Neural Synaptic Balance

Anonymous Author(s)
Affiliation
Address
email

Abstract

For a given additive cost function R (regularizer), a neuron is said to be in balance if the total cost of its input weights is equal to the total cost of its output weights. The basic example is provided by feedforward layered networks of ReLU units trained with L_2 regularizers, which exhibit balance after proper training. We develop a general theory that extends this phenomenon in three broad directions in terms of: (1) activation functions; (2) regularizers, including all L_p ($p > 0$) regularizers; and (3) architectures (non-layered, recurrent, convolutional, mixed activations). Gradient descent on the error function alone does not converge in general to a balanced state where every neuron is in balance, even when starting from a balanced state. However, gradient descent on the regularized error function must converge to a balanced state, and thus network balance can be used to assess learning progress. The theory is based on two local neuronal operations: scaling which is commutative, and balancing which is not commutative. Finally, and most importantly, given any initial set of weights, when local balancing operations are applied to each neuron in a stochastic manner, global order always emerges through the convergence of the stochastic algorithm to the same unique set of balanced weights. The reason for this convergence is the existence of an underlying strictly convex optimization problem where the relevant variables are constrained to a linear, only architecture-dependent, manifold. The theory is corroborated through simulations carried out on benchmark data sets. Balancing operations are entirely local and thus physically plausible in biological and neuromorphic networks.

1 Introduction

When large neural networks are trained on complex tasks, they produce large arrays of synaptic weights that have no clear structure and are difficult to interpret. Thus finding any kind of structure in the weights of large neural networks is of great interest. Here we study a particular kind of structure we call neural synaptic balance and the conditions under which it emerges. Neural synaptic balance is different from the biological notion of balance between excitation and inhibition [Froemke, 2015, Field et al., 2020, Howes and Shatalina, 2022, Kim and Lee, 2022, Shirani and Choi, 2023]. We use this term to refer to any systematic relationship between the input and output synaptic weights of individual neurons or layers of neurons. Here we consider the case where the cost of the input weights is equal to the cost of the output weights, where the cost is defined by some regularizer. One of the most basic examples of such a relationship is when the sum of the squares of the input weights of a neuron is equal to the sum of the squares of its output weights.

Basic Example: The basic example where this happens is with a neuron with a ReLU activation function inside a network trained to minimize an error function with L_2 regularization. If we multiply the incoming weights of the neuron by some $\lambda > 0$ (including the bias) and divide the outgoing weights of the neuron by the same λ, it is easy to see that this scaling operation does not affect in any way the contribution of the neuron to the rest of the network. Thus, the component of the overall

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

error function that depends only on the input-output function of the network is unchanged. However, the value of the L_2 regularizer changes with λ and we can ask what is the value of λ that minimizes the corresponding contribution given by:

$$\sum_{i \in IN} (\lambda w_i)^2 + \sum_{i \in OUT} (w_i/\lambda)^2 = \lambda^2 A + \frac{1}{\lambda^2} B \tag{1.1}$$

where IN and OUT denote the set of incoming and outgoing weights respectively, $A = \sum_{i \in IN} w_i^2$, and $B = \sum_{i \in OUT} w_i^2$. The product of the two terms on the right-hand side of Equation 1.1 is equal to AB and does not depend on λ. Thus, the minimum is achieved when these two terms are equal, which yields: $(\lambda^*)^4 = B/A$ for the optimal λ^*. The corresponding new set of weights, $v_i = \lambda^* w_i$ for the input weights and $v_i = w_i/\lambda^*$ for the outgoing weights, must be balanced: $\sum_{i \in IN} v_i^2 = \sum_{i \in OUT} v_i^2$. This is because its optimal scaling factor can only be $\lambda^* = 1$. Thus, we can define two operations that can be applied to the incoming and outgoing weights of a neuron: scaling and balancing. It is easy to check that scaling operations applied to any two neurons commute, whereas balancing operations do not commute if the two neurons are directly connected (Appendix). If a network of ReLU neurons is properly trained using a standard error function with an L_2 regularizer, at the end of training one observes a remarkable phenomenon: for each ReLU neuron, the norm of the incoming synaptic weights is approximately equal to the norm of the outgoing synaptic weights, i.e. every neuron is balanced.

There have been isolated previous studies of this kind of synaptic balance [Du et al., 2018, Stock et al., 2022] under special conditions. For instance, in Du et al. [2018], it is shown that if a deep network is initialized in a balanced state with respect to the sum of squares metric, and if training progresses with an infinitesimal learning rate, then balance is preserved throughout training. Here, we take a different approach aimed at uncovering the generality of neuronal balance phenomena, the learning conditions under which they occur, as well as new local balancing algorithms and their convergence properties. We study neural synaptic balance in its generality in terms of activation functions, regularizers, network architectures, and training stages. In particular, we systematically answer questions such as: Why does balance occur? Does it occur only with ReLU neurons? Does it occur only with L_2 regularizers? Does it occur only in fully connected feedforward architectures? Does it occur only at the end of training? And what happens if we balance neurons at random in a large network?

2 Generalization of the Activation Functions

What enables scaling ReLU neurons without changing their input-output function is the homogeneous property of ReLU activation function. An activation function f is said to be *homogeneous* if for every $\lambda > 0$, $f(\lambda x) = \lambda f(x)$. To fully characterize the class of homogeneous activation functions, we first define a new class of activation functions, corresponding to bilinear units (BiLU), consisting of two half-lines meeting at the origin.

Definition 2.1. *(BiLU) A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is bilinear (BiLU) if and only if $f(x) = ax$ when $x < 0$, and $f(x) = bx$ when $x \geq 0$, for some fixed parameters a and b in $\mathbb{R}$.*

BiLU units include linear units ($a = b$), ReLU units ($a = 0, b = 1$), leaky ReLU ($a = \epsilon; b = 1$) units, and symmetric linear units ($a = -b$), all of which can also be viewed as special cases of piece-wise linear units [Tavakoli et al., 2021], with a single hinge. One advantage of ReLU and more generally BiLU neurons, which is very important during backpropagation learning, is that their derivative is very simple and can only take one of two values (a or b). We have the following equivalence.

Proposition 2.2. *A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is homogeneous if and only if it is a BiLU activation function.*

Proof. Every function in BiLU is clearly homogeneous. Conversely, any homogeneous function f must satisfy: (1) $f(0x) = 0f(x) = f(0) = 0$; (2)$f(x) = f(1x) = f(1)x$ for any positive x; and (3) $f(x) = f(-u) = f(-1)u = -f(-1)x$ for any negative x. Thus f is in BiLU with $a = -f(-1)$ and $b = f(1)$. □

In the Appendix, we provide a simple proof that networks of BiLU neurons, even with a single hidden layer, have universal approximation properties.

While in the rest of this work we use BiLU neurons, it is possible to generalize the notions of scaling and balancing even further. To see this, suppose that there is a neuron with an activation function $f : \mathbb{R} \to R$, and functions $g : (a, b) \to \mathbb{R}$ and $h : (a, b) \to \mathbb{R}$, such that: $f(g(\lambda)x) = h(\lambda)f(x)$, for any $\lambda \in (a, b)$. Then if we multiply the incoming weights by $g(\lambda)$ and divide the outgoing weights by $h(\lambda) \neq 0$ (generalized scaling), we see again that the influence of the neuron on the rest of the network is unchanged. And thus, again, we can try to find the value of λ that minimizes the regularization cost (generalized balancing). Here we provide an example of such an activation function, with $g(\lambda) = \lambda$ and $h(\lambda) = \lambda^c$. Additional details are given in the Appendix.

Proposition 2.3. *The set of activation functions f satisfying $f(\lambda x) = \lambda^c f(x)$ for any $x \in \mathbb{R}$ and any $\lambda > 0$ consist of the functions of the form:*

$$f(x) = \begin{cases} Cx^c & \text{if} \quad x \geq 0 \\ Dx^c & \text{if} \quad x < 0. \end{cases} \tag{2.1}$$

where $c \in \mathbb{R}$, $C = f(1) \in R$, and $D = f(-1) \in \mathbb{R}$. We call these bi-power units (BiPU). If, in addition, we want f to be continuous at 0, we must have either $c > 0$, or $c = 0$ with $C = D$.

Note that in the general case where $c > 0$, C and D do not need to be equal. In particular, one of them can be equal to zero, and the other one can be different from zero giving rise to rectified power units.

3 Generalization of the Regularizers

As we have seen, given a BiLU neuron, scaling its input and output weights by λ and $1/\lambda$ respectively does not alter its contribution to the rest of the network and thus we can adjust λ to reduce or even minimize the contribution of the corresponding weights to the regularizer. It is reasonable to assume that the regularizer has the general additive form: $R(W) = \sum_w g_w(w)$ where W denotes all the weights in the network. Without much loss of generality, we can assume that the g_w are continuous, and lower-bounded by 0. To ensure the existence and uniqueness of a minimum during the balancing of any neuron, We will assume that each function g_w depends only on the magnitude $|w|$ of the corresponding weight, and that g_w monotonically increases from 0 to $+\infty$. Clearly, L_2, L_1 and more generally all L_p regularizers are special cases where, for $p > 0$, L^p regularization is defined by: $R(W) = \sum_w |w|^p$. Differentiability conditions can be added to be able to derive closed form solutions for the balance (optimal scaling). This is satisfied by all forms of L_p regularization, for $p > 0$. We have the following theorem.

Theorem 3.1. *(Balance and Regularizer Minimization) Assume an additive regularizer with the properties described above, where in addition we assume that the functions g_w are continuously differentiable, except perhaps at the origin. Then, for any neuron, there exists one optimal value λ^* that minimizes $R(W)$. This value must be a solution of the consistency equation:*

$$\lambda^2 \sum_{w \in IN(i)} w g'_w(\lambda w) = \sum_{w \in OUT(i)} w g'_w(w/\lambda) \tag{3.1}$$

Once the weights are rebalanced accordingly, the new weights must satisfy the generalized balance equation:

$$\sum_{w \in IN(i)} w g'(w) = \sum_{w \in OUT(i)} w g'(w) \tag{3.2}$$

In particular, if $g_w(w) = |w|^p$ for all the incoming and outgoing weights of neuron i, then the optimal value λ^* is unique and equal to:

$$\lambda^* = \Big(\frac{\sum_{w \in OUT(i)} |w|^p}{\sum_{w \in IN(i)} |w|^p} \Big)^{1/2p} = \Big(\frac{||OUT(i)||_p}{||IN(i)||_p} \Big)^{1/2} \tag{3.3}$$

After balancing, the decrease $\Delta R \geq 0$ in the value of the L_p regularizer $R = \sum_w |w|^p$ is given by:

$$\Delta R = \left(\Big(\sum_{w \in IN(i)} |w|^p \Big)^{1/2} - \Big(\sum_{w \in OUT(i)} |w|^p \Big)^{1/2} \right)^2 \tag{3.4}$$

After balancing neuron i, its new weights satisfy the generalized L_p balance equation:

$$\sum_{w \in IN(i)} |w|^p = \sum_{w \in OUT(i)} |w|^p \tag{3.5}$$

Proof. The results are obtained by setting the derivative of the regularizer with respect to the scaling factor λ to 0. Note that the theorem applies to regularizers combining different L_p's (e.g. of the form $\$alphaL_2 + \beta L_1$). The details are given in the Appendix. □

4 Generalization of the Architectures

It is straightforward to check that the scaling and balancing operations can be extended in the following cases (see Appendix for additional details):

1. Mixed networks containing both BiLU and non-BiLU units. One can just restrict those operations to the BiLU neurons.
2. Recurrent networks containing BiLU neurons, not just feedforward networks.
3. Networks that are not layered, or not fully connected.
4. In addition, scaling and balancing operations can be applied layer-wise to an entire layer of BiLU neurons in a tied manner, by using the same scaling factor λ with a single optimal value λ^* for all the neurons in the layer. In particular, this allows the application of scaling and balancing to convolutional layers of BiLU neurons.

5 Balancing Algorithms

Gradient Descent: When a network of BiLU neurons is trained by gradient descent to minimize an error function $E(W)$, such as the negative log-likelihood of the data, there is no reason for the final weights to be balanced. However, when a network is properly trained to minimize a regularized error function $\mathcal{E} = E(W) + R(W)$, the final weights ought to be balanced. The reason is that if a neuron is not in a balanced state at the end of training, then we can further reduce its contribution to R smoothly by balancing it. This implies that the gradient of $\mathcal{E}(W)$ is not equal to zero at the end of training, and thus training has not properly converged. The converse is that the degree of balance can be used as a proxy for assessing whether learning has converged or not.

Stochastic Balancing: More interestingly, we now investigate what happens if we fix the weights W of a network and iteratively balance its BiLU neurons.

Theorem 5.1. *(Convergence of Stochastic Balancing) Consider a network of BiLU neurons with an error function $\mathcal{E}(W) = E(W) + R(W)$ where R is any L_p ($p > 0$) regularizer. Let W denote the initial weights. When the neuronal stochastic balancing algorithm is applied throughout the network so that every neuron is visited from time to time, then $E(W)$ remains unchanged but $R(W)$ must converge to some finite value that is less or equal to the initial value, strictly less if the initial weights are not balanced. In addition, for every neuron i, $\lambda_i^*(t) \to 1$ and the weights themselves must converge to a limit W^* which is globally balanced, with $E(W) = E(W^*)$ and $R(W) \geq R(W^*)$, and with equality if only if W is already balanced. Finally, W^* is unique as it corresponds to the solution of a strictly convex optimization problem with special linear constraints that depend only on the network architecture (and not on W). Stochastic balancing projects to stochastic trajectories in the linear manifold that run from the origin to the unique optimal configuration.*

Proof. Each individual balancing operation leaves $E(W)$ unchanged because the BiLU neurons are homogeneous. Furthermore, each balancing operation reduces the regularization error $R(W)$, or leaves it unchanged. Since the regularizer is lower-bounded by zero, the value of the regularizer must approach a limit as the stochastic updates are being applied. However, this alone does not imply

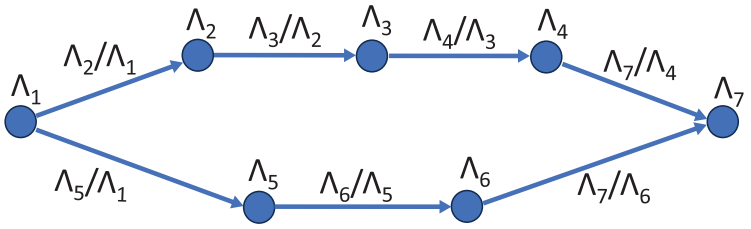

Figure 1: Two hidden units (1 and 7) connected by two different directed paths 1-2-3-4-7 and 1-5-6-7 in a BiLU network. Each unit i has a scaling factor Λ_i, and each directed edge from unit j to unit i has a scaling factor $M_{ij} = \Lambda_i/\Lambda_j$. The products of the M_{ij}'s along each path is equal to: $\frac{\Lambda_2}{\Lambda_1}\frac{\Lambda_3}{\Lambda_2}\frac{\Lambda_4}{\Lambda_3}\frac{\Lambda_7}{\Lambda_4} = \frac{\Lambda_5}{\Lambda_1}\frac{\Lambda_6}{\Lambda_5}\frac{\Lambda_7}{\Lambda_6} = \frac{\Lambda_7}{\Lambda_1}$. Therefore the variables $L_{ij} = \log M_{ij}$ must satisfy the linear equation: $L_{21} + L_{32} + L_{43} + L_{74} = L_{51} + L_{65} + L_{76}$ =$\log \Lambda_7 - \log \Lambda_1$.

that the weights are converging and whether the limit is unique or not. To address these issues, for simplicity, we use a continuous time notation. After a certain time t each neuron has been balanced a certain number of times. While the balancing operations are not commutative as balancing operations, they are commutative as scaling operations. Thus we can reorder the scaling operations and group them neuron by neuron so that, for instance, neuron i has been scaled by the sequence of scaling operations of the form:

$$S_{\lambda_1^*}(i)S_{\lambda_2^*}(i)\dots S_{\lambda^*_{n_{it}}}(i) = S_{\Lambda_i(t)}(i) \tag{5.1}$$

where n_{it} corresponds to the count of the last update of neuron i prior to time t, and:

$$\Lambda_i(t) = \prod_{1\leq n\leq n_{it}} \lambda_n^*(i) \tag{5.2}$$

For the input and output units, we can consider that their balancing coefficients λ^* are always equal to 1 (at all times) and therefore $\Lambda_i(t) = 1$ for any visible unit i. At time t the weight connecting unit j to unit i is given by: $w_{ij}(t) = w_{ij}(0)\Lambda_i(t)/\Lambda_j(t)$, where $w_{ij}(0)$ corresponds to the initial value. In the Appendix, we show upfront that for all BiLU units i, $\Lambda_i(t)$ converges to some limit $\Lambda_i > 0$, and thus the weights converge too. Here, we first suppose that the coefficients $\Lambda_i(t)$ converge to some limit Λ_i, and recover the convergence at the end from understanding the overall proof. As a result, for any L_p regularizer, the coefficients Λ_i corresponding to a globally balanced state must be solutions of the following optimization problem:

$$\min_{\Lambda} R(\Lambda) = \sum_{ij} |\frac{\Lambda_i}{\Lambda_j} w_{ij}|^p \tag{5.3}$$

under the simple constraints: $\Lambda_i > 0$ for all the BiLU hidden units, and $\Lambda_i = 1$ for all the visible (input and output) units. In this form, the problem is not convex. Introducing new variables $M_j = 1/\Lambda_j$ is not sufficient to render the problem convex. Using variables $M_{ij} = \Lambda_i/\Lambda_j$ is better, but still problematic for $0 < p \leq 1$. However, let us instead introduce the new variables $L_{ij} = \log(\Lambda_i/\Lambda_j)$. These are well defined since we know that $\Lambda_i/\Lambda_j > 0$. The objective now becomes:

$$\min R(L) = \sum_{ij} |e^{L_{ij}} w_{ij}|^p = \sum_{ij} e^{pL_{ij}} |w_{ij}|^p \tag{5.4}$$

This objective is strictly convex in the variables L_{ij}, as a sum of strictly convex functions (exponentials). However, to show that it is a convex optimization problem we need to study the constraints on the variables L_{ij}. In particular, from the set of Λ_i's it is easy to construct a unique set of L_{ij}. However what about the converse?

Definition 5.2. *A set of real numbers L_{ij}, one per connection of a given neural architecture, is self-consistent if and only if there is a unique corresponding set of numbers $\Lambda_i > 0$ (one per unit) such that: $\Lambda_i = 1$ for all visible units and $L_{ij} = \log \Lambda_i/\Lambda_j$ for every directed connection from a unit j to a unit i.*

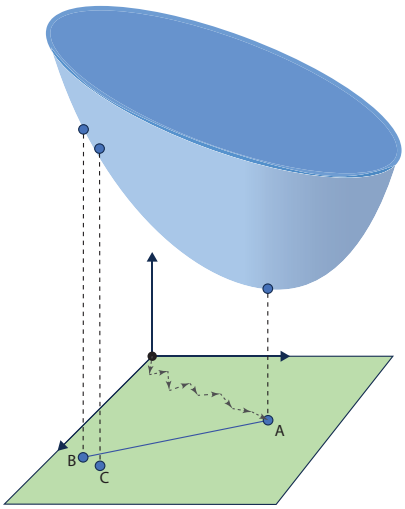

Figure 2: The problem of minimizing the strictly convex regularizer $R(L_{ij}) = \sum_{ij} e^{pL_{ij}} |w_{ij}|^p$ ($p > 0$), over the linear (hence convex) manifold of self-consistent configurations defined by the linear constraints of the form $\sum_\pi L_{ij} = 0$, where π runs over input-output paths. The regularizer function depends on the weights. The linear manifold depends only on the architecture, i.e., the graph of connections. This is a strictly convex optimization problem with a unique solution associated with the point A. At A the corresponding weights must be balanced, or else a self-consistent configuration of lower cost could be found by balancing any non-balanced neuron. Finally, any other self-consistent configuration B cannot correspond to a balanced state of the network, since there must exist balancing moves that further reduce the regularizer cost (see main text). Stochastic balancing produces random paths from the origin, where $L_{ij=} \log M_{ij} = 0$, to the unique optimum point A.

Remark 5.3. This definition depends on the graph of connections, but not on the original values of the synaptic weights. Every balanced state is associated with a self-consistent set of L_{ij}, but not every self-consistent set of L_{ij} is associated with a balanced state.

Proposition 5.4. *A set L_{ij} associated with a neural architecture is self-consistent if and only if $\sum_\pi L_{ij} = 0$ where π is any directed path connecting an input unit to an output unit or any directed cycle (for recurrent networks).*

Proof. If we look at any directed path π from unit i to unit j, it is easy to see that we must have:

$$\sum_\pi L_{kl} = \log \Lambda_i - \log \Lambda_j \tag{5.5}$$

This is illustrated in Figure 1. Thus along any directed path that connects any input unit to any output unit, we must have $\sum_\pi L_{ij} = 0$. In addition, for recurrent neural networks, if π is a directed cycle we must also have: $\sum_\pi L_{ij} = 0$. Thus in short we only need to add linear constraints of the form: $\sum_\pi L_{ij} = 0$. Any unit is situated on a path from an input unit to an output unit. Along that path, it is easy to assign a value Λ_i to each unit by simple propagation starting from the input unit which has a multiplier equal to 1. When the propagation terminates in the output unit, it terminates consistently because the output unit has a multiplier equal to 1 and, by assumption, the sum of the multipliers along the path must be zero. So we can derive scaling values Λ_i from the variables L_{ij}. Finally, it is easy to show that there are no clashes, i.e. that it is not possible for two different propagation paths to assign different multiplier values to the same unit i (see Appendix). □

Remark 5.5. Thus the constraints associated with being a self-consistent configuration of L_{ij}' s are all linear. This linear manifold of constraints depends only on the architecture, i.e., the graph of connections. The strictly convex function $R(L_{ij})$ depends on the actual weights W. Different sets of weights W produce different convex functions over the same linear manifold.

Remark 5.6. One could coalesce all the input units and all output units into a single unit, in which case a path from an input unit to and output unit becomes also a directed cycle. In this representation, the constraints are that the sum of the L_{ij} must be zero along any directed cycle. In general, it is not necessary to write a constraint for every path from input units to output units. It is sufficient to select a representative set of paths such that every unit appears in at least one path.

We can now complete the proof of Theorem 5.1. Given a neural network of BiLUs with a set of weights W, we can consider the problem of minimizing the regularizer $R(L_{ij})$ over the self-admissible configuration L_{ij}. For any $p > 0$, the L_p regularizer is strictly convex and the space of self-admissible configurations is linear and hence convex. Thus this is a strictly convex optimization

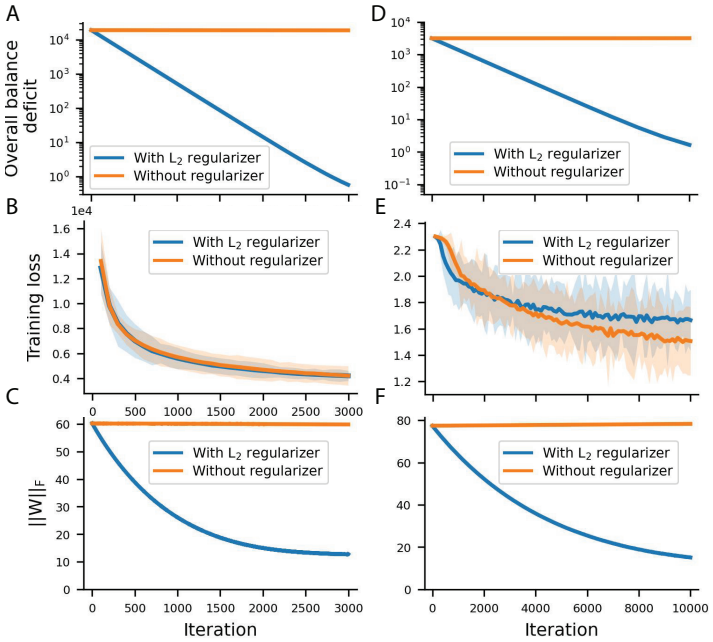

Figure 3: **SGD applied to E alone, in general, does not converge to a balanced state, but SGD applied to $E + R$ converges to a balanced state.** **(A-C)** Simulations use a deep fully connected autoencoder trained on the MNIST dataset. **(D-F)** Simulations use a deep locally connected network trained on the CFAR10 dataset. **(A,D)** Regularization leads to neural balance. **(B,E)** The training loss decreases and converges during training (these panels are not meant for assessing the quality of learning when using a regularizer). **(C,F)** Using weight regularization decreases the norm of weights. **(A-F)** Shaded areas correspond to one s.t.d around the mean (in some cases the s.t.d. is small and the shaded area is not visible).

problem that has a unique solution (Figure 2). Note that the minimization is carried over self-consistent configurations, which in general are not associated with balanced states. However, the configuration of the weights associated with the optimum set of L_{ij} (point A in Figure 2) must be balanced. To see this, imagine that one of the BiLU units–unit i in the network is not balanced. Then we can balance it using a multiplier λ_i^* and replace Λ_i by $\Lambda_i' = \Lambda_i \lambda^*$. It is easy to check that the new configuration including Λ_i' is self-consistent. Thus, by balancing unit i, we are able to reach a new self-consistent configuration with a lower value of R which contradicts the fact that we are at the global minimum of the strictly convex optimization problem.

We know that the stochastic balancing algorithm always converges to a balanced state. We need to show that it cannot converge to any other balanced state, and in fact that the global optimum is the only balanced state. By contradiction, suppose it converges to a different balanced state associated with the coordinates (L_{ij}^B) (point B in Figure 2). Because of the self-consistency, this point is also associated with a unique set of (Λ_i^B) coordinates. The cost function is continuous and differentiable in both the L_{ij}'s and the Λ_i's coordinates. If we look at the negative gradient of the regularizer, it is non-zero and therefore it must have at least one non-zero component $\partial R/\partial \Lambda_i$ along one of the Λ_i coordinates. This implies that by scaling the corresponding unit i in the network, the regularizer can be further reduced, and by balancing unit i the balancing algorithm will reach a new point (C in Figure 2) with lower regularizer cost. This contradicts the assumption that B was associated with a balanced stated. Thus, given an initial set of weights W, the stochastic balancing algorithm must always converge to the same and unique optimal balanced state W^* associated with the self-consistent point A. A particular stochastic schedule corresponds to a random path within the linear manifold from the origin (at time zero, all the multipliers are equal to 1, and therefore $M_{ij} = 1$ and $L_{ij} = 0$ for any i and any j) to the unique optimum point A.

□

Remark 5.7. From the proof, it is clear that the same result holds also for any deterministic balancing schedule, as well as for tied and non-tied subset balancing, e.g., for layer-wise balancing and tied layer-wise balancing. In the Appendix, we provide an analytical solution for the case of tied layer-wise balancing in a layered feed-forward network.

Remark 5.8. From the proof, it is also clear that the same convergence to the unique global optimum is observed if each neuron, when stochastically visited, is favorably scaled rather than balanced, i.e., it is scaled with a factor that reduces R but not necessarily minimizes R. Stochastic balancing can also be viewed as a form of EM algorithm where the E and M steps can be taken fully or partially.

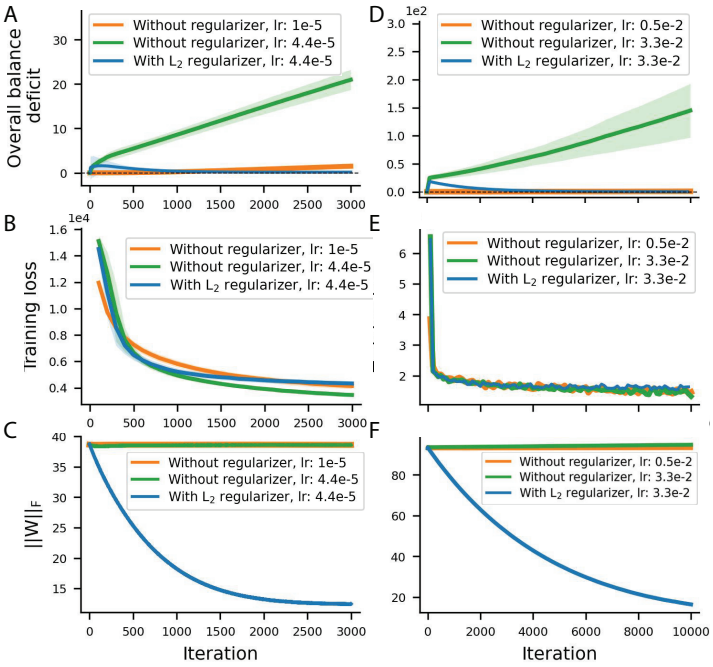

Figure 4: **Even if the starting state is balanced, SGD does not preserve the balance unless the learning rate is infinitely small.** **(A-C)** Simulations use a deep fully connected autoencoder trained on the MNIST dataset. **(D-F)** Simulations use a deep locally connected network trained on the CFAR10 dataset. **(A-F)** The initial weights are balanced using the stochastic balancing algorithm. Then the network is trained by SGD. **(A,D)** When the learning rate (lr) is relatively large, without regularization, the initial balance of the network is rapidly disrupted. **(B,E)** The training loss decreases and converges during training (these panels are not meant for assessing the quality of learning when using a regularizer). **(C,F)** Using weight regularization decreases the norm of the weights. **(A-F)** Shaded areas correspond to one s.t.d around the mean (in some cases the s.t.d. is small and the shaded area is not visible).

6 Simulations

To further corroborate the results, we ran multiple experiments. Here we report the results from two series of experiments. The first one is conducted using a six-layer, fully connected, autoencoder trained on MNIST [Deng, 2012] for a reconstruction task with ReLU activation functions in all layers and the sum of squares errors loss function. The number of neurons in consecutive layers, from input to output, is 784, 200, 100, 50, 100, 200, 784. Stochastic gradient descent (SGD) learning by backpropagation is used for learning with a batch size of 200.

The second one is conducted using three locally connected layers followed by three fully connected layers trained on CFAR10 [Krizhevsky and Hinton, 2009] for a classification task with leaky ReLU activation functions in the hidden layers, a softmax output layer, and the cross entropy loss function. The number of neurons in consecutive layers, from input to output, is 3072, 5000, 2592, 1296, 300, 100, 10. Stochastic gradient descent (SGD) learning by backpropagation is used for learning with a batch size of 5.

In all the simulation figures (Figures 3, 4, and 5) the left column presents results obtained from the first experiment, while the right column presents results obtained from the second experiment. While we used both L_1 and L_2 regularizers in the experiments, in the figures we report the results obtained with the L_2 regularizer, which is the most widely used regularizer. In Figures 3 and 4, training is done using batch gradient descent on the MNIST and CIFAR data. The balance deficit for a single neuron i is defined as: $\left(\sum_{w \in IN(i)} w^2 - \sum_{w \in OUT(i)} w^2\right)^2$, and the overall balance deficit is defined as the sum of these single-neuron balance deficits across all the hidden neurons in the network. The overall deficit is zero if and only if each neuron is in balance. In all the figures, $||W||_F$ denotes the Frobenius norm of the weights.

Figure 3 shows that learning by gradient descent with a L_2 regularizer results in a balanced state. Figure 4 shows that even when the network is initialized in a balanced state, without the regularizer the network can become unbalanced if the fixed learning rate is not very small. Figure 5 shows that the local stochastic balancing algorithm, by which neurons are randomly balanced in an asynchronous fashion, always converges to the same (unique) global balanced state.

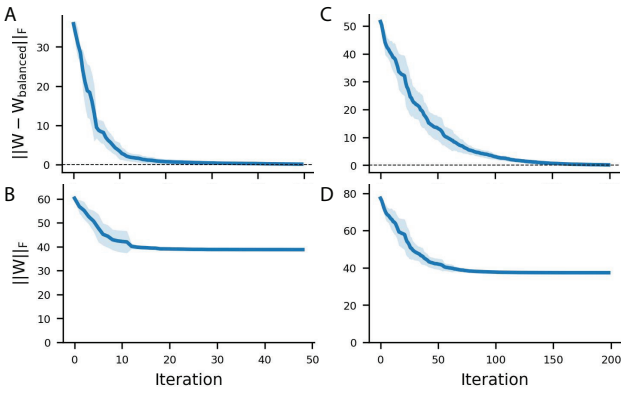

Figure 5: **Stochastic balancing converges to a unique global balanced state** **(A-B)** Simulations use a deep fully connected autoencoder trained on the MNIST dataset. **(C-D)** Simulations use a deep locally connected network trained on the CFAR10 dataset. **(A,C)** The weights of the network are initialized randomly and saved. The stochastic balancing algorithm is applied and the resulting balanced weights are denoted by $W_{balanced}$. The stochastic balancing algorithm is applied 1,000 different times. In all repetitions, the weights converge to the same value $W_{balanced}$. **(B,D)** Stochastic balancing decreases the norm of the weights. **(A-D)** Shaded areas correspond to one standard deviation around the mean.

7 Conclusion

While the theory of neural synaptic balance is a mathematical theory that stands on its own, it is worth considering some of its possible consequences and applications, at the theoretical, algorithmic, biological, and neuromorphic hardware levels. At the theory level, for instance, it suggests extending theorems obtained with ReLU neurons to BiLU neurons, using balance ideas to study learning in *linear* regularized networks, and using the manifolds of equivalent weights to study issues of overparameterization (e.g. the data needs only to specify the balanced state, not the entire equivalence class). At the algorithmic level, balancing algorithms could be used for instance to balance networks at any stage of learning, including at the beginning, and as an alternative way to regularize networks. Finally, because scaling and balancing are local operations, they are potentially of interest in physical, as opposed to digitally-simulated, neural networks. In particular, it would be interesting to know if some notion of balance applies to biological neurons. Unfortunately, current recording technologies do not allow the measurement of all incoming and outgoing synapses of a neuron. Perhaps some approximation could be obtained statistically and at the population level, or perhaps approximate measurements could be carried in very simple networks (e.g. *C. elegans*)or using neurons in culture. Finally, in neuromorphic hardware, the balance could be relevant for training spiking neural networks with low energy consumption [Sorbaro et al., 2020, Rueckauer et al., 2017]). In particular, ReLU scaling can influence the number of spikes generated in each layer and the average energy consumption at each layer. Similarly, in memristor networks [Ivanov et al., 2022, Liang and Wong, 2000]), L_2 minimization is directly connected to power consumption. Moreover, the issue of the limited conductivity range of memristors is mentioned in Ivanov et al. [2022] and in Ji et al. [2016] Therefore, a local algorithm to reduce the norm of the weights could help mitigate this issue as well.

The theory of neural synaptic balance explains some basic findings regarding L_2 balance in feedforward networks of ReLU neurons and extends them in several directions. The first direction is the extension to BiLU and other activation functions (BiPU). The second direction is the extension to more general regularizers, including all L_p ($p > 0$) regularizers. The third direction is the extension to non-layered architectures, recurrent architectures, convolutional architectures, as well as architectures with mixed activation functions. The theory is based on two local neuronal operations: scaling which is commutative, and balancing which is not commutative. Finally, and most importantly, given any initial set of weights, when local balancing operations are applied in a stochastic or deterministic manner, global order always emerges through the convergence of the balancing algorithm to the same unique set of balanced weights. The reason for this convergence is the existence of an underlying convex optimization problem where the relevant variables are constrained to a linear, only architecture-dependent, manifold. Scaling and balancing operations are local and thus may have applications in physical, non-digitally simulated, neural networks where the emergence of global order from local operations may lead to better operating characteristics and lower energy consumption.

References

P. Baldi. *Deep Learning in Science*. Cambridge University Press, Cambridge, UK, 2021.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 31, 2018.

Rachel E Field, James A D'amour, Robin Tremblay, Christoph Miehl, Bernardo Rudy, Julijana Gjorgjieva, and Robert C Froemke. Heterosynaptic plasticity determines the set point for cortical excitatory-inhibitory balance. *Neuron*, 106(5):842–854, 2020.

Robert C Froemke. Plasticity of cortical excitatory-inhibitory balance. *Annual review of neuroscience*, 38:195–219, 2015.

Oliver D Howes and Ekaterina Shatalina. Integrating the neurodevelopmental and dopamine hypotheses of schizophrenia and the role of cortical excitation-inhibition balance. *Biological psychiatry*, 2022.

Dmitry Ivanov, Aleksandr Chezhegov, Mikhail Kiselev, Andrey Grunin, and Denis Larionov. Neuromorphic artificial intelligence systems. *Frontiers in Neuroscience*, 16:1513, 2022.

Yu Ji, YouHui Zhang, ShuangChen Li, Ping Chi, CiHang Jiang, Peng Qu, Yuan Xie, and WenGuang Chen. Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

Dongshin Kim and Jang-Sik Lee. Neurotransmitter-induced excitatory and inhibitory functions in artificial synapses. *Advanced Functional Materials*, 32(21):2200497, 2022.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Faming Liang and Wing Hung Wong. Evolutionary monte carlo: Applications to cp model sampling and change point problem. *STATISTICA SINICA*, 10:317–342, 2000.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Data-dependent path normalization in neural networks. *arXiv preprint arXiv:1511.06747*, 2015.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:294078, 2017.

Farshad Shirani and Hannah Choi. On the physiological and structural contributors to the dynamic balance of excitation and inhibition in local cortical networks. *bioRxiv*, pages 2023–01, 2023.

Martino Sorbaro, Qian Liu, Massimo Bortone, and Sadique Sheik. Optimizing the energy consumption of spiking neural networks for neuromorphic applications. *Frontiers in neuroscience*, 14:662, 2020.

Christopher H Stock, Sarah E Harvey, Samuel A Ocko, and Surya Ganguli. Synaptic balancing: A biologically plausible local learning rule that provably increases neural network noise robustness without sacrificing task performance. *PLOS Computational Biology*, 18(9):e1010418, 2022.

A. Tavakoli, F. Agostinelli, and P. Baldi. SPLASH: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks*, 140:1–12, 2021. Also: arXiv:2006.08947.

Appendix

A Homogeneous and BiLU Activation Functions

In this section, we generalize the basic example of the introduction from the standpoint of the activation functions. In particular, we consider homogeneous activation functions (defined below). The importance of homogeneity has been previously identified in somewhat different contexts Neyshabur et al. [2015]. Intuitively, homogeneity is a form of linearity with respect to weight scaling and thus it is useful to motivate the concept of homogeneous activation functions by looking at other notions of linearity for activation functions. This will also be useful for Section E where even more general classes of activation functions are considered.

A.1 Additive Activation Functions

Definition A.1. *A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is additively linear if and only if $f(x + y) = f(x) = (f(y)$ for any real numbers x and y.*

Proposition A.2. *The class of additively linear activation functions is exactly equal to the class of linear activation functions, i.e., activation functions of the form $f(x) = ax$.*

Proof. Obviously linear activation functions are additively linear. Conversely, if f is additively linear, the following three properties are true:

(1) One must have: $f(nx) = nf(x)$ and $f(x/n) = f(x)/n$ for any $x \in \mathbb{R}$ and any $n \in \mathbb{N}$. As a result, $f(n/m) = nf(1)/m$ for any integers n and m ($m \neq 0$).

(2) Furthermore, $f(0 + 0) = f(0) + f(0)$ which implies: $f(0) = 0$.

(3) And thus $f(x - x) = f(x) + f(-x) = 0$, which in turn implies that $f(-x) = -f(x)$.

From these properties, it is easy to see that f must be continuous, with $f(x) = xf(1)$, and thus f must be linear. □

A.2 Multiplicative Activation Functions

Definition A.3. *A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is multiplicative if and only if $f(xy) = f(x)(f(y)$ for any real numbers x and y.*

Proposition A.4. *The class of continuous multiplicative activation functions is exactly equal to the class of functions comprising the functions: $f(x) = 0$ for every x, $f(x) = 1$ for every x, and all the even and odd functions satisfying $f(x) = x^c$ for $x \geq 0$, where c is any constant in $\mathbb{R}$.*

Proof. It is easy to check the functions described in the proposition are multiplicative. Conversely, assume f is multiplicative. For both $x = 0$ and $x = 1$, we must have $f(x) = f(xx) = f(x)f(x)$ and thus $f(0)$ is either 0 or 1, and similarly for $f(1)$. If $f(1) = 0$, then for any x we must have $f(x) = 0$ because: $f(x) = f(1x) = f(1)f(x) = 0$. Likewise, if $f(0) = 1$, then for any x we must have $f(x) = 1$ because: $1 = f(0) = f(0x) = f(0)f(x) = f(x)$. Thus, in the rest of the proof, we can assume that $f(0) = 0$ and $f(1) = 1$. By induction, it is easy to see that for any $x \geq 0$ we must have: $f(x^n) = f(x)^n$ and $f(x^{1/n}) = (f(x))^{1/n}$ for any integer (positive or negative). As a result, for any $x \in \mathbb{R}$ and any integers n and m we must have: $f(x^{n/m}) = f(x)^{n/m}$. By continuity this implies that for any $x \geq 0$ and any $r \in R$, we must have: $f(x^r) = f(x)^r$. Now there is some constant c such that: $f(e) = e^c$. And thus, for any $x > 0$, $f(x) = f(e^{\log x}) = [f(e)]^{\log x} = e^{c \log x} = x^c$. To address negative values of x, note that we must have $f[(-1)(-1 = f(1) = 1f(-1)^2$. Thus, $f(-1)$ is either equal to 1 or to -1. Since for any $x > 0$ we have $f(-x) = f(-1)f(x)$, we see that if $f(-1) = 1$ the function must be even ($f(-x) = f(x) = x^c$), and if $f(-1) = -1$ the function must be odd ($f(-x) = -f(x)$). □

We will return to multiplicative activation function in a later section.

A.3 Linearly Scalable Activation Functions

Definition A.5. *A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is linearly scalable if and only if $f(\lambda x) = \lambda f(x)$ for every $\lambda \in \mathbb{R}$.*

Proposition A.6. *The class of linearly scalable activation functions is exactly equal to the class of linear activation functions, i.e., activation functions of the form $f(x) = ax$.*

Proof. Obviously, linear activation functions are linearly scalable. For the converse, if f is linearly multiplicative we must have $f(\lambda x) = \lambda f(x) = x f(\lambda)$ for any x and any λ. By taking $\lambda = 1$, we get $f(x) = f(1)x$ and thus f is linear. □

Thus the concepts of linearly additive or linearly scalable activation function are of limited interest since both of them are equivalent to the concept of linear activation function. A more interesting class is obtained if we consider linearly scalable activation functions, where the scaling factor λ is constrained to be positive ($\lambda > 0$), also called homogeneous functions.

A.4 Homogeneous Activation Functions

Definition A.7. *(Homogeneous) A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is homogeneous if and only if: $f(\lambda x) = \lambda f(x)$ for every $\lambda \in \mathbb{R}$ with $\lambda > 0$.*

Remark A.8. Note that if f is homogeneous, $f(\lambda 0) = \lambda f(0) = f(0)$ for any $\lambda > 0$ and thus $f(0) = 0$. Thus it makes no difference in the definition of homogeneous if we set $\lambda \geq 0$ instead of $\lambda > 0$).

Remark A.9. Clearly, linear activation functions are homogeneous. However, there exists also homogeneous functions that are non-linear, such as ReLU or leaky ReLU activation functions.

We now provide a full characterization of the class of homogeneous activation functions.

A.5 BiLU Activation Functions

We first define a new class of activation functions, corresponding to bilinear units (BiLU), consisting of two half-lines meeting at the origin. This class contains all the linear functions, as well as the ReLU and leaky ReLU functions, and many other functions.

Definition A.10. *(BiLU) A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is bilinear (BiLU) if and only if $f(x) = ax$ when $x < 0$, and $f(x) = bx$ when $x \geq 0$, for some fixed parameters a and b in $\mathbb{R}$.*

These include linear units ($a = b$), ReLU units ($a = 0, b = 1$), leaky ReLU ($a = \epsilon; b = 1$) units, and symmetric linear units ($a = -b$), all of which can also be viewed as special cases of piece-wise linear units Tavakoli et al. [2021], with a single hinge. One advantage of ReLU and more generally BiLU neurons, which is very important during backpropagation learning, is that their derivative is very simple and can only take one of two values (a or b).

Proposition A.11. *A neuronal activation function $f : \mathbb{R} \to \mathbb{R}$ is homogeneous if and only if it is a BiLU activation function.*

Proof. Every function in BiLU is clearly homogeneous. Conversely, any homogeneous function f must satisfy: (1) $f(0x) = 0 f(x) = f(0) = 0$; (2)$f(x) = f(1x) = f(1)x$ for any positive x; and (3) $f(x) = f(-u) = f(-1)u = -f(-1)x$ for any negative x. Thus f is in BiLU with $a = -f(-1)$ and $b = f(1)$. □

In Appendix A, we provide a simple proof that networks of BiLU neurons, even with a single hidden layer, have universal approximation properties. In the next two sections, we introduce two fundamental neuronal operations, scaling and balancing, that can be applied to the incoming and outgoing synaptic weights of neurons with BiLU activation functions.

B Scaling

Definition B.1. *(Scaling) For any BiLU neuron i in network and any $\lambda > 0$, we let $S_\lambda(i)$ denote the synaptic scaling operation by which the incoming connection weights of neuron i are multiplied by λ and the outgoing connection weights of neuron i are divided by λ.*

Note that because of the homogeneous property, the scaling operation does not change how neuron i affects the rest of the network. In particular, the input-output function of the overall network remains unchanged after scaling neuron i bt any $\lambda > 0$. Note also that scaling always preserves the sign of the synaptic weights to which it is applied, and the scaling operation can never convert a non-zero synaptic weight into a zero synaptic weight, or vice versa.

As usual, the bias is treated here as an additional synaptic weight emanating from a unit clamped to the value one. Thus scaling is applied to the bias.

Proposition B.2. *(Commutativity of Scaling) Scaling operations applied to any pair of BiLU neurons i and j in a neural network commute: $S_\lambda(i)S_\mu(j) = S_\mu(j)S_\lambda(i)$, in the sense that the resulting network weights are the same, regardless of the order in which the scaling operations are applied. Furthermore, for any BiLU neuron i: $S_\lambda(i)S_\mu(i) = S_\mu(i)S_\lambda(i) = S_{\lambda\mu}(i)$.*

This is obvious. As a result, any set I of BiLU neurons in a network can be scaled simultaneously or in any sequential order while leading to the same final configuration of synaptic weights. If we denote by $1, 2, \ldots, n$ the neurons in I, we can for instance write: $\prod_{i\in I} S_{\lambda_i}(i) = \prod_{\sigma(i)\in I} S_{\lambda_{\sigma(i)}}(\sigma(i))$ for any permutation σ of the neurons. Likewise, we can collapse operations applied to the same neuron. For instance, we can write: $S_5(1)S_2(2)S_3(1)S_4(2) = S_{15}(1)S_8(2) = S_8(2)S_{15}(1)$

Definition B.3. *(Coordinated Scaling) For any set I of BiLU neurons in a network and any $\lambda > 0$, we let $S_\lambda(I)$ denote the synaptic scaling operation by which all the neurons in I are scaled by the same λ.*

C Balancing

Definition C.1. *(Balancing) Given a BiLU neuron in a network, the balancing operation $B(i)$ is a particular scaling operation $B(i) = S_{\lambda^*}(i)$, where the scaling factor λ^* is chosen to optimize a particular cost function, or regularizer, associated with the incoming and outgoing weights of neuron i.*

For now, we can imagine that this cost function is the usual L_2 (least squares) regularizer, but in the next section, we will consider more general classes of regularizers and study the corresponding optimization process. For the L_2 regularizer, as shown in the next section, this optimization process results in a unique value of λ^* such that sum of the squares of the incoming weights is equal to the sum of the squares of the outgoing weights, hence the term "balance". Note that obviously $B(B(i)) = B(i)$ and that, as a special case of scaling operation, the balancing operation does not change how neuron i contributes to the rest of the network, and thus it leaves the overall input-output function of the network unchanged.

Unlike scaling operations, balancing operations in general do not commute as balancing operations (they still commute as scaling operations). Thus, in general, $B(i)B(j) \neq B(j)B(i)$. This is because if neuron i is connected to neuron j, balancing i will change the connection between i and j, and, in turn, this will change the value of the optimal scaling constant for neuron j and vice versa. However, if there are no non-zero connections between neuron i and neuron j then the balancing operations commute since each balancing operation will modify a different, non-overlapping, set of weights.

Definition C.2. *(Disjoint neurons) Two neurons i and j in a neural network are said to be disjoint if there are no non-zero connections between i and j.*

Thus in this case $B(i)B(j) = S_{\lambda^*}(i)S_{\mu^*}(j) = S_{\mu^*}(j)S_{\lambda^*}(i) = B(j)B(i)$. This can be extended to disjoint sets of neurons.

Definition C.3. *(Disjoint Set of Neurons) A set I of neurons is said to be disjoint if for any pair i and j of neurons in I there are no non-zero connections between i and j.*

For example, in a layered feedforward network, all the neurons in a layer form a disjoint set, as long as there are no intra-layer connections or, more precisely, no non-zero intra-layer connections. All

the neurons in a disjoint set can be balanced in any order resulting in the same final set of synaptic weights. Thus we have:

Proposition C.4. *If we index by $1, 2, \ldots, n$ the neurons in a disjoint set I of BiLU neurons in a network, we have: $\prod_{i\in I} B(i) = \prod_{i\in I} S_{\lambda_i^*}(i) = \prod_{\sigma(i)\in I} S_{\lambda^*_{\sigma(i)}}(\sigma(i)) = \prod_{\sigma(i)\in I} B(\sigma(i))$ for any permutation σ of the neurons.*

Finally, we can define the coordinated balancing of any set I of BiLU neurons (disjoint or not disjoint).

Definition C.5. *(Coordinated Balancing) Given any set I of BiLU neurons (disjoint or not disjoint) in a network, the coordinated balancing of these neurons, written as $B_{\lambda^*}(I)$, corresponds to the coordinated scaling all the neurons in I by the same factor λ^*, Where λ^* minimizes the cost functions of all the weights, incoming and outgoing, associated with all the neurons in I.*

Remark C.6. While balancing corresponds to a full optimization of the scaling operation, it is also possible to carry a partial optimization of the scaling operation by choosing a scaling factor that reduces the corresponding contribution to the regularizer without minimizing it.

D General Framework and Single Neuron Balance

In this section, we generalize the kinds of regularizer to which the notion of neuronal synaptic balance can be applied, beyond the usual L_2 regularizer and derive the corresponding balance equations. Thus we consider a network (feedforward or recurrent) where the hidden units are BiLU units. The visible units can be partitioned into input units and output units. For any hidden unit i, if we multiply all its incoming weights $IN(i)$ by some $\lambda > 0$ and all its outgoing weights $OUT(i)$ by $1/\lambda$ the overall function computed by the network remains unchanged due to the BiLU homogeneity property. In particular, if there is an error function that depends uniquely on the input-output function being computed, this error remains unchanged by the introduction of the multiplier λ. However, if there is also a regularizer R for the weights, its value is affected by λ and one can ask what is the optimal value of λ with respect to the regularizer, and what are the properties of the resulting weights. This approach can be applied to any regularizer. For most practical purposes, we can assume that the regularizer is continuous in the weights (hence in λ) and lower-bounded. Without any loss of generality, we can assume that it is lower-bounded by zero. If we want the minimum value to be achieved by some $\lambda > 0$, we need to add some mild condition that prevents the minimal value from being approached as $\lambda \to 0^)$, or as $\lambda \to +\infty$. For instance, it is enough if there is an interval $[a, b]$ with $0 < a < b$ where R achieves a minimal value R_{min} and $R \geq R_{min}$ in the intervals $(0, a]$ and $[b, +\infty)$. Additional (mild) conditions must be imposed if one wants the optimal value of λ to be unique, or computable in closed form (see Theorems below). Finally, we want to be able to apply the balancing approach

Thus, we consider overall regularized error functions, where the regularizer is very general, as long as it has an additive form with respect to the individual weights:

$$\mathcal{E}(W) = E(W) + R(W) \quad \text{with} \quad R(W) = \sum_w g_w(w) \tag{D.1}$$

where W denotes all the weights in the network and $E(W)$ is typically the negative log-likelihood (LMS error in regression tasks, or cross-entropy error in classification tasks). We assume that the g_w are continuous, and lower-bounded by 0. To ensure the existence and uniqueness of minimum during the balancing of any neuron, We will assume that each function g_w depends only on the magnitude $|w|$ of the corresponding weight, and that g_w is monotonically increasing from 0 to $+\infty$ ($g_w(0) = 0$ and $\lim_{x\to+\infty} g_w(x) = +\infty$). Clearly, L_2, L_1 and more generally all L_p regularizers are special cases where, for $p > 0$, L^p regularization is defined by: $R(W) = \sum_w |w|^p$.

When indicated, we may require also that the functions g_w be continuously differentiable, except perhaps at the origin in order to be able to differentiate the regularizer with respect to the λ's and derive closed form conditions for the corresponding optima. This is satisfied by all forms of L_p regularization, for $p > 0$.

Remark D.1. Often one introduces scalar multiplicative hyperparameters to balance the effect of E and R, for instance in the form: $\mathcal{E} = E + \beta R$. These cases are included in the framework above: multipliers like β can easily be absorbed into the functions g_w above.

Theorem D.2. *(General Balance Equation). Consider a neural network with BiLU activation functions in all the hidden units and overall error function of the form:*

$$\mathcal{E} = E(W) + R(W) \quad \text{with} \quad R(W) = \sum_w g_w(w) \tag{D.2}$$

where each function $g_w(w)$ is continuous, depends on the magnitude $|w|$ alone, and grows monotonically from $g_w(0) = 0$ to $g_w(+\infty) = +\infty$. For any setting of the weights W and any hidden unit i in the network and any $\lambda > 0$ we can multiply the incoming weights of i by λ and the outgoing weights of i by $1/\lambda$ without changing the overall error E. Furthermore, there exists a unique value λ^* where the corresponding weights v ($v = \lambda^* w$ for incoming weights, $v = w/\lambda^*$ for the outgoing weights) achieve the balance equation:

$$\sum_{v \in IN(i)} g_w(v) = \sum_{w \in OUT(i)} g_w(v) \tag{D.3}$$

Proof. Under the assumptions of the theorem, E is unchanged under the rescaling of the incoming and outgoing weights of unit i due to the homogeneity property of BiLUs. Without any loss of generality, let us assume that at the beginning: $\sum_{w \in IN(i)} g_w(w) < \sum_{w \in OUT(i)} g_w(w)$. As we increase λ from 1 to $+\infty$, by the assumptions on the functions g_w, the term $\sum_{w \in IN(i)} g_w(\lambda w)$ increases continuously from its initial value to $+\infty$, whereas the term $\sum_{w \in OUT(i)} g_w)w/\lambda)$ decreases continuously from its initial value to 0. Thus, there is a unique value λ^* where the balance is realized. If at the beginning $\sum_{w \in IN(i)} g_w(w) > \sum_{w \in OUT(i)} g_w(w)$, then the same argument is applied by decreasing λ from 1 to 0. □

Remark D.3. For simplicity, here and in other sections, we state the results in terms of a network of BiLU units. However, the same principles can be applied to networks where only a subset of neurons are in the BiLU class, simply by applying scaling and balancing operations to only those neurons. Furthermore, not all BiLU neurons need to have the same BiLU activation function. For instance, the results still hold for a mixed network containing both ReLU and linear units.

Remark D.4. In the setting of Theorem D.2, the balance equations do not necessarily minimize the corresponding regularization term. This is addressed in the next theorem.

Remark D.5. Finally, zero weights ($w = 0$) can be ignored entirely as they play no role in scaling or balancing. Furthermore, if all the incoming or outgoing weights of a hidden unit were to be zero, it could be removed entirely from the network

Theorem D.6. *(Balance and Regularizer Minimization) We now consider the same setting as in Theorem D.2, but in addition, we assume that the functions g_w are continuously differentiable, except perhaps at the origin. Then, for any neuron, there exists at least one optimal value λ^* that minimizes $R(W)$. This value must be a solution of the consistency equation:*

$$\lambda^2 \sum_{w \in IN(i)} w g'_w(\lambda w) = \sum_{w \in OUT(i)} w g'_w(w/\lambda) \tag{D.4}$$

Once the weights are rebalanced accordingly, the new weights must satisfy the generalized balance equation:

$$\sum_{w \in IN(i)} w g'(w) = \sum_{w \in OUT(i)} w g'(w) \tag{D.5}$$

In particular, if $g_w(w) = |w|^p$ for all the incoming and outgoing weights of neuron i, then the optimal value λ^* is unique and equal to:

$$\lambda^* = \Big(\frac{\sum_{w \in OUT(i)} |w|^p}{\sum_{w \in IN(i)} |w|^p} \Big)^{1/2p} = \Big(\frac{||OUT(i)||_p}{||IN(i)||_p} \Big)^{1/2} \tag{D.6}$$

The decrease $\Delta R \geq 0$ in the value of the L_p regularizer $R = \sum_w |w|^p$ is given by:

$$\Delta R = \left(\Big(\sum_{w \in IN(i)} |w|^p \Big)^{1/2} - \Big(\sum_{w \in OUT(i)} |w|^p \Big)^{1/2} \right)^2 \tag{D.7}$$

After balancing neuron i, its new weights satisfy the generalized L_p balance equation:

$$\sum_{w \in IN(i)} |w|^p = \sum_{w \in OUT(i)} |w|^p \tag{D.8}$$

Proof. Due to the additivity of the regularizer, the only component of the regularizer that depends on λ has the form:

$$R(\lambda) = \sum_{w \in IN(i)} g_w(\lambda w) + \sum_{w \in OUT(i)} g_w(w/\lambda) \tag{D.9}$$

Because of the properties of the functions g_w, R_λ is continuously differentiable and strictly bounded below by 0. So it must have a minimum, as a function of λ where its derivative is zero. Its derivative with respect to λ has the form:

$$R'(\lambda) = \sum_{w \in IN(i)} w g'_w(\lambda w) + \sum_{w \in OUT(i)} (-w/\lambda^2) g'_w(w/\lambda) \tag{D.10}$$

Setting the derivative to zero, gives:

$$\lambda^2 \sum_{w \in IN(i)} w g'_w(\lambda w) = \sum_{w \in OUT(i)} w g'_w(w/\lambda) \tag{D.11}$$

Assuming that the left-hand side is non-zero, which is generally the case, the optimal value for λ must satisfy:

$$\lambda = \Big(\frac{\sum_{w \in OUT(i)} w g'_w(w/\lambda)}{\sum_{w \in IN(i)} w g'_w(\lambda w)} \Big)^{1/2} \tag{D.12}$$

If the regularizing function is the same for all the incoming and outgoing weights ($g_w = g$), then the optimal value λ must satisfy:

$$\lambda = \Big(\frac{\sum_{w \in OUT(i)} w g'(w/\lambda)}{\sum_{w \in IN(i)} w g'(\lambda w)} \Big)^{1/2} \tag{D.13}$$

In particular, if $g(w) = |w|^p$ then $g(w)$ is differentiable except possibly at 0 and $g'(w) = s(w)p|w|^{p-1}$, where $s(w)$ denotes the sign of the weight w. Substituting in Equation D.13, the optimal rescaling λ must satisfy:

$$\lambda^* = \Big(\frac{\sum_{w \in OUT(i)} w s(w) |w|^{p-1}}{\sum_{w \in IN(i)} w |w s(w)|^{p-1}} \Big)^{1/2p} = \Big(\frac{\sum_{w \in OUT(i)} |w|^p}{\sum_{w \in IN(i)} |w|^p} \Big)^{1/2p} = \Big(\frac{||OUT(i)||_p}{||IN(i)||_p} \Big)^{1/2} \tag{D.14}$$

At the optimum, no further balancing is possible, and thus $\lambda^* = 1$. Equation D.11 yields immediately the generalized balance equation to be satisfied at the optimum:

$$\sum_{w \in IN(i)} w g'(w) = \sum_{w \in OUT(i)} w g'(w) \tag{D.15}$$

In the case of L_P regularization, it is easy to check by applying Equation D.15, or by direct calculation that:

$$\sum_{w \in IN(i)} |\lambda^* w|^p = \sum_{w \in OUT(i)} |w/\lambda^*|^p \tag{D.16}$$

which is the generalized balance equation. Thus after balancing neuron, the weights of neuron i satisfy the L_p balance (Equation D.8). The change in the value of the regularizer is given by:

$$\Delta R = \sum_{w \in IN(i)} |w|^p + \sum_{w \in OUT(i)} |w|^p - \sum_{w \in IN(i)} |\lambda^* w|^p - \sum_{w \in OUT(i)} |w/\lambda^*|^p \tag{D.17}$$

By substituting λ^* by its explicit value given by Equation D.14 and collecting terms gives Equation D.7. □

Remark D.7. The monotonicity of the functions g_w is not needed to prove the first part of Theorem D.6. It is only needed to prove the uniqueness of λ^* in the L_p cases.

Remark D.8. Note that the same approach applies to the case where there are multiple additive regularizers. For instance with both L^2 and L^1 regularization, in this case the function f has the form: $g_w(w) = \alpha w^2 + \beta|w|$. Generalized balance still applies. It also applies to the case where different regularizers are applied in different disconnected portions of the network.

Remark D.9. The balancing of a single BiLU neuron has little to do with the number of connections. It applies equally to fully connected neurons, or to sparsely connected neurons.

E Scaling and Balancing Beyond BiLU Activation Functions

So far we have generalized ReLU activation functions to BiLU activation functions in the context of scaling and balancing operations with positive scaling factors. While in the following sections we will continue to work with BiLU activation functions, in this section we show that the scaling and balancing operations can be extended even further to other activation functions. The section can be skipped if one prefers to progress towards the main results on stochastic balancing.

Given a neuron with activation function $f(x)$, during scaling instead of multiplying and dividing by $\lambda > 0$, we could multiply the incoming weights by a function $g(\lambda)$ and divide the outgoing weights by a function $h(\lambda)$, as long as the activation function f satisfies:

$$f(g(\lambda)x) = h(\lambda)f(x) \tag{E.1}$$

for every $x \in \mathbb{R}$ to ensure that the contribution of the neuron to the rest of the network remains unchanged. Note that if the activation function f satisfies Equation E.1, so does the activation function $-f$. In Equation E.1, λ does not have to be positive–we will simply assume that λ belongs to some open (potentially infinite) interval (a, b). Furthermore, the functions g and h cannot be zero for $\lambda \in (a, b)$ since they are used for scaling. It is reasonable to assume that the functions g and h are continuous, and thus they must have a constant sign as λ varies over (a, b).

Now, taking $x = 0$ gives $f(0) = h(\lambda)f(0)$ for every $\lambda \in (a, b)$, and thus either $f(0) = 0$ or $h(\lambda) = 1$ for every $\lambda \in (a, b)$. The latter is not interesting and thus we can assume that the activation function f satisfies $f(0) = 0$. Taking $x = 1$ gives $f(g(\lambda)) = h(\lambda)f(1)$ for every λ in (a, b). For simplicity, let us assume that $f(x) = 1$. Then, we have: $f(g(\lambda)) = h(\lambda)$ for every λ. Substituting in Equation E.1 yields:

$$f(g(\lambda)x) = f(g(\lambda))f(x) \tag{E.2}$$

for every $x \in \mathbb{R}$ and every $\lambda \in (a, b)$. This relation is essentially the same as the relation that defines multiplicative activation functions over the corresponding domain (see Proposition A.4), and thus we can identify a key family of solutions using power functions. Note that we can define a new parameter $\mu = g(\lambda)$, where μ ranges also over some positive or negative interval I over which: $f(\mu x) = f(\mu)f(x)$.

E.1 Bi-Power Units (BiPU)

Let us assume that $\lambda > 0$, $g(\lambda) = \lambda$ and $h(\lambda) = \lambda^c$ for some $c \in \mathbb{R}$. Then the activation function must satisfy the equation:

$$f(\lambda x) = \lambda^c f(x) \tag{E.3}$$

for any $x \in \mathbb{R}$ and any $\lambda > 0$. Note that if $f(x) = x^c$ we get a multiplicative activation function. More generally, these functions are characterized by the following proposition.

Proposition E.1. *The set of activation functions f satisfying $f(\lambda x) = \lambda^c f(x)$ for any $x \in \mathbb{R}$ and any $\lambda > 0$ consist of the functions of the form:*

$$f(x) = \begin{cases} Cx^c & \text{if} \quad x \geq 0 \\ Dx^c & \text{if} \quad x < 0. \end{cases} \tag{E.4}$$

where $c \in \mathbb{R}$, $C = f(1) \in R$, and $D = f(-1) \in \mathbb{R}$. We call these bi-power units (BiPU). If, in addition, we want f to be continuous at 0, we must have either $c > 0$, or $c = 0$ with $C = D$.

Given the general shape, these activations functions can be called BiPU (Bi-Power-Units). Note that in the general case where $c > 0$, C and D do not need to be equal. In particular, one of them can be equal to zero, and the other one can be different from zero giving rise to "rectified power units" (Figure 6).

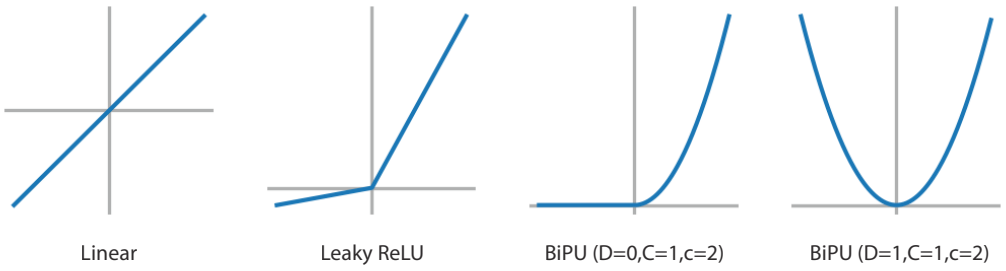

Figure 6

Proof. By taking $x = 1$, we get $f(\lambda) = f(1)\lambda^c$ for any $\lambda > 0$. Let $f(1) = C$. Then we see that for any $x > 0$ we must have: $f(x) = Cx^c$. In addition, for every $\lambda > 0$ we must have: $f(\lambda 0) = f(0) = \lambda^c f(0)$. So if $c = 0$, then $f(x) = C = f(1)$ for $x \geq 0$. If $c \neq 0$, then $f(0) = 0$. In this case, if we want the activation function to be continuous, then we see that we must have $c \geq 0$. So in summary for $x > 0$ we must have $f(x) = f(1)x^c = Cx^c$. For the function to be right continuous at 0, we must have either $f(0) = f(1) = C$ with $c = 0$ or $f(0) = 0$ with $c > 0$. We can now look at negative values of x. By the same reasoning, we have $f(\lambda(-1)) = f(-\lambda) = \lambda^c f(-1)$ for any $\lambda > 0$. Thus for any $x < 0$ we must have: $f(x) = f(-1)|x|^c = D|x|^c$ where $D = f(-1)$. Thus, if f is continuous, there are two possibilities. If $c = 0$, then we must have $C = f(1) = D(f-1)-$ and thus $f(x) = C$ everywhere. If $c \neq 0$, then continuity requires that $c > 0$. In this case $f(x) = Cx^c$ for $x \geq 0$ with $C = f(1)$, and $f(x) = Dx^c$ for $x < 0$ with $f(-1) = D$. In all cases, it is easy to check directly that the resulting functions satisfy the functional equation given by Equation E.3. □

E.2 Scaling BiPU Neurons

A BiPU neuron can be scaled by multiplying its incoming weight by $\lambda > 0$ and dividing its outgoing weights by $1/\lambda^c$. This will not change the role of the corresponding unit in the network, and thus it will not change the input-output function of the network.

E.3 Balancing BiPU Neurons

As in the case of BiLU neurons, we balance a multiplicative neuron by asking what is the optimal scaling factor λ that optimizes a particular regularizer. For simplicity, here we assume that the regularizer is in the L_p class. Then we are interested in the value of $\lambda > 0$ that minimizes the function:

$$\lambda^p \sum_{w \in IN} |w|^p + \frac{1}{\lambda^{pc}} \sum_{w \in OUT} |w|^p \tag{E.5}$$

A simple calculation shows that the optimal value of λ is given by:

$$\lambda^* = \Big(\frac{c\sum_{OUT}|w|^p}{\sum_{IN}|w|^p}\Big)^{1/p(c+1)} \tag{E.6}$$

Thus after balancing the weights, the neuron must satisfy the balance equation:

$$c\sum_{OUT}|w|^p = \sum_{IN}|w|^p \tag{E.7}$$

in the new weights w.

So far, we have focused on balancing individual neurons. In the next two sections, we look at balancing across all the units of a network. We first look at what happens to network balance when a network is trained by gradient descent and then at what happens to network balance when individual neurons are balanced iteratively in a regular or stochastic manner.

F Network Balance: Gradient Descent

A natural question is whether gradient descent (or stochastic gradient descent) applied to a network of BiLU neurons, with or without a regularizer, converges to a balanced state of the network, where all the BiLU neurons are balanced. So we first consider the case where there is no regularizer ($\mathcal{E} = E$). The results in Du et al. [2018] may suggest that gradient descent may converge to a balanced state. In particular, they write that for any neuron i:

$$\frac{d}{dt}\Big(\sum_{w\in IN(i)} w^2 - \sum_{w\in OUT(i)} w^2\Big) = 0 \tag{F.1}$$

Thus the gradient flow exactly preserves the difference between the L_2 cost of the incoming and outgoing weights or, in other words, the derivative of the L_2 balance *deficit* is zero. Thus if one were to start from a balanced state and use an infinitesimally small learning rate one ought to stay in a balanced state at all times.

However, it must be noted that this result was derived for the L_2 metric only, and thus would not cover other L_p forms of balance. Furthermore, it requires an infinitesimally small learning rate. In practice, when any standard learning rate is applied, we find that gradient descent does *not* converge to a balanced state (Figure 1). However, things are different when a regularizer term is included in the error functions as described in the following theorem.

Theorem F.1. *Gradient descent in a network of BiLU units with error function $\mathcal{E} = E + R$ where R has the properties described in Theorem D.6 (including all L_p) must converge to a balanced state, where every BiLU neuron is balanced.*

Proof. By contradiction, suppose that gradient descent converges to a state that is unbalanced and where the gradient with respect to all the weights is zero. Then there is at least one unbalanced neuron in the network. We can then multiply the incoming weights of such a neuron by λ and the outgoing weights by $1/\lambda$ as in the previous section without changing the value of E. Since the neuron is not in balance, we can move λ infinitesimally so as to reduce R, and hence $\mathcal{E}$. But this contradicts the fact that the gradient is zero. □

Remark F.2. In practice, in the case of stochastic gradient descent applied to $E + R$, at the end of learning the algorithm may hover around a balanced state. If the state reached by the stochastic gradient descent procedure is not approximately balanced, then learning ought to continue. In other words, the degree of balance could be used to monitor whether learning has converged or not. Balance is a necessary, but not sufficient, condition for being at the optimum.

Remark F.3. If early stopping is being used to control overfitting, there is no reason for the stopping state to be balanced. However, the balancing algorithms described in the next section could be used to balance this state.

G Network Balance: Stochastic or Deterministic Balancing Algorithms

In this section, we look at balancing algorithms where, starting from an initial weight configuration W, the BiLU neurons of a network are balanced iteratively according to some deterministic or stochastic schedule that periodically visits all the neurons. We can also include algorithms where neurons are partitioned into groups (e.g. neuronal layers) and neurons in each group are balanced together.

G.1 Basic Stochastic Balancing

The most interesting algorithm is when the BiLU neurons of a network are iteratively balanced in a purely stochastic manner. This algorithm is particularly attractive from the standpoint of physically implemented neural networks because the balancing algorithm is local and the updates occur randomly without the need for any kind of central coordination. As we shall see in the following section, the random local operations remarkably lead to a unique form of global order. The proof for the stochastic case extends immediately to the deterministic case, where the BiLU neurons are updated in a deterministic fashion, for instance by repeatedly cycling through them according to some fixed order.

G.2 Subset Balancing (Independent or Tied)

It is also possible to partition the BiLU neurons into non-overlapping subsets of neurons, and then balance each subset, especially when the neurons in each subset are disjoint of each other. In this case, one can balance all the neurons in a given subset, and repeat this subset-balancing operation subset-by-subset, again in a deterministic or stochastic manner. Because the BiLU neurons in each subset are disjoint, it does not matter whether the neurons in a given subset are updated synchronously or sequentially (and in which order). Since the neurons are balanced independently of each other, this can be called independent subset balancing. For example, in a layered feedforward network with no lateral connections, each layer corresponds to a subset of disjoint neurons. The incoming and outgoing connections of each neuron are distinct from the incoming and outgoing connections of any other neuron in the layer, and thus the balancing operation of any neuron in the layer does not interfere with the balancing operation of any other neuron in the same layer. So this corresponds to independent layer balancing,

As a side note, balancing a layer h, may disrupt the balance of layer $h+1$. However, balancing layers h and $h+2$ (or any other layer further apart) can be done without interference of the balancing processes. This suggests also an alternating balancing scheme, where one alternatively balances all the odd-numbered layers, and all the evenly-numbered layers.

Yet another variation is when the neurons in a disjoint subset are tied to each other in the sense that they must all share the same scaling factor λ. In this case, balancing the subset requires finding the optimal λ for the entire subset, as opposed to finding the optimal λ for each neuron in the subset. Since the neurons are balanced in a coordinated or tied fashion, this can be called coordinated or tied subset balancing. For example, tied layer balancing must use the same λ for all the neurons in a given layer. It is easy to see that this approach leads to layer synaptic balance which has the form (for an L_p regularizer):

$$\sum_i \sum_{w \in IN(i)} |w|^p = \sum_i \sum_{w \in OUT(i)} |w|^p \tag{G.1}$$

where i runs over all the neurons in the layer. This does *not* necessarily imply that each neuron in the layer is individually balanced. Thus neuronal balance for every neuron in a layer implies layer balance, but the converse is not true. Independent layer balancing will lead to layer balance. Coordinated layer balancing will lead to layer balance, but not necessarily to neuronal balance of each neuron in the layer. Layer-wise balancing, independent or tied, can be applied to all the layers and in a deterministic (e.g. sequential) or stochastic manner. Again the proof given in the next section for the basic stochastic algorithm can easily be applied to these cases (see also Appendix B).

G.3 Remarks about Weight Sharing and Convolutional Neural Networks

Suppose that two connections share the same weight so that we must have: $w_{ij} = w_{kl}$ at all times. In general, when the balancing algorithm is applied to neuron i or j, the weight w_{ij} will change and the same change must be applied to w_{kl}. The latter may disrupt the balance of neuron k or l. Furthermore, this may not lead to a decrease in the overall value of the regularizer R.

The case of convolutional networks is somewhat special, since *all* the incoming weights of the neurons sharing the same convolutional kernel are shared. However, in general, the outgoing weights are not shared. Furthermore, certain operations like max-pooling are not homogeneous. So if one trains a CNN with E alone, or even with $E + R$, one should not expect any kind of balance to emerge in the convolution units. However, all the other BiLU units in the network should become balanced by the same argument used for gradient descent above. The balancing algorithm applied to individual neurons, or the independent layer balancing algorithm, will not balance individual neurons sharing the same convolution kernel. The only balancing algorithm that could lead to some convolution layer balance, but not to individual neuronal balance, is the coordinated layer balancing, where the same λ is used for all the neurons in the same convolution layer, provided that their activation functions are BiLU functions.

We can now study the convergence properties of balancing algorithms.

H Convergence of Balancing Algorithms

We now consider the basic stochastic balancing algorithm, where BiLU neurons are iteratively and stochastically balanced. It is essential to note that balancing a neuron j may break the balance of another neuron i to which j is connected. Thus convergence of iterated balancing is not obvious. There are three key questions to be addressed for the basic stochastic algorithm, as well as all the other balancing variations. First, does the value of the regularizer converge to a finite value? Second, do the weights themselves converge to fixed finite values representing a balanced state for the entire network? And third, if the weights converge, do they always converge to the same values, irrespective of the order in which the units are being balanced? In other words, given an initial state W for the network, is there a unique corresponding balanced state, with the same input-output functionalities?

H.1 Notation and Key Questions

For simplicity, we use a continuous time notation. After a certain time t each neuron has been balanced a certain number of times. While the balancing operations are not commutative as balancing operations, they are commutative as scaling operations. Thus we can reorder the scaling operations and group them neuron by neuron so that, for instance, neuron i has been scaled by the sequence of scaling operations:

$$S_{\lambda_1^*}(i)S_{\lambda_2^*}(i)\dots S_{\lambda^*_{n_{it}}}(i) = S_{\Lambda_i(t)}(i) \tag{H.1}$$

where n_{it} corresponds to the count of the last update of neuron i prior to time t, and:

$$\Lambda_i(t) = \prod_{1\leq n\leq n_{it}} \lambda_n^*(i) \tag{H.2}$$

For the input and output units, we can consider that their balancing coefficients λ^* are always equal to 1 (at all times) and therefore $\Lambda_i(t) = 1$ for any visible unit i.

Thus, we first want to know if R converges. Second, we want to know if the weights converge. This question can be split into two sub-questions: (1) Do the balancing factors $\lambda_n^*(i)$ converge to a limit as time goes to infinity? Even if the $\lambda_n^*(i)$'s converge to a limit, this does not imply that the weights of the network converge to a limit. After a time t, the weight $w_{ij}(t)$ between neuron j and neuron i has the value $w_{ij}\Lambda_i(t)/\Lambda_j(t)$, where $w_{ij} = w_{ij}(0)$ is the value of the weight at the start of the stochastic balancing algorithm. Thus: (2) Do the quantities $\Lambda_i(t)$ converge to finite values, different from 0? And third, if the weights converge to finite values different from 0, are these values unique or not, i.e. do they depend on the details of the stochastic updates or not? These questions are answered by the following main theorem..

H.2 Convergence of the Basic Stochastic Balancing Algorithm to a Unique Optimum

Theorem H.1. *(Convergence of Stochastic Balancing) Consider a network of BiLU neurons with an error function $\mathcal{E}(W) = E(W) + R(W)$ where R satisfies the conditions of Theorem D.2 including all L_p ($p > 0$). Let W denote the initial weights. When the neuronal stochastic balancing algorithm is applied throughout the network so that every neuron is visited from time to time, then $E(W)$ remains unchanged but $R(W)$ must converge to some finite value that is less or equal to the initial value, strictly less if the initial weights are not balanced. In addition, for every neuron i, $\lambda_i^*(t) \to 1$ and $\Lambda_i(t) \to \Lambda_i$ as $t \to \infty$, where Λ_i is finite and $\Lambda_i > 0$ for every i. As a result, the weights themselves must converge to a limit W' which is globally balanced, with $E(W) = E(W')$ and $R(W) \geq R(W')$, and with equality if only if W is already balanced. Finally, W' is unique as it corresponds to the solution of a strictly convex optimization problem in the variables $L_{ij} = \log(\Lambda_i/\Lambda_j)$ with linear constraints of the form $\sum_\pi L_{ij} = 0$ along any path π joining an input unit to an output unit and along any directed cycle (for recurrent networks). Stochastic balancing projects to stochastic trajectories in the linear manifold that run from the origin to the unique optimal configuration.*

Proof. Each individual balancing operation leaves $E(W)$ unchanged because the BiLU neurons are homogeneous. Furthermore, each balancing operation reduces the regularization error $R(W)$, or leaves it unchanged. Since the regularizer is lower-bounded by zero, the value of the regularizer must approach a limit as the stochastic updates are being applied.

For the second question, when neuron i is balanced at some step, we know that the regularizer R decreases by:

$$\Delta R = \left(\Big(\sum_{w \in IN(i)} |w|^p \Big)^{1/2} - \Big(\sum_{w \in OUT(i)} |w|^p \Big)^{1/2} \right)^2 \tag{H.3}$$

If the convergence were to occur in a finite number of steps, then the coefficients $\lambda_i^*(t)$ must become equal and constant to 1 and the result is obvious. So we can focus on the case where the convergence does not occur in a finite number of steps (indeed this is the main scenario, as we shall see at the end of the proof). Since $\Delta R \to 0$, we must have:

$$\sum_{w \in IN(i)} |w|^p \to \sum_{w \in OUT(i)} |w|^p \tag{H.4}$$

But from the expression for λ^* (Equation D.14), this implies that for every i, $\lambda_n^*(i) \to 1$ as time increases ($n \to \infty$). This alone is not sufficient to prove that $\Lambda_i(t)$ converges for every i as $t \to \infty$. However, it is easy to see that $\Lambda_i(t)$ cannot contain a sub-sequence that approaches 0 or ∞ (Figure 7). Furthermore, not only ΔR converges to 0, but the series $\sum \Delta R$ is convergent. This shows that, for every i, $\Delta_i(t)$ must converge to a finite, non-zero value Δ_i. Therefore all the weights must converge to fixed values given by $w_{ij}(0)\Lambda_i/\Lambda_j$.

Finally, we prove that given an initial set of weights W, the final balanced state is unique and independent of the order of the balancing operations. The coefficients Λ_i corresponding to a globally balanced state must be solutions of the following optimization problem:

$$\min_{\Lambda} R(\Lambda) = \sum_{ij} |\frac{\Lambda_i}{\Lambda_j} w_{ij}|^p \tag{H.5}$$

under the simple constraints: $\Lambda_i > 0$ for all the BiLU hidden units, and $\Lambda_i = 1$ for all the visible (input and output) units. In this form, the problem is not convex. Introducing new variables $M_j = 1/\Lambda_j$ is not sufficient to render the problem convex. Using variables $M_{ij} = \Lambda_i/\Lambda_j$ is better, but still problematic for $0 < p \leq 1$. However, let us instead introduce the new variables $L_{ij} = \log(\Lambda_i/\Lambda_j)$. These are well defined since we know that $\Lambda_i/\Lambda_j > 0$. The objective now becomes:

$$\min R(L) = \sum_{ij} |e^{L_{ij}} w_{ij}|^p = \sum_{ij} e^{pL_{ij}} |w_{ij}|^p \tag{H.6}$$

This objective is strictly convex in the variables L_{ij}, as a sum of strictly convex functions (exponentials). However, to show that it is a convex optimization problem we need to study the constraints on

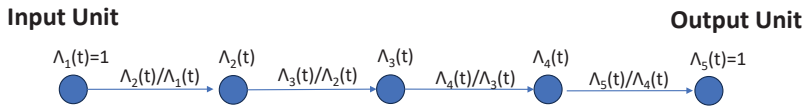

Figure 7: A path with three hidden BiLU units connecting one input unit to one output unit. During the application of the stochastic balancing algorithm, at time t each unit i has a cumulative scaling factor $\Lambda_i(t)$, and each directed edge from unit j to unit i has a scaling factor $M_{ij}(t) = \Lambda_i(t)/\Lambda_j(t)$. The $\lambda_i(t)$ must remain within a finite closed interval away from 0 and infinity. To see this, imagine for instance that there is a subsequence of $\Lambda_3(t)$ that approaches 0. Then there must be a corresponding subsequence of $\Lambda_4(t)$ that approaches 0, or else the contribution of the weight $w_{43}\Lambda_4(t)/\Lambda_3(t)$ to the regularizer would go to infinity. But then, as we reach the output layer, the contribution of the last weight $w_{54}\Lambda_5(t)/\Lambda_4(t)$ to the regularizer goes to infinity because $\Lambda_5(t)$ is fixed to 1 and cannot compensate for the small values of $\Lambda_4(t)$. And similarly, if there is a subsequence of $\Lambda_3(t)$ going to infinity, we obtain a contradiction by propagating its effect towards the input layer.

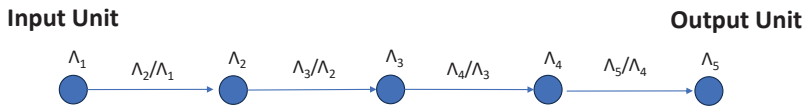

Figure 8: A path with five units. After the stochastic balancing algorithm has converged, each unit i has a scaling factor Λ_i, and each directed edge from unit j to unit i has a scaling factor $M_{ij} = \Lambda_i/\Lambda_j$. The products of the M_{ij}'s along the path is given by: $\frac{\Lambda_2}{\Lambda_1}\frac{\Lambda_3}{\Lambda_2}\frac{\Lambda_4}{\Lambda_3}\frac{\Lambda_5}{\Lambda_4} = \frac{\Lambda_5}{\Lambda_1}$. Accordingly, if we sum the variables $L_{ij} = \log M_{ij}$ along the directed path, we get $L_{21} + L_{32} + L_{43} + L_{54} = \log\Lambda_5 - \log\Lambda_1$. In particular, if unit 1 is an input unit and unit 5 is an output unit, we must have $\Lambda_1 = \Lambda_5 = 1$ and thus: $L_{21} + L_{32} + L_{43} + L_{54} = 0$. Likewise, in the case of a directed cycle where unit 1 and unit 5 are the same, we must have: $L_{21} + L_{32} + L_{43} + L_{54} + L_{15} = 0$.

the variables L_{ij}. From the set of Λ_i's it is easy to construct a unique set of L_{ij}. However what about the converse?

Definition H.2. *A set of real numbers L_{ij}, one per connection of a given neural architecture, is self-consistent if and only if there is a unique corresponding set of numbers $\Lambda_i > 0$ (one per unit) such that: $\Lambda_i = 1$ for all visible units and $L_{ij} = \log\Lambda_i/\Lambda_j$ for every directed connection from a unit j to a unit i.*

Remark H.3. This definition depends on the graph of connections, but not on the original values of the synaptic weights. Every balanced state is associated with a self-consistent set of L_{ij}, but not every self-consistent set of L_{ij} is associated with a balanced state.

Proposition H.4. *A set L_{ij} associated with a neural architecture is self-consistent if and only if $\sum_\pi L_{ij} = 0$ where π is any directed path connecting an input unit to an output unit or any directed cycle (for recurrent networks).*

Remark H.5. Thus the constraints associated with being a self-consistent configuration of L_{ij}' s are all linear. This linear manifold of constraints depends only on the architecture, i.e., the graph of connections. The strictly convex function $R(L_{ij})$ depends on the actual weights W. Different sets of weights W produce different convex functions over the same linear manifold.

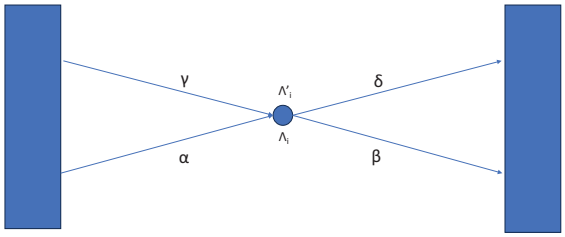

Figure 9: Consider two paths $\alpha + \beta$ and $\gamma + \delta$ from the input layer to the output layer going through the same unit i. Let us assume that the first path assigns a multiplier Λ_i to unit i and the second path assigns a multiplier Λ'_i to the same unit. By assumption we must have: $\sum_\alpha L_{ij} + \sum_\beta L_{ij} = 0$ for the first path, and $\sum_\gamma L_{ij} + \sum_\delta L_{ij} = 0$. But $\alpha + \delta$ and $\gamma + \beta$ are also paths from the input layer to the output layer and therefore: $\sum_\alpha L_{ij} + \sum_\delta L_{ij} = 0$ and $\sum_\gamma L_{ij} + \sum_\beta L_{ij} = 0$. As a result, $\sum_\alpha L_{ij} = \log \Lambda_i = \sum_\gamma L_{ij} = \Lambda'_i$. Therefore the assignment of the multiplier Λ_i must be consistent across different paths going through unit i.

Remark H.6. Note that one could coalesce all the input units and all output units into a single unit, in which case a path from an input unit to and output unit becomes also a directed cycle. In this representation, the constraints are that the sum of the L_{ij} must be zero along any directed cycle. In general, it is not necessary to write a constraint for every path from input units to output units. It is sufficient to select a representative set of paths such that every unit appears in at least one path.

Proof. If we look at any directed path π from unit i to unit j, it is easy to see that we must have:

$$\sum_\pi L_{kl} = \log \Lambda_i - \log \Lambda_j \tag{H.7}$$

This is illustrated in Figures 8 and 1. Thus along any directed path that connects any input unit to any output unit, we must have $\sum_\pi L_{ij} = 0$. In addition, for recurrent neural networks, if π is a directed cycle we must also have: $\sum_\pi L_{ij} = 0$. Thus in short we only need to add linear constraints of the form: $\sum_\pi L_{ij} = 0$. Any unit is situated on a path from an input unit to an output unit. Along that path, it is easy to assign a value Λ_i to each unit by simple propagation starting from the input unit which has a multiplier equal to 1. When the propagation terminates in the output unit, it terminates consistently because the output unit has a multiplier equal to 1 and, by assumption, the sum of the multipliers along the path must be zero. So we can derive scaling values Λ_i from the variables L_{ij}. Finally, we need to show that there are no clashes, i.e. that it is not possible for two different propagation paths to assign different multiplier values to the same unit i. The reason for this is illustrated in Figure 9. □

We can now complete the proof Theorem H.1. Given a neural network of BiLUs with a set of weights W, we can consider the problem of minimizing the regularizer $R(L_{ij}$ over the self-admissible configuration L_{ij}. For any $P > 0$, the L_p regularizer is strictly convex and the space of self-admissible configurations is linear and hence convex. Thus this is a strictly convex optimization problem that has a unique solution (Figure 2). Note that the minimization is carried over self-consistent configurations, which in general are not associated with balanced states. However, the configuration of the weights associated with the optimum set of L_{ij} (point A in Figure 2) must be balanced. To see this, imagine that one of the BiLU units–unit i in the network is not balanced. Then we can balance it using a multiplier λ_i^* and replace Λ_i by $\Lambda'_i = \Lambda_i \lambda^*$. It is easy to check that the new configuration including Λ'_i is self-consistent. Thus, by balancing unit i, we are able to reach a new self-consistent configuration with a lower value of R which contradicts the fact that we are at the global minimum of the strictly convex optimization problem.

We know that the stochastic balancing algorithm always converges to a balanced state. We need to show that it cannot converge to any other balanced state, and in fact that the global optimum is the only balanced state. By contradiction, suppose it converges to a different balanced state associated with the coordinates (L_{ij}^B) (point B in Figure 2). Because of the self-consistency, this point is also associated with a unique set of (Λ_i^B) coordinates. The cost function is continuous and differentiable in both the L_{ij}'s and the Λ_i's coordinates. If we look at the negative gradient of the regularizer, it is non-zero and therefore it must have at least one non-zero component $\partial R/\partial \Lambda_i$ along one of the Λ_i coordinates. This implies that by scaling the corresponding unit i in the network, the regularizer can be further reduced, and by balancing unit i the balancing algorithm will reach a new point (C in Figure 2) with lower regularizer cost. This contradicts the assumption that B was associated with a balanced stated. Thus, given an initial set of weights W, the stochastic balancing algorithm must always converge to the same and unique optimal balanced state W^* associated with the self-consistent point A. A particular stochastic schedule corresponds to a random path within the linear manifold from the origin (at time zero all the multipliers are equal to 1, and therefore $M_{ij} = 1$ and $L_{ij} = 0$) for any i and any j to the unique optimum point A. □

Remark H.7. It should be clear from the proof that the same result holds also for any deterministic balancing schedule, as well as for tied and non-tied subset balancing, e.g., for layer-wise balancing and tied layer-wise balancing. In the Appendix, we provide an analytical solution for the case of tied layer-wise balancing in a layered feed-forward network.

Remark H.8. It should be clear from the proof that the same convergence to the unique global optimum is observed if each neuron, when stochastically visited, is favorably scaled rather than balanced, i.e., it is scaled with a factor that reduces R but not necessarily minimizes R. Stochastic balancing can also be viewed as a form of EM algorithm where the E and M steps can be taken fully or partially.

I Universal Approximation Properties of BiLU Neurons

Here we show that any continuous real-valued function defined over a compact set of the Euclidean space can be approximated to any degree of precision by a network of BiLU neurons with a single hidden layer. As in the case of the similar proof given in Baldi [2021] using linear threshold gates in the hidden layer, it is enough to prove the theorem for a continuous function $f\colon 0, 1 \to \mathbb{R}$.

Theorem I.1. *(Universal Approximation Properties of BiLU Neurons) Let f be any continuous function from $[0, 1]$ to $\mathbb{R}$ and $\epsilon > 0$. Let g_λ be the ReLU activation function with slope $\lambda \in \mathbb{R}$s. Then there exists a feedforward network with a single hidden layer of neurons with ReLU activations of the form g_λ and a single output linear neuron, i.e., with BiLU activation equal to the identity function, capable of approximating f everywhere within ϵ (sup norm).*

Proof. To be clear, $g_\lambda(x) = 0$ for $x < 0$ and $g_\lambda(x) = \lambda x$ for $0 \leq x$. Since f is continuous over a compact set, it is uniformly continuous. Thus there exists $\alpha > 0$ such that for any x_1 and x_2 in the $[0, 1]$ interval:

$$|x_2 - x_1| < \alpha \implies |f(x_2) - f(x_1)| < \epsilon \tag{I.1}$$

Let N be an integer such that $1 < N\alpha$, and let us slice the interval $[0, 1]$ into N consecutive slices of width $h = 1/N$, so that within each slice the function f cannot jump by more than ϵ. Let us connect the input unit to all the hidden units with a weight equal to 1. Let us have N hidden units numbered $1, \ldots, N$ with biases equal to $0, 1/N, 2/N, ..., N_1/N$ respectively and activation function of the form g_{λ_k}. It is essential that different units be allowed to have different slopes λ_k. The input unit is connected to all the hidden units and all the weights on these connections are equal to 1. Thus when x is in the k-th slice, $(k-1)/N \leq x < k/N$, all the units from $k+1$ to N have an output equal to 0, and all the units from 1 to k have an output determined by the corresponding slopes. All the hidden units are connected to the output unit with weights $\beta_1, \ldots, \beta_N$, and β_0 is the bias of the output unit. We want the output unit to be linear. In order for the ϵ approximation to be satisfied, it is sufficient if in the $(k-1)/N \leq x < k/N$ interval, the output is equal to the line joining the point $f((k-1)/N)$ to the point $f(k/N)$. In other words, if $x \in [(k-1)/N, k/N)$, then we want the output of the network to be:

$$\beta_0 + \sum_{i=1}^{k} \beta_i \lambda_i (x - (i-1)h) = f(\frac{k-1}{N}) + \frac{f(\frac{k}{N}) - f(\frac{k-1}{N})}{h}(x - (k-1)h) \tag{I.2}$$

By equating the y-intercept and slope of the lines on the left-hand side and the righ- hand side of Equation I.2, we can solve for the weights β's and the slopes λ's. □

As in the case of the similar proof using linear threshold functions in the hidden layer (see Baldi [2021],) this proof can easily be adapted to continuous functions defined over a compact set of $\mathbb{R}^n$, even with a finite number of finite discontinuities, and into $\mathbb{R}^m$.

J Analytical Solution for the Unique Global Balanced State

Here we directly prove the convergence of stochastic balancing to a unique final balanced state, and derive the equations for the balanced state, in the special case of tied layer balancing (as opposed to single neuron balancing). The Proof and the resulting equations are also valid for stochastic balancing (one neuron at a time) in a layered architecture comprising a single neuron per layer. Let us call tied layer scaling the operation by which all the incoming weights to a given layer of BiLU neurons are multiplied by $\lambda > 0$ and all the outgoing weights of the layer are multiplied by $1/\lambda$, again leaving the training error unchanged. Let us call layer balancing the particular scaling operation corresponding to the value of λ that minimizes the contribution of the layer to the L_2 (or any other L_p) regularizer value. This optimal value of λ^* results in layer-wise balance equations: the sum of the squares of all the incoming weights of the layer must be equal to the sum of the squares of all the outgoing weights of the layer in the L_2 case, and similarly in all L^P cases.

Theorem J.1. *Assume that tied layer balancing is applied iteratively and stochastically to the layers of a layered feedforward network of BiLU neurons. As long as all the layers are visited periodically, this procedure will always converge to the same unique set of weights, which will satisfy the layer-balance equations at all layers, irrespective of the details of the schedule. Furthermore, the balance state can be solved analytically.*

Proof. Every time a layer balancing operation is applied, the training error remains the same, and the L_2 (or any other L_p) regularization error decreases or stays the same. Since the regularization error is always positive, it must converge to a certain value. Using the same arguments as in the proof of Theorem H.1, the weights must also converge to a stable configuration, and since the configuration is stable all its layers must satisfy the layer-wise balance equation. The key remaining question is why is this configuration unique and can we solve it analytically? Let $A_1, A_2, \ldots A_N$ denote the matrices of connections between the layers of the network. Let $\Lambda_1, \Lambda_2, \ldots, \Lambda_{N-1}$ be $N-1$ strictly positive multipliers, representing the limits of the products of the corresponding λ_i^* associated with each balancing step at layer i, as in the proof of Theorem H.1. In this notation, layer 0 is the input layer and layer N is the output layer (with $\Lambda_0 = 1$ and $\Lambda_N = 1$).

After converging, each matrix A_i becomes the matrix $\Lambda_i/\Lambda_{i-1}A_i = M_i A_i$ for $i = 1 \ldots N$, with $M_i = \lambda_i/\Lambda_{i-1}$. The multipliers M_i must minimize the regularizer while satisfying $M_1 \ldots M_N = 1$ to ensure that the training error remains unchanged. In other words, to find the values of the M_i's we must minimize the Lagrangian:

$$\mathcal{L}(M_1, \ldots, M_N) = \sum_{i=1}^{N} ||M_i A_i||^2 + \mu(1 - \prod_{i=1}^{N} M_i) \tag{J.1}$$

written for the L^2 case in terms of the Frobenius norm, but the analysis is similar in the general L_p case. From this, we get the critical equations:

$$\frac{\partial \mathcal{L}}{\partial M_i} = 2M_i||A_i||^2 - \mu M_1 \ldots M_{i-1} M_{i+1} \ldots M_N = 0 \quad \text{for } i = 1, \ldots, N \quad \text{and} \quad \prod_{i=1}^{N} M_i = 1 \tag{J.2}$$

As a resut, for every i:

$$2M_i||A_i||^2 - \frac{\mu}{M_i} = 0 \quad \text{or} \quad \mu = 2M_i^2||A_i||^2 \tag{J.3}$$

Thus each $M_i > 0$ can be expressed in a unique way as a function of the Lagrangian multiplier μ as: $M_i = (\mu/2||A_i||^2)^{1/2}$. By writing again that the product of the $M_i$is equal to 1, we finally get:

$$\mu^N = 2^N \prod_{i=1}^{N} ||A_i||^2 \quad \text{or} \quad \mu = 2 \prod_{i=1}^{N} ||A_i||^{2/N} \tag{J.4}$$

Thus we can solve for M_i:

$$M_i = \frac{\mu}{2||A_i||^2} = \frac{\prod_{i=1}^{N} ||A_i||^{2/N}}{||A_i||^2} \qquad \text{for } \; i = 1, \ldots, N \tag{J.5}$$

Thus, in short, we obtain a unique closed-form expression for each M_i. From there, we infer the unique and final state of the weights, where $A_i^* = M_i A_i = \Lambda_i A_l / \Lambda_{l-1}$. Note that each M_i depends on all the other M_j's, again showcasing how the local balancing algorithm leads to a unique global solution. □

K Computer Resources

The simulations we have described do not require major computing resources. They were all performed using Google Colab and the NVIDIA TESLA T4 GPU that it provides.

L Code Availability

The code for reproducing the simulation results is available under the Apache 2.0 license at: https://anonymous.4open.science/r/a-theory-of-neural-synaptic-balance-00C1

References

P. Baldi. *Deep Learning in Science*. Cambridge University Press, Cambridge, UK, 2021.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*, 31, 2018.

Rachel E Field, James A D'amour, Robin Tremblay, Christoph Miehl, Bernardo Rudy, Julijana Gjorgjieva, and Robert C Froemke. Heterosynaptic plasticity determines the set point for cortical excitatory-inhibitory balance. *Neuron*, 106(5):842–854, 2020.

Robert C Froemke. Plasticity of cortical excitatory-inhibitory balance. *Annual review of neuroscience*, 38:195–219, 2015.

Oliver D Howes and Ekaterina Shatalina. Integrating the neurodevelopmental and dopamine hypotheses of schizophrenia and the role of cortical excitation-inhibition balance. *Biological psychiatry*, 2022.

Dmitry Ivanov, Aleksandr Chezhegov, Mikhail Kiselev, Andrey Grunin, and Denis Larionov. Neuromorphic artificial intelligence systems. *Frontiers in Neuroscience*, 16:1513, 2022.

Yu Ji, YouHui Zhang, ShuangChen Li, Ping Chi, CiHang Jiang, Peng Qu, Yuan Xie, and WenGuang Chen. Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

Dongshin Kim and Jang-Sik Lee. Neurotransmitter-induced excitatory and inhibitory functions in artificial synapses. *Advanced Functional Materials*, 32(21):2200497, 2022.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Faming Liang and Wing Hung Wong. Evolutionary monte carlo: Applications to cp model sampling and change point problem. *STATISTICA SINICA*, 10:317–342, 2000.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Data-dependent path normalization in neural networks. *arXiv preprint arXiv:1511.06747*, 2015.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:294078, 2017.

Farshad Shirani and Hannah Choi. On the physiological and structural contributors to the dynamic balance of excitation and inhibition in local cortical networks. *bioRxiv*, pages 2023–01, 2023.

Martino Sorbaro, Qian Liu, Massimo Bortone, and Sadique Sheik. Optimizing the energy consumption of spiking neural networks for neuromorphic applications. *Frontiers in neuroscience*, 14:662, 2020.

Christopher H Stock, Sarah E Harvey, Samuel A Ocko, and Surya Ganguli. Synaptic balancing: A biologically plausible local learning rule that provably increases neural network noise robustness without sacrificing task performance. *PLOS Computational Biology*, 18(9):e1010418, 2022.

A. Tavakoli, F. Agostinelli, and P. Baldi. SPLASH: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks*, 140:1–12, 2021. Also: arXiv:2006.08947.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have included all the main points of the paper in the abstract.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The majority of our results are theorems backed up by mathematical proofs. We discuss at lenght that balancing improves the value of the regularizer only (it leaves the valuue of the data-dependent component of the error unchanged). We also mention that while it would be interesting to study any kind of balance in biological neural networks, current technnological limirations do not allow recording all the incoming and outgoing synaptic strengths of a neuron.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All the theorems and propositions have clear assumptions and all the proofs are complete and have been checked carefully multiple times. Details of some of the proofs are provided in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have provided all the explanations necessary for reproducing the experimental results in the technical appendix and also provided the code for reproducing our experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have provided an anonymous link to our code which is available in the appendix and also uploaded our code as supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the required details in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are included in all images.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided this information in the computer resources section in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in our paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our paper has no conceivable direct societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The only assets that we have use are the MNIST and CIFAR-10 datasets and we have cited these datasets in the paper properly.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our research does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our research does not involve any human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.