PLOS ONE

RESEARCH ARTICLE

Can neural networks benefit from objectives that encourage iterative convergent computations? A case study of ResNets and object classification

Samuel Lippl[1,2,3]*, **Benjamin Peters**[1,4], **Nikolaus Kriegeskorte**[1,2,5,6]

1 Zuckerman Mind Brain Behavior Institute, Columbia University, New York, NY, United States of America, **2** Department of Neuroscience, Columbia University, New York, NY, United States of America, **3** Center for Theoretical Neuroscience, Columbia University, New York, NY, United States of America, **4** School of Psychology and Neuroscience, University of Glasgow, Glasgow, United Kingdom, **5** Department of Psychology, Columbia University, New York, NY, United States of America, **6** Affiliated member, Electrical Engineering, Columbia University, New York, NY, United States of America

* samuel.lippl@columbia.edu

Citation: Lippl S, Peters B, Kriegeskorte N (2024) Can neural networks benefit from objectives that encourage iterative convergent computations? A case study of ResNets and object classification. PLoS ONE 19(3): e0293440. https://doi.org/10.1371/journal.pone.0293440

Editor: Xiao Luo, University of California Los Angeles, UNITED STATES

Received: October 27, 2023

Accepted: March 5, 2024

Published: March 21, 2024

Data Availability Statement: All relevant data are available from zenodo.org (DOI 10.5281/zenodo.8428514).

Funding: SL was supported by the Gatsby Charitable Foundation under Grant No. GAT3708 (https://www.gatsby.org.uk/) and the Simons Foundation under Grant No. 542939SPI (https://www.simonsfoundation.org/). BP received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 841578

Abstract

Recent work has suggested that feedforward residual neural networks (ResNets) approximate iterative recurrent computations. Iterative computations are useful in many domains, so they might provide good solutions for neural networks to learn. However, principled methods for measuring and manipulating iterative convergence in neural networks remain lacking. Here we address this gap by 1) quantifying the degree to which ResNets learn iterative solutions and 2) introducing a regularization approach that encourages the learning of iterative solutions. Iterative methods are characterized by two properties: iteration and convergence. To quantify these properties, we define three indices of iterative convergence. Consistent with previous work, we show that, even though ResNets can express iterative solutions, they do not learn them when trained conventionally on computer-vision tasks. We then introduce regularizations to encourage iterative convergent computation and test whether this provides a useful inductive bias. To make the networks more iterative, we manipulate the degree of weight sharing across layers using soft gradient coupling. This new method provides a form of recurrence regularization and can interpolate smoothly between an ordinary ResNet and a "recurrent" ResNet (i.e., one that uses identical weights across layers and thus could be physically implemented with a recurrent network computing the successive stages iteratively across time). To make the networks more convergent we impose a Lipschitz constraint on the residual functions using spectral normalization. The three indices of iterative convergence reveal that the gradient coupling and the Lipschitz constraint succeed at making the networks iterative and convergent, respectively. To showcase the practicality of our approach, we study how iterative convergence impacts generalization on standard visual recognition tasks (MNIST, CIFAR-10, CIFAR-100) or challenging recognition tasks with partial occlusions (Digitclutter). We find that iterative convergent computation, in these tasks, does not provide a useful inductive bias for ResNets. Importantly, our approach may be useful for investigating other network architectures and tasks as well

(https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en). NK received funding from the National Science Foundation under Grant No. 1948004 (https://www.nsf.gov/). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

and we hope that our study provides a useful starting point for investigating the broader question of whether iterative convergence can help neural networks in their generalization.

Introduction

An iterative method solves a difficult estimation or optimization problem by starting from an initial guess and repeatedly applying a transformation that is known to improve the estimate, leading to a sequence of estimates that converges to the solution. Iterative methods provide a powerful approach to finding exact or approximate solutions where direct methods fail (e.g., for difficult inverse problems or solutions to systems of equations that are nonlinear and/or large).

Recurrent neural networks (RNNs) iteratively apply the same transformation to their internal representation, suggesting that they may learn algorithms similar to the iterative methods used in mathematics and engineering. The idea of iterative refinement of a representation has also driven recent progress in the context of feedforward networks. New architectures, based on the idea of iterative refinement, have allowed for the training of very deep feedforward models with hundreds of layers. Prominent architectures for achieving high depth are residual [ResNets; 1] and highway networks [2], which use skip connections to drive the network to learn the residual: a pattern of adjustments to the input, thus encouraging the model to learn successive refinements of a representation of the input that is shared across layers.

These architectures combine two ideas. The first is to use skip connections to alleviate the problem of vanishing or exploding gradients [3]. The second is to make these skip connections fixed identity connections, such that the layers learn successive refinements of a shared representational format.

The second idea relates residual and highway networks to RNNs and iterative methods. Learning a single transformation that can be iteratively applied is attractive because it enables trading speed for accuracy by iterating longer [4]. In addition, a preference for an iterative solution may provide a useful inductive bias for certain computational tasks.

Moreover, deep neural networks were originally inspired by neuroscience and have, in turn, contributed to a better understanding of human cognition and, in particular, vision. The primate visual system, in turn, has often been suggested to perform iterative refinement [5, 6]. This principle could therefore allow us to better understand the shared ideas behind deep neural networks and primate vision. Moreover, enhancing ResNets' preference for an iterative solution could potentially make them more similar to the primate visual system. In the reverse direction, it is still unclear under which conditions (or to what extent) primate visual cortex actually implements an approximately convergent computation and investigating this question in deep neural networks could serve as a useful model for investigating it in primates.

However, it is unclear whether ResNets indeed learn solutions akin to iterative methods and, if they do, whether this is a useful inductive bias. The two defining features of iterative methods are (a) iteration and (b) convergence. Here we introduce a set of three indices (Convergence Index, Recurrence Index, and Divergence Index) that can measure the degree to which ResNets implement an iteratively convergent computation. We then analyze to what extent these features emerge in ResNets. In order to investigate whether iterative convergence provides a useful inductive bias, we introduce two simple modifications of classical ResNets and study their impact on a number of datasets. First, we study CIFAR-10, CIFAR-100, and MNIST [7, 8] as examples of classical vision tasks, assessing the networks' performance and sample efficiency. Since iterative and convergent inductive biases may be more useful for tasks

that require some degree of recurrence, we also assess the networks' performance and sample efficiency on several variations of *Digitclutter*, a task which requires the recognition of multiple digits that occlude each other [9].

To study the effect of iteration, we manipulate the degree of weight sharing in ResNets, smoothly interpolating between ordinary and recurrent ResNets. We find that a higher degree of weight sharing tends to make the network more iterative, but does not result in improved performance or sample efficiency. This suggests that in ordinary ResNets, recurrent connections do not provide a useful inductive bias and the networks can harness the additional computational flexibility provided by non-recurrent residual blocks.

Recurrence implies iteration, but not convergence, and so is not sufficient for a network to implement an iterative method as defined above. ResNets, whether they are recurrent (i.e. sharing weights across layers) or not, are therefore neither required nor encouraged to learn a convergent algorithm during training. We demonstrate empirically that ResNets in general do not exhibit convergent behavior and that recurrent ResNets are more convergent than non-recurrent networks. To study the effect of convergence, we upper bound the residual blocks' Lipschitz constant. This modification adversely impacts performance, suggesting that the non-convergent behavior in ordinary ResNets is not merely due to lack of incentive, but underpins the networks' high performance. Across convergent ResNets, a higher degree of weight sharing does not negatively affect performance. This suggests that convergent ResNets, in contrasts to non-convergent ones, do not benefit from the increased computational flexibility of non-recurrent residual blocks.

Taken together, our results suggest that an inductive bias favoring an iterative convergent solution does not outweigh the computational flexibility of non-recurrent residual blocks for the considered tasks. Importantly, our findings do not imply that neural networks cannot benefit from iterative convergence. Rather, this principle may only be useful for a different task and/or network architecture. The methods we have introduced for measuring and manipulating iterative convergence can be helpful for investigating alternative settings as well.

Related work

Prior theoretical work has focused on explaining the success of ResNets [1] and the more general class of highway networks [2] by studying the learning dynamics in ResNets [3, 10, 11] and their interpretation as an ensemble of shallow networks [12, 13], as a discretized dynamical system [14–16], and as performing iterative refinement.

The iterative refinement hypothesis

Our work builds on [17] who argue that the sequential application of the residual blocks in a ResNet iteratively refines a representational estimate. Their work builds on observations that dropping out residual blocks, shuffling their order, or evaluating the last block several times retains reasonable performance [12, 18] and can be used for training [19]. Another set of methods uses such perturbations to train deep neural networks, using stochastic depth [19–21]. Other methods learn to evaluate a limited number of layers that depends on the input [22, 23] or learn increasingly fine-grained object categories across layers [24]. Instead of using perturbations to encourage stability of the trained network, [25] propose a method inspired by dynamical systems theory to guarantee such stability in their model.

Iterative refinement and inverse problems

The iterative refinement hypothesis is particularly important in the context of inverse problems, which are often solved using iterative methods. This is particularly relevant for ResNets

trained for perceptual tasks since perception is often conceptualized as an inverse problem [5, 6]. [26] modeled visual cortex using recurrent neural networks that iteratively infer the latent causes of a hierarchical Bayesian model. Though these networks have been applied to complex datasets [27], most models either learn the inverse model [28, 29] or define an analytically invertible forward model [30, 31]. A notable exception are invertible ResNets [32], whose inverse can be computed through a fixed point iteration. Another set of works starts out with a classical iterative algorithm and unfolds this algorithm to create a deep network, approaching a similar problem from the opposite direction [33–36]. Moreover, [37] refine CNNs to yield an inference algorithm on a specific generative model [recently implemented by [38]].

Recurrent and residual neural networks

The idea of iterative refinement has motivated an increasing number of recurrent convolutional neural networks being applied to computer vision without an explicit implementation of iterative inference [9, 39]. ResNets may be seen as RNNs that have been unfolded for a fixed number of iterations. Sharing weights between the different blocks allows us to train a recurrent residual neural network [40]. In this framework, recurrent residual neural networks may be seen as a special case of residual neural networks where the weights are equal between all blocks. [41, 42] relaxed this constraint by defining the weights of the different blocks as a linear combination of a smaller set of underlying weights. The work by [41] is particularly interesting in this context, as their method yielded a more efficient parameterization of Wide ResNets [43], which generally have more channels, but far fewer layers than the architectures we consider here.

Inductive bias of recurrent operations on visual tasks

The inverse problem perspective on perception suggests that ordinary recognition tasks in computer vision may already benefit from an iteratively convergent inductive bias. For other tasks, we have additional reasons to believe that recurrent processing may be beneficial. For example, object recognition in the primate ventral stream can be well captured by feedforward models [44, 45] but benefits from recurrent processing under challenging conditions. This includes tasks where the presented objects are partially occluded or degraded [46] or tasks that involve perceptual grouping according to local statistical regularities [47] or object-level information [48]. These observations have inspired tasks such as Digitclutter and Digitdebris [9] as well as Pathfinder [49] and cABC [50]. For these datasets, recurrent networks have been shown to outperform corresponding feedforward control models [9, 51].

Implicit recurrent computations

Beyond a potentially useful inductive bias, recurrent neural networks can provide additional computational advantages. If the recurrent operation converges to a fixed point, this fixed point can be determined more efficiently by classical iterative algorithms such as ODE solvers [52]. Moreover, in this case, recurrent backpropagation [53, 54] can compute the parameters' gradients much more efficiently than backpropagation through time [55] because it does not require storing the intermediate activation. Its memory cost is therefore constant in depth. This method has inspired several promising models in computer vision. Deep equilibrium models [56] harness a quasi-Newton method to find the fixed point of their recurrent operation. They have recently been applied to object recognition and image segmentation tasks [57]. These works empirically demonstrated the existence of a fixed point under their parameterization. Other models enforce such a fixed point by the Lipschitz constraints we here employ as well. [25] use an upper bound based on the Frobenius norm, whereas [58] approximate the

Lipschitz constant using a vector-Jacobian product. Our method (see below) is based on [59] and has recently been employed (for different purposes) by [32, 60].

Do ResNets implement iterative computations?

Iterative methods are characterized by two key properties: iteration (i.e., recurrent computation) and convergence. In this section, we examine whether residual neural networks have these properties. We start by showing that ResNets *can express iterative algorithms*, but then demonstrate that ResNets *do not automatically learn iterative algorithms*. We show that this observation extends to large-scale neural networks trained on real data. Finally, we introduce a paradigm that allows us to compare neural networks to iterative methods.

ResNets can represent iterative computations

A popular application for iterative methods is given by inverse problems of the form $x = f(z)$. It is often not possible to directly compute the inverse, f^{-1}, of the forward model, because of analytical or combinatorial intractability. Instead, approximations [61] or iterative error-corrective algorithms [62] are used. Consider the linear forward model $x = f(z) \coloneqq (\alpha_1 z_1, \alpha_2 z_2)$. Though f has an analytical inverse, it serves as an illustrative example. Based on an input x, we can infer the latent variable z using the iterative update

$$\hat{z}^{(0)} \coloneqq x, \quad \hat{z}_i^{(t+1)} \coloneqq \hat{z}_i^{(t)} - s_i \cdot \epsilon_i \cdot (\alpha_i \hat{z}_i^{(t)} - x_i), \quad s_i \coloneqq \mathrm{sign}(\alpha_i),$$

where $\hat{z}^{(T)}$, for some T, is the estimate of z and $\epsilon_i > 0$ should be sufficiently small. In the case of a linear function, any $\epsilon_i \leq 1$ works. The example given in Fig 1 uses $\epsilon_1 = 0.3$, $\epsilon_2 = 0.8$.

This update can be implemented in a small ResNet. Fig 1a contains a schematic illustration of one block of this network. The representation spanned by an encoder in the beginning of the network contains an input representation x and a representation of the current estimate z. In the hidden layer of the residual block, we determine the positive and negative prediction errors and use them to update z:

$$\begin{aligned} p_i^{(t)} &\coloneqq \mathrm{ReLU}(\alpha_i z_i^{(t)} - x_i), \quad n_i^{(t)} \coloneqq \mathrm{ReLU}(x_i - \alpha_i z_i^{(t)}), \\ z_i^{(t+1)} &\coloneqq z_i^{(t)} - s_i \epsilon_i p_i^{(t)} + s_i \epsilon_i n_i^{(t)}. \end{aligned}$$

This recurrent residual building block would implement an iteratively convergent computation in a ResNet. The linear model is, of course, a trivial case, but serves as an illustration of the

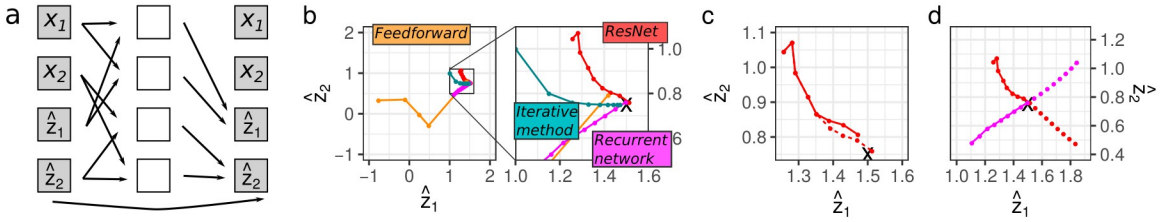

Fig 1. Illustration of iterative computations in ResNets. a A recurrent ResNet implementing a simple error-corrective inverse model. The prediction based on the current estimate $\hat{z}$ is compared to the input x (via positive and negative errors p and n). The error is used to update $\hat{z}$, whereas x is simply retained throughout all blocks. **b** Trajectories of the estimates $\hat{z}_1$, $\hat{z}_2$ across blocks in a feedforward network, iterative steps in an the error-corrective algorithm, residual blocks in a trained ResNet, and residual blocks in a trained recurrent ResNet. All four methods converge to the correct estimate (indicated by black 'x'). **c** Dropping out the fourth block (unbroken line) has a minor impact on the ResNet. **d** If the last block is iteratively applied to the final estimate, the value diverges for both the residual and the recurrent network (broken lines), indicating that they do not learn a convergent structure.

https://doi.org/10.1371/journal.pone.0293440.g001

appeal of this approach. A wide neural network is a flexible function approximator and learning to represent the prediction error instead of the prediction is easier in many cases [24].

ResNets do not automatically learn iterative computations

The fact that ResNets can express iterative computations does not imply that they necessarily learn iterative solutions when trained via backpropagation.

Here we consider the simple example from above to better understand the behavior that may emerge. We highlight three behaviors that distinguish iterative from non-iterative computations and will examine the behavior of large-scale neural networks in the following sections. As an example, we train a conventional ResNet, and a ResNet that uses the same weights in each block (equivalent to a recurrent network) to invert the function $f(z) \coloneqq \left(\frac{3}{2}z_1, \frac{3}{4}z_2\right)$. We contrast their behavior with the iterative error-corrective algorithm outlined above.

Due to the lack of constraints, a non-residual feedforward neural network changes its representation in every layer. As a consequence, the linear decoder at the end of the network is not aligned to the intermediate representation and early readout (as depicted by the orange dots in Fig 1b) leads to a meaningless estimate. In contrast, the skip connections encourage a ResNet to use the same representational format across blocks. This is to say that its intermediate representations are better aligned with the final decoder. Early readout is therefore possible and the representation across blocks will approach the final estimate. As a consequence, the across-block dynamics of the non-residual network are meaningless, whereas the recurrent and residual network's early readouts are close to the final estimate and approach it in a smooth manner, just like the error-corrective algorithm (Fig 1b). Since iterative methods iteratively refine their initial estimate, their behavior is more similar to the ResNets' monotonic convergence.

Aside from their smooth convergence, a fundamental property of iterative methods is their recurrence (i.e., the repeated use of the same computation). This means that dropping out an earlier block has the same effect as dropping out the last one. We can relax the requirement for exact repetition (weight sharing) and require merely similar computations. Fig 1c illustrates that the trained ResNet is indeed relatively robust to block dropout.

Yet the learned models fall short of an iterative method, which is apparent from a third mode of investigation. Iterative convergence would imply that applying the last block's transformation iteratively should keep the readout in the vicinity of the actual estimate. This is clearly not the case in our toy example (Fig 1d). Rather than representing a convergent estimate, this result is more compatible with understanding ResNets as approaching their final estimate at a constant speed and, in the case of late readout, moving past this estimate and overshooting at the same speed. Notably, both the non-recurrent and recurrent networks exhibit this behavior.

Iterative convergence indices

This behavior is not surprising. After all, the network is trained to work for a fixed number of steps and not constrained to stay within the vicinity of its final estimate if more steps are added. However, it reveals that even recurrent ResNets do not automatically learn iterative convergent computations. To assess the extent to which they do learn iterative convergent computations we define three continuous indices, which measure convergence, recurrence, and divergence (defined below).

We evaluate the indices for six instances of ResNet-101, trained on CIFAR-10 [7]. This ResNet consists of three stages of 16 residual blocks with 16, 32, and 64 channels, respectively, using the architecture recommended by [63]. S1 Appendix includes details on the architecture and the training paradigm. The ResNet achieved 5.2% classification error on the validation set.

To characterize the extent to which the ResNets have learned an iterative convergent computation, we introduce three indices measuring different aspects of such computations.

Convergence Index. Viewing ResNets as performing iterative refinement suggests that each stage gradually approaches its final estimate before passing this estimate to the next stage using a downsampling layer. By passing the estimate at each of the residual blocks to the next stage, we can monitor how the stages approach their final estimate across blocks (see Fig 2a, left panel). In accordance with previous results [17], we find that all stages smoothly approach their final estimate, confirming the earlier intuition of a shared representational format. To measure the rate of convergence, we compute the area under the curve (AUC) of the classification error, which we call the Convergence Index. We invert and normalize this value such that a Convergence Index of 0 corresponds chance level read-out at each residual block, whereas a Convergence Index of 1 corresponds to an instant convergence to the final classification error at the first residual block. Fig 2b, left panel, depicts this value for each stage, and averaged across stages.

Note that while the Convergence Index can be used to compare different networks with the same architecture, comparing networks with different architectures (e.g. different numbers of layers) is less straightforward. For example, if a network with twice the number of layers applies the same set of operations in the first half of its layers and then simply applies the identity operation in the latter half, it will have a much smaller Convergence Index. From one perspective, this makes sense, as it converges more quickly. However, from another perspective, the two networks apply the same set of operations and should therefore have the same Convergence Index. A comparison between different network architectures will therefore require committing to one of these two intuitions.

Recurrence Index. To measure the degree of recurrence, we evaluated the effect of dropping out individual blocks on the error rate of the network (see Fig 2a, middle panel). In a non-recurrent ResNet, dropping out earlier blocks may have a stronger effect on the error rate than dropping out the last block. In contrast, in a recurrent ResNet, the effect on error rate is the same for dropping out either earlier blocks or the last block. We therefore computed the difference in error rate observed after these two manipulations. We summarized the behavior by the AUC, which we refer to as Recurrence Index (RI). We invert and normalize this value such that the RI is 0 if dropping out any block leads to an error rate at chance level and the RI

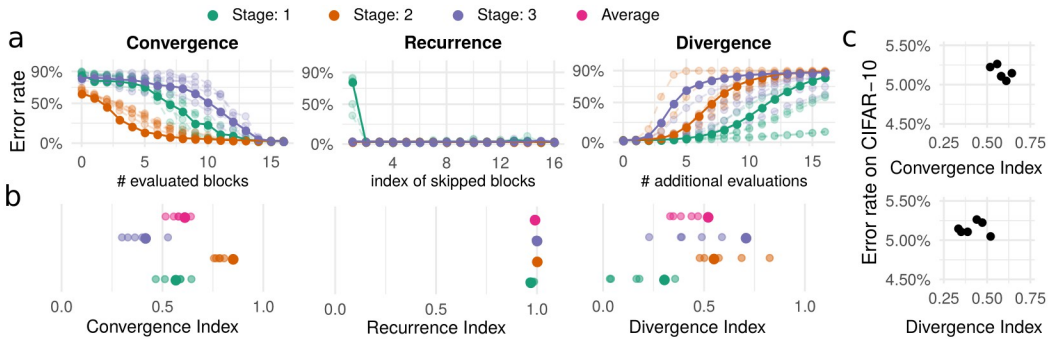

Fig 2. Iterative convergence in ResNets with standard training. a The different perturbation methods (early read-out for determining convergence, dropping-out blocks for determining recurrence, and additional evaluations of the last block for determining divergence) are illustrated for the three stages of the ResNet. The x axis depicts the residual block targeted by the perturbation and the y axis the error rate resulting from the corresponding perturbation (chance performance at 90%). For clarity, one of the six instances is emphasized in the plots. **b** The resulting index values for each stage (small translucent dots) and their averages across instances (large dots). **c** The error rate for the individual network instances is plotted against Convergence and Divergence Index.

https://doi.org/10.1371/journal.pone.0293440.g002

is 1 in the case of a recurrent algorithm. Importantly, a network can be recurrent even if it does not implement an iteratively convergent algorithm. The RI captures this, as it is concerned with the difference in performance between removing different blocks in the network. Put differently, dropping a particular block can have a substantial effect on network performance and still give rise to a high RI, as long as this effect is equally high for early and late blocks. Even though we study non-recurrent ResNets, dropping out any block other than the very first leads to a negligible drop in performance, replicating previous results [12, 17]. As a consequence, the RI, across all stages, is close to 1.0 (see Fig 2b, middle panel).

Divergence Index. ResNets may either converge to their final estimate or simply approach it in a sequence of steps. An iterative algorithm should not be negatively affected by additional applications of the same function. To examine this property, we apply the last block of each stage for an additional up to sixteen steps (see Fig 2a, right panel) and determine the AUC (Divergence Index, DI, see Fig 2b, right panel). We find that no stage is particularly robust to such additional evaluations, though the first stage has the lowest DI, indicating that it is the most robust. This suggests that ResNets approach and move away from their final estimate in a sequence of steps, with their computations bearing little similarity to an iterative convergent algorithm. A high DI does not indicate that the ResNet has failed in some way. After all, it was not trained to be robust to such perturbations. However, it indicates that the ResNet may not implement an iterative convergent computation.

Manipulating convergence and iteration in residual networks

We provided indices measuring convergence, recurrence, and divergence to assess the degree to which a ResNet implements an iterative method. Even though they are able to, ResNets do not necessarily learn to implement a purely iterative method. In particular, they show divergent behavior. Nevertheless, as we have shown above, their behavior does show some similarity to iterative methods and their success has been attributed to these similarities [17, 18]. This suggests that even though the parameterization and optimization does not promote the emergence of an iterative method in a ResNet, a ResNet with iteratively convergent behavior may still have a better inductive bias. To test this hypothesis, we therefore here control the inductive bias, namely recurrence and convergence, of ResNets.

Soft gradient coupling can interpolate smoothly between ordinary and recurrent optimization

We propose a method to blend between recurrent and non-recurrent networks without changing the architecture or the loss landscape. The method is motivated by the observation that we can train a recurrent neural network by sharing the different blocks' gradients [17, 40]. In ordinary ResNets, the residual block t with the weights W_t is changed by following the gradient $\Delta_t = \partial_{W_t} L$, whereas RNNs impose as gradient

$$\Delta = T^{-1} \sum_{t=1}^{T} \partial_{W_t} L, \tag{1}$$

where the weights across residual blocks within a stage must start from the same initialization. The former means that we do not employ any inductive bias towards recurrence, whereas the latter imposes a possibly overly restrictive function space on the architecture. To address both limitations, we propose **soft gradient coupling**, which uses as its update rule

$$\tilde{\Delta}_t = (1 - \lambda)\Delta_t + \lambda\Delta, \quad \lambda \in [0, 1]. \tag{2}$$

For $\lambda < 1$, this retains the entire space of computations enabling both non-recurrent as well as recurrent computations. However, for $\lambda > 0$, the optimization is biased to find more recurrent optima. In contrast to penalty regularizations or strict weight sharing models [41, 42], this does not change the network or loss landscape, but simply the accessibility of different local minima of the loss landscape.

For networks with coupled gradients (i. e., $0 < \lambda \leq 1$), we initialize their weights recurrently (i.e., all residual blocks within one stage share the same initialization). We unshare batch norm statistics as suggested by [17], but leave their parameters (softly) coupled.

Spectral normalization can guarantee convergence in residual networks

Iterative methods preserve a stable output when applied repeatedly. In contrast, the output of the ResNets diverged when the last block was applied repeatedly beyond the number of steps it was trained for. In order to control the degree of convergence in a ResNet, we constrain the Lipschitz constant L of the residual function f. L is defined as the minimal value such that for any input x, y, $\|f(x) - f(y)\| \leq L\|x - y\|$. The smaller L, the more stable f. The Lipschitz constant is hard to determine accurately as it is a global property of f. We therefore determine an upper bound $\hat{L}(f) \geq L$ based on the linear operations within f. The next section will detail how this upper bound is computed. Using this upper bound, we replace f by its **spectral normalization**

$$\tilde{f}(x) \coloneqq f(x)/c, \quad c \coloneqq \max(\mu/\hat{L}(f), 1), \tag{3}$$

for a certain value μ. If $\hat{L}(f) \leq \mu$, the corrective factor c will not change the function. If $\hat{L}(f) > \mu$, it will set the corresponding upper bound of $\tilde{f}$ at $\hat{L}(\tilde{f}) = \mu$, constraining the residual function's Lipschitz constant.

For a given input x, a recurrent residual stage is defined by the iterative application $z_0 \coloneqq x$, $z_t \coloneqq R(z_{t-1})$, where R defines the recurrently applied residual block. We wish to guarantee that z_t eventually converges to a fixed point z_∞ and hope that empirically, z_T, the representation after the specified number of iterations, will be close to this fixed point. According to the Banach fixed point theorem, one way to guarantee such convergence is to require that the residual block's Lipschitz constant L_R be smaller than one.

We can achieve this by replacing the residual connections between adjacent blocks by residual connections between the input x to the stage and each block, i. e. $\tilde{R}(z) \coloneqq x + \tilde{f}(z)$. $\tilde{R}$ has the same Lipschitz constant as $\tilde{f}$. Setting $\mu < 1$, we therefore guarantee that $\tilde{R}$ converges to a fixed point defined by x. We call this network the *properly convergent ResNet* (PCR).

To get a network more similar to an ordinary ResNet, we consider $R(z) \coloneqq z + \tilde{f}(z)$. Though convergence is not guaranteed for the network defined by this residual block, we will demonstrate its empirical convergence. We thus call this network the *improperly convergent ResNet* (ICR).

Upper bound on the Lipschitz constant

To determine an upper bound on the Lipschitz constant of f, we use an alternative characterization based on the Jacobian $J_f(x)$. The *spectral norm* $\|A\|_2$ of a matrix A is defined as its maximal singular value. The Lipschitz constant is then given by

$$L(f) = \max_x \|J_f(x)\|_2.$$

According to the chain rule,

$$J_f(x) = J_{\mathrm{Conv}_2} \cdot J_{\mathrm{ReLU}}(z_2) \cdot J_{\mathrm{BN}_2 \circ \mathrm{Conv}_1} \cdot J_{\mathrm{ReLU}}(z_1) \cdot J_{\mathrm{BN}_1}.$$

Here z_1 and z_2 are given by the representation at the appropriate intermediate stages of the residual function, i. e. z_1 is the representation after BN_1 and z_2 is the representation after BN_2. The Jacobians of the batch normalizations and convolutions do not depend on the input as these are linear operations. Since J_{ReLU} and J_{BN_1} are both diagonal matrices, they commute and therefore,

$$\begin{aligned} J_f(x) &= J_{\mathrm{Conv}_2} \cdot J_{\mathrm{ReLU}}(z_2) \cdot J_{\mathrm{BN}_2 \circ \mathrm{Conv}_1} \cdot J_{\mathrm{BN}_1} \cdot J_{\mathrm{ReLU}}(z_1) \\ &= J_{\mathrm{Conv}_2} \cdot J_{\mathrm{ReLU}}(z_2) \cdot J_{\mathrm{BN}_2 \circ \mathrm{Conv}_1 \circ \mathrm{BN}_1} \cdot J_{\mathrm{ReLU}}(z_1). \end{aligned}$$

We have therefore split the J_f into a product of two constant functions and two Jacobian of the rectified linear unit.

The spectral norm $\|\cdot\|_2$ is known to be sub-multiplicative. This means that for matrices A and B, $\|BA\|_2 \leq \|B\|_2 \|A\|_2$. Moreover, J_{ReLU}, depending on whether its input is positive or negative is given by a diagonal matrix with ones or zeros on the diagonal. Its singular values are therefore at most 1. Putting this together, we can upper bound the Lipschitz constant as

$$\begin{aligned} L(f) = \max_x \|J_f(x)\|_2 &\leq \max_x \|J_{\mathrm{Conv}_2}\|_2 \|J_{\mathrm{ReLU}}(z_2)\|_2 \|J_{\mathrm{BN}_2 \circ \mathrm{Conv}_1 \circ \mathrm{BN}_1}\|_2 \\ &\leq \|J_{\mathrm{Conv}_2}\|_2 \|J_{\mathrm{BN}_2 \circ \mathrm{Conv}_1 \circ \mathrm{BN}_1}\|_2 =: \hat{L}(f). \end{aligned}$$

Theoretically, we could determine the maximal singular value of the convolution using a singular value decomposition. However, the singular value decomposition of such a large matrix is computationally expensive. Instead, Yoshida & Miyato [59] lay out how the maximal singular value can be approximated using a power iteration [64]. The only difference to their method consists in the fact that our first convolution additionally involves multiplying the input by the diagonal matrices given by the two batch normalizations.

Results

To assess our hypotheses, we considered non-recurrently initialized (ordinary) ResNets as well as recurrently initialized ResNets with coupling parameters 0, 0.5, 0.9, and 1. In addition, we considered properly and improperly convergent ResNets across the same coupling parameters, setting $\mu = 0.95$ (we only trained these networks on CIFAR-10). We trained several instances of all these ResNets with 8, 16, 32, and 64 channels in the first stage on classical visual recognition tasks (CIFAR-10, MNIST, and CIFAR-100) as well as *Digitclutter*, a challenging task with partially occluded objects, which has previously been observed to benefit from recurrent connections [9].

Soft gradient coupling improves iterative recurrence indices

As Fig 3a shows, soft gradient coupling indeed improves iterative convergence, increasing the Convergence Index and decreasing the Divergence Index. The Recurrence Index is centered closely around 1 (see S1 Fig). Convergence and Divergence Index, on the other hand, tend to increase and decrease, respectively, both with higher coupling parameters and with a higher number of channels. Notably, this trend appears to not hold up for a fully recurrent ResNet, corresponding to a coupling parameter of 1. These results show that soft gradient coupling is an effective way of manipulating a ResNet's behavior. Increasing weight similarity across blocks leads to more iterative convergence in a ResNet.

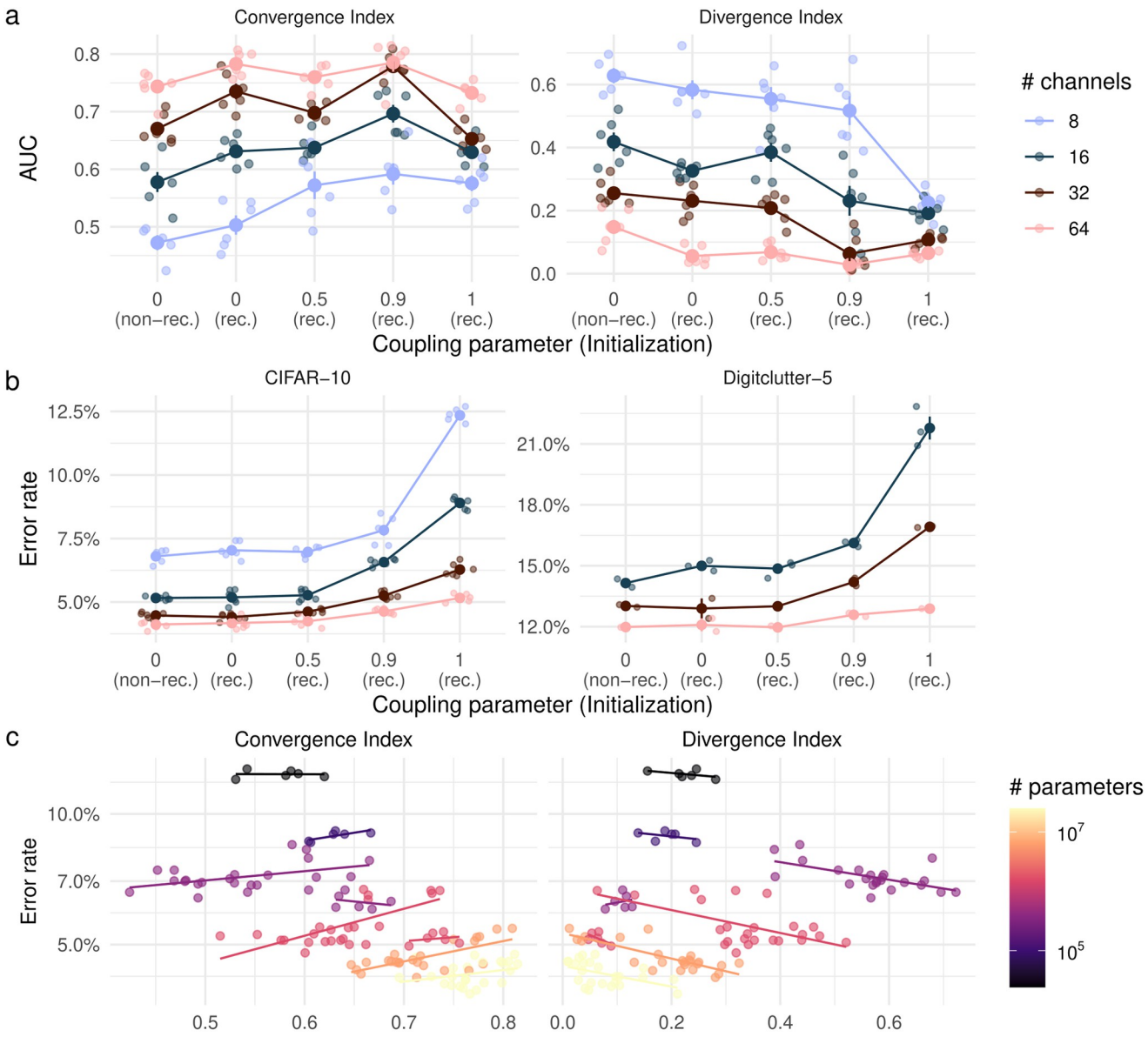

Fig 3. Iterative convergence and performance of coupled ResNets. a Effect of gradient coupling and initialization (rec.: recurrent; non-rec.: non-recurrent) on indices of iterative convergence for architectures with different numbers of channels. **b** Effect of gradient coupling and initialization on the performance on CIFAR-10 and Digitclutter-5. **c** Relationship between performance and iterative convergence, i.e., Convergence (left) and Divergence Index (right). Models with the same number of parameters are visualized by the same color and individual lines. Results in **a**, **c** are on CIFAR-10.

https://doi.org/10.1371/journal.pone.0293440.g003

Moreover, increasing width makes networks converge faster and diverge more slowly. This could potentially be mediated by the fact that the wide networks' increased computational expressivity allows them to move closer to the ultimate target within the first layers (faster convergence). This, in turn, would mean that the last layer would have had less work to do, which could have attenuated detrimental effects of its repeated application (slower divergence).

Spectral normalization makes ResNets convergent

Both the properly and improperly convergent ResNets have a high Convergence Index as well as a Divergence Index at almost zero (see Fig 4a for ICRs and S2 Fig for PCRs). The

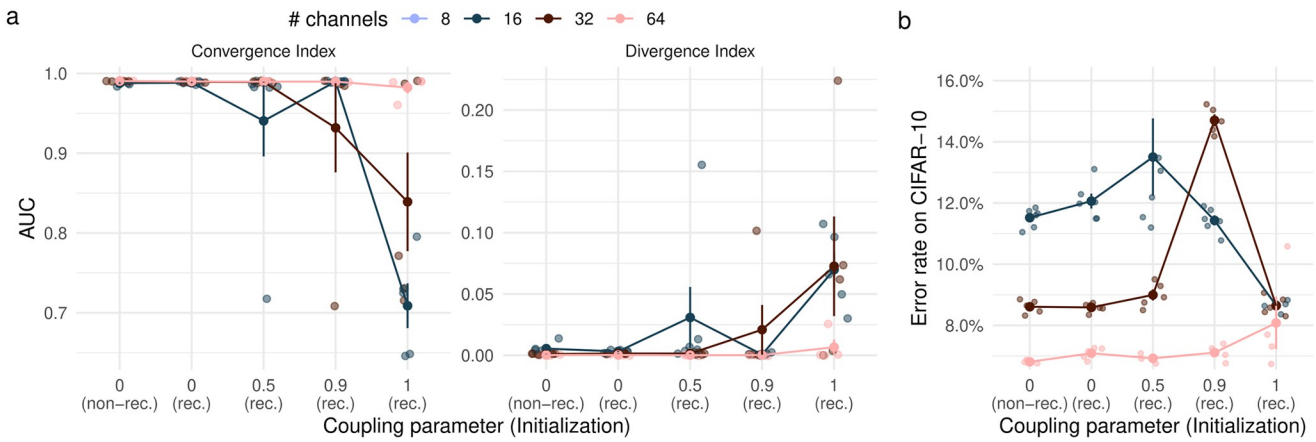

Fig 4. Iterative convergence and performance of improperly convergent ResNets. a Effects of gradient coupling and initialization (rec.: recurrent; non.-rec.: non-recurrent) on iterative convergence indices. **b** Error rates on CIFAR-10 as a function of gradient coupling and initialization.

https://doi.org/10.1371/journal.pone.0293440.g004

Recurrence Index is again centered around one (see S1 Fig). For improperly convergent ResNets with 16 or 32 channels in the first stage, higher coupling parameters generally have a lower Convergence Index and a higher Divergence Index. Nevertheless, these indices indicate that the spectrally normalized ResNets exhibit much more convergent behavior than the ordinary networks. This was only guaranteed for the recurrent, properly convergent ResNets and is therefore an important observation. Perhaps surprisingly, we find that higher recurrence does not necessarily lead to more convergent behavior. This may indicate that less recurrent ResNets use their increased expressivity to more quickly approach their final estimate.

Stronger iterative convergence does not provide a useful inductive bias

We first assessed the effect of gradient coupling on the performance of non-convergent ResNets. As Fig 3b shows, a higher coupling parameter consistently leads to a higher error rate, both for CIFAR-10 and Digitclutter. Additional supporting experiments on CIFAR-100, MNIST, the Digitclutter task, and on sample efficiency can be found in S3 Fig. However, for intermediate coupling parameters of 0.5 and 0.9, this increase in error rate is smaller for networks with higher capacity (i.e., more channels). This effect can also be seen from the relationship between iterative convergence indices and performance. In particular, Fig 3c demonstrates that performance is higher for ResNets with a higher Convergence Index and a lower Divergence Index. This effect, however, is driven by the fact that ResNets with a higher number of channels also show higher measures of iterative convergence (see Fig 3a). When controlling for the number of parameters (see lines in Fig 3c), we find no clear relationship between Convergence and Divergence Index and performance. In part, higher divergence index even seems to lead to higher performance, which would seem to indicate that *less* iterative computations lead to a better inductive bias. This is most likely caused by the fact that we manipulated the divergence index using recurrence regularization. More specifically, higher recurrence regularization seems to lead to a lower divergence index and lower performance.

We then assessed the effect of convergence regularization on performance by training several convergent ResNets on CIFAR-10 (see Fig 4b for ICRs). Convergence regularization led to a higher error rate across all coupling parameters and architectures. A notable exception is the fully coupled ResNet with 16 channels, which performs equally with and without convergence

regularization. This suggests that convergence is not a useful inductive bias in ResNets. Taken together, these experiments suggest that iterative convergence may not provide a useful inductive bias for ResNets.

Discussion

We introduced soft gradient coupling, a new method of recurrence regularization, and demonstrated that this enabled us to manipulate iterative convergence properties in ResNets. To measure these properties, we introduced three indices of iterative convergence, quantifying the effect of perturbations previously introduced in the literature [12, 17].

Iterative methods are considered powerful approaches in particular for solving difficult inverse problems. However, here we did not find iterative convergence to be a useful inductive bias for ResNets. Moreover, we found that higher degrees of weight sharing did not improve a ResNet's parameter efficiency. One reason for this may be that soft gradient coupling or the spectral normalization are the wrong methods for this purpose or require a different optimization strategy. We explored minor variations of soft gradient coupling. More specifically, we selectively coupled only the last eight layers or used a non-uniform kernel in coupling the different layers. These variations did not have an overall effect on our findings (see S1 Appendix). Similarly, future research could explore selectively applying spectral normalization only to the last layers of each stage.

Our findings also suggest, however, that deep feedforward computations should perhaps not be characterized as iterative refinement on a latent representation, but simply as a sequence of operations smoothly approaching their final estimate. Our conclusions are based on experiments on four visual classification tasks. Visual tasks have been proposed to be inverse problems and therefore lend themselves to iterative inference algorithms. Recognition tasks like Digitclutter that involve partial occlusions of objects have in particular been shown to benefit from recurrent computations [9, 46]. However, an iterative method like an error-corrective algorithm that would require a forward model of the data may be more complex and therefore harder to learn than a purely discriminative model. Hence, for the four tested tasks, ResNets may learn direct inference rather than error-corrective inference via a forward model.

The primate visual system has been suggested to solve an inverse problem using iterative refinement [5, 6]. Deep neural networks, in turn, have emerged as the best image-computable models of the ventral visual stream [65–67]. Recently, recurrent neural networks have been shown to predict the temporal dynamics of the ventral stream better than any other computational model [39, 68]. In light of these successes, the fact that the behavior of ResNets is more consistent with direct inference than iterative refinement may highlight an important discrepancy between these models and the primate visual system.

Uncovering the role of iterative computations in the visual system may benefit from more discerning tests of iterative computations. We have focused here on classical object classification (perhaps made harder by occlusions). Performance on this task has been proposed to benefit from iterative computations, but object classification can also be solved without such computations (even with occlusions, in most cases). Moreover, if ResNets implemented iterative computations at all, they would implement ones that are focused on local interactions. Future research may benefit from tasks for which high performance necessitates iterative computations uncovering long-range context. This may be the case, for example, for Pathfinder or cABC.

It is also possible that a different architecture than ResNets would benefit more easily from an iteratively convergent inductive bias. The iterative convergence indices introduced here, as

well as our proposed methods for manipulating the degree of iterative convergence, may be useful for an investigation of alternative tasks and/or networks.

Although it did not improve performance here, soft gradient coupling provides a method for smoothly interpolating between feedforward and recurrent neural networks. More generally, soft gradient coupling provides a simple way to encourage sets of weights to remain similar to each other. This technique may find further use in relaxing weight-sharing constraints and studying the benefit of various forms of weight sharing, including recurrence and convolution, in deep neural networks.

Supporting information

S1 Appendix. The appendix contains details on the used software and the ResNets' architecture and training paradigm. It also details a few variations on gradient coupling that were explored. As detailed in the appendix, the findings on these variations were consistent with the results presented in the main article.
(PDF)

S1 Fig. Recurrence Index for gradient-coupled and improperly convergent ResNets. a Recurrence Index for gradient-coupled ResNets. **b** Recurrence Index for improperly convergent ResNets.
(PDF)

S2 Fig. Results on properly convergent ResNets (PCRs) trained on CIFAR-10. a The indices of iterative convergence demonstrate that the PCRs indeed converge. **b** As the error rate on CIFAR-10 indicates, PCRs tend to perform a bit worse than the improperly convergent ResNets we studied in the main article.
(PDF)

S3 Fig. Additional experiments on gradient-coupled ResNets. a Performance of gradient-coupled ResNets on variations of Digitclutter with a different number of overlapping digits and different size of training data. **b** Performance of gradient-coupled ResNets on CIFAR-100, **c** CIFAR-10 with few training data, and **d** MNIST.
(PDF)

S4 Fig. Performance of training variations on CIFAR-10. a The effect of initializing batch-norm with $\gamma = 0.1$ instead of $\gamma = 1$. **b** The effect of using a triangular kernel for gradient coupling instead of a uniform kernel. **c** A variation of gradient coupling where the first five blocks in each stage were uncoupled.
(PDF)

S5 Fig. The indices of iterative convergence plotted against the coupling parameters for the different training variations.
(PDF)

S1 File. Repository for the model training and analysis. This repository contains the code to train the models, the resulting performance metrics, and code to analyse these metrics.
(ZIP)

Author Contributions

Conceptualization: Samuel Lippl, Benjamin Peters, Nikolaus Kriegeskorte.

Formal analysis: Samuel Lippl.

Funding acquisition: Nikolaus Kriegeskorte.

Investigation: Samuel Lippl, Benjamin Peters.

Methodology: Samuel Lippl, Benjamin Peters.

Resources: Nikolaus Kriegeskorte.

Supervision: Benjamin Peters, Nikolaus Kriegeskorte.

Validation: Benjamin Peters.

Visualization: Samuel Lippl, Nikolaus Kriegeskorte.

Writing – original draft: Samuel Lippl, Benjamin Peters.

Writing – review & editing: Samuel Lippl, Benjamin Peters, Nikolaus Kriegeskorte.

References

1. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770–778.
2. Srivastava RK, Greff K, Schmidhuber J. Training very deep networks. Advances in neural information processing systems. 2015; 28:2377–2385.
3. Hochreiter S. Untersuchungen zu dynamischen neuronalen Netzen; 1991.
4. Spoerer CJ, Kietzmann TC, Mehrer J, Charest I, Kriegeskorte N. Recurrent neural networks can explain flexible trading of speed and accuracy in biological vision. PLOS Computational Biology. 2020; 16 (10):1–27. https://doi.org/10.1371/journal.pcbi.1008215 PMID: 33006992
5. Poggio T, Torre V, Koch C. Computational vision and regularization theory. Nature. 1985; 317 (6035):314–319. https://doi.org/10.1038/317314a0 PMID: 2413361
6. Pizlo Z. Perception viewed as an inverse problem. Vision Research. 2001; 41(24):3145–3161. https://doi.org/10.1016/S0042-6989(01)00173-0 PMID: 11711140
7. Krizhevsky A. Learning Multiple Layers of Features from Tiny Images; 2009.
8. LeCun Y, Cortes C, Burges CJ. MNIST handwritten digit database. 2010;.
9. Spoerer CJ, McClure P, Kriegeskorte N. Recurrent Convolutional Neural Networks: A Better Model of Biological Object Recognition. Frontiers in Psychology. 2017; 8. https://doi.org/10.3389/fpsyg.2017.01551 PMID: 28955272
10. Orhan E, Pitkow X. Skip Connections Eliminate Singularities. In: International Conference on Learning Representations; 2018.
11. Balduzzi D, Frean M, Leary L, Lewis J, Ma KWD, McWilliams B. The Shattered Gradients Problem: If resnets are the answer, then what is the question? In: International Conference on Machine Learning; 2017. p. 342–350.
12. Veit A, Wilber MJ, Belongie S. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R, editors. Advances in Neural Information Processing Systems 29. Curran Associates, Inc.; 2016. p. 550–558. Available from: http://papers.nips.cc/paper/6556-residual-networks-behave-like-ensembles-of-relatively-shallow-networks.pdf.
13. Huang F, Ash J, Langford J, Schapire R. Learning Deep ResNet Blocks Sequentially using Boosting Theory. In: International Conference on Machine Learning; 2018. p. 2058–2067.
14. E W. A Proposal on Machine Learning via Dynamical Systems. Communications in Mathematics and Statistics. 2017; 5(1):1–11. https://doi.org/10.1007/s40304-017-0103-z
15. Haber E, Ruthotto L. Stable architectures for deep neural networks. Inverse Problems. 2017; 34 (1):014004. https://doi.org/10.1088/1361-6420/aa9a90
16. E W, Ma C, Wu L. Barron Spaces and the Compositional Function Spaces for Neural Network Models. arXiv:190608039. 2019;.
17. Jastrzebski S, Arpit D, Ballas N, Verma V, Che T, Bengio Y. Residual Connections Encourage Iterative Inference. In: International Conference on Learning Representations; 2018.
18. Greff K, Srivastava RK, Schmidhuber J. Highway and Residual Networks learn Unrolled Iterative Estimation. In: International Conference on Learning Representations; 2017.
19. Huang G, Sun Y, Liu Z, Sedra D, Weinberger KQ. Deep networks with stochastic depth. In: European conference on computer vision. Springer; 2016. p. 646–661.

20. Hu H, Dey D, Hebert M, Bagnell JA. Learning anytime predictions in neural networks via adaptive loss balancing. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33; 2019. p. 3812–3821.
21. Press O, Smith NA, Levy O. Improving Transformer Models by Reordering their Sublayers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics; 2020. p. 2996–3005. Available from: https://www.aclweb.org/anthology/2020.acl-main.270.
22. Graves A. Adaptive computation time for recurrent neural networks. arXiv:160308983. 2016;.
23. Figurnov M, Collins MD, Zhu Y, Zhang L, Huang J, Vetrov D, et al. Spatially adaptive computation time for residual networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 1039–1048.
24. Zamir AR, Wu TL, Sun L, Shen WB, Shi BE, Malik J, et al. Feedback Networks; 2017. p. 1308–1317. Available from: http://openaccess.thecvf.com/content_cvpr_2017/html/Zamir_Feedback_Networks_CVPR_2017_paper.html.
25. Ciccone M, Gallieri M, Masci J, Osendorfer C, Gomez F. Nais-net: Stable deep networks from non-autonomous differential equations. In: Advances in Neural Information Processing Systems. vol. 31; 2018. p. 3025–3035.
26. Rao RPN, Ballard DH. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. Nature Neuroscience. 1999; 2(1):79–87. https://doi.org/10.1038/4580 PMID: 10195184
27. Wen H, Han K, Shi J, Zhang Y, Culurciello E, Liu Z. Deep Predictive Coding Network for Object Recognition. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR; 2018. p. 5266–5275. Available from: http://proceedings.mlr.press/v80/wen18a.html.
28. Rezende DJ, Mohamed S, Wierstra D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In: International Conference on Machine Learning; 2014. p. 1278–1286.
29. Kingma DP, Welling M. Auto-encoding variational bayes. arXiv:13126114. 2013;.
30. Rezende D, Mohamed S. Variational Inference with Normalizing Flows. In: International Conference on Machine Learning; 2015. p. 1530–1538.
31. Gomez AN, Ren M, Urtasun R, Grosse RB. The reversible residual network: Backpropagation without storing activations. In: Advances in neural information processing systems; 2017. p. 2214–2224.
32. Behrmann J, Grathwohl W, Chen RT, Duvenaud D, Jacobsen JH. Invertible residual networks. In: International Conference on Machine Learning. PMLR; 2019. p. 573–582.
33. Gregor K, LeCun Y. Learning fast approximations of sparse coding. In: International Conference on Machine Learning; 2010. p. 399–406.
34. Hershey JR, Roux JL, Weninger F. Deep unfolding: Model-based inspiration of novel deep architectures. arXiv:14092574. 2014;.
35. Wisdom S, Powers T, Pitton J, Atlas L. Interpretable recurrent neural networks using sequential sparse recovery. arXiv:161107252. 2016;.
36. Wisdom S, Powers T, Pitton J, Atlas L. Deep recurrent NMF for speech separation by unfolding iterative thresholding. In: 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). IEEE; 2017. p. 254–258.
37. Nguyen T, Ho N, Patel A, Anandkumar A, Jordan MI, Baraniuk RG. A Bayesian Perspective of Convolutional Neural Networks through a Deconvolutional Generative Model. arXiv:181102657. 2019;.
38. Huang Y, Gornet J, Dai S, Yu Z, Nguyen T, Tsao DY, et al. Neural networks with recurrent generative feedback. arXiv:200709200. 2020;.
39. Kubilius J, Schrimpf M, Kar K, Rajalingham R, Hong H, Majaj N, et al. Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc.; 2019. p. 12805–12816. Available from: https://proceedings.neurips.cc/paper/2019/file/7813d1590d28a7dd372ad54b5d29d033-Paper.pdf.
40. Liao Q, Poggio T. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv:160403640. 2016;.
41. Savarese P, Maire M. Learning Implicitly Recurrent CNNs Through Parameter Sharing. In: International Conference on Learning Representations; 2019. Available from: https://openreview.net/forum?id=rJgYxn09Fm.
42. Oh J, Wang J, Tang S, Sjoding MW, Wiens J. Relaxed Parameter Sharing: Effectively Modeling Time-Varying Relationships in Clinical Time-Series. In: Machine Learning for Healthcare Conference; 2019. p. 27–52.

43. Zagoruyko S, Komodakis N. Wide Residual Networks. In: Richard C Wilson ERH, Smith WAP, editors. Proceedings of the British Machine Vision Conference (BMVC). BMVA Press; 2016. p. 87.1–87.12. Available from: https://dx.doi.org/10.5244/C.30.87.

44. Hung CP, Kreiman G, Poggio T, DiCarlo JJ. Fast readout of object identity from macaque inferior temporal cortex. Science. 2005; 310(5749):863–866. https://doi.org/10.1126/science.1117593 PMID: 16272124

45. Majaj NJ, Hong H, Solomon EA, DiCarlo JJ. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. Journal of Neuroscience. 2015; 35(39):13402–13418. https://doi.org/10.1523/JNEUROSCI.5181-14.2015 PMID: 26424887

46. Wyatte D, Jilk DJ, O'Reilly RC. Early recurrent feedback facilitates visual object recognition under challenging conditions. Frontiers in psychology. 2014; 5:674. https://doi.org/10.3389/fpsyg.2014.00674 PMID: 25071647

47. Roelfsema PR, Scholte HS, Spekreijse H. Temporal constraints on the grouping of contour segments into spatially extended objects. Vision Research. 1999; 39(8):1509–1529. https://doi.org/10.1016/S0042-6989(98)00222-3 PMID: 10343818

48. Vecera SP, Farah MJ. Is visual image segmentation a bottom-up or an interactive process? Perception & Psychophysics. 1997; 59(8):1280–1296. https://doi.org/10.3758/BF03214214 PMID: 9401461

49. Linsley D, Kim J, Veerabadran V, Windolf C, Serre T. Learning long-range spatial dependencies with horizontal gated recurrent units. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc.; 2018. Available from: https://proceedings.neurips.cc/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf.

50. Kim J, Linsley D, Thakkar K, Serre T. Disentangling neural mechanisms for perceptual grouping. In: International Conference on Learning Representations; 2020.

51. Linsley D, Kim J, Ashok A, Serre T. Recurrent neural circuits for contour detection. In: International Conference on Learning Representations; 2020.

52. Chen RT, Rubanova Y, Bettencourt J, Duvenaud DK. Neural ordinary differential equations. In: Advances in Neural Information Processing Systems. vol. 31; 2018. p. 6571–6583.

53. Pineda FJ. Generalization of back-propagation to recurrent neural networks. Physical review letters. 1987; 59(19):2229. https://doi.org/10.1103/PhysRevLett.59.2229 PMID: 10035458

54. Almeida LB. A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment. In: Artificial Neural Networks: Concept Learning. IEEE Press; 1990. p. 102–111.

55. Werbos PJ. Generalization of backpropagation with application to a recurrent gas market model. Neural networks. 1988; 1(4):339–356. https://doi.org/10.1016/0893-6080(88)90007-X

56. Bai S, Kolter JZ, Koltun V. Deep equilibrium models. In: Advances in Neural Information Processing Systems; 2019. p. 690–701.

57. Bai S, Koltun V, Kolter JZ. Multiscale Deep Equilibrium Models. In: Advances in Neural Information Processing Systems. vol. 33; 2020.

58. Linsley D, Karkada Ashok A, Govindarajan LN, Liu R, Serre T. Stable and expressive recurrent vision models. Advances in Neural Information Processing Systems. 2020; 33.

59. Yoshida Y, Miyato T. Spectral norm regularization for improving the generalizability of deep learning. arXiv:170510941. 2017;.

60. Miyato T, Kataoka T, Koyama M, Yoshida Y. Spectral Normalization for Generative Adversarial Networks. In: International Conference on Learning Representations; 2018.

61. Gershman S, Goodman N. Amortized inference in probabilistic reasoning. In: Proceedings of the annual meeting of the cognitive science society. vol. 36; 2014.

62. Donoho DL. Compressed sensing. IEEE Transactions on information theory. 2006; 52(4):1289–1306. https://doi.org/10.1109/TIT.2006.871582

63. He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual networks. In: European conference on computer vision. Springer; 2016. p. 630–645.

64. von Mises R, Pollaczek-Geiringer H. Praktische Verfahren der Gleichungsauflösung. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik. 1929; 9 (1):58–77. https://doi.org/10.1002/zamm.19290090105

65. Khaligh-Razavi SM, Kriegeskorte N. Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. PLOS Computational Biology. 2014; 10(11):e1003915. https://doi.org/10.1371/journal.pcbi.1003915 PMID: 25375136

66. Cadieu CF, Hong H, Yamins DLK, Pinto N, Ardila D, Solomon EA, et al. Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition. PLOS Computational Biology. 2014; 10(12):e1003963. https://doi.org/10.1371/journal.pcbi.1003963 PMID: 25521294

67. Güçlü U, van Gerven MAJ. Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream. The Journal of Neuroscience. 2015; 35(27):10005. https://doi.org/10.1523/JNEUROSCI.5023-14.2015 PMID: 26157000

68. Kietzmann TC, Spoerer CJ, Sörensen LKA, Cichy RM, Hauk O, Kriegeskorte N. Recurrence is required to capture the representational dynamics of the human visual system. Proceedings of the National Academy of Sciences. 2019; 116(43):21854–21863. https://doi.org/10.1073/pnas.1905544116 PMID: 31591217