Imitating Deep Learning Dynamics via Locally Elastic Stochastic Differential Equations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Understanding the training dynamics of deep learning models is perhaps a necessary step toward demystifying the effectiveness of these models. In particular, how do data from different classes gradually become separable in their feature spaces when training neural networks using stochastic gradient descent? In this paper, we model the evolution of features during deep learning training using a set of stochastic differential equations (SDEs) that each corresponding to a training sample. As a crucial ingredient in our modeling strategy, each SDE contains a drift term that reflects the impact of backpropagation at an input on the features of all samples. Our main finding uncovers a sharp phase transition phenomenon regarding the *intra-class* impact: if the SDEs are *locally elastic* in the sense that the impact is more significant on samples from the same class as the input, the features of training data become linearly separable—meaning vanishing training loss; otherwise, the features are not separable, no matter how long the training time is. In the presence of local elasticity, moreover, an analysis of our SDEs shows the emergence of a simple geometric structure called neural collapse of the features. Taken together, our results shed light on the decisive role of local elasticity underlying the training dynamics of neural networks. We corroborate our theoretical analysis with experiments on a synthesized dataset of geometric shapes and on CIFAR-10.

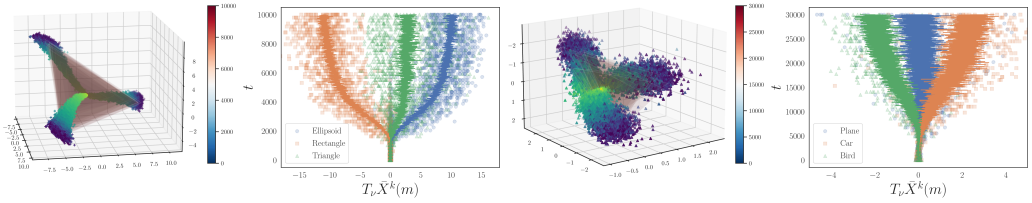
1 Introduction

Deep learning models have achieved significant empirical success in the past decade across a wide spectrum of domains spanning computer vision, natural language processing, and reinforcement learning [LBH15, SHM+16, YHPC18]. An interesting reality along with this remarkable achievement is the substantial difficulty in understanding deep neural networks, which is evidenced by the fact that almost all important advances concerning architecture design and optimization for deep learning are based on heuristics, without much input from a theoretical perspective [HZRS16, DCLT18, IS15, KB14].

Toward opening these black-box models and unveiling their formidable details, an important step is to quantitatively understand the impacts of backpropagation in deep learning training. While there is a continued effort to demystify how simple optimization methods give rise to impressive generalization performance, for example, [ZBH+16, KMN+16, ADH+19], this is by no means an easy problem, perhaps to due to the daunting nonconvex nature of neural networks. Accordingly, for near-term purposes, a more practical approach is to take a phenomenological viewpoint by relating some simple empirical patterns to the effectiveness of deep learning models.

In this spirit, in this paper we are interested in how data from different classes gradually become separable in their feature space by repetitively calling backpropagation. From a phenomenological viewpoint, one can address this question by first analyzing the impact of a single update using

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) GEOMNIST in $\mathbb{R}^3$. (b) GEOMNIST in $\mathbb{R}$. (c) CIFAR in $\mathbb{R}^3$. (d) CIFAR in $\mathbb{R}$.

Figure 1: **Separation of features (logits).** GEOMNIST dataset ((a)—(b))) and CIFAR dataset ((c)—(d)) trained on a variant of the ALEXNET model. Separation in $\mathbb{R}$ was done by projecting to $\nu \in \mathbb{R}^3$, which is set to be $\bar{\boldsymbol{X}}^1(T) - \bar{\boldsymbol{X}}^2(T)$ for a large T.

stochastic gradient on the performance of the neural networks. More precisely, imagine that the gradient is evaluated at an image of a cat, how does the hidden representation of another image—say, an image of another cat or an image of a plane—evolve because of the backpropagation. Recent work ansers this question by introducing a phenomenon called local elasticity, which, roughly speaking, says that the impact is in general larger on a similar sample (an image of another cat) than on a dissimilar sample (an image of a plane) [HS19].

Following this observation, we propose a model that captures the interaction between different training samples during deep learning training using a set of stochastic differential equations (SDEs) that reflects local elasticity in neural networks. Characterizing the intra-class and inter-class effects is an essentialingredient of our modeling strategy – each SDE contains a drift term that imitates the impact of backpropagation at a specific training data on the features of all samples.

Our main finding uncovers a sharp phase transition phenomenon regarding the *intra-class* and *inter-class* impact: if the SDEs are *locally elastic* in the sense that the impact is more significant on samples from the same class as the input (intra-class effect is strictly greater than inter-class effect), the features of training data is guaranteed to be linearly separable—meaning vanishing training loss; otherwise, the features are not separable, no matter how long the training time is. This result provides a convincing theoretical evidence to the conjecture of the presence of *local elasticity* in deep learning [HS19]. Our model is also quite accurate in simulating the feature dynamic of deep learning. As can be seen in Figure 1, the dynamic of predicted logits is pretty close to the real dynamic of deep learning on both synthetic and real datasets, indicating a well-suited model for theoretical and practical purposes. Moreover, in the presence of local elasticity, our SDEs also predict the emerge of a simple geometric structure called neural collapse of the features [PHD20].

Taken together, our results shed light on the decisive role of local elasticity underlying the training dynamics of neural networks. We corroborate our theoretical analysis with experiments on a synthesized dataset of geometric shapes as well as on CIFAR-10. The experimental evidence consistently backs our model and provides new insights into the dynamics of deep learning trainng.

1.1 Related Work

Dynamics in deep neural nets Many properties of linear deep neural nets are relatively well understood, such the loss landscapes [Kaw16], trajectory-based convergence [ACGH18, DH19], and implicit acceleration [ACH18]. Exact solutions of the training dynamics can be obtained in certain initialization schemes [SMG13, SMG19, LG18]. When there is nonlinearity, various assumptions are usually made. [SS96, GAS+19] studied the dynamics of shallow neural nets with non-linearity; the NTK literature ([JGH18, ADH+19, DHP+19]); linearize the network function of an infinitely wide neural net at initialization, which is similar to that of deep Gaussian process literature ([DL13, HJ15, LBN+17]), a treatise comparing and contrasting them can be found in [Yan19]. Although being approximations, NTKs are usually used when studying optimization trajectories of NN such as [LL18]. also appears implicitly in many works when studying the optimization trajectories of NN training [SHN+18, AZLS19, DLL+19, JT19, CCGZ20].

SGD as SDEs in Neural Nets The study of dynamics or trajectories of weights in deep neural nets via SDEs relies on more precise characterization of stochasticity [LTW17, MHB17]. Built on top of this formalism, [SKYL17] studied the trade-off between batch size and learning rate, [JKA+17], analyzed factors influencing the quality of local minima, [CS18] studied the behaviour of the SGD

near local minima, [SSJ20] studied the effect of learning rates. Although SGD-SDE approximation requires an infinitesimal learning rate, [LMA21] verifies the SDE approximation can be meaningful in practical settings and obtained necessary conditions for the validate of such approximation.

Local Elasticity and Phenomenological Models Local elasticity is proposed in [HS19] as a phenomenological approach to reasoning the behaviors of neural networks and this phenomenon has inspired several works on generalization bounds [DHS20], and improving on the neural tangent kernel (NTK) [CHS20].

2 Binary Separation via LE-SDE

2.1 Notations and assumptions

Throughout the entire paper, we work with the following setup and assumptions. For ease of reading, vectors and matrices are written in boldface; and we denote by $[n]$ the set $\{1, \ldots, n\}$.

Classification problem. Consider a K-class classification problem, where $K \geq 2$ classes each has n training examples. Denote by $\boldsymbol{z}_i^k \in \mathbb{R}^d$ the i-th sample of the k-th class, and $y_i^k \in [K]$ its label, where $i \in [n], k \in [K]$. A neural net is a function $f : \mathbb{R}^d \to \mathbb{R}^K$ that maps the samples to logits (the pre-activation of the softmax).

Feature vectors. We denote by $\boldsymbol{X}_i^k(m) \in \mathbb{R}^p$ a p-dimensional feature of the i-th sample in the k-th class learned by the neural net at iteration m, for example, it can be the logit or the output of the second-to-last layer. Assume the initial values $\boldsymbol{X}_i^k(0)$, are i.i.d. samples from some distribution for each $i \in [n], k \in [K]$. We use $i, j \in [n]$ as indices for an individual sample, and $k, l \in [K]$ for classes. We use the capital letters J and L to indicate random samples from $\mathrm{Unif}([n])$ and $\mathrm{Unif}([K])$, respectively.

Training dynamics. We model the training dynamics in neural nets under SGD with an emphasis on *local elasticity*. At the m-th iteration, it samples the J_m-th sample from the L_m-th class, where $J_m \sim \mathrm{Unif}([n])$ and $L_m \sim \mathrm{Unif}([K])$. Training on $\boldsymbol{z}_{I_m}^{J_m}$ exerts an effect on the feature of another data sample $\boldsymbol{z}_i^k$ in the form of

$$\boldsymbol{X}_i^k(m+1) - \boldsymbol{X}_i^k(m) = h \cdot E_{k,L_m}(m) \boldsymbol{X}_{J_m}^{L_m}(m) + \sqrt{h} \boldsymbol{\zeta}_i^k(m) \quad \text{for } i \in [n], k \in [K] \tag{1}$$

where h is the step-size, and $\boldsymbol{\zeta}_i^k(m)$ is the noise term that modelled as a Gaussian noise. The scalar E_{k,L_m} measures the strength of *local elasticity* that $\boldsymbol{z}_{J_m}^{L_m}$ exerts on $\boldsymbol{z}_i^k$ at iteration m. We assume $\boldsymbol{X}_i^k(0)$, $\boldsymbol{\zeta}_i^k(m)$ are jointly independent.

Local elasticity. Clearly, by writing the notation as $E_{k,l}(m)$, we make the assumption that such effect approximately only depends on the class l, k, and time m. We write the effect matrix as $\boldsymbol{E}(m) = (E_{k,l}(m))_{k,l=1}^K$. For the ease of exposition, we assume $\boldsymbol{E}$ only consists of two values $\alpha(m)$ and $\beta(m)$, with $\alpha(m)$ representing the *intra-class* effect and $\beta(m)$ the *inter-class* effect.By writing the equation (1), we also assume the LE effect is "*proportional*" to the feature $\boldsymbol{X}_{J_m}^{L_m}(m)$ itself. We generalize this point in Section 3 by introducing a transformation matrix $\boldsymbol{H}$ on the features.

2.2 Binary LE-SDE

Our construction of (1) emphasizes the effect of intra- and inter-class effects in the *feature dynamics*, and thus distinguish from the usual weight dynamics that common in the literature. Before deriving the most general form of our locally elastic SDE (LE-SDE), we will familiarize the reader with our model by demonstrating it in the binary classification ($K = 2$) with one-dimensional feature ($p = 1$).

Let the intra-class effect be $E_{11} = E_{22} = \alpha$, and inter-class effect be $E_{12} = E_{21} = \beta$, both being time-independent. Expanding (1), for $1 \leq i, j \leq n$ and $m \geq 0$, when we train with the J_m-th training example comes from class L_m, we have

$$\begin{cases} X_i^1(m) &= X_i^1(m-1) + h \cdot \alpha X_{J_m}^{L_m}(m-1) + \sqrt{h} \cdot \zeta_i^{L_m}(m-1), \\ X_j^2(m) &= X_j^2(m-1) + h \cdot \beta X_{J_m}^{L_m}(m-1) + \sqrt{h} \cdot \zeta_j^{L_m}(m-1). \end{cases}$$

In the limit of $h \to 0$, we can show that $X_i^k(m)$ approximates some continuous-time stochastic processes $X_i^k(t)$ (under the identification of $t = mh$), governed by the following set of stochastic

differential equations (SDEs):

$$\mathrm{d}X_i^k(t) = \left(\frac{\alpha}{2}\bar{X}^k(t) + \frac{\beta}{2}\bar{X}^{3-k}(t)\right)\mathrm{d}t + \sigma\,\mathrm{d}W_i^k(t), \quad t \geq 0, \tag{2}$$

where $\bar{X}^k(t) := \left(X_1^k(t) + \cdots + X_n^k(t)\right)/n$, and W_i^k are independent standard Wiener processes. The detailed derivation can be found in the Appendix.

Now averaging over i for each k in the above display, we obtain the following set of two ordinary differential equations (ODEs) governing the per-class means $\bar{X}^k(t)$ for $k = 1, 2$:

$$\mathrm{d}\bar{X}^k(t) = \left(\frac{\alpha}{2}\bar{X}^k(t) + \frac{\beta}{2}\bar{X}^{3-k}(t)\right)\mathrm{d}t + \sigma\,\mathrm{d}\frac{W_i^k(t) + \cdots + W_i^k(t)}{n}.$$

Taking the limit of $n \to \infty$, we observe $\sigma\mathrm{d}\frac{W^1(t)+\cdots+W^n(t)}{n} \Rightarrow 0$ weakly. Thus, the above display converges weakly to the following ODE:

$$\frac{\mathrm{d}\bar{X}^k(t)}{\mathrm{d}t} = \frac{\alpha}{2}\bar{X}^k(t) + \frac{\beta}{2}\bar{X}^{3-k}(t). \tag{3}$$

With the initial conditions $\mathbb{E}X_i^k(0) = c_k$, the solution to the above ODE is

$$\bar{X}^1(t) = \frac{c_1 - c_2}{2}\mathrm{e}^{\frac{\alpha-\beta}{2}t} + \frac{c_1 + c_2}{2}\mathrm{e}^{\frac{\alpha+\beta}{2}t}, \quad \bar{X}^2(t) = -\frac{c_1 - c_2}{2}\mathrm{e}^{\frac{\alpha-\beta}{2}t} + \frac{c_1 + c_2}{2}\mathrm{e}^{\frac{\alpha+\beta}{2}t}.$$

In the finite-sample setting, we may replace $\bar{X}^1, \bar{X}^2$ in the SDE (2) by their deterministic solutions and obtain

$$\begin{cases} X_i^1(t) &= \frac{c_1-c_2}{2}\mathrm{e}^{\frac{\alpha-\beta}{2}t} + \frac{c_1+c_2}{2}\mathrm{e}^{\frac{\alpha+\beta}{2}t} - c_1 + X_i^1(0) + \sigma W_i^1(t), \\ X_j^2(t) &= -\frac{c_x-c_y}{2}\mathrm{e}^{\frac{\alpha-\beta}{2}t} + \frac{c_x+c_y}{2}\mathrm{e}^{\frac{\alpha+\beta}{2}t} - c_2 + X_j^2(0) + \sigma W_j^2(t). \end{cases}$$

We are now ready to derive the condition under which these $2n$ feature vectors become *asymptotically separable*, that is, either $\min_i X_i^1(t) > \max_j X_j^2(t)$ or $\max_i X_i^1(t) < \min_j X_j^2(t)$ as $t \to \infty$.

Theorem 2.1. *Given features vectors $X_i^1(t)$, $X_j^2(t)$ for $i, j \in [n]$, as $\boldsymbol{t} \to \infty$ and large n,*

1. *if $\alpha > \beta$, they are asymptotically separable with probability tending to one,*
2. *if $\alpha \leq \beta$, they are asymptotically separable with probability tending to zero.*

This result indicates a sharp phase transition when α is *just* above β, i.e. in the regime of *local elasticity*. As long as the intra-class effect is slightly greater than the inter-class effect, separation is guaranteed. This simple model already captures local elasticity and reveals the important role it plays in the *perfect separation* of training samples. We can generalize this model to more realistic settings – when there are multiple classes, when features are high dimensional, and when the LE matrix $\boldsymbol{E}$ is time-dependent, which form the core of the next section.

3 General LE-SDE Model

Now we consider the general case where $K \geq 2$ and the feature vectors are p-dimensional with $p \geq K$. Inquisitive readers may have already noticed that Theorem 2.1 only asserts the *emergence* of separation of features, while being inconclusive to their relative orders at separation, i.e., *which class converges to where*? This drawback is intrinsic to the toy model since neither intra-class nor inter-class effect identifies different classes.

In the case of multiple classes, we may want to guarantee a stronger separation – to know which class converges to where. To ensure this property, we need to adjust the raw feature of class $\boldsymbol{X}_{J_m}^{L_m}$ with a proper transformation that incorporates the label information into the dynamic. This motivates the following modification of dynamics (1), by adding an extra transformation $\boldsymbol{H}_{k,L_m} \in \mathbb{R}^{p\times p}$ in front of the feature. For $k \in [K]$, $i \in [n]$, we have the following at iteration m,

$$\boldsymbol{X}_i^k(m) = \boldsymbol{X}_i^k(m-1) + h \cdot E_{k,L_m}(m)\boldsymbol{H}_{k,L_m}(m)\boldsymbol{X}_{J_m}^{L_m}(m-1) + \sqrt{h}\boldsymbol{\zeta}_i^k(m-1). \tag{4}$$

The $\boldsymbol{H}_{k,L_m}(m)$ term models the LE effect to be *proportional* to a linear "transformation" of the features. The dynamics in (1) is a special case when $\boldsymbol{H}_{k,l}(m) \equiv \boldsymbol{I}_p$ for all $k, l \in [K]$. By picking a

proper $\boldsymbol{H}$, we can overcome the limitation in our toy example that not knowing which class converges to where. We specify interesting choices of $\boldsymbol{H}$ in later sections.

Sometimes, we omit the specific index i , and write $\tilde{\boldsymbol{X}}^k(m)$ instead of $\boldsymbol{X}_i^k(m)$, to indicate one sample from the distribution $\mathcal{D}^k(m)$ of all the features of class k at iteration m. As in Section 2.2, we can derive a continuous dynamics of (4) in the limit of $h \to 0$ in the same way for (2). Writing $\tilde{\boldsymbol{X}} = (\tilde{\boldsymbol{X}}^k)_{k=1}^K \in \mathbb{R}^{Kp}$ for the concatenation of per-class features, and similarly $\bar{\boldsymbol{X}} = (\bar{\boldsymbol{X}}^k)_{k=1}^K \in \mathbb{R}^{Kp}$ be the concatenation of per-class mean features. Our model (4) approximates the following SDE with the identification $t = mh$, as $h \to 0$. We term this as the LE-SDE.

$$\mathrm{d}\tilde{\boldsymbol{X}}_t = \boldsymbol{M}_t \bar{\boldsymbol{X}}_t \,\mathrm{d}t + \boldsymbol{\Sigma}_t^{\frac{1}{2}} \,\mathrm{d}\boldsymbol{W}_t, \tag{5}$$

where $\boldsymbol{W}_t$ is the standard Wiener process in $\mathbb{R}^{Kp}$, $\boldsymbol{\Sigma}_t$ the covariance matrix, and $\boldsymbol{M}_t \in \mathbb{R}^{Kp \times Kp}$ a $K \times K$ block matrix, with each block of size $p \times p$. The (k,l)-th block of $\boldsymbol{M}_t$ is $E_{k,l}(t)\boldsymbol{H}_{k,l}(t)/K$ when $l \neq k$, and $E_{l,l}(t)\boldsymbol{H}_{l,l}(t)/K$ when $l = k$. The rationale of dividing K is simply because we assume the data is balanced, and therefore each of the K possible classes has equal chance to appear in the dynamic, as what we proved in (2) where $K = 2$. We generalize this in Section 5 to model imbalance classes.

Taking expectation with respect to the randomness arising from sampling $\tilde{\boldsymbol{X}}_t$ from its distribution, we have the per-class mean $\bar{\boldsymbol{X}}_t$ satisfies the following system, which we term as LE-ODE:

$$\bar{\boldsymbol{X}}_t' = \boldsymbol{M}_t \bar{\boldsymbol{X}}_t. \tag{6}$$

Under the assumptions in Section 2.1, we define $\gamma(t) = \min\left\{\alpha(t) - \beta(t), \alpha(t) + (K-1)\beta(t)\right\}$,

$$A(t) = \int_0^t \alpha(\tau)\,\mathrm{d}\tau, B(t) = \int_0^t \beta(\tau)\,\mathrm{d}\tau, \Gamma(t) = \min\left\{A(t) - B(t), A(t) + (K-1)B(t)\right\}. \tag{7}$$

3.1 The Separation Theorem

As we discussed in Theorem 2.1, the LE-SDE allows us to derive separability result. We say the feature vectors $\left\{(\boldsymbol{X}_i^k)_{i\in[n]}\right\}_{k\in[K]}$ are *separation* if for any two classes $k \neq l$, there exists a hyperplane in $\mathbb{R}^p$ that linearly separates the features of the two classes.

To characterize the separation result as in Theorem 2.1, we need to put conditions on $\alpha(t), \beta(t)$ as therein. In the case when $\gamma(t) \leq 0$, we already know as in Theorem 2.1, the classes cannot separate even pairwisely. Therefore, we will focus on the more interesting case when $\gamma(t) > 0$. We now state the following characterization theorem of separability for the general LE-SDE dynamics.

Theorem 3.1 (Separation of LE-SDE)**.** *Under our working assumptions in Section 2.1, and in the case of local elasticity (i.e. $\gamma(t) > 0$), assume $\boldsymbol{H} = (\boldsymbol{H}_{ij})_{ij}$ is semi-positive definite with positive diagonal entries. As $t \to \infty$, we have*[1]*:*

1. *if $\gamma(t) = \omega(1/t)$, the features are separable with probability tending to 1;*
2. *if $\gamma(t) = o(1/t)$, and the number of per-class-feature n tending to ∞ at an arbitrarily slow rate, the features are asymptotically pairwise inseparable with probability 0.*

This theorem sheds light on the crucial impact of local elasticity effect on separation in the general case. The proof to Theorem 3.1 as well as discussions on the empirically best ways of choosing the universal direction $\boldsymbol{\nu}$ are detailed in the Appendix.

3.2 Two Specific Models

We next discuss two specific choices of the $\boldsymbol{H}$ matrix that allows us to analyze $\tilde{\boldsymbol{X}}$ precisely.

3.2.1 Isotropic Feature Learning Model

As what we did in Section 2.2, we can simply choose $\boldsymbol{H}_{lk} = \boldsymbol{I}_p$ to be identity matrix. This choice of $\boldsymbol{H}$ is SDP and thus we can apply Theorem 3.1 to get conditions on universal separation. And in this case, the solution of $\bar{\boldsymbol{X}}(t)$ can be computed analytically

[1] Here $\gamma(t) = \omega(1/t)$ stands for $\gamma(t) \gg 1/t$.

Proposition 3.2. *Set $\boldsymbol{H}_{k,l} = \boldsymbol{I}_p$, the solution to the LE-ODE (6) is given by*

$$\bar{\boldsymbol{X}}(t) = \boldsymbol{c}\mathrm{e}^{\frac{1}{K}A(t)-\frac{1}{K}B(t)} + (\boldsymbol{1}_K \otimes \boldsymbol{c}_0)\,\mathrm{e}^{\frac{1}{K}A(t)+\frac{K-1}{K}B(t)}, \tag{8}$$

where $\boldsymbol{c} \in \mathbb{R}^{Kp}$ and $\boldsymbol{c}_0 \in \mathbb{R}^p$ are constants with $\sum_{k=1}^K \boldsymbol{c}_k = \boldsymbol{0}$ and $\bar{\boldsymbol{X}}(0) = \boldsymbol{c} + \boldsymbol{c}_0$.

The derivation of Equation (8) is given in the Appendix. From this display, we can easily re-construct Theorem 3.1 in this special case. Since the difference of a feature from class k and a feature from class l at time t is given by $(\boldsymbol{c}_k - c_l)\mathrm{e}^{\frac{1}{K}A(t)-\frac{1}{K}B(t)} + \Sigma^{1/2}(\boldsymbol{W}_t^k - \boldsymbol{W}_t^l)$, as long as the first deterministic term dominates the random second term, we can guarantee separation, which is precisely the conditions specified in theorem 3.1. We term this model as the *isotropic feature model*, or I-model for short, since the $\boldsymbol{H}$ matrix has identity matrices as its blocks and consequently the dynamic does not prescribe any preferred directions for each class.

3.2.2 Logits-as-Features Model

An important type of features in neural nets is the *logits*, the outputs of the neural net before the softmax layer. A logits is K-dimensional, and in a well-trained neural net, the logits of a learnt data instance from the k-th class should have its k-th logit being the largest, and heuristically, the other coordinates should be approximately equal and negative. As we detail in the Appendix, the exact dynamics of neural net training push the logits $\tilde{\boldsymbol{X}}^k$ by it *margin*, namely, $\boldsymbol{e}_k - \mathrm{softmax}(\tilde{\boldsymbol{X}}^k)$, which roughly align with the direction of $\boldsymbol{e}_k - \boldsymbol{1}_p/K$. This provides partial hints as how should we choose the direction of the local elasticity effect: we can choose $\boldsymbol{H}_{l,k}$ such that it always converts $\tilde{X}^k$ to the direction of $\boldsymbol{d}_k$. Hence we may set $\boldsymbol{H}_{ij}$ as follows:

$$\boldsymbol{H}_{ij} = \boldsymbol{H}^j \coloneqq \boldsymbol{d}_j\boldsymbol{d}_j^\top / \|\boldsymbol{d}_j\|_2^2 \in \mathbb{R}^{p\times p}, \quad \boldsymbol{d}_j \coloneqq \boldsymbol{e}_j - \frac{1}{K}\boldsymbol{1}_p \in \mathbb{R}^p, \quad j \in [K]. \tag{9}$$

We give more discussions on the validity of this choice in the Appendix. Roughly speaking, the map $\boldsymbol{x} \mapsto \boldsymbol{H}_{ij}\boldsymbol{x}$ projects $\boldsymbol{x}$ in the direction of $\boldsymbol{d}_j$ and ideally aligns $\boldsymbol{x}$ with $\boldsymbol{d}_j$ after iterative mapping, so $\boldsymbol{H}_{ij}$ can be viewed as an approximation to the nonlinear transformation in the exact dynamics in the sense that the direction of their stationary point coincides. Note $\boldsymbol{H}$ does not satisfy the condition in Theorem 3.1 since it's not symmetric; yet, its structural property gives the following proposition.

Proposition 3.3. *Set $\boldsymbol{H}$ as in Equation (9), the solution to the LE-ODE (6) is given by*

$$\bar{\boldsymbol{X}}(t) = \boldsymbol{c}_0 + C_1\boldsymbol{d}\mathrm{e}^{\frac{1}{K}A(t)-\frac{1}{K}B(t)} + \left(\sum_{l=1}^{K-1} C_{2l}\boldsymbol{f}_l\right)\mathrm{e}^{\frac{1}{K}A(t)+\frac{1}{K(K-1)}B(t)}, \tag{10}$$

where $\boldsymbol{f}_l$'s are fixed vectors in $\mathbb{R}^{K^2}$, $\boldsymbol{c}_0 \in \mathbb{R}^{K^2}$ is a constant vector with $K(K-1)$ degrees of freedom, and $C_1, C_{2l} \in \mathbb{R}$, for $l \in [K-1]$ are constants.

The specific form of $\boldsymbol{f}_l$ is not the focus here, Equation (10) already allows us to prove the statement of Theorem 3.1 also holds for this choice of $\boldsymbol{H}$. The proof of this result is deferred to the Appendix, where we also provides the analytical solution of $\boldsymbol{f}_l$'s when $K \leq 3$.

We term this model as the logits-as-features model, or L-model for short, since it is an elaborated model specifically for logits. In Section 4, we give concrete demonstrations of our abstract feature vector $\tilde{X}$ as logits using the L-model. We numerically simulates our LE-ODE, and compare the predicted dynamic of it to the real deep learning training dynamic. The experimental results therein provide strong empirical support for the validity of our L-model.

3.3 Connection with Neural Collapse

Neural collapse is recent phenomenological finding of the geometry of the features learned by deep learning at convergence [PHD20]. Simply speaking, taking our L-model as an example, with balanced training samples the logit vectors from different classes at convergence can be shown to form an equiangular tight frame (ETF). ETFs are the best way to spread K unit vectors in an ambient space of p dimensions. Formally, we say a set of vectors $\{\boldsymbol{s}_i\}_{i=1}^K$ form an ETF in $\mathbb{R}^p$ if they are the columns from a matrix $\boldsymbol{S} = \sqrt{\frac{K}{K-1}}\boldsymbol{Q}\left(\boldsymbol{I}_K - \frac{1}{K}\boldsymbol{1}_K\boldsymbol{1}_K^\top\right)$, where $\boldsymbol{Q} \in \mathbb{R}^{p\times K}$, and $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}_K$. As a direct corollary of Proposition 3.3 when $\boldsymbol{H}$ is set according to Equation (9), we find that our L-model also predicts the existence of Neural Collapse, from the local elasticity point of view.

Corollary 3.4 (Neural Collapse of LE-SDE)**.** *In the L-model, when $\gamma(t) > 0$ and there exists some $T > 0$ such that when $\boldsymbol{B}(t) < 0$ for $t \geq T$, then $\left\{\bar{\boldsymbol{X}}^k(t)/\|\bar{\boldsymbol{X}}^k(t)\|\right\}_{k=1}^K$ form an ETF as $t \to \infty$.*

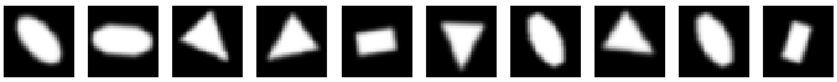


Figure 2: Samples from GEOMNIST dataset.

4 Experiments

We perform various experiments to test out theory, we will choose *logits* as our protagonist.

4.1 Setup

Datasets and Models. We generate a dataset consisting of $K = 3$ simple geometric shapes (RECTANGLE, ELLIPSOID, and TRIANGLE) that are rotated to various angles, which we conveniently name it by Geometric-MNIST or GEOMNIST for short. A few samples from GEOMNIST are shown in Figure 2. We use a varying number of training samples per class $n_{\text{tr}} \in \{40, 240, 800, 2400\}$ with validation sample per-class being $n_{\text{val}} = \{10, 60, 200, 600\}$. We also pollute each label class by randomly choosing $p_{\text{err}} \cdot n_{\text{tr}}$ samples to flip its label to another class (uniform across all other classes). In this setup, we fix $n_{\text{tr}} = n_{\text{val}} = 500$ and set $p_{\text{err}} \in \{0.1, 0.2, \ldots, 0.8\}$. In addition to GEOMNIST, we used CIFAR-10 ([Kri09], wich we denote by CIFAR) for a more realistic scenario with 5000 training samples 1000 validation samples per class. We vary the total number of classes $K \in [10]$. We use variants of the ALEXNET model ([KSH12]), which consists of two convolutional layers and three fully-connected layers activated by the `ReLU` function.

Training Configurations. All models are trained for $T = 10^5$ iterations (for GEOMNIST) or $T = 3 \times 10^5$ iterations (for CIFAR) with a learning rate of 0.005 and a batch size of 1 under the softmax cross-entropy loss. We performed all simulations on a `GeForce RTX 2080 Ti` GPU with a total training time spanning multiple weeks.

Estimation Procedures. Each experiment is repeated for $n_{\text{trial}} = 100$ independent runs to estimate $\bar{\boldsymbol{X}}(t)$. We will use both isotropic feature learning model (Section 3.2.1) and logits-as-features model (Section 3.2.2), which recall we denote by L-model and I-model respectively. We will use L-model only when $K = 3$. To estimate $\alpha(t)$ and $\beta(t)$, we first estimate $A(t)$ and $B(t)$ by

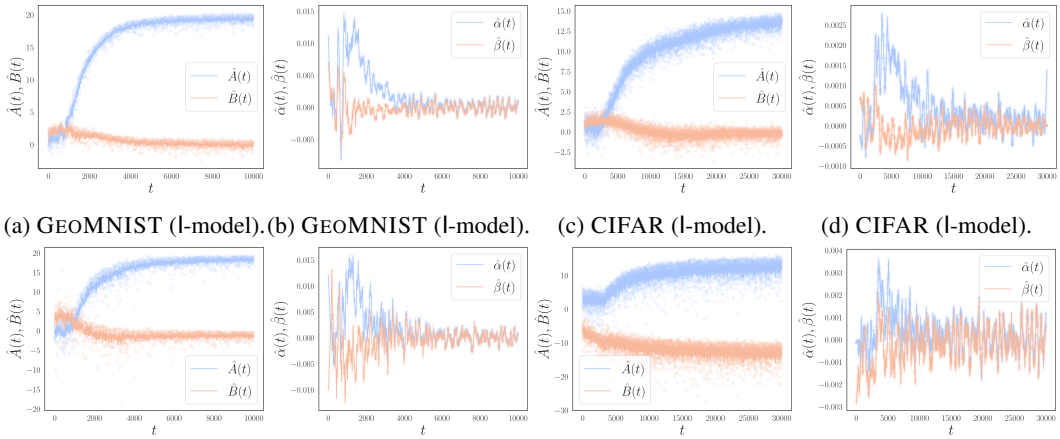
$$
\begin{aligned}
&\text{(I-model)} \quad \begin{cases} \hat{A}(t) &= \operatorname{avg} \operatorname{avg}_k \log \left| \frac{\check{\boldsymbol{X}}(\bar{\boldsymbol{X}}^k - \check{\boldsymbol{X}})^{K-1}}{\boldsymbol{c}_0 \boldsymbol{c}_k^{K-1}} \right|, \\ \hat{B}(t) &= -\operatorname{avg} \operatorname{avg}_k \log \left| \frac{\boldsymbol{c}_0}{\boldsymbol{c}_k} \frac{\bar{\boldsymbol{X}}^k - \check{\boldsymbol{X}}}{\check{\boldsymbol{X}}} \right|, \end{cases} \qquad \check{\boldsymbol{X}}_t \coloneqq \operatorname{avg}_l \bar{\boldsymbol{X}}_t^l, \\
&\text{(L-model)} \quad \begin{cases} \hat{A}(t) &= A'(t) + 2B'(t), \\ \hat{B}(t) &= 2(B'(t) - A'(t)), \end{cases} \quad \begin{cases} A'(t) &\coloneqq \log \left| \left\langle \bar{\boldsymbol{X}}^\top \boldsymbol{v}_1 \right\rangle \right|, \\ B'(t) &\coloneqq \log \left| \left\langle \bar{\boldsymbol{X}}^\top \left(\boldsymbol{I}_9 - \boldsymbol{v}_1 \boldsymbol{d}^\top \right) \boldsymbol{v}_2 \right\rangle \right|, \end{cases}
\end{aligned}
\tag{11}
$$

where vector division is interpreted entry-wise. We write avg_l for averaging over the class index, avg for averaging over the coordinates, and define $\langle \boldsymbol{X} \rangle(t) \coloneqq \boldsymbol{X}(t)/\boldsymbol{X}(0)$. We choose the vectors $\boldsymbol{v}_1 = \frac{1}{4}[1, 1, 1, 1, 1, 0, 0, 0, 0]$ and $\boldsymbol{v}_2 = \frac{1}{3}[2, -1, -1, -1, 2, -1, 0, 0, 0]$, and explain how and why in the Appendix. With $A(t)$ and $B(t)$ estimated, we use Savitzky–Golay filter to obtain $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ through numerical differentiation. For GEOMNIST and CIFAR datasets, we choose window sizes 191 and 551 respectively in Figure 3, and 21 and 21 respectively in Figure 5.

We assess the tail of $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ by a *tail index* defined as $r_\alpha \coloneqq \sup_s \{s : \lim_{t\to\infty} \alpha(t) \cdot t^s < \infty\}$ and r_β is defined similarly. We estimate $\hat{r}_\alpha$ by fix an interval $[T_1, T_2]$ with $T_1 =, T_2$ sufficiently large we may ignore terms with smaller order and have $\hat{r}_\alpha = 1 - \operatorname{avg}_{T_1 \le t \le T_2} \frac{\log \alpha(t)}{\log(1+t)}$, and similarly for $\hat{r}_\beta$. We used the estimates from the last $T_2 - T_1 = 1000$ iterations for averaging in our experiments.

4.2 Results

Local Elasticity in Neural Net Training Local elasticity manifests from our model as the heaviness of the tail of $\gamma(t) = \alpha(t) - \beta(t)$. In Figure 3, we plot the estimations $\hat{A}(t)$, $\hat{B}(t)$, $\hat{\alpha}(t)$, and $\hat{\beta}(t)$ using both I-model and L-model. We note that (i) the estimations from two models are visually similar, especially in the late stage of training when t is large; (ii) the major difference lies in the initial stage, where the estimates from the L-model behaves slightly more wild. This is not very surprising due to the effect of the unknown constant offset $\boldsymbol{c}_0$ in the L-model; (iii) estimations under I-model are more stable, shown by smaller variance compared with that of L-model. There are several



(a) GEOMNIST (l-model). (b) GEOMNIST (l-model). (c) CIFAR (l-model). (d) CIFAR (l-model).

(e) GEOMNIST (L-model). (f) GEOMNIST (L-model). (g) CIFAR (L-model). (h) CIFAR (L-model).

Figure 3: **Estimated $\hat{A}(t)$, $\hat{B}(t)$, $\alpha(t)$, and $\beta(t)$.** The first row was estimated using l-model and the second L-model; the first two columns are on GEOMNIST and the last two on CIFAR. The first and the third rows show $\hat{A}(t)$ and $\hat{B}(t)$ and the other two rows $\hat{\alpha}(t)$ and $\hat{\beta}(t)$.

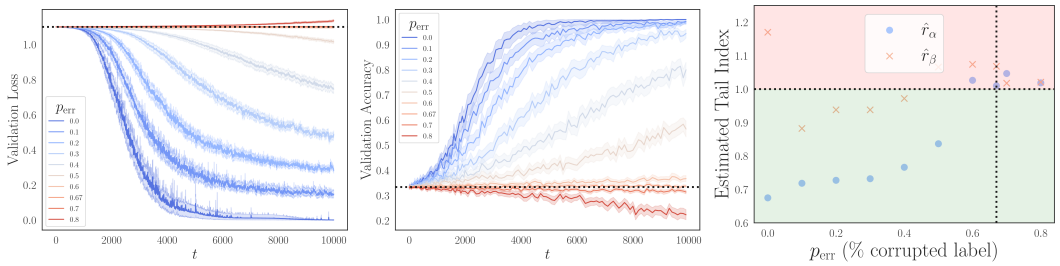
common aspects in which $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ behave similarly on both datasets: (i) $\alpha(t)$ rises as training progresses and peaks at the time when the training loss drops the fastest; (ii) both $\alpha(t)$ and $\beta(t)$ exhibited convergent behaviour when t is large, and $\alpha(t)$ dominates $\beta(t)$ eventually.

Phase Transition of Separability Theorem 3.1 states that separation of features under the LE-SDE takes place when $\gamma(t) = \alpha(t) - \beta(t) = \omega(1/t)$, or roughly speaking, when $r_\gamma = \min\{r_\alpha, r_\beta\} < 1$. Although we cannot directly control $\alpha(t)$ and $\beta(t)$, we can bias them by tuning the label corruption percentage p_{err}. When $p_{\text{err}} \approx p^*_{\text{err}} := 2/3$, we are in effect assigning labels completely at random and thus we expect a phase transition of separability should happen around p^*_{err}. This is indeed the story depicted in Figure 4: Figures 4a and 4b shows that the validation loss and accuracy for $p \geq p_{\text{err}}$ is not increasing along time and in Figure 4c we observe the the minimum tail index of $\alpha(t)$ and $\beta(t)$ crosses 1 from below around $p_{\text{err}} = 0.6$, entering the non-separable regime (shaded in red) from the separable regime (shaded in green) given in Theorem 3.1.

Simulating DNN Dynamics via LE-ODE Having estimated $\alpha(t)$ and $\beta(t)$, it is natural to ask, to what capacity can our LE-ODE models recover the real dynamics of deep neural nets? We use forward Euler method to simulate the L-model using $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ estimated from either l-model or L-model. The initial value for the k-th class is set to be $\boldsymbol{\zeta}^K \sim \mathcal{N}_K(\mathbf{0}, \sigma_k \boldsymbol{I}_3)$ for all $k \in [K]$ with $\sigma_k = \left\| \bar{\boldsymbol{X}}^k(0) \right\|_2 / \sqrt{K}$ with $\bar{\boldsymbol{X}}^k(0)$ sampled from simulations in a DNN. We show in Figure 5 the simulated solution (solid line) with errorbars depicting one standard deviation over 500 independent runs, overlaying on the real dynamics from DNNs in the background (shaded transparent markers). Since the moving average may reduce the magnitudes of $\alpha(t)$ and $\beta(t)$, we rescale the simulated paths such that its first coordinate is approximately equal to the ground truth at convergence. Note that estimations from L-model can faithfully recover the genuine dynamics from neural nets whereas those from l-model fail, notably in Figure 5a, where the simulated paths preserve relative magnitude but fail to identify the correct order of three logits. This is due to the nature of l-model that does not distinguish logits, and thus l-model-based estimations would arguably perform poorly when the supervision is strong.

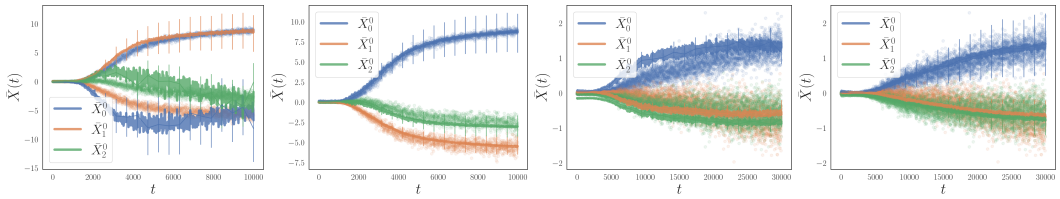
5 Discussion and Future Works

In this paper, we introduced the LE-SDE/ODE model that drew inspiration from the local elasticity phenomenon. Conditions for sharp phase transition of separability of features was derived. We also showed that once the elasticity strengths $\alpha(t)$ and $\beta(t)$ are well estimated, our model can faithfully simulate the dynamics of neural nets. This link between a tractable mathematical model (LE-SDE/ODE) and an expressive yet complicated black box would potentially yield greater benefits towards demystifying the magic of deep models. We outline a few interesting problems for future research.



(a) Validation loss versus p_{err}. (b) Validation accuracy versus p_{err}. (c) Tail index versus p_{err}.

Figure 4: **Phase transition of separability. (a)—(b)** Validation loss and accuracy suggest separation fails for $p_{\text{err}} \geq p^*_{\text{err}} = 2/3$. The dashed line in (a) carries the value at initialization and overlaps with the case where $p_{\text{err}} = 0.6$; the dashed line in (b) is $p^*_{\text{err}} = 2/3$, when labels are assigned completely at random. **(c)** Tail indices of $\alpha(t)$ and $\beta(t)$. Note that $\gamma(t) = \alpha(t) - \beta(t)$ crosses, 1 entering the non-separable regime (shaded in red) from the separable regime (shaded in green), around the same p_{err} that cross the dashed lines in (a) and (b), as predicted by Theorem 3.1.



(a) GEOMNIST (l-model).(b) GEOMNIST (L-model). (c) CIFAR (l-model). (d) CIFAR (L-model).

Figure 5: **Simulated LE-ODE solutions versus genuine dynamics.** We use $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ estimated from l-model ((a) and (c)) or L-model ((b) and (d)) and numerically simulate the solution under the L-model. The results were overlaid with true dynamics from neural nets. We note both recovers true dynamics reasonably well on CIFAR while l-model-based estimate fails to recover the dynamics when trained on GEOMNIST.

Imbalanced Datasets. As discussed in Section 3, we can incorporate imbalanced datasets in training easily. Recall each block of $\boldsymbol{M}_t$ in Equation (5) takes the form of $E_{k,l}\boldsymbol{H}_{k,l}/K$, where $1/K$ signifies that each class has the same $1/K$ probability of being sampled during any iteration in training. This can be generalized by changing the (k,l)-th block to $E_{k,l}\boldsymbol{H}_{k,l} \cdot p_l$ for $\sum_l p_l = 1$, where p_l is the probability of class l being sampled.

Corrupted Data with Noisy Labels. We can also model the case when there data are polluted by corrupted labels. Let $\boldsymbol{P} = (p_{k,l})_{k,l} \in \mathbb{R}^{K \times K}$ be a column-stochastic matrix where $p_{k,l}$ represents the probability of a sample from class k mis-labelled as class l, we can replace the matrix $\boldsymbol{E}$ by $\boldsymbol{EP}$. Aa well-defined model needs further assumption on the structure of $\boldsymbol{P}$ and we leave this theoretical modelling to future work.

Two-Stage Behavior and Fine-Grained Analyses. In Section 4, we observed a clear two-stage behaviour of our numerical simulation of our LE-SDE as well as in the real deep learning dynamics. The first stage is a *de-randomization* stage that gradually eliminates the effect of random initialization and searches for the correct directions to be separated. The second stage is an *amplification* stage, where the model amplifies the magnitudes of features in those directions. This empirical observation naturally leads to many interesting questions: Is this a universal phenomenon in deep learning, and what are the conditions to guarantee to enter the second stage? Can our LE-SDE predicts such two-stage phenomenon theoretically? What is the role of local elasticity played in this transition? We leave the investigation of those to future works.

References

[ACGH18] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[ADH+19] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[CCGZ20] Zixiang Chen, Yuan Cao, Quanquan Gu, and Tong Zhang. A generalized neural tangent kernel analysis for two-layer neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[CHS20] Shuxiao Chen, Hangfeng He, and Weijie J. Su. Label-aware neural tangent kernel: Toward better generalization and local elasticity, 2020.

[CKL+21] Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability, 2021.

[CS18] Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DH19] Simon Du and Wei Hu. Width provably matters in optimization for deep linear neural networks. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2019.

[DHP+19] Simon S Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *arXiv preprint arXiv:1905.13192*, 2019.

[DHS20] Zhun Deng, Hangfeng He, and Weijie J Su. Toward better generalization bounds with locally elastic stability. *arXiv preprint arXiv:2010.13988*, 2020.

[DL13] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[DLL+19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[GAS+19] Sebastian Goldt, Madhu S Advani, Andrew M Saxe, Florent Krzakala, and Lenka Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *arXiv preprint arXiv:1906.08632*, 2019.

[HJ15] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

[HS19] Hangfeng He and Weijie J. Su. The local elasticity of neural networks. *CoRR*, abs/1910.06943, 2019.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[JKA+17] Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[JT19] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*, 2019.

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima. *arXiv preprint arXiv:1605.07110*, 2016.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arxiv. org, 2014.

[KMN+16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[Kri09] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[LBN+17] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[LG18] Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv preprint arXiv:1809.10374*, 2018.

[LL18] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *arXiv preprint arXiv:1808.01204*, 2018.

[LMA21] Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling sgd with stochastic differential equations (sdes). *arXiv preprint arXiv:2102.12470*, 2021.

[LTW17] Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 2101–2110. PMLR, 2017.

[MHB17] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

[PHD20] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, Sep 2020.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[SHN+18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[SKYL17] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[SMG13] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[SMG19] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[SS96] David Saad and Sara A Solla. Dynamics of on-line gradient descent learning for multilayer neural networks. *Advances in neural information processing systems*, pages 302–308, 1996.

[SSJ20] Bin Shi, Weijie J Su, and Michael I Jordan. On learning rates and schrodinger operators. *arXiv preprint arXiv:2004.06977*, 2020.

[Yan19] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[ZBH+16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5 for the discussions.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.1, also see the beginning of Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 2.2, and the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the code, data, and instructions needed to reproduce the main experimental results in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See figures in Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We used CIFAR-10 and the AlexNet architecture. We cited them both in Section 4.
   (b) Did you mention the license of the assets? [N/A] ?
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We constructed a dataset of geometric shapes which we call GeoMNIST. We include code generating this dataset in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]