Lazy vs hasty: linearization in deep networks impacts learning schedule based on example difficulty

Anonymous authors
Paper under double-blind review

Abstract

Among attempts at giving a theoretical account of the success of deep neural networks, a recent line of work has identified a so-called 'lazy' training regime in which the network can be well approximated by its linearization around initialization. Here we investigate the comparative effect of the lazy (linear) and feature learning (non-linear) regimes on subgroups of examples based on their difficulty. Specifically, we show that easier examples are given more weight in feature learning mode, resulting in faster training compared to more difficult ones. In other words, the non-linear dynamics tends to sequentialize the learning of examples of increasing difficulty. We illustrate this phenomenon across different ways to quantify example difficulty, including c-score, label noise, and in the presence of easy-to-learn spurious correlations. Our results reveal a new understanding of how deep networks prioritize resources across example difficulty.

1 Introduction

Understanding the performance of deep learning algorithms has been the subject of intense research efforts in the past few years, driven in part by observed phenomena that seem to defy conventional statistical wisdom (Neyshabur et al., 2015; Zhang et al., 2017; Belkin et al., 2019). Notably, many such phenomena have been analyzed rigorously in simpler contexts of high dimensional linear or random feature models (e.g., Hastie et al., 2022; Bartlett et al., 2021), which shed a new light on the crucial role of overparametrization in the performance of such systems. These results also apply to the so-called *lazy training* regime (Chizat et al., 2019), in which a deep network can be well approximated by its linearization at initialization, characterized by the neural tangent kernel (NTK) (Jacot et al., 2018; Du et al., 2019b; Allen-Zhu et al., 2019). In this regime, deep networks inherit the inductive bias and generalization properties of kernelized linear models.

It is also clear, however, that deep models cannot be understood solely through their kernel approximation. Trained outside the lazy regime (e.g., Woodworth et al., 2020), they are able to learn adaptive representations, with a time-varying tangent kernel (Fort et al., 2020) that specializes to the task during training (Kopitkov & Indelman, 2020; Baratin et al., 2021; Paccolat et al., 2021; Ortiz-Jiménez et al., 2021). Several results showed examples where their inductive bias cannot be characterized in terms of a kernel norm (Savarese et al., 2019; Williams et al., 2019), or where they provably outperform any linear method (Malach et al., 2021). Yet, the specific mechanisms by which the two regimes differ, which could explain the performance gaps often observed in practice (Chizat et al., 2019; Geiger et al., 2020), are only partially understood.

Our work contributes new qualitative insights into this problem, by **investigating the comparative effect of the lazy and feature learning regimes on the training dynamics of various groups of examples of increasing difficulty**. We do so by means of a control parameter that modulates linearity of the parametrization and smoothly interpolates the two training regimes (Chizat et al., 2019; Woodworth et al., 2020). We provide empirical evidence and theoretical insights suggesting that **the feature learning regime puts higher weight on easy examples** at the beginning of training, which results in an increased learning speed compared to more difficult examples. This can be understood as an instance of the **simplicity bias** brought forward in recent work (Arpit et al., 2017; Rahaman et al., 2019; Kalimeris et al., 2019), which

we show here to be much more pronounced in the feature learning regime. It also resonates with the old idea that generalization can benefit from some curriculum learning strategy (Elman, 1993; Bengio et al., 2009).

Contributions

- We introduce and test the hypothesis of qualitatively different example importance between linear and non-linear regimes;
- Using adequately normalized plots, we present a unified picture using 4 different ways to quantify example difficulty, where easy examples are prioritized in the non-linear regime. We illustrate empirically this phenomenon across different ways to quantify example difficulty, including c-score (Jiang et al., 2021), label noise, and spurious correlation to some easy-to-learn set of features.
- We illustrate some of our insights in a simple quadratic model amenable to analytical treatment.

The general setup of our experiments is introduced in Section 2. Section 3 is our empirical study, which begins with an illustrative example on a toy dataset (Section 3.1), followed by experiments on CIFAR 10 in two setups where example difficulty is quantified using respectively c-scores and label noise (Section 3.2). We also examine standard setups where easy examples are those with strong correlations between their labels and some spurious features (Section 3.3). Section 4 illustrates our findings with a theoretical analysis of a specific class of quadratic models, whose training dynamics is solvable in both regimes. We conclude in Section 5.

Related work The neural tangent kernel was initially introduced in the context of infinitely wide networks, for a specific parametrization that provably leads to the lazy regime (Jacot et al., 2018). Such a regime allows to cast deep learning as a linear model using a fixed kernel, enabling import of well-known results from linear models, such as guarantees of convergence to a global optimum (Du et al., 2019b;a; Allen-Zhu et al., 2019). On the other hand, it is also clear that the kernel regime does not fully capture the behavior of deep models – including, in fact, infinitely wide networks (Yang & Hu, 2021). For example, in the so-called mean field limit, training two-layer networks by gradient descent learns adaptive representations (Chizat & Bach, 2018; Mei et al., 2018) and it can be shown that the inductive bias cannot be characterized in terms of a RKHS norm (Savarese et al., 2019; Williams et al., 2019). Performance gaps between the two regimes are also often observed in practice (Chizat et al., 2019; Arora et al., 2019; Geiger et al., 2020).

Subsequent work showed and analyzed how, for a fixed (finite-width) network, the scaling of the model at initialization also controls the transition between the lazy regime, governed by the empirical neural tangent kernel, and the standard feature learning regime (Chizat et al., 2019; Woodworth et al., 2020; Agarwala et al., 2020). Following this prior work, in our experiments below we use a scaling parameter $\alpha > 0$ that modulates linearity of the parametrization and allows us to smoothly interpolates between the vanilla training ($\alpha = 1$) where features are learned and the lazy ($\alpha \to \infty$) regime where they are not. We compare training runs with various values of α and empirically assess linearity with several metrics described in Section 2 below.

Our results are in line with a group of work showing how deep networks learn patterns and functions of increasing complexity during training (Arpit et al., 2017; Kalimeris et al., 2019). An instance of this is the so-called spectral bias empirically observed in Rahaman et al. (2019) where a sum of sinusoidal signals is incrementally learned from low frequencies to high frequencies, or in Zhang et al. (2021) that use Fourier decomposition to analyze the function learned by a deep convolutional network on a vision task. the spectral bias is well understood in linear regression where the gradient dynamics favours the large singular directions of the feature matrix. For neural networks in the lazy regime, spectral analysis of the neural tangent kernel have been investigated for architectures and data (e.g, uniform data on the sphere) allowing for explicit computations (e.g., decomposition of the NTK in terms of spherical harmonics) (Bietti & Mairal, 2019; Basri et al., 2019; Yang & Salman, 2019). This is a setup where the spectral bias can be rigorously analyzed, i.e., we can get explicit information of the type of functions that are learned quickly and generalize well. In this context, our work specifically focuses on comparing the lazy and the standard feature learning regimes.

Our theoretical model in Section 4 reproduces some of the key technical ingredients of known analytical results (Saxe et al., 2014; Gidel et al., 2019) on deep linear networks. These results show how, in the context

of multiclass classification or matrix factorization, the principal components of the input-output correlation matrix are learned sequentially from the highest to the lowest mode. Note however that in the framework of these prior works, the number of modes is bounded by the output dimension - which reduces to one in the context of regression or binary classification setting. By contrast, our theoretical analysis applies to the components of the vector of labels $\boldsymbol{Y} \in \mathbb{R}^n$ in the eigenbasis of the kernel $\boldsymbol{X}\boldsymbol{X}^\top$ defined by the input matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. Thus, despite the technical similarities, our framework approaches the old problem of the relative learning speed of different modes in factorized models from a novel angle. In particular, it allows us to frame the notion of example difficulty in this context.

Example difficulty is a loosely defined concept that has been the subject of intense research recently. Ways to quantify example difficulty for a model/algorithm to learn individual examples include e.g. self-influence (Koh & Liang, 2017), example forgetting (Toneva et al., 2019), TracIn (Pruthi et al., 2020), C-scores (Jiang et al., 2021), or prediction depth (Baldock et al., 2021). We believe that a comprehensive theory of generalization in deep learning will require to understand how neural networks articulate learning and memorization to both fit the head (easy examples) and the tail (difficult examples) of the data distribution (Hooker et al., 2020; Feldman & Zhang, 2020; Sagawa et al., 2020a;b).

2 Setup

We consider neural networks $f_{\boldsymbol{\theta}}$ parametrized by $\boldsymbol{\theta} \in \mathbb{R}^p$ (i.e. weights and biases for all layers), and trained by minimizing some task-dependent loss function $\ell(\boldsymbol{\theta}) := \sum_{i=1}^n \ell_i(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$ computed on a training dataset $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, using variants of gradient descent,

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}^{(t)}), \tag{1}$$

with some random initialization $\boldsymbol{\theta}^{(0)}$ and a chosen learning rate $\eta > 0$.

Linearization A Taylor expansion and the chain rule give the corresponding updates $f^{(t)} := f_{\boldsymbol{\theta}^{(t)}}$ for any network output, at first order in the learning rate,

$$f^{(t+1)}(\mathbf{x}) \simeq f^{(t)}(\mathbf{x}) - \eta \sum_{i=1}^n K^{(t)}(\mathbf{x}, \mathbf{x}_i) \nabla \ell_i, \tag{2}$$

which depend on the time-varying **tangent kernel** $K^{(t)}(\mathbf{x}, \mathbf{x}') := \nabla_{\boldsymbol{\theta}} f^{(t)}(\mathbf{x})^\top \nabla_{\boldsymbol{\theta}} f^{(t)}(\mathbf{x}')$. The *lazy regime* is one where this kernel remains nearly constant throughout training. Training the network in this regime thus corresponds to training the linear predictor defined by

$$\bar{f}_{\boldsymbol{\theta}}(\mathbf{x}) := f_{\boldsymbol{\theta}^{(0)}}(\mathbf{x}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)})^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^{(0)}}(\mathbf{x}). \tag{3}$$

Modulating linearity In our experiments, following Chizat et al. (2019), we modulate the level of "non-linearity" during training of a deep network with a scalar parameter $\alpha \geq 1$, by replacing our prediction $f_{\boldsymbol{\theta}}$ by

$$f_{\boldsymbol{\theta}}^{\alpha}(\mathbf{x}) := f_{\boldsymbol{\theta}^{(0)}}(\mathbf{x}) + \alpha \left(f_{\boldsymbol{\theta}}(\mathbf{x}) - f_{\boldsymbol{\theta}^{(0)}}(\mathbf{x}) \right) \tag{4}$$

and by rescaling the learning rate as $\eta_\alpha = \eta/\alpha^2$. In this setup, gradient descent steps in parameter space are rescaled by $1/\alpha$ while steps in function space (up to first order) are in $O(1)$ in α.[1] α can also be viewed as controlling the *level of feature adaptivity*, where large values of α result in linear training where features are not learned. We will also experiment with $\alpha < 1$ below, which enhances adaptivity compared the standard regime ($\alpha = 1$). The goal of this procedure is two-fold: (i) to be able to smoothly interpolate between the standard regime at $\alpha = 1$ and the linearized one, (ii) to work with models that are nearly linearized yet practically trainable with gradient descent.

[1] Under some assumptions such as strong convexity of the loss, it was shown (Chizat et al., 2019, Thm 2.4) that as $\alpha \to \infty$, the gradient descent trajectory of $f^{\alpha}_{\boldsymbol{\theta}^{(t)}}$ gets uniformly close to that of the linearization $\bar{f}_{\boldsymbol{\theta}^{(t)}}$.

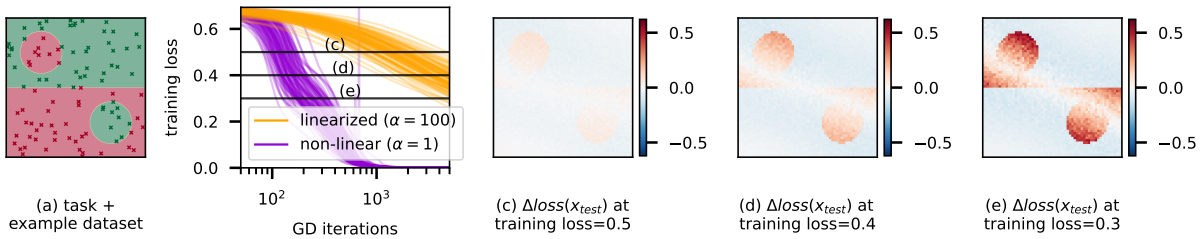

Figure 1: 100 randomly initialized runs of a 4 layers MLP trained on the yin-yang dataset **(a)** using gradient descent in both the non-linear ($\alpha = 1$) and linearized ($\alpha = 100$) setting. The training losses **(b)** show a speed-up in the non-linear regime: in order to compare both regimes at equal progress, we normalize by comparing models extracted at equal training loss thresholds **(c)**, **(d)** and **(e)**. We visualize the differences $\Delta\text{loss}\,(x_{\text{test}}) = \text{loss} f_{\text{non-linear}}\,(x_{\text{test}}) - \text{loss} f_{\text{linear}}\,(x_{\text{test}})$ for test points paving the 2d square $[-1, 1]^2$ using a color scale. We observe that these differences are not uniformly spread across examples: instead they suggest a comparative bias of the non-linear regime towards correctly classifying easy examples (large areas of the same class), whereas difficult examples (e.g. the small disks) are boosted in the linear regime.

Linearity measures In order to assess linearity of training runs for empirically chosen values of the re-scaling factor α, we track three different metrics during training (fig 2 right):

- **Sign similarity** counts the proportion of ReLUs in all layers that have kept the same activation status (0 or > 0) from initialization.

- **Tangent kernel alignment** measures the similarity between the Gram matrix $\boldsymbol{K}^{(t)}_{ij} = K^{(t)}(\mathbf{x}_i, \mathbf{x}_j)$ of the tangent kernel with its initial value $\boldsymbol{K}^{(0)}$, using kernel alignment (Cristianini et al., 2001)

$$\text{KA}(\boldsymbol{K}^{(t)}, \boldsymbol{K}^{(0)}) = \frac{\text{Tr}[\boldsymbol{K}^{(t)}\boldsymbol{K}^{(0)}]}{\|\boldsymbol{K}^{(t)}\|_F \|\boldsymbol{K}^{(0)}\|_F} \qquad (\|\cdot\|_F \text{ is the Froebenius norm}) \tag{5}$$

- **Representation alignment** measures the similarity of the last non-softmax layer representation $\phi^{(t)}_R(x)$ with its initial value $\phi^{(0)}_R$, in terms of the kernel alignment (eq. 5) of the corresponding Gram matrices $(\boldsymbol{K}^{(t)}_R)_{ij} = \phi^{(t)}_R(\mathbf{x}_i)^\top \phi^{(t)}_R(\mathbf{x}_j)$.

3 Empirical Study

3.1 A motivating example on a toy dataset

We first explore the effect of modulating the training regime for a binary classification task on a toy dataset with 2d inputs, for which we can get a visual intuition. We use a fully-connected network with 4 layers and ReLU activations. For 100 independent initial parameter values, we generate 100 training examples uniformly on the square $[-1, 1]^2$ from the yin-yang dataset (fig. 1.a). We perform 2 training runs for $\alpha = 1$ and $\alpha = 100$ using (full-batch) gradient descent with learning rate 0.01.

Global training speed-up and normalization After the very first few iterations, we observe a speed-up in training progress of the non-linear regime (fig. 1.b). This is consistent with previously reported numerical experiments on the lazy training regime (Chizat et al., 2019, section 3). This raises the question of whether this acceleration comes from a global scaling in all directions, or if it prioritizes certain particular groups of examples. We address this question by comparing the training dynamics at equal progress: we counteract the difference in training speed by normalizing by the mean training loss, and we compare the linear and non-linear regimes at common thresholds ((c), (d) and (e) horizontal lines in fig. 1.b).

Comparing linear and non-linear regimes At every threshold value, we compute the predictions on test examples uniformly paving the 2d square. We compare both regimes on individual test examples by plotting (fig. 1.c, 1.d, 1.e) the differences in loss values,

$$\Delta\text{loss}\left(x_{\text{test}}\right) = \text{loss} f_{\text{non-linear}}\left(x_{\text{test}}\right) - \text{loss} f_{\text{linear}}\left(x_{\text{test}}\right) \tag{6}$$

Red (resp. blue) areas indicate a lower test loss for the linearized (resp. non-linear) model. Remarkably, the resulting picture is not uniform: these plots suggest that compared to the linear regime, the non-linear training dynamics speeds up for specific groups of examples (the large top-right and bottom-left areas) at the expense of examples in more intricate areas (both the disks and the areas between the disks and the horizontal boundary).

3.2 Hastening easy examples

We now experiment with deeper convolutional networks on CIFAR10, in two setups where the training examples are split into groups of varying difficulty. Additional experiments with various other choices of hyperparameters and initialization seed are reported in Appendix F.

3.2.1 Example difficulty using C-scores

In this section we quantify example difficulty using consistency scores (C-scores) (Jiang et al., 2021). Informally, C-scores measure how likely an example is to be well-classified by models trained on subsets of the dataset that do not contain it. Intuitively, examples with a high C-score share strong regularities with a large group of examples in the dataset. Formally, given a choice of model f, for each (input, label) pair (x, y) in a dataset $\mathcal{D}$, Jiang et al. (2021) defines its empirical consistency profile as:

$$\hat{C}_{\mathcal{D},n}\left(x,y\right) = \hat{\mathbb{E}}^{r}_{D \overset{n}{\sim} \mathcal{D}\setminus\{(x,y)\}}\left[\mathbb{P}\left(f\left(x;D\right)=y\right)\right], \tag{7}$$

where $\hat{\mathbb{E}}^{r}$ is the empirical average over r subsets D of size n uniformly sampled from $\mathcal{D}$ excluding (x, y), and $f(\cdot, D)$ is the model trained on D. A scalar C-score is obtained by averaging the consistency profile over various values of $n = 1, \cdots, |\mathcal{D}| - 1$. For CIFAR10 we use pre-computed scores available as https://github.com/pluskid/structural-regularity.

While training, we compute the loss separately on 10 subsets of the training set ranked by increasing C-scores deciles (e.g. examples in the last subset are top-10% C-scores), for both the training set and test set. We also train a linearized copy of this network (so as to share the same initial conditions) with $\alpha = 100$. The $\alpha = 1$ run is trained for 200 epochs (64 000 SGD iterations) whereas in order to converge the $\alpha = 100$ run is trained for 1 000 epochs (320 000 SGD iterations). Similarly to fig. 1, we normalize training progress using the mean training loss in order to compare regimes at equal progress. We check (fig. 2 top right) that the model with $\alpha = 100$ indeed stays in the linear regime during the whole training run, since all 3 linearity metrics that we report essentially remain equal to 1. By contrast, in the non-linear regime ($\alpha = 1$), a steady decrease of linearity metrics as training progresses indicates a rotation of the NTK and the representation kernel, as well as lower sign similarity of the ReLUs.

The results are shown in fig. 2 (top left). As one might expect, examples with high C-scores are learned faster during training than examples with low C-scores in both regimes. Remarkably, this effect is amplified in the non-linear regime compared to the linear one, as we can observe by comparing e.g. the top (resp. bottom) decile in light green (resp dark blue). This illustrates a relative acceleration of the non-linear regime in the direction of easy examples.

3.2.2 Example difficulty using label noise

In this section we use label noise to define difficult examples. We train a ResNet18 on CIFAR10 where 15% of the training examples are assigned a wrong (random) label. We compute the loss and accuracy independently on the regular examples with their true label, and the noisy examples whose label is flipped. In parallel, we train a copy of the initial model in the linearized regime with $\alpha = 100$. Fig. 2 bottom right shows the 3 linearity metrics during training in both regimes.

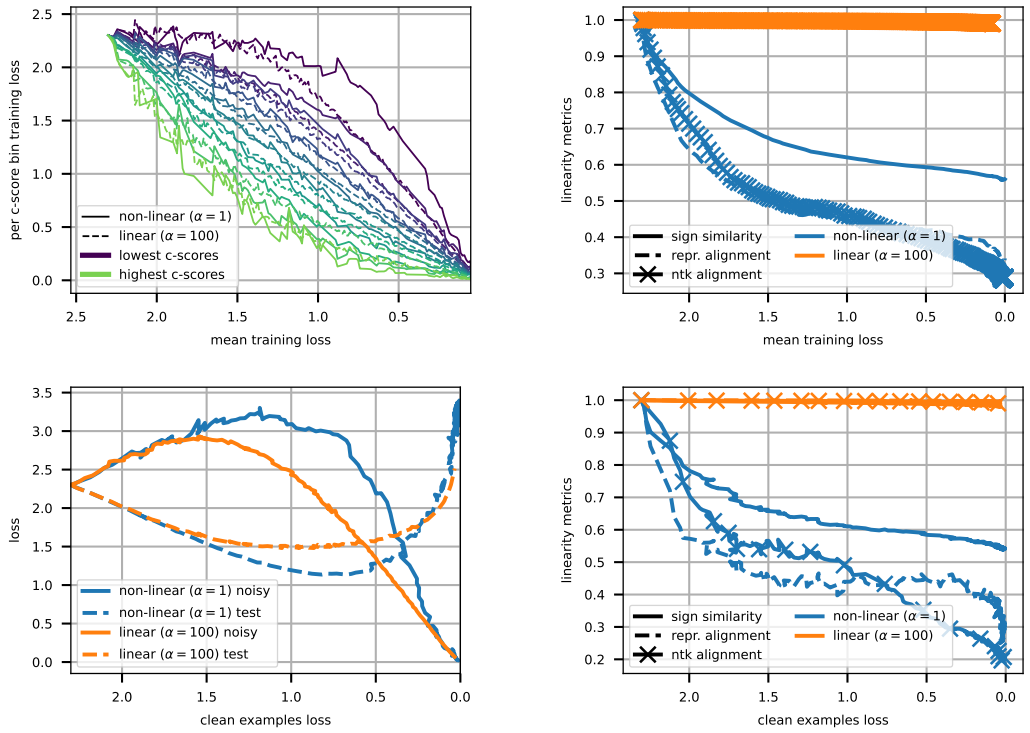

Figure 2: Starting from the same initial parameters, we train 2 ResNet18 models with $\alpha = 1$ (standard training) and $\alpha = 100$ (linearized training) on CIFAR10 using SGD with momentum. **(Top left)** We compute the training loss separately on 10 subgroups of examples ranked by their C-scores. Training progress is normalized by the mean training loss on the x-axis. Unsurprisingly, in both regimes examples with high C-scores are learned faster. Remarkably, this ranking is more pronounced in the non-linear regime as can be observed by comparing dashed and solid lines of the same color. **(Bottom left)** We randomly flip the class of 15% of the training examples. At equal progress (measured by equal clean examples loss), the non-linear regime prioritizes learning clean examples and nearly ignores noisy examples compared to the linear regime since the solid curve remains higher for the non-linear regime. Concomitantly, the non-linear test loss reaches a lower value. **(Right)** On the same training run, as a sanity check we observe that the $\alpha = 100$ training run remains in the linear regime throughout since all metrics stay close to 1, whereas in the $\alpha = 1$ run, the NTK and representation kernel rotate, and a large part of ReLU signs are flipped. These experiments are completed in Appendix F with accuracy plots for the same experiments, and with other experiments with varying initial model parameters and mini-batch order.

The results are shown in fig. 2 (bottom left). In both regimes, the training process begins with a phase where only examples with true labels are learned, causing the loss on examples with wrong labels to increase. A second phase starts (in this run) at 1.25 clean examples loss for the non-linear regime and 1.5 clean loss for the linear regime, where random labels are getting memorized. We see that the first phase takes a larger part of the training run in the non-linear regime than in the linear regime. We interpret this as the fact that the non-linear regime prioritizes learning easy examples. As a consequence, the majority of the training process is dedicated to learning the clean examples in the non-linear regime, whereas in the linear regime both the clean and the noisy labels are learned simultaneously passed the 1.5 clean loss mark. Concomitantly, comparing the sweet spot (best test loss) of the two regimes indicates a higher robustness to label noise thus a clear advantage for generalization in the non-linear regime.

3.3 Spurious correlations

We now examine setups where easy examples are those with strong correlations between their labels and some spurious feature (Sagawa et al., 2020b). We experiment with CelebA (Liu et al., 2015) and Waterbirds (Wah et al., 2011) datasets.

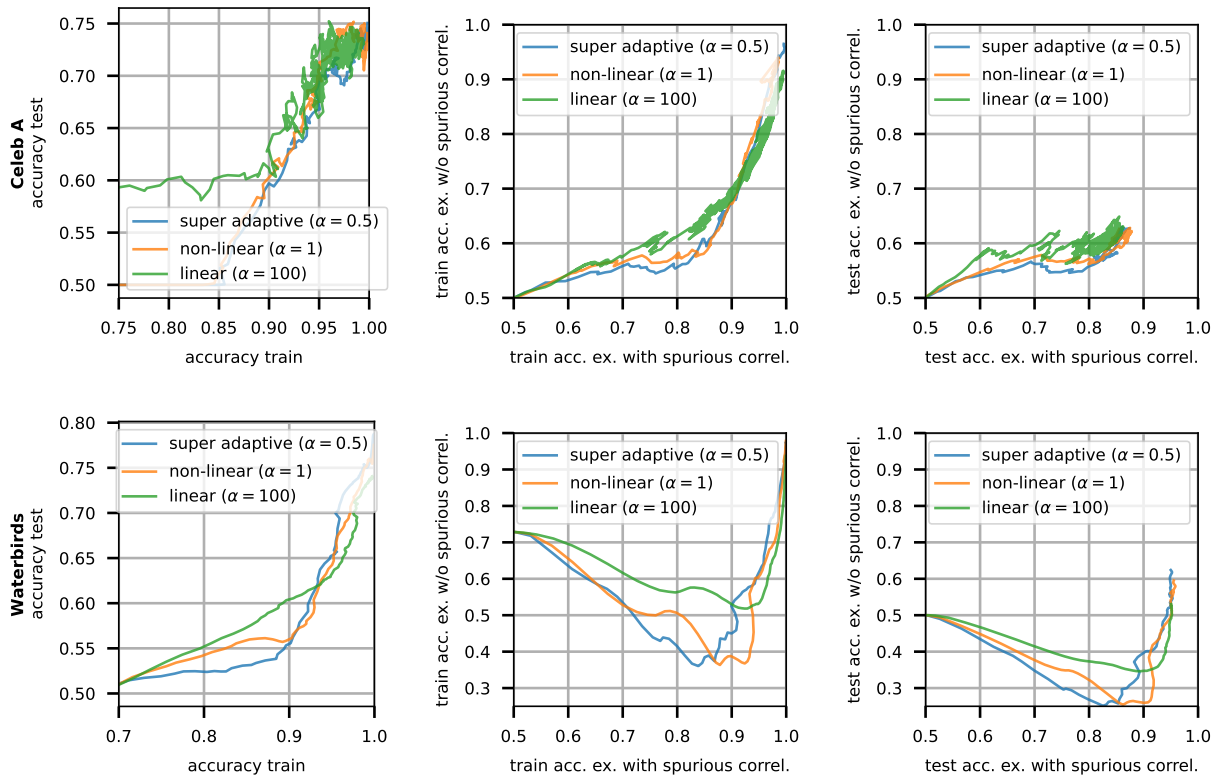

Figure 3: We visualize the trajectories of training runs on 2 spurious correlations setups, by computing the accuracy on 2 separate subsets: one with examples that contain the spurious feature (`with spurious`), the other one without spurious correlations (`w/o spurious`). On Celeb A **(top row)**, the attribute 'blond' is spuriously correlated with the gender 'woman'. In the first phase of training we observe that **(left)** the test accuracy is essentially higher for the linear run, which can be further explained by observing that **(middle)** the training accuracy for `w/o spurious` examples increases faster in the linear regime than in non-linear regimes at equal `with spurious` training accuracy. **(right)** A similar trend holds for test examples. In this first part the linear regime is less sensitive to the spurious correlation (easy examples) thus gets better robustness. **(bottom row)** On Waterbirds, the background (e.g. a lake) is spuriously correlated with the label (e.g. a water bird). **(left)** We observe the same hierarchy between the linear run and other runs. In the first training phase, the linear regime is less prone to learning the spurious correlation: the `w/o spurious` accuracy stays higher while the `with spurious` examples are learned (**(middle)** and **(right)**). These experiments are completed in fig. 12 in Appendix F with varying initial model parameters and mini-batch order.

CelebA (Liu et al., 2015) is a collection of photographs of celebrities' faces, each annotated with its attributes, such as hair color or gender. Similarly to Sagawa et al. (2020b), our task is to classify pictures based on whether the person is blond or not. In this dataset, the attribute "the person is a woman" is spuriously correlated with the attribute "the person is blond", since the attribute "blond" is over-represented among women (24% are blond) compared to men (2% are blond).

We use 20 000 examples of CelebA to train a ResNet18 classifier on the task of predicting whether a person is blond or not, using SGD with learning rate 0.01, momentum 0.9 and batch size 100. We also extract a balanced dataset with 180 (i.e. the total number of blond men in the test set) examples in each of 4 categories: blond man, blond woman, non-blond man and non-blond woman. While training progresses, we measure the loss and accuracy on the subgroups man and woman. Starting from the same initial conditions, we train 3 different classifiers with $\alpha \in \{.5, 1, 100\}$.

Waterbirds (Wah et al., 2011) is a smaller dataset of pictures of birds. We reproduce the experiments of Sagawa et al. (2020a) where the task is to distinguish land birds and water birds. In this dataset, the

background is spuriously correlated with the type of bird: water birds are typically photographed on a background such as a lake, and similarly there is generally no water in the background of land birds. There are exceptions: a small part of the dataset consists of land birds on a water background and vice versa (e.g. a duck walking in the grass).

We use 4 795 training examples of the Waterbirds dataset. Since the dataset is smaller, we start from a pre-trained ResNet18 classifier from default PyTorch models (pre-trained on ImageNet). We replace the last layer with a freshly initialized binary classification layer, and we set batch norm layers to evaluation mode[2] We train using SGD with learning rate 0.001, momentum 0.9 and minibatch size 100.

From the training set, we extract a balanced dataset with 180 examples in each of 4 groups: land birds on land background, land birds on water background, water bird on land background, and water bird on land background, which we group in two sets: `with spurious` when the type of bird and the background agree, and `w/o spurious` otherwise. While training, we measure the accuracy separately on these 2 sets. We train 3 different classifiers with $\alpha \in \{.5, 1, 100\}$.

Looking at fig. 3 left, for both the Waterbirds and the CelebA experiments we identify two phases: in the first phase the test accuracy is higher for the linear $\alpha = 100$ run than for other runs. In the second phase all 3 runs seem to converge, with a slight advantage for non-linear runs in Waterbirds. Taking a closer look at the first phase in fig. 3 middle and right, we understand this difference in test accuracy in light of spurious and non-spurious features: in the non-linear regime, the training dynamics learns the majority examples faster, at the cost of being more prone to spurious correlations. This can be seen both on the balanced training set and the balanced test set.

4 Theoretical Insights

The goal here is to illustrate some of our insights in a simple setup amenable to analytical treatment.

4.1 A simple quadratic model

We consider a standard linear regression analysis: given n input vectors $\mathbf{x}_i \in \mathbb{R}^d$ with their corresponding labels $y_i \in \mathbb{R}$, $1 \leq i \leq n$, the goal is to fit a linear function $f_{\boldsymbol{\theta}}(\mathbf{x})$ to the data by minimizing the least-squares loss $\ell(\boldsymbol{\theta}) := \frac{1}{2}\sum_i (f_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i)^2$. We will focus on a specific quadratic parametrization, which can be viewed as a subclass of two-layer networks with linear activations.

Notation We denote by $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ the matrix of inputs and by $\boldsymbol{y} \in \mathbb{R}^n$ the vector of labels. We consider the singular value decomposition (SVD),

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{M}\boldsymbol{V}^\top := \sum_{\lambda=1}^{r_X} \sqrt{\mu_\lambda}\boldsymbol{u}_\lambda \boldsymbol{v}_\lambda^\top \tag{8}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times n}, \boldsymbol{V} \in \mathbb{R}^{d\times d}$ are orthogonal and $\boldsymbol{M}$ is rectangular diagonal. r_X denotes the rank of $\boldsymbol{X}$, $\mu_1 \geq \cdots \geq \mu_{r_X} > 0$ are the non zero (squared) singular values; we also set $\mu_\lambda = 0$ for $r_X < \lambda \leq \max(n, d)$. Left and right singular vectors extend to orthonormal bases $(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n)$ and $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_d)$ of $\mathbb{R}^n$ and $\mathbb{R}^d$, respectively. In what follows we assume, without loss of generality[3], that the vector of labels has positive components in the basis $\boldsymbol{u}_\lambda$, i.e., $y_\lambda := \boldsymbol{u}_\lambda^\top \boldsymbol{y} \geq 0$ for $\lambda = 1, \cdots n$.

Parametrization We consider the following class of functions

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}, \quad \boldsymbol{\theta} := \frac{1}{2}\sum_{\lambda=1}^{d} w_\lambda^2\, \boldsymbol{v}_\lambda \tag{9}$$

[2]In order not to interfere with α scaling (see section 2), and since we consider that batch norm is a whole different theoretical challenge by itself, we chose to turn it off, and instead keep the mean and variance buffers to their pre-trained value. See appendix D for further discussion of this point.

[3]One can use the reflection invariance $\sqrt{\mu_\lambda}\boldsymbol{u}_\lambda \boldsymbol{v}_\lambda^\top = \sqrt{\mu_\lambda}(-\boldsymbol{u}_\lambda)(-\boldsymbol{v}_\lambda)^\top$ of the SVD to flip the sign of y_λ.

In this setting, the least-squares loss is minimized by gradient descent over the vector parameter $\mathbf{w} = [w_1, \cdots w_d]^\top$. Given an initialization $\mathbf{w}^0$, we want to compare the solution of the vanilla gradient (non linear) dynamics with the solution of the lazy regime, which corresponds to training the linearized function (3), given in our setting by $\bar{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}) = \bar{\boldsymbol{\theta}}^\top \mathbf{x}$ where $\bar{\boldsymbol{\theta}} = \sum_{\lambda=1}^d w_\lambda w_\lambda^0 \, \boldsymbol{v}_\lambda$.

4.2 Gradient dynamics

For simplicity, we analyze the continuous-time counterpart of gradient descent,

$$\mathbf{w}(0) = \mathbf{w}^0, \quad \dot{\mathbf{w}}(t) = -\nabla_{\mathbf{w}} \ell(\boldsymbol{\theta}(t)) \tag{10}$$

where the dot denotes the time-derivative. Making the gradients explicit and differentiating (9) yield

$$\dot{\boldsymbol{\theta}}(t) = -\boldsymbol{\Sigma}(t)(\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*), \quad \boldsymbol{\Sigma}(t) = \boldsymbol{V} \mathrm{Diag}(\mu_1 w_1^2(t), \cdots, \mu_d w_d^2(t)) \boldsymbol{V}^\top \tag{11}$$

where $\boldsymbol{\theta}^*$ is the solution of the linear dynamics.[4] Note how $\boldsymbol{\Sigma}(t)$ is obtained from the input correlation matrix $\boldsymbol{X}^\top \boldsymbol{X}$ by rescaling each eigenvalue μ_λ by the time-varying factor w_λ^2. By contrast, the lazy regime is described by the equation obtained from (11) by replacing $\boldsymbol{\Sigma}(t)$ the constant matrix $\boldsymbol{\Sigma}(0)$.

In the proposition below (proved in Appendix A), we consider the system (10, 11) initialized as $\boldsymbol{\theta}^0 := \frac{1}{2} \sum_{\lambda=1}^d (w_\lambda^0)^2 \, \boldsymbol{v}_\lambda$ where we assume that $w_\lambda^0 \neq 0$ for all λ. We denote by $\tilde{y}_\lambda$ the components of the input-label correlation vector $\boldsymbol{X}^\top \boldsymbol{y} \in \mathbb{R}^d$ in the basis $\boldsymbol{v}_\lambda$: we have $\tilde{y}_\lambda = \sqrt{\mu_\lambda} y_\lambda$ for $1 \leq \lambda \leq r_X$ and 0 when $r_X < \lambda \leq d$.

Proposition 1 *The solution of (10, 11) is given by,*

$$\boldsymbol{\theta}(t) = \sum_{\lambda=1}^d \theta_\lambda(t) \boldsymbol{v}_\lambda, \qquad \theta_\lambda(t) = \begin{cases} \dfrac{\theta_\lambda^0 \theta_\lambda^*}{\theta_\lambda^0 - e^{-2\tilde{y}_\lambda t}(\theta_\lambda^0 - \theta_\lambda^*)} & \text{if } \theta_\lambda^* \neq 0 \\ \dfrac{\theta_\lambda^0}{1 + 2\mu_\lambda \theta_\lambda^0 t} & \text{if } \theta_\lambda^* = 0 \end{cases} \tag{12}$$

By contrast, the solution in the linearized regime where $\boldsymbol{\Sigma}(t) \approx \boldsymbol{\Sigma}(0)$ is,

$$\theta_\lambda(t) = \theta_\lambda^* + e^{-\mu_\lambda \theta_\lambda^0 t}(\theta_\lambda^0 - \theta_\lambda^*) \tag{13}$$

4.3 Discussion

While the gradient dynamics (12,13) converge to the same solution $\boldsymbol{\theta}^*$, we see that the convergence rates of the various modes differ in the two regimes. To quantify this, for each dynamical mode $1 \leq \lambda \leq r_X$ such that $|\theta_\lambda^*| \neq 0$, let $t_\lambda(\epsilon), t_\lambda^{\mathrm{lin}}(\epsilon)$ be the times required for θ_λ to be ϵ-close to convergence, i.e $|\theta_\lambda - \theta_\lambda^*| = \epsilon$, in the two regimes. Substituting into (12) and (13) we find that, close to convergence $\epsilon \ll \theta_\lambda^*$ and for a small initialization, $\theta_\lambda^0 \ll \theta_\lambda^*$,

$$t_\lambda(\epsilon) = \frac{1}{\tilde{y}_\lambda} \log \frac{\theta_\lambda^*}{\epsilon \theta_\lambda^0}, \qquad t_\lambda^{\mathrm{lin}}(\epsilon) = \frac{1}{\mu_\lambda \theta_\lambda^0} \log \frac{\theta_\lambda^*}{\epsilon} \tag{14}$$

up to terms in $O(\epsilon/\theta_\lambda^*, \theta_\lambda^0/\theta_\lambda^*)$. Two remarks are in order:

Linearization impacts learning schedule. Specifically, while the learning speed of each mode depends on the components of the label vector in the non linear regime, through $\tilde{y}_\lambda = \sqrt{\mu_\lambda} y_\lambda$, it does not in the linearized one. Thus, the ratio of ϵ-convergence times for two given modes λ, λ' is $t_\lambda / t_{\lambda'} = \tilde{y}_\lambda / \tilde{y}_{\lambda'}$ in the non-linear case and $t_\lambda^{\mathrm{lin}} / t_\lambda^{\mathrm{lin}} = \mu_{\lambda'} \theta_{\lambda'}^0 / \mu_\lambda \theta_\lambda^0$ in the linearized case, up to logarithmic factors.

[4] Explicitly, $\boldsymbol{\theta}^* = \boldsymbol{\Sigma}^+ \boldsymbol{X}^\top \boldsymbol{y} + P_\perp(\boldsymbol{\theta}^0)$ where $\boldsymbol{\Sigma}^+$ is the pseudoinverse of the input correlation matrix $\boldsymbol{\Sigma} = \boldsymbol{X}^\top \boldsymbol{X}$ and $P_\perp$ projects onto the null space of $\boldsymbol{X}$. It decomposes as $\boldsymbol{\theta}^* = \sum_{\lambda=1}^d \theta_\lambda^* \boldsymbol{v}_\lambda$ in the basis $\boldsymbol{v}_\lambda$, where $\theta_\lambda^* = y_\lambda / \sqrt{\mu_\lambda}$ for $1 \leq \lambda \leq r_X$ and $\theta_\lambda^* = \theta_\lambda^0$ for $r_X < \lambda \leq d$.

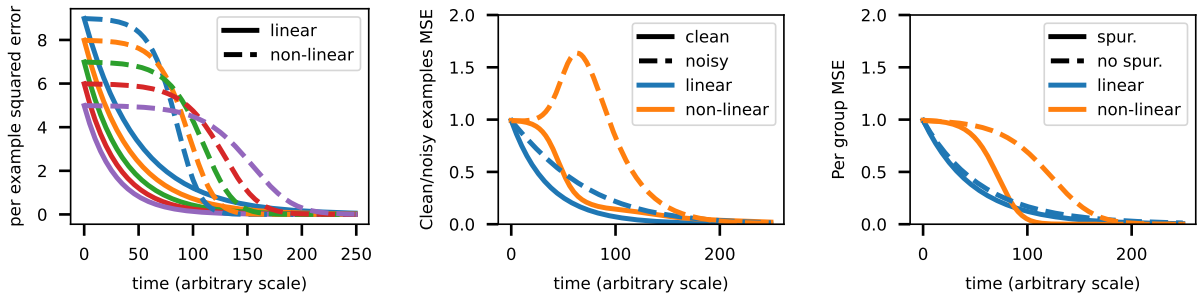

Figure 4: **(left)** Different input/label correlation (example 1): examples are learned in a flipped order in the two regimes. **(middle)** Label noise (example 2): the non-linear dynamics prioritizes learning the clean labels **(right)** Spurious correlations (example 3): the non-linear dynamics prioritizes learning the spuriously correlated feature.

Sequentialization of learning. Non-linearity, together with a vanishingly small initialization, induces a sequentialization of learning for the various modes (see also Gidel et al., 2019, Thm 2). To see this, pick a mode λ and consider, for any other mode λ', the value $\theta_{\lambda'}(t_\lambda)$ at the time $t_\lambda := t_\lambda(\epsilon)$ where λ reaches ϵ-convergence. Let us also write $\theta^0_\lambda = \sigma\tilde{\theta}^0_\lambda$ where σ is small and $\tilde{\theta}^0_\lambda = O(1)$ as $\sigma \to 0$. Then elementary manipulations show the following:

$$\theta_{\lambda'}(t_\lambda) = \frac{\theta^*_{\lambda'}\tilde{\theta}^0_{\lambda'}}{\tilde{\theta}^0_{\lambda'} + \left[\epsilon\tilde{\theta}^0_{\lambda'}/\theta^*_{\lambda'}\right]^{2\frac{\tilde{y}_{\lambda'}}{\tilde{y}_\lambda}}\sigma^{2\frac{\tilde{y}_{\lambda'}}{\tilde{y}_\lambda}-1}} =_{\sigma\to 0} \begin{cases} \theta^*_{\lambda'} & \tilde{y}_{\lambda'} > \tilde{y}_\lambda \\ 0 & \tilde{y}_{\lambda'} < \tilde{y}_\lambda \end{cases} \tag{15}$$

In words, for fixed $\epsilon > 0$ and in the limit of small initialization, the mode λ gets ϵ-close to convergence before any of the subdominant mode deviates from their (vanishing) initial value: the modes are learned sequentially.

4.4 Mode vs example difficulty

We close this section by illustrating the link between *mode* and *example difficulty* on three concrete examples of structure for the training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$. In what follows, we consider the overparametrized setting where $d \geq n$. We denote by $\boldsymbol{e}_1, \cdots \boldsymbol{e}_d$ the canonical basis of $\mathbb{R}^d$.

For each of the examples below, we consider an initialization $\mathbf{w}^0 = \sqrt{\sigma}[1 \cdots 1]^\top$ with small $\sigma > 0$ and the corresponding solutions in Prop. 1 for both regimes.

Example 1 We begin with a rather trivial setup where each mode corresponds to a training example. It will illustrate how non-linearity can *reverse* the learning order of the examples. Given a sequence of strictly positive numbers $\mu_1 \geq \cdots \mu_n > 0$, we consider the training data,

$$\mathbf{x}_i = \sqrt{\mu_i}\boldsymbol{e}_i, \quad y_i = \mu_{n-i+1}/\sqrt{\mu_i}, \qquad 1 \leq i \leq n \tag{16}$$

In the linearized regime, $f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i)$ converges to y_i at the linear rate $\sigma\mu_i$; the model learns faster the examples with higher μ_i, hence with *lower* index i. In the non linear regime, the examples are learned sequentially according to the value $\tilde{y}_i = \sqrt{\mu_i}y_i = \mu_{n-i+1}$, hence from *high to low* index i. Thus in this setting, linearization flips the learning order of the training examples (see Fig. 4 left).

In examples 2 and 3 below we aim at modelling situations where the labels depend on low-dimensional (in this case, one dimensional) projections on the inputs (e.g. an image classification problem where the labels mainly depend on the low frequencies of the image). These two examples mirror the two sets of experiments in Sections 3.2.2 and 3.3, respectively.

Example 2 We consider a simple classification setup on linearly separable data with label noise. Here we assume $d > n$. Conditioned on a set of binary labels $y_i = \pm 1$, the inputs are given by

$$\mathbf{x}_i = \kappa_i y_i \boldsymbol{e}_1 + \eta \boldsymbol{e}_{i+1} \quad 1 \leq i \leq n \tag{17}$$

where $\kappa_i = \pm 1$ is some 'label flip' variable and $\eta > 0$. We assume we have q 'noisy' examples with flipped labels, i.e. $\kappa_i = -1$, where $1 \leq q < \lceil n/2 \rceil$.

The SVD of the feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ defined by (17) can be made explicit. In particular, the top left singular vector is $\boldsymbol{u}_1 = \bar{\boldsymbol{y}}/\sqrt{n}$, where $\bar{\boldsymbol{y}} \in \mathbb{R}^n$ denotes the vector of noisy labels $\bar{y}_i := \kappa_i y_i$. This singles out a dominant mode $\boldsymbol{y}_1 := (\boldsymbol{u}_1^\top \boldsymbol{y})\boldsymbol{u}_1$ of the label vector $\boldsymbol{y}$ that is *learned first* by the non-linear dynamics. Explicitly,

$$\boldsymbol{y}_1 = (1 - \frac{2q}{n})\bar{\boldsymbol{y}} \tag{18}$$

For a small noise ratio $q/n \ll 1$, this yields $y_{1i} \approx y_i$ for clean examples and $-y_i$ for noisy ones: fitting the dominant mode amounts to learning the clean examples – while assigning the wrong label to the noisy ones. This is illustrated in Fig 4 (middle plot).

Example 3 We consider a simple spurious correlation setup (Sagawa et al., 2020b), obtained by adding to (17) a 'core' feature that separates all training points. Here we assume $d > n + 1$. Conditioned on a set of binary labels $y_i = \pm 1$, the inputs are given by

$$\mathbf{x}_i = \kappa_i y_i \boldsymbol{e}_1 + \lambda y_i \boldsymbol{e}_2 + \eta \boldsymbol{e}_{i+2}, \quad 1 \leq i \leq n \tag{19}$$

for some binary variable $\kappa_i = \pm 1$, scaling factor $\lambda \in (0, 1)$, and $\eta > 0$. Given $1 \leq q < \lceil n/2 \rceil$, we assume we have a majority group of $n - q$ training examples with $\kappa_i = 1$, whose label is spuriously correlated with the spurious feature $\boldsymbol{e}_2$, and a minority group of q examples with $\kappa_i = -1$.

The analysis is similar as in Example 2. For small noise ratio and scaling factor λ, the non-linear dynamics enhances the bias towards fitting first the majority group of ('easy') examples with spuriously correlated labels. This illustrates an increased sensitivity of the non linear regime to the spurious feature – at least in the first part of training. This is shown in Fig 4 (right plot).

5 Conclusion

The recent emphasis on the lazy training regime, where deep networks behave as linear models amenable to analytical treatment, begs the question of the specific mechanisms and implicit biases which differentiates it from the full-fledged feature learning regime of the algorithms used in practice. In this paper, we investigated the comparative effect of the two regimes on subgroups of examples based on their difficulty. We provided experiments in various setups suggesting that easy examples are given more weight in non-linear training mode (deep learning) than in linear training, resulting in a comparatively higher learning speed for these examples. We illustrated this phenomenon across various ways to quantify examples difficulty, through c-scores, label noise, and correlations to some easy-to-learn spurious features. We complemented these empirical observations with a theoretical analysis of a quadratic model whose training dynamics is tractable in both regimes. We believe that our findings makes a step towards a better understanding of the underlying mechanisms that drive the good generalization properties observed in practice.

Broader Impact Statement

This work proposes to improve our understanding of the mechanisms behind deep learning models training. There is no clear direct application of these theoretical observations to create harmful tools, however as for any technology, machine learning models can be used for malicious intentions. We also acknowledge that deep learning uses a significant amount of compute capacity when scaled to global tech companies, which in some places is powered by carbon emitting power plants, thus contributes to global warming.

References

Atish Agarwala, Jeffrey Pennington, Yann Dauphin, and Sam Schoenholz. Temperature check: theory and practice for training models with softmax-cross-entropy losses, 2020. URL https://arxiv.org/abs/2010.07344.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/allen-zhu19a.html.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dbc4d84bfcfe2284ba11beffb853a8c4-Paper.pdf.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 233–242. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arpit17a.html.

Robert John Nicholas Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=WWRBHhH158K.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2269–2277. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/baratin21a.html.

Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems 32*, pp. 4761–4771. 2019. URL http://papers.nips.cc/paper/8723-the-convergence-rate-of-neural-networks-for-learned-functions-of-different-frequencies.pdf.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In *Advances in Neural Information Processing Systems 32*, pp. 12893–12904. 2019. URL http://papers.nips.cc/paper/9449-on-the-inductive-bias-of-neural-tangent-kernels.pdf.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 3036–3046. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz Kandola. On kernel-target alignment. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/file/1f71e393b3809197ed66df836fe833e5-Paper.pdf.

Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 2019a. URL http://proceedings.mlr.press/v97/du19c.html.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=S1eK3i09YQ.

Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99, 1993.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5850–5861. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/405075699f065e43581f27d67bb68478-Paper.pdf.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, nov 2020. doi: 10.1088/1742-5468/abc4de. URL https://doi.org/10.1088%2F1742-5468%2Fabc4de.

Thomas George. NNGeometry: Easy and Fast Fisher Information Matrices and Neural Tangent Kernels in PyTorch, February 2021. URL https://doi.org/10.5281/zenodo.4532597.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f39ae9ff3a81f499230c4126e01f421b-Paper.pdf.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949 – 986, 2022. doi: 10.1214/21-AOS2133. URL https://doi.org/10.1214/21-AOS2133.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248 [cs.LG]*, 2020.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NIPS*, pp. 8571–8580. 2018.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C. Mozer. Characterizing structural regularities of labeled data in overparameterized models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5034–5044. PMLR, 2021. URL http://proceedings.mlr.press/v139/jiang21k.html.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Dmitry Kopitkov and Vadim Indelman. Neural spectrum alignment: Empirical study. In *International Conference on Artificial Neural Networks*, pp. 168–179. Springer, 2020.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Eran Malach, Pritish Kamath, Emmanuel Abbe, and Nathan Srebro. Quantifying the benefit of using differentiable learning over tangent kernels. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7379–7389. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/malach21a.html.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1806579115. URL https://www.pnas.org/content/115/33/E7665.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *ICLR workshop track*, 2015.

Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34, 2021.

Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (4):044001, 2021.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=ryxGuJrFvS.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pp. 8346–8356. PMLR, 2020b.

Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT)*, volume PMLR 99, pp. 2667–2690, 2019.

Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*, 2014.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Francis Williams, Matthew Trager, Daniele Panozzo, Claudio Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1f6419b1cbe79c71410cb320fc094775-Paper.pdf.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

Greg Yang and Edward J. Hu. Tensor programs IV: feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 2021. URL http://proceedings.mlr.press/v139/yang21c.html.

Greg Yang and Hadi Salman. A fine grained spectral perspective on neural networks. *arxiv preprint arXiv:1907.10599[cs.LG]*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Xiao Zhang, Haoyi Xiong, and Dongrui Wu. Rethink the connections among generalization, memorization, and the spectral bias of dnns. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 3392–3398. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/467. URL https://doi.org/10.24963/ijcai.2021/467. Main Track.

A Details on the theoretical analysis

We provide some technical details for the statements and proposition of Section 4.

A.1 Gradient dynamics

We first derive the equations (11). Using twice the chain rule yields

$$\dot{\boldsymbol{\theta}} = \sum_{\lambda=1}^{d} \dot{w_\lambda} \nabla_{w_\lambda} \boldsymbol{\theta} = -\sum_{\lambda=1}^{d} \nabla_{w_\lambda} \ell(\boldsymbol{\theta}) \nabla_{w_\lambda} \boldsymbol{\theta} = -\sum_{\lambda=1}^{d} \left(\nabla_{w_\lambda} \boldsymbol{\theta}^\top \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) \right) \nabla_{w_\lambda} \boldsymbol{\theta} \tag{20}$$

By (9), we have $\nabla_{w_\lambda} \boldsymbol{\theta} = w_\lambda \boldsymbol{v}_\lambda$; moreover

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \boldsymbol{\Sigma}\boldsymbol{\theta} - \boldsymbol{X}^\top \boldsymbol{y} = \boldsymbol{\Sigma}(\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) \tag{21}$$

where $\boldsymbol{\Sigma} = \boldsymbol{X}^\top \boldsymbol{X}$ is the input correlation matrix and $\boldsymbol{\theta}^* := \boldsymbol{\Sigma}^+ \boldsymbol{X}^\top \boldsymbol{y} + P_\perp(\boldsymbol{\theta}^0)$ is the solution of the linear dynamics. Substituting into (20)gives,

$$\dot{\boldsymbol{\theta}}(t) = -\sum_{\lambda=1}^{d} w_\lambda^2(t) \boldsymbol{v}_\lambda \boldsymbol{v}_\lambda^\top \boldsymbol{\Sigma}(\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) := -\boldsymbol{\Sigma}(t)(\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*) \tag{22}$$

Since $\boldsymbol{\Sigma}$ is diagonal in the basis $\boldsymbol{v}_\lambda$, $\boldsymbol{\Sigma}(t)$ is obtained from $\boldsymbol{\Sigma}$ by rescaling each of its eigenvalues μ_λ by the time varying factor $w_\lambda^2 = 2\theta_\lambda$.

A.2 Proof of Proposition 1

The system (22) is diagonal in the basis $\boldsymbol{v}_\lambda$, so the dynamics decouples for the parameter components θ_λ in this basis. In fact we have,

$$\boldsymbol{\theta} = \sum_{\lambda=1}^{d} \theta_\lambda \boldsymbol{v}_\lambda, \qquad \dot{\theta}_\lambda = -2\mu_\lambda \theta_\lambda (\theta_\lambda - \theta_\lambda^*) \tag{23}$$

The above equations can be put in the form

$$\frac{\dot{\theta}_\lambda}{\theta_\lambda(\theta_\lambda - \theta_\lambda^*)} = -2\mu_\lambda \tag{24}$$

We distinguish two cases (note that it follows from our assumptions that $\theta_\lambda^* \geq 0$ for all λ):

Case 1: $\theta_\lambda^* = 0$. In this case, (24) becomes

$$\frac{d}{dt}\left[-\frac{1}{\theta_\lambda}\right] = -2\mu_\lambda \quad \Rightarrow \quad \frac{1}{\theta_\lambda(t)} = 2\mu_\lambda t + c \tag{25}$$

where the constant c is fixed by the initial condition $\theta_\lambda(0) = \theta_\lambda^0$ to be $c = 1/\theta_\lambda^0$. Thus,

$$\theta_\lambda(t) = \frac{\theta_\lambda^0}{1 + 2\mu_\lambda \theta_\lambda^0 t} \tag{26}$$

Case 2: $\theta_\lambda^* > 0$. We also focus on the case where $0 < \theta_\lambda^0 < \theta_\lambda^*$; it can be checked (e.g. by a post hoc inspection on the solution) that this implies $0 < \theta_\lambda < \theta_\lambda^*$ for all t. In this case, (24) can be written as

$$\frac{1}{\theta_\lambda^*}\left(\frac{-\dot{\theta}_\lambda}{\theta_\lambda^* - \theta_\lambda} - \frac{\dot{\theta}_\lambda}{\theta_\lambda}\right) = -2\mu_\lambda \tag{27}$$

and hence

$$\frac{d}{dt}\left[\log(\theta_\lambda^* - \theta_\lambda) - \log(\theta_\lambda)\right] = -2\mu_\lambda \theta_\lambda^* \quad \Rightarrow \quad \frac{\theta_\lambda^* - \theta_\lambda}{\theta_\lambda} = c e^{-2\mu_\lambda \theta_\lambda^* t} \tag{28}$$

where the constant $c > 0$ is fixed by the initial condition to be $c = \frac{\theta_\lambda^* - \theta_\lambda^0}{\theta_\lambda^0}$. Solving for θ_λ finally gives

$$\theta_\lambda(t) = \frac{\theta_\lambda^0 \theta_\lambda^*}{\theta_\lambda^0 - e^{-2\tilde{y}_\lambda t}(\theta_\lambda^0 - \theta_\lambda^*)} \tag{29}$$

where we substituted $\tilde{y}_\lambda := \sqrt{\mu_\lambda} y_\lambda = \mu_\lambda \theta_\lambda^*$. The case where $0 < \theta_\lambda^* < \theta_\lambda^0$ is similar and yields the same expression.

B Code

Code for reproducing the experiments is available at: https://anonymous.4open.science/r/lin_vs_nonlin-B3F7/.

In the `linearization_utils.py` file there are 2 new contributed classes extensively used in the project:

- `LinearizationProbe` implements the linearization metrics (end of section 2): sign similarity, NTK alignment using NNGeometry (George, 2021) and representation kernel alignment;
- `ModelLinearKnob` implements prediction of a model parameterized by the α scaling (section 2).

C Experimental details

Unless specified otherwise, all model parameters are initialized using the default Pytorch initialization.

C.1 CIFAR10 with C-scores

We train a ResNet18 with SGD with learning rate 0.01, momentum 0.9 and bach size 125. In order to use pre-computed C-scores even for a held-out test set, we split the CIFAR10 dataset into 40 000 train examples and 10 000 test examples.

C.2 CIFAR10 with noisy examples

We train a ResNet18 with SGD with learning rate 0.01, momentum 0.9 and bach size 125.

D A note on batch norm and α scaling

We chose to refrain from using batch norm in our experiments since it adds unnecessary complexity for analysis, and deep models without batch norm already show intriguing generalization properties that are not fully understood by current theory. Note that most theoretical advances currently focus on models that do not use batch norm.

In addition, the α scaling (section 2) is not well defined in the presence of batch norm in training mode. Indeed, in training mode, batch norm works by evaluating the mean and variance using examples of the current mini-batch. When computing $\alpha\left(f_{\boldsymbol{\theta}}(\mathbf{x}) - f_{\boldsymbol{\theta}_0}(\mathbf{x})\right)$ for large α, a slight change in parameters is boosted by α to produce a large change in function space. This is expected, and this is balanced by tiny values of the learning rate (shrinked by $\frac{1}{\alpha^2}$). But small changes in mean and variance are also boosted, and this is probably not expected. For this reason, we chose to limit the scope of the current paper to models without batch norm.

As an exception, in the Waterbirds experiment of section 3.3, we used a pre-trained ResNet18 network, that includes batch norm layers. We chose to fine-tune in evaluation mode, where the mean and variance are not computed using examples of the mini-batch, but are instead constant, kept at their pre-trained values. This way, all other parameters (including batch norm's γ and β) are "standard" parameters that can be linearized.

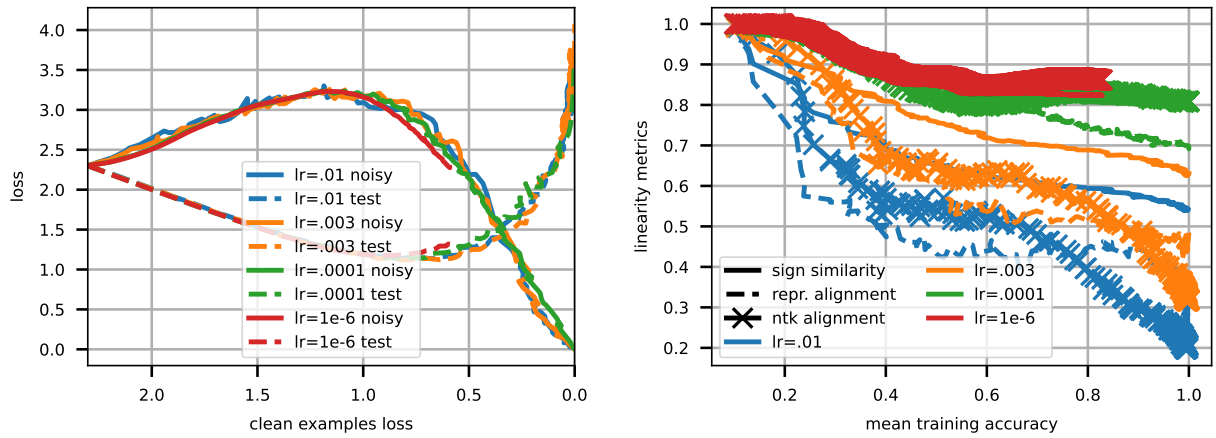

Figure 5: same as fig. 2, with $\alpha = 1$ and varying learning rates in $\{0.01, 0.003, 10^{-4}, 10^{-6}\}$. In this experiment, we rule out the role of the learning rate in learning speed of easy/difficult examples, since regardless of the learning rate, all runs follow the same trajectory as measured by noisy examples accuracy during training, and test examples accuracy during training. This shows that modulating the learning rate plays a different role as the α scaling.

E Interplay between learning rate and α scaling

It has been observed in (Li et al., 2019) that the initial learning rate used during training plays an important role in the final outcome, in that models trained with large initial learning rate usually get better generalization on a test set. In order to rule out the role of the learning rate in our easy/difficult examples setup, we compared models trained with $\alpha = 1$ and varying learning rate in a broad range (4 orders of magnitude).

Figure 5 shows our result on the noisy vs clean examples experiment (similar to figure 2 in the main body). Our plots are also normalized by clean example loss, even if runs with small learning rate require many more training iterations to attain a similar progress (measured in e.g. training loss). We observe that even if using a smaller learning rate keeps the training dynamics in a more linearized regime (on the right), it is still less linearized than our runs scaled by α: compare the $\text{lr} = 10^{-6}$ to the $\alpha = 100$ run e.g., since both use the same learning rate once rescaled by α, but the linearity metrics stay close to 1 in the α scaled run, whereas even with this tiny learning rate, the $\text{lr} = 10^{-6}$ run in 5 is in a slightly non-linear regime as the linearity metrics drop to 0.9. When looking at figure 5 left, we also see that both the training accuracy on noisy examples, and the test accuracy seem to follow the same trajectory regardless of the learning rate, at least in the first part of training, whereas in figure 2, modulating the value of α causes the trajectories to diverge.

F Additional experiments

In order to verify that our findings also hold true on other architectures, we reproduce the CIFAR10 experiments that use a ResNet18 network, with a VGG11 network. Figure 9 presents results on the C-scores experiment, and figure 8 presents results on the noisy example experiment.

In figure 7, figure 12 and figure 13, we vary the network initial parameters and SGD minibatch in order to check for variability, and in order to verify that our results are not cherry-picked.

In figure 11, we also include the accuracy plots corresponding to the experiments in sections 3.2.2 and 3.2.1 and presented in figure 2. In figure 10, we plot the difference between non-linear and linear loss for the same experiments.

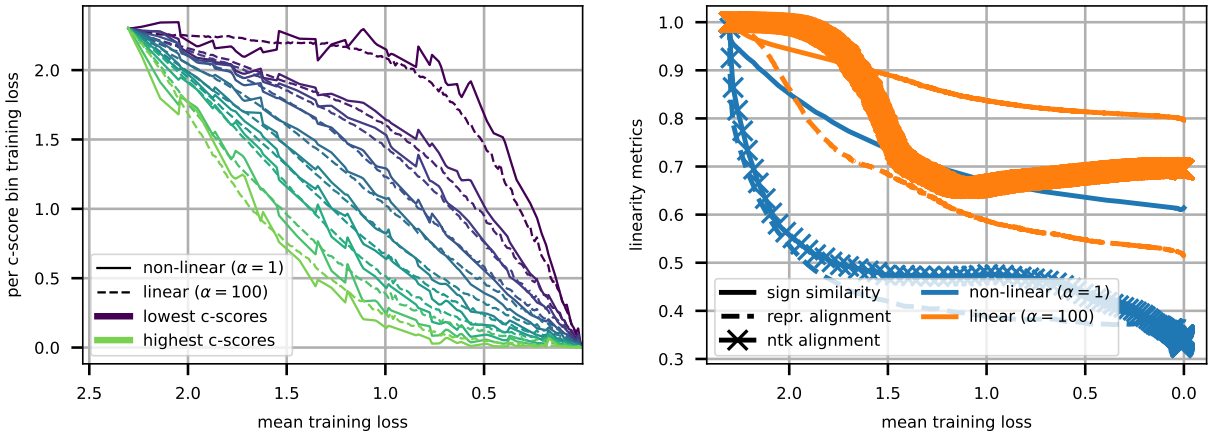

Figure 6: same as fig. 2, with a VGG11 network instead

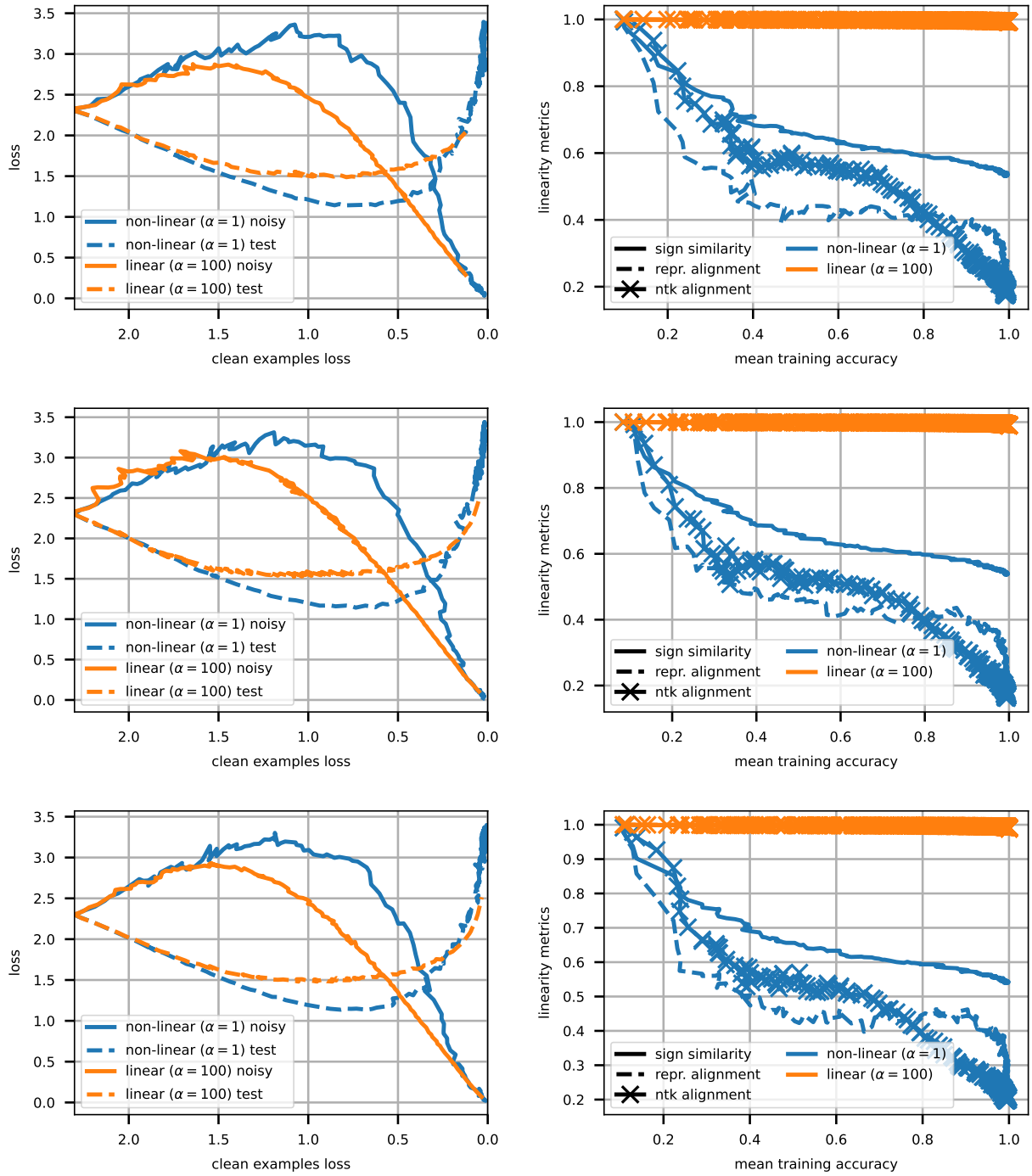

Figure 7: same as fig. 2, varying seed (model initialization and mini-batch order)

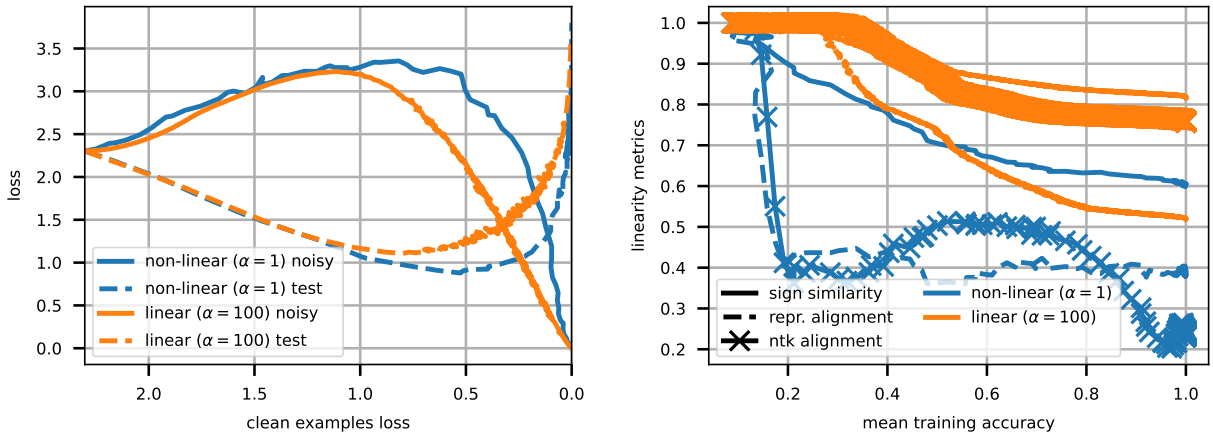

Figure 8: same as fig. 2, with a VGG11 network instead

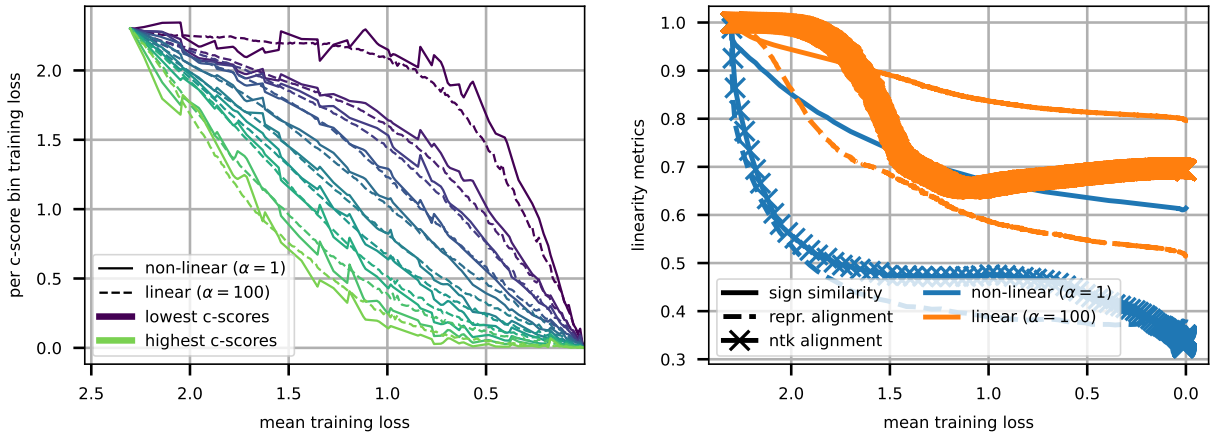

Figure 9: same as fig. 2, with a VGG11 network instead

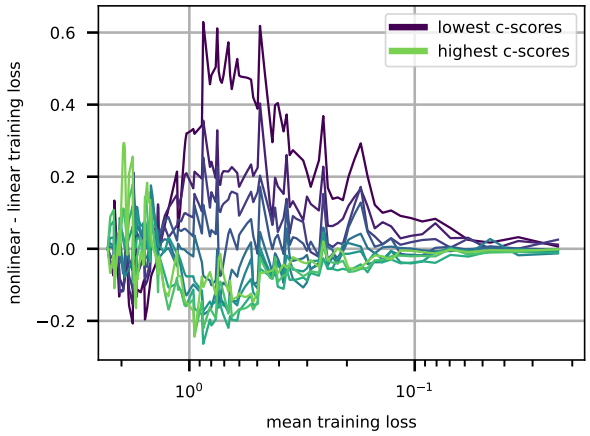

Figure 10: same as fig. 2, but we instead plot the differences between non-linear and linear regimes of the loss computed on groups of examples ranked by C-Score. Similarly to figure 2, we observe that at equal training loss (on the x-axis), the loss on high C-score examples is lower for the non-linear regime, and conversely for low C-score examples.

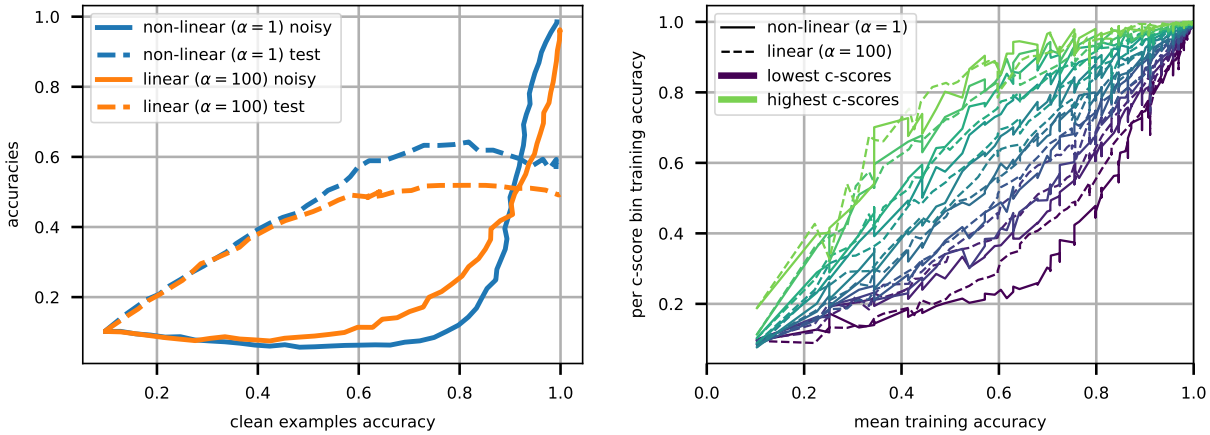

Figure 11: Accuracy plots for the experiments of fig. 2. We observe similar trends by looking at the accuracy: **(left)** the non-linear run starts memorizing noisy (difficult) examples later in training than the linear run (solid curve), while simultaneously the test accuracy reaches a higher value for the non-linear run. **(right)** Lower C-score examples are learned comparatively faster in the linear regime, whereas learning curves are more spread in the non-linear regime which indicates a comparatively higher hierarchy between learning groups of examples ranked by their C-score.

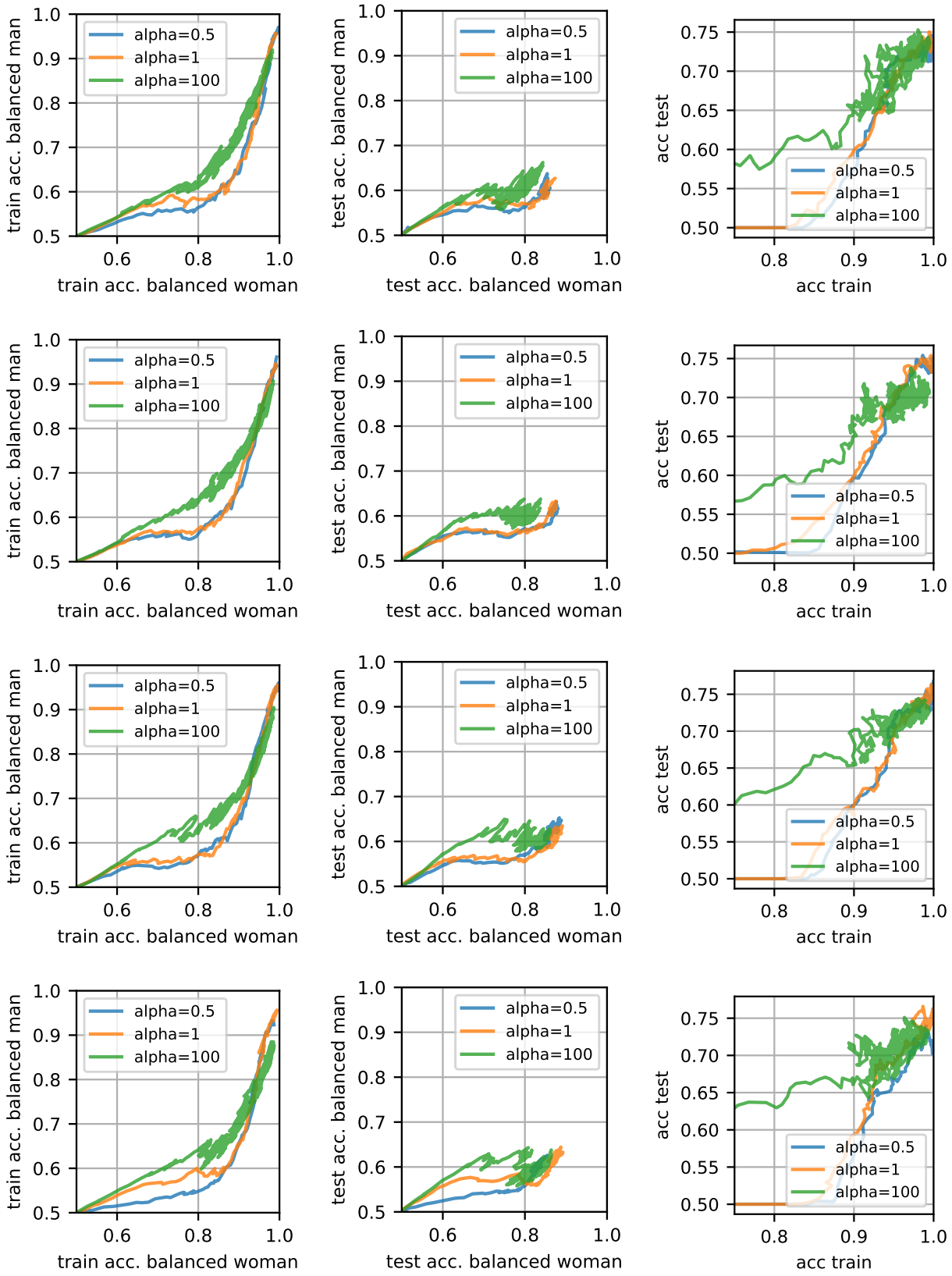

Figure 12: same as fig 3 with varying seed (model initialization and minibatch order)

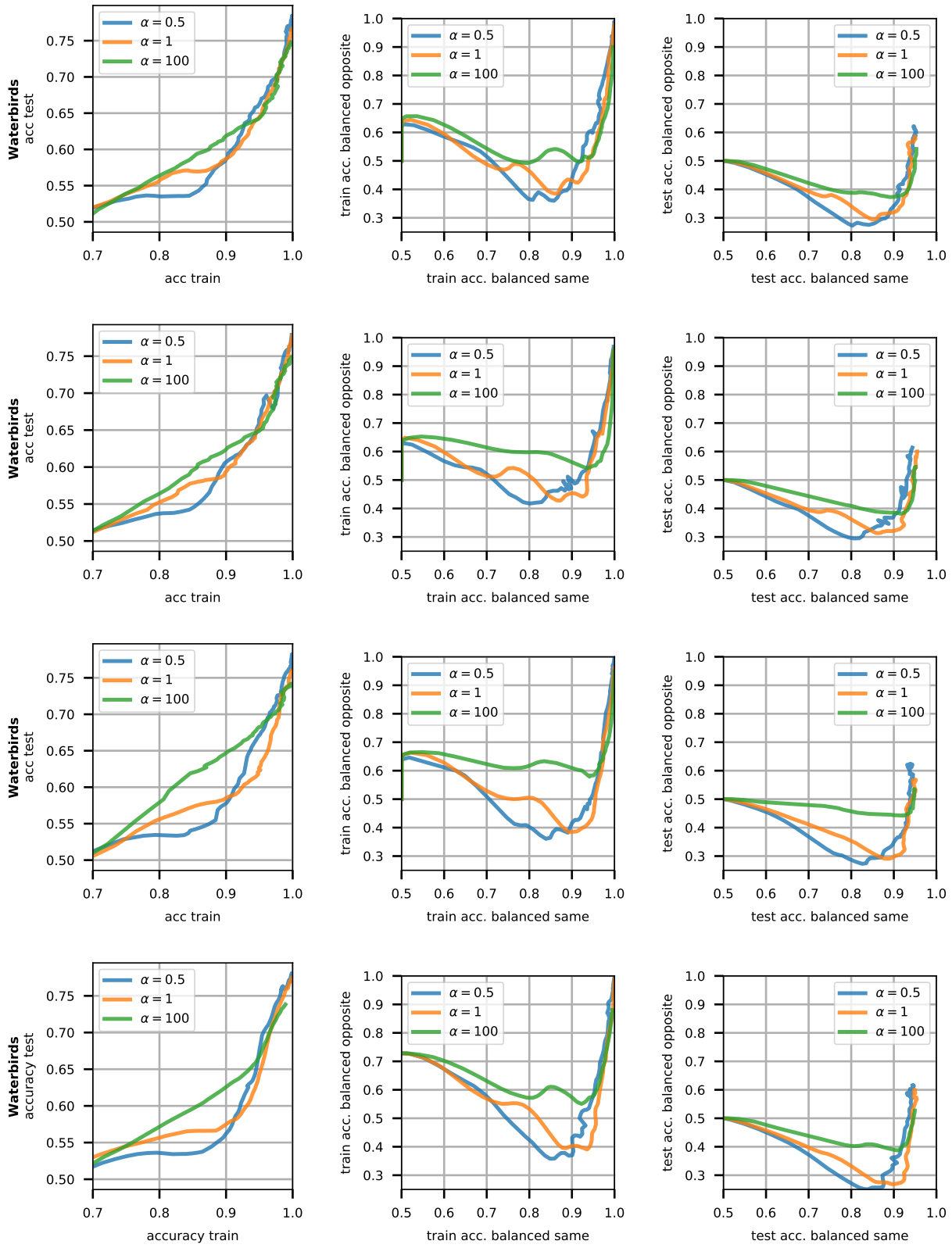

Figure 13: same as fig 3 with varying seed (model initialization and minibatch order)