Weight-Balancing Fixes and Flows 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. This paper makes two contributions to our understanding of these networks. The first is to describe a simple procedure for *gauge-fixing:* 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 weight matrix is of minimal $\ell_{p,q}$-norm; the weights at each hidden unit are said to be *balanced* when this norm is stationary with respect to rescaling transformations. The optimal rescaling factors are computed in an iterative fashion via simple multiplicative updates, and the updates are notable in that (a) they do not require the tuning of learning rates, (b) they operate in parallel on the rescaling factors at all hidden units, and (c) they converge monotonically to a global minimizer of the $\ell_{p,q}$-norm. The paper's second contribution is to analyze the optimization landscape for learning in these networks. We suppose that the network's loss function consists of two terms—one that is invariant to rescaling transformations, measuring predictive accuracy, and another (a regularizer) that breaks this invariance, penalizing large weights. We show how to derive a *weight-balancing flow* such that the regularizer remains minimal with respect to rescaling transformations as the weights descend in the loss function. This weight-balancing flow reduces to an ordinary gradient flow for ℓ_2-norm regularization, but not otherwise. Though gradient flow serves as a conceptual foundation for deep learning, our analysis suggests a canonical pairing of alternative flows and regularizers.

1 Introduction

Many recent studies of deep learning have focused on the important role of symmetries (Bronstein et al., 2021; Kunin et al., 2021; Tanaka & Kunin, 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

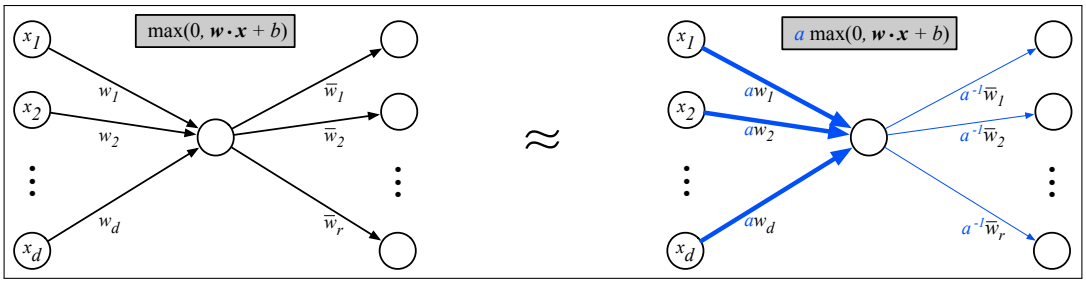

Figure 1: A rectified linear unit (ReLU) has the same effect if its incoming weights $\mathbf{w}$ and bias b are rescaled by some factor $a>0$ while its outgoing weights $\overline{\mathbf{w}}$ are rescaled by the inverse factor a^{-1}. When $a>1$, the rescaling increases the magnitudes of incoming weights and decreases the magnitudes of outgoing weights; when $a<1$, the effects are reversed.

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 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 forms of learning 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; Stock et al., 2019; 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. Given this distinction, it remains a matter of great significance to understand the different criteria that can be used to break this invariance.

1.1 Inspiration from physics

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). The gauge degrees of freedom in this setting are typically *fixed* to simplify or highlight certain physics of interest. For interest, one choice is the Coulomb gauge, which has the property that it minimizes the volume integral of the squared vector potential (Gubarev et al., 2001). Another choice is the Lorenz gauge, which simplifies certain wave equations. It is natural to ask if there are analogous gauge-fixing conditions for deep neural networks—conditions, for instance, that minimize the norms of weights and/or simplify the dynamics of learning.

With these goals in mind, this paper investigates a family of norm-minimizing gauge transformations for feedforward networks with rescaling symmetries. These transformations are designed to minimize the entriwise $\ell_{p,q}$-norm (with $p,q\geq 1$) of the network's weight matrix $\mathbf{W}$, defined as

$$\|\mathbf{W}\|_{p,q} = \left(\sum_i \left(\sum_j |W_{ij}|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}, \tag{1}$$

without changing the end-to-end function that the network computes. Here W_{ij} is the incoming weight at unit i from unit j. A particular transformation is specified by a set of multiplicative rescaling factors, one at each hidden unit of the network.

The first main contribution of this paper is to show how to obtain rescaling factors that minimize the entriwise matrix norm in eq. (1). More concretely, we derive simple multiplicative updates to compute these

rescaling factors (Algorithm 1), 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 that they do not require the setting or tuning of learning rates.

The results in this paper extend earlier important work on this problem. Most relevant is the work on ENorm (Stock et al., 2019)—an elegant procedure, also based on multiplicative updates, to minimize the ℓ_p-norm of the weights in multilayer feedforward networks via rescaling transformations. The updates for ENorm are derived from block coordinate descent of the norm in eq. (1) for the special case $p = q$; the blocks are formed by grouping hidden units in the same layer. In this paper, we use a different approach; in particular, we derive our multiplicative updates from a so-called auxiliary function—an approach familiar from popular algorithms such as nonnegative matrix factorization (Lee & Seung, 2000) and Expectation-Maximization (EM) (Dempster et al., 1977). This approach yields a more versatile algorithm in two ways. First, it leads to updates that are applied in parallel at all the hidden units of the network, and thus unlike those derived from block coordinate descent, they are not specialized to layered networks whose structure suggests a natural partition (i.e., blocking) of the hidden units. Second, it yields closed-form updates to minimize the $\ell_{p,q}$-norm despite the more complicated gradients that arise when $p \neq q$. This more general case is of interest for deep learning because the so-called *max-norm* emerges from eq. (1) in the limit $q \to \infty$ (Neyshabur et al., 2015a).

1.2 From statics to dynamics

Norm-minimizing rescaling transformations also have the interesting property that they balance the incoming and outgoing weights at each hidden unit in the network. The balancedness of weights across layers has been a key tool for proving convergence theorems in deep learning (Du et al., 2018; Arora et al., 2019). We also analyze the optimization landscape in networks with rescaling symmetries and consider how the balancedness of weights can be preserved during learning—that is, in the course of adapting the network's weights and biases.

This analysis yields an interesting counterpart to Noether's theorem for physical systems (Noether, 1918). When a physical system is described by a Lagrangian, its dynamics are specified by a least action principle, and Noether's theorem gives a recipe to deduce the conserved quantities that are generated by its symmetries. The networks we study in this paper have a symmetry group of rescaling transformations, but here we have needed to solve a sort of *inverse* problem. Specifically, in our case, the conserved quantities are specified by the choice of regularizer, and instead we have needed a recipe to deduce a compatible dynamics for learning—that is, one in which the desired quantities are conserved.

It is easy to see how this inverse problem arises. Let $\mathcal{E}(\mathbf{\Theta})$ measure the training error of a network in terms of its weights and biases, denoted collectively by $\mathbf{\Theta} = (\mathbf{W}, \mathbf{b})$. Also, let $\mathcal{R}(\mathbf{W})$ denote a regularizer that penalizes large weights in some way. When this regularizer is minimized with respect to rescaling transformations, it induces a particular balance of the incoming and outgoing weights at each hidden unit of the network; see Fig. 1. This minimality is expressed by the zeros of derivatives with respect to rescaling factors at each hidden unit, and these zeros are the quantities that we seek to conserve for a weight-balancing flow. Thus we start from a regularizer $\mathcal{R}(\mathbf{W})$ and seek a dynamics that not only *conserves* these zeros, but also *descends* in a loss function of the form $\mathcal{L}(\mathbf{\Theta}) = \mathcal{E}(\mathbf{\Theta}) + \lambda \mathcal{R}(\mathbf{W})$ for any $\lambda \geq 0$.

The second main contribution of this paper is to give a recipe for such dynamics—namely, to derive weight-balancing flows such that a regularizer $\mathcal{R}(\mathbf{W})$ remains minimal with respect to rescaling transformations throughout the course of learning. This recipe is given by Theorem 3.3, and it has an especially simple form

$$\boxed{\frac{d}{dt}\left(W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}\right) = -W_{ij}\frac{\partial \mathcal{L}}{\partial W_{ij}}.} \tag{2}$$

Additionally, in Theorem 3.4, we prove that these flows descend under fairly general conditions in the loss function, $\mathcal{L}(\mathbf{\Theta})$. The dynamics in eq. (2) reduces (for nonzero weights) to an ordinary gradient flow, namely $\dot{\mathbf{W}} = \frac{1}{2}(\partial \mathcal{L}/\partial \mathbf{W})$, when the weights are regularized by their ℓ_2-norm (i.e., taking $\mathcal{R}(\mathbf{W}) = \frac{1}{2}\sum_{ij} W_{ij}^2$), but

it suggests a much richer set of flows for other regularizers, including but not limited to those based on the $\ell_{p,q}$-norm in eq. (1).

These results also build on previous work. In particular, Tanaka & Kunin (2021) have given an elegant and broadly unifying treatment of these ideas, one that both extends our conceptual understanding of Noether's theorem and also shows procedurally how to derive flows with these properties in overparameterized networks with general symmetry groups. Their treatment in turn builds on the highly influential work of Wibisono et al. (2016), who introduced Bregman Lagrangians in order to analyze accelerated methods through the lens of continuous-time dynamics. These approaches have already showcased the remarkable power and scope of variational methods. Nevertheless, we believe that some readers will appreciate the alternate (and somewhat more pedestrian) path that we have taken to obtain similar results, as well as the particular focus of this paper on networks with rescaling symmetries. In the learning of these overpararmeterized networks, either one can proceed in a top-down fashion, starting from a time-dependent Lagrangian with kinetic energies derived from Bregman distances, or one can proceed in a bottom-up fashion, starting from a regularized loss function and insisting that the weights remain balanced in a manner consistent with the choice of regularizer. Either way, however, one reaches the same conclusion. *Though gradient flow has served as a conceptual foundation for deep learning, these frameworks suggest a canonical pairing of alternative flows and regularizers.* Likewise, each weight-balancing flow in this paper can also be viewed as a type of generalized gradient flow, one in which the weights are reparameterized in a purposeful way and the gradient is preconditioned by the inverse Hessian of the regularizer.

The paper is organized as follows. Section 2 considers how to minimize the $\ell_{p,q}$-norm in eq. (1) with respect to rescaling transformations and proves the convergence of simple multiplicative updates to a global minimizer. Though the paper offers a mostly theoretical study, this section also presents some illustrative experimental results. Section 3 analyzes the optimization landscape in feedforward networks with rescaling symmetries and derives the weight-balancing flows in eq. (2). Finally, section 4 discusses many directions for future work.

2 Minimum-norm gauge fixing

In this section we consider how to balance the weights in feedforward networks with homogeneous activation functions without changing the end-to-end function that these networks compute. To begin, section 2.1 describes the symmetry group of rescaling transformations in these networks, and section 2.2 presents multiplicative updates to optimize over the elements of this group. Further intuitions are developed in section 2.3, which analyzes the fixed points of these updates, and also in section 2.4, which relates the updates to those considered in previous work. Finally, section 2.5 gives a formal proof that these updates converge to a global minimizer, and section 2.6 demonstrates empirically that they converge quickly 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 function $g : \Re \to \Re$ at each non-input unit. 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

$$h_i = g\Big(\sum_j W_{ij} h_j + b_i\Big) \tag{3}$$

to the remaining units, and setting $f_i(\mathbf{x}) = h_{n+i-r}$ from the i^{th} output unit. Since the network is feedforward, its weight matrix is strictly lower triangular. We also assume that the output units are unconnected (with $W_{ij}=0$ if $j \in \mathcal{O}$ and $i>j$) and that the same activation function is used at every hidden unit.

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

$$g(az) = ag(z) \tag{4}$$

for all $a>0$. In this paper we focus on networks of rectified linear hidden units (ReLUs), with the activation function $g(z) = \max(0, z)$ at all $i \in \mathcal{H}$, but the property in eq. (4) 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. (4), 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.

Our results in this paper apply generally to feedforward networks (i.e., those represented by directed graphs) with homogeneous activation functions. For obvious reasons, the case of layered architectures is of special interest, but we will not assume this structure in what follows. We will assume, however, that the weight matrix $\mathbf{W}$ is such that the network has no *vacuous* hidden units, or equivalently, that every hidden unit of the network plays some role in the computation of mapping $\mathbf{x} \to \mathbf{f}(\mathbf{x})$. For example, a minimal requirement is that for each $i \in \mathcal{H}$, there exists some $j<i$ such that $W_{ij} \neq 0$, and also some $j>i$ such that $W_{ji} \neq 0$.

In feedforward networks with homogeneous activation functions, a gauge transformation is specified by a set of rescaling factors, one for each hidden unit. We use

$$\mathcal{A} = \{\mathbf{a} \in \Re^n \,|\, a_i > 0 \text{ if } i \in \mathcal{H}, a_i = 1 \text{ otherwise}\} \tag{5}$$

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

$$W_{ij} \leftarrow W_{ij}(a_i/a_j), \tag{6}$$

$$b_i \leftarrow b_i a_i. \tag{7}$$

It may seem redundant in eq. (5) 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. (6–7) 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 $\mathbf{a} \in \mathcal{A}$. We also use $\mathbf{W}' \sim \mathbf{W}$ and $\mathbf{W}' \overset{\mathbf{a}}{\sim} \mathbf{W}$ to denote these same equivalences for pairs of weight matrices.

2.2 Multiplicative updates

The first goal of this paper is to solve the following problem. Given a feedforward network with weights $\mathbf{W}_0$ and biases $\mathbf{b}_0$, we seek the functionally equivalent network, with $(\mathbf{W}, \mathbf{b}) \sim (\mathbf{W}_0, \mathbf{b}_0)$, such that $\mathbf{W}$ has the smallest possible norm in eq. (1). We provide an iterative solution to this problem in Algorithm 1, which computes $\mathbf{W}$ and $\mathbf{b}$ via a *sequence* of rescaling transformations. Each rescaling transformation is in turn implemented by a multiplicative update of the form in eqs. (6–7). 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. In this section, we describe the basic ingredients of this algorithm.

Algorithm 1 Given a network with weights $\mathbf{W}_0$ and biases $\mathbf{b}_0$, this procedure returns a functionally equivalent network whose rescaled weights $\mathbf{W}$ and biases $\mathbf{b}$ minimize the norm $\|\mathbf{W}\|_{p,q}$ in eq. (1) up to some tolerance $\delta > 0$. The set $\mathcal{H}$ contains the indices of the network's hidden units. The rescaled weights and biases are computed via a sequence of multiplicative updates. The key step per update is to compute the rescaling factor a_i at each hidden unit $i \in \mathcal{H}$.

procedure $(\mathbf{W}, \mathbf{b}) = \textsc{MinNorm}(\mathbf{W}_0, \mathbf{b}_0, \mathcal{H}, p, q, \delta)$
  $(\mathbf{W}, \mathbf{b}, \mathbf{a}) \leftarrow (\mathbf{W}_0, \mathbf{b}_0, \mathbf{1})$ ▷ *Initialize.*
  repeat
    for all (i, j) **do**
      $\pi_{ij}(\mathbf{W}) \leftarrow \frac{|W_{ij}|^p}{\sum_k |W_{ik}|^p}$ ▷ *Compute stochastic matrix.*
    for all i **do**
      $\rho_i \leftarrow \left(\sum_j |W_{ij}|^p\right)^{\frac{q}{p}}$. ▷ *Compute per-unit regularizers.*
    for all $i \in \mathcal{H}$ **do**
      $a_i \leftarrow \left[\frac{\sum_j \rho_j(\mathbf{W})\, \pi_{ji}(\mathbf{W})}{\rho_i(\mathbf{W})}\right]^{\frac{1}{4\max(p,q)}}$ ▷ *Compute rescaling factors.*
    for all (i, j) **do**
      $W_{ij} \leftarrow W_{ij}(a_i / a_j)$ ▷ *Rescale weights.*
    for all i **do**
      $b_i \leftarrow b_i a_i$ ▷ *Rescale biases.*
      $\delta_i \leftarrow |a_i - 1|$
  until $\max_i(\delta_i) < \delta$ ▷ *Iterate until convergence.*

To begin, we define two intermediate quantities that play important recurring roles in both the multiplicative updates of this section and the weight-balancing flows of section 3. The first of these is the *per-unit regularizer*

$$\rho_i(\mathbf{W}) = \left(\sum\nolimits_j |W_{ij}|^p\right)^{\frac{q}{p}}, \tag{8}$$

which we use to denote the local contribution to the norm in eq. (1) from the incoming weights at the i^{th} unit in the network. The second of these quantities is the *stochastic matrix*

$$\pi_{ij}(\mathbf{W}) = \frac{|W_{ij}|^p}{\sum_k |W_{ik}|^p} \tag{9}$$

whose columns sum to one. Note that these quantities depend on the values of p and q, in the $\ell_{p,q}$-norm, but to avoid an excess of notation, we will not explicitly indicate this dependence. Another notable property is that these quantities only depend on the weights through their magnitudes.

We now explain the roles played by these quantities in Algorithm 1. In this algorithm, the key step per update is to compute the rescaling factor a_i at each hidden unit of the network. In terms of the above quantities, this rescaling factor takes the especially simple form

$$a_i = \left[\frac{\sum_j \rho_j(\mathbf{W})\, \pi_{ji}(\mathbf{W})}{\rho_i(\mathbf{W})}\right]^{\frac{1}{4\max(p,q)}}. \tag{10}$$

Finally, we note that many of the computations in eqs. (8–10) can be easily parallelized, and also that the updates in eqs. (6–7) can be applied in parallel after computing the rescaling factors in eq. (10).

We derive the form of these rescaling factors (and particularly, the curious value of their outer exponent) in section 2.5. 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 $\ell_{p,q}$-norm*

$$\operatorname*{argmin}_{\mathbf{W},\mathbf{b}} \|\mathbf{W}\|_{p,q} \quad \text{such that} \quad (\mathbf{W},\mathbf{b}) \sim (\mathbf{W}_0,\mathbf{b}_0). \tag{11}$$

Also, the intermediate solutions from these updates yield monotonically decreasing values for these norms.

We defer the proof of this theorem so that we may first develop more basic intuitions. We begin by contrasting the goals and guarantees of this gauge-fixing procedure versus those of typical learning algorithms. Notably, the updates in Algorithm 1 by themselves do *not* constitute a learning algorithm; they only optimize over the set $\mathcal{A}$ of gauge transformations in eq. (5), and they do not change the end-to-end functions of the networks to which they are applied. The theorem shows that the optimization in eq. (11) is more tractable than the problem of learning, and as a result, the multiplicative updates for gauge fixing have stronger guarantees than (say) learning via back-propagated gradients. In particular, these updates converge monotonically to a global minimizer of the norm $\|\mathbf{W}\|_{p,q}$, 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, and this has indeed served as motivation for many previous studies of symmetry in deep learning (Neyshabur et al., 2015a; Badrinarayanan et al., 2015; Meng et al., 2019; Stock et al., 2019; Armenta et al., 2021; Zhao et al., 2022).

2.3 Analysis of fixed points

We can obtain further intuition for the multiplicative updates in Algorithm 1 by analyzing their fixed points. We begin with an elementary observation of where they occur.

Proposition 2.2 (Fixed Points). *If a weight matrix $\mathbf{W}$ is a fixed point of the multiplicative updates in Algorithm 1, then all the rescaling factors in eq. (10) are equal to unity: i.e., $a_i = 1$ for all i.*

Proof. The converse of this proposition is trivial, but we desire to prove the other direction of implication. We proceed by induction. It is clear that $a_1 = 1$ because the network must have least one input unit and the rescaling factors at all non-hidden units are fixed to one. Let $i > 1$ and suppose that $a_j = 1$ for all $j < i$. The i^{th} unit in the network is either hidden ($i \in \mathcal{H}$) or not hidden ($i \notin \mathcal{H}$). If the latter, then as before we have trivially that $a_i = 1$. If the former, then there must exist some j such that $W_{ij} \neq 0$ (since otherwise the unit would be unaffected by the network's inputs, a condition that we exclude). Note that the weight W_{ij} is multiplicatively rescaled in eq. (6) by the ratio a_i/a_j. Since $W_{ij} \neq 0$, a fixed point occurs only when $a_i = a_j$, and since $a_j = 1$ by the inductive hypothesis, it follows that $a_i = 1$. □

Next we consider what the fixed points of Algorithm 1 signify. Recall that the updates were introduced to minimize the norm in eq. (1). The next lemma shows that the fixed points correspond to global minima.

Lemma 2.3 (Global Optimality). *If a weight matrix $\mathbf{W}$ is a fixed point of the multiplicative updates in Algorithm 1, then there exists no weight matrix $\mathbf{W}' \sim \mathbf{W}$ such that $\|\mathbf{W}'\|_{p,q} < \|\mathbf{W}\|_{p,q}$.*

Proof. Suppose that $\mathbf{W}' \overset{\mathbf{a}}{\sim} \mathbf{W}$, and consider how $\|\mathbf{W}'\|_{p,q}^q$ depends on the rescaling 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

$$F(\mathbf{a}) = \frac{1}{q} \sum_i \left(\sum_j |W_{ij}|^p (a_i/a_j)^p \right)^{\frac{q}{p}}. \tag{12}$$

It is instructive to examine the dependence of this function on the logarithms of the scaling factors, $\log a_i$, rather than the scaling factors themselves. Doing so, we find:

$$F(\mathbf{a}) = \frac{1}{q} \sum_i \exp\left(\frac{q}{p} \log \sum_j e^{p\left[\log a_i - \log a_j + \log |W_{ij}|\right]} \right). \tag{13}$$

It follows from basic composition laws of convex functions (Boyd & Vandenberghe, 2004) that $\|\mathbf{W}'\|_{p,q}^q$ is itself a convex function of the vector $\log \mathbf{a} = (\log a_1, \log a_2, ..., \log a_n)$, and this in turn implies that *any stationary point of F is a global minimizer of F*. To locate such a point, we examine where the partial derivatives of F vanish. Starting from eq. (12), a tedious but straightforward calculation gives

$$\frac{\partial F}{\partial a_i} = \frac{1}{a_i} \left[\left(\sum_j |W_{ij}|^p (a_i/a_j)^p \right)^{\frac{q}{p}} - \sum_j \left(\sum_k |W_{jk}|^p (a_j/a_k)^p \right)^{\frac{q}{p}-1} |W_{ji}|^p (a_j/a_i)^p \right]. \tag{14}$$

Now suppose that $\mathbf{W}$ is a fixed point of the multiplicative updates. Then by the previous proposition, it must be the case that the right hand side of eq. (10) is equal to unity. This occurs when the denominator and numerator of eq. (10) are equal, or equivalently when

$$\left(\sum_j |W_{ij}|^p \right)^{\frac{q}{p}} = \sum_j \left(\sum_k |W_{jk}|^p \right)^{\frac{q}{p}-1} |W_{ji}|^p. \tag{15}$$

Eq. (15) can be viewed as a balancing condition that holds at fixed points of the multiplicative updates: it equates a sum over incoming weights (on the left) with a sum over outgoing weights (on the right). Comparing the last two equations, we see another consequence of this balance; it implies that gradient of F in eq. (14) vanishes when $\mathbf{a} = \mathbf{1}$, where $\mathbf{1} \in \Re^n$ is the vector of all ones. But this implies that a global minimum of $F(\mathbf{a})$ is obtained when $\mathbf{a}$ is the *identity* gauge transformation. In this case, 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 $\|\mathbf{W}'\|_{p,q} < \|\mathbf{W}\|_{p,q}$. □

To summarize, we have *not yet* shown that the updates in Algorithm 1 converge. But we have shown that if they *do* converge, they yield a functionally equivalent network whose weights minimize the norm in eq. (1).

2.4 Relation to previous work

Consider the special case of the norm in eq. (1) when $p = q$. This special case leads to many simplifications. When $p = q$, for example, the partial derivative in eq. (14) reduces to

$$\frac{\partial F}{\partial a_i} = \frac{1}{a_i} \left[\sum_j |W_{ij}|^p (a_i/a_j)^p - \sum_j |W_{ji}|^p (a_j/a_i)^p \right]. \tag{16}$$

This expression is sufficiently simple to pursue a strategy of coordinate descent, minimizing $F(\mathbf{a})$ with respect to one rescaling factor at a time. In particular, consider the rescaling factor a_i at some hidden unit ($i \in \mathcal{H}$), and suppose that $a_j = 1$ for all $j \neq i$. Then the partial derivative in eq. (16) vanishes when

$$a_i = \left[\frac{\sum_j |W_{ji}|^p}{\sum_j |W_{ij}|^p} \right]^{\frac{1}{2p}}. \tag{17}$$

Eq. (17) is the basis for the ENorm algorithm (Stock et al., 2019), which uses multiplicative updates with rescaling factors, as in eqs. (6–7), to minimize the $\ell_{p,p}$-norm of the weight matrix. The updates for ENorm are derived from *block* coordinate descent of eq. (12) for the special case $p = q$; the blocks are composed of hidden units in the same layer of a deep network (for which the partial derivatives in eq. (16) decouple). As its name suggests, the procedure aims to equalize the p-norms of incoming and outgoing weights.

When $p=q$, the astute reader may have noticed a mismatch in the outer exponents of eqs. (10) and (17); the former is $\frac{1}{4p}$ for the multiplicative updates in this paper, while the latter is $\frac{1}{2p}$ for ENorm. The origin of this discrepancy, by a factor of two, will be further explained in section 2.5. At a high level, the discrepancy arises because ENorm is based on block coordinate descent whereas the updates in this paper are applied in parallel at all the hidden units of the network. The larger exponent of ENorm corresponds to a more aggressive update, but one that is only applied to the rescaling factors in one layer of hidden units. The smaller exponent in our approach is needed to provide a similar guarantee of monotonic convergence when all the rescaling factors are updated in parallel. Compared to previous work, our results can be viewed as extending the reach of parallelizable multiplicative updates in two directions—first, to the larger family of feedforward (but not strictly layered) networks, and second, to the larger family of norms when $p \neq q$.

There have also been many previous studies of *learning* in feedforward networks with rescaling symmetries (Neyshabur et al., 2015a; Badrinarayanan et al., 2015; Meng et al., 2019; Stock et al., 2019; Armenta et al., 2021; Zhao et al., 2022). We review these in section 3 where we discuss the problem of learning.

2.5 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).

We prove the theorem with the aid of two additional lemmas. The first lemma considers the effect of a single rescaling transformation in the gauge-fixing procedure of Algorithm 1. This lemma is at the heart of the algorithm: it shows that each multiplicative update reduces the entry-wise $\ell_{p,q}$-norm of the weight matrix.

Lemma 2.4 (Monotone Improvement)**.** *Let $\mathbf{W}' \overset{\mathbf{a}}{\sim} \mathbf{W}$, where $\mathbf{a} \in \mathcal{A}$ is the vector of rescaling factors whose elements for $i \in \mathcal{H}$ are given by eq. (10). Then $\|\mathbf{W}'\|_{p,q} \leq \|\mathbf{W}\|_{p,q}$, and the inequality is strict unless $\mathbf{W}' = \mathbf{W}$.*

Proof. Recall the function $F : \mathcal{A} \to \Re$ from eq. (12), and recall in particular that $F(\mathbf{a}) = q^{-1}\|\mathbf{W}'\|_{p,q}^q$ if $\mathbf{W}' \overset{\mathbf{a}}{\sim} \mathbf{W}$. To prove the lemma, we must show equivalently that $F(\mathbf{a}) \leq F(\mathbf{1})$ when $\mathbf{a} \in \mathcal{A}$ is given by eq. (10) and also that this inequality is strict unless $\mathbf{a} = 1$. The proof is based on constructing a so-called auxiliary function, as in the derivations of the EM algorithm (Dempster et al., 1977), nonnegative matrix factorization (Lee & Seung, 2000), and the convex-concave procedure (Yuille & Rangarajan, 2003). Specifically, we seek an auxiliary function $G : \mathcal{A} \to \Re$ with the following three properties:

(i) $F(\mathbf{a}) \leq G(\mathbf{a})$ for all $\mathbf{a} \in \mathcal{A}$.

(ii) $F(\mathbf{1}) = G(\mathbf{1})$ where $\mathbf{1}$ is the vector of all ones.

(iii) $G : \mathcal{A} \to \Re$ has a unique global minimizer at $\mathbf{a} \in \mathcal{A}$ given by eq. (10).

Suppose, for instance, that we can construct a function with these properties. Then at the minimizer $\mathbf{a} \in \mathcal{A}$ given by eq. (10), we have at once that

$$F(\mathbf{a}) \leq G(\mathbf{a}) \leq G(\mathbf{1}) = F(\mathbf{1}), \tag{18}$$

where the inequalities in eq. (18) follow from properties (i) and (iii) of the auxiliary function and the equality follows from property (ii). This proves the first part of the lemma. Now if $\mathbf{a} \neq \mathbf{1}$ in eqs. (10) and (18), then $G(\mathbf{a}) < G(\mathbf{1})$ (because G has a *unique* global minimizer) and by extension $F(\mathbf{a}) < F(\mathbf{1})$. On the other hand, if $\mathbf{a} = \mathbf{1}$, then $\mathbf{W}$ is a fixed point of the multiplicative updates, and this proves the second part of the lemma.

The more challenging part of the proof is to *construct* an auxiliary function with these properties. As shorthand, let $r = \max(p, q)$, and consider the function

$$G(\mathbf{a}) \;=\; \tfrac{1}{2r} \sum_{ij} \rho_i(\mathbf{W})\, \pi_{ij}(\mathbf{W}) \left(a_i^{2r} + a_j^{-2r}\right) \;+\; \max\left(0, 1 - \tfrac{q}{p}\right) F(\mathbf{1}). \tag{19}$$

We claim that eq. (19) satisfies the three properties listed above, and we verify them in order of difficulty. First, we verify property (ii): comparing eqs. (12) and (19), we see by simple substitution that $F(\mathbf{1}) = G(\mathbf{1})$. Next we verify property (iii). We begin by observing that $\sum_j |W_{ij}|^p > 0$ and $\sum_j |W_{ji}|^p > 0$ for all $i \in \mathcal{H}$; these conditions are necessary (as mentioned in section 2.1) to ensure that each hidden unit lies on some directed path from the network's inputs to outputs. It follows that the coefficients of a_i^{2r} and a_j^{-2r} in eq. (19) are strictly positive, and hence $G(\mathbf{a})$ is *strongly convex* in the variable $(a_1^r, a_2^r, \ldots, a_n^r)$. By setting $\partial G / \partial a_i^r = 0$ for all $i \in \mathcal{H}$, we obtain the solution in eq. (10), and by strong convexity this solution must correspond to a unique global minimizer. This verifies property (iii) of the auxiliary function, and it also accounts for the peculiar exponent of $\frac{1}{4r}$ where $r = \max(p, q)$, that appears in the multiplicative update.

Finally, we verify property (i) of the auxiliary function. To do this, we must work out separately the cases for $p < q$ and $p > q$. These regimes require different arguments to establish the lower-bounding property of the auxiliary function. We also remind the reader of the definitions in eqs. (8–9).

- **Case 1:** $p \leq q$. To verify property (i) in this case we must show that $F(\mathbf{a}) \leq G(\mathbf{a})$ when $p \leq q$. Starting from eq. (12) for $F(\mathbf{a})$, we can derive the auxiliary function in eq. (19) via two simple inequalities:

$$\begin{aligned}
F(\mathbf{a}) \;&=\; \tfrac{1}{q} \sum_i \left(\sum_j |W_{ij}|^p (a_i / a_j)^p \right)^{\frac{q}{p}}, && (20)\\
&=\; \tfrac{1}{q} \sum_i \rho_i(\mathbf{W}) \left(\sum_j \pi_{ij}(\mathbf{W})\, (a_i / a_j)^p \right)^{\frac{q}{p}}, && (21)\\
&\leq\; \tfrac{1}{q} \sum_i \rho_i(\mathbf{W}) \sum_j \pi_{ij}(\mathbf{W}) (a_i / a_j)^q, && (22)\\
&\leq\; \tfrac{1}{2q} \sum_{ij} \rho_i(\mathbf{W})\, \pi_{ij}(\mathbf{W}) \left(a_i^{2q} + a_j^{-2q}\right), && (23)\\
&=\; G(\mathbf{a}) \quad \text{for} \quad p \leq q. && (24)
\end{aligned}$$

  In eq. (22) we have used Jensen's inequality, exploiting the fact that π_{ij} is a stochastic matrix, and in eq. (23) we have appealed to the inequality of arithmetic and geometric means. Finally, eq. (24) follows upon resubstituting eq. (9) into the previous line. This verifies property (i) for the case $p \leq q$.

- **Case 2:** $p \geq q$. Now we must show that $F(\mathbf{a}) \leq G(\mathbf{a})$ when $p \geq q$. Here we exploit the fact that a concave function (e.g., $x^{\frac{q}{p}}$) lies below any one of its tangents. It follows that

$$\begin{aligned}
F(\mathbf{a}) \;&=\; \tfrac{1}{q} \sum_i \left(\sum_j |W_{ij}|^p (a_i / a_j)^p \right)^{\frac{q}{p}}, && (25)\\
&\leq\; F(\mathbf{1}) \;+\; \tfrac{1}{p} \sum_{ij} \left(\sum_k |W_{ik}|^p \right)^{\frac{q}{p} - 1} |W_{ij}|^p \left((a_i / a_j)^p - 1 \right), && (26)\\
&=\; F(\mathbf{1}) \;+\; \tfrac{1}{p} \sum_{ij} \rho_i(\mathbf{W})\, \pi_{ij}(\mathbf{W}) \left((a_i / a_j)^p - 1 \right), && (27)\\
&\leq\; F(\mathbf{1}) \left(1 - \tfrac{q}{p}\right) \;+\; \tfrac{1}{2p} \sum_{ij} \rho_i(\mathbf{W})\, \pi_{ij}(\mathbf{W}) \left(a_i^{2p} + a_j^{-2p}\right) && (28)\\
&=\; G(\mathbf{a}) \quad \text{for} \quad p \geq q. && (29)
\end{aligned}$$

In sum, the auxiliary function $G(\mathbf{a})$ is derived by identifying those parts of $\|\mathbf{W}'\|_{p,q}^q$ that can be bounded by elementary inequalities. In doing so, we find that different bounds are required for the cases $p<q$ and $p>q$, and these differences account for the exponent $r = \max(p, q)$ that appears in the multiplicative updates. □

Lemma 2.4 shows that each multiplicative update in Algorithm 1 decreases $\|\mathbf{W}\|_{p,q}$ unless $\mathbf{W}$ is itself a fixed point. The lemma also rules out oscillations between distinct global minima. But this by itself is not enough to prove that the updates converge. To do this, we must also show that none of the rescaling factors increase without bound. This is the content of the next lemma.

Lemma 2.5 (Compactness of Sublevel Sets)**.** *Let $\mathcal{C} = \{\mathbf{W}' \,|\, \mathbf{W}' \sim \mathbf{W}, \|\mathbf{W}'\|_{p,q} \leq \|\mathbf{W}\|_{p,q}\}$ denote the sublevel set of weight matrices whose $\ell_{p,q}$-norm does not exceed that of $\mathbf{W}$. This set is compact.*

Proof. Let $F : \mathcal{A} \to \Re$ be the function, as defined in eq. (12), that computes the change in the $\ell_{p,q}$-norm of the weight matrix after a rescaling transformation. To prove the lemma, we must show equivalently that the sublevel set given by $\mathcal{F}_{\mathbf{1}} = \{\mathbf{a} \in \mathcal{A} \,|\, F(\mathbf{a}) \leq F(\mathbf{1})\}$ is compact. 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 boundedness will follow from the fact that the network has finite depth; a similar result has been obtained for the special case $p=q$ (Stock et al., 2019). 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

$$\frac{a_i}{a_j} \leq \frac{(qF(\mathbf{1}))^{\frac{1}{q}}}{|W_{ij}|}, \tag{30}$$

because otherwise the ij^{th} term of the sum in eq. (12) 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

$$a_i \;=\; \frac{a_{j_k}}{a_{j_0}} \;=\; \prod_{\ell=1}^{k} \frac{a_{j_\ell}}{a_{j_{\ell-1}}} \;\leq\; \prod_{\ell=1}^{k} \frac{(qF(\mathbf{1}))^{\frac{1}{q}}}{|W_{j_\ell j_{\ell-1}}|}, \tag{31}$$

where the inequality follows from eq. (30). Likewise, since $a_{j_m} = 1$ and $a_{j_k} = a_i$, it follows that

$$\frac{1}{a_i} \;=\; \frac{a_{j_m}}{a_{j_k}} \;=\; \prod_{\ell=k+1}^{m} \frac{a_{j_\ell}}{a_{j_{\ell-1}}} \;\leq\; \prod_{\ell=k+1}^{m} \frac{(qF(\mathbf{1}))^{\frac{1}{q}}}{|W_{j_\ell j_{\ell-1}}|} \tag{32}$$

Eqs. (31–32) 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. □

The previous two lemmas establish the necessary conditions for Meyer's monotone convergence theorem (Meyer, 1976). Armed with these lemmas, we can now prove Theorem 2.1.

Proof of Theorem 2.1. Let $\mathbf{W}_0$ denote an initial weight matrix. From Lemma 2.5, it follows that the set $\mathcal{C} = \{\mathbf{W} \,|\, \mathbf{W} \sim \mathbf{W}_0, \|\mathbf{W}\|_{p,q} \leq \|\mathbf{W}_0\|_{p,q}\}$ is compact. From Lemma 2.4, it follows that each multiplicative update yields a weight matrix $\mathbf{W} \in \mathcal{C}$ whose $\ell_{p,q}$-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, from eq. (10) we note that the multiplicative coefficients 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 monotone convergence theorem (Meyer, 1976) in the setting where fixed points occur at (and only at) global minima of $\|\mathbf{W}\|_{p,q}$ in $\mathcal{C}$. □

2.6 Demonstration of convergence

In the last section, we proved the *asymptotic* convergence of the multiplicative updates in Algorithm 1. These updates would be of limited use, however, if their rate of convergence was impractically slow. One would not expect this to be the case, given the widespread use of multiplicative updates in other problems with nonnegativity constraints (Lee & Seung, 1999; Sha et al., 2007). In this section, we use the multiplicative updates in Algorithm 1 to rebalance the weights in different networks and for different values of p and q. Empirically we find that they converge to reasonable precision within a modest number of iterations.

Fig. 2 plots the convergence of the multiplicative updates in Algorithm 1 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 panels in the figure plot the ratio $\|\mathbf{W}\|_{p,q}/\|\mathbf{W}_0\|_{p,q}$ as a function of the number of multiplicative updates, where $\|\mathbf{W}_0\|_{p,q}$ and $\|\mathbf{W}\|_{p,q}$ are respectively the $\ell_{p,q}$-norms, defined in eq. (1), of the initial and updated weight matrix. Results are shown for several values for p and q.

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 iterations. The panels show that conventionally initialized networks are (i) far from minimal as measured by the $\ell_{p,q}$-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.

3 Learning and regularization

The rescaling symmetry in Fig. 1 also has important consequences for learning in a feedforward network with homogeneous activation functions (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). The goal of learning is to discover the weights and biases that minimize the network's loss function on a data set of training examples. In this section we examine the conditions for learning under which some norm or regularizer of the weight matrix (e.g., the $\ell_{p,q}$-norm) remains minimal with respect to rescaling transformations. Equivalently, these are the conditions for learning under the incoming and outgoing weights at each hidden unit remain *balanced* with respect to this norm. 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).

Our ultimate goal in this section is to derive the *weight-balancing flows* in eq. (2). Section 3.1 begins with a short review of gradient flow in networks with homogeneous activation functions. After this review, section 3.2 derives a more general family of flows; our main results, stated in Theorems 3.3 and 3.4, are that these flows descend in a regularized loss function while preserving the minimality of the regularizer with respect to rescaling transformations. Section 3.3 analyzes the specific form of these flows for regularizers based on the $\ell_{p,q}$-norm in eq. (1). Finally, section 3.4 discusses related work in deep learning that also capitalizes on the rescaling symmetries of feedforward networks.

3.1 Gradient flow

There are many forms of learning in deep networks. Perhaps the simplest to analyze is gradient flow (Elkabetz & Cohen, 2021), in which the network's parameters are adapted in continuous time along the negative gradient of the loss function. Gradient flows can be derived for any differentiable loss function, and they are widely used to study the behavior of gradient descent in the limit of small learning rates.

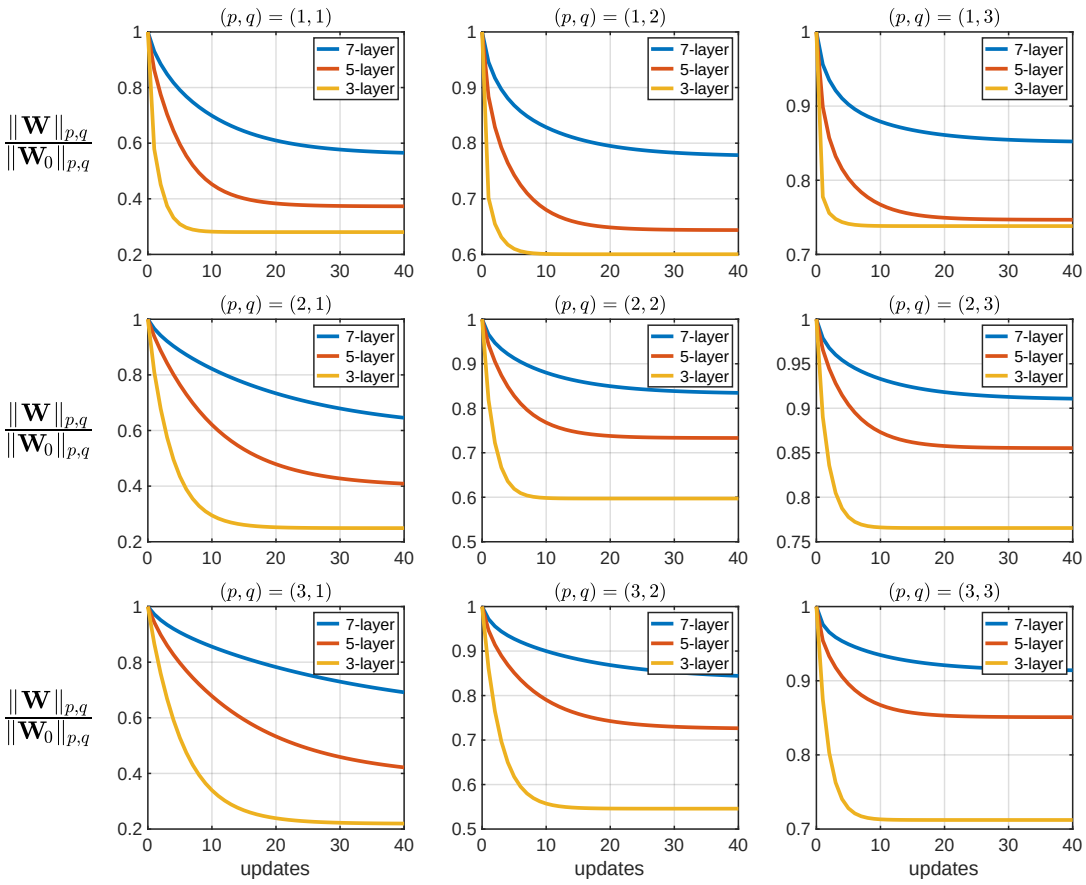

Figure 2: Convergence of gauge-fixing multiplicative updates in three randomly initialized networks with differing numbers of hidden layers but the same overall numbers of input, hidden, and output units. The updates minimize the $\ell_{p,q}$-norm of the weight matrix, and each panel shows the results for a different set of values for p and q. See text for details.

There are two properties of gradient flow in neural networks that we will need to generalize for the minimality-preserving flows in eq. (2). The first is the property of *descent*—namely, that the loss function of a network decreases over time. This is simple to demonstrate for gradient flow. Let $\mathbf{\Theta} = (\mathbf{W}, \mathbf{b})$ denote the weights and biases of the network, and suppose that the network is trained to minimize a loss function $\mathcal{L}(\mathbf{\Theta})$. Then under gradient flow, at non-stationary points of the loss function, we see that

$$\frac{d\mathcal{L}}{dt} = \frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}} \cdot \frac{d\mathbf{\Theta}}{dt} = -\left(\frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}}\right)^{\top} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}}\right) < 0. \tag{33}$$

From eq. (33) it is also clear that fixed points of gradient flow correspond to stationary points of the loss function. In the next section, we will see how these properties are preserved in more general flows.

Another important property of gradient flow emerges in feedforward networks with homogeneous activation functions. This property is a consequence of the rescaling symmetry in Fig. 1; when such a network is trained by gradient flow, their rescaling symmetries give rise to *conservation laws*, one at each hidden unit of the network (Du et al., 2018; Kunin et al., 2021; Bronstein et al., 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). These conservation laws do not depend on the details of the network's loss function, requiring

only that it is also under invariant[1] under the symmetry group of rescaling transformations. In particular, for any such loss function, it is known that the quantity

$$\Delta_i = b_i^2 + \sum_j W_{ij}^2 - \sum_j W_{ji}^2 \tag{34}$$

is conserved under gradient flow at each hidden unit $i \in \mathcal{H}$. The connection between symmetries and conservation laws is well known in physical systems from Noether's theorem (Noether, 1918), but it is worth noting that the dynamics of gradient flow were not historically derived from a Lagrangian.

Our next step is to derive the recover the conserved quantities in eq. (34) as a special case of a more general framework. In doing so, we will arrive—via a somewhat different route—at results analogous to those obtained from the continuous-time dynamics of damped Lagrangian systems (Tanaka & Kunin, 2021; Wibisono et al., 2016). For now, we wish simply to emphasize the logic that has been used to derive many conservation laws in neural networks: to start, one assumes that the network is trained by gradient flow in its weights and biases, and then, inspired by Noether's theorem in physical systems, one derives the conserved quantities that are implied by the network's symmetries. In the next section, we shall flip this script on its head.

3.2 Weight-balancing flows

In this section we investigate a larger family of flows for feedforward networks with homogeneous activation functions. When properly initialized, these flows will lead to conserved quantities even when the network's loss function is *not* invariant under rescaling transformations. In particular, we will be interested in conserving the balance of incoming and outgoing weights at each of the network's hidden units, and we will be interested in loss functions with regularizers that penalize large weights.

For the problems we consider, these loss functions take a standard form with two competing terms. The first term measures the predictive accuracy of the network. Let $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$ denote a training set of T labeled examples, and let $\mathcal{C}(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ denote the cost when a network's actual output $f(\mathbf{x})$ is evaluated against some reference output $\mathbf{y}$. We use

$$\mathcal{E}(\boldsymbol{\Theta}) = \frac{1}{T} \sum_t \mathcal{C}(\mathbf{y}_t, \mathbf{f}(\mathbf{x}_t)) \tag{35}$$

to denote the error obtained by averaging these costs over the training set. This error is the first term in the network's loss function; the second term is a regularizer, $\mathcal{R}(\mathbf{W})$, that penalizes[2] large weights. We denote the loss function by

$$\mathcal{L}(\boldsymbol{\Theta}) = \mathcal{E}(\boldsymbol{\Theta}) + \lambda \mathcal{R}(\mathbf{W}), \tag{36}$$

where the constant $\lambda \geq 0$ determines the amount of regularization. Note that while the error in eq. (35) 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. (6–7). On the other hand, typical regularizers are not invariant to rescaling transformations of the weights, and these are the sorts of regularizers that we consider here.

When a regularizer penalizes large weights, it is not only useful to prevent overfitting; in a feedforward network with rescaling symmetries, it also induces a natural way to measure how well the weights are balanced. Consider the effect of a rescaling transformation on the weights at any hidden unit $i \in \mathcal{H}$. A rescaling transformation with $a_i < 1$ will redistribute the magnitudes of these weights in a forward direction (from incoming weights to outgoing weights), while a rescaling transformation with $a_i > 1$ will redistribute the magnitudes of weights in a backward direction (from outgoing to incoming weights). This motivates the following definition.

[1] Typical regularizers break this invariance, but it is possible to construct those that do not (Neyshabur et al., 2015a).

[2] The regularizer penalizes large weights, but not large biases, as only the former cause the outputs of the network to depend sensitively on the inputs to the network.

Definition 3.1 (Weight balancedness)**.** *In a feedforward network with homogeneous activation functions, we say that the weights are balanced with respect to the regularizer $\mathcal{R}(\mathbf{W})$ if the regularizer is stationary with respect to infinitesimal rescaling transformations of the form in eq. (6).*

In section 2, we examined the minimality of the $\ell_{p,q}$-norm in eq. (1) with respect to rescaling transformations. The main ideas of this section, however, do not depend on that specific choice, and therefore in what follows we state them in the more general terms of the above definition. To begin, we describe more precisely what it means for the regularizer $\mathcal{R}(\mathbf{W})$, or indeed any differentiable function of the weight matrix, to be stationary at some point with respect to infinitesimal rescaling transformations. The following lemma has been operating implicitly in many calculations of the previous section.

Lemma 3.2 (Stationarity)**.** *Let $\mathcal{K}(\mathbf{W})$ be a differentiable function of the weight matrix in a feedforward network with homogeneous activation functions. Then $\mathcal{K}$ is stationary with respect to rescaling transformations at $\mathbf{W}$ if and only if*

$$0 \;=\; \sum_j \left(W_{ij} \frac{\partial \mathcal{K}}{\partial W_{ij}} - W_{ji} \frac{\partial \mathcal{K}}{\partial W_{ji}} \right) \quad \text{for all} \quad i \in \mathcal{H}. \tag{37}$$

Proof. Consider a rescaling transformation of the form $a_i = 1 + \delta a_i$ for all $i \in \mathcal{H}$, where δa_i denotes an infinitesimal variation. Under such a transformation, the weights of the network are rescaled via eq. (6) as $W_{ij} \leftarrow W_{ij}(1 + \delta a_i)/(1 + \delta a_j)$, so that to lowest order, their change is given by

$$\delta W_{ij} = W_{ij}(\delta a_i - \delta a_j). \tag{38}$$

Now we can work out (again to lowest order) the change $\delta\mathcal{K} = \mathcal{K}(\mathbf{W} + \delta\mathbf{W}) - \mathcal{K}(\mathbf{W})$ that results from the variation in eq. (38). From the chain rule we obtain

$$\delta\mathcal{K} = \sum_{ij} \frac{\partial \mathcal{K}}{\partial W_{ij}} \delta W_{ij} = \sum_{ij} \frac{\partial \mathcal{K}}{\partial W_{ij}} W_{ij}(\delta a_i - \delta a_j) = \sum_i \delta a_i \sum_j \left(\frac{\partial \mathcal{K}}{\partial W_{ij}} W_{ij} - \frac{\partial \mathcal{K}}{\partial W_{ji}} W_{ji} \right). \tag{39}$$

Now consider the statement of the lemma. If eq. (37) is satisfied, then it is clear from eq. (39) that $\delta\mathcal{K} = 0$, proving one direction of implication. Likewise if $\delta\mathcal{K} = 0$ for *arbitrary* infinitesimal variations δa_i, then each coefficient of δa_i in eq. (39) must vanish, proving the other direction. □

Regularizers were originally introduced to avoid overfitting of the training data by large networks (Goodfellow et al., 2016), but it is now widely appreciated that they also serve other purposes. It has been observed, for example, that regularizers can help to learn better models on the *training* data (Krizhevsky et al., 2012), suggesting that smaller weights in deep networks lead to better behaved gradients. Likewise, it has been observed that highly unbalanced weights lead to much slower training in networks with homogeneous activation functions; 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." All of the above observations suggest a role for learning algorithms that actively balance the weights of feedforward networks with homogeneous activation functions. Such algorithms can in turn be derived from the weight-balancing flows of the following theorem.

Theorem 3.3 (Weight-balancing flows)**.** *Let $\mathcal{L}(\boldsymbol{\Theta})$ denote the regularized loss in eq. (36), and consider a network whose weights are initially balanced with respect to the regularizer $\mathcal{R}(\mathbf{W})$ and whose biases at hidden units are frozen at zero. Then the weights will remain balanced with respect to $\mathcal{R}(\mathbf{W})$ if they evolve as*

$$\frac{d}{dt}\left(W_{ij} \frac{\partial \mathcal{R}}{\partial W_{ij}} \right) = -W_{ij} \frac{\partial \mathcal{L}}{\partial W_{ij}}. \tag{40}$$

The theorem shows how to construct a canonical, weight-balancing flow for any differentiable regularizer. Generalized flows of this nature for deep learning were first derived in the work of Tanaka & Kunin (2021), and they did so by building on the Bregman Lagrangians introduced by Wibisono et al. (2016). Theorem 3.3 builds further on the work of Tanaka & Kunin (2021), who also stated that "a future direction is to extend our analysis to the case of rescale symmetry of the homogeneous activation functions such as ReLU." We have not, in fact, extended their analysis in any meaningful way; our more modest goal in this paper has been to offer a proof of Theorem 3.3 that is stripped down to its bare essentials. We do this by deriving the flows of eq. (40) in a bottom-up fashion, starting from the requirement that balanced weights should remain balanced, as opposed to a top-down fashion, starting from a time-dependent Bregman Lagrangian and then taking certain limits of the Euler-Lagrange equations. Before proving the theorem, we also emphasize the additional stipulation that the biases of all hidden (but not output) units are frozen at zero: i.e., $\dot{b}_i = b_i = 0$ for all $i \in \mathcal{H}$. fAs we shall see, this requirement[3] arises because we defined balancedness with respect to a regularizer $\mathcal{R}(\mathbf{W})$ that penalizes large weights *but not large biases*.

Proof. By Lemma 3.2, the weights are balanced with respect to the regularizer $\mathcal{R}(\mathbf{W})$ if they satisfy the stationarity conditions (substituting $\mathcal{R}$ for $\mathcal{K}$) in eq. (37). We prove the theorem by showing that the zeros in these stationarity conditions are conserved quantities analogous to those in eq. (34). As shorthand, let

$$Q_i = \sum_j \left(W_{ij} \frac{\partial \mathcal{R}}{\partial W_{ij}} - W_{ji} \frac{\partial \mathcal{R}}{\partial W_{ji}} \right) \tag{41}$$

denote the imbalance of incoming and outgoing weights at a particular hidden unit. To prove that Q_i is conserved, we must show that $\dot{Q}_i = 0$ when the weights of the network evolve under the flow in eq. (40) from an initially balanced state. It follows from eq. (40) that

$$\frac{dQ_i}{dt} = -\sum_j \left(W_{ij} \frac{\partial \mathcal{L}}{\partial W_{ij}} - W_{ji} \frac{\partial \mathcal{L}}{\partial W_{ji}} \right). \tag{42}$$

Now we recall from eq. (36) that the network's loss function consists of two terms, an empirical error $\mathcal{E}(\mathbf{\Theta})$ that is invariant under the rescaling transformations in eqs. (6–7) and a regularizer $\mathcal{R}(\mathbf{W})$ with respect to which the weights are initially balanced. Substituting eq. (36) into (42), we obtain

$$\frac{dQ_i}{dt} = -\sum_j \left(W_{ij} \frac{\partial \mathcal{E}}{\partial W_{ij}} - W_{ji} \frac{\partial \mathcal{E}}{\partial W_{ji}} \right) - \lambda \sum_j \left(W_{ij} \frac{\partial \mathcal{R}}{\partial W_{ij}} - W_{ji} \frac{\partial \mathcal{R}}{\partial W_{ji}} \right). \tag{43}$$

Consider the first term on the right side of eq. (43) If the biases of hidden units are frozen at zero, then $\mathcal{E}(\mathbf{\Theta})$ is invariant under (and hence stationary with respect to) rescaling transformations of the weights *alone*, and the first term vanishes by Lemma 3.2 (substituting $\mathcal{E}$ for $\mathcal{K}$). Now consider the second term in eq. (43); we see from eq. (41) that this term is proportional to Q_i itself, so that

$$\frac{dQ_i}{dt} = -\lambda Q_i. \tag{44}$$

Finally, we observe that $Q_i = 0$ at time $t = 0$ if the weights are initially balanced. In this case, the only solution to eq. (44) is the trivial one with $Q_i = \dot{Q}_i = 0$ for all time. But this is equivalent to the statement that the weights remain indefinitely balanced. □

The calculation in this proof yields another insight. From eq. (44), we see that Q_i decays exponentially to zero *if the weights are not initialized in a balanced state*. The decay is caused by the regularizer, an effect

[3]Interestingly, there are other reasons besides weight-balancing to learn ReLU networks with zero biases. It has been noted, for example, that zero biases are necessary 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 flow (Lyu & Li, 2020).

that Tanaka & Kunin (2021) describe as the Noether learning dynamics. (In their framework, the Bregman kinetic energy plays the role of the regularizer, breaking the rescaling symmetry of the Lagrangian, and this broken symmetry in turn undoes the conservation law for the so-called Noether charge Q_i.) Though Q_i decays to zero in eq. (44) from an initially unbalanced state, we note that this decay may be slow for the typically small values of the regularization hyperparameter $\lambda > 0$ that are used in practice. 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.

A weight-balancing flow is only useful insofar as it also reduces the network's loss function. The weight-balancing flows in eq. (40) do not strictly follow the gradient of the loss function in eq. (36). Nevertheless, our final theorem shows that under relatively mild conditions these flows have the same property of descent.

Theorem 3.4 (Balanced descent)**.** *Let $\omega_{ij} = \log|W_{ij}|$. Then the weight-balancing flow in eq. (40) descends everywhere that the loss function is not stationary with respect to $\boldsymbol{\omega}$ and the regularizer has a positive definite Hessian with respect to $\boldsymbol{\omega}$:*

$$\frac{d\mathcal{L}}{dt} < 0 \quad \textit{wherever} \quad \left|\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right| > 0 \quad \textit{and} \quad \frac{\partial^2 \mathcal{R}}{\partial \boldsymbol{\omega} \partial \boldsymbol{\omega}^\top} \succ \mathbf{0}. \tag{45}$$

Proof. The property of descent emerges for these flows in a similarly straightforward way as for other generalized flows (Wibisono et al., 2016; Tanaka & Kunin, 2021). We begin by observing that the flow in eq. (40) takes a simpler form in terms of the variable $\boldsymbol{\omega}$. Since $\omega_{ij} = \log|W_{ij}|$, we have equivalently that $\omega_{ij} = \frac{1}{2}\log W_{ij}^2$, and differentiating we find $\frac{\partial \omega_{ij}}{\partial W_{ij}} = \frac{1}{W_{ij}}$. We can use the chain rule to differentiate in eq. (40) with respect to $\boldsymbol{\omega}$ instead of $\mathbf{W}$. In this way we find (in vector notation) that

$$\frac{d}{dt}\left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{\omega}}\right) = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}. \tag{46}$$

This equation specifies *implicitly* how the weights evolve in time through the form of the regularizer, $\mathcal{R}(\mathbf{W})$. To derive an explicit form for this evolution, we differentiate through the left side of the equation:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{\omega}}\right) \;=\; \frac{\partial^2 \mathcal{R}}{\partial \boldsymbol{\omega}\, \partial \boldsymbol{\omega}^\top} \cdot \frac{d\boldsymbol{\omega}}{dt}. \tag{47}$$

Note how the Hessian of the regularizer appears in this equation; as shorthand in what follows we denote this Hessian by $\mathbf{H}(\boldsymbol{\omega}) = \frac{\partial^2 \mathcal{R}}{\partial \boldsymbol{\omega}\, \partial \boldsymbol{\omega}^\top}$. Now suppose that $\mathbf{H}(\boldsymbol{\omega})$ is positive definite (hence also invertible) at the current values of the weights. Then combining eqs. (46–47), we see that

$$\boxed{\frac{d\boldsymbol{\omega}}{dt} = \; -\mathbf{H}^{-1}(\boldsymbol{\omega})\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right).} \tag{48}$$

In sum, we have shown that if $\mathbf{H}(\boldsymbol{\omega})$ is positive definite, then the weight-balancing flow in eq. (40) has the same dynamics as eq. (48). Thus we can also interpret these dynamics as a generalized gradient flow in which the weights are reparameterized in terms of $\omega_{ij} = \log|W_{ij}|$ and the gradient is preconditioned [4] by the inverse Hessian $\mathbf{H}^{-1}(\boldsymbol{\omega})$. Now suppose further that the gradient $\partial\mathcal{L}/\partial\boldsymbol{\omega}$ does not vanish. Then we have

$$\frac{d\mathcal{L}}{dt} \;=\; \frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} \cdot \frac{d\boldsymbol{\omega}}{dt} \;=\; -\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right)^\top \mathbf{H}^{-1}(\boldsymbol{\omega})\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right) \;<\; 0. \tag{49}$$

This suffices to prove the theorem, but further intuition may be gained by comparing the argument in eq. (49) to the analogous one for gradient flow in eq. (33). The main differences are the appearance of the inverse Hessian preconditioner (via the regularizer) and the change of variables (from W_{ij} to ω_{ij}). □

[4] Note that the gradient is preconditioned by the inverse Hessian of the *regularizer*, not the inverse Hessian of the *loss function*. The latter (about which one can say little) depends on the network's fit to the training data; the former does not.

So far we have only considered flows that adapt the *weights* of the network. In these flows, we have shown that the weights remain balanced if the biases at all *hidden* units are frozen at zero. Finally we show that this property of descent generalizes in a straightforward way to flows that also adapt the biases at *output* units.

Corollary 3.5 (Biased outputs)**.** *Suppose that the biases at output units are adapted by gradient flow in parallel with the weight-balancing flow of Theorem 3.3. This flow also descends in the regularized loss.*

Proof. In this case the loss function evolves in response to the changes of both the weights and the biases at output units. Extending eq. (49), we find

$$\frac{d\mathcal{L}}{dt} \;=\; \frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} \cdot \frac{d\boldsymbol{\omega}}{dt} + \sum_{i \in \mathcal{O}} \frac{\partial \mathcal{L}}{\partial b_i} \frac{db_i}{dt} \;=\; -\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right)^\top \mathbf{H}^{-1}(\boldsymbol{\omega}) \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}}\right) \; - \sum_{i \in \mathcal{O}} \left(\frac{\partial \mathcal{L}}{\partial b_i}\right)^2. \tag{50}$$

Thust $\mathcal{L}$ is strictly decreasing wherever (i) the hessian $\mathbf{H}(\boldsymbol{\omega})$ is positive definite (as in the previous theorem) and (ii) $\mathcal{L}$ is not stationary with respect to *both* the weights of the network and the biases at output units. □

3.3 Flows for $\ell_{p,q}$-norm regularization

In the last section we derived weight-balancing flows for any regularizer $\mathcal{R}(\mathbf{W})$. The derivation was general, assuming only that the regularizer was differentiable. In this section we work out the flow in eq. (40) for regularizers based on the $\ell_{p,q}$-norm in eq. (1). Specifically, we consider regularizers of the form

$$\mathcal{R}(\mathbf{W}) = \frac{1}{q} \sum_i \left(\sum_j |W_{ij}|^p \right)^{\frac{q}{p}}. \tag{51}$$

There are two main results in this section. The first is to derive a more suggestive form of this flow for discretization or numerical integration. The second is to show that it descends in the network's loss function.

To build intuition, we begin by noting some special cases of interest. The weight-balancing flow in eq. (40) reduces to familiar forms for the simplest choices of p and q in eq. (51). For example, when $p=q=2$, this flow simplifies (for *nonzero* weights) to $\dot{W}_{ij} = -\frac{1}{2}\frac{\partial \mathcal{L}}{\partial W_{ij}}$, which can be approximated by gradient descent

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}} \tag{52}$$

with a small learning rate $\eta > 0$. On the other hand, when $p = q = 1$, the flow in eq. (40) simplifies to $\dot{W}_{ij} = -|W_{ij}|\frac{\partial \mathcal{L}}{\partial W_{ij}}$, which can be approximated by the *exponentiated* gradient descent

$$W_{ij} \leftarrow W_{ij} \exp\left(-\eta \cdot \operatorname{sign}(W_{ij}) \cdot \frac{\partial \mathcal{L}}{\partial W_{ij}}\right) \tag{53}$$

Similar updates based on exponentiated gradients have been studied in many different contexts (Kivinen & Warmuth, 1997; Arora et al., 2012; Bernstein et al., 2020). With these special cases in mind, we now return to the more general situation.

We begin by recalling the per-unit regularizer $\rho_i(\mathbf{W})$ defined in eq. (8) and the stochastic matrix with elements $\pi_{ij}(\mathbf{W})$ defined in eq. (9). These quantities also play important roles in this section. For example, starting from the regularizer in eq. (51), it is a straightforward exercise to verify that

$$W_{ij} \frac{\partial \mathcal{R}}{\partial W_{ij}} \;=\; \pi_{ij}(\mathbf{W}) \rho_i(\mathbf{W}). \tag{54}$$

We use this relation repeatedly in what follows. Our first use is to demonstrate an interesting correspondence that arises for the regularizer in eq. (51).

Lemma 3.6 (Correspondence)**.** *Let $V_{ij} = W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}$ for the regularizer in eq. (51). Then the elements V_{ij} collectively determine, up to a sign, the corresponding elements W_{ij} of the network's weight matrix.*

Proof. The correspondence follows easily from eq. (54) and the fact that $\boldsymbol{\pi}(\mathbf{W})$ is a stochastic matrix whose rows sum to one. Thus we have

$$\rho_i(\mathbf{W}) = \sum\nolimits_j W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}} \;=\; \sum\nolimits_j V_{ij}, \tag{55}$$

$$\pi_{ij}(\mathbf{W}) = \big[\rho_i(\mathbf{W})\big]^{-1}\left(W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}\right) \;=\; \frac{V_{ij}}{\sum_k V_{ik}}, \tag{56}$$

$$|W_{ij}| = \big[\pi_{ij}(\mathbf{W})\big]^{\frac{1}{p}}\;\big[\rho_i(\mathbf{W})\big]^{\frac{1}{q}}. \tag{57}$$

Alternatively, the lemma states that there is a one-to-one correspondence between the matrices with elements $V_{ij} = W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}$ and the matrices with elements W_{ij} in any single orthant of weight space. □

We are now ready to prove corollaries of Theorems 3.3 and 3.4 for the special case of $\ell_{p,q}$–norm regularization. The first of these expresses the weight-balancing flow for $\ell_{p,q}$-norm regularization in a more suggestive form.

Corollary 3.7 (Minimum-norm flows)**.** *If the regularizer $\mathcal{R}(\mathbf{W})$ is given by eq. (51), then in any open orthant of weight space, the weight-balancing flow in eq. (40) is equivalent to integrating*

$$\frac{dV_{ij}}{dt} + \lambda V_{ij} \;=\; -W_{ij}\frac{\partial \mathcal{E}}{\partial W_{ij}}, \tag{58}$$

where the signs of the weights W_{ij} are determined by the orthant and the magnitudes are determined by the correspondence in eqs. (55–57).

Proof. The loss function in eq. (40) is given by $\mathcal{L}(\boldsymbol{\Theta}) = \mathcal{E}(\boldsymbol{\Theta}) + \lambda\mathcal{R}(\mathbf{W})$. Thus the weight-balancing flow can be written equivalently as

$$\frac{d}{dt}\left[W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}\right] + \lambda W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}} \;=\; -W_{ij}\frac{\partial \mathcal{E}}{\partial W_{ij}}. \tag{59}$$

The expression in eq. (58) is obtained by substituting the identities in eqs. (54–56), and in a particular orthant of weight space, these identities establish a one-to-one correspondence between the matrices $\mathbf{V}$ and $\mathbf{W}$. □

We note that eq. (58) is of the form required for a first-order exponential integrator method. We leave it to future work to develop effective discretizations of eq. (58) that could serve as simple learning rules. Our final result provides further motivation for this line of work.

Corollary 3.8 (Almost everywhere descent)**.** *If the regularizer $\mathcal{R}(\mathbf{W})$ is given by eq. (51), then the weight-balancing flow in eq. (40) decreases the regularized loss in every open orthant of weight space.*

Proof. Consider the regularizer in eq. (51) as a function of the variable $\omega_{ij} = \log|W_{ij}|$. To prove the result, we show that the Hessian $\mathbf{H}(\boldsymbol{\omega}) = \frac{\partial^2 \mathcal{R}}{\partial \boldsymbol{\omega}\partial \boldsymbol{\omega}^\top}$ in Theorem 3.4 is positive definite whenever no weight W_{ij} is equal to zero. The proof requires two steps—first, to compute the Hessian, and second, to show that it is positive definite. For the first step, we begin by evaluating the time-derivative in the flow of eq. (40). Here

we find, by repeated differentiation, that

$$\frac{d}{dt}\left[W_{ij}\frac{\partial \mathcal{R}}{\partial W_{ij}}\right] = \frac{d}{dt}\Big[\pi_{ij}(\mathbf{W})\rho_i(\mathbf{W})\Big], \tag{60}$$

$$= \left[\sum_k \frac{\partial \pi_{ij}}{\partial W_{ik}}\dot{W}_{ik}\right]\rho_i(\mathbf{W}) \;+\; \pi_{ij}(\mathbf{W})\left[\sum_k \frac{\partial \rho_i}{\partial W_{ik}}\dot{W}_{ik}\right], \tag{61}$$

$$= p\pi_{ij}(\mathbf{W})\sum_k \Big[\delta_{jk} - \pi_{ik}(\mathbf{W})\Big]\frac{\dot{W}_{ik}}{W_{ik}}\,\rho_i(\mathbf{W}) \;+\; q\pi_{ij}(\mathbf{W})\sum_k \Big[\pi_{ik}(\mathbf{W})\rho_i(\mathbf{W})\Big]\frac{\dot{W}_{ik}}{W_{ik}} \tag{62}$$

$$= \pi_{ij}(\mathbf{W})\rho_i(\mathbf{W})\Big[p\,\dot{\omega}_{ij} \;+\; (q-p)\sum_k \pi_{ik}(\mathbf{W})\,\dot{\omega}_{ik}\Big]. \tag{63}$$

We can now read off the nonzero elements of $\mathbf{H}(\boldsymbol{\omega})$ from the relation between eqs. (47) and (63). Let $\mathbf{U}$ be any nonzero matrix of the same size as $\mathbf{W}$, and let $\mathbf{u}$ be the flattened representation of $\mathbf{U}$ as a vector. Then from eqs. (47) and (63) we have

$$\mathbf{u}^\top\mathbf{H}(\boldsymbol{\omega})\,\mathbf{u} = \; p\sum_{ij}\pi_{ij}(\mathbf{W})\rho_i(\mathbf{W})\,U_{ij}^2 \;+\; (q-p)\sum_{ijk}\pi_{ij}(\mathbf{W})\rho_i(\mathbf{W})\pi_{ik}(\mathbf{W})\,U_{ij}U_{ik}, \tag{64}$$

$$= \; p\sum_{ij}\rho_i(\mathbf{W})\pi_{ij}(\mathbf{W})\left(U_{ij} - \sum_k \pi_{ik}(\mathbf{W})\,U_{ik}\right)^2 + \; q\sum_i \rho_i(\mathbf{W})\left(\sum_j \pi_{ij}(\mathbf{W})\,U_{ij}\right)^2, \tag{65}$$

$$> 0 \quad \text{for all} \quad \mathbf{v} \neq \mathbf{0}. \tag{66}$$

The strict inequality in the last line is justified by the following observations: (i) both $\pi_{ij}(\mathbf{W})$ and $\rho_i(\mathbf{W})$ are strictly positive when no weights are exactly equal to zero; (ii) the left term in eq. (65) only vanishes when $\mathbf{U}$ has constant rows; (iii) the right term only vanishes when for all rows $\sum_j \pi_{ij}(\mathbf{W})U_{ij} = 0$. Since $\mathbf{U}$ is nonzero, however, it is not possible to satisfy both of these conditions simultaneously, so that one term and/or the other must be strictly positive. Finally, since $\mathbf{u}^\top\mathbf{H}(\boldsymbol{\omega})\mathbf{u} > 0$ for any nonzero $\mathbf{u}$, we conclude that $\mathbf{H}(\boldsymbol{\omega})$ is positive definite in every open orthant of the weight space. The result then follows from Theorem 3.4. □

3.4 Other related work

Many previous studies have investigated the properties of feedforward networks with rescaling symmetries. Our own study was strongly motivated by several of these earlier works. Neyshabur et al. (2015a) showed that stochastic gradient descent (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.

Closest in approach to our study is the work on ENorm (Stock et al., 2019). As mentioned previously, ENorm is an elegant procedure, also based on multiplicative updates, to minimize the ℓ_p-norm of the weights in feedforward networks with rescaling symmetries. Stock et al. (2019) trained networks by interweaving updates from SGD and ENorm with ℓ_2-norm regularization; interestingly, the networks in these experiments generalized better on test data. This improvement suggests to interweave the more general updates in section 2 of this paper (minimizing the $\ell_{p,q}$-norm of the weights) with learning rules based on the generalized flows in section 3 of this paper (which balance the weights with respect to the $\ell_{p,q}$-norm). Our study lays the foundation for this further exploration.

Even more recent work has suggested that learning can be accelerated by 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. Our approach differs from work on neural teleportation 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.

4 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; Tanaka & Kunin, 2021; Gluch & Urbanke, 2021; Armenta & Jodoin, 2021). 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 of a feedforward network without changing the function that it computes. Specifically, we derived closed-form multiplicative updates that minimize the $\ell_{p,q}$-norm of the weight matrix over the equivalence class of networks that are related by rescaling transformations. We have also derived weight-balancing flows that preserve the minimality of *any* (differentiable) regularizer over the course of learning.

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). Another question is how gauge-fixing may relates to (or may be incorporated with) schemes such as batch normalization (Ioffe & Szegedy, 2015), weight normalization (Salimans & Kingma, 2016), and layer normalization (Ba et al., 2016). It will require a mix of empirical and theoretical investigation to understand the interplay of these methods (van Laarhoven, 2017). In this work, we have not relied heavily on the machinery of Bregman Lagrangians (Wibisono et al., 2016; Tanaka & Kunin, 2021), but it is clear that they provide a powerful framework for further progress.

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 $\ell_{p,q}$-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 the weight-balancing flows they yield from Theorem 3.3. 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, 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.

J. Ba, K. Lei, J. Ryan, and G. E. Hinton. Layer normalization, 2016. arxiv:1607.06450.

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.

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.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

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

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.

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*, pp. 44–51, 2011.

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

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 448–456, 2015.

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, 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.

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*, 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.

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

T. Salimans and D. P. Kingma. Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In D. D Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 901–909. Curran Associates, Inc., 2016.

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.

P. Stock, B. Graham, R. Gribonval, and H. Jégou. Equi-normalization of neural networks. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

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.

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.

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.