-layer neural network with a square activation function, trained via the NP algorithm. Both runs achieve low training and test errors, with the DFT of each row in the first layer and each column in the second layer concentrated around a specific frequency.

C.2.2 PVR

We run the NP algorithm until the training error drops below 0.005. In the first iteration, the constrained NCF is maximized using projected gradient ascent with step-size 0.25 for 6000 iterations. In subsequent iteration, we use step-size 0.5 for 5000 iterations. The initial weights are sampled uniformly from the unit-norm sphere, and we use $H = 10$ different random initializations to identify the most dominant KKT point.

The scalar δ is chosen in a similar way as in Appendix C.1.1, except that in the first iteration δ is first set to 0.5. After each neuron addition, we trained the network using full-batch gradient descent for 500,000 iterations with varying step-sizes. We begin with 0.005 and compute the training loss every 50,000 iterations. If the loss increases, step-size is halved and the weights are reset to their previous state. This halving occurs at most four times. Also, these many iterations are not needed in the initial stages of training, it becomes

important later, where the optimization can be slower to converge. Figure 12 depicts the evolution of training and test error, along with the absolute value of the learned weights, for two additional independent runs.

We introduce one modification to the NP algorithm for the PVR task. After every iteration, we balance the weights so that the incoming and outgoing weights of each hidden neuron have same norm, without changing the network output. To do this, we use the method od Saul (2023), specifically Algorithm 1 with $p = q = 2$. This extra step is motivated by the fact that, under gradient flow with small initialization and ReLU activation, the norm of incoming and outgoing weights of each hidden neuron remains nearly balanced Du et al. (2018). However, when training with gradient descent, especially with large step sizes or large number of iterations, the weights can become unbalanced. We observe this unbalance in NP when applied to the PVR task. Importantly, such unbalance can alter which neurons the NP algorithm chooses to add, as discussed in Appendix D.6. To prevent this behavior, the weights are rebalanced.

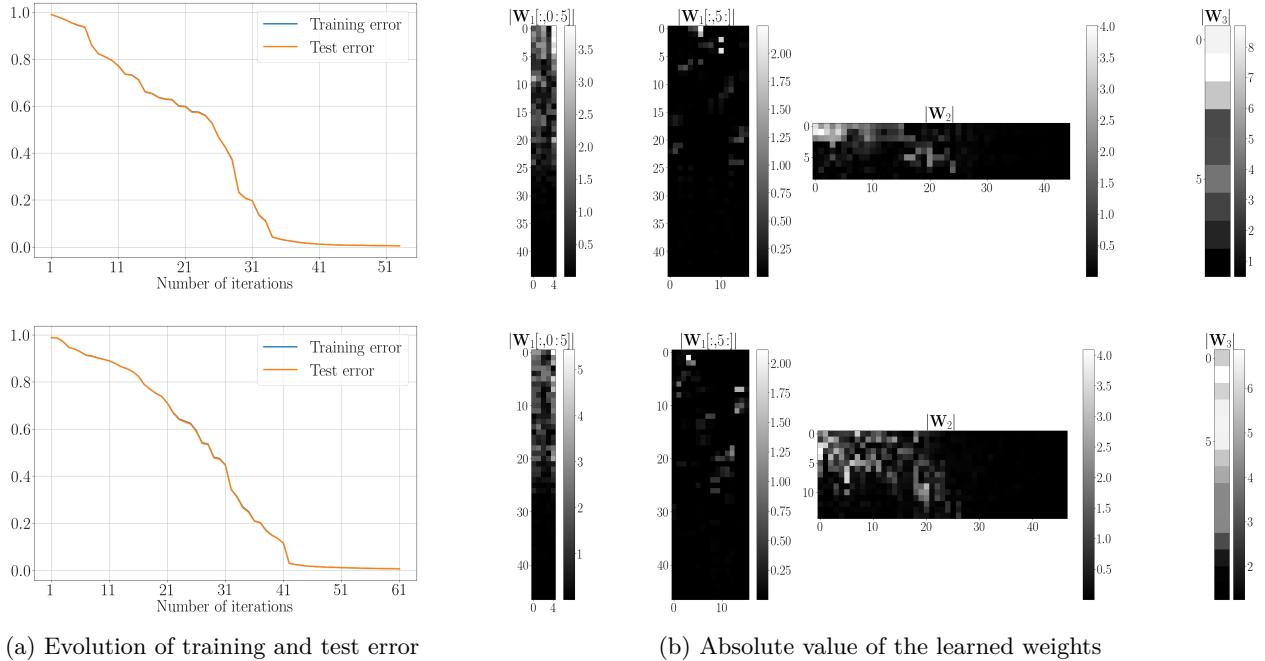


Figure 12: The results from two additional independent runs of learning the PVR task using a three-layer neural network activation function $\sigma(x) = \max(0, x)$, trained via the NP algorithm. Both runs achieve low training and test errors, and the weights of the first layer associated with \mathbf{x} are sparse, with dominant entries localized within each row

D Additional Discussion and Results

D.1 Proof of Lemma 6

We will show that $\mathbf{z}(t) := \frac{1}{\delta} \mathbf{s}_\delta \left(\frac{t}{\delta^{L-2}} \right)$ is the solution of

$$\dot{\mathbf{s}} = \nabla_{\mathbf{s}} g(\mathbf{s}), \mathbf{s}(0) = \mathbf{s}_0.$$

Note that, $\mathbf{z}(0) = \mathbf{s}_\delta(0)/\delta = \mathbf{s}_0$. Next,

$$\dot{\mathbf{z}} = \frac{1}{\delta^{L-1}} \dot{\mathbf{s}}_\delta \left(\frac{t}{\delta^{L-2}} \right) = \frac{1}{\delta^{L-1}} \nabla_{\mathbf{s}} g \left(\mathbf{s}_\delta \left(\frac{t}{\delta^{L-2}} \right) \right) = \nabla_{\mathbf{s}} g \left(\frac{1}{\delta} \mathbf{s}_\delta \left(\frac{t}{\delta^{L-2}} \right) \right) = \nabla_{\mathbf{s}} g(\mathbf{z}),$$

where the second equality follows from the definition of $\mathbf{s}_\delta(t)$, and the third equality holds since $\nabla_{\mathbf{s}} g(\cdot)$ is $(L-1)$ -homogeneous. This completes the proof.

D.2 Lojasiewicz's Inequality: An Example

To derive Theorem 7, we assumed that $\bar{\mathbf{w}}_n$ is a local minimum of $\tilde{\mathcal{L}}(\mathbf{w}_n)$ such that Lojasiewicz's inequality is satisfied in the neighborhood of $\bar{\mathbf{w}}_n$ with $\alpha \in \left(0, \frac{L}{2(L-1)}\right)$. We present a simple example where this assumption is true with $\alpha = \frac{1}{2}$.

Suppose $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{d \times 2d} \times \mathbb{R}^{2d}$ is the training dataset such that $y_i = (\mathbf{w}_*^\top \mathbf{x}_i)^p$, for some $p \geq 2$ and $i \in [2d]$, where $\mathbf{w}_* \in \mathbb{R}^d$. We assume that $\min_{i \in [2d]} |\mathbf{w}_*^\top \mathbf{x}_i| = \eta > 0$, $\mathbf{w}_*^\top \mathbf{x}_i > 0$, for all $1 \leq i \leq d$, and $\mathbf{w}_*^\top \mathbf{x}_i < 0$ for all $d+1 \leq i \leq 2d$. Let $\mathbf{X}_d \in \mathbb{R}^{d \times d}$ denote the submatrix formed by first d column of \mathbf{X} and $\rho > 0$ denotes the minimum singular value of $\mathbf{X}_d^\top \mathbf{X}_d$. Consider a two-layer neural network with $H \geq 2$ neurons:

$$\mathcal{H}(\mathbf{x}; \{v_i, \mathbf{u}_i\}_{i=1}^H) = \sum_{i=1}^H v_i \sigma(\mathbf{u}_i^\top \mathbf{x}),$$

where $\sigma(x) = \max(0, x)^p$. Thus, the training loss is

$$\mathcal{L}(\{v_i, \mathbf{u}_i\}_{i=1}^H) = \frac{1}{2} \left\| \sum_{i=1}^H v_i \sigma(\mathbf{X}^\top \mathbf{u}_i) - \mathbf{y} \right\|_2^2$$

Lemma 20. *Consider the following optimization problem:*

$$\tilde{\mathcal{L}}(v_1, \mathbf{u}_1) = \frac{1}{2} \|v_1 \sigma(\mathbf{X}^\top \mathbf{u}_1) - \mathbf{y}\|_2^2,$$

Let $(\bar{v}_1, \bar{\mathbf{u}}_1) = (\alpha, \beta \mathbf{w}_*)$, where $\alpha \beta^p = 1$ and $\beta > 0$. Then, $(\bar{v}_1, \bar{\mathbf{u}}_1)$ is a local minimum of $\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)$ such that Lojasiewicz's inequality is satisfied in the neighborhood of $(\bar{v}_1, \bar{\mathbf{u}}_1)$ with $\alpha = \frac{1}{2}$: there exists $\mu_1, \gamma > 0$ such that

$$\|\nabla \tilde{\mathcal{L}}(v_1, \mathbf{u}_1)\|_2 \geq \mu_1 \left(\tilde{\mathcal{L}}(v_1, \mathbf{u}_1) - \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) \right)^{\frac{1}{2}}, \text{ if } (v_1 - \bar{v}_1)^2 + \|\mathbf{u}_1 - \bar{\mathbf{u}}_1\|_2^2 \leq \gamma^2. \quad (51)$$

Proof. Define $\mathbf{e} = v_1 \sigma(\mathbf{X}^\top \mathbf{u}_1) - \mathbf{y}$, then

$$\nabla_{v_1} \tilde{\mathcal{L}}(v_1, \mathbf{u}_1) = \sigma(\mathbf{X}^\top \mathbf{u}_1)^\top \mathbf{e}, \nabla_{\mathbf{u}_1} \tilde{\mathcal{L}}(v_1, \mathbf{u}_1) = v_1 \mathbf{X} \text{diag}(\sigma'(\mathbf{X}^\top \mathbf{u}_1)) \mathbf{e}.$$

We first show that $\nabla_{v_1} \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) = 0$ and $\nabla_{\mathbf{u}_1} \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) = \mathbf{0}$. For $1 \leq i \leq d$, since $\bar{\mathbf{u}}_1^\top \mathbf{x}_i > 0$, we have $\sigma'(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) > 0$ and

$$\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i = \alpha \max(0, \beta \bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p - (\bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p = \alpha \beta^p (\bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p - (\bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p = 0.$$

For $d+1 \leq i \leq 2d$, since $\bar{\mathbf{u}}_1^\top \mathbf{x}_i < 0$, we have $\sigma'(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) = 0 = \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i)$ and

$$\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i = \alpha \max(0, \beta \bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p - (\bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p = -(\bar{\mathbf{u}}_1^\top \mathbf{x}_i)^p.$$

Hence,

$$\begin{aligned} \nabla_{v_1} \tilde{\mathcal{L}}(v_1, \mathbf{u}_1) &= \sum_{i=1}^d \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) (\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i) + \sum_{i=d+1}^{2d} \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) (\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i) = 0, \\ \nabla_{\mathbf{u}_1} \tilde{\mathcal{L}}(v_1, \mathbf{u}_1) &= \bar{v}_1 \left(\sum_{i=1}^d \mathbf{x}_i \sigma'(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) (\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i) + \sum_{i=d+1}^{2d} \mathbf{x}_i \sigma'(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) (\bar{v}_1 \sigma(\bar{\mathbf{u}}_1^\top \mathbf{x}_i) - y_i) \right) = \mathbf{0}. \end{aligned}$$

We next show that $(\bar{v}_1, \bar{\mathbf{u}}_1)$ is a local minimum of $\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)$. We use \mathbf{b} to denote any unit-norm vector orthogonal to \mathbf{w}_* . We will show next that there exists $\gamma_1 > 0$ such that if $\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 \leq \gamma_1$, then

$$\tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) \leq \tilde{\mathcal{L}}(\bar{v}_1 + \epsilon_1, (1 + \epsilon_2) \bar{\mathbf{u}}_1 + \epsilon_3 \mathbf{b}),$$

which would imply $(\bar{v}_1, \bar{\mathbf{u}}_1)$ is a local minimum of $\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)$. For any $1 \leq i \leq d$, we have

$$\begin{aligned} & ((\bar{v}_1 + \epsilon_1)\sigma(\mathbf{x}_i^\top((1 + \epsilon_2)\bar{\mathbf{u}}_1 + \epsilon_3\mathbf{b})) - y_i)^2 - (\bar{v}_1\sigma(\mathbf{x}_i^\top\bar{\mathbf{u}}_1) - y_i)^2 \\ &= ((\alpha + \epsilon_1)(\beta(1 + \epsilon_2)\mathbf{x}_i^\top\mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top\mathbf{b})^p - y_i)^2 \\ &= (\alpha\beta^p(\mathbf{x}_i^\top\mathbf{w}_*)^p + r_i - y_i)^2 = r_i^2 > 0 \end{aligned} \quad (52)$$

The first equality follows because $\beta\mathbf{x}_i^\top\mathbf{w}_* > 0$, hence if $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, then $\beta(1 + \epsilon_2)\mathbf{x}_i^\top\mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top\mathbf{b} > 0$. We use r_i to denote the residual term $(1 + \epsilon_1)(\beta(1 + \epsilon_2)\mathbf{x}_i^\top\mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top\mathbf{b})^p - \alpha\beta^p(\mathbf{x}_i^\top\mathbf{w}_*)^p$. Next, for $d+1 \leq i \leq 2d$, we have

$$\begin{aligned} & ((\bar{v}_1 + \epsilon_1)\sigma(\mathbf{x}_i^\top((1 + \epsilon_2)\bar{\mathbf{u}}_1 + \epsilon_3\mathbf{b})) - y_i)^2 - (\bar{v}_1\sigma(\mathbf{x}_i^\top\bar{\mathbf{u}}_1) - y_i)^2 \\ &= ((\alpha + \epsilon_1)\sigma(\beta(1 + \epsilon_2)\mathbf{x}_i^\top\mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top\mathbf{b}) - y_i)^2 - (y_i)^2 \\ &= (y_i)^2 - (y_i)^2 = 0. \end{aligned} \quad (53)$$

The second equality follows because $\beta\mathbf{x}_i^\top\mathbf{w}_* < 0$, hence if $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, then $\beta(1 + \epsilon_2)\mathbf{x}_i^\top\mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top\mathbf{b} < 0$. From the above two equations, we get that $(\bar{v}_1, \bar{\mathbf{u}}_1)$ is a local minimum of $\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)$.

We next prove eq. (51). Note that

$$\|\nabla_{v_1}\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)\|_2^2 + \|\nabla_{\mathbf{u}_1}\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)\|_2^2 \geq \|\nabla_{\mathbf{u}_1}\tilde{\mathcal{L}}(v_1, \mathbf{u}_1)\|_2^2 = v_1^2 \mathbf{e}^\top \text{diag}(\sigma'(\mathbf{X}^\top \mathbf{u}_1)) \mathbf{X}^\top \mathbf{X} \text{diag}(\sigma'(\mathbf{X}^\top \mathbf{u}_1)) \mathbf{e}.$$

We use $(\tilde{v}_1, \tilde{\mathbf{u}}_1) := (\bar{v}_1 + \epsilon_1, (1 + \epsilon_2)\bar{\mathbf{u}}_1 + \epsilon_3\mathbf{b})$ to denote a vector in the neighborhood of $(\bar{v}_1, \bar{\mathbf{u}}_1)$. If $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, then, for $d+1 \leq i \leq 2d$,

$$\mathbf{x}_i^\top \tilde{\mathbf{u}}_1 = \beta(1 + \epsilon_2)\mathbf{x}_i^\top \mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top \mathbf{b} \leq 0,$$

which implies $\sigma(\mathbf{x}_i^\top \tilde{\mathbf{u}}_1) = 0 = \sigma'(\mathbf{x}_i^\top \tilde{\mathbf{u}}_1)$. Hence,

$$\text{diag}(\sigma'(\mathbf{X}^\top \tilde{\mathbf{u}}_1)) \mathbf{X}^\top \mathbf{X} \text{diag}(\sigma'(\mathbf{X}^\top \tilde{\mathbf{u}}_1)) = \begin{bmatrix} \text{diag}(\sigma'(\mathbf{X}_d^\top \tilde{\mathbf{u}}_1)) \mathbf{X}_d^\top \mathbf{X}_d \text{diag}(\sigma'(\mathbf{X}_d^\top \tilde{\mathbf{u}}_1)) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Therefore,

$$\begin{aligned} \|\nabla_{v_1}\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 + \|\nabla_{\mathbf{u}_1}\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 &\geq (\bar{v}_1 + \epsilon_1)^2 \rho \sum_{i=1}^d (\sigma'(\mathbf{x}_i^\top \tilde{\mathbf{u}}_1)((\bar{v}_1 + \epsilon_1)\sigma(\mathbf{x}_i^\top \tilde{\mathbf{u}}_1) - y_i))^2 \\ &= (\alpha + \epsilon_1)^2 \rho \sum_{i=1}^d (\sigma'(\mathbf{x}_i^\top \tilde{\mathbf{u}}_1)r_i)^2, \end{aligned}$$

where r_i is same as in eq. (52). The first inequality is true since the minimum singular value of $\mathbf{X}_d^\top \mathbf{X}_d$ is ρ . The last equality holds true if $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, as shown in eq. (52). If $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, then

$$\mathbf{x}_i^\top \tilde{\mathbf{u}}_1 = \beta(1 + \epsilon_2)\mathbf{x}_i^\top \mathbf{w}_* + \epsilon_3\mathbf{x}_i^\top \mathbf{b} \geq \beta\mathbf{x}_i^\top \mathbf{w}_*/2, \text{ for all } 1 \leq i \leq d.$$

Since $\mathbf{x}_i^\top \mathbf{w}_* \geq \eta$, for all $1 \leq i \leq d$, we get

$$\|\nabla_{v_1}\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 + \|\nabla_{\mathbf{u}_1}\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 \geq (\alpha + \epsilon_1)^2 p \rho \left(\frac{\beta\eta}{2}\right)^{p-1} \sum_{i=1}^d r_i^2. \quad (54)$$

If $|\epsilon_2|, |\epsilon_3|$ is sufficiently small, from eq. (52) and eq. (53), we know

$$\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1) - \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) = \sum_{i=1}^d r_i^2.$$

Hence, from eq. (54) and the above equation, we get

$$\begin{aligned} \|\nabla_{v_1} \tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 + \|\nabla_{\mathbf{u}_1} \tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1)\|_2^2 &\geq (\alpha + \epsilon_1)^2 p \rho \left(\frac{\beta \eta}{2} \right)^{p-1} \left(\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1) - \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) \right) \\ &\geq p \rho \frac{\alpha^2}{4} \left(\frac{\beta \eta}{2} \right)^{p-1} \left(\tilde{\mathcal{L}}(\tilde{v}_1, \tilde{\mathbf{u}}_1) - \tilde{\mathcal{L}}(\bar{v}_1, \bar{\mathbf{u}}_1) \right), \end{aligned}$$

where the last inequality holds if $|\epsilon_1| < \alpha/2$. Hence, eq. (51) holds true in a sufficiently small neighborhood of $(\bar{v}_1, \bar{\mathbf{u}}_1)$. \square

D.3 What happens if $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = 0$?

In Theorem 7, we showed that, near the saddle point $(\bar{\mathbf{w}}_n, \mathbf{0})$, weights in \mathbf{w}_z remain small in magnitude but converge in direction to the constrained NCF corresponding to $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z)$. The proof of Theorem 7 proceeds by first showing that the output of the neural network $\mathcal{H}(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ can be decomposed into a leading term that is independent of \mathbf{w}_z , another term $\mathcal{H}_1(\mathbf{x}; \mathbf{w}_n, \mathbf{w}_z)$ that is homogeneous in \mathbf{w}_z and other residual terms that are small. Then, it was shown that the evolution of \mathbf{w}_z near the saddle point is close to the gradient flow of $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z)$, where $\bar{\mathcal{H}}_1(\mathbf{x}; \mathbf{w}_z) = \mathcal{H}_1(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z)$.

Here, we consider the scenario when $\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = 0$, for all \mathbf{w}_z . Technically, we can still apply Theorem 7. Since the constrained NCF is zero, all unit-norm vectors are its KKT point. Therefore, directional convergence holds trivially at initialization. However, this argument, while technically correct, does not capture the behavior observed in our experiments. Empirically, we find that even in this setting, the weights in \mathbf{w}_z converge in direction, however, the residual terms play an important role in determining where they converge. To investigate this phenomenon, we next study the problem of matrix decomposition using three-layer linear neural network.

Deep linear network. Suppose $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a rank d matrix with singular values $\{s_1, s_2, \dots, s_d\}$ and singular vectors $\{\mathbf{u}_i, \mathbf{v}_i\}_{i=1}^d$. Consider using a three-layer linear neural network to learn \mathbf{S} , then the training loss becomes

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) = \frac{1}{2} \|\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 - \mathbf{S}\|_F^2,$$

where each of the weight matrices is in $\mathbb{R}^{d \times d}$. Here, the input can be assumed to be the identity matrix, the output of the neural network is $\mathcal{H}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) = \mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1$, and \mathbf{S} is the label. The gradient of the training loss with respect to the weights is as follows:

$$\begin{aligned} \nabla_{\mathbf{W}_1} \mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) &= \mathbf{W}_2^\top \mathbf{W}_3^\top (\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 - \mathbf{S}), \\ \nabla_{\mathbf{W}_2} \mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) &= \mathbf{W}_3^\top (\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 - \mathbf{S}) \mathbf{W}_1^\top, \\ \nabla_{\mathbf{W}_3} \mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) &= (\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 - \mathbf{S}) \mathbf{W}_1^\top \mathbf{W}_2^\top. \end{aligned}$$

Define

$$\bar{\mathbf{W}}_1 = \begin{bmatrix} s_1^{1/3} \mathbf{v}_1^\top \\ \mathbf{0} \end{bmatrix}, \bar{\mathbf{W}}_2 = \begin{bmatrix} s_1^{1/3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \bar{\mathbf{W}}_3 = \begin{bmatrix} s_1^{1/3} \mathbf{u}_1 & \mathbf{0} \end{bmatrix}, \quad (55)$$

then $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$ is a saddle point of the training loss. This is true since

$$\begin{aligned} \nabla_{\mathbf{W}_1} \mathcal{L}(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3) &= - \begin{bmatrix} s_1^{1/3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} s_1^{1/3} \mathbf{u}_1^\top \\ \mathbf{0} \end{bmatrix} \sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top = \mathbf{0}, \\ \nabla_{\mathbf{W}_2} \mathcal{L}(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3) &= - \begin{bmatrix} s_1^{1/3} \mathbf{u}_1^\top \\ \mathbf{0} \end{bmatrix} \sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top \begin{bmatrix} s_1^{1/3} \mathbf{v}_1 & \mathbf{0} \end{bmatrix} = \mathbf{0}, \\ \nabla_{\mathbf{W}_3} \mathcal{L}(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3) &= \sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top \begin{bmatrix} s_1^{1/3} \mathbf{v}_1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} s_1^{1/3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{0}, \end{aligned}$$

where we used mutual orthogonality of singular vectors and $\mathbf{S} - \overline{\mathbf{W}}_3 \overline{\mathbf{W}}_2 \overline{\mathbf{W}}_1 = \sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top$. Let us look at the output of neural network near the saddle point $(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3)$. More specifically, let

$$\mathbf{W}_1 = \begin{bmatrix} s_1^{1/3} \mathbf{v}_1^\top \\ \mathbf{A}_1 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} s_1^{1/3} & \mathbf{b}_2^\top \\ \mathbf{a}_2 & \mathbf{C}_2 \end{bmatrix}, \mathbf{W}_3 = \begin{bmatrix} s_1^{1/3} \mathbf{u}_1 & \mathbf{B}_3 \end{bmatrix},$$

where $\mathbf{a}_2, \mathbf{b}_2$ are vectors. Here, $\mathbf{A}_1, \mathbf{a}_2, \mathbf{b}_2, \mathbf{C}_2, \mathbf{B}_3$ belong to \mathbf{w}_z and other weights belong to \mathbf{w}_n . Next, we can write

$$\begin{aligned} \mathcal{H}(\overline{\mathbf{w}}_n, \mathbf{w}_z) &= \mathcal{H}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) = \begin{bmatrix} s_1^{1/3} \mathbf{u}_1 & \mathbf{B}_3 \end{bmatrix} \begin{bmatrix} s_1^{1/3} & \mathbf{b}_2^\top \\ \mathbf{a}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} s_1^{1/3} \mathbf{v}_1^\top \\ \mathbf{A}_1 \end{bmatrix} \\ &= \begin{bmatrix} s_1^{1/3} \mathbf{u}_1 & \mathbf{B}_3 \end{bmatrix} \begin{bmatrix} s_1^{2/3} \mathbf{v}_1^\top + \mathbf{b}_2^\top \mathbf{A}_1 \\ s_1^{1/3} \mathbf{a}_2 \mathbf{v}_1^\top + \mathbf{C}_2 \mathbf{A}_1 \end{bmatrix} \\ &= s_1 \mathbf{u}_1 \mathbf{v}_1^\top + s_1^{1/3} \mathbf{u}_1 \mathbf{b}_2^\top \mathbf{A}_1 + s_1^{1/3} \mathbf{B}_3 \mathbf{a}_2 \mathbf{v}_1^\top + \mathbf{B}_3 \mathbf{C}_2 \mathbf{A}_1. \end{aligned}$$

Hence,

$$\mathcal{H}(\overline{\mathbf{w}}_n, \mathbf{0}) = s_1 \mathbf{u}_1 \mathbf{v}_1^\top, \mathcal{H}_1(\overline{\mathbf{w}}_n, \mathbf{w}_z) = s_1^{1/3} \mathbf{u}_1 \mathbf{b}_2^\top \mathbf{A}_1 + s_1^{1/3} \mathbf{B}_3 \mathbf{a}_2 \mathbf{v}_1^\top, \mathcal{H}_2(\overline{\mathbf{w}}_n, \mathbf{w}_z) = \mathbf{B}_3 \mathbf{C}_2 \mathbf{A}_1.$$

The inner product between the residual error $\overline{\mathbf{S}} := \mathbf{S} - \overline{\mathbf{W}}_3 \overline{\mathbf{W}}_2 \overline{\mathbf{W}}_1$ and $\mathcal{H}_1(\overline{\mathbf{w}}_n, \mathbf{w}_z)$ is

$$\text{trace} \left(\overline{\mathbf{S}}^\top \mathcal{H}_1(\overline{\mathbf{w}}_n, \mathbf{w}_z) \right) = s_1^{1/3} \text{trace} \left(\left(\sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top \right)^\top \mathbf{u}_1 \mathbf{b}_2^\top \mathbf{A}_1 \right) + s_1^{1/3} \text{trace} \left(\left(\sum_{i=2}^d s_i \mathbf{u}_i \mathbf{v}_i^\top \right)^\top \mathbf{B}_3 \mathbf{a}_2 \mathbf{v}_1^\top \right) = 0.$$

Thus, the inner product between the residual error and $\mathcal{H}_1(\overline{\mathbf{w}}_n, \mathbf{w}_z)$ is 0. Now, when initialized near $(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3)$, the experiment in Figure 13 shows that weights in \mathbf{w}_z remain small in norm but converge in direction towards a KKT point of constrained NCF defined with respect to the residual error and $\mathcal{H}_2(\overline{\mathbf{w}}_n, \mathbf{w}_z)$. Therefore, even when $\mathcal{N}_{\overline{\mathbf{y}}, \overline{\mathcal{H}}_1}(\mathbf{w}_z) = 0$, for all \mathbf{w}_z , the weights in \mathbf{w}_z remain small in magnitude and converge in direction during the initial stages of training. Moreover, the direction of convergence is determined by the constrained NCF defined with respect to the residual error and $\mathcal{H}_2(\overline{\mathbf{w}}_n, \mathbf{w}_z)$.

Squared ReLU activation. Consider a three-layer neural network with squared ReLU activation function $\sigma(x) = \max(0, x)^2$. Suppose the training set contains two samples $\{\mathbf{x}_1, y_1\} = \{\mathbf{1}/d, 1\}$ and $\{\mathbf{x}_2, y_2\} = \{-\mathbf{1}/d, 1\}$, where $\mathbf{1}$ is the all-one vector in \mathbb{R}^d . The training loss becomes

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) = \frac{1}{2} \|\mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_1)) - y_1\|_2^2 + \frac{1}{2} \|\mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_2)) - y_2\|_2^2,$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_3 \in \mathbb{R}^{1 \times d}$. Define

$$\overline{\mathbf{W}}_1 = \begin{bmatrix} \mathbf{1}^\top \\ \mathbf{0} \end{bmatrix}, \overline{\mathbf{W}}_2 = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \overline{\mathbf{W}}_3 = [1 \quad \mathbf{0}], \quad (56)$$

then $(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3)$ is a saddle point of the training loss. To show this, first note that

$$\begin{aligned} \bar{y}_1 &:= y_1 - \overline{\mathbf{W}}_3 \sigma(\overline{\mathbf{W}}_2 \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_1)) = 1 - \sigma(\sigma(\mathbf{1}^\top \mathbf{1}/d)) = 0, \\ \bar{y}_2 &:= y_2 - \overline{\mathbf{W}}_3 \sigma(\overline{\mathbf{W}}_2 \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_2)) = 1 - \sigma(\sigma(-\mathbf{1}^\top \mathbf{1}/d)) = 1. \end{aligned}$$

Hence,

$$\begin{aligned} \nabla_{\mathbf{W}_1} \mathcal{L}(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3) &= -\text{diag}(\sigma'(\overline{\mathbf{W}}_1 \mathbf{x}_2)) \overline{\mathbf{W}}_2^\top \text{diag}(\sigma'(\overline{\mathbf{W}}_2 \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_2))) \overline{\mathbf{W}}_3^\top \mathbf{x}_2^\top \bar{y}_2 = \mathbf{0}, \\ \nabla_{\mathbf{W}_2} \mathcal{L}(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3) &= -\text{diag}(\sigma'(\overline{\mathbf{W}}_2 \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_2))) \overline{\mathbf{W}}_3^\top \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_2)^\top \bar{y}_2 = \mathbf{0}, \\ \nabla_{\mathbf{W}_3} \mathcal{L}(\overline{\mathbf{W}}_1, \overline{\mathbf{W}}_2, \overline{\mathbf{W}}_3) &= -\sigma(\overline{\mathbf{W}}_2 \sigma(\overline{\mathbf{W}}_1 \mathbf{x}_2)) \bar{y}_2 = \mathbf{0}, \end{aligned}$$

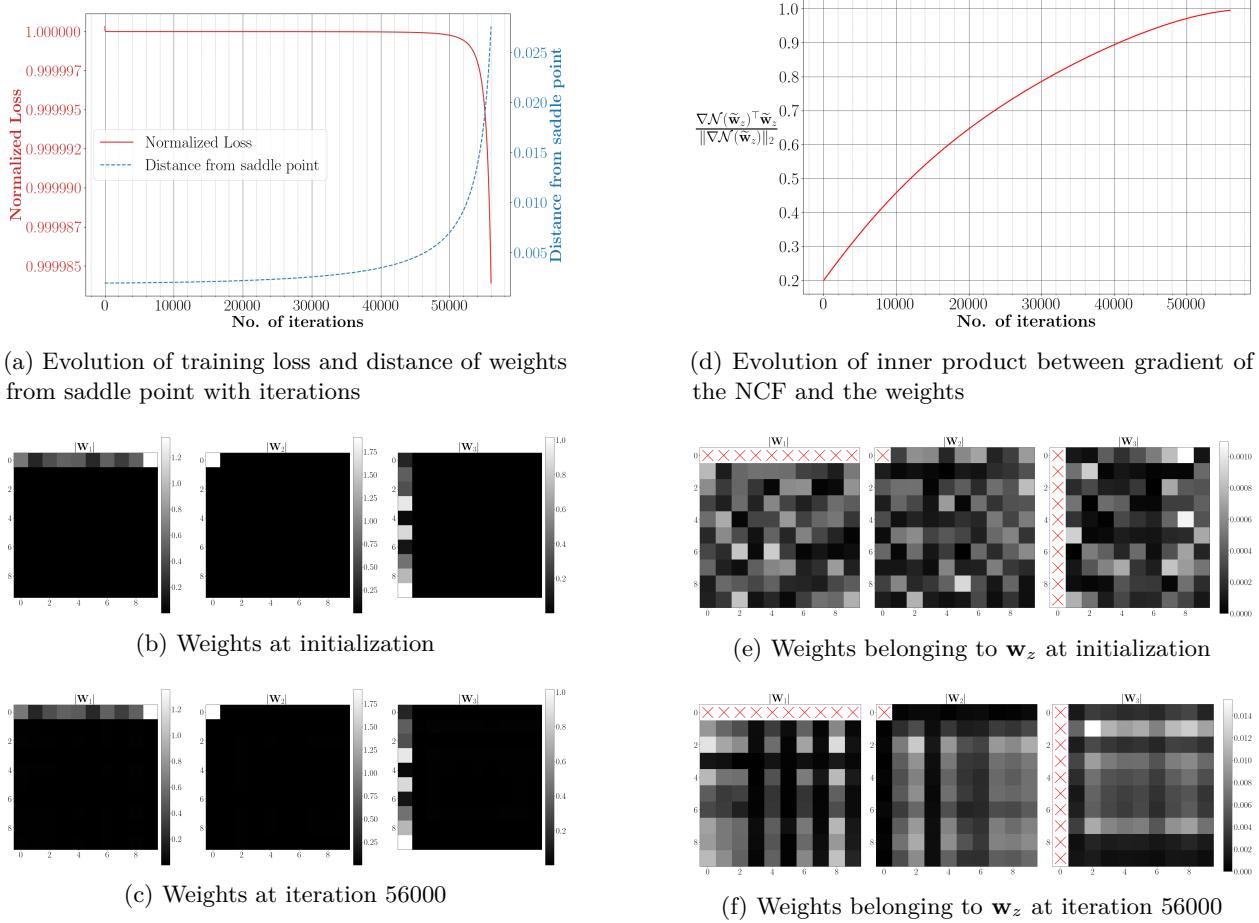


Figure 13: **(Gradient descent dynamics near saddle point)** We train a three-layer linear neural network using gradient descent whose output is $\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1$, where $\mathbf{W}_3 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_2 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_1 \in \mathbb{R}^{10 \times 10}$ are the trainable weights. The entries of label matrix $\mathbf{S} \in \mathbb{R}^{10 \times 10}$ are drawn from the standard normal distribution. The weights are initialized near a saddle point where the incoming and outgoing weights of the last nine neurons of each layer is zero, just as the saddle point defined in eq. (55); these weights form \mathbf{w}_z and remaining form \mathbf{w}_n . Panel (a) depicts the evolution of the training loss (normalized by the loss at saddle point) and the distance of the weights from the saddle point (normalized by the norm of the weights at saddle point). Panel (b) and (c) shows the weights at initialization and at iteration 56000, respectively. We observe that the training loss does not change much, the weights remain near the saddle point and \mathbf{w}_z remains small. Panel (d) shows the evolution of $\nabla \mathcal{N}_{\bar{\mathbf{S}}, \bar{\mathcal{H}}_2}(\tilde{\mathbf{w}}_z)^\top \tilde{\mathbf{w}}_z / \|\nabla \mathcal{N}_{\bar{\mathbf{S}}, \bar{\mathcal{H}}_2}(\tilde{\mathbf{w}}_z)\|_2$, where $\tilde{\mathbf{w}}_z := \mathbf{w}_z / \|\mathbf{w}_z\|_2$, which confirms that \mathbf{w}_z has converged in direction to a KKT point of the constrained NCF defined with respect to the residual error $\bar{\mathbf{S}}$ and $\bar{\mathcal{H}}_2(\mathbf{w}_z) := \mathcal{H}_2(\bar{\mathbf{w}}_n, \mathbf{w}_z)$. Panel (e) and (f) depicts the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 56000, where the weights belonging to \mathbf{w}_n are crossed out.

where we used $\sigma(\bar{\mathbf{W}}_1 \mathbf{x}_2) = \mathbf{0}$ and $\sigma'(0) = 0$.

Let us look the output of neural network near the saddle point $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$. More specifically, let

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{1}^\top \\ \mathbf{A}_1 \end{bmatrix}, \mathbf{W}_2 = \begin{bmatrix} 1 & \mathbf{b}_2^\top \\ \mathbf{a}_2 & \mathbf{C}_2 \end{bmatrix}, \mathbf{W}_3 = [1 \quad \mathbf{b}_3^\top],$$

where $\mathbf{a}_2, \mathbf{b}_2, \mathbf{b}_3$ are vectors. Here, $\mathbf{A}_1, \mathbf{a}_2, \mathbf{b}_2, \mathbf{C}_2, \mathbf{b}_3$ belong to \mathbf{w}_z and other weights belong to \mathbf{w}_n . Thus, using Lemma 8, we can write

$$\bar{\mathcal{H}}_1(\mathbf{x}; \mathbf{w}_z) = \mathcal{H}_1(\mathbf{x}; \bar{\mathbf{w}}_n, \mathbf{w}_z) = \mathbf{b}_3 \sigma(\mathbf{a}_2^\top \sigma(\bar{\mathbf{W}}_1 \mathbf{x})) + \bar{\mathbf{W}}_3 \text{diag}(\sigma'(\bar{\mathbf{W}}_2 \sigma(\bar{\mathbf{W}}_1 \mathbf{x}))) \mathbf{b}_2^\top \sigma(\mathbf{A}_1 \mathbf{x}),$$

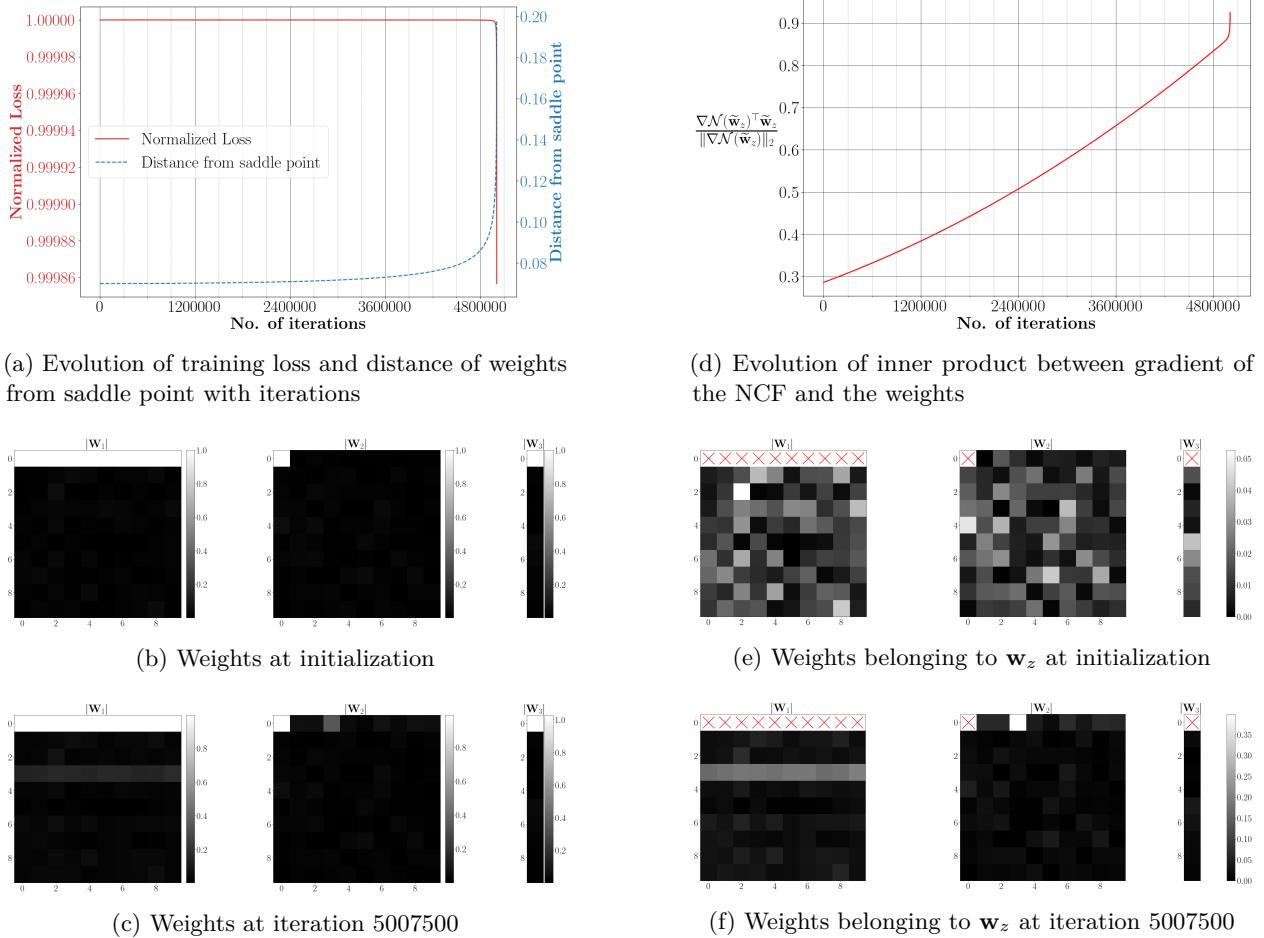


Figure 14: **(Gradient descent dynamics near saddle point)** We train a three-layer neural network with squared ReLU activation using gradient descent whose output is $\mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}))$, where $\mathbf{W}_3 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_2 \in \mathbb{R}^{10 \times 10}$, $\mathbf{W}_1 \in \mathbb{R}^{10 \times 10}$ are the trainable weights. The weights are initialized near a saddle point where the incoming and outgoing weights of the last nine neurons of each layer is zero, just as the saddle point defined in eq. (56); these weights form \mathbf{w}_z and remaining form \mathbf{w}_n . Panel (a) depicts the evolution of the training loss (normalized by the loss at saddle point) and the distance of the weights from the saddle point (normalized by the norm of the weights at saddle point). Panel (b) and (c) shows the weights at initialization and at iteration 5007500, respectively. We observe that the training loss does not change much, the weights remain near the saddle point and \mathbf{w}_z remains small. Panel (d) shows the evolution of $\nabla \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_2}(\tilde{\mathbf{w}}_z)^\top \tilde{\mathbf{w}}_z / \|\nabla \mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_2}(\tilde{\mathbf{w}}_z)\|_2$, where $\tilde{\mathbf{w}}_z := \mathbf{w}_z / \|\mathbf{w}_z\|_2$, which confirms that \mathbf{w}_z has approximately converged in direction to a KKT point of the constrained NCF defined with respect to the residual error $\bar{\mathbf{y}}$ and $\bar{\mathcal{H}}_2$. Panel (e) and (f) depicts the weights belonging to \mathbf{w}_z of every layer, at initialization and at iteration 5007500, where the weights belonging to \mathbf{w}_n are crossed out. See the text for more details.

which implies

$$\mathcal{N}_{\bar{\mathbf{y}}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = \bar{\mathbf{y}}_1 \bar{\mathcal{H}}_1(\mathbf{x}_1; \mathbf{w}_z) + \bar{\mathbf{y}}_2 \bar{\mathcal{H}}_1(\mathbf{x}_2; \mathbf{w}_z) = 0 + 0 = 0.$$

The second equality follows since $\bar{\mathbf{y}}_1 = 0$ and $\sigma(\bar{\mathbf{W}}_1 \mathbf{x}_2) = \mathbf{0}$. Thus, the constrained NCF is zero for all \mathbf{w}_z .

We next take a closer look at the network output near the saddle point. Since $\bar{y}_1 = 0$, we only analyze the case when the input is \mathbf{x}_2 .

$$\begin{aligned}\mathcal{H}(\mathbf{x}_2; \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3) &= [1 \quad \mathbf{b}_3^\top] \sigma \left(\begin{bmatrix} 1 & \mathbf{b}_2^\top \\ \mathbf{a}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \sigma(-\mathbf{1}^\top \mathbf{1}/\sqrt{d}) \\ \sigma(\mathbf{A}_1 \mathbf{x}_2) \end{bmatrix} \right) \\ &= [1 \quad \mathbf{b}_3^\top] \sigma \left(\begin{bmatrix} 1 & \mathbf{b}_2^\top \\ \mathbf{a}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} 0 \\ \sigma(\mathbf{A}_1 \mathbf{x}_2) \end{bmatrix} \right) \\ &= [1 \quad \mathbf{b}_3^\top] \sigma \left(\begin{bmatrix} \mathbf{b}_2^\top \sigma(\mathbf{A}_1 \mathbf{x}_2) \\ \mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}_2) \end{bmatrix} \right) = \sigma(\mathbf{b}_2^\top \sigma(\mathbf{A}_1 \mathbf{x}_2)) + \mathbf{b}_3^\top \sigma(\mathbf{C}_2 \sigma(\mathbf{A}_1 \mathbf{x}_2)).\end{aligned}$$

Define $\bar{\mathcal{H}}_2(\mathbf{x}; \mathbf{w}_z) := \sigma(\mathbf{b}_2^\top \sigma(\mathbf{A}_1 \mathbf{x}))$, then $\bar{\mathcal{H}}_2(\mathbf{x}; \mathbf{w}_z)$ is a homogeneous function in \mathbf{w}_z with least degree of homogeneity in the above decomposition. Empirically, if initialized near $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$, the experiment in Figure 14 shows that weights in \mathbf{w}_z remain small in norm but converge in direction towards a KKT point of constrained NCF defined with respect to the residual error and $\bar{\mathcal{H}}_2$. Since $\bar{y}_1 = 0$, the objective of this constrained NCF is $\bar{y}_2 \bar{\mathcal{H}}_2(\mathbf{x}_2; \mathbf{w}_z)$.

Thus, even when $\mathcal{N}_{\bar{y}, \bar{\mathcal{H}}_1}(\mathbf{w}_z) = 0$, for all \mathbf{w}_z , the weights in \mathbf{w}_z remain small in magnitude and converge in direction during the initial stages of training. Moreover, this direction is determined by the constrained NCF defined with respect to the residual error and $\bar{\mathcal{H}}_2$.

D.4 Maximizing Sum of Homogeneous Functions via Gradient Flow

This section contains the proof of Lemma 10 and Lemma 11

Proof of Lemma 10: We will first show that there exists a time T such that

$$\|\mathbf{w}_i(t)\|_2 = \alpha_i(t) e^{2t\mathcal{G}_i(\mathbf{w}_i^*)}, \text{ for all } t \geq T \text{ and for all } i \in [m], \quad (57)$$

where T is sufficiently large. Also, $\alpha_i(t) \in [\kappa_1, \kappa_2]$, for all $t \geq T$ and $i \in [m]$, where $\kappa_2 \geq \kappa_1 > 0$. If the above equation is true, then the proof can be finished in the following way. Suppose $\zeta := \max_{j \in [m]} \mathcal{G}_j(\mathbf{w}_j^*)$. For any $i \in [m]$, if $\mathcal{G}_i(\mathbf{w}_i^*) < \zeta$, then

$$\lim_{t \rightarrow \infty} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{i=1}^m \|\mathbf{w}_i(t)\|_2^2}} \leq \lim_{t \rightarrow \infty} \frac{\kappa_2 e^{2t\mathcal{G}_i(\mathbf{w}_i^*)}}{\kappa_1 e^{2t\zeta}} = 0.$$

Else, if $\mathcal{G}_i(\mathbf{w}_i^*) = \zeta$, then

$$\lim_{t \rightarrow \infty} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{i=1}^m \|\mathbf{w}_i(t)\|_2^2}} \geq \lim_{t \rightarrow \infty} \frac{\kappa_1 e^{2t\zeta}}{\sqrt{m} \kappa_2 e^{2t\zeta}} > 0.$$

We next show that eq. (57) is true. For all $i \in [m]$, since \mathbf{w}_i^* is a second-order KKT point, from Kumar & Haupt (2025b, Lemma 24), there exists a $\gamma \in (0, 1)$ such that if $\mathbf{s}_{i0}^\top \mathbf{w}_i^* > 1 - \gamma$ and $\|\mathbf{s}_{i0}\|_2 = 1$, then the solution $\mathbf{s}_i(t)$ of

$$\dot{\mathbf{s}}_i = \nabla \mathcal{G}_i(\mathbf{s}_i), \mathbf{s}_i(0) = \mathbf{s}_{i0},$$

satisfies $\|\mathbf{s}_i(t)\|_2 = \beta_i(t) e^{2t\mathcal{G}_i(\mathbf{w}_i^*)}$, for all $t \geq 0$. Here, $\beta_i(t) \in [\eta_1, \eta_2]$, for all $t \geq 0$ and $i \in [m]$, where $\eta_2 \geq \eta_1 > 0$.

Now, we may assume that there exists a sufficiently large time T^* such that

$$\frac{\mathbf{w}_i(t)^\top \mathbf{w}_i^*}{\|\mathbf{w}_i(t)\|_2} > 1 - \gamma, \text{ for all } t \geq T^* \text{ and for all } i \in [m].$$

Suppose $\tilde{\mathbf{s}}_i(t)$ denotes the solution of

$$\dot{\mathbf{s}}_i = \nabla \mathcal{G}_i(\mathbf{s}_i), \mathbf{s}_i(0) = \mathbf{w}_i(T^*) / \|\mathbf{w}_i(T^*)\|_2,$$

then $\|\tilde{\mathbf{s}}_i(t)\|_2 = \beta_i(t)e^{2t\mathcal{G}_i(\mathbf{w}_i^*)}$, for all $t \geq 0$, and $\beta_i(t) \in [\eta_1, \eta_2]$, for all $t \geq 0$ and $i \in [m]$, where $\eta_2 \geq \eta_1 > 0$. Also, using Lemma 6, we have

$$\tilde{\mathbf{s}}_i(t) = \frac{\mathbf{w}_i(t+T^*)}{\|\mathbf{w}_i(T^*)\|_2}, \text{ for all } t \geq 0.$$

Therefore, $\|\mathbf{w}_i(t)\|_2 = \alpha_i(t)e^{2\mathcal{G}_i(\mathbf{w}_i^*)t}$ and $\alpha_i(t) \in [\theta_1\eta_1, \theta_2\eta_2]$, for all $t \geq T^*$, where $\theta_1 = \min_{i \in [m]} \|\mathbf{w}_i(T^*)\|_2$ and $\theta_2 = \max_{i \in [m]} \|\mathbf{w}_i(T^*)\|_2$. \square

Proof of Lemma 11: For all $i \in [m]$, since $\mathcal{G}_i(\mathbf{w}_{i0}) > 0$, from (Kumar & Haupt, 2025a, Lemma 13), we get $\|\mathbf{w}_i(t)\|_2 \geq \|\mathbf{w}_{i0}\|_2 = 1 > 0$, for all $t \geq 0$. Next, note that

$$\frac{d}{dt} \left(\frac{\mathcal{G}_i(\mathbf{w}_i)}{\|\mathbf{w}_i\|_2^L} \right) = \nabla \mathcal{G}_i(\mathbf{w}_i)^\top \left(\mathbf{I} - \frac{\mathbf{w}_i \mathbf{w}_i^\top}{\|\mathbf{w}_i\|_2^2} \right) \frac{\nabla \mathcal{G}_i(\mathbf{w}_i)}{\|\mathbf{w}_i\|_2^L} \geq 0,$$

which implies $\mathcal{G}_i(\mathbf{w}_i(t))/\|\mathbf{w}_i(t)\|_2$ increases with time. Hence,

$$\frac{1}{2} \frac{d\|\mathbf{w}_i(t)\|_2^2}{dt} = L\mathcal{G}_i(\mathbf{w}_i) = L\|\mathbf{w}_i\|_2^L \mathcal{G}_i(\mathbf{w}_i/\|\mathbf{w}_i\|_2) \geq L\|\mathbf{w}_i\|_2^L \mathcal{G}_i(\mathbf{w}_{i0}/\|\mathbf{w}_{i0}\|_2) = L\|\mathbf{w}_i\|_2^L \mathcal{G}_i(\mathbf{w}_{i0}) \geq 0, \quad (58)$$

which implies

$$\frac{d\|\mathbf{w}_i\|_2}{dt} \geq L\|\mathbf{w}_i\|_2^{L-1} \mathcal{G}_i(\mathbf{w}_{i0}).$$

Taking $\|\mathbf{w}_i\|_2^{L-1}$ to the LHS and integrating from 0 to $t \in [0, T_i]$, we get

$$\frac{1}{L-2} \left(\frac{1}{\|\mathbf{w}_i(0)\|_2^{L-2}} - \frac{1}{\|\mathbf{w}_i(t)\|_2^{L-2}} \right) \geq L\mathcal{G}_i(\mathbf{w}_{i0})t.$$

Using $\|\mathbf{w}_i(0)\|_2 = \|\mathbf{w}_{i0}\|_2 = 1$, and simplifying the above equation we get

$$\|\mathbf{w}_i(t)\|_2^{L-2} \geq \frac{1}{1 - tL(L-2)\mathcal{G}_i(\mathbf{w}_{i0})}.$$

Since $\lim_{t \rightarrow T_i} \|\mathbf{w}_i(t)\|_2 = \infty$, the above equation implies $T_i \leq 1/(L(L-2)\mathcal{G}_i(\mathbf{w}_{i0}))$. Next, since $\mathcal{G}_i(\mathbf{w}_i(t)/\|\mathbf{w}_i(t)\|_2)$ increases with time and $\lim_{t \rightarrow T_i} \mathcal{G}_i(\mathbf{w}_i(t)/\|\mathbf{w}_i(t)\|_2) = \mathcal{G}_i(\mathbf{w}_i^*)$, we get

$$\frac{1}{2} \frac{d\|\mathbf{w}_i(t)\|_2^2}{dt} = L\mathcal{G}_i(\mathbf{w}_i) \leq L\|\mathbf{w}_i\|_2^L \mathcal{G}_i(\mathbf{w}_i^*), \quad (59)$$

Simplifying the above equation similarly gives us

$$\|\mathbf{w}_i(t)\|_2^{L-2} \leq \frac{1}{1 - tL(L-2)\mathcal{G}_i(\mathbf{w}_i^*)}.$$

Since $\lim_{t \rightarrow T_i} \|\mathbf{w}_i(t)\|_2 = \infty$, the above equation implies $T_i \geq 1/(L(L-2)\mathcal{G}_i(\mathbf{w}_i^*))$.

Next, if $T_i = T^*$, then $\lim_{t \rightarrow T_i} \|\mathbf{w}_i(t)\|_2 = \infty$, and if $T_i > T^*$, then $\lim_{t \rightarrow T_i} \|\mathbf{w}_i(t)\|_2 < \infty$. Since $T_i = T^*$, for some $i \in [m]$, we get

$$\lim_{t \rightarrow T_i} \sum_{i=1}^m \|\mathbf{w}_i(t)\|_2^2 = \infty.$$

Finally, from the proof of (Kumar & Haupt, 2025a, Lemma 2), we have

$$\|\mathbf{w}_i(t)\|_2 = \frac{\alpha_i(t)}{(T_i - t)^{1/(L-2)}}, \text{ for all } t \in [0, T_i] \text{ for all } i \in [m],$$

where $\alpha_i(t) \in [\kappa_1, \kappa_2]$. Hence, if $T_i > T^*$, then $\|\mathbf{w}_i(T^*)\|_2 < \infty$, which implies

$$\lim_{t \rightarrow T^*} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{j=1}^m \|\mathbf{w}_j(t)\|_2^2}} = \lim_{t \rightarrow T^*} \frac{\|\mathbf{w}_i(T^*)\|_2}{\sqrt{\sum_{j=1}^m \|\mathbf{w}_j(T^*)\|_2^2}} = 0.$$

Next, if $T_i = T^*$, then

$$\lim_{t \rightarrow T^*} \frac{\|\mathbf{w}_i(t)\|_2}{\sqrt{\sum_{j=1}^m \|\mathbf{w}_j(t)\|_2^2}} = \lim_{t \rightarrow T^*} \frac{\alpha_i(t)}{\sqrt{\sum_{j=1}^m \alpha_j^2(t) \left(\frac{T^*-t}{T_j-t}\right)^{2/(L-2)}}} \geq \frac{\kappa_1}{\sqrt{m}\kappa_2} > 0.$$

□

D.5 Equivalence between OMP and NP for Diagonal Linear Networks

In this subsection, we show that for two-layer diagonal linear networks, the NP algorithm is equivalent to OMP, under certain assumptions. For an input $\mathbf{x} \in \mathbb{R}^d$, the network output is

$$\mathcal{H}(\mathbf{x}; \mathbf{v}, \mathbf{u}) = \mathbf{x}^\top (\mathbf{v} \odot \mathbf{u}),$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ are the weights of first and second layer, respectively. Thus, there are d number of neurons, where u_j and v_j , the j th entry of \mathbf{u} and \mathbf{v} , represent the incoming and outgoing weights of the j th neuron. Suppose $\{\mathbf{x}_i, y_i\}_{i=1}^n$ is the training dataset, and let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$, $\mathbf{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$. The training loss can be written as

$$\mathcal{L}(\mathbf{v}, \mathbf{u}) = \frac{1}{2} \sum_{i=1}^n (\mathbf{x}_i^\top (\mathbf{v} \odot \mathbf{u}) - y_i)^2 = \frac{1}{2} \|\mathbf{X}(\mathbf{v} \odot \mathbf{u}) - \mathbf{y}\|_2^2.$$

Note that, the j th neuron only takes the j th coordinate as input. Therefore, unlike feed-forward neural network, the neurons here can not be exchanged, without changing the network output. As a result, the constrained NCF would be different for each neuron, even though they are in the same layer. Now, let us look at the iterations of NP algorithm for this network, starting with the first iteration.

First iteration: We begin by maximizing the constrained NCF for each neuron:

$$\max_{v_j^2 + u_j^2 = 1} v_j u_j \mathbf{X}[:, j]^\top \mathbf{y}, \text{ for all } j \in [n].$$

Each problem admits a closed-form expression of its global maximum: $|\mathbf{X}[:, j]^\top \mathbf{y}|/2$, for $(\bar{v}_j, \bar{u}_j) = (1/\sqrt{2}, \text{sign}(\mathbf{X}[:, j]^\top \mathbf{y})/\sqrt{2})$. Let $j_1 = \arg \max_{j \in [n]} |\mathbf{X}[:, j]^\top \mathbf{y}|$, then the most dominant KKT point corresponds to the j_1 th neuron. Thus, we minimize the following training loss via gradient descent:

$$\mathcal{L}(u_{j_1}, v_{j_1}) = \frac{1}{2} \|\mathbf{X}[:, j_1] v_{j_1} u_{j_1} - \mathbf{y}\|_2^2,$$

with initialization $(v_{j_1}(0), u_{j_1}(0)) = (\delta/\sqrt{2}, \delta \text{sign}(\mathbf{X}[:, j_1]^\top \mathbf{y})/\sqrt{2})$.

In Lemma 21, we prove that for sufficiently small δ and step-size, gradient descent converges to the global minimum. Importantly, aside from the origin and global minimum, there are no spurious stationary points. Moreover, the global minimum of the above problem is same as the global minimum of

$$g(\beta) = \frac{1}{2} \|\mathbf{X}[:, j_1] \beta - \mathbf{y}\|_2^2,$$

which is a convex problem. If β^* is its optimal solution, then the residual $\bar{\mathbf{y}} = \mathbf{y} - \mathbf{X}[:, j_1] \beta^*$. In summary, maximizing the constrained NCF for each neuron is equivalent to identifying which column of \mathbf{X} has highest absolute correlation with \mathbf{y} . Then, that column is used to find the best fit for the labels \mathbf{y} . These are exactly the steps of the first iteration of OMP Pati et al. (1993).

Second iteration: Let $(v_{j_1}^*, u_{j_1}^*)$ denote the weights after the first iteration. We next maximize the constrained NCF for each remaining neuron:

$$\max_{v_j^2 + u_j^2 = 1} v_j u_j \mathbf{X}[:, j]^\top \bar{\mathbf{y}}, \text{ for all } j \in [n] \setminus \{j_1\}.$$

Again, the closed-form maximum is $|\mathbf{X}[:, j]^\top \bar{\mathbf{y}}|/\sqrt{2}$, for $(\bar{v}_j, \bar{u}_j) = (1/\sqrt{2}, \text{sign}(\mathbf{X}[:, j]^\top \bar{\mathbf{y}})/\sqrt{2})$. Let $j_2 = \arg \max_{j \in [n] \setminus \{j_1\}} |\mathbf{X}[:, j]^\top \bar{\mathbf{y}}|$, then the most dominant KKT point corresponds to the j_2 th neuron. Thus, we minimize the following training loss via gradient descent:

$$\mathcal{L}(u_{j_1}, v_{j_1}, u_{j_2}, v_{j_2}) = \frac{1}{2} \|\mathbf{X}[:, j_1]v_{j_1}u_{j_1} + \mathbf{X}[:, j_2]v_{j_2}u_{j_2} - \mathbf{y}\|_2^2,$$

with initialization $(v_{j_2}(0), u_{j_2}(0)) = (\delta/\sqrt{2}, \delta \text{sign}(\mathbf{X}[:, j_2]^\top \bar{\mathbf{y}})/\sqrt{2})$ and $(v_{j_1}(0), u_{j_1}(0)) = (v_{j_1}^*, u_{j_1}^*)$.

Unlike the first iteration, here additional stationary points appear, making global convergence of gradient descent to global minimum harder to prove. Nevertheless, if we assume convergence to the global minimum, then this step is equivalent to finding the the global minimum of

$$g(\beta_1, \beta_2) = \frac{1}{2} \|\mathbf{X}[:, j_1]\beta_1 + \mathbf{X}[:, j_2]\beta_2 - \mathbf{y}\|_2^2.$$

Thus, in the second iteration, maximizing the constrained NCF for each remaining neuron is equivalent to identifying which remaining column of \mathbf{X} has highest absolute correlation with the residual error $\bar{\mathbf{y}}$. Then, this new column, along with the previous one, is used to find the best fit for the labels \mathbf{y} . This is exactly same as the second iteration of the OMP algorithm.

The subsequent iterations are same as second iteration. It also matches with the subsequent iterations of OMP, under the same assumption: the gradient descent iterates converge to the global minimum. In summary, for two-layer diagonal linear networks, the NP algorithm is equivalent to OMP, provided gradient descent reaches the global minimum at every iteration.

Experiment: We next empirically demonstrate that the NP algorithm for two-layer diagonal networks does match with the OMP solution and is different from the minimum ℓ_1 -norm solution. Suppose

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & -0.1 \\ 0 & 1 & 1 + \frac{0.2}{3} \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Minimizing ℓ_1 -norm. Consider the following optimization problem:

$$\min_{\mathbf{z}} \|\mathbf{z}\|_1, \text{ such that } \mathbf{Xz} = \mathbf{b}.$$

The solution of the above optimization problem is $\mathbf{z}^* = [1, 2, 0]^\top$, where $\|\mathbf{z}^*\|_1 = 3$.

OMP. In the first iteration of OMP, the third column has maximum absolute correlation with \mathbf{b} , which is used to find the best fit for \mathbf{b} . This produces

$$\mathbf{z}_1 = [0, 0, 1.772].$$

In the second iteration, the first column has maximum absolute correlation with the residual error, which is used, along with the third column, to find the best fit for \mathbf{b} . This produces

$$\mathbf{z}_2 = [1.1875, 0, 1.875],$$

which exactly fits \mathbf{b} , and the algorithm stops here. Thus, OMP finds a sparse solution different from the minimum ℓ_1 -norm solution.

NP. In the first iteration, third neuron has the most dominant KKT point. Thus, we minimize

$$\mathcal{L}(u_3, v_3) = \frac{1}{2} \|\mathbf{X}[:, 3]v_3u_3 - \mathbf{b}\|_2^2,$$

via gradient descent with step-size 0.001 and initialization $(v_3(0), u_3(0)) = (\delta/\sqrt{2}, \delta/\sqrt{2})$, where $\delta = 0.01$. The gradient descent converges to $(v_3^*, u_3^*) = (1.33, 1.33)$, implying $v_3^*u_3^* = 1.77$. Hence, output of the first iteration is same as the first iteration of OMP.

In the second iteration, first neuron has the most dominant KKT point. Thus, we minimize

$$\mathcal{L}(u_1, v_1, u_3, v_3) = \frac{1}{2} \|\mathbf{X}[:, 1]v_1u_1 + \mathbf{X}[:, 3]v_3u_3 - \mathbf{b}\|_2^2,$$

via gradient descent with step-size 0.001 and initialization $(v_3(0), u_3(0)) = (1.33, 1.33)$, $(v_1(0), u_1(0)) = (\delta/\sqrt{2}, \delta/\sqrt{2})$, where $\delta = 0.01$. The gradient descent converges to $(v_3^*, u_3^*) = (1.37, 1.37)$, $(v_1^*, u_1^*) = (1.09, 1.09)$, implying $v_3^*u_3^* = 1.876$, $v_1^*u_1^* = 1.188$. Hence, output of the second iteration is also same as the second iteration of OMP. Therefore, the NP algorithm learns the same output as OMP, which is different from the minimum ℓ_1 -norm solution.

Lemma 21. *Consider minimizing the following optimization problem via gradient descent:*

$$\mathcal{L}(u, v) := \frac{1}{2} \|\mathbf{x}vu - \mathbf{y}\|_2^2,$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and, without loss of generality, let $\|\mathbf{x}\|_2 = 1$. Suppose the initial weights are $(v(0), u(0)) = (\delta/\sqrt{2}, \delta \text{sign}(\mathbf{x}^\top \mathbf{y})/\sqrt{2})$. Then, for sufficiently small $\delta > 0$ and sufficiently small step-size, the gradient descent converges to the global minimum, that is, $v(\infty)u(\infty) = \mathbf{x}^\top \mathbf{y}$.

Proof. We first show that the stationary points of $\mathcal{L}(u, v)$ are either the origin or the global minimum. At any stationary point (\bar{u}, \bar{v}) , we have

$$\begin{aligned} 0 &= \nabla_v \mathcal{L}(\bar{u}, \bar{v}) = \bar{u} \mathbf{x}^\top (\mathbf{x}\bar{v}\bar{u} - \mathbf{y}) = \bar{u}(\bar{v}\bar{u} - \mathbf{x}^\top \mathbf{y}), \\ 0 &= \nabla_u \mathcal{L}(\bar{u}, \bar{v}) = \bar{v} \mathbf{x}^\top (\mathbf{x}\bar{v}\bar{u} - \mathbf{y}) = \bar{v}(\bar{v}\bar{u} - \mathbf{x}^\top \mathbf{y}). \end{aligned}$$

The origin is clearly a stationary point. If either \bar{v} or \bar{u} is non-zero, then $\bar{v}\bar{u} = \mathbf{x}^\top \mathbf{y}$, that is, the stationary point is a global minimum.

The next set of inequalities show that for all sufficiently small $\delta > 0$, the loss at initialization is strictly smaller than at the origin.

$$\begin{aligned} \mathcal{L}(u(0), v(0)) &= \frac{1}{2} (u(0)^2 v(0)^2 - 2u(0)v(0)\mathbf{x}^\top \mathbf{y} + \|\mathbf{y}\|_2^2) \\ &= \frac{1}{2} \left(\frac{\delta^4}{4} - \delta^2 |\mathbf{x}^\top \mathbf{y}| + \|\mathbf{y}\|_2^2 \right) \\ &\leq \frac{1}{2} \left(-\frac{\delta^2}{2} |\mathbf{x}^\top \mathbf{y}| + \|\mathbf{y}\|_2^2 \right) \leq \|\mathbf{y}\|_2^2 / 2 = \mathcal{L}(0, 0), \end{aligned}$$

where the first inequality is true for all sufficiently small $\delta > 0$.

We next show that for sufficiently small $\delta > 0$ and sufficiently small step-size, the gradient descent converges to a stationary point while the loss decreases at each step. Since under these conditions convergence to the origin is impossible, the iterates must approach the global minimum.

We first prove some key properties of gradient descent. The gradient descent iterates can be written as

$$\begin{aligned} v(t+1) &= v(t) - \eta u(t) \mathbf{x}^\top (\mathbf{x}v(t)u(t) - \mathbf{y}) = v(t) - \eta u(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}), \\ u(t+1) &= u(t) - \eta v(t) \mathbf{x}^\top (\mathbf{x}v(t)u(t) - \mathbf{y}) = u(t) - \eta v(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}). \end{aligned}$$

If $\mathbf{x}^\top \mathbf{y} > 0$, then $v(t) = u(t)$, for all $t \geq 0$. This can be shown via induction. It is trivially true for $t = 0$. Suppose it holds for some $t \geq 0$. Then,

$$v(t+1) = v(t) - \eta u(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}) = u(t) - \eta v(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}) = u(t+1),$$

where the second equality uses $v(t) = u(t)$. This proves the claim.

If $\mathbf{x}^\top \mathbf{y} < 0$, then $v(t) = -u(t)$, for all $t \geq 0$. This also can be shown via induction. It is trivially true for $t = 0$. Suppose it holds for some $t \geq 0$. Then,

$$v(t+1) = v(t) - \eta u(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}) = -u(t) + \eta v(t)(v(t)u(t) - \mathbf{x}^\top \mathbf{y}) = -u(t+1),$$

where the second equality uses $v(t) = -u(t)$, which proves the claim.

We next show that $|v(t)| \leq \sqrt{|\mathbf{x}^\top \mathbf{y}|}$, if $\eta \in (0, 1/(2|\mathbf{x}^\top \mathbf{y}|))$ and $\delta \in (0, \sqrt{2|\mathbf{x}^\top \mathbf{y}|})$. If $\mathbf{x}^\top \mathbf{y} > 0$, then $v(t) = u(t)$, implying

$$v(t+1) = v(t) - \eta v(t)(v^2(t) - \mathbf{x}^\top \mathbf{y}) = v(t)(1 - \eta(v^2(t) - \mathbf{x}^\top \mathbf{y})).$$

Let $h(v) := v(1 - \eta(v^2 - \mathbf{x}^\top \mathbf{y}))$. Then, $h(0) = 0$ and $h(\sqrt{\mathbf{x}^\top \mathbf{y}}) = \sqrt{\mathbf{x}^\top \mathbf{y}}$. Also, for all $v \in [0, \sqrt{\mathbf{x}^\top \mathbf{y}}]$,

$$h'(v) = (1 - \eta(v^2 - \mathbf{x}^\top \mathbf{y})) - 2\eta v^2 = 1 - 3\eta v^2 + \eta \mathbf{x}^\top \mathbf{y} \geq 1 - 2\eta \mathbf{x}^\top \mathbf{y} > 0,$$

where the last inequality is true since $\eta < 1/(2\mathbf{x}^\top \mathbf{y})$. Thus, if $v \in [0, \sqrt{\mathbf{x}^\top \mathbf{y}}]$, then $h(v) \in [0, \sqrt{\mathbf{x}^\top \mathbf{y}}]$. Since $v(t+1) = h(v(t))$ and $v(0) = \delta/\sqrt{2} < \sqrt{\mathbf{x}^\top \mathbf{y}}$, we get $|v(t)| \leq \sqrt{|\mathbf{x}^\top \mathbf{y}|}$, for all $t \geq 0$.

Next, if $\mathbf{x}^\top \mathbf{y} < 0$, then $v(t) = -u(t)$ and

$$v(t+1) = v(t) + \eta v(t)(-v^2(t) - \mathbf{x}^\top \mathbf{y}) = v(t)(1 - \eta(v^2(t) + \mathbf{x}^\top \mathbf{y})).$$

Let $g(v) := v(1 - \eta(v^2 + \mathbf{x}^\top \mathbf{y}))$. Then, $g(0) = 0$ and $g(\sqrt{-\mathbf{x}^\top \mathbf{y}}) = \sqrt{-\mathbf{x}^\top \mathbf{y}}$. Also, for all $v \in [0, \sqrt{-\mathbf{x}^\top \mathbf{y}}]$,

$$g'(v) = (1 - \eta(v^2 + \mathbf{x}^\top \mathbf{y})) - 2\eta v^2 = 1 - 3\eta v^2 - \eta \mathbf{x}^\top \mathbf{y} \geq 1 + 2\eta \mathbf{x}^\top \mathbf{y} \geq 0,$$

where the last inequality is true since $\eta \leq 1/(-2\mathbf{x}^\top \mathbf{y})$. Thus, if $v \in [0, \sqrt{-\mathbf{x}^\top \mathbf{y}}]$, then $g(v) \in [0, \sqrt{-\mathbf{x}^\top \mathbf{y}}]$. Since $v(t+1) = g(v(t))$ and $v(0) = \delta/\sqrt{2} \leq \sqrt{-\mathbf{x}^\top \mathbf{y}}$, we get $|v(t)| \leq \sqrt{|\mathbf{x}^\top \mathbf{y}|}$, for all $t \geq 0$.

Since the gradient descent iterates remain bounded, the loss $\mathcal{L}(v, u)$ has Lipschitz gradient along the entire trajectory, for a sufficiently large Lipschitz constant. Consequently, as shown in Nesterov (2013, Chapter 1.2.3), gradient descent with a sufficiently small step size converges to a stationary point while the loss decreases at each iteration. Moreover, for all sufficiently small δ , the loss at initialization is strictly lower than the loss at the origin. Therefore, gradient descent cannot converge to the origin and must instead converge to the global minimum. This completes the proof. \square

D.6 Impact of Rescaling the Weights

Consider training a three-layer ReLU network with the NP algorithm. Suppose $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$ is the set of learned weights at the end of some iteration, and let $\bar{\mathbf{y}}$ be the corresponding residual error. To add a neuron in the next iteration, the following two objectives are maximized via projected gradient ascent:

$$\max_{\|\mathbf{b}_3\|_2^2 + \|\mathbf{a}_2\|_2^2 = 1} \sum_{i=1}^n \bar{y}_i \mathbf{b}_3 \sigma(\mathbf{a}_2^\top \sigma(\bar{\mathbf{W}}_1 \mathbf{x}_i)), \quad \max_{\|\mathbf{b}_2\|_2^2 + \|\mathbf{a}_1\|_2^2 = 1} \sum_{i=1}^n \bar{y}_i \bar{\mathbf{W}}_3 \text{diag}(\sigma'(\bar{\mathbf{W}}_2 \sigma(\bar{\mathbf{W}}_1 \mathbf{x}_i))) \mathbf{b}_2 \sigma(\mathbf{a}_1^\top \mathbf{x}_i)$$

Although $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$ is a stationary point of the training loss, it is not isolated: symmetries present in the network generate continuous manifolds of stationary points. For example, for any $c > 0$, $(c\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3/c)$ is also a stationary point of the training loss.

Now, suppose instead of $(\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3)$, the set of learned weights are $(c\bar{\mathbf{W}}_1, \bar{\mathbf{W}}_2, \bar{\mathbf{W}}_3/c)$, for some $c > 0$. We next describe the impact of this rescaling of the weights on the addition of the neuron in the next iteration. Note that, the network output does not change because

$$\sigma(\bar{\mathbf{W}}_3/c \sigma(c\bar{\mathbf{W}}_2 \sigma(c\bar{\mathbf{W}}_1 \mathbf{x}))) = \sigma(\bar{\mathbf{W}}_3 \sigma(\bar{\mathbf{W}}_2 \sigma(\bar{\mathbf{W}}_1 \mathbf{x}))),$$

where the equality follows from 1-positive homogeneity of the ReLU activation. Hence, the residual error is unchanged by this rescaling. To add a neuron in this case, the following two function will be maximized via projected gradient ascent:

$$\max_{\|\mathbf{b}_3\|_2^2 + \|\mathbf{a}_2\|_2^2 = 1} c \sum_{i=1}^n \bar{y}_i \mathbf{b}_3 \sigma(\mathbf{a}_2^\top \sigma(\bar{\mathbf{W}}_1 \mathbf{x}_i)), \quad \max_{\|\mathbf{b}_2\|_2^2 + \|\mathbf{a}_1\|_2^2 = 1} \frac{1}{c} \sum_{i=1}^n \bar{y}_i \bar{\mathbf{W}}_3 \text{diag}(\sigma'(\bar{\mathbf{W}}_2 \sigma(\bar{\mathbf{W}}_1 \mathbf{x}_i))) \mathbf{b}_2 \sigma(\mathbf{a}_1^\top \mathbf{x}_i),$$

where we used 0-homogeneity of $\sigma'(x)$. Thus, one objective is multiplied by c and the other by $1/c$. If c deviates sufficiently from 1, the most dominant KKT point in this case can be different from the previous case, and consequently a different neuron will be added. In short, although rescaling preserves network output and stationarity, it can alter which neurons the NP algorithm chooses to add.

The above discussion raises a natural question: what is the appropriate scaling of the weights? To answer this, we return to training via gradient flow. For ReLU activation and small initialization, the norm of incoming and outgoing weights of each hidden neuron remains nearly balanced throughout training (Du et al., 2018). In fact, if weights are balanced at initialization, they stay balanced during the training. Consequently, an appropriate scaling is to ensure that the weights of each hidden neuron are balanced. However, gradient descent with large step-size and/or large number of iterations can produce unbalance in the weights. For such unbalanced weights, Saul (2023) provides an algorithm that rescales the weight to get balanced weights, without altering the network output.

The NP algorithm also contains implicit mechanisms that encourages balancedness in the weights. In the first iteration, the weights are initialized along a KKT point of the constrained NCF and then minimized via gradient descent. From Lemma 3, the weights at the KKT points are balanced. Hence, if the step-size is small, the weights can be expected to remain balanced during gradient descent iterations. In later iterations, the incoming and outgoing weights of the newly added neurons are along a KKT point of an appropriate constrained NCF, which are also balanced. Thus, if the weights at the end of the previous iteration were balanced, then the weights would remain balanced after adding the neuron. Hence, if the step-size is small, the weights can be expected to remain balanced during gradient descent iterations. Nevertheless, if the step-size is large and/or large number of iterations are used, the weights could become unbalanced. This can be rectified by using the algorithm in Saul (2023) at the end of each iteration of NP.

Finally, although our discussion focused on three-layer ReLU networks, the arguments extend to deeper networks. For activation functions of the form $\sigma(x) = \max(x, \alpha x)^p$ with $p \geq 2$, the only quantitative change is that the norm of incoming weights should be \sqrt{p} times the outgoing weights, rather than being equal. Note, however, that the balancing algorithm in Saul (2023) is designed for ReLU/Leaky ReLU; how to generalize it to $\max(x, \alpha x)^p$ is unclear. In our experiments for the PVR task, which uses ReLU activation, we balance the weights after every iteration of NP; in other experiments we found that weights typically remain nearly balanced throughout training.