How to Train Your Wide Neural Network Without Backprop: An Input-Weight Alignment Perspective

Akhilan Boopathy [1] **Ila Fiete** [1]

Abstract

Recent works have examined theoretical and empirical properties of wide neural networks trained in the Neural Tangent Kernel (NTK) regime. Given that biological neural networks are much wider than their artificial counterparts, we consider NTK regime wide neural networks as a possible model of biological neural networks. Leveraging NTK theory, we show theoretically that gradient descent drives layerwise weight updates that are aligned with their input activity correlations weighted by error, and demonstrate empirically that the result also holds in finite-width wide networks. The alignment result allows us to formulate a family of biologically-motivated, backpropagation-free learning rules that are theoretically equivalent to backpropagation in infinite-width networks. We test these learning rules on benchmark problems in feedforward and recurrent neural networks and demonstrate, in wide networks, comparable performance to backpropagation. The proposed rules are particularly effective in low data regimes, which are common in biological learning settings.

1. Introduction

Deep neural networks trained with gradient descent are surprisingly successful in solving a variety of tasks and exhibit a moderate degree of generalization (Zhang et al., 2017) and transferability (Yosinski et al., 2014). However, it has remained theoretically difficult to understand what aspects of the training data the networks use – and how – to achieve their success. Recent theoretical work has established a number of convergence and generalization results for wide networks under the Neural Tangent Kernel (NTK) training regime because its analytical tractability (Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019; 2020). Wide neural networks are also important models of *biological neural networks*. Indeed, the evidence seems to suggest that neural circuits in the brain are substantially wider and shallower than current deep learning models (M Colonnier, 1981). Thus, wide (artificial) neural networks may more accurately model the brain.

In this work, we leverage the tractability of the NTK regime of wide neural networks together with the suitability of wide neural networks as a model of biological systems to make progress in two directions: First, we investigate the learned representations of wide neural networks through the perspective of *alignment* between the weights of a neural network and statistics of input data. Prior work has shown such alignment effects in the specialized settings of deep linear networks (Saxe et al., 2019) and training with random labels (Maennel et al., 2020); we extend these works by investigating alignment in nonlinear, wide neural networks. Our analysis indicates that even though neural network functions and training dynamics are complex and nonlinear, under an NTK training regime, intermediate layer weights of neural networks trained with gradient descent collect simple layerwise statistics of their inputs, weighted by errors on the output.

Second, the locality of the weights' dependence on the data propagated through the network prompts us to consider whether wide neural networks can be trained with simplified learning rules that reproduce the alignment effect without using backpropagation, the most common implementation of gradient descent in neural networks. Backpropagation is difficult to implement biologically, and alternative learning rules may yield better understanding of how the brain learns (Seung, 2003; Fiete & Seung, 2006; Bengio et al., 2016; Lillicrap et al., 2016; Liao et al., 2016; Nøkland, 2016; Bellec et al., 2020; Bartunov et al., 2018; Richards et al., 2019; Roth et al., 2019). Biologically-plausible learning rules also can have practical advantages including easier implementation on neuromorphic hardware and reduced computational cost. In this work, we focus on data efficiency, which is practically relevant to many low data, real-world tasks. This advantage is also highly relevant to modeling learning in the brain; massive amounts of training data are

[1]Massachusetts Institute of Technology. Correspondence to: Akhilan Boopathy <akhilan@mit.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

often unavailable in biological learning settings.

In this paper, we make the following specific contributions[1]:

- We show theoretically that gradient descent on infinite-width neural networks in the NTK regime produces weights that are aligned with network layers in the following sense: the correlation matrix of the weight change at a particular layer is equal to an *error-weighted* correlation matrix of the layer values at the same layer.
- We propose an *alignment score* to quantify alignment in finite-width networks and empirically demonstrate input-weight alignment in wide, finite-width neural networks trained in the NTK regime.
- We develop a family of backpropagation-free learning rules that are designed to reproduce the input-weight alignment effect. Under the NTK training regime, we prove the equivalence between these learning rules and gradient descent in the infinite-width limit.
- We empirically show that the learning rules achieve comparable performance to gradient descent on wide convolutional neural networks (CNNs) and wide recurrent neural networks (RNNs) in the NTK training regime. The proposed learning rules outperform other backprop-free baseline learning rules including feedback alignment (FA) (Lillicrap et al., 2016) and direct feedback alignment (DFA) (Nøkland, 2016). Moreover, the proposed Align learning rules are especially effective in low data settings, with Align-ada *outperforming* gradient descent in very low data settings.

2. Related Work

Wide neural networks. Recently, randomly initialized infinite-width deep neural networks have been shown to be equivalent to Gaussian processes (de G. Matthews et al., 2018; Lee et al., 2018). Moreover, gradient descent on infinite-width networks can be viewed as a kernel method: gradient descent on infinite width networks is equivalent to kernel regression under the Neural Tangent Kernel (NTK) (Jacot et al., 2018). This insight has been used to train neural networks exactly in the infinite limit (Arora et al., 2019). NTK theory has also been used to show theoretically and empirically that wide *finite-width* neural networks evolve as *linearized* models in terms of their parameters (Lee et al., 2019). (Lee et al., 2020) further shows that the correspondence between finite-width wide networks and infinite width networks is strongest at small learning rates among other results. Networks trained in the NTK regime are also practically effective on low data tasks (Arora et al., 2020).

We highlight that many results in the above works are only applicable under a specific *neural tangent* parameterization, or equivalently under standard parameterization under a specific choice of width-dependent learning rate (Lee et al., 2019), which leads to the NTK infinite-width limit. Recent work has shown that there are an infinite number of infinite-width limits, with certain limits, including the NTK limit, exhibiting an equivalence to kernel regression and other limits allowing for *feature learning* in the infinite-limit (Yang & Hu, 2020). In this work we focus on the NTK limit as it is the most studied and has the most analytical tools available. We leave an extension of our results to other infinite-width limits as future work to be explored.

Biologically-plausible learning. In an effort to develop accurate models of learning in the brain, researchers have developed a number of biologically-plausible learning algorithms for neural networks. These algorithms avoid some of the biologically implausible aspects of backpropagation, the standard algorithm to train artificial neural networks. These aspects include 1) the fact that forward propagation weights must equal backward propagation weights throughout the training process, known as the weight transport problem (Lillicrap et al., 2016), 2) the asymmetry of a nonlinear forward propagation computation and a linear backward propagation computation (Bengio et al., 2016) and 3) the alternation of forward and backward passes through the network and the computation of nonlinear activation function derivatives during the backward pass (Nøkland, 2016).

Sign symmetry (Liao et al., 2016; Xiao et al., 2018) and feedback alignment (FA) (Lillicrap et al., 2016; Song et al., 2021) address weight transport by showing that backward weights need not equal the corresponding forward weights for effective learning. Specifically, sign symmetry imposes *sign-concordance* between the forward and backward weights, relaxing the need for continual transport of the exact forward weights to the backward weights. Removing the need for a biological mechanism to continuously synchronize two different synaptic weights is arguably a major step towards biological plausibility. As we will show, our proposed methods, similar to sign symmetry, avoid continual weight transport but require a correspondence between forward and backward weights at initialization. Building on FA, Direct feedback alignment (DFA) further shows the effectiveness of passing feedback directly to intermediate layers, alleviating the need for a sequential layerwise backward pass (Nøkland, 2016; Launay et al., 2020; Refinetti et al., 2021). Greedy layerwise learning takes a different approach and trains different layers of a neural networks independently, avoiding global feedback altogether (Nøkland & Hiller Eidnes, 2019). Many other learning rules *learn* a feedback pathway, includ-

[1]Our code is released at: https://github.com/FieteLab/Wide-Network-Alignment

ing target propagation-based algorithms (Lee et al., 2015; Meulemans et al., 2020; Ororbia et al., 2018; Ororbia & Mali, 2019; Ororbia et al., 2020a;b; Manchev & Spratling, 2020), weight mirroring (Akrout et al., 2019), predictive coding (Millidge et al., 2020; Ororbia & Kifer, 2022), and eligibility trace-based algorithms for RNNs (Roth et al., 2019; Marschall et al., 2020; Williams & Zipser, 1989; Bellec et al., 2020). Some of these biologically-motivated algorithms can have a number of practical advantages relative to backpropagation including 1) asynchronous updating of weights at different layers of a network, 2) reduced memory costs from having to store intermediate layer activation values, 3) reduced synaptic wiring in the feedback path. The resulting computational efficiencies can be particularly great on neuromorphic hardware, where forward and backward network weights are represented by physically separate wiring on a VLSI circuit.

Currently proposed biologically-motivated algorithms often require tuning feedback weights to be aligned with forward weights to achieve good performance; algorithms entirely avoiding such weight transport have difficulties scaling to larger-scale tasks (Bartunov et al., 2018). In this paper, we propose learning rules that avoid any tuning of feedback weights. However, unlike previous works we prove an equivalence between our learning rules and backpropagation in the limit of infinite-width networks. We believe this provides a strong theoretical reason to expect our learning rules to be as scalable as gradient descent given sufficiently wide networks.

3. Alignment Between Layers and Weights

In this section, we introduce the concept of alignment between the layers and weights of a neural network. We then prove that this alignment occurs in infinite-width networks, showing that weights in infinite width networks capture layerwise statistics of input data. Furthermore, we introduce an *alignment score* metric to quantify alignment in finite width networks. Using this metric, we empirically demonstrate alignment between layers and weights in finite-width networks.

3.1. Input-Weight Alignment

We consider an N-layer neural network $f(x)$ with input x. We denote the activation function as σ and weights and biases W_l and b_l at layer l. Throughout this paper, we use the neural tangent parameterization of weights (Jacot et al., 2018). We denote the pre-activation layer values at layer l as functions $z_l(x)$ of input x. The layer pre-activations are defined as:

$$z_l(x) = \frac{1}{\sqrt{m_{l-1}}} W_l \sigma(z_{l-1}(x)) + b_l \tag{1}$$

for $l = 1, ...N$ where $f(x) = z_N(x)$ and m_{l-1} is the dimensionality of $z_{l-1}(x)$. For notational convenience we denote $\sigma(z_0(x)) = x$. Next, following the settings of (Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019), we assume that the network parameters are trained with continuous time gradient flow of learning rate η on a supervised learning task with loss function $\mathbb{E}_{p_x}[L(f(x), y(x))]$ where $y(x)$ are targets and p_x denotes the finite-supported training distribution over inputs x. We denote the parameter values at time t of training with superscript $^{(t)}$: layer l weights and biases at training time t are denoted $W_l^{(t)}$ and $b_l^{(t)}$. Similarly, we denote the overall network function and intermediate-layer pre-activations at time t as $f^{(t)}(x)$ and $z_l^{(t)}(x)$ respectively.

To define the concept of alignment, it is convenient to define a layerwise *weight-change correlation matrix* $\Delta_l^{(t)}$:

$$\Delta_l^{(t)} = (W_l^{(t)} - W_l^{(0)})^T (W_l^{(t)} - W_l^{(0)}). \tag{2}$$

This matrix represents the correlation between different *columns* of the weight change since initialization, $W_l^{(t)} - W_l^{(0)}$. This matrix captures all the information about $W_l^{(t)} - W_l^{(0)}$ up to rotations of the columns since if $W_l^{(t)} - W_l^{(0)}$ has singular value decomposition USV^T, then $\Delta_l^{(t)} = VS^TSV^T$. Intuitively, the elements of $\Delta_l^{(t)}$ represent the similarity between post-activation units of layer $l-1$ in terms of how they impact the next layer. We focus on weight change correlation instead of weight correlation $W_l^{(t)T} W_l^{(t)}$ because in the NTK training regime, the scale of weight initializations is larger than weight updates (Jacot et al., 2018). Thus, the weight correlation does not significantly change over the course of training, and it is more relevant to consider the weight change correlation.

We also define a layerwise *weighted input activity correlation matrix* $\Sigma_{l,q}^{(t)}$ as the correlation between post-activations under a particular pairwise weighting of inputs $q(x_1, x_2)$:

$$\Sigma_{l,q}^{(t)} = \mathbb{E}_{x_1 \sim p_x, x_2 \sim p_x}[\sigma(z_{l-1}^{(t)}(x_1)) q(x_1, x_2) \sigma(z_{l-1}^{(t)}(x_2))^T] \tag{3}$$

Finally, we say that activities and weights are aligned for a weighting q when the activity correlation matrix is proportional to the weight change correlation matrix:

$$\Sigma_{l,q}^{(t)} = k \Delta_l^{(t)} \tag{4}$$

for some positive constant k. When the weights are aligned with input activities, the weights can be fully specified as a function of the corresponding layer's input correlation matrix up to rotation. In other words, alignment implies that weights capture layer-local statistics of the input data.

Note that our notion of alignment is stronger than the one proposed in (Maennel et al., 2020) since we not only require the eigenvectors of the weight change correlation matrix

and input activity correlation matrix to be equal, but also require the corresponding eigenvalues to be proportional to each other. We also note the following additional differences with (Maennel et al., 2020): 1) we use the weight change correlation matrix instead of the weight correlation matrix, 2) we consider the input correlation matrix instead of the input *covariance* matrix which normalizes layer activations to have mean zero, 3) we compute input correlation matrices over a general pairwise weighting $q(x_1, x_2)$ instead of restricting the weighting to the delta function $q(x_1, x_2) = \delta(x_1, x_2)$.

3.2. Alignment in infinite-width

In the infinite-width limit, layerwise inputs and weights are aligned under a specific weighting $q_l^{(t)}(x_1, x_2)$ that depends only on the gradients of the network at initialization and an *integrated-error* function $\delta^{(t)}(x) \in \mathbb{R}^{m_N}$:

$$\delta^{(t)}(x) = \eta \int_0^t \nabla_{z_N} L(f^{(\tau)}(x), y(x)) d\tau \tag{5}$$

This quantity captures the averaged influence x has on the function $f(x)$ over the course of training. Note that this quantity only depends on the output values of the network and effectively measures the average error incurred by the network outputs on point x across the course of training. We also define a layerwise kernel $\Gamma_l(x_1, x_2) \in \mathbb{R}^{m_N \times m_N}$ as:

$$\Gamma_l(x_1, x_2) = \nabla_{z_l} f^{(0)}(x_1)^T \nabla_{z_l} f^{(0)}(x_2) \tag{6}$$

This quantifies the similarity in the gradients of x_1 and x_2 at layer l. It depends only on the initial configuration of the network. Finally, we define $q_l^{(t)}(x_1, x_2) \in \mathbb{R}$ as:

$$q_l^{(t)}(x_1, x_2) = \delta^{(t)}(x_1)^T \Gamma_l(x_1, x_2) \delta^{(t)}(x_2) \tag{7}$$

Intuitively, pairs of points which influence the network $f(x)$ similarly will be weighted positively and points which have opposite influences on the network will be weighted negatively. Note that this weighting produces an input activity correlation $\Sigma^{(t)}_{l,q_l^{(t)}}$ whose only dependence on the targets $y(x)$ is through the integrated-error $\delta^{(t)}(x)$. Therefore, $\Sigma^{(t)}_{l,q_l^{(t)}}$ is an *error-weighted* correlation matrix of the data. We can then show alignment between input activations and weights in the infinite-width limit:

Proposition 1. *Assume layers are randomly initialized according to the neural tangent initialization (Jacot et al., 2018). Also, assume that the nonlinearity $\sigma(\cdot)$ has bounded first and second derivatives. Suppose the network is trained with continuous gradient flow with learning rate η on the mean-squared error loss for T time such that $\max_x ||y(x) - f^{(t)}(x)||_2$ is uniformly bounded in $t \in [0, T]$. Then, in the simultaneous limit $\lim_{m_{N-1}, m_{N-2}, \ldots m_1 \to \infty}$:*

$$\frac{1}{\sqrt{m_{l-1}}} || \frac{1}{m_{l-1}} \Sigma^{(0)}_{l,q_l^{(t)}} - \Delta_l^{(t)} ||_{op} \in \mathcal{O}\left(\frac{1}{\min\{m_1, \ldots m_{N-1}\}}\right) \tag{8}$$

Proof. See Appendix A for a proof. □

Proposition 1 equates the layerwise input correlation at time 0 with $m_{l-1}\Delta_l^{(t)}$. Conceptually, as layer widths increase, the feedback Jacobian $\nabla_{z_l} f^{(t)}(x)$ and intermediate-layer activations change vanishingly little over the course of training, making the weight change correlation proportional to the input correlation. See Appendix A for a discussion of the width dependent factors in the above expression. The interpretation of the result hinges on the nature of the pairwise weighting $q_l^{(t)}(x_1, x_2)$. As discussed, $q_l^{(t)}(x_1, x_2)$ intuitively captures the similarity in the errors from x_1 and x_2. Thus, Proposition 1 can be interpreted as saying that the weight changes of an infinite-width neural network reflect error-weighted correlation statistics of the inputs from corresponding intermediate layers.

3.3. Alignment in finite-width

Although Proposition 1 applies to infinite-width networks, we show that alignment approximately holds in wide finite-width networks. We empirically quantify alignment through an *alignment score* that measures the similarity between the weight change correlation $\Delta_l^{(t)}$ and the time 0 input correlation $\Sigma^{(0)}_{l,q_l^{(t)}}$. We wish to use a matrix similarity measure that is scale invariant, while being maximized for proportional matrices. Cosine similarity is a natural and commonly used measure with these properties, which we use to define an alignment score S:

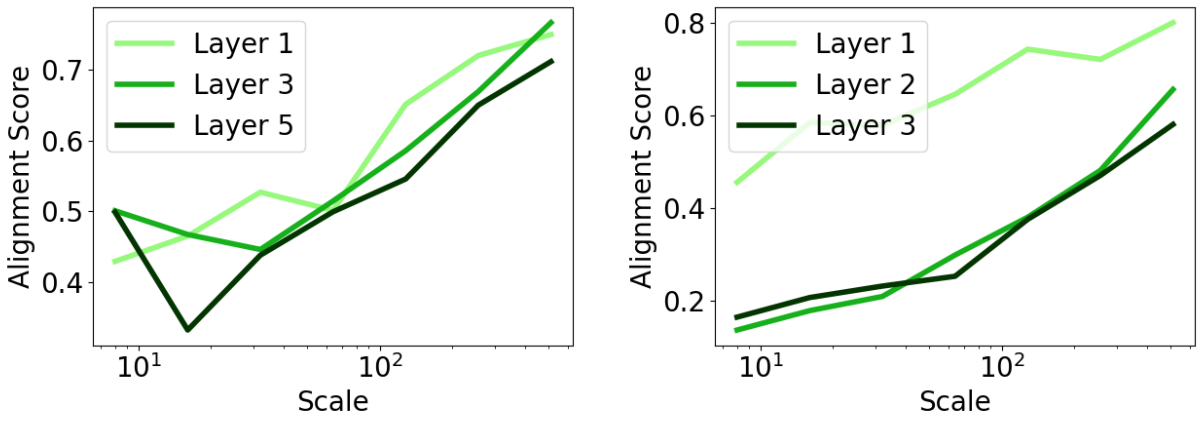
$$S = tr\left(\Delta_l^{(t)} \Sigma^{(0)}_{l,q_l^{(t)}}\right) tr\left(\Delta_l^{(t)2}\right)^{-1/2} tr\left(\Sigma^{(0)2}_{l,q_l^{(t)}}\right)^{-1/2} \tag{9}$$

This score is in the range $[-1, 1]$ with 1 indicating perfect alignment. For wide neural networks, it may not be computationally efficient to operate directly on $\Delta_l^{(t)}$ and $\Sigma^{(0)}_{l,q_l^{(t)}}$ because their sizes scale quadratically with layer width. Instead, the score can be computed in terms of $\Delta_l^{(t)} z$ and $\Sigma^{(0)}_{l,q_l^{(t)}} z$ when z is sampled from a unit Gaussian $N(0, I)$:

$$S = \mathbb{E}_z\left[z^T \Delta_l^{(t)} \Sigma^{(0)}_{l,q_l^{(t)}} z\right] \left(\mathbb{E}_z\left[||\Delta_l^{(t)} z||_2^2\right] \mathbb{E}_z\left[||\Sigma^{(0)}_{l,q_l^{(t)}} z||_2^2\right]\right)^{-1/2} \tag{10}$$

$\Delta_l^{(t)} z$ and $\Sigma^{(0)}_{l,q_l^{(t)}} z$ are computed as a series of matrix-vector products:

$$\Delta_l^{(t)} z = (W_l^{(t)} - W_l^{(0)})^T (W_l^{(t)} - W_l^{(0)}) z \tag{11}$$



(a) CIFAR-10

(b) KMNIST

Figure 1: Alignment scores at different layers of convolutional neural networks with different layer widths trained on the CIFAR-10 (8 layers) and KMNIST (4 layers). The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers.

$$\Sigma^{(0)}_{l,q_l^{(t)}} z = m_{l-1}(W_l^{*(t)} - W_l^{(0)})^T (W_l^{*(t)} - W_l^{(0)}) z \quad (12)$$

where $W_l^{*(t)}$ is defined by the following dynamics, starting from initial condition $W_l^{*(0)} = W_l^{(0)}$:

$$\dot{W}_l^{*(t)} = \frac{\eta}{\sqrt{m_{l-1}}} \times \mathbb{E}_{p_x}\left[\nabla_{z_l} f^{(0)}(x) \nabla_{z_N} L(f^{(t)}(x), y(x)) \sigma(z_{l-1}^{(0)}(x))^T\right]. \quad (13)$$

$W_l^{*(t)}$ is computed by simply training a network with the above learning rule. This alignment computation naturally yields an interpretation of the alignment score as a similarity between the parameters of a network and the corresponding network trained with initialized activations $\sigma(z_{l-1}^{(0)}(x))$ and initialized feedback Jacobian $\nabla_{z_l} f^{(0)}(x)$. We further investigate variants of the learning rule of Equation (13) in Section 4.

Next, we use the alignment scores to evaluate alignment in finite-width networks trained on CIFAR-10 (Krizhevsky, 2009) and KMNIST (Clanuwat et al., 2018), as a function of network width. We train an 8 (CIFAR-10) or 4 (KMNIST) layer ReLU-activated CNN on the mean squared error loss and vary the number of convolutional filters in the intermediate layers from 8 to 512. See Section 5 for further architectural and hyperparameter details. In Figure 1, we find that alignment scores increase with network width, consistent with our prediction of perfect alignment in infinite width. Generally, alignments are lower in later layers of network, suggesting the infinite-width correspondence holds best in early layers. Nevertheless, the results illustrate that in sufficiently wide finite-width networks, intermediate layer weights indeed capture local statistics of the corresponding layers' inputs. See Appendix L Tables 3, 4 for full tabulated results for these networks. In Appendix L Table 5, we also tabulate alignment scores for an 8 layer Tanh-activated CNN trained on CIFAR-10; we find qualitatively similar patterns indicating that the alignment effect empirically holds across activation functions. In Appendix L Figure 8, we plot the evolution of alignment scores during training find that they are initially high and slowly decrease during training as parameters drift from initialization. We also plot baseline alignment scores in networks with randomly permuted weights after training and find low alignment, validating the statistical significance of our alignment scores.

3.4. Discussion

The alignment results demonstrate that the weights of a wide neural network trained by gradient descent can be described by *local* layerwise activation statistics. This is surprising since in general settings, intermediate layer weights can encode *global* information about a network; we show that in the wide network regime, this role of global information is limited. The alignment results directly link trained weights to activation statistics at their corresponding layers, providing greater interpretability into trained networks. To our knowledge, this result is the most general alignment result of its kind found so far. Moreover, as we will demonstrate next, it both conceptually and mathematically motivates simplified learning rules which rely less on coordinated global feedback than backpropagation.

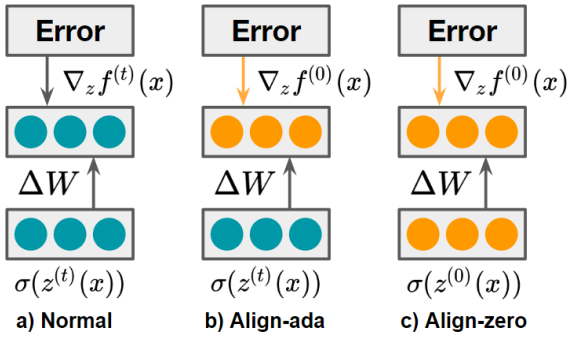


Figure 2: Visual comparison of weight updates ΔW in normal training (via gradient descent) and alignment-based training (Align-ada and Align-zero).

4. Simplified Training Rules Produce Alignment

Prompted by the observation that wide finite-width networks exhibit high input-weight alignment, we propose a family of simplified learning rules designed to produce the alignment effect. We show theoretically that the learning rules are equivalent to gradient descent in the infinite width limit. In Section 5, we perform extensive experiments assessing the performance of these learning rules in finite-width networks. Before considering simplified training rules, we examine the learning rule of gradient descent:

$$\dot{W}_l^{(t)} = \frac{\eta}{\sqrt{m_{l-1}}} \times \mathbb{E}_{p_x}[\nabla_{z_l} f^{(t)}(x) \nabla_{z_N} L\big(f^{(t)}(x), y(x)\big) \sigma(z_{l-1}^{(t)}(x))^T d\tau] \tag{14}$$

For concision, we omit the learning rule for biases which is found by removing the width scaling factor and $\sigma(z_{l-1}^{(t)}(x))^T$. In the infinite-width limit, we expect $\nabla_{z_l} f^{(t)}(x)$ and $\sigma(z_{l-1}^{(t)}(x))$ to remain unchanged over the course of training, which enables input-weight alignment. The simplified learning rules are found by setting both of these quantities to their values at initialization. Specifically, we find the *Align-zero* learning rule by setting both quantities to their initialization values at time 0:

$$\dot{W}_{l,al\cdot 0}^{(t)} = \frac{\eta}{\sqrt{m_{l-1}}} \times \mathbb{E}_{p_x}[\nabla_{z_l} f^{(0)}(x) \nabla_{z_N} L(f_{al\cdot 0}^{(t)}(x), y(x)) \sigma(z_{l-1}^{(0)}(x))^T] \tag{15}$$

where $f_{al\cdot 0}^{(t)}$ represents the network with parameter values $W_{l,al\cdot 0}^{(t)}$. The difference between the rule of Equation (15) and the earlier alignment computation rule of Equation (13) is that the previous rule uses output gradients $\nabla_{z_N} L(f^{(t)}(x), y(x))$ of the original network $f(x)$ trained with gradient descent, while the above rule uses output gradients $\nabla_{z_N} L(f_{al\cdot 0}^{(t)}(x), y(x))$ of the network $f_{al\cdot 0}(x)$ trained with the above learning rule. Unlike the earlier rule, the above learning rule does not require an additional network $f(x)$ trained with gradient descent; it can be used by itself to compute $f_{al\cdot 0}^{(t)}(x)$. Note that Align-zero appears similar to the learning rule of (Lee et al., 2019); see Appendix D for a discussion of the key differences.

Recognizing that network activations might change slightly in wide but finite-width networks, we may alter the Align-zero rule to allow parameter learning to *adapt* to the locally available values $\sigma(z_{l-1}^{(t)}(x))$, instead of fixing them to their values at initialization. This results in the *Align-ada* learning rule:

$$\dot{W}_{l,al\cdot ada}^{(t)} = \frac{\eta}{\sqrt{m_{l-1}}} \times \mathbb{E}_{p_x}[\nabla_{z_l} f^{(0)}(x) \nabla_{z_N} L(f_{al\cdot ada}^{(t)}(x), y(x)) \sigma(z_{l-1}^{(t)}(x))^T] \tag{16}$$

where $f_{al\cdot ada}(x)$ denotes the network with parameters $W_{l,al\cdot ada}^{(t)}$ trained by Align-ada. See Appendix C, Algorithms 1 and 2, for pseudocode and full procedures for training with Align-zero and Align-ada; the difference between the two algorithms is highlighted. Also see Figure 2 for a visual comparison of Align-zero and Align-ada.

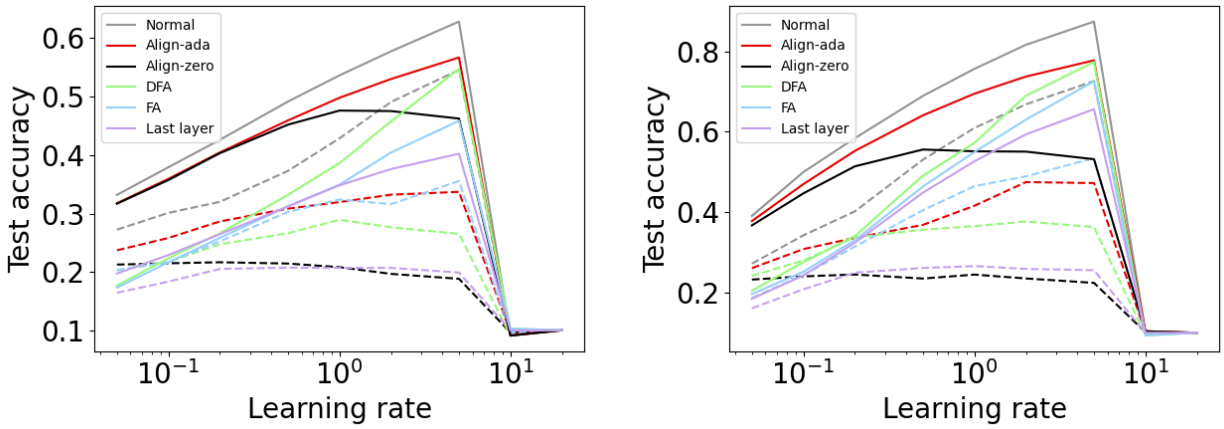
Both Align-zero and Align-ada avoid continual weight transport as they replace the backpropagation Jacobian $\nabla_{z_l} f^{(t)}(x)$ with a fixed feedback Jacobian $\nabla_{z_l} f^{(0)}(x)$. However, the rules still require computing the fixed-over-learning feedback function $\nabla_{z_l} f^{(0)}(x)$, the backpropagation Jacobian at time 0. This may difficult to compute biologically; nevertheless, we believe that such a fixed feedback function would be significantly easier to explain biologically compared to backpropagation which requires constant synchronization or mirroring of the forward and feedback weights. Indeed, previously proposed biologically-motivated learning rules also avoid weight transport, but impose some similarity between forward and backward weights (Liao et al., 2016).

Next, we show that Align-ada and Align-zero are *equivalent* to gradient descent in the limit of infinite-width:

Proposition 2. *Suppose networks of the same initialization are trained with Align-ada, Align-zero and gradient descent under the setting of Proposition 1. Then, in the simultaneous limit* $\lim_{m_{N-1}, m_{N-2}, \ldots m_1 \to \infty}$*:*

$$||f^{(t)}(x) - f_{al\cdot 0}^{(t)}(x)||, \;\; ||f^{(t)}(x) - f_{al\cdot ada}^{(t)}(x)|| \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, \ldots m_{N-1}\}}}) \tag{17}$$

Proof. See Appendix B for a proof. □



(a) CIFAR-10

(b) KMNIST

Figure 3: Test set accuracies of different learning rules vs. learning rate on CNNs trained on CIFAR-10 and KMNIST. Solid lines: 256 conv. filters per layer; dashed: 8 conv. filters.

This result establishes that simplified, backprop-free learning rules designed to produce input-weight alignment are *as effective as gradient descent in the infinite-width limit*. Moreover, given our observation that wide finite-width networks exhibit high input-weight alignment, Proposition 2 suggests that these learning rules might be effective loss-optimizing rules in finite-width networks. Finally, while we use the notation of feedforward networks, our analysis also applies to recurrent neural networks and our learning rules can be adapted to recurrent networks as shown in Appendix E.

Although Proposition 2 shows that the specific methods Align-ada and Align-zero are equivalent to gradient descent in the infinite width limit, we believe our analysis is applicable to a large family of learning rules. As a specific example, in Appendix G, we propose an additional Align variant named *Align-prop* in which the approximation of the backpropagation Jacobian $\nabla_{z_l} f^{(t)}(x)$ is allowed to change over the course of training. Like Align-ada and Align-zero, Align-prop is equivalent to gradient descent in infinite width networks. The presence of multiple rules equivalent to gradient descent in infinite width networks suggests the possibility of many potentially biologically-plausible learning rules with strong theoretical guarantees in wide neural networks.

In the next section, we investigate the empirical effectiveness of Align methods.

5. Experiments on Task Performance

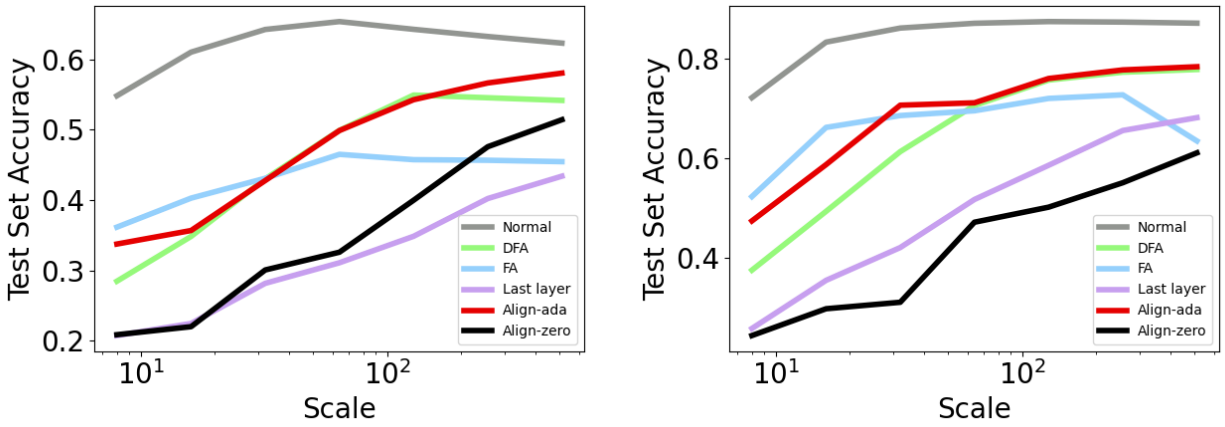
In this section, we train finite-width networks using Align-ada and Align-zero and compare their performance to normal training by gradient descent and other backprop-free baseline learning rules. We assess the efficacy of Align-ada and Align-zero across different network widths and learning rates. See Appendix G for experiments on Align-prop. We conduct experiments on CNNs trained on CIFAR-10 (Krizhevsky, 2009), Small CIFAR-10 (Arora et al., 2020), KMNIST (Clanuwat et al., 2018) and ImageNet (Russakovsky et al., 2015) and RNNs trained on an addition task. Note that our goal with these experiments is not to optimize performance to beat state-of-the-art results on the tasks but rather to demonstrate both the validity of the theoretical work for finite-width networks and to show that Align methods approach the performance of backpropagation while outperforming biologically-motivated baselines. Although we primarily consider the NTK training regime, in Appendix H we present results under standard parameterization. In Appendix I, we conduct additional experiments over multiple random seeds to assess the statistical significance of our results. In Appendix J, we evaluate the training time of Align methods vs. backpropagation.

5.1. Results on Add task in RNNs

We defer results on the Add task to Appendix K; in summary, we find that Align methods perform comparably to gradient descent in wide recurrent neural networks.

5.2. Results on CIFAR-10 and KMNIST

Dataset and baselines We compare various learning methods (gradient descent (denoted Normal), training only the last layer of the network (denoted Last layer), DFA (Nøkland, 2016), FA (Lillicrap et al., 2016)) on CNNs with 7 (CIFAR-10) or 3 (KMNIST) convolutional layers followed by global average pooling and a final fully connected



(a) CIFAR-10 (b) KMNIST

Figure 4: Test set accuracies of different learning rules vs. network scale of CNNs trained on CIFAR-10 and KMNIST.

layer. We defer additional architecture and hyperparameter details and descriptions of baselines to Appendix F.

Varying learning rate In Figure 3, we plot the performance of Align methods vs. baselines over a grid search of learning rates in $[0.05, 20]$ for a narrow and wide network. At small learning rates, the performance of Align-ada, Align-zero and Normal is very close on both CIFAR-10 and KMNIST. This may be because of higher input-weight alignment at lower learning rates, and therefore greater correspondence between the Align learning rules and backprop. The gap between the methods grows modestly with increased learning rates and, as theoretically expected, the gap grows faster in the narrow network. On the wide network, across the entire range of learning rates tested, Align-ada outperforms all non-backprop baselines. All methods have a sharp drop in performance at the same learning rate value of 10, at which point training becomes unstable.

Varying network width In Figure 4, we compare performance of different learning rules under architectures of different widths. While Normal performs well at all network widths, Align-ada performs nearly as well as gradient descent at large widths; Align-zero performs worse, but still significantly improves with network width. Align-zero's relatively worse performance suggests that layers $\sigma(z_{l-1}^{(t)}(x))$ vary considerably more over the course of training than gradients $\nabla_{z_l} f^{(t)}(x)$ in finite width networks. At the largest width tested, Align-ada matches or outperforms all baselines except Normal on both CIFAR-10 and KMNIST. See Appendix L for full tabulated results.

Note on performance While our accuracies are generally low relative to standardly trained networks, they are in similar ranges to prior results for comparably sized networks trained with NTK-based linearized training methods (Lee et al., 2019). This validates that our wide networks indeed are in the NTK regime. It may be possible to improve performance by extending Align methods to non-NTK regimes via, for example, slow timescale weight transport of backward weights; we leave such extensions as a future work.

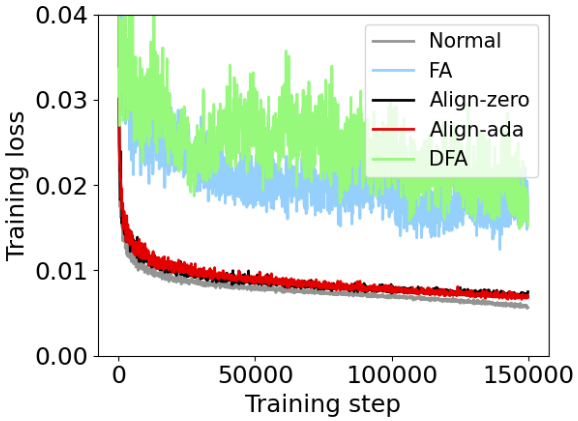


Figure 5: Training loss curves of different learning rules.

5.3. Results on ImageNet

Dataset and baselines We train an 8 layer CNN on ImageNet, comparing Align-ada and Align-zero to Normal, FA and DFA. Additional details on the dataset, architecture, hyperparameters and baselines are included in Appendix F. We aim not to achieve competitive performance on ImageNet,

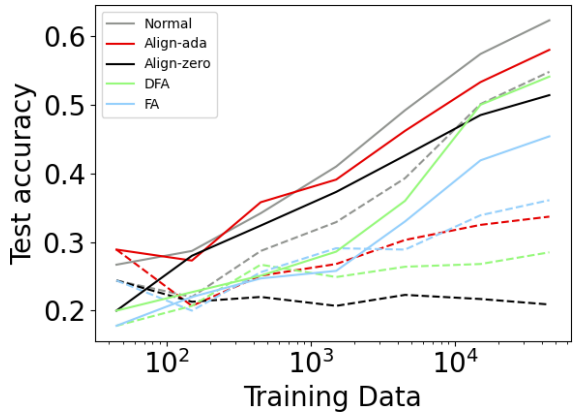


Figure 6: Test set accuracies of different learning rules vs. training set size on CNNs trained on Small CIFAR-10. Solid lines: 512 conv. filters per layer, dashed: 8 conv. filters.

but to illustrate that Align methods can closely match the training dynamics of normal training in the NTK regime even on challenging real-world datasets.

Performance during training In Figure 5, we find that the training losses of Align-ada and Align-zero closely match that of Normal, with a small gap only emerging near 1.5×10^5 training steps. By contrast, DFA and FA do not approach the performance of Normal. This suggests that Align methods may be significantly more scalable than other backprop-alternative learning rules.

5.4. Results in the low-data regime: Small CIFAR-10

Dataset and baselines We construct Small CIFAR-10 as a random subset of points from the full CIFAR-10 training set. Otherwise, we use the same settings as CIFAR-10.

Varying amount of training data In Figure 6, we plot the performance of Align methods vs. baselines when trained on different amounts of CIFAR-10 training data. At low levels of training data, Align methods perform comparably to Normal, with Align-ada consistently outperforming Normal on the lowest data setting. Low data settings may induce high input-weight alignment, and thereby similar learning dynamics between Align rules and backprop; this can allow better relative performance of Align rules. Align rules may also have a regularizing effect, allowing them to outperform backprop in very low data settings. Note that in the low data regime, Align methods perform relatively well even in the narrow network. The results demonstrate the efficacy of Align methods in low data regimes, which may be common in biological learning settings. See Appendix L for full tabulated results.

6. Conclusion

In this paper, we found that weights of infinite width networks and wide finite width networks trained by backprop capture simple layerwise statistics of their inputs weighted by output errors. We used this finding to develop backprop-free learning rules that are equivalent to gradient descent in infinite-width networks. In a variety of empirical settings, the rules are comparable to gradient descent and outperform biologically-motivated baselines, and are especially effective on low data tasks.

References

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. Deep learning without weight transport. 32, 2019.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. *NeurIPS*, 2019.

Arora, S., Du, S. S., Li, Z., Salakhutdinov, R., Wang, R., and Yu, D. Harnessing the power of infinitely wide deep nets on small-data tasks. *ICLR*, 2020.

Bartunov, S., Santoro, A., Richards, B. A., Marris, L., Hinton, G. E., and Lillicrap, T. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *NeurIPS*, 2018.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 2020.

Bengio, Y., Lee, D.-H., Jorg Bornschein, T. M., and Lin, Z. Towards biologically plausible deep learning. *arXiv preprint*, 2016.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature. *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

de G. Matthews, A. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *ICLR*, 2018.

Fiete, I. and Seung, H. Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys Rev Lett*, 97(4):048104, 2006. ISSN 0031-9007 (Print).

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: convergence and generalization in neural networks. *NeurIPS*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Launay, J., Poli, I., Boniface, F., and Krzakala, F. Direct feedback alignment scales to modern deep learning tasks and architectures. *NeurIPS*, 2020.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. *ECML/PKDD*, 2015.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *ICLR*, 2018.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

Lee, J., Schoenholz, S. S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. Finite versus infinite neural networks: an empirical study. *NeurIPS*, 2020.

Liao, Q., Leibo, J. Z., and Poggio, T. How important is weight symmetry in backpropagation? *AAAI*, 2016.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 2016.

M Colonnier, J. O. Number of neurons and synapses in the visual cortex of different species. *Rev Can Biol.*, 1981.

Maennel, H., Alabdulmohsin, I., Tolstikhin, I., Baldock, R. J. N., Bousquet, O., Gelly, S., and Keysers, D. What do neural networks learn when trained with random labels? *arXiv preprint*, 2020.

Manchev, N. and Spratling, M. Target propagation in recurrent neural networks. *Journal of Machine Learning Research*, 21(7):1–33, 2020.

Marschall, O., Cho, K., and Savin, C. A unified framework of online learning algorithms for training recurrent neural networks. *JMLR*, 2020.

Meulemans, A., Carzaniga, F. S., Suykens, J. A. K., Sacramento, J., and Grewe, B. F. A theoretical framework for target propagation. *NeurIPS*, 2020.

Millidge, B., Tschantz, A., and Buckley, C. L. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint*, 2020.

Nøkland, A. and Hiller Eidnes, L. Training neural networks with local error signals. *ICML*, 2019.

Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. *ICLR*, 2019.

Nøkland, A. Direct feedback alignment provides learning in deep neural networks. *NeurIPS*, 2016.

Ororbia, A. and Kifer, D. The neural coding framework for learning generative models. *arXiv preprint*, 2022.

Ororbia, A., Mali, A., Giles, C. L., and Kifer, D. Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):4267–4278, 2020a. doi: 10.1109/TNNLS.2019.2953622.

Ororbia, A., Mali, A., Kifer, D., and Giles, C. L. Large-scale gradient-free deep learning with recursive local representation alignment. *arXiv preprint*, 2020b.

Ororbia, A. G. and Mali, A. Biologically motivated algorithms for propagating local target representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4651–4658, Jul. 2019. doi: 10.1609/aaai.v33i01.33014651.

Ororbia, A. G., Mali, A., Kifer, D., and Giles, C. L. Conducting credit assignment by aligning local representations. *arXiv preprint*, 2018.

Refinetti, M., d'Ascoli, S., Ohana, R., and Goldt, S. Align, then memorise: the dynamics of learning with feedback alignment. *ICML*, 2021.

Richards, B., Lillicrap, T., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R., Berker, A., Ganguli, S., Gillon, C., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G., Miller, K., Naud, R., Pack, C., and Kording, K. A deep learning framework for neuroscience. *Nature Neuroscience*, 22: 1761–1770, 11 2019. doi: 10.1038/s41593-019-0520-2.

Roth, C., Kanitscheider, I., and Fiete, I. Kernel rnn learning (kernl). 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Saxe, A. M., McClelland, J. L., and Ganguli, S. A mathematical theory of semantic development in deep neural networks. *PNAS*, 2019.

Seung, H. S. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073, 2003. ISSN 0896-6273 (Print).

Song, G., Xu, R., and Lafferty, J. Convergence and alignment of gradient descent with random backpropagation weights. *NeurIPS*, 2021.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Xiao, W., Chen, H., Liao, Q., and Poggio, T. Biologically-plausible learning algorithms can scale to large datasets. *arXiv preprint*, 2018.

Yang, G. and Hu, E. J. Feature learning in infinite-width neural networks. *arXiv preprint*, 2020.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? *NeurIPS*, 2014.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Appendix

A. Proof of Proposition 1

We first prove a lemma that will be used in the proofs of both Propositions 1 and 2. The lemma intuitively demonstrates that 1) the weights and activations of infinitely wide neural networks change an infinitesimally small amount during training, 2) thus, the feedback Jacobian $\nabla_\theta f^{(t)}(x)$ changes infinitesimally little during training, 3) the approximations of the feedback Jacobian of Align rules have infinitesimally small approximation error. The proof of Proposition 1 proceeds by quantifying the discrepancy between the actual weight at each layer during training by gradient descent and the weight if the weight updates were computed with a fixed feedback Jacobian $\nabla_\theta f^{(0)}(x)$ instead of $\nabla_\theta f^{(t)}(x)$. As network width grows, this discrepancy is shown to approach 0 at a rate faster than the rate at which weight changes approach 0. Thus, in the infinite width limit, the weight changes behave *as if* the feedback Jacobian were fixed at initialization. This means the weight change correlation matrix can be exactly equated to the time 0 input correlation matrix.

Lemma 1. Assume layers are randomly initialized according to the neural tangent initialization (Jacot et al., 2018). Also, assume that the nonlinearity $\sigma(\cdot)$ has bounded first and second derivatives. Suppose the network $f^{(t)}(x)$ is trained with Align-ada, Align-zero or gradient descent of learning rate η on the mean-squared error loss for T time such that $\max_x ||y(x) - f^{(t)}(x)||_2$ is uniformly bounded in $t \in [0, T]$. Denote θ as the parameters of the network, and for Align-zero and Align-ada, define $J^{(t)}$ as the approximation of $\nabla_\theta f^{(t)}(x)$ such that the corresponding network evolves as:

$$\dot{f}^{(t)}(x) = \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[J^{(t)}(x)(y(x) - f^{(t)}(x))] \tag{18}$$

Note that for Align-zero, $J^{(t)} = \nabla_\theta f^{(0)}(x)$. For gradient descent, define $J^{(t)} = \nabla_\theta f^{(t)}(x)$. For Align-ada, $J^{(t)}$ is defined by first expanding $\nabla_\theta f^{(0)}(x)$ in terms of the weights and activations of the network, then replacing activations $z^{(0)}(x)$ at time 0 with current activations $z^{(t)}(x)$. Then, in the simultaneous limit $\lim_{m_{N-1}, m_{N-2}, ... m_1 \to \infty}$:

$$\forall l = 1, ...N-1, \sqrt{\mathbb{E}[||\frac{1}{\sqrt{m_l}} z_l^{(t)}(x) - \frac{1}{\sqrt{m_l}} z_l^{(0)}(x)||_2^2]} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{19}$$

$$\forall l = 1, ...N, ||\frac{1}{\sqrt{m_{l-1}}} W_l^{(t)} - \frac{1}{\sqrt{m_{l-1}}} W_l^{(0)}||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{20}$$

$$\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{21}$$

$$\max_x ||J^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{22}$$

Proof. We adapt Lemma 1 of (Jacot et al., 2018) to show that weights and intermediate layer activations of networks trained with Align-zero, Align-ada or gradient descent do not change over the course of training in the simultaneous limit $\lim_{m_{N-1}, m_{N-2}, ... m_1 \to \infty}$.

Define $a_l^{(t)} = \sqrt{\mathbb{E}[||\frac{1}{\sqrt{m_l}} z_l^{(t)}(x)||_2^2]}$, $w_l^{(t)} = ||\frac{1}{\sqrt{m_{l-1}}} W_l^{(t)}||_{op}$, $\tilde{a}_l^{(t)} = \sqrt{\mathbb{E}[||\frac{1}{\sqrt{m_l}} z_l^{(t)}(x) - \frac{1}{\sqrt{m_l}} z_l^{(0)}(x)||_2^2]}$, $\tilde{w}_l^{(t)} = ||\frac{1}{\sqrt{m_{l-1}}} W_l^{(t)} - \frac{1}{\sqrt{m_{l-1}}} W_l^{(0)}||_{op}$, where $W_l^{(t)}$ and $z_l^{(t)}(x)$ represent the weights and layers of the network trained with Align-zero, Align-ada or gradient descent. Note that the derivatives of $\tilde{w}_l^{(t)}$ can be bounded as:

$$\dot{\tilde{w}}_l^{(t)} \leq ||\frac{1}{\sqrt{m_{l-1}}} \dot{W}_l^{(t)}||_{op} \leq \frac{1}{\sqrt{m_{l-1}}} \mathcal{P}(a_{l-1}^{(0)}, a_{l-1}^{(t)}, w_{l+1}^{(0)}, w_{l+1}^{(t)}, w_{l+2}^{(0)}, w_{l+2}^{(t)}, ...w_N^{(0)}, w_N^{(t)}) \max_x ||y(x) - f^{(t)}(x)||_2 \tag{23}$$

where $\mathcal{P}$ is a polynomial with bounded positive coefficients not depending on the width of any layers in the network. This is because weight changes at layer l depend on the size of the previous layer (a_{l-1}) and the sizes of weights in later layers through the feedback process ($w_{l+1}, w_{l+2}, ...w_N$). Note that the coefficients of $\mathcal{P}$ can be bounded because $\sigma(\cdot)$ has a bounded first derivative. Similarly, the derivatives of $\tilde{a}_l^{(t)}$ can be bounded as:

$$\dot{\tilde{a}}_l^{(t)} \leq \frac{1}{\sqrt{m_l}} \mathcal{P}(a_0^{(0)}, a_0^{(t)}, ...a_{l-1}^{(0)}, a_{l-1}^{(t)}, w_1^{(0)}, w_1^{(t)}, ...w_N^{(0)}, w_N^{(t)}) \max_x ||y(x) - f^{(t)}(x)||_2 \tag{24}$$

where $\mathcal{P}$ is another polynomial with bounded positive coefficients not depending on the width of any layers in the network. This is because the intermediate layer activations at layer l change as the weights in earlier layers change, which depend on w and a as described above. Next, define $A(t)$ as:

$$\sum_{l=0}^{N-1} a_l^{(0)} + \tilde{a}_l^{(t)} + w_{l+1}^{(0)} + \tilde{w}_{l+1}^{(t)} \tag{25}$$

Observe that using the previous bounds on $\dot{\tilde{w}}_l^{(t)}$ and $\dot{\tilde{a}}_l^{(t)}$, the derivative of $A(t)$ can be bounded as:

$$\frac{d}{dt}A(t) \leq \frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}\mathcal{P}(A(t)) \max_x ||y(x) - f^{(t)}(x)||_2 \tag{26}$$

where $\mathcal{P}$ is a polynomial with coefficients not depending on the width of any layers in the network. Finally, observe that $A(0)$ is stochastically bounded as it is only a function of $a_l^{(0)}$ and $w_l^{(0)}$. By Grönwall's inequality, we find that $A(t)$ can be uniformly bounded in an interval $[0, \tau]$, with τ approaching T as $\min\{m_1, ...m_{N-1}\} \to \infty$. This implies $\mathcal{P}(A(t))$ can be uniformly bounded in this interval. Recall also our assumption that $\max_x ||y(x) - f^{(t)}(x)||_2$ is also uniformly bounded in this interval. Then, due to the $\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}}$ coefficient in the bound for $\frac{d}{dt}A(t)$, in the simultaneous limit $\lim_{m_{N-1},m_{N-2},...m_1\to\infty}$, $\frac{d}{dt}A(t)$ converges uniformly to 0 at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}})$ in any interval $[0, \tau]$ where $\tau < T$. Thus, $A(t) = A(0)$ in this limit. Therefore, in the simultaneous limit $\lim_{m_{N-1},m_{N-2},...m_1\to\infty}$, $\tilde{a}_l^{(t)}$ and $\tilde{w}_l^{(t)}$ converge uniformly to 0 in $[0, T]$ at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}})$.

Next, we prove that in the same limit,

$$\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{27}$$

$$\max_x ||J^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{28}$$

First, observe that for $l = 1, ...N-1$, $\nabla_{z_l} f_i^{(t)}(x)$ can be expanded as:

$$\nabla_{z_l} f_i^{(t)}(x) = diag(\sigma'(z_{l-1}^{(t)}(x)))\frac{1}{\sqrt{m_{l-1}}}W_l^{(t)T}...\frac{1}{\sqrt{m_{N-2}}}W_{N-1}^{(t)T}diag(\sigma'(z_{N-1}^{(t)}(x)))\frac{1}{\sqrt{m_{N-1}}}W_{N,:,i}^{(t)T} \tag{29}$$

Similarly, $\nabla_{z_l} f_i^{(0)}(x)$ can be expanded as:

$$\nabla_{z_l} f_i^{(0)}(x) = diag(\sigma'(z_{l-1}^{(0)}(x)))\frac{1}{\sqrt{m_{l-1}}}W_l^{(0)T}...\frac{1}{\sqrt{m_{N-2}}}W_{N-1}^{(0)T}diag(\sigma'(z_{N-1}^{(0)}(x)))\frac{1}{\sqrt{m_{N-1}}}W_{N,:,i}^{(0)T} \tag{30}$$

Thus, $\nabla_\theta f^{(t)}(x)$ and $J^{(t)}(x)$, which are formed of products of layerwise Jacobians $\nabla_z f(x)$ with intermediate layer activations $\sigma(z(x))$, can be written as sum of products of terms involving $z_l(x)$ and $\frac{1}{\sqrt{m_{l-1}}}W_l$ at time t and time 0. Specifically, since $\nabla_\theta f^{(t)}(x)$ is a function of weights and intermediate layer activations at time t:

$$\begin{aligned}\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \leq &\sum_{l=1}^{N} \tilde{w}_l^{(t)}\mathcal{P}_{w,l}(a_0^{(0)}, a_0^{(t)}, ...a_{N-1}^{(0)}, a_{N-1}^{(t)}, w_1^{(0)}, w_1^{(t)}, ...w_N^{(0)}, w_N^{(t)}) \\ &+ \sum_{l=0}^{N-1} \tilde{a}_l^{(t)}\mathcal{P}_{a,l}(a_0^{(0)}, a_0^{(t)}, ...a_{N-1}^{(0)}, a_{N-1}^{(t)}, w_1^{(0)}, w_1^{(t)}, ...w_N^{(0)}, w_N^{(t)})\end{aligned} \tag{31}$$

where $\mathcal{P}_{w,l}$ and $\mathcal{P}_{a,l}$ are polynomials with positive coefficients not depending on any layer's width. This is because the time-variation of the gradient of any parameter in the network is constrained either by the variation in a weight $\tilde{w}_l^{(t)}$ or

variation in a layer's activations $\tilde{a}_l^{(t)}$. Similarly,

$$\max_x ||J^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \leq \sum_{l=1}^{N} \tilde{w}_l^{(t)} \mathcal{P}_{w,l}(a_0^{(0)}, a_0^{(t)}, ...a_{N-1}^{(0)}, a_{N-1}^{(t)}, w_1^{(0)}, w_1^{(t)}, ...w_N^{(0)}, w_N^{(t)}) + \sum_{l=0}^{N-1} \tilde{a}_l^{(t)} \mathcal{P}_{a,l}(a_0^{(0)}, a_0^{(t)}, ...a_{N-1}^{(0)}, a_{N-1}^{(t)}, w_1^{(0)}, w_1^{(t)}, ...w_N^{(0)}, w_N^{(t)}) \quad (32)$$

because for all three of Align-ada, Align-zero and gradient descent, $J^{(t)}(x)$ is a function of weights and intermediate layer activations at time t or time 0. Note that the left hand size is 0 for Align-zero since $J^{(t)}(x) = \nabla_\theta f^{(0)}(x)$. Using the boundedness of $A(t)$, observe that $\max_x ||J^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op}$ and $\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op}$ converge to 0 at the same rate as $\tilde{W}_l^{(t)}$ and $\tilde{a}_l^{(t)}$. Therefore,

$$\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \quad (33)$$

$$\max_x ||J^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \quad (34)$$

□

Next, we prove Proposition 1:

Proposition 1. Assume layers are randomly initialized according to the neural tangent initialization (Jacot et al., 2018). Also, assume that the nonlinearity $\sigma(\cdot)$ has bounded first and second derivatives. Suppose the network is trained with continuous gradient flow with learning rate η on the mean-squared error loss for T time such that $\max_x ||y(x) - f^{(t)}(x)||_2$ is uniformly bounded in $t \in [0, T]$. Then, in the simultaneous limit $\lim_{m_{N-1}, m_{N-2}, ...m_1 \to \infty}$:

$$\frac{1}{\sqrt{m_{l-1}}} || \frac{1}{m_{l-1}} \Sigma^{(0)}_{l, q_l^{(t)}} - \Delta_l^{(t)} ||_{op} \in \mathcal{O}(\frac{1}{\min\{m_1, ...m_{N-1}\}}) \quad (35)$$

Proof. First, by Lemma 1, observe that in the simultaneous limit $\lim_{m_{N-1}, m_{N-2}, ...m_1 \to \infty}$:

$$\sqrt{\mathbb{E}[|| \frac{1}{\sqrt{m_l}} z_l^{(t)}(x) - \frac{1}{\sqrt{m_l}} z_l^{(0)}(x) ||_2^2]} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \quad (36)$$

$$|| \frac{1}{\sqrt{m_{l-1}}} W_l^{(t)} - \frac{1}{\sqrt{m_{l-1}}} W_l^{(0)} ||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \quad (37)$$

$$\max_x ||\nabla_\theta f^{(t)}(x) - \nabla_\theta f^{(0)}(x)||_{op} \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \quad (38)$$

Since $\nabla_{z_l} f(x)$ is a component of $\nabla_\theta f(x)$, corresponding to bias parameters, this result implies that $\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x)||_{op}]$ also converges to 0 at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}})$.

Next, we define $\tilde{W}^{(t)}$ as:

$$\tilde{W}_l^{(t)} = W_l^{(t)} - W_l^{(0)} - \frac{1}{\sqrt{m_{l-1}}} \eta \mathbb{E}_{p_x}[\nabla_{z_l} f^{(0)}(x) [\int_0^t (y(x) - f^{(\tau)}(x)) d\tau] \sigma(z_{l-1}^{(0)}(x))^T] \quad (39)$$

Note that $\tilde{W}_l^{(0)} = 0$. Taking the derivative of both sides:

$$\begin{aligned}\dot{\tilde{W}}_l^{(t)} &= \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[\nabla_{z_l} f^{(t)}(x)(y(x) - f^{(t)}(x))\sigma(z_{l-1}^{(t)}(x))^T] \\ &\quad - \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[\nabla_{z_l} f^{(0)}(x)(y(x) - f^{(t)}(x))\sigma(z_{l-1}^{(0)}(x))^T] \\ &= \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[\nabla_{z_l} f^{(t)}(x)(y(x) - f^{(t)}(x))(\sigma(z_{l-1}^{(t)}(x)) - \sigma(z_{l-1}^{(0)}(x)))^T] \\ &\quad + \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[(\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x))(y(x) - f^{(t)}(x))\sigma(z_{l-1}^{(0)}(x))^T] \quad (40)\end{aligned}$$

Bounding the norm of $\dot{\tilde{W}}_l^{(t)}$:

$$\begin{aligned}||\dot{\tilde{W}}_l^{(t)}||_{op} &\leq \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x)(y(x) - f^{(t)}(x))(\sigma(z_{l-1}^{(t)}(x)) - \sigma(z_{l-1}^{(0)}(x)))^T||_{op}] \\ &\quad + \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[||(\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x))(y(x) - f^{(t)}(x))\sigma(z_{l-1}^{(0)}(x))^T||_{op}] \\ &\leq \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x)||_{op}||y(x) - f^{(t)}(x)||_2||\sigma(z_{l-1}^{(t)}(x)) - \sigma(z_{l-1}^{(0)}(x))||_2] \\ &\quad + \frac{1}{\sqrt{m_{l-1}}}\eta\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x)||_{op}||y(x) - f^{(t)}(x)||_2||\sigma(z_{l-1}^{(0)}(x))||_2] \\ &\leq c\eta \max_x ||\nabla_{z_l} f^{(t)}(x)||_{op} \max_x ||y(x) - f^{(t)}(x)||_2 \mathbb{E}_{p_x}[||\frac{1}{\sqrt{m_{l-1}}}(z_{l-1}^{(t)}(x) - z_{l-1}^{(0)}(x))||_2] \\ &\quad + \eta\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x)||_{op}] \max_x ||y(x) - f^{(t)}(x)||_2 \max_x ||\frac{1}{\sqrt{m_{l-1}}}\sigma(z_{l-1}^{(0)}(x))||_2 \quad (41)\end{aligned}$$

where c is the Lipschitz constant of σ. Next, observe that $\max_{t\in[0,T]} \max_x ||\nabla_{z_l} f^{(t)}(x)||_{op}$ and $\max_{t\in[0,T]} \max_x ||\frac{1}{\sqrt{m_{l-1}}}\sigma(z_{l-1}^{(0)}(x))||_2$ are bounded in the simultaneous limit $\lim_{m_{N-1},m_{N-2},\ldots m_1\to\infty}$. Moreover, $\mathbb{E}_{p_x}[||\frac{1}{\sqrt{m_{l-1}}}(z_{l-1}^{(t)}(x) - z_{l-1}^{(0)}(x))||_2]$ and $\mathbb{E}_{p_x}[||\nabla_{z_l} f^{(t)}(x) - \nabla_{z_l} f^{(0)}(x)||_{op}]$ approach 0 in this limit at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,\ldots m_{N-1}\}}})$. Since $\max_x ||y(x) - f^{(t)}(x)||_2$ is uniformly bounded in $t \in [0,T]$ by assumption, $\max_{t\in[0,T]} ||\dot{\tilde{W}}_l^{(t)}||_{op}$ also approaches 0 in this limit at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,\ldots m_{N-1}\}}})$. Thus,

$$\begin{aligned}\lim_{m_{N-1},m_{N-2},\ldots m_1\to\infty} ||\tilde{W}_l^{(t)}||_{op} &\leq \lim_{m_{N-1},m_{N-2},\ldots m_1\to\infty} \int_0^t ||\dot{\tilde{W}}_l^{(\tau)}||d\tau \\ &\leq \int_0^t \lim_{m_{N-1},m_{N-2},\ldots m_1\to\infty} ||\dot{\tilde{W}}_l^{(\tau)}||d\tau \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1,\ldots m_{N-1}\}}}) \quad (42)\end{aligned}$$

Next, observe that:

$$\begin{aligned}\Sigma_{l,q_l^{(t)}}^{(0)} &= \mathbb{E}_{x_1\sim p_x, x_2\sim p_x}[\sigma(z_{l-1}^{(0)}(x_1))\delta^{(t)}(x_1)^T\nabla_{z_l} f^{(0)}(x_1)^T\nabla_{z_l} f^{(0)}(x_2)\delta^{(t)}(x_2)\sigma(z_{l-1}^{(0)}(x_2))^T] \\ &= m_{l-1}(W_l^{(t)} - W_l^{(0)} - \tilde{W}_l^{(0)})^T(W_l^{(t)} - W_l^{(0)} - \tilde{W}_l^{(0)}) \quad (43)\end{aligned}$$

This implies:

$$\begin{aligned}&\frac{1}{\sqrt{m_{l-1}}}||\frac{1}{m_{l-1}}\Sigma_{l,q_l^{(t)}}^{(0)} - \Delta_l^{(t)}||_{op} \\ &= \frac{1}{\sqrt{m_{l-1}}}||(W_l^{(t)} - W_l^{(0)} - \tilde{W}_l^{(t)})^T(W_l^{(t)} - W_l^{(0)} - \tilde{W}_l^{(t)}) - (W_l^{(t)} - W_l^{(0)})^T(W_l^{(t)} - W_l^{(0)})||_{op} \\ &\leq 2\frac{1}{\sqrt{m_{l-1}}}||(W_l^{(t)} - W_l^{(0)})||_{op}||\tilde{W}_l^{(t)}||_{op} + \frac{1}{\sqrt{m_{l-1}}}||\tilde{W}_l^{(t)T}\tilde{W}_l^{(0)}||_{op} \quad (44)\end{aligned}$$

Finally, note that $\frac{1}{\sqrt{m_{l-1}}}||(W_l^{(t)} - W_l^{(0)})||_{op}$ converges to 0 at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}})$. Combining this with the convergence of $||\tilde{W}_l^{(t)}||_{op}$ to 0, we find that $\frac{1}{\sqrt{m_{l-1}}}||\frac{1}{m_{l-1}}\Sigma^{(0)}_{l,q_l^{(t)}} - \Delta_l^{(t)}||_{op}$ converges to 0 at rate $\mathcal{O}(\frac{1}{\min\{m_1,...m_{N-1}\}})$. □

Interpretation of the width scaling in Proposition 1 Observe that as network width approaches infinity, both weights and intermediate layer activations change vanishingly over the course of training. Thus, the corresponding correlation matrices $\Delta_l^{(t)}$ and $\Sigma^{(0)}_{l,q_l^{(t)}}$ also approach zero under the appropriate width-dependent normalizations. In light of this, it is important to consider whether Proposition 1 really shows an alignment effect between $\Delta_l^{(t)}$ and $\Sigma^{(0)}_{l,q_l^{(t)}}$ or merely holds because the correlation matrices individually approach zero.

Note that since $||W_l^{(t)} - W_l^{(0)}||_{op}$ grows as $o(\sqrt{m_{l-1}})$, the weight change correlation $||\Delta_l^{(t)}||_{op}$ grows as $o(m_{l-1})$. Thus, $\frac{1}{\sqrt{m_{l-1}}}||\Delta_l^{(t)}||_{op}$ grows as $o(\sqrt{m_{l-1}})$, which will generally *not* converge to zero. Similarly, activated layer norms $||\sigma(z_{l-1}^{(0)}(x))||_2$ grow as $O(m_{l-1})$, implying $||\Sigma^{(0)}_{l,q_l^{(t)}}||_{op}$ grows as $O(m_{l-1}^2)$. Thus, $\frac{1}{\sqrt{m_{l-1}}}||\frac{1}{m_{l-1}}\Sigma^{(0)}_{l,q_l^{(t)}}||_{op}$ grows as $O(\sqrt{m_{l-1}})$, which also will not converge to zero in general. Thus, neither term in the Proposition 1 expression individually approaches zero, and the result does not trivially follow from the individual terms separately approaching zero. Therefore, the convergence of the norm difference in Proposition 1 demonstrates that the two correlation matrices point in the same direction away from the origin and indeed can be interpreted as an alignment effect.

B. Proof of Proposition 2

The proof of Proposition 2 starts by quantifying the difference between the output of an Align-trained network and a backprop-trained network during training. This difference is shown to be bounded by the discrepancy between the feedback Jacobian $\nabla_\theta f^{(t)}(x)$ of the backprop network, its true value in the Align-trained network, and its approximation in the Align-trained network. By Lemma 1, these discrepancies approach 0 in infinite-width networks; thus, Align rules approach gradient descent in the infinite width limit.

Proposition 2 Suppose networks $f^{(t)}(x)$, $f^{(t)}_{al\cdot 0}(x)$ and $f^{(t)}_{al\cdot ada}(x)$ are trained with gradient descent, Align-zero and Align-ada respectively and share the same initialization. Assume layers are randomly initialized according to the neural tangent initialization (Jacot et al., 2018). Also, assume that the nonlinearity $\sigma(\cdot)$ has bounded first and second derivatives. Suppose all methods use learning rate η on the mean-squared error loss for T time such that $\max_x ||y(x) - f^{(t)}(x)||_2$ is uniformly bounded in $t \in [0, T]$. Then, in the simultaneous limit $\lim_{m_{N-1},m_{N-2},...m_1\to\infty}$:

$$||f^{(t)}(x) - f^{(t)}_{al\cdot 0}(x)||,\ ||f^{(t)}(x) - f^{(t)}_{al\cdot ada}(x)|| \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1, ...m_{N-1}\}}}) \tag{45}$$

Proof. First, note that $f^{(t)}(x)$ evolves as:

$$\dot{f}^{(t)}(x) = \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[\nabla_\theta f^{(t)}(x)(y(x) - f^{(t)}(x))] \tag{46}$$

where θ are the parameters of $f^{(t)}$. The term in expectation corresponds to the gradient of the total loss (over all points) with respect to the network parameters θ, while $\nabla_\theta f^{(t)}(x)^T$ indicates how parameter changes affect the value of $f^{(t)}$ evaluated at point x.

We consider a general alternative learning procedure that trains a network $f_*^{(t)}$ which replaces the feedback Jacobian $\nabla_\theta f^{(t)}(x)$ with a general function $J_*^{(t)}(x)$. $f_*^{(t)}$ evolves as:

$$\dot{f}_*^{(t)}(x) = \eta \nabla_\theta f_*^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(y(x) - f_*^{(t)}(x))] \tag{47}$$

Note that here, parameter changes, corresponding to the term in expectation, depend on $J_*^{(t)}(x)$ and not on the true feedback Jacobian $\nabla_\theta f_*^{(t)}(x)$. Thus, in general, the above rule avoids exact backpropagation. However, parameter changes affect the network output according to the true feedback Jacobian $\nabla_\theta f_*^{(t)}(x)$.

Define $\Delta_1^{(t)}(x) = \nabla_\theta f_*^{(t)}(x) - \nabla_\theta f^{(t)}(x)$ and $\Delta_2^{(t)}(x) = J_*^{(t)}(x) - \nabla_\theta f^{(t)}(x)$. Next, finding the difference of $\dot{f}^{(t)}(x)$ and $\dot{f}_*^{(t)}(x)$:

$$\begin{aligned}
\dot{f}_*^{(t)}(x) - \dot{f}^{(t)}(x) &= \eta \nabla_\theta f_*^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(y(x) - f_*^{(t)}(x))] - \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[\nabla_\theta f^{(t)}(x)(y(x) - f^{(t)}(x))] \\
&= \eta \nabla_\theta f_*^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(y(x) - f^{(t)}(x))] - \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[\nabla_\theta f^{(t)}(x)(y(x) - f^{(t)}(x))] \\
&\quad + \eta \nabla_\theta f_*^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(f^{(t)}(x) - f_*^{(t)}(x))] \\
&= \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(y(x) - f^{(t)}(x))] - \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[\nabla_\theta f^{(t)}(x)(y(x) - f^{(t)}(x))] \\
&\quad + \eta \Delta_1^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(y(x) - f^{(t)}(x))] + \eta \nabla_\theta f_*^{(t)}(x)^T \mathbb{E}[J_*^{(t)}(x)(f^{(t)}(x) - f_*^{(t)}(x))] \\
&= \eta \Delta_1^{(t)}(x)^T \mathbb{E}[(\nabla_\theta f^{(t)}(x) + \Delta_2^{(t)}(x))(y(x) - f^{(t)}(x))] + \eta \nabla_\theta f^{(t)}(x)^T \mathbb{E}[\Delta_2^{(t)}(x)(y(x) - f^{(t)}(x))] \\
&\quad + \eta (\nabla_\theta f^{(t)}(x) + \Delta_1^{(t)}(x))^T \mathbb{E}[(\nabla_\theta f^{(t)}(x) + \Delta_2^{(t)}(x))(f^{(t)}(x) - f_*^{(t)}(x))] \quad (48)
\end{aligned}$$

Bounding the derivative of the discrepancy between $f^{(t)}(x)$ and $f_*^{(t)}(x)$:

$$\begin{aligned}
\frac{d}{dt}||f_*^{(t)}(x) - f^{(t)}(x)||_2 &\leq ||\dot{f}_*^{(t)}(x) - \dot{f}^{(t)}(x)||_2 \\
&\leq \eta \max_x ||\Delta_1^{(t)}(x)||_{op} (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_2^{(t)}(x)||_{op}) \max_x ||y(x) - f^{(t)}(x)||_2 \\
&\quad + \eta \max_x ||\nabla_\theta f^{(t)}(x)||_{op} \max_x ||\Delta_2^{(t)}(x)||_{op} \max_x ||y(x) - f^{(t)}(x)||_2 \\
&+ \eta (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_1^{(t)}(x)||_{op}) (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_2^{(t)}(x)||_{op}) \max_x ||f^{(t)}(x) - f_*^{(t)}(x)||_2
\end{aligned} \quad (49)$$

Taking the integral of both sides:

$$\begin{aligned}
&||f_*^{(t)}(x) - f^{(t)}(x)||_2 \\
&\leq \int_0^t \eta \max_x ||\Delta_1^{(\tau)}(x)||_{op} (\max_x ||\nabla_\theta f^{(\tau)}(x)||_{op} + \max_x ||\Delta_2^{(\tau)}(x)||_{op}) \max_x ||y(x) - f^{(\tau)}(x)||_2 d\tau \\
&\quad + \int_0^t \eta \max_x ||\nabla_\theta f^{(\tau)}(x)||_{op} \max_x ||\Delta_2^{(\tau)}(x)||_{op} \max_x ||y(x) - f^{(\tau)}(x)||_2 d\tau \\
&+ \eta (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_1^{(t)}(x)||_{op}) (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_2^{(t)}(x)||_{op}) \max_x ||f^{(t)}(x) - f_*^{(t)}(x)||_2
\end{aligned} \quad (50)$$

Define

$$\begin{aligned}
\alpha(t) &= \int_0^t \eta \max_x ||\Delta_1^{(\tau)}(x)||_{op} (\max_x ||\nabla_\theta f^{(\tau)}(x)||_{op} + \max_x ||\Delta_2^{(\tau)}(x)||_{op}) \max_x ||y(x) - f^{(\tau)}(x)||_2 d\tau \\
&\quad + \int_0^t \eta \max_x ||\nabla_\theta f^{(\tau)}(x)||_{op} \max_x ||\Delta_2^{(\tau)}(x)||_{op} \max_x ||y(x) - f^{(\tau)}(x)||_2 d\tau
\end{aligned} \quad (51)$$

$$\beta(t) = \eta (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_1^{(t)}(x)||_{op}) (\max_x ||\nabla_\theta f^{(t)}(x)||_{op} + \max_x ||\Delta_2^{(t)}(x)||_{op}) \quad (52)$$

$$u(t) = \max_x ||f_*^{(t)}(x) - f^{(t)}(x)||_2 \quad (53)$$

Then, the above inequality implies:

$$u(t) \leq \alpha(t) + \int_0^t \beta(\tau) u(\tau) d\tau \quad (54)$$

By Grönwall's inequality:

$$u(t) \leq \alpha(t) \exp\left(\int_0^t \beta(\tau) d\tau \right) \quad (55)$$

Finally, observe that if in the simultaneous limit $\lim_{m_{N-1},m_{N-2},...m_1\to\infty}$, $\max_x ||\Delta_1^{(t)}(x)||_{op}$ and $\max_x ||\Delta_2^{(t)}(x)||_{op}$ converge uniformly to 0 in range $[0,T]$, then, in the same limit, $\alpha(t)$ converges to 0 at the same rate implying $u(t)$ converges to 0 at the same rate.

Finally, observe that by Lemma 1, $\max_x ||\Delta_1^{(t)}(x)||_{op}$ and $\max_x ||\Delta_2^{(t)}(x)||_{op}$ converge to 0 at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}})$. Thus, $u(t)$ converges to 0 at rate $\mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}})$, yielding the result:

$$||f^{(t)}(x) - f_{al\cdot 0}^{(t)}(x)||,\ ||f^{(t)}(x) - f_{al\cdot ada}^{(t)}(x)|| \in \mathcal{O}(\frac{1}{\sqrt{\min\{m_1,...m_{N-1}\}}}) \tag{56}$$

□

C. Pseudocode for Align-zero and Align-ada

Algorithm 1 Align-zero training

Require: Training points $\mathcal{X}$, training steps T, layer widths $m_1, m_2, ...m_L$, learning rate η

Initialize weights $W_1^{(0)}, b_1^{(0)}, ...W_N^{(0)}, b_N^{(0)}$
for Mini-batch $x \in \mathcal{X}$ **do**
  for $l = 1, ...N$ **do**
    $z_l^{(0)}(x) = \frac{1}{\sqrt{m_{l-1}}} W_l^{(0)} \sigma(z_{l-1}^{(0)}(x)) + b_l^{(0)}$
  end for
  $g_N(x) = I$ // Finding $\nabla_{z_l} f^{(0)}(x)$
  for $l = N-1, ...1$ **do**
    $g_l(x) = diag(\sigma'(z_l^{(0)}(x)))\frac{1}{\sqrt{m_l}} W_{l+1}^T g_{l+1}(x)$
  end for
end for
for $t = 1, ...T$ **do**
  Sample mini-batch $x \in \mathcal{X}$
  for $l = 1, ...N$ **do**
    $z_l^{(t-1)}(x) = \frac{1}{\sqrt{m_{l-1}}} W_l^{(t-1)} \sigma(z_{l-1}^{(t-1)}(x)) + b_l^{(t-1)}$
  end for
  $W_l^{(t)} = W_l^{(t-1)} + \frac{\eta}{\sqrt{m_{l-1}}} g_l(x) \nabla_{z_N(x)} L(z_N^{(t-1)}(x), y(x)) \sigma(z_{l-1}^{(0)}(x))^T$
  $b_l^{(t)} = b_l^{(t-1)} + \eta g_l(x) \nabla_{z_N(x)} L(z_N^{(t-1)}(x), y(x))$
end for
Return $W_1^{(T)}, b_1^{(T)}, ...W_N^{(T)}, b_N^{(T)}$

Algorithm 2 Align-ada training

Require: Training points $\mathcal{X}$, training steps T, layer widths $m_1, m_2, ...m_L$, learning rate η
  Initialize weights $W_1^{(0)}, b_1^{(0)}, ...W_N^{(0)}, b_N^{(0)}$
  for Mini-batch $x \in \mathcal{X}$ **do**
    for $l = 1, ...N$ **do**
      $z_l^{(0)}(x) = \frac{1}{\sqrt{m_{l-1}}} W_l^{(0)} \sigma(z_{l-1}^{(0)}(x)) + b_l^{(0)}$
    end for
    $g_N(x) = I$ // Finding $\nabla_{z_l} f^{(0)}(x)$
    for $l = N-1, ...1$ **do**
      $g_l(x) = diag(\sigma'(z_l^{(0)}(x))) \frac{1}{\sqrt{m_l}} W_{l+1}^T g_{l+1}(x)$
    end for
  end for
  for $t = 1, ...T$ **do**
    Sample mini-batch $x \in \mathcal{X}$
    for $l = 1, ...N$ **do**
      $z_l^{(t-1)}(x) = \frac{1}{\sqrt{m_{l-1}}} W_l^{(t-1)} \sigma(z_{l-1}^{(t-1)}(x)) + b_l^{(t-1)}$
    end for
    $W_l^{(t)} = W_l^{(t-1)} + \frac{\eta}{\sqrt{m_{l-1}}} g_l(x) \nabla_{z_N(x)} L(z_N^{(t-1)}(x), y(x)) \sigma(z_{l-1}^{(t-1)}(x))^T$
    $b_l^{(t)} = b_l^{(t-1)} + \eta g_l(x) \nabla_{z_N(x)} L(z_N^{(t-1)}(x), y(x))$
  end for
  Return $W_1^{(T)}, b_1^{(T)}, ...W_N^{(T)}, b_N^{(T)}$

D. Remark on difference between Align-zero and (Lee et al., 2019)

Our Align-zero learning rule differs subtly but importantly from the linearized networks of (Lee et al., 2019). Both formulations use the learning dynamics of Equation (13) for their parameters – which is linear in the parameters. However, the two formulations apply Equation (13) as a function of their respective activation dynamics $f_{al\cdot 0}^{(t)}(x), f_{lin}^{(t)}(x)$. While our activation dynamics $f_{al\cdot 0}^{(t)}(x)$ are a nonlinear neural network with parameters $W_{l,al\cdot 0}^{(t)}$ and $b_{l,al\cdot 0}^{(t)}$, the activation dynamics $f_{lin}^{(t)}(x)$ of (Lee et al., 2019) are themselves linear and *not a neural network*. Rather, $f_{lin}^{(t)}(x)$ is the first order term in the Taylor expansion of the neural network $f_{al\cdot 0}^{(t)}(x)$:

$$f_{al\cdot 0}^{(t)}(x) = f^{(0)}(x) + \nabla_\theta f^{(0)}(x)^T(\theta^{(t)} - \theta^{(0)}) + O((\theta^{(t)} - \theta^{(0)})^2) = f_{lin}^{(t)}(x) + O((\theta^{(t)} - \theta^{(0)})^2) \tag{57}$$

where $\theta^{(t)}$ represents the parameters of both functions. While the two types of linearization behave similarly near initialization, our approach linearizes only the parameter dynamics, while the approach of (Lee et al., 2019) also linearizes the network's activation dynamics. We focus on only linearizing the parameter dynamics because our goal is to study learning rules for actual (non-linearized) neural networks.

E. Implementation of Align-zero and Align-ada in recurrent neural networks

Although throughout this paper we use the notation of feedforward neural networks, note that our analysis applies to recurrent neural networks as well since RNNs produce feedforward networks when unrolled over time. For example, in the notation of feedforward networks, $z_l(x)$ would refer to the pre-activated hidden state of the network at time step l of recurrent processing. However, there are typically three key differences with the feedforward case: 1) losses $L(\hat{y}_l(x), y_l(x))$ can be accumulated over different time steps where $\hat{y}_l(x)$ is the predicted label depending on state $z_l(x)$, 2) inputs can be fed in at different time steps, 3) that recurrent weights and biases are tied at all time-steps (i.e. $W = W_1 = W_2 = ...; b = b_1 = b_2 = ...$), although these weights can change over the course of training. Thus, parameter updates must be summed over all losses and all recurrent time-steps where the parameters are applied. The resulting Align-ada learning rule for the recurrent weights of an RNN is:

$$\dot{W}_{al\cdot ada}^{(t)} = \frac{\eta}{\sqrt{m}} \times \sum_{i,j:j>i} \mathbb{E}_{p_x}[\nabla_{z_i} \hat{y}_j^{(0)}(x) \nabla_{\hat{y}_j} L(\hat{y}_j^{(t)}(x), y_j(x)) \sigma(z_{i-1}^{(t)}(x))^T] \tag{58}$$

The learning rules for other parameters (i.e. biases, readout-weights, etc.) can be found similarly, and the adaptation of the Align-zero learning rule can be found analogously.

F. Implementation details

All experiments are run on an Nvidia Tesla K20Xm GPU. Unless otherwise mentioned, all networks are trained under the NTK parameterization as defined in Equation 1; see Appendix H for experiments under standard parameterization. For alignment measurements, we compute alignment scores using Equation (10), estimating expectations with 100 Gaussian samples. For all randomness, the random seed is set to 99.

Baselines We primarily compare Align-ada and Align-zero against biologically-motivated learning algorithms, although, for reference, we also compare against gradient descent (denoted Normal). Among biologically-motivated algorithms, we compare with training only the last fully connected layer of the network, leaving intermediate layer parameters fixed, which provides a useful baseline to check that other methods are optimizing intermediate layer parameters (denoted Last layer). We also compare with feedback alignment (denoted FA) (Lillicrap et al., 2016) which uses fixed random feedback weights; and with direct feedback alignment (denoted DFA) (Nøkland & Hiller Eidnes, 2019), which replaces the x-dependent feedback Jacobian $\nabla_{z_l} f^{(0)}(x)$ of Align-zero with a constant random matrix.

CIFAR-10 experiments We train on CNN architecture with 7 convolutional layers followed by global average pooling and a fully connected layer. All convolutional layers use 3-by-3 convolutions with the same number of filters, which is varied from 8 to 512. All but the 4th and 6th convolutional layers have stride 1-by-1, with the other two using stride 2-by-2. All convolution layers are followed by batch normalization and a ReLU non-linearity, unless otherwise mentioned. Networks are trained on a mean squared error loss. Following (Novak et al., 2019), labels are constructed as one hot encoded labels minus 0.1 so that they have mean 0. Images are scaled in range $[0, 1]$. For each training method, test set accuracies are evaluated at each training epoch, and the best results reported. A learning rate grid search in $\{1, 2, 5\}$ is performed for all training methods and the best result reported unless unless otherwise mentioned. See our experiments varying learning rate for observations on training instability beyond a learning rate of 5. We use a batch size of 100, and all methods are trained for 150 epochs unless otherwise mentioned. Best results are reported over all epochs trained unless otherwise reported. For experiments on Small CIFAR-10, we randomly sample a subset of points from the full CIFAR-10 training set to use as training points.

KMNIST experiments We train on CNN architecture with 3 convolutional layers followed by global average pooling and a fully connected layer. All convolutional layers use 3-by-3 convolutions with the same number of filters, which is varied from 8 to 512. The first layer uses stride 1-by-1, with the other two using stride 2-by-2. All convolution layers are followed by batch normalization and a ReLU non-linearity. Networks are trained on a mean squared error loss. Following (Novak et al., 2019), labels are constructed as one hot encoded labels minus 0.1 so that they have mean 0. Images are scaled in range $[0, 1]$. For each training method, test set accuracies are evaluated at each training epoch, and the best results reported. A learning rate grid search in $\{1, 2, 5\}$ is performed for all training methods and the best result reported unless unless otherwise mentioned. See our experiments varying learning rate for observations on training instability beyond a learning rate of 5. We use a batch size of 100, and all methods are trained for 100 epochs unless otherwise mentioned. Best results are reported over all epochs trained unless otherwise reported.

Add task experiments The task has a sequence of iid Bernoulli inputs $x(t)$ and desired labels $y(t)$ constructed as:

$$y(t) = 0.5 + 0.5x(t) * \delta(t-2) - 0.25x(t) * \delta(t-5) \tag{59}$$

where $*$ denotes convolution. The dataset consists of 400 labeled input sequences of length 100, split into a training dataset of 300 sequences and a test dataset of 100 sequences. Given parameters W_h, b_h, W_i, W_o, b_o, the recurrent neural network to predict $y(t)$ is given by:

$$z_{k+1}(x) = \frac{1}{\sqrt{m}} W_h \sigma(z_k(x)) + b_h + W_i x_k \tag{60}$$

$$\hat{y}_k(x) = W_o \sigma(z_k(x)) + b_o \tag{61}$$

where k refers to the current recurrent time step, m is the size of the hidden state $z_k(x)$, and x_k and $\hat{y}_k(x)$ refer to the current time input and predicted output respectively. We vary the number of hidden units in the recurrent state in a range from 8 to 4096. We train all methods with a learning rate of 0.001 and batch size 50 for 200 epochs on the mean squared error loss.

ImageNet experiments We train on CNN architecture with 7 convolutional layers followed by global average pooling and a fully connected layer. All convolutional layers use 3-by-3 convolutions with 512 filters. The first convolutional layer has stride 4-by-4, followed alternating convolutional layers of stride 2-by-2 and 1-by-1. All convolution layers are followed by batch normalization and a ReLU non-linearity. Due to the high memory cost of storing backward projection weights for DFA on ImageNet, we apply the same set of backward projection weights at each spatial location for all layers. Networks are trained on a mean squared error loss with one-hot-encoded labels. We train on ILSVRC-12 images (Russakovsky et al., 2015) using the following pre-processing steps: we select random crops of the original images with scales in range $[0.08, 1.0]$ and aspect ratio in range $[0.75, 1.34]$; the images are then re-scaled to 224-by-224 pixels. Next, we randomly flip images horizontally and apply a channel-wise normalization. We use a batch size of 100, and train networks using a learning rate of 5 for 150000 steps.

G. Additional Align variant: Align-prop

In this section, we consider an additional variant of Align called Align-prop. Align-prop is equivalent to a variant of Feedback Alignment (Lillicrap et al., 2016) in which the fixed feedback weights are set equal to the feedforward weights at initialization. Although Align-prop is similar to Align-ada, it has the following key difference: Align-ada uses a fixed feedback Jacobian $\nabla_{z_l} f^{(0)}(x)$ to estimate gradients at each layer while Align-prop's approximation of the feedback Jacobian is not fixed. This is because in Align-prop's approximation of the true feedback Jacobian $\nabla_{z_l} f^{(t)}(x)$, although the feedback weights are fixed, the derivatives of activations in intermediate layers change over the course of training. Thus, Align-prop must propagate backwards through each feedback weights at each training step while Align-ada can completely avoid such propagation by storing $\nabla_{z_l} f^{(0)}(x)$.

Furthermore, we note that Align-prop satisfies the conditions required of Align-ada and Align-zero in Proposition 2. This is because Align-prop's approximation of the true feedback Jacobian $\nabla_{z_l} f^{(t)}(x)$ changes infinitesimally from initialization in infinite width networks since activations in infinite width networks change infinitesimally over the course of training. Thus, Align-prop approaches gradient descent in the limit of infinite width networks.

We conduct experiments comparing Align-prop to other baselines on 8 layer CNNs trained on CIFAR-10; we leave a more extensive empirical study of Align-prop as a future work. We experiment on wide networks with width $\times 128$, $\times 256$ and $\times 512$. Otherwise, we use the same architecture and hyperparameter settings used in our main experiments on CIFAR-10 (see Appendix F).

As observed in Table 1, Align-prop outperforms all biologically-motivated baselines including other Align variants. We suspect that this is because Align-prop is allowed to adapt its approximation of the layerwise feedback Jacobian over the course of training: the activation patterns in the backward pass of Align-prop match those used during the forward pass.

Table 1: Test set accuracies of 8-layer convolutional neural networks with different layer widths on the CIFAR-10 dataset with different methods of training. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. For each scale, the best performing biologically-motivated method is in **bold**.

Scale	x128	x256	x512
Align-prop	**59.2**%	**59.2**%	**59.6**%
Align-ada	54.2%	56.6%	58.0%
Align-zero	40.0%	47.6%	51.4%
DFA (Nøkland, 2016)	54.9%	54.5%	54.1%
FA (Lillicrap et al., 2016)	45.7%	45.6%	45.4%
Last layer	34.8%	40.2%	43.4%
Normal	64.2%	63.2%	62.3%

H. Standard parameterization experiments

In this section, we train networks with Align-ada and Align-zero under standard parameterization. We note that our theoretical analysis and algorithmic contributions are primarily applicable to networks trained in the NTK training regime; nevertheless, we consider to what extent our proposed learning rules can extend to standard parameterization.

Recall that in Equation 1, the network is defined under NTK parameterization as:

$$z_l(x) = \frac{1}{\sqrt{m_{l-1}}} W_l \sigma(z_{l-1}(x)) + b_l \tag{62}$$

where m_{l-1} is the dimensionality of $z_{l-1}(x)$. Under *standard parameterization*, we omit the width dependent normalization in the definition of the network:

$$z_l(x) = W_l \sigma(z_{l-1}(x)) + b_l \tag{63}$$

However, under standard parameterization, weight initializations are appropriately scaled such that the distribution of networks are the same as under NTK parameterization *at initialization*. During training, weights and biases evolve differently under the two parameterizations. See (Jacot et al., 2018) for further discussion of the difference between NTK and standard parameterization.

We train CNNs with 7 convolutional layers under the standard parameterization on CIFAR-10. We follow the training procedure detailed in Appendix F with the following differences: we perform an extensive grid search over learning rates in $\{0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, and we use Adam (Kingma & Ba, 2015) to optimize network parameters. To better understand the differences in training behavior between standard parameterization and NTK parameterization, we also consider two additional settings: 1) training only for 1 epoch, 2) training only for 1 epoch with a small learning rate of 0.0005.

As observed in Appendix L Table 10, Normal achieves $> 85\%$ test set accuracy on the widest networks tested. Align-ada incurs a performance gap with Normal of $20 - 30\%$, although its absolute performance in this setting is higher than under the NTK regime (see Appendix L Table 6 for tabulated results in the NTK regime). Align-zero is unstable in this training setting. These experiments validate that the Align methods' comparable performance to gradient descent relies on the NTK regime, as expected by our theoretical analysis.

We also observe that as in settings more closely matching the NTK regime, the gap between Align-ada and gradient descent decreases. In particular, we find more comparable performance when limiting training to 1 epoch. This is practically relevant in settings with a limited computational budget available for training: in these settings, Align-ada performs closer to the maximum performance expected (i.e. by training with gradient descent). We also observe that when training with a small learning rate, the test set accuracy gap between Align-ada and Normal decreases, particularly for narrower networks. This is reasonable since in the NTK training regime, parameters move in a small neighborhood around their initialization points (Jacot et al., 2018), which can be qualitatively matched under standard parameterization by restricting training time and learning rate.

I. Seed robustness experiments

To verify the statistical significance of our results, we conduct multiple trials of one of our experiments under multiple random seeds. Specifically, we train CNNs with 7 convolutional layers at a width scale of $\times 256$ on CIFAR-10 using 5 new random seeds. The architecture and hyperparameters are chosen to be consistent with our main experiments; in particular, we train for 150 epochs.

As observed in Table 2, the trends of the results over multiple random seeds are consistent with what we observe in our main experiments (see Table 6). Moreover, the standard deviations of the test set accuracies are less than 1% for each method. This gives us confidence in the statistical significance of our results.

Table 2: Test set accuracies of 8-layer convolutional neural with 256 filters per intermediate layer on the CIFAR-10 dataset with different methods of training. Means and standard deviations are reported over 5 random trials with different random seeds.

Method	Align-ada	Align-zero	DFA	FA	Normal
Accuracy	$56.4 \pm 0.4\%$	$46.9 \pm 0.5\%$	$55.3 \pm 0.7\%$	$46.1 \pm 0.5\%$	$63.1 \pm 0.4\%$

J. Training time of Align methods

In this section, we evaluate the training time of Align methods compared to backpropagation. We highlight that this is done to assess the possibility of using Align methods as a learning rule for practical applications rather than to evaluate

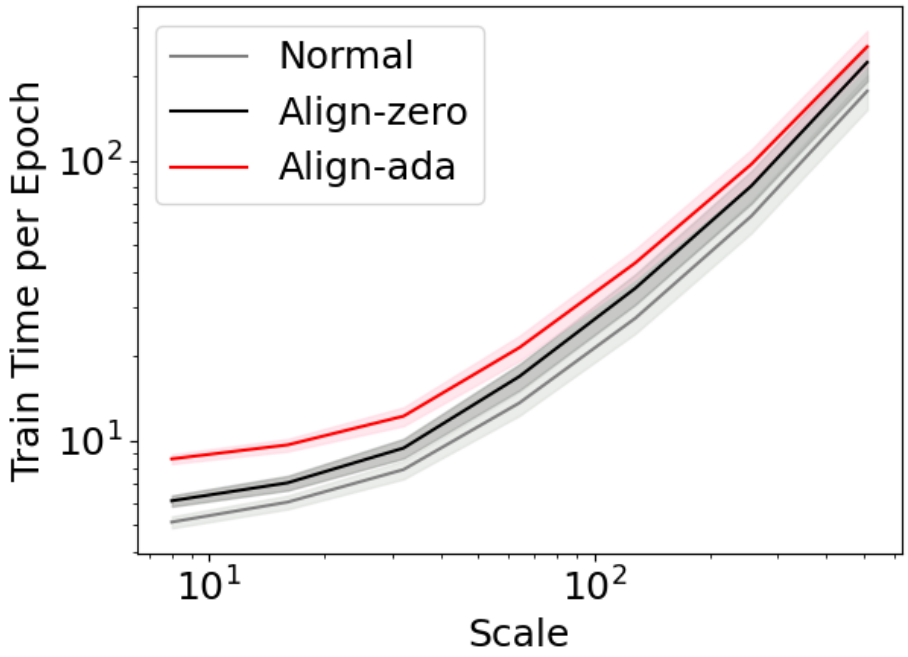


Figure 7: Per epoch training times of different learning rules vs. network scale of CNNs trained on CIFAR-10. Margins indicate standard errors over 15 trials.

the biological plausibility of Align methods. Furthermore, we note that while the implementation of backprop has been extensively optimized due to its popularity, Align methods have not be optimized to run efficiently on standard computing hardware.

We compare training times of Align-ada and Align-zero with backprop. We evaluate on CNNs with 7 convolutional layers at a variety of architectural scales. The architecture is chosen to be consistent with our main experiments. For each training method and architecture scale, we measure training times over 15 trials.

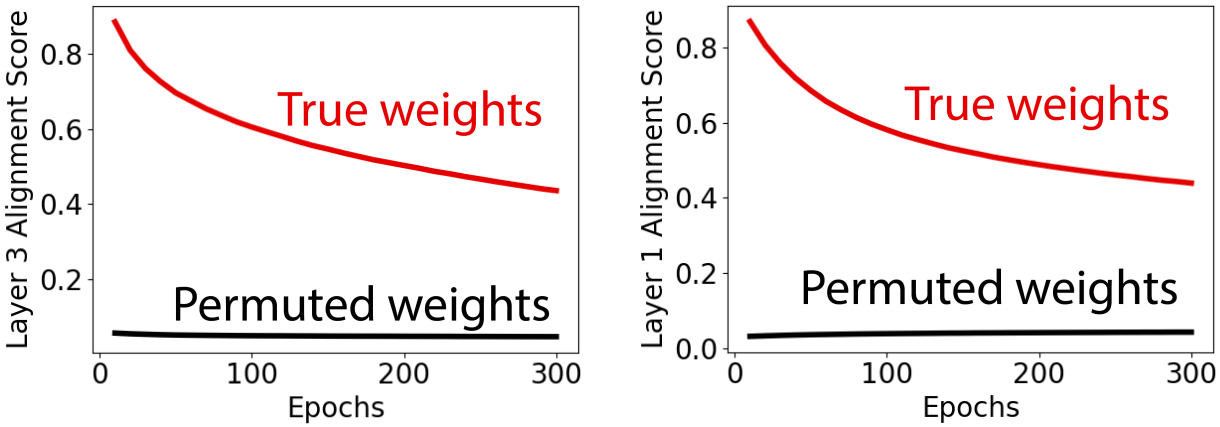
As observed in Figure 7, we find that Align methods achieve generally comparable training times to backprop, with training times of all methods scaling similarity with network width. Align methods appear to be nearly a fixed factor slower than backprop, with Align-zero faster than Align-ada. This is because our implementation of Align methods stores additional information about the network state at initialization in order to approximate gradients.

K. Results on Add task

Dataset and baselines We train standard ReLU-activated RNNs on the Add task used as a benchmark for biologically-plausible RNN training rules in (Marschall et al., 2020). Additional details on the dataset, architectures and hyperparameters are included in Appendix F. We compare Align-ada and Align-zero to Normal and Readout-only, which trains only readout weights and fixes all other weights.

Varying network width In Appendix L Figure 9, we plot the training and test set losses of different learning rules. Although we find a large gap between Normal and alignment-based methods at small width, at larger widths, the gap decreases with Align methods performing comparably to Normal at a network scale of 512 hidden units. At large widths, the performance of Align methods far exceed Readout-only training, indicating that tuning the hidden layer representation is useful even at large width when random representations provide enough information to adequately perform the task. See Appendix L for full tabulated results.

L. Additional tables and figures



(a) CIFAR-10 (b) KMNIST

Figure 8: Alignment scores of convolutional neural networks over the course of training on the CIFAR-10 and KMNIST. As a baseline, alignment scores in networks with randomly permuted weights are also plotted. Training is conducted for 300 epochs to measure the evolution of alignment scores over a longer time-scale.

Table 3: Alignment scores of 8-layer ReLU-activated convolutional neural networks with different layer widths trained on the CIFAR-10 dataset. The network is trained for 150 epochs at a learning rate of 1. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. Alignment scores are shown for all intermediate layers of each network after training. Non-monotonicity with layer width is due to statistical fluctuations.

Alignment	x8	x16	x32	x64	x128	x256	x512
Layer 1	0.430	0.465	0.527	0.502	0.651	0.720	0.749
Layer 2	0.490	0.433	0.409	0.409	0.656	0.658	0.760
Layer 3	0.501	0.467	0.446	0.515	0.586	0.669	0.766
Layer 4	0.372	0.297	0.350	0.371	0.497	0.703	0.767
Layer 5	0.499	0.333	0.438	0.499	0.546	0.650	0.711
Layer 6	0.301	0.363	0.329	0.321	0.419	0.500	0.603
Layer 7	0.476	0.382	0.432	0.405	0.333	0.398	0.450

Table 4: Alignment scores of 4-layer convolutional neural networks with different layer widths trained on the KMNIST dataset. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. Alignment scores are shown for all intermediate layers of each network after training. Non-monotonicity with layer width is due to statistical fluctuations.

Alignment	x8	x16	x32	x64	x128	x256	x512
Layer 1	0.456	0.585	0.579	0.646	0.743	0.721	0.800
Layer 2	0.137	0.179	0.210	0.299	0.381	0.482	0.656
Layer 3	0.165	0.207	0.232	0.253	0.376	0.471	0.581

Table 5: Alignment scores of 8-layer Tanh-activated convolutional neural networks with different layer widths trained on the CIFAR-10 dataset. The network is trained for 100 epochs at a learning rate of 2. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. Alignment scores are shown for all intermediate layers of each network after training. Non-monotonicity with layer width is due to statistical fluctuations.

Alignment	x8	x16	x32	x64	x128	x256	x512
Layer 1	0.459	0.605	0.681	0.816	0.829	0.925	0.935
Layer 2	0.600	0.513	0.565	0.825	0.785	0.912	0.945
Layer 3	0.660	0.539	0.480	0.695	0.811	0.905	0.932
Layer 4	0.470	0.324	0.504	0.741	0.788	0.887	0.919
Layer 5	0.621	0.606	0.608	0.800	0.779	0.849	0.910
Layer 6	0.478	0.547	0.537	0.684	0.698	0.819	0.871
Layer 7	0.439	0.675	0.530	0.587	0.595	0.759	0.827

Table 6: Test set accuracies of 8-layer convolutional neural networks with different layer widths on the CIFAR-10 dataset with different methods of training. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. For each scale, the best performing biologically-motivated method is in **bold**.

Scale	x8	x16	x32	x64	x128	x256	x512
Align-ada	33.7%	35.7%	42.8%	**49.9%**	54.2%	**56.6%**	**58.0%**
Align-zero	20.9%	22.0%	30.1%	32.6%	40.0%	47.6%	51.4%
DFA (Nøkland, 2016)	28.5%	34.8%	43.0%	**49.9%**	**54.9%**	54.5%	54.1%
FA (Lillicrap et al., 2016)	**36.1%**	**40.3%**	**43.1%**	46.5%	45.7%	45.6%	45.4%
Last layer	20.7%	22.5%	28.2%	31.1%	34.8%	40.2%	43.4%
Normal	54.8%	61.0%	64.2%	65.3%	64.2%	63.2%	62.3%

Table 7: Test set accuracies of 4-layer convolutional neural networks with different layer widths on the KMNIST dataset with different methods of training. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. For each scale, the best performing biologically-motivated method is in **bold**.

Scale	x8	x16	x32	x64	x128	x256	x512
Align-ada	47.5%	58.8%	**70.7%**	**71.2%**	**76.1%**	**77.8%**	**78.4%**
Align-zero	24.5%	29.9%	31.2%	47.2%	50.3%	55.2%	61.2%
DFA (Nøkland, 2016)	37.6%	49.4%	61.4%	70.7%	75.8%	77.3%	77.9%
FA (Lillicrap et al., 2016)	**52.4%**	**66.3%**	68.6%	69.6%	72.1%	72.8%	63.5%
Last layer	25.9%	35.6%	42.2%	51.8%	58.7%	65.6%	68.2%
Normal	72.2%	83.4%	86.2%	87.2%	87.5%	87.4%	87.2%

Table 8: Training and test loss for RNNs trained with normal, Align-ada, Align-zero and readout-only training on the Add task. Training losses are higher than test set losses due to the test set being an easier task: random chance loss is 9.359 on the training set and 9.312 on the test set.

Scale	x8	x16	x32	x64	x128	x256	x512	x1024	x2048	x4096
Align-ada, Train loss	6.652	1.944	0.782	0.772	0.786	0.535	0.474	0.455	0.450	0.466
Align-ada, Test loss	6.795	1.890	0.716	0.707	0.718	0.461	0.394	0.374	0.369	0.388
Align-zero, Train loss	5.228	2.235	1.272	0.621	0.741	0.807	0.482	0.448	0.442	0.447
Align-zero, Test loss	5.227	2.211	1.205	0.556	0.663	0.742	0.408	0.367	0.364	0.372
Readout-only, Train loss	3.008	2.798	2.011	1.872	1.674	2.655	2.202	1.881	2.042	1.647
Readout-only, Test loss	2.957	2.794	1.990	1.855	1.601	2.623	2.077	1.811	1.976	1.608
Normal, Train loss	1.168	0.553	0.495	0.527	0.474	0.440	0.429	0.408	0.412	0.429
Normal, Test loss	1.133	0.488	0.411	0.461	0.399	0.366	0.352	0.333	0.339	0.357

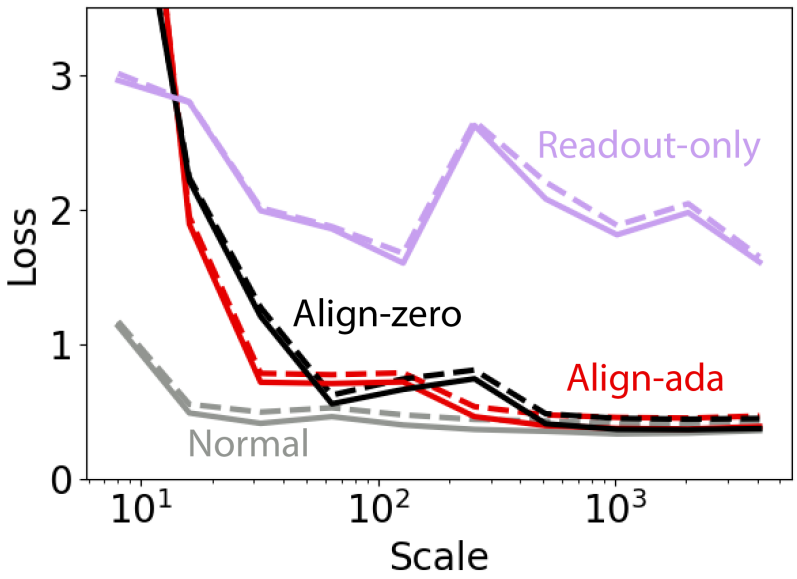


Figure 9: Loss of different learning rules (Normal, Align-ada, Align-zero, Readout-only) as a function of architecture scale on a RNN. Solid lines: test loss; dashed: train loss.

Table 9: Test set accuracies of 8-layer convolutional neural networks with different layer widths on the CIFAR-10 dataset with different methods of training (Align-ada, Align-zero, FA (Lillicrap et al., 2016), DFA (Nøkland, 2016), Normal). Results are reported when trained on different numbers of randomly sampled CIFAR-10 training points. The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. For each scale and dataset size, the best performing biologically-motivated method is in **bold**.

Training data	Method	x8	x16	x32	x64	x128	x256	x512
45	Align-ada	**28.9%**	**31.1%**	**28.9%**	**31.1%**	**31.1%**	**31.1%**	**28.9%**
45	Align-zero	24.4%	17.8%	22.2%	24.4%	28.9%	24.4%	20.0%
45	FA	24.4%	24.4%	24.4%	26.7%	26.7%	26.7%	17.8%
45	DFA	17.8%	28.9%	**28.9%**	28.9%	28.9%	28.9%	20.0%
45	Normal	24.4%	17.8%	22.2%	20.0%	22.2%	24.4%	26.7%
150	Align-ada	20.7%	22.7%	24.0%	**27.3%**	**30.0%**	**30.0%**	27.3%
150	Align-zero	**21.3%**	20.0%	**24.7%**	23.3%	29.3%	27.3%	**28.0%**
150	FA	20.0%	**23.3%**	23.3%	25.3%	25.3%	25.3%	22.0%
150	DFA	20.7%	20.7%	21.3%	21.3%	21.3%	21.3%	22.7%
150	Normal	22.0%	24.0%	24.7%	24.7%	29.3%	28.7%	28.7%
450	Align-ada	25.1%	25.3%	**28.2%**	**30.4%**	**33.3%**	**33.3%**	**35.8%**
450	Align-zero	22.0%	22.0%	26.4%	26.0%	29.1%	32.0%	32.4%
450	FA	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	24.7%
450	DFA	**26.7%**	**26.7%**	26.7%	28.0%	28.0%	28.0%	25.1%
450	Normal	28.7%	29.6%	32.0%	31.3%	32.7%	33.1%	34.2%
1500	Align-ada	26.8%	**29.9%**	**31.3%**	**35.5%**	**36.1%**	**40.2%**	**39.1%**
1500	Align-zero	20.7%	23.1%	28.4%	31.2%	33.2%	37.0%	37.3%
1500	FA	**29.1%**	29.1%	29.1%	29.1%	29.1%	29.1%	25.8%
1500	DFA	24.9%	28.8%	30.1%	30.1%	30.1%	30.1%	28.6%
1500	Normal	32.9%	33.1%	36.5%	39.5%	39.3%	40.3%	41.0%
4500	Align-ada	**30.3%**	**33.1%**	**34.9%**	**40.0%**	**42.3%**	**45.2%**	**46.2%**
4500	Align-zero	22.3%	23.1%	30.1%	31.8%	36.4%	41.9%	42.6%
4500	FA	28.9%	31.6%	33.5%	35.0%	35.0%	35.0%	33.0%
4500	DFA	26.4%	30.2%	34.4%	39.0%	40.4%	40.4%	36.0%
4500	Normal	39.3%	43.8%	45.5%	48.4%	47.9%	49.2%	49.2%
15000	Align-ada	32.5%	35.2%	40.6%	46.1%	49.8%	**51.8%**	**53.3%**
15000	Align-zero	21.7%	23.1%	30.4%	31.8%	38.9%	46.2%	48.5%
15000	FA	**33.9%**	**38.9%**	40.4%	44.2%	44.2%	44.2%	41.9%
15000	DFA	26.8%	33.2%	**40.7%**	**47.1%**	**51.0%**	51.0%	50.0%
15000	Normal	50.1%	54.4%	58.9%	58.1%	57.3%	57.2%	57.4%
45000	Align-ada	33.7%	35.7%	42.8%	**49.9%**	54.2%	**56.6%**	**58.0%**
45000	Align-zero	20.9%	22.0%	30.1%	32.6%	40.0%	47.6%	51.4%
45000	FA	**36.1%**	**40.3%**	**43.1%**	46.5%	45.7%	45.6%	45.4%
45000	DFA	28.5%	34.8%	43.0%	**49.9%**	**54.9%**	54.5%	54.1%
45000	Normal	54.8%	61.0%	64.2%	65.3%	64.2%	63.2%	62.3%

Table 10: Test set accuracies of 8-layer convolutional neural networks with different layer widths on the CIFAR-10 dataset with different methods of training. Networks are trained under standard parameterization using Adam optimization (Kingma & Ba, 2015). The scale of the architecture is denoted $\times n$, where n is the number of filters in intermediate layers. Accuracies are reported when trained with 150 epochs, 1 epoch, and 1 epoch with a small learning rate.

Scale	x8	x16	x32	x64	x128	x256	x512
Standard parameterization, 150 epochs							
Align-ada	34.7%	41.1%	47.4%	54.4%	58.6%	61.7%	64.9%
Align-zero	17.3%	16.2%	15.4%	13.8%	14.0%	14.3%	12.6%
Normal	60.9%	73.3%	79.1%	82.8%	84.9%	86.5%	87.6%
Standard parameterization, 1 epoch							
Align-ada	26.1%	27.7%	35.4%	38.1%	40.7%	41.4%	38.0%
Align-zero	14.3%	11.0%	14.1%	12.3%	12.1%	11.2%	12.0%
Normal	37.3%	43.8%	53.4%	58.7%	59.2%	58.5%	46.2%
Standard parameterization, 1 epoch + small learning rate							
Align-ada	13.0%	24.0%	35.4%	37.7%	40.7%	41.4%	37.9%
Align-zero	8.9%	11.0%	14.1%	10.9%	12.1%	11.2%	11.5%
Normal	17.9%	32.1%	46.4%	55.2%	57.7%	58.5%	46.2%