Towards Robust ResNet: A Small Step but A Giant Leap

Jingfeng Zhang[1] Bo Han[2] Laura Wynter[3] Kian Hsiang Low[1] Mohan Kankanhalli[1]

[1]National University of Singapore, [2]RIKEN-AIP, [3]IBM Research

Abstract

This paper presents a simple yet principled approach to boosting the robustness of the *residual network* (ResNet) that is motivated by the dynamical system perspective. Namely, a deep neural network can be interpreted using a partial differential equation, which naturally inspires us to characterize ResNet by an explicit Euler method. Our analytical studies reveal that the step factor h in the Euler method is able to control the robustness of ResNet in both its training and generalization. Specifically, we prove that a small step factor h can benefit the training robustness for back-propagation; from the view of forward-propagation, a small h can aid in the robustness of the model generalization. A comprehensive empirical evaluation on both vision *CIFAR-10* and text *AG-NEWS* datasets confirms that a small h aids both the training and generalization robustness.

1 Introduction

Deep neural networks (DNNs) have reached an unprecedented level of predictive accuracy in several real-world application domains (e.g., text processing [1, 2], image recognition [3, 4], and video analysis [5, 6]) due to their capability of approximating any universal function [7]. However, DNNs are often difficult to train due in a large part to the vanishing gradient phenomenon [8, 9]. The *residual network* (ResNet) [3] is proposed to alleviate this issue using its key component known as the skip connection, which creates "bypass" path for information propagation [10].

Nevertheless, it remains challenging to achieve robustness in training a very deep DNN. As the DNN becomes deeper, it requires more careful tuning of the model hyperparameters (e.g., learning rate, number of layers, choice of optimizer and so on) to perform well. This issue can be mitigated using normalization techniques such as *batch normalization* (BN) [9], layer normalization [11], and group normalization [12], among which BN is most widely used. BN normalizes the inputs of each layer to enable a robust training of DNN. It has been shown that BN provides a smoothing effect on the optimization landscape [13], thus ameliorating the issue of a vanishing gradient [9].

Unfortunately, even coupled with identity mapping and BN, we have empirically observed that a very deep DNN can potentially (and surprisingly) experience an exploding gradient, thus impairing its robustness in training. As a result, (a) our experiments in Section 4.1 reveal that at early training iterations, gradients become very large at shallow layers, which makes weights changing abruptly, i.e., the magnitude of weights suddenly becomes very large. These large weights cause violent feature transformations. (b) Our experiments in Appendix 6.1, show that gradients of deep layers are large and bumpy at later-on training iterations, which destabilize training procedures. Those phenomena are where training procedure suffers. Furthermore, noisy data (e.g., images with mosaics and texts with spelling errors) can also adversely impact the training procedure, in which the noise may be amplified through forward propagation of features, thus degrading its robustness in generalization.

Paper accepted at IJCAI-2019.

This paper presents a simple yet principled approach to boosting the robustness of ResNet in both training and generalization, which is motivated by a dynamical systems perspective [14, 15, 16, 17]. Namely, we can view a DNN from the perspective of a partial differential equation (PDE). This naturally inspires us to characterize ResNet by an explicit Euler method. We show that the step factor h in the Euler method can serve to control the robustness of ResNet in both training and generalization. Specifically, in our paper training robustness refers to the stability of training with increasing depth, larger learning rate, different types of chosen optimizer. Generalization robustness refers to how well the trained model can handle the testing dataset, where its distribution mismatches with that of the training dataset. To analyze effects of step factor h, we theoretically prove that a small h can benefit the training robustness from the view of back-propagation; from the view of forward-propagation, the small h can help the generalization robustness. We empirically conduct experiments on both vision and text datasets, i.e., *CIFAR-10* [18] and *AG-NEWS* [2]. The comprehensive experiments demonstrate that small h can benefit the training and generalization robustness. To sum up, by introducing this new hyper-parameter the step factor h, we do not increase the trainable parameters, but significantly boost the robustness of training and generalization of deep ResNet.

2 Related literature

Before delving into robust ResNet (Section 3), we provide an overview of several prerequisites, including ResNet, batch normalization and partial differential equations.

2.1 Brief summary of ResNet

Residual network (ResNet) is a variant of the deep neural network, which exhibits compelling accuracy and convergence properties [3]. Its key component is the skip connection. Thus, ResNet consists of many stacked residual blocks, and its general form is shown below:

$$\begin{aligned}\mathbf{y}_n &= \mathbf{x}_n + \mathcal{F}(\mathbf{x}_n);\\ \mathbf{x}_{n+1} &= I(\mathbf{y}_n),\end{aligned}$$

where $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ are the input and output of the n-th block, $\mathcal{F}$ is a residual block containing feature transformation, e.g., convolutional operation or affine transformation, and I is an element-wise operation, e.g., ReLU function [19] or identity mapping [10].

Based on the core idea of the skip connection, variants of ResNet have been proposed for specific tasks, e.g., DenseNet for image recognition [4], and VDCNN with shortcut connections for text classification [1].

2.2 Batch normalization

Batch normalization (BN) has been widely adopted for training deep neural networks [9], which normalizes layer input over each training mini-batch. Specifically, for the input of a layer with dimension d, $\mathbf{x} = (x^{(1)}, \ldots, x^{(k)}, \ldots, x^{(d)})$, BN first normalizes each scalar feature $\hat{x}^{(k)} = \frac{x^{(k)} - \mu^{(k)}}{\sigma^{(k)}}$ independently, where $\mu^{(k)} = \mathrm{E}[x^{(k)}]$ and $\sigma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$ are computed over a mini-batch of size m. Then, it performs an affine transformation of each normalized value by $\mathrm{BN}(x^{(k)}) = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$, where $\gamma = (\gamma^{(1)}, \ldots, \gamma^{(k)}, \ldots, \gamma^{(d)})$ and $\beta = (\beta^{(1)}, \ldots, \beta^{(k)}, \ldots, \beta^{(d)})$ are learned during training.

BN has been shown to smooth the optimization landscape of deep neural network [13]. This offers more stable gradients for robust training of deep neural network, thus effectively ameliorating the vanishing gradient problem [9].

2.3 Characterizing deep neural network by a partial differential equation

Deep neural networks (DNN) represent the underlying nonlinear feature transformations, so that the transformed features can be matched with their target values, such as categorical labels for classification and continuous quantities for regression. Namely, DNN transforms the input data through multiple layers, and the feature in the last layer should be linearly separable [20].

Turning to the dynamical systems view, a partial differential equation (PDE) can characterize the motion of particles [21, 22]. Motivated by this viewpoint, we can characterize the feature transformations in DNN as a system of the first-order PDE, which can be formulated as:

Definition 1. *Feature transformations in DNN can be characterized as the first-order partial differential equation (PDE):*

$$\dot{\mathbf{x}} = f(t, \mathbf{x}), \tag{1}$$

where $\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t}$, $f : T \times X \to R^d$, $T \subseteq R^+$, *and* $X \subseteq R^d$. $t \in T$ *is the time along the feature transformations, in which* $t \geq 0$. $\mathbf{x} \in X$ *is the feature vector of dimension* d.

Given an initial data $\mathbf{x}_0$ as the input, a PDE gives rise to the initial value problem (**IVP**):

$$\dot{\mathbf{x}} = f(t, \mathbf{x});\ \mathbf{x}(0) = \mathbf{x}_0, \tag{2}$$

where $\mathbf{x}_0$ is a specified initial input of feature vector. $\mathbf{x}(t)$ is the transformed feature at the time t.

A solution to an IVP is a function of time t. For example, we have IVP $\dot{x} = -2.3x, x(0) = 1.0$. Thus, we can easily derive its analytic solution $x(t) = e^{-2.3t}$ as shown in Figure 1. However, for more complicated PDEs, it is not always feasible to obtain an analytic solution. Rather, their solutions are approximated by numerical methods, of which the explicit Euler method (shown in Definition 2 in next section) is the most classic example.

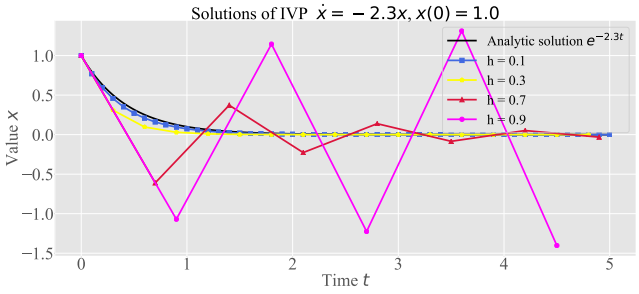


Figure 1: The analytic solution of an IVP and its approximations by an explicit Euler method with different step factors h.

3 Towards Robust ResNet

Residual network (ResNet) is one exemplar of DNN. ResNet can be described by the explicit Euler method (Definition 2), which naturally brings us a Euler-viewed ResNet (Eq. (4)). Inspired by the stability of the Euler method, we emphasize the efficacy of the step factor h for ResNet, which enables ResNet to have smoother feature transformations. Such smoothness means that feature transitions take a small change at every adjacent block during the forward propagation. This smoothness has two-fold benefits: preventing information explosion over the depth (Section 3.2), and filtering noise in the input features (Section 3.3). All these together drive the significant robustness gains in the ResNet.

3.1 Connections of ResNet with the Euler method

Consider the following definition of the explicit Euler method.

Definition 2. *The explicit Euler method [21, 22] can be represented as follows.*

$$\mathbf{x}_{n+1} = \mathbf{x}_n + hf(t_n, \mathbf{x}_n), \tag{3}$$

where $\mathbf{x}_n$ *is the approximation of* $\mathbf{x}(t_n)$.

Given the solution $\mathbf{x}(t_n)$ at some time t_n, the PDE $\dot{\mathbf{x}} = f(t, \mathbf{x})$ tells "in which direction to continue". At time t_n, the explicit Euler method computes this direction $f(t_n, \mathbf{x}_n)$, and follows it when a small time step changes ($t_n \to t_n + h$).

In order to obtain the reasonable approximation, the step factor h in Euler's method has to be chosen to be "small enough". Its magnitude depends on the PDEs and its specified initial inputs. For instance, in Figure 1, we leverage Euler method with various step factors h to approximate the IVP $\dot{x} = -2.3x$, $x(0) = 1.0$. It is clearly shown that the approximation of x fluctuates more with large h, hence the smaller h approximates the true function better.

Euler-viewed ResNet: Taking $h = 1.0$ and $f(t_n, \mathbf{x}_n) = \mathcal{F}(\mathbf{x}_n)$, we can use the Euler method to characterize the forward propagation of the original ResNet structure [20]. Specifically, to generalize ResNet by the Euler method (Eq. (3)), we characterize the forward propagation of an Euler-viewed ResNet as follows.

Following the conventional analysis [10], ResNet is a stacked structure [3], where its residual block with step factor $h \in R^+$ performs the following computation:

$$\begin{aligned} \mathbf{y}_n &= \mathbf{x}_n + h\mathcal{F}(\mathbf{x}_n, W_n, \mathrm{BN}_n); \\ \mathbf{x}_{n+1} &= I(\mathbf{y}_n). \end{aligned} \tag{4}$$

At layer n, $\mathbf{x}_n$ is the input features to the n-th Residual block $\mathcal{F}$. BN_n are the parameters of batch normalization, which applies directly after (or before) the transformation function W_n, i.e., convolutional operations or affine transformation matrix. The function I is an element-wise operation, e.g., identity mapping or ReLu.

As analysed in the chapter "analysis of the Euler method" [23], for any fixed time interval t, the Euler method approximation is better refined with smaller h. However, it is not obvious whether small h can helps on robustness of deep ResNet in terms of training and generalization.

To answer this question, we prove that the reduced step factor h in Eq. (4) can boost its robustness. Specifically, the benefits of this new parameter h are as follows. (a) Training robustness (Section 3.2): Small h helps the information back-propagation during the training, and thus prevents its explosion over depth. (b) Generalization robustness (Section 3.3): With greater smoothness in feature forward-propagation, the noise in the features is reduced rather than amplified, thus mitigating the negative effects of noise and enhancing generalization. We will now theoretically explore the efficacy of small h for training and generalization robustness.

3.2 Training robustness: Importance of small h on information back-propagation

To simplify the analysis, let I be the identity mapping operation from Eq. (4), i.e., $\mathbf{x}_{n+1} = \mathbf{y}_n$. Recursively, we have

$$\mathbf{x}_N = \mathbf{x}_n + h \sum_{i=n}^{N-1} \mathcal{F}(\mathbf{x}_i, W_i, BN_i), \tag{5}$$

for any deeper block N and any shallower block n. Eq. (5) is used to analyse the information back-propagation. Denoting the loss function as L, by the chain rule, we have:

$$\frac{\partial L}{\partial \mathbf{x}_n} = \frac{\partial L}{\partial \mathbf{x}_N}\frac{\partial \mathbf{x}_N}{\partial \mathbf{x}_n} = \frac{\partial L}{\partial \mathbf{x}_N}\big(1 + h\frac{\partial}{\partial \mathbf{x}_n}\sum_{i=n}^{i=N-1}\mathcal{F}(\mathbf{x}_i, W_i, BN_i)\big). \tag{6}$$

Eq. (6) shows that the back-propagated information $\frac{\partial L}{\partial \mathbf{x}_n}$ can be decomposed into two additive terms: a term $\frac{\partial L}{\partial \mathbf{x}_N}$ (the first term) that propagates information directly without going through any weight layers, and a term $h\frac{\partial L}{\partial \mathbf{x}_N}\big(\frac{\partial}{\partial \mathbf{x}_n}\sum_{i=n}^{i=N-1}\mathcal{F}(\mathbf{x}_i, W_i, BN_i)\big)$ (the second term), which propagates through weight layers between n and N.

The first term as stated by [10] ensures that information $\frac{\partial L}{\partial \mathbf{x}_n}$ does not vanish. However, the second term can blow-up the information $\frac{\partial L}{\partial \mathbf{x}_n}$, especially when the weights of layers are large. The standard approach for tackling this is to apply batch normalization (BN) after transforming functions [9].

let firstly us examine the effects of BN. For analysis purposes, we examine one residual block, taking $N = n + 1$, $\mathcal{F}(\mathbf{x}_n) = \mathrm{BN}(W_n\mathbf{x}_n)$, $\mathbf{x}'_n = W_n\mathbf{x}_n$, and $\mathrm{BN}(\mathbf{x}'_n) = \gamma\frac{\mathbf{x}'_n - \bar{\mu}}{\bar{\sigma}} + \beta$, where BN is an element-wise operation. For a batch size of m, there are m transformed $\mathbf{x}'_n$ with mean vector $\bar{\mu} = (\mu_1, \mu_2, \ldots, \mu_m)^T$ and standard deviation vector $\bar{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)^T$. Thus, we get

$$\frac{\partial \mathcal{F}}{\partial \mathbf{x}_n} = W_n\frac{\gamma}{\bar{\sigma}}. \tag{7}$$

Taking σ_n as the smallest one among elements in $\bar{\sigma}$, we get

$$||\frac{\partial \mathcal{F}}{\partial \mathbf{x}_n}|| \leq \frac{\gamma}{\sigma_n}||W_n||. \tag{8}$$

To sum up, we have

$$||\frac{\partial L}{\partial \mathbf{x}_n}|| \leq ||\frac{\partial L}{\partial \mathbf{x}_{n+1}}||\big(1 + h(\frac{\gamma}{\sigma_n}||W_n||)\big). \tag{9}$$

As observed by [13], $\bar{\sigma}$ tends to be large, thus BN gives the positive effect of constraining the explosive back-propagated information (Eq. (9)). However, when networks go deeper, $\bar{\sigma}$ tends to be highly uncontrolled in practice, and the back-propagated information is still accumulated over the depth and can again blow up. For example, Figure 2 shows that gradients are very large and unstable at early training iterations (red line), such that when we train deep ResNet on *CIFAR-10* and deep ResNet on *AG-NEWS*, the performance is significantly worse compared to their shallower counterparts, as shown in Figure 3 (red line). By contrast, Figure 2 shows that, a reduced h can serve to re-constrain the explosive back-propagated information (blue line).

In addition, from Eq. (9), the effect of h is on the top BN, which can give extra constraints on explosive back-propagated information. It tells us that even without BN, reducing h can also serve to stabilize the training procedure. In other words, our reduced h offers enhanced robustness even without BN. We carry out experiments supporting this observation in Figure 5 in Section 4.1.

Now, let us take a special case to examine the importance of small h on information back-propagation.

Proposition 1. *let $n = 0$ and $N = D$, where D is the index number of last residual block. Supposing at any transformation block i, we have $||\frac{\partial \mathcal{F}}{\partial \mathbf{x}_i}|| \leq \mathcal{W}$, where $\mathcal{W}$ upper bounds the effects of BN, i.e., $\frac{\gamma}{\sigma}||W|| \leq \mathcal{W}$. Then, we have $||\frac{\partial L}{\partial \mathbf{x}_0}|| \leq ||\frac{\partial L}{\partial \mathbf{x}_D}||(1 - h + h(1 + \mathcal{W})^D)$.*

Proof.

$$\frac{\partial L}{\partial \mathbf{x}_0} = \frac{\partial L}{\partial \mathbf{x}_D}\big(1 + h\frac{\partial}{\partial \mathbf{x}_n}\sum_{i=0}^{i=D-1}\mathcal{F}(\mathbf{x}_i)\big); \tag{10}$$

$$||\frac{\partial L}{\partial \mathbf{x}_0}|| = ||\frac{\partial L}{\partial \mathbf{x}_D}||\big(1 + h||\frac{\partial}{\partial \mathbf{x}_n}\sum_{i=0}^{i=D-1}\mathcal{F}(\mathbf{x}_i)||\big);$$

$$||\frac{\partial L}{\partial \mathbf{x}_0}|| \leq ||\frac{\partial L}{\partial \mathbf{x}_D}||(1 + h\sum_{i=0}^{D-1}(\mathcal{W}(1 + \mathcal{W})^i))$$

$$= ||\frac{\partial L}{\partial \mathbf{x}_D}||(1 - h + h(1 + \mathcal{W})^D).$$

□

Note that $1 + \mathcal{W}$ is bigger than 1, and the back-propagated information explodes exponentially w.r.t the depth D, which directly affects the gradients. This hurts the training robustness of ResNet. However, reducing h can give extra control of the back-propagated information, since the increasing term $(1 + \mathcal{W})^D$ will be constrained by h and the back-propagated information $\frac{\partial L}{\partial \mathbf{x}_0}$ will less likely explode. This guides us that, when ResNet becomes deeper, we should reduce h to a smaller value.

3.3 Generalization robustness: Importance of small h on information forward-propagation

It is obvious that the reduced h gives extra control of the feature transformations in ResNet. Namely, it makes feature transformations smoother, i.e., $||\mathbf{x}_{n+1} - \mathbf{x}_n||$ is smaller, as features go through ResNet block by block. In other words, it forces $\mathbf{x}_{n+1}$ to take a reduced change compared with $\mathbf{x}_n$.

More importantly, as features forward propagate through the deep ResNet, negative effects of the input noise are reduced and constrained over the depth. We theoretically show that a reduced h can help stabilize the output $f(\mathbf{x}) \in R^m$ of ResNet against the input noise, where $\mathbf{x}$ can be a vector of

pixels or vectorization of texts. Suppose we have a perturbed copy $\mathbf{x}_\epsilon$ of $\mathbf{x}$, we expect $f(\mathbf{x}_\epsilon)$ is close to $f(\mathbf{x})$, that is

$$\forall \mathbf{x} : \mathbf{x} \approx \mathbf{x}_\epsilon \implies f(\mathbf{x}) \approx f(\mathbf{x}_\epsilon).$$

Let $\mathbf{x}_N$ be the transformed feature of ResNet starting at input feature $\mathbf{x}_0$, and $\mathbf{x}_N^\epsilon$ starting at $\mathbf{x}_0^\epsilon$, where $\mathbf{x}_0^\epsilon$ is a perturbed copy of $\mathbf{x}_0$, i.e., $||\mathbf{x}_0^\epsilon - \mathbf{x}_0|| \leq \epsilon$.

$$\begin{aligned} ||\mathbf{x}_N^\epsilon - \mathbf{x}_N|| &= ||\mathbf{x}_0^\epsilon + h\sum_{i=0}^{N-1} \mathcal{F}(\mathbf{x}_i^\epsilon) - \mathbf{x}_0 - h\sum_{i=0}^{N-1} \mathcal{F}(\mathbf{x}_i)|| \\ &= ||\epsilon + h(\sum_{i=0}^{N-1} \mathcal{F}(\mathbf{x}_i^\epsilon) - \sum_{i=0}^{N-1} \mathcal{F}(\mathbf{x}_i))|| \\ &\leq \epsilon + h\sum_{i=0}^{N-1} ||\mathcal{F}(\mathbf{x}_i^\epsilon) - \mathcal{F}(\mathbf{x}_i)||. \end{aligned} \tag{11}$$

Eq. (11) shows that the noise with input features is amplified along the depth of ResNet. However, with the introduction of a reduced h (e.g., from 1 to 0.1), we limit the noise amplification, and provide the extra capacity for filtering out the input noise.

Let us take a special case to illustrate the importance of small h on information forward-propagation.

Proposition 2. *Consider a deeper block $N = \mathrm{D}$ as the last residual block in ResNet, and suppose at any intermediate transformation i, $||\mathcal{F}(\mathbf{x}_i^\epsilon) - \mathcal{F}(\mathbf{x}_i)|| \leq \mathcal{W}$. The noise at layer D is denoted by $||\mathbf{x}_D^\epsilon - \mathbf{x}_D|| = \epsilon_D$, thus we have*

$$\epsilon_D \leq \epsilon + hD\mathcal{W}. \tag{12}$$

Its proof follows directly from the Eq. (11). Eq. (12) informs us that the small h can compensate the negative effects of noise accumulated over the depth D. In other words, when ResNet is deeper (larger D), we expect h to be smaller; while when ResNet is shallower, we allow h to be larger.

Here, we should be informed that, for a given depth of ResNet, we cannot reduce step factor h infinitely small. In the limiting case, if we take $h = 0$, there would be no transformations from initial feature $\mathbf{x}_0$ to the final state. In this case, the noise during the transformation is perfectly bounded, but it also smooths out all transformations.

4 Experiments

In this section, we conduct experiments on the vision dataset *CIFAR-10* [18] and the text dataset *AG-NEWS* [2]. We also employ a synthetic binary dataset *TWO-MOON* in Section 4.2 to illustrate input noise filtering along the forward propagation. We fix our step factor $h = 0.1$ and compare it with the original ResNet, which corresponds to $h = 1.0$, in Sections 4.1 and 4.2. We discuss how to select the step factor h in Section 4.3.

For the vision dataset *CIFAR-10*, the residual block $\mathcal{F}$ contains two 2-D convolutional operations [3]. For the text dataset *AG-NEWS*, the residual block $\mathcal{F}$ contains two 1-D convolutional operations [1]. For the synthetic dataset *TWO-MOON*, the residual block $\mathcal{F}$ contains two affine transformation matrices.

To tackle the dimensional mismatch in the shortcut connections of ResNet, we adopt the same practice as in [3] for *CIFAR-10* and [1] for *AG-NEWS*, which use convolutional layers of the kernel of size one to match the dimensions.

4.1 Small h helps training robustness

Small h stabilizes gradient and encourages smaller weights To illustrate the reason that small h can give extra training robustness compared with original ResNet (corresponding to $h = 1.0$), we train the ResNet with depth 218 on *CIFAR-10* and collect statistics w.r.t. weights and its gradients of

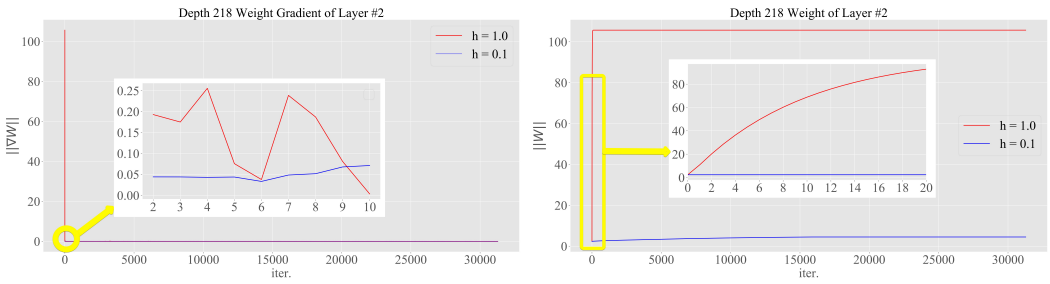


Figure 2: Gradients and weights dynamics over training iterations.

layer #2, as shown in the Figure 2. The statistic of ResNet with depth 110 and 218 over different layers are put into the Appendix 6.1.

Analysis 1 (exploding gradients problem). Figure 2 and Figures in the Appendix 6.1 show that in the early training stages, for the case of larger h, i.e., $h = 1.0$, there are the large (exploding) gradients, which makes the weights quickly reaching a plateau over a few initial iterations (red line). So at the early stage, this causes violent feature transformations, which may amplify the data collection noise in the training inputs. Thus it degrades DNN's performance. In addition, as shown in the Appendix 6.1, during the later-on training iterations, ResNet with larger h tends to have larger and bumpy gradients at its deep layers (red line); it may cause overshooting issues, thus adversely affecting its convergence.

The philosophy of describing DNN to PDE is that we expect features transform slowly and steadily along with the depth, where at the final layer features could be linearly classifiable. So we do not favor unstable gradients which cause the weights changing abruptly to large, leading to violent feature transformations.

Analysis 2 (large weights problem). Another observation is that larger h, e.g., $h = 1.0$ tends to encourage larger weights during the training procedure (red line). By showing this, we calculate averaged weights of layer #2 across all iterations. To avoid the coincidence, we carry on repeated experiments over two different random seeds. $h = 1.0$ corresponds to 51.12 and 105.63, but $h = 0.1$ corresponds to much smaller values 4.54 and 4.10. As it is shown by [24], a model with large weights is more complex than a model with smaller weights. It is a sign of a network that overfits the training data. In practice, we prefer to choose the simpler models (Occam's razor principle), so that we favor models with smaller weights. In addition, as stated by [25]. Large weights make the network unstable, minor variation or statistical noise on the expected inputs will result in large differences in the output.

In order to ameliorate such problems, existing tricks are brought up. E.g., (a) warming up training using small LR [3] and gradient clipping; this could counter the side effects of problems of large gradients and bumpy gradients. (b) weights decay, e.g., L2 regularizer; this could give an extra penalty on large weights during the training, such that it encourages to use simpler NN described by smaller weights. However, over different domains, highly trained experts are required fine-tuned deep neural network [26], e.g., how many iterations we should warm up training, how small LR is required, how much we should penalize weights during training. In addition, adding weight decays introduce extra computation for training the deep neural network. Choosing smaller LR will significantly affect training convergence, which requires more computational power to converge to the optimal.

By contrast, without specializing such tricks, smaller h, e.g. $h = 0.1$, can stabilize gradients over all training iterations; it encourages smaller weight for deep ResNet (blue line). In addition, small h does not introduce extra computation and is compatible with other stabilizing tricks.

Small h enables training deeper networks: To verify training robustness of small h with increasing depth of ResNet, in Figure 3 we compare ResNet with reduced step h ($h = 0.1$) and original ResNet (corresponding to $h = 1.0$) over various depths. We train ResNet using optimizer of SGD with 0.9 momentum; LR is fixed to 0.1 with 15 training epochs (*AG-NEWS*) and start LR is 0.1 with 80 epochs, divided by 10 at epoch #40 and #60 respectively (*CIFAR-10*). Each configuration has 5 trials with different random seeds. We provide median test accuracy with the standard deviation plotted as the shaded color.

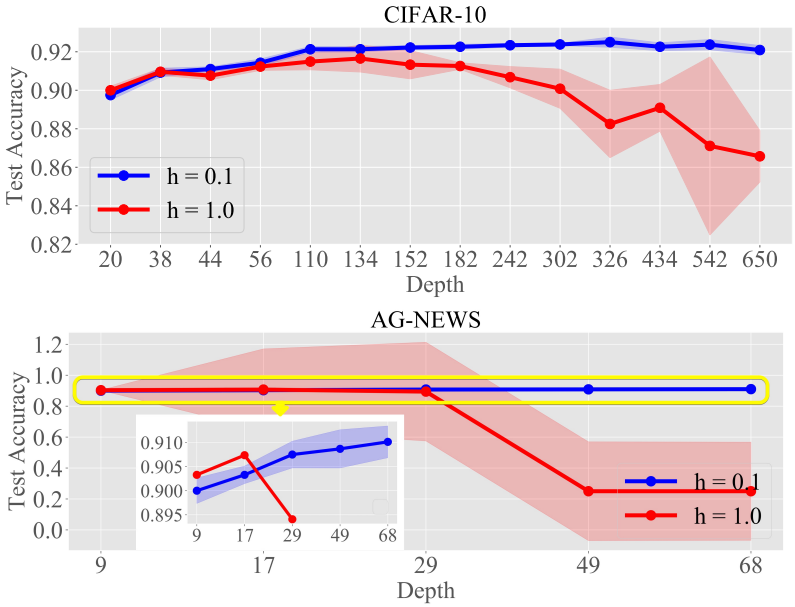


Figure 3: Training robustness comparisons of ResNet with increasing depths.

In both vision and text datasets, our method ($h = 0.1$) outperforms the existing method (corresponding to $h = 1.0$), when the networks go deeper. In particular, As the depth of ResNet increases, the existing method ($h = 1.0$) has the degradation problem, but ours (small h) has increasing (or stabilizing) performance. It is noted that, the blue shaded area ($h = 0.1$) is much thinner compared with the red one ($h = 1.0$). This shows that our method ($h = 0.1$) has a smaller variance of the test accuracy over different random seeds. This demonstrates that small h offers training robustness as well.

To sum up, as theoretically shown in Section 3.2, the reduced h can prevent the explosion of back-propagated information over the depth, thus making the training procedure of ResNet more robust to increasing depth.

Small h enables larger learning rate (LR): In Figure 4, we train ResNet with depth 218 for CIFAR-10 dataset and compare its training performance over different learning rates. we put more comparisons graphs into the Appendix 6.2, i.e, ResNet with depth 29 and 49 for *AG-NEWS*, and ResNet with depth 44 and 110 for *CIFAR-10*. We train ResNet using optimizer of SGD 0.9 momentum with 80 epochs, with different start LR divided by 10 at epoch #40 and #60 respectively (*CIFAR-10*), and with different fixed LR (*AG-NEWS*). Each configuration has 5 trails with different random seeds. Other hyperparameters remain the same. We provide test accuracy with different start LR. The standard deviation is plotted as the error bar.

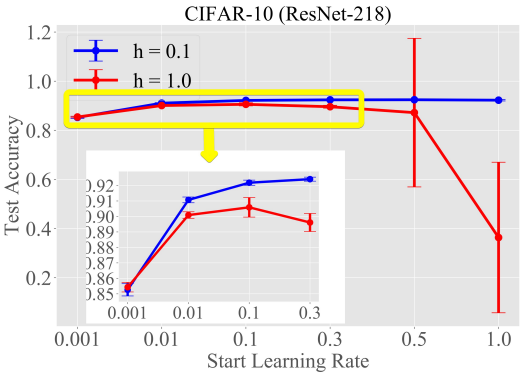


Figure 4: Training robustness comparisons of ResNet with different learning rates.

Figure 4 shows that our methods, i.e., smaller h (blue line) can enable larger LR for training without degrading its performance, due to that stable and smaller gradient throughout all training iterations over all layers (analyzed above). Compared with the larger h (red line), our method also has a narrow error bar, which signifies smaller performance variance. In addition, as shown in the Figure 13 in the Appendix 6.2, our method (blue line) has faster convergence rate, by utilizing larger training rate, e.g, start LR $= 0.3$. To sum up, small h makes the training procedure of ResNet more

robust to larger LR, so that it can enjoy the benefits using larger LR, i.e., faster convergence, with little chance of performance degradation.

Small h helps networks without applying BN: To verify the effectiveness of small h on improving robustness of the training procedure without applying BN, we compare ResNet with reduced step factor h ($h = 0.1$) and the original ResNet (corresponding to $h = 1.0$) without BN (other hyperparameters remain the same). We train ResNet using optimizer of SGD with 0.9 momentum; LR is fixed to 0.1 with 15 training epochs (AG-NEWS) and start LR is 0.1 with 80 training epochs, divided by 10 at epoch #40 and #60 respectively (*CIFAR-10*). Each configuration has 5 trials with different random seeds. We provide median test accuracy with the standard deviation plotted as the shaded color.

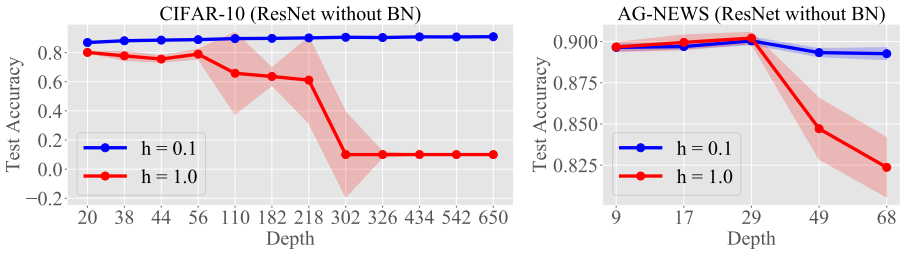


Figure 5: Training robustness comparisons of ResNet without applying BN.

Figure 5 shows that, without using BN, training plain ResNet ($h = 1.0$) is unstable and exhibits large variance for both vision *CIFAR-10* and text *AG-NEWS* datasets, especially when the network is deep. Particularly, without using BN, even training a ResNet-110 on *CIFAR-10* fails at a high chance (2/5). However, with the reduced $h = 0.1$, training performance improves significantly and exhibits low variance. As theoretically shown in Section 3.2 reduced h has beneficial effects on top of BN. This can help the back-propagated information by preventing its explosion, and thus enhance the training robustness of deep networks.

Small h helps networks on a different optimizer - ADAM: In order to verify that our method is robustness to different types of optimizer, In Figure 6, we train ResNet with depth 218 for *CIFAR-10* dataset and compare its training performance over different start LR, but use another popular optimizer - ADAM [27]. ADAM optimizer leverages the power of adaptive methods to find individual LR for each trainable parameter.

we put more comparisons graphs in the Appendix 6.3, i.e, ResNet with depth 44 and 110 for *CIFAR-10*. We train ResNet with 80 epochs, with different start LR divided by 10 at epoch #40 and #60 respectively (*CIFAR-10*). Each configuration has 5 trails with different random seeds. Other hyperparameters remain the same. We provide median test accuracy with the standard deviation plotted as the error bar.

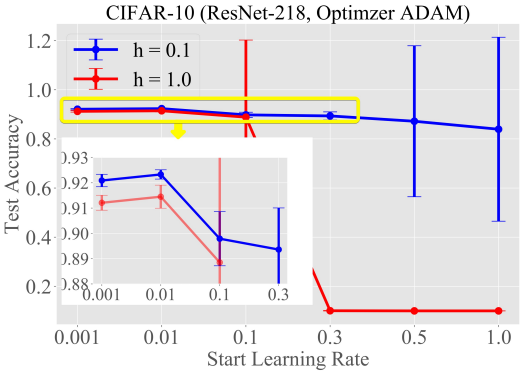


Figure 6: Training robustness comparisons of ResNet using optimizer ADAM with different learning rates.

In Figure 6, under the context of ADAM optimizer, our method (blue line) still exhibits great merits for training robustness. In particular, when start LR is bigger than 0.3, the existing method ($h = 1.0$) fails to train ResNet, however our method ($h = 0.1$) can still stabilizing the training procedure.

What is more, we also compared ResNet with small h and original ResNet ($h = 1.0$) over different ways of weight initialization. We put this comparision in the Appendix 6.4.

4.2 Small h helps generalization robustness

Synthetic data for noise filtering: To give insights on why small h can help the generalization robustness of ResNet, let us first consider a synthetic data example of using ResNet for binary classification task, namely separating noisy "red and blue" points in a 2-D plane. We train a vanilla ResNet (without BN) with $h = 1.0$ and ResNet with $h = 0.1$ on the left of Figure 7, and perform the feature transformations on the test set ("Transformation starting position" of Figure 7) using the learned ResNet.

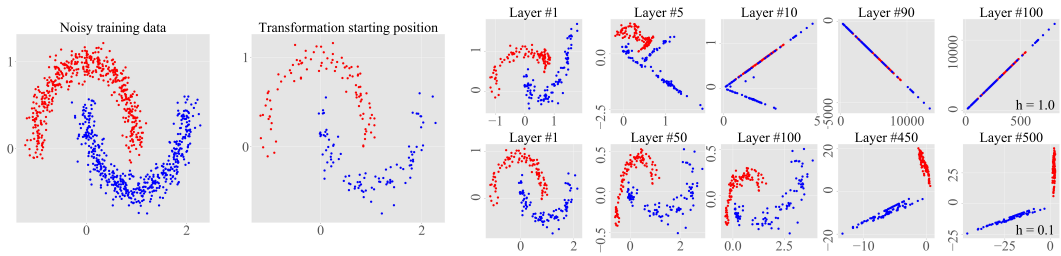


Figure 7: Illustration of feature transformations through the vanilla ResNet with step factor of h = 1.0 and h = 0.1 respectively.

A series of figures in the upper right position of Figure 7 illustrates that the features are transformed through forward-propagation in vanilla ResNet (no BN, $h = 1.0$). It shows that the noise in the input features leads to the mixing of red and blue points, sabotaging the generalization of ResNet. The reason for this phenomenon is that, with large h, features undergo violent transformations between two adjacent blocks due to larger weights, and negative effects of noise are amplified along with the depth.

On the other hand, with small h features undergo smooth transformations at every adjacent residual block, and thus the input noise is gradually filtered out, which leads to the correct classification (A series of figures in the lower right position in Figure 7). As stated in Section 3.3, small h can help to bound the negative effects of noise in input features. With small h, noise amplification along the depth is effectively constrained.

We also made animations visualizing the transformations and conducted experiments comparing the effects of BN on smoothing out noise. Interested readers may go to the *Dropbox* link for reference(supplement materials).

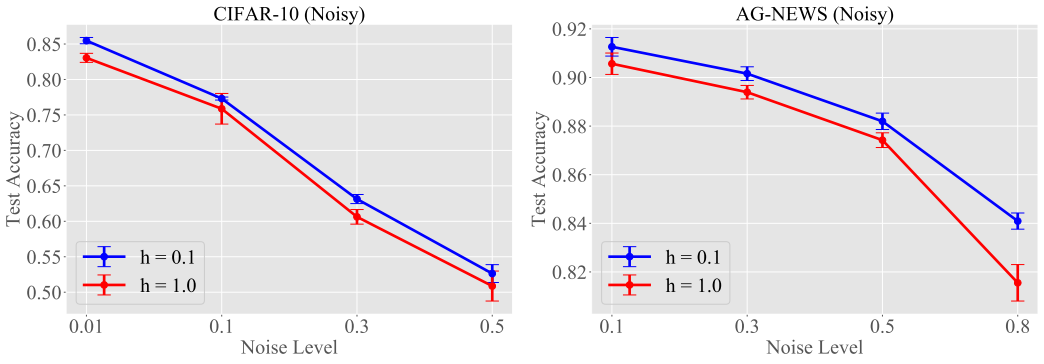


Figure 8: Performance of $h = 1.0$ and $h = 0.1$ on generalization robustness.

Real-world data for noise filtering: To verify the effectiveness of small h on generalization robustness of ResNet, in Figure 8 we train on noisy input data with various noise level (illustrations of noisy input are in the Appendix 6.5). We provide the test accuracy on clean input data of both visual and text dataset *CIFAR-10* (depth-218 ResNet) and *AG-NEWS* (depth-49 ResNet). We using optimizer of SGD with 0.9 momentum with starting LR 0.1 divided by 10 at epoch $\#40$ and $\#60$

(*CIFAR-10*), fixed LR to 0.01 (*AG-NEWS*). We compare test accracy of ResNet with reduced step factor ($h = 0.1$) and original ResNet ($h = 1.0$) Other hyperparameters remain the same. Each configuration takes 5 trails each with different random seeds. The standard deviation is plotted as the error bar.

Figure 8 shows that, at different noise levels, our method ($h = 0.1$) continuously outperforms the original method (corresponding to $h = 1.0$). We observe that ResNet with reduced h has the smaller variance compared to its counterpart under different noise levels. In other words, our method is robust to training on noisy input by bounding the negative effects of noise. By taking the smooth transformations, it gradually filters out the noise in the input features along the forward propagation of ResNet. Thus, our method offers better generalization robustness.

4.3 How to select step factor h

Last but not least, in Figure 9 we perform a grid search of h from 0.001 to 20 for *CIFAR-10* and from 0.001 to 1.0 for *AG-NEWS* to explore h selection strategies for optimal performance. We train ResNets over different step factors, using optimizer of SGD with 0.9 momentum with starting LR 0.1 divided by 10 at epoch #40 and #60 (*CIFAR-10*), and fixed LR to 0.1 (*AG-NEWS*). We provide the median test accuracy, with each configuration taking 5 trails with different random seeds. The standard deviation is plotted as the error bar.

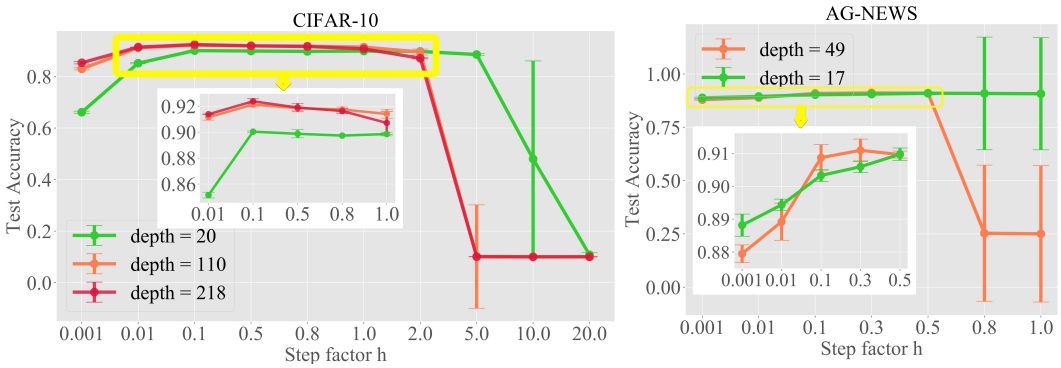


Figure 9: Selection of step factor h for ResNet with various depths.

Figure 9 shows that the proper h is near 0.1 for ResNet-110, 218 and $h \in [0.1, 20]$ for ResNet-20 for *CIFAR-10* dataset; for *AG-NEWS* dataset the proper h is near 0.3 for ResNet-49 and near 0.5 for ResNet-17.

When h is small, error bar is small as well, such that performance variance is small. By comparison, when h is very large, e.g., $h \geq 5$ for ResNet-110 (*CIFAR-10*), training becomes very unstable, which leads to training failure with a very high chance. Another example is ResNet-17, 49 for *AG-NEWS*. When $h \geq 0.8$, in our experiments, 3 out of 5 training procedures fail.

In addition, we also observe that when h is very small, e.g., $h = 0.001$, it will degrade its generalization performance, though the training variance is very small. This resonates the statement in the Section 3.3, i.e., too small h will smooth out the useful transformations.

To sum up, the guideline for selection of h is "smaller but not too small". To be specifical, for a deep ResNet we prefer h to be smaller, e.g., $h = 0.1$ for ResNet-218 (*CIFAR-10*), but for shallow ResNet, we can tolerate h to become larger, e.g., $h = 2.0$ for ResNet-20 (*CIFAR-10*). In addition, once depth of ResNet is fixed, the chosen h should not be too small, e.g., $h = 0.001$ and 0.01. Otherwise it smooths out useful transformations, which leads to performance degradation.

5 Conclusion

This paper proposed a simple but principled approach to enhance the robustness of ResNet. Motivated by the dynamical system view, we characterize ResNet by an explicit Euler method. We theoretically find that the step factor h in the Euler method can control the robustness of ResNet in both training

and generalization. From the view of back-propagation, we prove that a small h can benefit the training robustness, while from the view of forward-propagation, we prove that a small h can help the generalization robustness. We conduct comprehensive experiments on vision *CIFAR-10* and text *AG-NEWS* datasets. Experiments confirm that small h can benefit the training and generalization robustness. Future work can explore several promising directions: (a) How to transfer the experience of small h to other network structures, e.g., RNN for natural language processing [28], (b) How to handle the noisy labels y [29], and (c) Other means for choosing the step size h, e.g., using Bayesian optimization [30, 31, 32, 33].

References

[1] H. Schwenk, L. Barrault, A. Conneau, and Y. LeCun. Very deep convolutional networks for text classification. In *Proc. EACL*, pages 1107–1116, 2017.

[2] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proc. NeurIPS*, pages 649–657, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE CVPR*, pages 770–778, 2016.

[4] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *Proc. IEEE CVPR*, pages 2261–2269, 2017.

[5] A.-A. Liu, Z. Shao, Y. Wong, J. Li, Y.-T. Su, and M. Kankanhalli. LSTM-based multi-label video event detection. *Multimedia Tools and Applications*, 78(1):677–695, 2019.

[6] N. Xu, A.-A. Liu, Y. Wong, Y. Zhang, W. Nie, Y. Su, and M. Kankanhalli. Dual-stream recurrent neural network for video captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256, 2010.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, pages 630–645, 2016.

[11] L. J. Ba, R. Kiros, and G. E. Hinton. Layer normalization. arxiv:1607.06450, 2016.

[12] Y. Wu and K. He. Group normalization. In *Proc. ECCV*, pages 3–19, 2018.

[13] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In *Proc. NeurIPS*, pages 2488–2498, 2018.

[14] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1), 2017.

[15] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Proc. NeurIPS*, pages 6572–6583, 2018.

[16] Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proc. ICML*, pages 3282–3291, 2018.

[17] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. arxiv:1804.04272, 2018.

[18] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Univ. Toronto, 2009.

[19] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proc. ICML*, pages 807–814, 2010.

[20] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[21] K. E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2 edition, 2008.

[22] U. M. Ascher. *Numerical methods for evolutionary differential equations*, volume 5 of *Computational Science and Engineering*. SIAM, 2008.

[23] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[24] Nikhil Ketkar et al. *Deep Learning with Python*. Springer, 2017.

[25] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[26] Marc-André Zöller and Marco F Huber. Survey on automated machine learning. *arXiv preprint arXiv:1904.12054*, 2019.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[28] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*, pages 1724–1734, 2014.

[29] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proc. NeurIPS*, pages 8536–8546, 2018.

[30] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *JMLR*, 13:1809–1837, 2012.

[31] N. Srinivas, A. Krause, S. Kakade, and M. W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pages 1015–1022, 2010.

[32] E. A. Daxberger and B. K. H. Low. Distributed batch Gaussian process optimization. In *Proc. ICML*, pages 951–960, 2017.

[33] T. N. Hoang, Q. M. Hoang, R. Ouyang, and K. H. Low. Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proc. AAAI*, 2018.

6 Appendix

6.1 Small h stabilizes gradient and encourages smaller weights

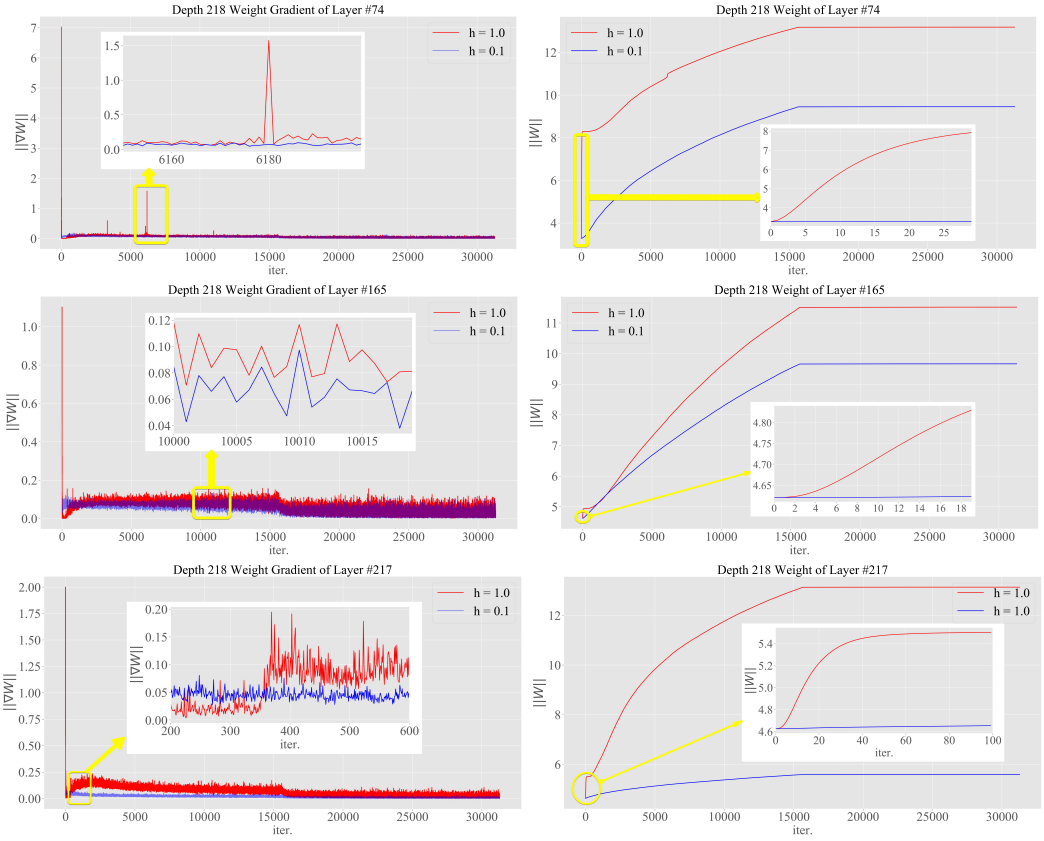


Figure 10: Gradients and weights dynamics over training iterations of ResNet-218 on *CIFAR-10*.

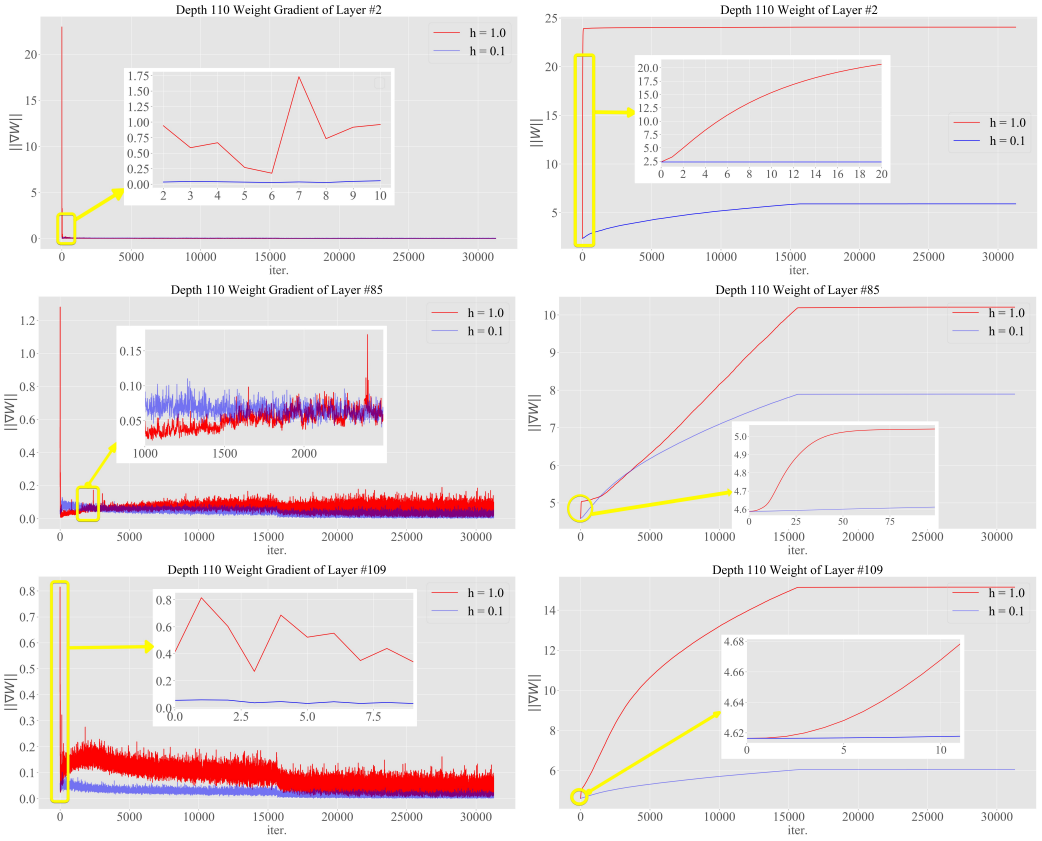


Figure 11: Gradients and weights dynamics over training iterations of ResNet-110 on *CIFAR-10*.

6.2 Small h enables larger learning rate (LR)

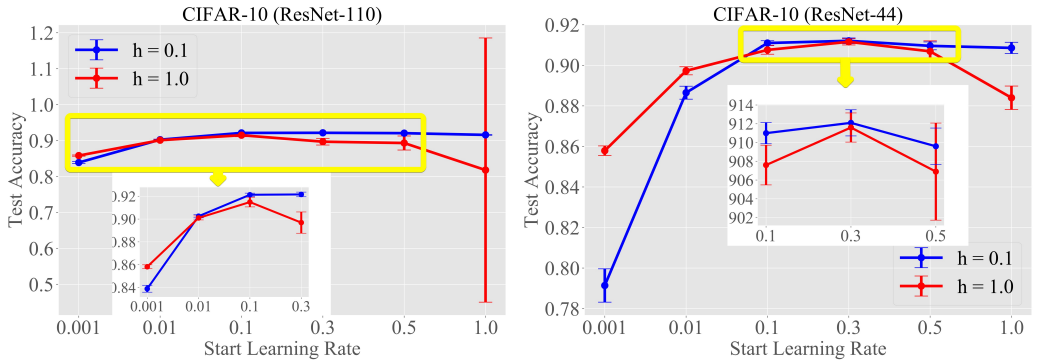


Figure 12: Training robustness comparisons of ResNet with different learning rates on *CIFAR-10*. We train ResNet using optimizer SGD with 0.9 momentum. Start LR is divided by 10 at epoch #40 and #60 respectively. Each configuration takes 5 trials with different random seeds. The other hyperparameters remain the same. We provide test accuracy with different start learning rates. The standard deviation is plotted at the error bar.

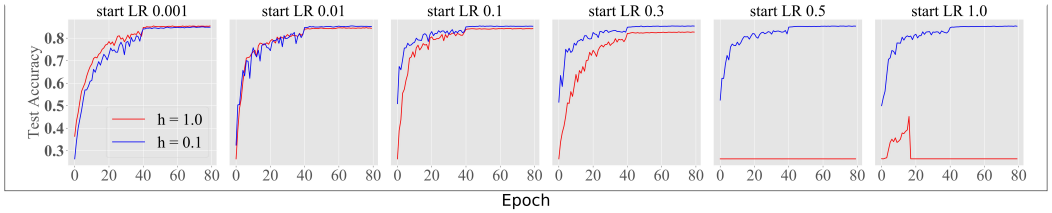


Figure 13: Convergence rate visualization on different learning rates(LR). We train ResNet-218 on *CIFAR-10*, using optimizer SGD with 0.9 momentum. LR is divided by 10 at epoch #40 and #60 respectively. We provide test accuracy over epochs.

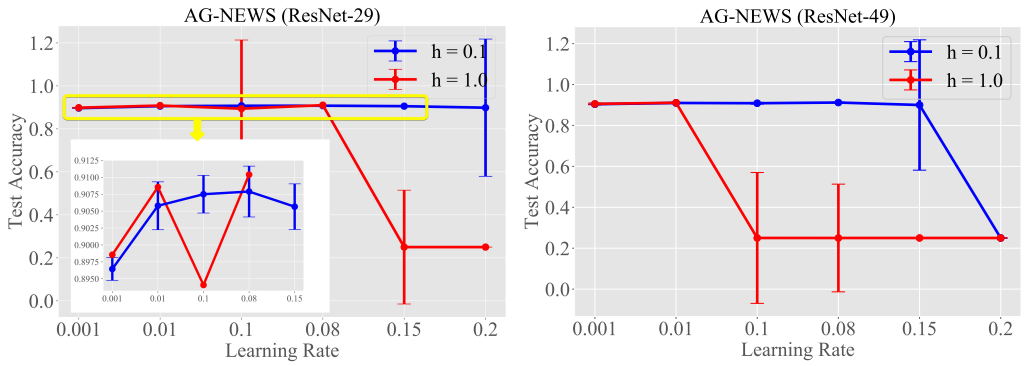


Figure 14: Training robustness comparisons of ResNet with different learning rates on *AG-NEWS*. We train ResNet using optimizer SGD with 0.9 momentum. LR is fixed throughout 15 epochs training. Each configuration takes 5 trials with different random seeds. The other hyperparameters remain the same. We provide test accuracy with different learning rates. The standard deviation is plotted at the error bar. For the zoomed patch, we do not show the error bar of red line for clarity.

6.3 Small h helps networks on a different optimizer - ADAM:

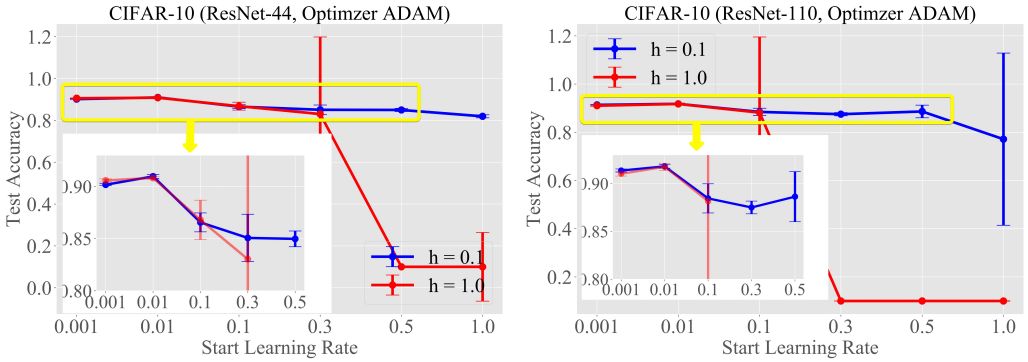


Figure 15: Training robustness comparisons of ResNet with different learning rates using ADAM optimizer. Start LR is divided by 10 at epoch #40 and #60 respectively. Each configuration takes 5 trials with different random seeds. The other hyperparameters remain the same. We provide test accuracy with different start learning rates. The standard deviation is plotted at the error bar.

6.4 Comparisons on different types of weight initialization on training robustness

We carry on experiments comparing different ways of weight initialization. Our experiments show that our method (small h) is still competitive on using different types of initialization on various depths of ResNet.

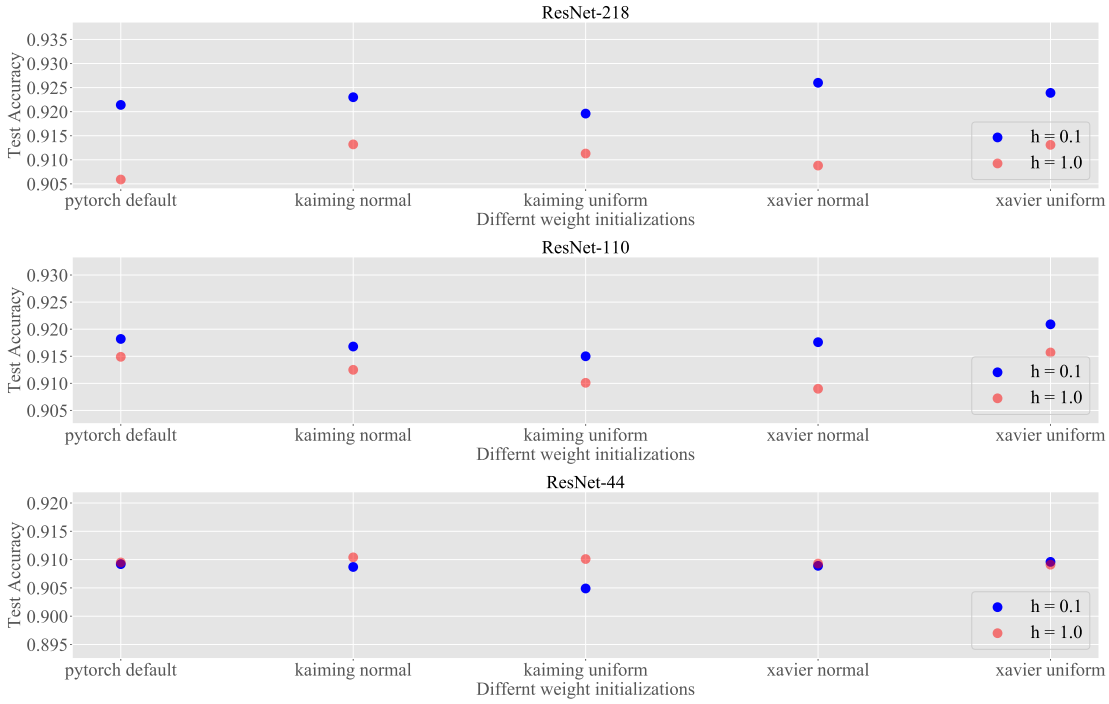
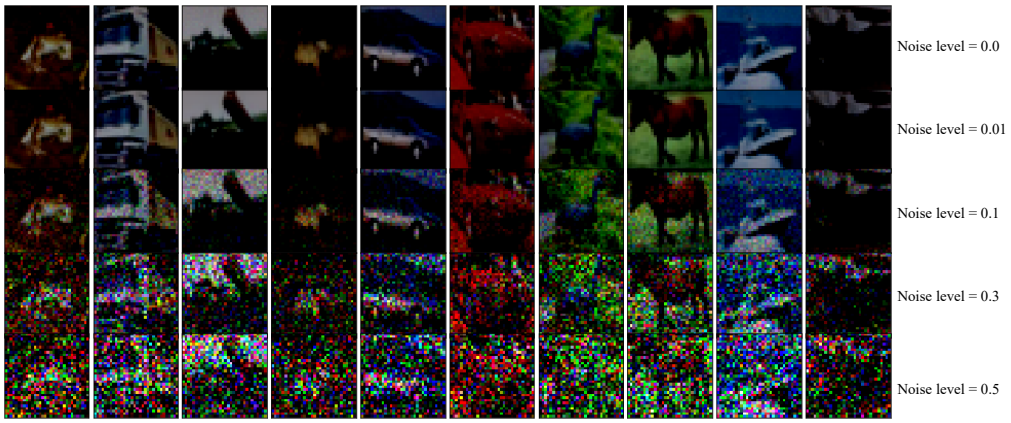


Figure 16: Comparisons on different types of weight initialization. We train ResNet on *CIFAR-10* using optimizer SGD with 0.9 momentum with start LR 0.1 divided by at 10 at epoch #40 and #60 respectively. We provide the test accuracy over different ways of weight initialization.

6.5 Visualization on noisy input

In Section 3.3, we train ResNet with $h = 0.1$ and $h = 1.0$ on noisy data (i.e., input has perturbations), and test it on clean data.

For the training dataset *CIFAR-10*, we inject Gaussian noise at every normalized pixel with zero mean and a different standard deviation to represent different noise levels. For the training dataset *AG-NEWS*, we choose different proportions (different noise levels) of characters randomly in the texts and alter them.



video games good for children computer games can promote problem-solving and team-building in children, say games industry experts. (Noise level = 0.0)

vedeo games good for dhildlenzcospxter games can iromote problem-sorvtng and teai-building in children, sby games industry experts. (Noise level = 0.1)

video nawvs zgood foryxhilqretngomvumer games cahcprocotubpnoblex-szbvina and tqlmmbuaddiagjin whipdren, saywgsmes ildustry exmrrts. (Noise level = 0.3)

tmdeo gakec jgopd brr cgildrenjcoogwdeh lxdeu vanspromote xrobkeh-svlkieo and termwwuojvinguinfcojbdses, sacosamlt cndgstoyaagpbrus. (Noise level = 0.5)

vizwszgbrwjtguihcxfoatbhivrrwvq cxmpgugflziwls clfnzrommtohprtblef-solvynx rnjnyiaf-gjwlcergwklskqibdtjn,aoty gameshinzustrm oxpertsdm (Noise level = 0.8)

Figure 17: Input noise visualization of *CIFAR-10* and *AG-NEWS*