The Minimum-Norm Gauge for Deep Learning

Anonymous authors
Paper under double-blind review

Abstract

Feedforward neural networks with homogeneous activation functions possess a gauge symmetry: the functions they compute do not change when the incoming and outgoing weights at any hidden unit are rescaled by reciprocal positive values. There are other important properties of these networks, however, that are not invariant under such transformations. For example, deep networks with highly unbalanced weights may be slower to train or harder to compare and interpret. We describe a simple procedure for gauge-fixing in these networks; this procedure computes multiplicative rescaling factors—one at each hidden unit—that rebalance the weights of these networks without changing the end-to-end functions that they compute. Specifically, given an initial network with arbitrary weights, the procedure determines the functionally equivalent network whose weights are as small as possible (as measured by their ℓ_p-norm); this transformed network also has the property that the norms of incoming and outgoing weights at each hidden unit are exactly balanced. The rescaling factors that perform this transformation are found by solving a convex optimization, and we derive simple multiplicative updates that provably converge to its solution. Next we analyze the optimization landscape in these networks and derive conditions under which this minimum-norm solution is preserved during learning. Finally we explore the effects of gauge-fixing on the speed and outcomes of learning by stochastic gradient descent. On multiple problems in classification we find that gauge-fixing leads to faster descent in the regularized log-loss.

1 Introduction

Many recent studies of deep learning have focused on the important role of symmetries (Bronstein et al., 2021; Kunin et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). In large part these studies were inspired by the role that symmetries play in our understanding of the physical world (Anderson, 1972; Zee, 2016). Of particular interest, in both physics and machine learning, are so-called *gauge* symmetries (Gross, 1992); these are symmetries that arise when a model is overparameterized—that is, when the model is formulated or expressed in terms of more parameters than its essential degrees of freedom.

One such model in machine learning is a feedforward neural network with rectified linear hidden units. Such a network is specified by the values of its weights, but the function it computes does not change when the incoming and outgoing weights at any hidden unit are inversely rescaled by some positive value (Glorot et al., 2011). This invariance is illustrated in Fig. 1. In this case, the gauge symmetry arises from the freedom to choose an arbitrary rescaling factor at each hidden unit.

This particular symmetry of deep learning has already led to many important findings. For example, it is known that this symmetry gives rise to a conservation law: at each hidden unit, there is a synaptic balance—defined as the difference in the sums of squared incoming and outgoing weights—that does not change when networks are trained by gradient flow (i.e., gradient descent in the limit of an infinitesimally small learning rate) (Du et al., 2018). From this conservation law follows another important observation: if the weights are initially balanced across layers, then they remain so during training, a key condition for proving certain

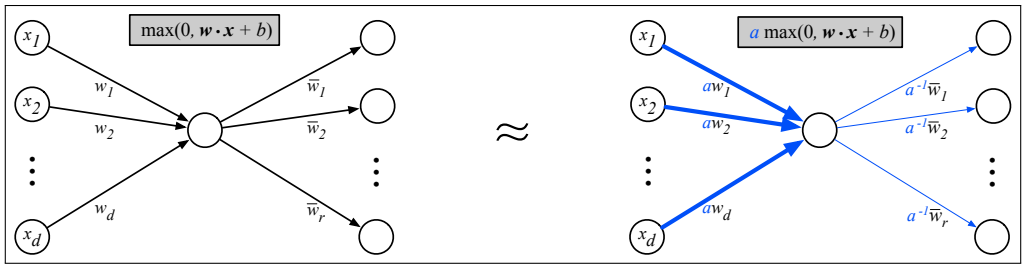

Figure 1: A rectified linear unit (ReLU) has the same effect if its incoming weights $\mathbf{w}$ and bias b are rescaled by some factor $a > 0$ while its outgoing weights $\overline{\mathbf{w}}$ are rescaled by the inverse factor a^{-1}.

convergence results (Arora et al., 2019). It is also possible, by analyzing the synaptic flows across adjacent layers, to devise more powerful pruning algorithms (Tanaka et al., 2020). Finally, a number of authors have proposed more sophisticated learning rules that are invariant to these rescaling transformations (Neyshabur et al., 2015a; Meng et al., 2019) or that break this invariance in purposeful ways (Badrinarayanan et al., 2015; Armenta et al., 2021; Zhao et al., 2022). These latter works highlight an important distinction: though the functions computed by deep networks may be invariant to rescaling transformations, there are other important properties of these networks—such as the speed and eventual outcomes of learning—that are not.

Notwithstanding these contributions, as well as progress on many other fronts (Arora et al., 2018; Belkin et al., 2019; Papyan et al., 2020; Zhang et al., 2021a), our theoretical understanding of deep learning has not kept pace with its demand in real-world applications (LeCun et al., 2015). This gap provides the larger context for this work. Large multilayer networks can be fickle to train (Glorot & Bengio, 2010; Sutskever et al., 2013) and hard to interpret (Zhang et al., 2021b). Despite these challenges—which are often magnified for new or unfamiliar tasks—most practitioners are still drawn to highly overparameterized models. In the further development and analysis of these models, there may be no tool more broadly useful than symmetry.

To proceed, we consider how similar ideas have been developed to study the physical world. Perhaps the most familiar gauge symmetry arises in classical electrodynamics (Jackson, 2002). Electric and magnetic fields are often calculated from their corresponding potentials; these are the scalar electric potential $\Phi(\mathbf{r}, t)$ and the magnetic vector potential $\mathbf{A}(\mathbf{r}, t)$. When the fields are expressed in terms of these potentials, Maxwell's equations are invariant under the gauge transformation

$$\begin{aligned} \Phi &\leftarrow \Phi - \tfrac{\partial \chi}{\partial t}, \\ \mathbf{A} &\leftarrow \mathbf{A} + \nabla \chi, \end{aligned} \tag{1}$$

where $\chi(\mathbf{r}, t)$ is any twice-differentiable function of space and time. In practice, this gauge degree of freedom is often *fixed* to simplify or highlight certain physics of interest. For example, one choice is the Coulomb gauge, satisfying $\nabla \cdot \mathbf{A} = 0$, which has the property that it minimizes the volume integral (Gubarev et al., 2001) of $\|\mathbf{A}\|^2$. Another choice is the Lorenz gauge, satisfying $\nabla \cdot \mathbf{A} = \frac{1}{c} \frac{\partial \varphi}{\partial t}$ (where c is the speed of light), which simplifies certain wave equations. It is natural to ask if there are analogous gauge-fixing conditions for the weights of deep neural networks—for example, a rescaling transformation that *minimizes the norm of the weights*, or that *simplifies the dynamics of learning*. We will see that in both cases the answer is yes; in fact, there is a single rescaling transformation that achieves both these objectives. It is also one that is simple to formulate and compute.

With these goals in mind, this paper investigates a family of norm-minimizing gauge transformations for feedforward networks with rescaling symmetries. These gauge transformations are designed to minimize the ℓ_p-norm of the weights in a network without changing the end-to-end function that the network computes. (The value of $p > 0$ may be chosen by the practitioner.) A particular gauge transformation is specified by a set of multiplicative rescaling factors, one for each hidden unit of the network. The paper's main technical

contribution is show how to obtain these rescaling factors by solving a problem in convex optimization. More concretely, we derive simple multiplicative updates for performing this optimization, analogous to those for nonnegative matrix factorization (Lee & Seung, 1999; Gillis, 2021), and we prove that these updates converge monotonically to fixed points that represent minimum-norm solutions (Theorem 2.1). Notably, these multiplicative updates are parameter-free in the sense that they do not require the setting or tuning of learning rates.

Norm-minimizing gauge transformations also have the interesting property that they rebalance the weights of a network in a special way. In particular, we obtain the following result (Lemma 2.2): given a network with arbitrary weights, the norm-minimizing gauge transformation is also the gauge transformation that equates the p-norms of incoming and outgoing weights at each hidden unit in the network. The balancing of weights across layers has been a key tool for proving convergence theorems in deep learning (Du et al., 2018; Arora et al., 2019). Thus, a second contribution of this paper is to show that for a large class of multilayer networks, this balance can be restored at any time during the learning process by a well-chosen gauge transformation. In particular, this can be done no matter how the weights of the network were initialized and subsequently adapted.

A final contribution of this paper is to explore the effects of gauge-fixing on the dynamics of learning (and vice-versa). Here we identify a natural pairing of gauge-fixing conditions and regularization penalties for deep learning in networks with homogeneous activation functions. In these networks, we show (Theorem 3.4) that if the gauge and regularizer are appropriately paired (e.g., minimum ℓ_2-norm gauge-fixing and ℓ_2-norm weight decay), then the gauge-fixing condition is preserved by the dynamics of gradient flow in the regularized loss function. Minimum-norm gauge transformations also provide an experimental probe for the implicit effect of weight decay on learning rates (van Laarhoven, 2017). In practice, deep networks are not trained by gradient flow, so for empirical results, we investigate how gauge-fixing affects the performance of stochastic gradient descent (SGD). On multiple data sets we find that gauge-fixing leads to consistently faster descent in the regularized loss.

2 Minimum-norm gauge fixing

In this section we consider how to shrink the weights in feedforward networks with homogeneous activation functions without changing the end-to-end function that these networks compute. Specifically, section 2.1 describes the symmetry group of rescaling transformations in these networks, section 2.2 presents multiplicative updates to optimize over the elements of this group, section 2.3 proves the convergence of these updates to a global minimizer, and section 2.4 illustrates how these updates work in practice.

2.1 Preliminaries

Our interest lies in feedforward networks that parameterize vector-valued functions $f : \Re^d \to \Re^r$. The following notation will be useful. We denote the indices of a network's input, hidden, and output units, respectively, by $\mathcal{I}$, $\mathcal{H}$, and $\mathcal{O}$, and we order the units so that $\mathcal{I} = \{1, \ldots, d\}$, $\mathcal{H} = \{d+1, \ldots, n-r\}$, and $\mathcal{O} = \{n-r+1, \ldots, n\}$ where n is the total number of units. Let $\mathbf{x} \in \Re^d$ denote an input to the network and $f(\mathbf{x}) \in \Re^r$ its corresponding output. The mapping $\mathbf{x} \to f(\mathbf{x})$ is determined by the network's weight matrix $\mathbf{W} \in \Re^{n \times n}$, biases $\mathbf{b} \in \Re^n$, and activation functions $g_i : \Re \to \Re$ at each non-input unit (where $i \in \mathcal{H} \cup \mathcal{O}$). The mapping $\mathbf{x} \to \mathbf{f}(\mathbf{x})$ can be computed by the feedforward procedure that sequentially activates all of the units in the network—that is, setting $h_i = x_i$ for the input units, propagating

$$h_i = g_i\Big(\sum_j W_{ij} h_j + b_i\Big) \tag{2}$$

to the remaining units, and setting $f_i(\mathbf{x}) = h_{n+i-r}$ according to the i^{th} output unit. Since the network is feedforward, its weight matrix is strictly lower triangular. For simplicity, we assume that the output units of the network are linear (with $g_i(z) = z$ for all $i \in \mathcal{O}$) and unconnected (with $W_{ij} = 0$ if $j \in \mathcal{O}$ and $i > j$).

A rescaling symmetry arises at each hidden unit of the network whose activation function is positive homogeneous of degree one (Neyshabur et al., 2015a; Dinh et al., 2017)—that is, whose activation function satisfies

$$g_i(az) = ag_i(z) \tag{3}$$

for all $a>0$. In this paper we focus on networks of rectified linear hidden units (ReLUs), with the activation function $g_i(z) = \max(0, z)$ at all $i \in \mathcal{H}$, but the property in eq. (3) is also satisfied by linear units (Saxe et al., 2013), leaky ReLUs (He et al., 2015), and maxout units (Goodfellow et al., 2013). As illustrated in Fig. 1, when a hidden unit's activation function satisfies eq. (3), the function f computed by the network does not change when the unit's incoming weights (including the bias) and the outgoing weights are rescaled by reciprocal amounts. Note that in layered architectures with homogeneous activation functions, the network's overall mapping f will also be a homogeneous function of its parameters (i.e., weights and biases) (Lyu & Li, 2020) with a degree equal to the depth of the network; we refer to such architectures as *homogeneous networks*. Our theoretical results in this paper apply generally to feedforward networks with homogeneous activation functions, but our empirical investigations focus on the special case of layered architectures.

In these networks, a gauge transformation is specified by a set of rescaling factors, one for each hidden unit. We use

$$\mathcal{A} = \{\mathbf{a} \in \Re^n \,|\, a_i > 0 \text{ if } i \in \mathcal{H}, a_i = 1 \text{ otherwise}\} \tag{4}$$

to denote the set of these gauge transformations. Then under a particular rescaling, represented by some $\mathbf{a} \in \mathcal{A}$, the network's weights and biases are transformed multiplicatively as

$$W_{ij} \leftarrow W_{ij}(a_i/a_j), \tag{5}$$

$$b_i \leftarrow b_i a_i. \tag{6}$$

It may seem redundant in eq. (4) to introduce rescaling factors at non-hidden units only to constrain them to be equal to one. With this notation, however, we can express the transformations in eqs. (5–6) without distinguishing between the network's different types of units. As shorthand, we write $(\mathbf{W}', \mathbf{b}') \sim (\mathbf{W}, \mathbf{b})$ to denote that two networks are equivalent up to some (unspecified) rescaling, and we write $(\mathbf{W}', \mathbf{b}') \overset{\mathbf{a}}{\sim} (\mathbf{W}, \mathbf{b})$ to denote that they are equivalent up to the particular rescaling $\mathbf{a} \in \mathcal{A}$.

2.2 Multiplicative updates

We can now state the first main result of this paper; it is a solution to the following problem. Let $p > 0$, and let

$$\|\mathbf{W}\|_{p,p} = \left(\sum_{ij} |W_{ij}|^p \right)^{1/p} \tag{7}$$

denote the entry-wise p-norm of the weight matrix $\mathbf{W}$. Given an initial weight matrix $\mathbf{W}_0$, we consider how to compute the gauge transformation in eqs. (5–6) that minimizes the p-norm in eq. (7). Algorithm 1 describes an iterative procedure to compute this *norm-minimizing* gauge transformation via a sequence of multiplicative updates of the form in eqs. (5–6). For each update, the key step is to compute the rescaling factor a_i at each hidden unit $i \in \mathcal{H}$ from a ratio comparing the magnitudes of its ingoing and outgoing weights; this step is derived later in eq. (12). Our first main result is contained in the following theorem:

Theorem 2.1 (Convergence of Multiplicative Updates)**.** *For all $\mathbf{W}_0 \in \Re^{n \times n}$ and $\mathbf{b}_0 \in \Re^n$, the multiplicative updates in Algorithm 1 converge to a global minimizer of the entry-wise p-norm*

$$\operatorname*{argmin}_{\mathbf{W},\mathbf{b}} \|\mathbf{W}\|_{p,p} \quad \text{such that} \quad (\mathbf{W}, \mathbf{b}) \sim (\mathbf{W}_0, \mathbf{b}_0). \tag{8}$$

In addition, the intermediate solutions from these updates yield monotonically decreasing values for these p-norms.

Algorithm 1 Procedure to compute a *norm-minimizing* gauge transformation for a network with weights $\mathbf{W}_0 \in \Re^{n\times n}$ and biases $\mathbf{b}_0 \in \Re^n$. The procedure computes rescaled weights $\mathbf{W}$ and biases $\mathbf{b}$ via a sequence of multiplicative updates that minimize the sum $\sum_{ij} |W_{ij}|^p$ up to some tolerance $\delta > 0$. The network's hidden units are specified by the set $\mathcal{H}$. The key step per update (shaded) is to compute the rescaling factor a_i at each hidden unit $i \in \mathcal{H}$.

procedure $(\mathbf{W}, \mathbf{b}) = \textsc{MinNorm}(\mathbf{W}_0, \mathbf{b}_0, \mathcal{H}, p, \delta)$
  $\mathbf{W} \leftarrow \mathbf{W}_0$, $\mathbf{b} \leftarrow \mathbf{b}_0$ ▷ *Initialize the rescaled weights and biases.*
  repeat
    for all $i \in \mathcal{H}$ **do** ▷ *Compute the rescaling factor at each hidden unit.*
      $a_i \leftarrow \left(\frac{\sum_{j>i} |W_{ji}|^p}{\sum_{j<i} |W_{ij}|^p} \right)^{\frac{1}{4p}}$
    for all $i \in \mathcal{H}$ **do** ▷ *Rescale the weights and biases.*
      $b_i \leftarrow b_i a_i$
      for all $j < i$ **do**
        $W_{ij} \leftarrow W_{ij}(a_i / a_j)$
  until $\max_i |a_i - 1| < \delta$ ▷ *Iterate until convergence.*

Before proving the theorem, it is worth contrasting the goals and guarantees of this gauge-fixing procedure versus those of typical learning algorithms for deep networks. Most importantly, the updates in Algorithm 1 do *not* constitute a learning algorithm; they only optimize over the set $\mathcal{A}$ of gauge transformations in eq. (4), and they do not change the end-to-end functions of the networks to which they are applied. The main content of the theorem is that the optimization in eq. (8) is considerably more tractable than the problem of learning, and as a result, the multiplicative updates for gauge fixing have much stronger guarantees than (say) learning via back-propagated gradients. In particular, these updates converge monotonically to a global minimizer of the p-norm $\|\mathbf{W}\|_{p,p}$, and they do not require the tuning of any hyperparameters, such as learning rates. One might hope, therefore, that such gauge-fixing procedures—wherever beneficial—could piggyback on top of existing learning algorithms without much extra cost. We explore these ideas further in subsequent sections, but for now, we take up the task of proving Theorem 2.1.

2.3 Proof of convergence

Theorem 2.1 is proved by showing that the multiplicative updates in Algorithm 1 satisfy the preconditions for Meyer's convergence theorem (Meyer, 1976). Meyer's result itself builds on the convergence theory of Zangwill (1969). Our tools are similar to those that have been used to derive the convergence of other multiplicative updates with nonnegativity constraints (Lee & Seung, 1999; Saul et al., 2003; Sha et al., 2007) as well as more general iterative procedures in statistical learning (Dempster et al., 1977; Wu, 1983; Yuille & Rangarajan, 2003; Gunawardana & Byrne, 2005; Sriperumbudur & Lanckriet, 2012). At the same time, we have needed to develop some more specialized machinery, not only for the specific updates in Algorithm 1, but also to to prove convergence to a global (as opposed to local) minimizer.

We prove the theorem via a sequence of lemmas. Our first lemma answers the following question: given a fixed weight matrix $\mathbf{W}$, when is it possible (or more precisely, when *isn't* it possible) to find a matrix $\mathbf{W}' \sim \mathbf{W}$ such that $\|\mathbf{W}'\|_{p,p} < \|\mathbf{W}\|_{p,p}$?

Lemma 2.2 (Weight Balancedness)**.** *Suppose that the p-norms of incoming and outgoing weights at each hidden unit are equal: namely,*

$$\sum_j |W_{ij}|^p = \sum_j |W_{ji}|^p \tag{9}$$

for all $i \in \mathcal{H}$. Then there exists no weight matrix $\mathbf{W}' \sim \mathbf{W}$ such that $\|\mathbf{W}'\|_{p,p} < \|\mathbf{W}\|_{p,p}$.

Proof. Suppose that $\mathbf{W}' \overset{\mathbf{a}}{\sim} \mathbf{W}$, and consider how $\|\mathbf{W}'\|_{p,p}^p$ depends on the gauge transformation $\mathbf{a} \in \mathcal{A}$. This dependence is captured (up to a multiplicative constant) by the continuous function $F : \mathcal{A} \to \Re$, where

$$F(\mathbf{a}) = \frac{1}{p} \sum_{ij} |W_{ij}|^p (a_i/a_j)^p. \tag{10}$$

Note that minimizing F over $\mathcal{A}$ is tantamount to a convex optimization in the transformed variable $\log \mathbf{a} = (\log a_1, \log a_2, ..., \log a_n)$; therefore, any stationary point of F corresponds to a global minimum of F. The partial derivatives of F are given by

$$\frac{\partial F}{\partial a_i} = \frac{1}{a_i} \left[\sum_j |W_{ij}|^p (a_i/a_j)^p - \sum_j |W_{ji}|^p (a_j/a_i)^p \right]. \tag{11}$$

Now suppose that $\sum_j |W_{ij}|^p = \sum_j |W_{ji}|^p$ as in eq. (9). Then from eq. (11), it follows that a global minimum of F is obtained by the *identity* gauge transformation—that is, the gauge transformation with $\mathbf{a} = \mathbf{1}$ where $\mathbf{1} \in \Re^n$ is the vector of all ones. But if a global minimum occurs at $\mathbf{a} = \mathbf{1}$, then by definition there exists no $\mathbf{a}' \in \mathcal{A}$ such that $F(\mathbf{a}') < F(\mathbf{1})$, or equivalently, there exists no $\mathbf{W}' \sim \mathbf{W}$ such that $\|\mathbf{W}'\|_{p,p} < \|\mathbf{W}\|_{p,p}$. □

The previous lemma shows that the balancedness of weights implies the minimality of their norm. The next lemma considers the effect of a single rescaling transformation in the gauge-fixing procedure of Algorithm 1. Here we prove that each update reduces the entry-wise p-norm of the weight matrix.

Lemma 2.3 (Monotone Improvement)**.** *Let $F : \mathcal{A} \to \Re$ be defined as in eq. (10), and let $\tilde{\mathbf{a}} \in \mathcal{A}$ be the vector with elements*

$$\tilde{a}_i = \left(\frac{\sum_{j>i} |W_{ji}|^p}{\sum_{j<i} |W_{ij}|^p} \right)^{\frac{1}{4p}} \tag{12}$$

for $i \in \mathcal{H}$ and $\tilde{a}_i = 1$ otherwise. Then $F(\tilde{\mathbf{a}}) \leq F(\mathbf{1})$, and the inequality is strict unless $F(\mathbf{1})$ is a global minimum with $\tilde{\mathbf{a}} = \mathbf{1}$.

Proof. The proof is based on constructing an auxiliary function, as in the derivations of the Expectation-Maximization algorithm (Dempster et al., 1977), nonnegative matrix factorization (Lee & Seung, 2000), and the convex-concave procedure (Yuille & Rangarajan, 2003). Here we define the auxiliary function $G : \mathcal{A} \to \Re$ by

$$G(\mathbf{a}) = \frac{1}{2p} \sum_{ij} |W_{ij}|^p \left(a_i^{2p} + a_j^{-2p} \right). \tag{13}$$

Note that $(a_i/a_j)^p \leq \frac{1}{2}(a_i^{2p} + a_j^{-2p})$ from the inequality of arithmetic and geometric means. Hence it follows from eq. (10) that

$$F(\mathbf{a}) \leq G(\mathbf{a}) \tag{14}$$

for all $\mathbf{a} \in \mathcal{A}$. The partial derivatives of this auxiliary function are given by

$$\frac{\partial G}{\partial a_i} = \sum_j |W_{ij}|^p a_i^{2p-1} - \sum_j |W_{ji}|^p a_i^{-2p-1}, \tag{15}$$

and for $i \in \mathcal{H}$ they vanish at $\tilde{\mathbf{a}} \in \mathcal{A}$ where the elements of $\tilde{a}_i$ are defined by eq. (12). We claim that this stationary point at $\tilde{\mathbf{a}}$ is the unique global minimizer of G. To see this, note that minimizing G over $\mathcal{A}$ is tantamount to a convex optimization in the variable $\mathbf{a}^p = (a_1^p, a_2^p, ..., a_n^p)$; moreover, this optimization is *strongly* convex because $\sum_j |W_{ij}|^p > 0$ and $\sum_j |W_{ji}|^p > 0$ for all $i \in \mathcal{H}$. (The first condition is necessary for a hidden unit to respond to the network's inputs, and the second is necessary for it to influence the network's outputs; if either fails, then the hidden unit has no effect on the network's input-output mapping,

a situation we can exclude without loss of generality.) From this property of strong convexity, it follows that $\tilde{\mathbf{a}}$ is the unique global minimizer. Combining this observation with eq. (14), we have

$$F(\tilde{\mathbf{a}}) \leq G(\tilde{\mathbf{a}}) \leq G(\mathbf{1}) = F(\mathbf{1}), \tag{16}$$

which proves the first part of the lemma. Now if $\tilde{\mathbf{a}} \neq \mathbf{1}$, then $G(\tilde{\mathbf{a}}) < G(\mathbf{1})$ (since G has a unique global minimizer) and by extension $F(\tilde{\mathbf{a}}) < F(\mathbf{1})$. To prove the second part of the lemma, we suppose that $\tilde{\mathbf{a}} = \mathbf{1}$, or equivalently that $G(\mathbf{1})$ is the minimum of G. Then the partial derivatives in eq. (15) must vanish at $\mathbf{a} = \mathbf{1}$, implying that $\sum_j |W_{ij}|^p = \sum_j |W_{ji}|^p$ for all $i \in \mathcal{H}$. But this is exactly the condition from the previous lemma—namely, that the p-norms of incoming and outgoing weights are exactly balanced—and this exact balancing occurs only when $F(\mathbf{1})$ is a global minimum. □

Lemma 2.3 shows that each multiplicative update in Algorithm 1 serves to decrease the entry-wise p-norm $\|\mathbf{W}\|_{p,p}$; it also rules out oscillations between distinct global minima. But this by itself is not enough to prove that the updates converge to a finite (i.e., bounded) solution. For this we also need the following lemma.

Lemma 2.4 (Compactness of sublevel sets)**.** *Let $F : \mathcal{A} \to \Re$ be defined as in eq. (10). Then the sublevel set given by $\mathcal{F}_{\mathbf{1}} = \{\mathbf{a} \in \mathcal{A} \,|\, F(\mathbf{a}) \leq F(\mathbf{1})\}$ is compact.*

Proof. It follows from the continuity of F that its sublevel sets are closed; thus it remains only to show that $\mathcal{F}_{\mathbf{1}}$ is bounded. At a high level, this result will follow from the fact that the network has bounded depth. In particular, suppose $\mathbf{a} \in \mathcal{F}_{\mathbf{1}}$ with $F(\mathbf{a}) \leq F(\mathbf{1})$. Then if $W_{ij} \neq 0$, it must be the case that

$$\frac{a_i}{a_j} \leq \frac{\|\mathbf{W}\|_{p,p}^p}{|W_{ij}|^p}, \tag{17}$$

because otherwise the ij^{th} term of the sum in eq. (10) would by itself exceed $F(\mathbf{1})$. Let $j_0 \to j_1 \to \cdots \to j_m$ denote an m-step path through the network that starts at some input unit ($j_0 \in \mathcal{I}$), passes through the i^{th} hidden unit after k steps (so that $j_k = i$), ends at some output unit ($j_m \in \mathcal{O}$), and traverses only nonzero weights $W_{j_{\ell-1}j_\ell} \neq 0$ in the process. Note that there must exist at least one such path if the i^{th} hidden unit contributes in some way to the function computed by the network. Since $a_{j_0} = 1$ and $a_{j_k} = a_i$, it follows that

$$a_i \;=\; \frac{a_{j_k}}{a_{j_0}} \;=\; \prod_{\ell=1}^{k} \frac{a_{j_\ell}}{a_{j_{\ell-1}}} \;\leq\; \prod_{\ell=1}^{k} \frac{\|\mathbf{W}\|_{p,p}^p}{|W_{j_\ell j_{\ell-1}}|^p}, \tag{18}$$

where the inequality follows from eq. (17). Likewise, since $a_{j_m} = 1$ and $a_{j_k} = a_i$, it follows that

$$\frac{1}{a_i} \;=\; \frac{a_{j_m}}{a_{j_k}} \;=\; \prod_{\ell=k+1}^{m} \frac{a_{j_\ell}}{a_{j_{\ell-1}}} \;\leq\; \prod_{\ell=k+1}^{m} \frac{\|\mathbf{W}\|_{p,p}^p}{|W_{j_\ell j_{\ell-1}}|^p} \tag{19}$$

Eqs. (18–19) provide upper and lower bounds on a_i for all $\mathbf{a} \in \mathcal{F}_{\mathbf{1}}$. Thus $\mathcal{F}_1$ is closed and bounded, hence compact. □

With these results from the previous two lemmas, we can now prove Theorem 1.

Proof of Theorem 2.1. It follows from Lemma 2.4 that the set $\mathcal{C} = \{\mathbf{W} \,|\, \mathbf{W} \sim \mathbf{W}_0, \|\mathbf{W}\|_{p,p} \leq \|\mathbf{W}_0\|_{p,p}\}$ is compact. Likewise it follows from Lemma 2.3 that starting from $\mathbf{W}_0$, each multiplicative update yields a weight matrix $\mathbf{W} \in \mathcal{C}$ whose p-norm is less than or equal to the previous one (with equality occurring only when $\mathbf{W}$ is both a global minimizer and a fixed point of the updates). Finally we note that the multiplicative coefficients in eq. (12) are a continuous function of the weights from which they are derived. The procedure in Algorithm 1 therefore satisfies the preconditions of compactness, strict monotonicity, and continuity for Meyer's convergence theorem (Meyer, 1976) in the setting where fixed points occur at (and only at) global minima of $\|\mathbf{W}\|_{p,p}$ in $\mathcal{C}$. □

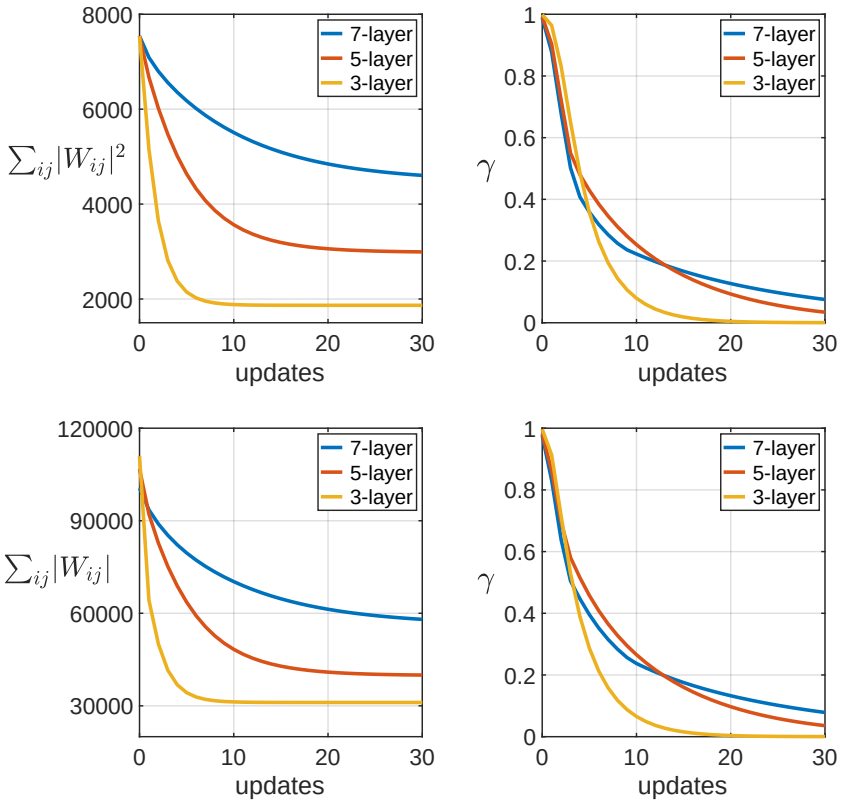

Figure 2: Convergence of gauge-fixing multiplicative updates in three randomly initialized networks with differing numbers of hidden layers but the same overall numbers of input, hidden, and output units. The panels show the results, respectively, for updates that minimize the ℓ_2-norm (*top*) and ℓ_1-norm (*bottom*) of the weights. Likewise they plot the value of the objective function (*left*) and the maximum degree of imbalance (*right*) in eq. (20) across all hidden units.

2.4 Demonstration of convergence

Fig. 2 plots the convergence of the multiplicative updates (with $p = 1, 2$) for three randomly initialized networks with differing numbers of hidden layers but the same overall numbers of input (200), hidden (3750), and output (10) units. From shallowest to deepest, the networks had 200-2500-1250-10 units, 200-2000-1000-500-250-10 units, and 200-1000-750-750-500-500-250-10 units. The networks were initialized with zero-valued biases and zero-mean Gaussian random weights whose variances were inversely proportional to the fan-in at each unit (He et al., 2015). The left panels of the figure plot the value of $\|\mathbf{W}\|_{p,p}^p$, and the right panels plot the maximal imbalance $\gamma \in [0, 1]$ across all hidden units, computed as

$$\gamma \;=\; \max_{i \in \mathcal{H}} \left[\frac{\sum_j (|W_{ij}|^p - |W_{ji}|^p)}{\sum_j (|W_{ij}|^p + |W_{ji}|^p)} \right] . \tag{20}$$

As expected, the updates take longer to converge in deeper networks (where imbalances must propagate through more layers), but in general a high degree of convergence is obtained for a modest number of updates. *The panels show that conventionally initialized networks are (i) far from minimal as measured by the ℓ_p-norm of their weights and (ii) easily rebalanced by a sequence of rescaling transformations.* Finally we note that the results in Fig. 2 did not depend sensitively on the value of the random seed used to generate them (though they might in networks with much smaller numbers of weights).

3 Minimum-norm learning with gradient flow

The rescaling symmetry in Fig. 1 also has important consequences for learning (Kunin et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021; Neyshabur et al., 2015a; Meng et al., 2019; Badrinarayanan et al., 2015; Armenta et al., 2021; Zhao et al., 2022). In this section we examine the conditions for learning under which the incoming and outgoing weights at each hidden unit remain balanced, as defined in Lemma 2.2; equivalently, these are the conditions for learning under which the entry-wise p-norm of the weight matrix remains minimal with respect to rescaling transformations. Our interest lies mainly in the following question: are there conditions such that the gauge-fixing procedure of the last section can be *one and done* at the outset of learning? To answer this question, we must understand whether learning and gauge-fixing are complementary procedures or whether they are operating in some way at cross-purposes (e.g., the former undoing the latter).

There are many forms of learning in deep networks. Perhaps the simplest to analyze is gradient flow (Elkabetz & Cohen, 2021), where a network's weights and biases are adapted in continuous time to decrease its loss function. Gradient flow provides a reasonable approximation to the behavior of deep networks with small learning rates. In this section we analyze how the entry-wise p-norm of the weight matrix evolves under gradient flow. Section 3.1 analyzes this evolution for an unregularized loss function (i.e., no weight decay), while section 3.2 does the same for a regularized one. Our main result (Theorem 3.4) is to derive a regularized gradient flow under which the entry-wise p-norm of the weight matrix remains minimal with respect to rescaling transformations; this is done for any $p>0$ and for any amount of regularization.

3.1 Unregularized flows

As in the previous section, we focus on feedforward networks with homogeneous activation functions. Let $C(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ denote the cost when a network's actual output $f(\mathbf{x})$ is evaluated against some reference output $\mathbf{y}$. Suppose further that the network is trained to minimize the average empirical loss

$$L = \frac{1}{T}\sum_{t=1}^{T} C(\mathbf{y}_t, \mathbf{f}(\mathbf{x}_t)), \tag{21}$$

on some training set $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$ of labeled examples. Note that while the loss in eq. (21) may depend in a complicated way on the network's weights and biases, it is necessarily invariant to the rescaling transformations of these parameters in eqs. (5–6).

Many authors have observed that the rescaling symmetry in Fig. 1 gives rise to a conservation law when these networks are trained by gradient flow (Du et al., 2018; Kunin et al., 2021; Bronstein et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). The connection between symmetries and conservation laws is well known from Noether's theorem (Noether, 1918), but it is worth noting that the dynamics of gradient flow were not historically derived from a Lagrangian. Some recent works, however, have explored the connection to the dynamics of damped Lagrangian systems (Wibisono et al., 2016; Tanaka & Kunin, 2021).

In this work we will consider a slightly generalized family of gradient flows in which the learning rate of each parameter is modulated by its magnitude. In particular, we suppose that

$$\dot{W}_{ij} = -|W_{ij}|^{2-p} \cdot \frac{\partial L}{\partial W_{ij}}, \tag{22}$$

$$\dot{b}_i = -|b_i|^{2-p} \cdot \frac{\partial L}{\partial b_i}, \tag{23}$$

where we have highlighted the additional modulating terms in blue. It is easy to see that the standard form of gradient descent is recovered for $p=2$. This small bit of extra generality will be all that is needed for us to obtain results for the evolution of the entry-wise matrix p-norm $\|\mathbf{W}\|_{p,p}$, where $p>0$. It also leads to our first result of this section.

Lemma 3.1 (Conservation Law for Gradient Flow)**.** *Consider a feedforward network with homogeneous activation functions. At each hidden unit $i \in \mathcal{H}$, let*

$$\Delta_i \;=\; |b_i|^p + \sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p \tag{24}$$

measure the difference in norms of incoming and outgoing weights (including the bias) for some $p>0$. Then this difference is a constant of the motion under the gradient flow in eqs. (21–23).

Proof. The proof is an extension of previous results for the special case $p=2$ (Du et al., 2018; Kunin et al., 2021; Bronstein et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). Differentiating Δ_i in eq. (24) with respect to time, we find:

$$\dot{\Delta}_i = \dot{\mathbf{b}} \cdot \frac{\partial \Delta_i}{\partial \mathbf{b}} + \dot{\mathbf{W}} \cdot \frac{\partial \Delta_i}{\partial \mathbf{W}}, \tag{25}$$

$$= \dot{b}_i \frac{\partial \Delta_i}{\partial b_i} + \sum_j \left[\dot{W}_{ij} \frac{\partial \Delta_i}{\partial W_{ij}} + \dot{W}_{ji} \frac{\partial \Delta_i}{\partial W_{ji}} \right], \tag{26}$$

$$= \dot{b}_i \frac{p|b_i|^p}{b_i} + \sum_j \left[\dot{W}_{ij} \frac{p|W_{ij}|^p}{W_{ij}} - \dot{W}_{ji} \frac{p|W_{ji}|^p}{W_{ij}} \right]. \tag{27}$$

Next we use the gradient flow in eqs. (22–23) to replace the time-derivatives in eq. (27) of the network's weights and biases:

$$\dot{\Delta}_i = -p \frac{\partial L}{\partial b_i} b_i - p \sum\nolimits_j \left[\frac{\partial L}{\partial W_{ij}} W_{ij} - \frac{\partial L}{\partial W_{ji}} W_{ji} \right], \tag{28}$$

$$= -p \left[\frac{\partial L}{\partial \mathbf{b}} \cdot \frac{d\mathbf{b}}{da_i} + \frac{\partial L}{\partial \mathbf{W}} \cdot \frac{d\mathbf{W}}{da_i} \right] \Bigg|_{\mathbf{a=1}}, \tag{29}$$

$$= -p \left[\frac{dL}{da_i} \right] \Bigg|_{\mathbf{a=1}}, \tag{30}$$

$$= \; 0. \tag{31}$$

Here, the final steps follow from the form of the rescaling transformations in eqs. (5–6) and the invariance of the loss in eq. (21) to these transformations. □

The conservation law in Lemma 3.1 was derived from the gradient flows in eq. (22–23). To proceed, we focus on networks in which the biases at all hidden units are **frozen at zero**. By this we simply mean that for all $i \in \mathcal{H}$, we set $\dot{b}_i = b_i = 0$ in place of the flow of eq. (23). Doing so, we obtain the following result as an immediate but important corollary.

Corollary 3.2. *Consider a feedforward network with homogeneous activation functions whose biases at all hidden units are frozen at zero. Then in place of eq. (24), we obtain the conserved quantities*

$$\Delta_i \;=\; \sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p. \tag{32}$$

This corollary has important implications for minimum-norm gauge-fixing. The mathematical connection is the following: the difference that appears in eq. (32) as a conserved quantity is also the partial derivative that appears in eq. (11) for minimizing the p-norm of the network's weight matrix. Put another way, the network's symmetry group of rescaling transformations gives rise to multiple constants of the motion—one per hidden unit—and it is precisely when all these constants of the motion *vanish* that the network's weight

matrix has a minimal entry-wise p-norm with respect to these transformations. From this observation we obtain the following theorem.

Theorem 3.3 (Minimality-Preserving Flows)**.** *Suppose that the weight matrix* $\mathbf{W}$ *has been initialized and/or rescaled by the gauge-fixing procedure of Algorithm 1 such that* $\|\mathbf{W}\|_{p,p}$ *cannot be further minimized by any rescaling transformation. Then this property of minimality is preserved by the gradient flow in eqs. (21–23) if in addition the biases at all hidden units are frozen at zero.*

Proof. When a hidden unit has zero bias (i.e., $b_i = 0$), the conserved quantity in eq. (24) reduces to the simple difference $\Delta_i = \sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p$ in eq. (32). Since $\|\mathbf{W}\|_{p,p}$ cannot be further minimized, the partial derivative in eq. (11) must vanish at the identity gauge transformation $\mathbf{a} = \mathbf{1}$. It follows that $\sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p = 0$ at each hidden unit $i \in \mathcal{H}$. But this quantity is conserved by the gradient flow in eqs. (21–23) subject to the additional constraint $b_i = \dot{b}_i = 0$. It follows that the weights at future times must also satisfy $\sum_j |W_{ij}|^p = \sum_j |W_{ji}|^p$, and thus by Lemma 2.2, the minimality property is preserved. □

The significance of Theorem 3.3 is that it establishes relatively mild conditions under which the network's weight matrix $\mathbf{W}$ never leaves the submanifold of weight space in which its entry-wise p-norm $\|\mathbf{W}\|_{p,p}$ is minimized with respect to rescaling transformations. The first of these conditions is that the hidden (but not the output) units have zero biases, and the second is that the learning rates of weights and biases are modulated by a power of their magnitudes. It is interesting that the minimality-preserving flows in Theorem 3.3 provide independent motivation for these practices, both of which have been previously studied for different reasons. It has been noted, for example, that zero-valued biases are required for rectified-linear units to learn *intensity-equivariant* representations of sensory inputs (Hinton et al., 2011; Mohan et al., 2020); these are representations in which the hidden-layer activations scale in proportion to the intensity of visual or auditory signals. Such networks also have certain margin-maximizing properties when they are trained by gradient descent (Lyu & Li, 2020). Additionally, we note that for $p = 1$, the modulated gradient flow in eqs. (22–23) approximates a discrete update that is additive in the *log* domain; in such an update, parameters of fixed sign are multiplied by the elements of an exponentiated gradient. Similar updates have been studied in many different contexts (Kivinen & Warmuth, 1997; Arora et al., 2012; Bernstein et al., 2020).

3.2 Regularized flows

Our goal in this section is to generalize Theorem 3.3 to learning in the presence of a regularizer. Regularization is a common practice to avoid overfitting of the training data by large networks (Goodfellow et al., 2016). In this paper we consider regularized losses of the form

$$L_\lambda \;=\; \frac{1}{T}\sum_t C(\mathbf{y}_t, \mathbf{f}(\mathbf{x}_t)) + \frac{\lambda}{p}\|\mathbf{W}\|_{p,p}^p \tag{33}$$

with $\lambda > 0$. Regularizers are designed to prevent overfitting by penalizing large weights. But if regularizers were originally introduced for this reason, it is now widely appreciated that they also serve other purposes. It has been observed, for example, that regularizers help to learn better models on the *training* data (Krizhevsky et al., 2012), suggesting that smaller weights lead to better behaved gradients. Likewise, it has been observed that highly unbalanced weights lead to much slower training in homogeneous networks; the reason is that partial derivatives such as $\partial L/\partial W_{ij}$ scale *inversely* as the weights under rescaling transformations (Neyshabur et al., 2015a; Dinh et al., 2017). More generally, it has been argued (van Laarhoven, 2017) that "by decreasing the scale of the weights, weight decay increases the effective learning rate" and that "if no regularization is used the weights can grow unbounded, and the effective learning rate goes to 0."

These observations suggest a natural pairing of regularization and minimum-norm gauge-fixing. In particular, if regularization has purely dynamical benefits—if (say) smaller or more balanced weights lead to faster learning—then one might expect similar benefits by minimizing the norm of the weights with respect to

rescaling transformations. Our final theorem examines the interplay between regularization and gauge-fixing during learning. Intuitively, it states that if the regularizer and gauge-fixing condition are appropriately *paired*, then the minimality of the weight matrix is also preserved under gradient flow in the *regularized* loss function of eq. (33). Specifically, we consider the dynamics

$$\dot{W}_{ij} = -|W_{ij}|^{2-p} \cdot \frac{\partial L_\lambda}{\partial W_{ij}}, \tag{34}$$

$$\dot{b}_i = -|b_i|^{2-p} \quad \cdot \quad \frac{\partial L_\lambda}{\partial b_i} \quad \text{for} \quad i \in \mathcal{O}, \tag{35}$$

where the biases at all hidden units are frozen at zero (i.e., $b_i = \dot{b}_i = 0$ for all $i \in \mathcal{H}$). With this flow we obtain the following result.

Theorem 3.4 (Pairing Theorem)**.** *Suppose that the weight matrix* $\mathbf{W}$ *has been initialized and/or rescaled by the gauge-fixing procedure of Algorithm 1 such that* $\|\mathbf{W}\|_{p,p}$ *cannot be further minimized. Then this property of minimality is preserved by the regularized gradient flow in eqs. (34–35); specifically this is the flow (for any* $\lambda > 0$*) that pairs the gauge-fixing condition with the corresponding entry-wise p-norm regularizer.*

Proof. Again we proceed by differentiating the difference $\Delta_i = \sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p$ with respect to time. Since the hidden-unit biases are frozen at zero, we have

$$\dot{\Delta}_i = \dot{\mathbf{W}} \cdot \frac{\partial \Delta_i}{\partial \mathbf{W}} = \sum_j \left[\dot{W}_{ij} \frac{\partial \Delta_i}{\partial W_{ij}} + \dot{W}_{ji} \frac{\partial \Delta_i}{\partial W_{ji}} \right]. \tag{36}$$

Next write $L_\lambda = L_0 + \lambda R$ where L_0 is the unregularized loss in eq. (21) and $R = (1/p)\|\mathbf{W}\|_{p,p}^p$. Note that the gradient flow for $\dot{\mathbf{W}}$ has two components, one generated by the gradient of L_0, the other by the gradient of the regularizer, R. We have already shown, via Theorem 3.3, that Δ_i is unchanged by the component from L_0. It therefore suffices to consider only the component from R. From this component, we see that

$$\begin{aligned}
\dot{\Delta}_i &= -\lambda \sum_j \left[|W_{ij}|^{2-p} \frac{\partial R}{\partial W_{ij}} \frac{\partial \Delta_i}{\partial W_{ij}} + |W_{ij}|^{2-p} \frac{\partial R}{\partial W_{ji}} \frac{\partial \Delta_i}{\partial W_{ji}} \right], \\
&= -\lambda \sum_j \left[\frac{W_{ij}^2}{|W_{ij}|^p} \frac{|W_{ij}|^p}{W_{ij}} \frac{p|W_{ij}|^p}{W_{ij}} - \frac{W_{ji}^2}{|W_{ji}|^p} \frac{|W_{ji}|^p}{W_{ji}} \frac{p|W_{ji}|^p}{W_{ji}} \right], \\
&= -\lambda p \sum_j \Big[|W_{ij}|^p - |W_{ji}|^p \Big], \\
&= -\lambda \, p \, \Delta_i.
\end{aligned} \tag{37}$$

By assumption $\mathbf{W}$ is initialized such that $\|\mathbf{W}\|_{p,p}$ is minimal with respect to rescaling transformations. Thus as before, the partial derivative in eq. (11) must vanish at the identity gauge transformation $\mathbf{a} = \mathbf{1}$, implying that $\Delta_i = \sum_j |W_{ij}|^p - \sum_j |W_{ji}|^p = 0$ at each hidden unit $i \in \mathcal{H}$. But this also implies, via eq. (37), that $\dot{\Delta}_i = 0$, and hence that Δ_i is a *vanishing* constant of the motion for all $i \in \mathcal{H}$. The theorem then follows from Lemma 2.2. □

The calculation in this proof yields another insight. From eq. (37), we see that Δ_i decays exponentially to zero in the presence of regularization, so that *even in the absence of a gauge-fixing procedure*, the weights at hidden units are driven to satisfy the balancedness condition of Lemma 2.2. Note, however, that the time constant of this decay varies *inversely* with the magnitude of the regularization hyperparameter $\lambda > 0$. In practice, typical values of λ are quite small. Thus gauge-fixing can be viewed as a way of balancing the weights *at the outset and throughout the entire course of learning*, rather than relying on the asymptotic effects of regularization to do so in the limit.

4 Minimum-norm learning with stochastic gradient descent

In previous sections, we have analyzed the interplay between gauge-fixing and learning via gradient flow. In this section, we investigate empirically how gauge-fixing affects the performance of stochastic gradient descent. Our goal is to observe these effects in a controlled setting where experiments can be meaningfully compared along a few essential axes of variation. There are, of course, many aspects of deep learning for which the theorems of the previous section do not have any obvious implications, and here we attempt to limit the potentially combinatorial number of experimental choices to a few essential ones.

4.1 Experimental results

We focus on the cases of ℓ_p-norm regularization with $p=1,2$. For ℓ_2-norm regularization, the gradient flow in eqs. (34–35) can be approximated by the additive updates

$$W_{ij} \leftarrow W_{ij} - \eta \big\langle \tfrac{\partial C}{\partial W_{ij}} \big\rangle - \eta \lambda W_{ij}, \tag{38}$$

$$b_i \leftarrow b_i - \eta \big\langle \tfrac{\partial C}{\partial b_i} \big\rangle \quad \text{for} \quad i \in \mathcal{O}, \tag{39}$$

where $\eta > 0$ is a learning rate and $\langle \cdot \rangle$ indicates a mini-batch estimate of the gradient of the cost in eq. (21). Likewise for ℓ_1-norm regularization, the gradient flow in eqs. (34–35) can be approximated by the multiplicative updates

$$W_{ij} \leftarrow W_{ij} \exp\Big(-\eta \operatorname{sign}(W_{ij}) \big\langle \tfrac{\partial C}{\partial W_{ij}} \big\rangle - \eta \lambda\Big), \tag{40}$$

$$b_i \leftarrow b_i \exp\Big(-\eta \operatorname{sign}(b_i) \big\langle \tfrac{\partial C}{\partial b_i} \big\rangle\Big) \quad \text{for} \quad i \in \mathcal{O}. \tag{41}$$

The updates in eqs. (40–41) can be viewed as exponentiated gradient updates for parameters whose signs are fixed at initialization but whose magnitudes are adapted over time (Kivinen & Warmuth, 1997; Arora et al., 2012; Bernstein et al., 2020). As shorthand, we refer to eqs. (38–39) as updates for stochastic gradient descent (SGD) and eqs. (40–41) as updates for exponentiated (stochastic) gradient descent (EGD).

We experimented with the gradient-based updates in eqs. (38–41) both with and without an additional procedure for gauge-fixing. For the latter, we simply called Algorithm 1 before each step of SGD or EGD to minimize $\|\mathbf{W}\|_{p,p}$ up to some tolerance δ. In general, this additional procedure did not incur much overhead relative to the mini-batch updates for learning. Note that the amount of overhead is determined essentially by the number of rescaling transformations computed by Algorithm 1. This number may be moderate at initialization (as shown by Figure 2), but once the gauge has been fixed, the property of norm-minimality is preserved by gradient flow (as shown by Theorem 3.4) and only violated to $O(\eta^2)$ by each discrete-time update in eqs. (38–41). Therefore many fewer rescaling transformations are required to minimize $\|\mathbf{W}\|_{p,p}$ throughout the course of learning. The relative overhead of the gauge-fixing procedure also decreases in proportion to the size of the mini-batch.

Our experiments examined the effects of gauge-fixing on the optimization of the regularized loss in eq. (33). We studied problems in multiway classification where each input $\mathbf{x}_t$ in the training set was labeled by a unary vector $\mathbf{y}_t$ that provided a one-hot encoding of the correct class. In these experiments we optimized the regularized *log-loss* where the cost per example was computed via the softmax operation:

$$C(\mathbf{y}, \mathbf{f}(\mathbf{x})) = -\textstyle\sum_\alpha y_\alpha \log \rho_\alpha \quad \text{where} \quad \rho_\alpha = \dfrac{e^{f_\alpha(\mathbf{x})}}{\sum_\beta e^{f_\beta(\mathbf{x})}}. \tag{42}$$

We experimented on the four publicly available datasets shown in Table 1. Also, for each data set, we performed a singular value decomposition (Eckart & Young, 1936) to project the inputs into a smaller space of 200 dimensions; this was done to reduce training times.

We trained three-layer, five-layer, and seven-layer networks on these datasets, where each network had a total of 3750 hidden units. From shallowest to deepest, the networks had hidden layers with 2500-1250 hidden

Dataset	Examples	Dimension	Classes	Abbrev.
MNIST-digits (LeCun et al., 1998)	60000	784	10	DIGITS
Fashion-MNIST (Xiao et al., 2017)	60000	784	10	FASHION
EMNIST-balanced (Cohen et al., 2017)	112800	784	47	CHARS
CIFAR10 (Krizhevsky, 2009)	50000	3072	10	CIFAR10

Table 1: Data sets of grayscale (LeCun et al., 1998; Xiao et al., 2017; Cohen et al., 2017) and color (Krizhevsky, 2009) images that were used to train the networks in this paper.

units, 2000-1000-500-250 hidden units, and 1000-750-750-500-500-250 hidden units; the number of output units was equal to the number of classes (either 10 or 47). We used an initial learning rate of $\eta = 0.001$ and a weight decay of $\lambda = 10^{-4}$ in the networks with ℓ_2-norm regularization and an initial learning rate of $\eta = 0.1$ and a weight decay of $\lambda = 5 \times 10^{-6}$ in those with ℓ_1-norm regularization. All networks were trained for 30 epochs with a mini-batch size of 32, and we reduced the learning rate by a factor of 0.95 after each epoch. As suggested by Theorem 3.4, we only adapted the biases for output units, while those for hidden units were frozen at zero. We allowed the multiplicative updates in Algorithm 1 to converge to a precision of $\delta = 10^{-3}$. In all, we present a summary of results from 240 experiments—one for each type of network (with three, five, or seven layers) on each data set in Table 1, with either ℓ_1-norm or ℓ_2-norm regularization, starting from five different random seeds, and evaluating the performance with and without an additional procedure for gauge-fixing.

Fig. 3 plots the regularized log-loss and training error rates in the experiments with ℓ_2-norm regularization as a function of the number of epochs of training. The left panels in this figure show that the minimum-norm gauge leads to consistently faster descent in the regularized log-loss. The right panels show that gauge-fixing also leads to lower error rates on the training set; this is an indication that gauge-fixing affects the course of learning in ways beyond the mere rescaling of weights. (Note that two networks cannot be equivalent up to rescaling transformations if they have different error rates.)

Fig. 4 shows the same plots for the experiments with ℓ_1-norm regularization. Here again we observe that the minimum-norm gauge leads to faster descent in the regularized log-loss, but in this case we do not see a similar effect on the training error rates. It should be noted, however, that the magnitude of these effects is largely controlled by the amount of regularization. By increasing λ, one can (arbitrarily) increase the gap between the curves in the left panels, while by decreasing λ, one can weaken the regularizer in an attempt to obtain smaller error rates. We have fixed the value of λ in these experiments to understand, in a controlled setting, how gauge-fixing affects the optimization of the regularized log-loss (which is the actual quantity being optimized). However, the right panels of Fig. 4 suggest that weaker regularizers (i.e., smaller λ) may be more appropriate when the weights are being continually rebalanced by rescaling transformations.

4.2 Related Work

Many previous studies have investigated how to reformulate gradient-based learning in networks with rescaling symmetries. In a seminal paper, Neyshabur et al. (2015a) showed that SGD performs poorly in highly unbalanced networks, and in its place, they proposed PathSGD, a rescaling-invariant procedure that approximates steepest descent with respect to a special path-based regularizer. Notably, this regularizer has the distinguishing property that it computes the minimum value of a max-norm regularizer, where the minimum is performed over all networks equivalent up to rescaling (Neyshabur et al., 2015b). PathSGD was followed by other formulations of rescaling-invariant learning. For example, Badrinarayanan et al. (2015) fixed the rescaling degrees of freedom in multilayer networks by constraining certain weight vectors to have unit norm, while Meng et al. (2019) showed how to perform SGD in the vector space of paths (as opposed to weights), where the rescaling-invariant value of a path is given by the product of its weights.

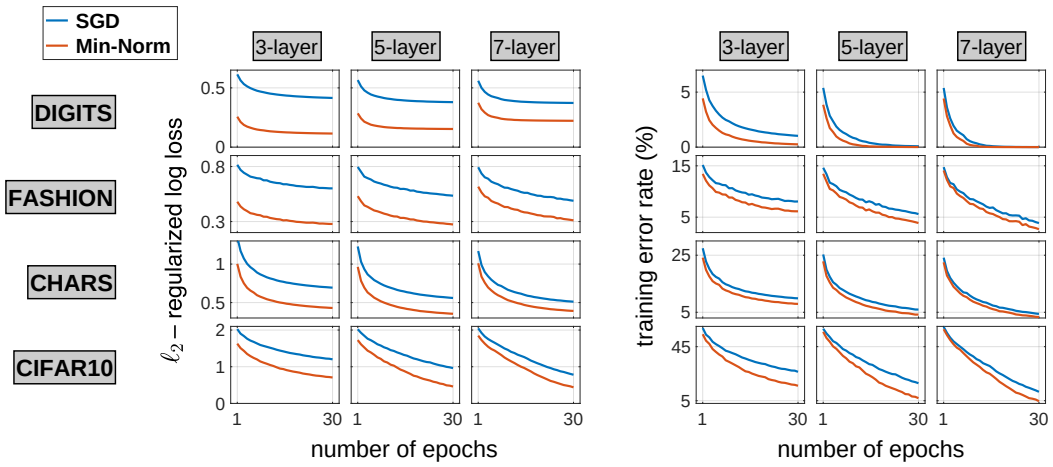

Figure 3: Comparison of SGD with and without rescaling transformations to minimize the sum of the squares of the weights. The left and right panels plot the ℓ_2-regularized log loss and the training error rate, respectively, as a function of the number of epochs. Results are shown for three different architectures and four different data sets. Each curve is the result from averaging over five random seeds.

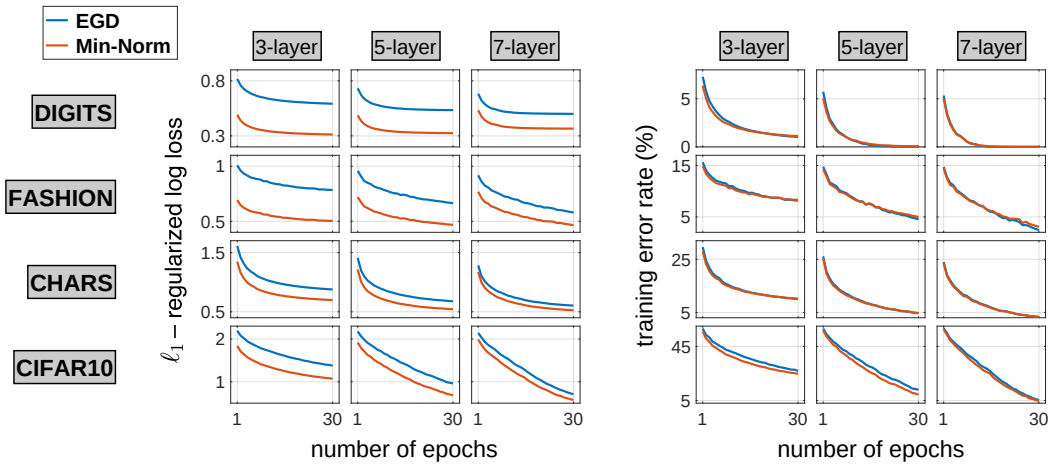

Figure 4: Comparison of EGD with and without rescaling transformations to minimize the sum of the magnitudes of the weights. The left and right panels plot the ℓ_1-regularized log loss and the training error rate, respectively, as a function of the number of epochs. Results are shown for three different architectures and four different data sets. Each curve is the result from averaging over five random seeds.

More recent work has considered how to accelerate learning with particular rescaling transformations. For instance, Armenta et al. (2021) showed that the magnitudes of backpropagated gradients are boosted on average by randomly rescaling the weights before or in the middle of the learning—a process they call *neural teleportation*. Likewise, Zhao et al. (2022) explored how to choose symmetry group transformations that purposefully increase or maximize the norms of gradients for learning. Because these gradients are computed with respect to particular training examples, this approach can be viewed as a *data-driven* procedure for manipulating the optimization landscape via symmetry group transformations.

Within this body of work, our main contribution is to derive the gauge-fixing multiplicative updates in Algorithm 1 that minimize the ℓ_p-norms of weights under rescaling transformations. This gauge-fixing

criterion yields another rescaling-invariant procedure for learning, one motivated by the same underlying appeals to symmetry as in earlier studies (Neyshabur et al., 2015a; Badrinarayanan et al., 2015; Meng et al., 2019), but more closely aligned (via Theorem 3.4) with the widespread use of ℓ_p-norm regularization. Our approach differs from work on neural teleportation (Armenta et al., 2021; Zhao et al., 2022) by employing rescaling transformations to minimize the norms of weights rather than to increase the norms of gradients. These approaches may have similar effects in practice; we note, however, that the norms of gradients are unbounded above with respect to the (non-compact) group of rescaling transformations, and therefore one must be careful to identify the regime in which they serve as a reliable proxy for rates of convergence.

5 Discussion

Deep learning is a revolutionary technology whose workings are not fully understood. In this paper, we have shown that further understanding may be gained from the symmetries of multilayer networks and the analogies they suggest to physical systems (Bronstein et al., 2021; Kunin et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). As in Lagrangian mechanics, these symmetries lead to conserved quantities when networks are trained by dynamical flows; as in gauge theories, they reveal redundant degrees of freedom that can be fixed in opportune ways.

In this paper we have focused specifically on the rescaling symmetries of homogeneous networks. Inspired by these symmetries, we have shown how to rebalance the weights $\mathbf{W}$ of a feedforward network without changing the function that it computes. Specifically, we derived closed-form multiplicative updates that minimize the entry-wise p-norm $\|\mathbf{W}\|_{p,p}$ over the equivalence class of networks that are related by rescaling transformations. We also showed that this property of minimality was preserved by gradient flow in a correspondingly regularized loss function. Finally we considered how these rescaling symmetries might be exploited in conjunction with SGD. Our experimental results provide further evidence that learning can be accelerated by fixing (or otherwise controlling for) the degrees of freedom associated with these symmetries (Neyshabur et al., 2015a; Badrinarayanan et al., 2015; Meng et al., 2019; Armenta et al., 2021; Zhao et al., 2022).

There are many questions deserving of further investigation. One important question is how to combine gauge-fixing with accelerated gradient-based methods, such as those involving momentum (Polyak, 1964) or adaptive learning schedules (Kingma & Ba, 2015; Duchi et al., 2010; Tieleman & Hinton, 2018). These methods may already be compensating, to some extent, for the rescaling symmetries in the optimization landscape, but if so, these effects need to be further clarified.

Other potential benefits of gauge-fixing are suggested in the more familiar setting of matrix factorization (Horn & Johnson, 2012). The basic problem of factorization is underdetermined: any matrix can be written in an infinite number of ways as the product of two or more other matrices. But consider the wealth of information that is revealed by certain canonical factorizations of large matrices: for example, from the singular value decomposition, it is straightforward to compute the low-rank approximation that is optimal in a least-squares sense (Eckart & Young, 1936). It is natural to ask whether the functions computed by multilayer networks can be represented in a similarly canonical way, and if so, whether such representations might suggest more effective strategies for pruning, compressing, or otherwise approximating their weight matrices. The search for such representations provides yet another motivation for gauge-fixing.

Finally we note that there are many possible criteria for gauge-fixing in multilayer networks with rescaling symmetries. In this paper we studied how to minimize the entry-wise p-norm of the weight matrix, a problem we found to be especially tractable. It would be interesting to study other criteria for gauge-fixing and to derive the conditions (analogous to those in Theorems 3.3–3.4) under which these criteria are preserved by gradient-based learning. We believe that the present work can provide a template for these further investigations—and also that such investigations will reveal a similarly rich mathematical structure.

References

P. W. Anderson. More is different. *Science*, 177(4047):393–396, 1972.

M. Armenta and P.-M. Jodoin. The representation theory of neural networks. *Mathematics*, 9(24), 2021.

M. A. Armenta, T. Judge, N. Painchaud, Y. Skandarani, C. Lemaire, G. G. Sanchez, P. Spino, and P. M. Jodoin. Neural teleportation, 2021. arXiv:2012.01118.

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

S. Arora, N. Cohen, and E. Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In J. Dy and A. Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, pp. 244–253, 2018.

S. Arora, N. Cohen, N. Golowich, and W. Hu. A convergence analysis of gradient descent for deep linear neural networks. In *Proceedings of the 8th International Conference on Learning Representations*, 2019.

V. Badrinarayanan, B. Mishra, and R. Cipolla. Understanding symmetries in deep networks. In *Proceedings of the 8th NeurIPS Workshop on Optimization for Machine Learning*, 2015.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling the modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences USA*, 116(32):15849–15854, 2019.

J. Bernstein, J. Zhao, M. Meister, M.-Y. Liu, A. Anandkumar, and Y. Yue. Learning compositional functions via multiplicative weight updates. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems 33*, pp. 13319–13330, 2020.

M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges, 2021. arxiv:2104.13478.

G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. EMNIST: an extension of MNIST to handwritten letters, 2017. arXiv:1702.05373.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In D. Precup and Y. W. Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, pp. 1019–1028, 2017.

S. S. Du, W. Hu, and J. D. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In S. Bengio, H. M Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 382–393, 2018.

J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Proceedings of the 23rd Conference on Learning Theory*, pp. 257–269, 2010.

C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218, 1936.

O. Elkabetz and N. Cohen. Continuous vs. discrete optimization of deep neural networks. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan (eds.), *Advances in Neural Information Processing Systems 34*, pp. 4947–4960, 2021.

N. Gillis. *Nonnegative Matrix Factorization*. SIAM, 2021.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and D. M. Titterington (eds.), *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudík (eds.), *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

G. Gluch and R. Urbanke. Noether: the more things change, the more they stay the same, 2021. arXiv:2104.05508.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1319–1327, 2013.

D. J. Gross. Gauge theory—past, present, and future? *Chinese Journal of Physics*, 30:955–972, 1992.

F. V. Gubarev, L. Stodolsky, and V. I. Zakharov. On the significance of the vector potential squared. *Physical Review Letters*, 86(11):2220–2222, 2001.

A. Gunawardana and W. Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6:2049–2073, 2005.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-11)*, pp. 44–51, 2011.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

J. D. Jackson. From Lorenz to Coulomb and other explicit gauge transformations. *American Journal of Physics*, 70:917–928, 2002.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun (eds.), *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.

D. Kunin, J. Sagastuy-Breña, S. Ganguli, D. L. K. Yamins, and H. Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401: 788–791, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp (eds.), *Advances in Neural Information Processing Systems 13*, pp. 556–562. MIT Press, 2000.

K. Lyu and J. Li. Gradient descent maximizes the margin of homogeneous neural networks. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Q. Meng, S. Zheng, H. Zhang, W. Chen, Q. Ye, Z.-M. Ma, N. Y, and T.-Y. Liu. G-SGD: Optimizing ReLU neural networks in its positively scale-invariant space. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

R. R. Meyer. Sufficient conditions for the convergence of monotonic mathematical programming algorithms. *Journal of Computer and System Sciences*, 12(1):108–121, 1976.

S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR-20)*, 2020. URL `https://openreview.net/forum?id=HJlSmC4FPS`.

B. Neyshabur, R. Salakhutdinov, and N. Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2422–2430, 2015a.

B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Proceedings of the 28th Conference on Learning Theory*, pp. 1376–1401, 2015b.

E. Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pp. 235–257, 1918.

V. Papyan, X. Y. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences USA*, 117(40):24652–24663, 2020.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Lawrence K. Saul, Fei Sha, and Daniel D. Lee. Statistical signal processing with nonnegativity constraints. In *Proceedings of the 8th European Conference on Speech Communication and Technology*, pp. 1001–1004, 2003.

A. M. Saxe, J. L. McClelland, and S. Ganguil. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Y. Bengio and Y. LeCun (eds.), *Proceedings of the 2nd International Conference on Learning Representations*, 2013.

F. Sha, Y. Lin, L. K. Saul, and D. D. Lee. Multiplicative updates for nonnegative quadratic programming. *Neural Computation*, 19:2004–2031, 2007.

B. K. Sriperumbudur and G. R. G. Lanckriet. A proof of convergence of the concave-convex procedure using zangwill's theory. *Neural Computation*, 24:1391–1407, 2012.

I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1139–1147, 2013.

H. Tanaka and D. Kunin. Noether's learning dynamics: Role of symmetry breaking in neural networks. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan (eds.), *Advances in Neural Information Processing Systems 34*, pp. 25646–25660, 2021.

H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems 33*, pp. 6377–6389, 2020.

T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2018.

T. van Laarhoven. L2 regularization versus batch and weight normalization, 2017. arxiv:1706.04340.

A. Wibisono, A. C. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences USA*, 113(47):E7351–E7358, 2016.

C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, 2017. arXiv:1708.07747.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

W. J. Zangwill. *Nonlinear programming: A unified approach*. Prentice Hall, 1969.

A. Zee. *Fearful Symmetry: The Search for Beauty in Modern Physics*. Princeton University Press, 2016.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021a.

Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021b.

B. Zhao, N. Dehmamy, R. Walters, and R. Yu. Symmetry teleportation for accelerated optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35*, 2022.