Towards Robust ResNet: A Small Step but a Giant Leap

Jingfeng Zhang[1] , **Bo Han**[2] , **Laura Wynter**[3] , **Bryan Kian Hsiang Low**[1] and **Mohan Kankanhalli**[1]
[1]Department of Computer Science, National University of Singapore
[2]RIKEN Center for Advanced Intelligence Project
[3]IBM Research, Singapore
j-zhang@comp.nus.edu.sg, bo.han@riken.jp, lwynter@sg.ibm.com, {lowkh, mohan}@comp.nus.edu.sg

Abstract

This paper presents a simple yet principled approach to boosting the robustness of the *residual network* (ResNet) that is motivated by a dynamical systems perspective. Namely, a deep neural network can be interpreted using a partial differential equation, which naturally inspires us to characterize ResNet based on an explicit Euler method. This consequently allows us to exploit the step factor h in the Euler method to control the robustness of ResNet in both its training and generalization. In particular, we prove that a small step factor h can benefit its training and generalization robustness during backpropagation and forward propagation, respectively. Empirical evaluation on real-world datasets corroborates our analytical findings that a small h can indeed improve both its 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 [Conneau *et al.*, 2017; Zhang *et al.*, 2015], image recognition [He *et al.*, 2016a; Huang *et al.*, 2017], and video analysis due to their capability of approximating any universal function. However, DNNs are often difficult to train due in a large part to the vanishing gradient phenomenon [Glorot and Bengio, 2010; Ioffe and Szegedy, 2015]. The *residual network* (ResNet) [He *et al.*, 2016a] was proposed to alleviate this issue using its key component known as the skip connection, which creates a "bypass" path for information propagation [He *et al.*, 2016b].

Nevertheless, it remains challenging to achieve robustness in training a very deep DNN. As the DNN becomes deeper, it requires more careful tuning of its model hyperparameters (e.g., learning rate, number of layers, choice of optimizer) to perform well. This issue can be mitigated using normalization techniques such as *batch normalization* (BN) [Ioffe and Szegedy, 2015], layer normalization [Lei Ba *et al.*, 2016], and group normalization [Wu and He, 2018], among which BN is most widely used. BN normalizes the inputs of each layer to enable robust training of DNNs. It has been shown that BN provides a smoothing effect of the optimization landscape [Santurkar *et al.*, 2018], thus ameliorating the issue of a vanishing gradient [Ioffe and Szegedy, 2015].

Unfortunately, we have observed in our experiments (Section 4.1) that a very deep DNN can potentially (and surprisingly) experience exploding gradients in shallow layers in early training iterations, even when it is coupled with BN. As a result, its weights change drastically, which in turn causes violent feature transformations, hence impairing its robustness in training. Our subsequent experiments [Zhang *et al.*, 2019a] also show that the gradients are large and bumpy in deep layers in the later training iterations, which destabilizes the training procedure. Besides, noisy data (e.g., images with mosaics and texts with spelling errors) can adversely impact the training procedure, in which the noise may be amplified through forward propagation of features, thus degrading its robustness in generalization.

This paper presents a simple yet principled approach to boosting the robustness of ResNet in both training and generalization that is motivated by a dynamical systems perspective [Chen *et al.*, 2018; Lu *et al.*, 2018; Ruthotto and Haber, 2018; Weinan, 2017]. Namely, a DNN can be interpreted using a partial differential equation, which naturally inspires us to characterize ResNet based on an explicit Euler method. This consequently allows us to exploit the step factor h in the Euler method to control the robustness of ResNet in both its training and generalization. In our work here, **training robustness** refers to the stability of model training with an increasing depth, a larger learning rate, and different types of optimizer, while **generalization robustness** refers to how well a trained model generalizes to classify test data whose distribution may not match that of the training data. To analyze the effects of step factor h, we prove that a small h can benefit the training and generalization robustness of ResNet during backpropagation and forward propagation, respectively. Empirical evaluation on real-world vision- and text-based datasets corroborates our analytical findings that a small h can indeed improve both its training and generalization robustness.

2 Background and Notations

Before delving into a robust ResNet (Section 3), we review the necessary background information which includes

ResNet, batch normalization, and partial differential equations.

A *residual network* (ResNet) is a variant of the DNN that exhibits competitive predictive accuracy and convergence properties [He *et al.*, 2016a]. The ResNet comprises many stacked residual blocks with a key component called the skip connection, each of which has the following structure:

$$\mathbf{y}_n \triangleq \mathbf{x}_n + \mathcal{F}(\mathbf{x}_n)\,, \quad \mathbf{x}_{n+1} \triangleq I(\mathbf{y}_n) \tag{1}$$

for layer n where $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ are, respectively, the input and output of residual block n, $\mathcal{F}$ is a residual block performing feature transformations (e.g., convolutional operations or affine transformation), and I is a component-wise operation (e.g., ReLU function [Nair and Hinton, 2010] or identity mapping [He *et al.*, 2016b]). Based on the core idea of the skip connection, variants of ResNet have been proposed for specific tasks such as DenseNet for image recognition [Huang *et al.*, 2017] and VDCNN with shortcut connections for text classification [Conneau *et al.*, 2017].

Batch normalization (BN) has been widely adopted for training DNNs [Ioffe and Szegedy, 2015] and normalizes the input of a layer over each mini-batch of the training data. Specifically, for the input $\mathbf{x} \triangleq (x^{(k)})^\top_{k=1,\ldots,d}$ of a layer with dimension d, BN first normalizes each scalar input feature $\hat{x}^{(k)} \triangleq (x^{(k)} - \mu^{(k)})/\sigma^{(k)}$ for $k = 1, \ldots, d$ independently where $\mu \triangleq (\mu^{(k)} \triangleq \mathbb{E}[x^{(k)}])^\top_{k=1,\ldots,d}$ and $\sigma \triangleq (\sigma^{(k)} \triangleq \sqrt{\mathrm{Var}[x^{(k)}]})^\top_{k=1,\ldots,d}$ are computed over a mini-batch of size m. Then, it performs an affine transformation of each normalized scalar input feature $\hat{x}^{(k)}$ by $\mathrm{BN}(x^{(k)}) \triangleq \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$ where $\gamma \triangleq (\gamma^{(k)})^\top_{k=1,\ldots,d}$ and $\beta \triangleq (\beta^{(k)})^\top_{k=1,\ldots,d}$ are learned during training. BN has been shown to smooth the optimization landscape of DNNs [Santurkar *et al.*, 2018]. This offers more stable gradients for the robust training of DNNs, thus ameliorating the issue of a vanishing gradient [Ioffe and Szegedy, 2015].

A DNN performs nonlinear feature transformations of the input such that the transformed features can match the corresponding target output (e.g., categorical labels for classification and continuous quantities for regression). To achieve this, a DNN transforms the input through multiple layers such that the transformed features in the last layer become linearly separable [Haber and Ruthotto, 2017].

Let us now consider the dynamical systems perspective: A PDE can characterize the motion of particles [Ascher, 2008; Atkinson, 2008]. This motivates us to characterize the feature transformations in a DNN as a system of first-order PDEs:

Definition 1. *The feature transformations in a DNN can be characterized by a first-order PDE $f : T \times X \to \mathbb{R}^d$ where $T \subseteq \mathbb{R}^+ \cup \{0\}$ and $X \subseteq \mathbb{R}^d$:*

$$\dot{\mathbf{x}} \triangleq \partial\mathbf{x}/\partial t \triangleq f(t, \mathbf{x})$$

where $t \in T$ is the time along the feature transformations and $\mathbf{x} \in X$ is a feature vector of dimension d.

Let $\mathbf{x}(t)$ denote a transformed feature vector at time t. Given an initial input feature vector $\mathbf{x}(0)$, a PDE $\dot{\mathbf{x}} = f(t, \mathbf{x})$ gives rise to the *initial value problem* (IVP). A solution to an IVP is a function of time t. For example, consider the IVP:

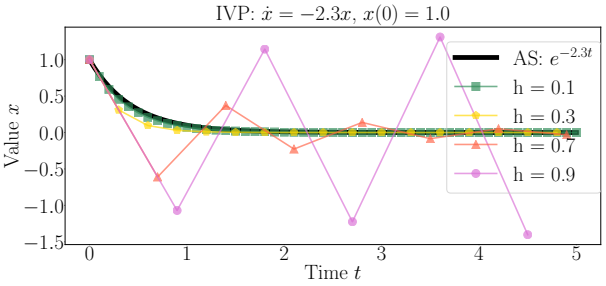

Figure 1: The *analytic solution* (AS) of an IVP and its approximations by an explicit Euler method with different step factors h.

$\dot{x} = -2.3x$ and $x(0) = 1$. Then, its analytic solution can be derived as $x(t) = \exp(-2.3t)$, as shown in Fig. 1. For more complicated PDEs, it is not always possible to derive an analytic solution. So, their solutions are approximated by numerical methods, of which the explicit Euler method (see Definition 2 below) is the most classic example.

3 Towards Robust ResNet

A ResNet can naturally be described by the explicit Euler method (Definition 2), which gives us an Euler view of ResNet (3). Inspired by the stability of the Euler method, the step factor h for ResNet can enable smoother feature transformations due to gradual feature transitions at every consecutive block during forward propagation. The benefits of such smoothness are two-fold: (a) preventing information explosion over a large depth (Section 3.2) and (b) mitigating the adverse effect of noise in the input features (Section 3.3), both of which can significantly boost the robustness of ResNet.

3.1 Connections of ResNet with the Euler Method

Definition 2. *The explicit Euler method [Atkinson, 2008; Ascher, 2008] can be represented by*

$$\mathbf{x}_{n+1} \triangleq \mathbf{x}_n + h\, f(t_n, \mathbf{x}_n) \tag{2}$$

where $\mathbf{x}_0 \triangleq \mathbf{x}(0)$ and $\mathbf{x}_n$ is an approximation of $\mathbf{x}(t_n)$.

Given the solution $\mathbf{x}(t_n)$ at time t_n, the PDE $\dot{\mathbf{x}} = f(t, \mathbf{x})$ intuitively indicates "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 over a small time step from t_n to $t_n + h$. To obtain a reasonable approximation, the step factor h in the Euler method has to be chosen to be "sufficiently small". Its magnitude depends on the PDE and its given initial input. For instance, in Fig. 1, we use the Euler method with various step factors h to approximate the solution of the IVP: $\dot{x} = -2.3x$ and $x(0) = 1$. It can be observed that the approximation of x improves with a smaller h.

By setting $h = 1$ and $f(t_n, \mathbf{x}_n) = \mathcal{F}(\mathbf{x}_n)$, it can be observed that the Euler method (2) characterizes the forward propagation of the original ResNet structure (1) [Haber and Ruthotto, 2017]. To generalize ResNet using the Euler method (2), we characterize the forward propagation of an *Euler view of ResNet* in the following way: Following the convention in [He *et al.*, 2016b], ResNet stacks multiple

residual blocks with step factor $h \in \mathbb{R}^+$, each of which has the following structure:

$$\mathbf{y}_n \triangleq \mathbf{x}_n + h\,\mathcal{F}(\mathbf{x}_n, W_n, \mathrm{BN}_n)\,, \quad \mathbf{x}_{n+1} \triangleq I(\mathbf{y}_n) \tag{3}$$

for layer n where batch normalization BN_n applies directly after (or before) the feature transformation function parameterized by weights W_n (e.g., convolutional operations or affine transformation matrix).

For any fixed t, the Euler method provides a more refined approximation with a smaller h, as analyzed in [Butcher, 2016]. Can a small h also improve the robustness of a deep ResNet in training and generalization? To answer this question, we will prove that a small h can prevent information explosion during backpropagation over a large depth (Section 3.2) and mitigate the adverse effect of noise in the input features during forward propagation (Section 3.3).

3.2 Training Robustness: Effect of Small h on Information Backpropagation

To simplify the analysis, let I in (3) be an identity mapping, that is, $\mathbf{x}_{n+1} = \mathbf{y}_n$. Then, by recursively applying (3),

$$\mathbf{x}_N = \mathbf{x}_n + h \sum_{i=n}^{N-1} \mathcal{F}(\mathbf{x}_i, W_i, \mathrm{BN}_i) \tag{4}$$

for any block N deeper than any shallower block n. We will use (4) to analyze the information backpropagation. Let the loss function be denoted by L. Using chain rule,

$$\frac{\partial L}{\partial \mathbf{x}_n} = \frac{\partial L}{\partial \mathbf{x}_N}\frac{\partial \mathbf{x}_N}{\partial \mathbf{x}_n} = \frac{\partial L}{\partial \mathbf{x}_N}\left(\mathbf{1} + h\frac{\partial}{\partial \mathbf{x}_n}\sum_{i=n}^{N-1}\mathcal{F}(\mathbf{x}_i, W_i, \mathrm{BN}_i)\right) \tag{5}$$

where $\mathbf{1}$ denotes an identity matrix. From (5), the backpropagated information $\partial L/\partial \mathbf{x}_n$ can be decomposed into two additive terms: (a) The first term $\partial L/\partial \mathbf{x}_N$ propagates information directly without going through the weights of any layer, and (b) the second term $h(\partial L/\partial \mathbf{x}_N)\partial(\sum_{i=n}^{N-1}\mathcal{F}(\mathbf{x}_i, W_i, \mathrm{BN}_i))/\partial \mathbf{x}_n$ propagates through the weights of layers $n, \ldots, N-1$. The first term ensures that information $\partial L/\partial \mathbf{x}_n$ does not vanish [He *et al.*, 2016b]. However, the second term can blow up the information $\partial L/\partial \mathbf{x}_n$, especially when the weights of layers $n, \ldots, N-1$ are large. A standard approach to resolving this issue is to apply BN after the feature transformation function [Ioffe and Szegedy, 2015].

Let us first study the effect of BN. To ease analysis, we examine the residual block n where $N = n+1$, $\mathcal{F}(\mathbf{x}_n, W_n, \mathrm{BN}_n) \triangleq \mathrm{BN}_n(\mathbf{x}'_n \triangleq W_n\mathbf{x}_n)$, and $\mathrm{BN}_n(\mathbf{x}'_n) \triangleq \gamma(\mathbf{x}'_n - \mu)/\sigma + \beta$ is a component-wise operation. Let σ_n denote the smallest component in σ. Then, $\|\partial\mathcal{F}(\mathbf{x}_n)/\partial \mathbf{x}_n\| \leq \|\gamma\|\|W_n\|/\sigma_n$. Consequently, it follows from (5) that

$$\left\|\frac{\partial L}{\partial \mathbf{x}_n}\right\| \leq \left\|\frac{\partial L}{\partial \mathbf{x}_{n+1}}\right\|\left(1 + \frac{h}{\sigma_n}\|\gamma\|\|W_n\|\right). \tag{6}$$

As observed in [Santurkar *et al.*, 2018], σ tends to be large. So, BN has the effect of constraining the explosive backpropagated information (6). However, as a ResNet grows deeper, σ tends to be highly uncontrollable in practice and the backpropagated information still accumulates over a large depth and can once again blow up. For example, Fig. 2 shows that its gradients are very large and unstable in early training iterations (red line). As a result, when a deep ResNet is trained on the CIFAR-10 or AG-NEWS dataset, its performance is much worse than that of its shallower counterparts, as shown in Fig. 3 (red line). In contrast, Fig. 2 also shows that a reduced h can serve to re-constrain the explosive backpropagated information (blue line). In fact, even without BN, reducing h can still serve to stabilize the training procedure, as detailed in [Zhang *et al.*, 2019a].

Our first theoretical result analyzes the effect of small h on information backpropagation:

Proposition 1. *Let $n = 0$ and $N = D$ where D is the index of last residual block. Suppose that $\|\partial\mathcal{F}(\mathbf{x}_i)/\partial \mathbf{x}_i\| \leq \|\gamma\|\|W_i\|/\sigma_i \leq \mathcal{W}$ for $i = 0, \ldots, D-1$. Then,*

$$\left\|\frac{\partial L}{\partial \mathbf{x}_0}\right\| \leq \left\|\frac{\partial L}{\partial \mathbf{x}_D}\right\|\left(\sqrt{d} + h\mathcal{W}\right)^D.$$

Its proof is in [Zhang *et al.*, 2019a]. Note that $\mathcal{W}$ intuitively upper bounds the effects of BN. Also, since $\sqrt{d} + h\mathcal{W} \geq 1$, the backpropagated information explodes exponentially w.r.t. the depth D, which directly affects the gradients and hurts the training robustness of a ResNet. Fortunately, reducing h can give extra control of the backpropagated information since the term $(\sqrt{d} + h\mathcal{W})^D$ can be constrained by h and the backpropagated information $\partial L/\partial \mathbf{x}_0$ is thus less likely to explode. This demonstrates that when a ResNet grows deeper, the step factor h should be reduced to a smaller value.

3.3 Generalization Robustness: Effect of Small h on Information Forward Propagation

It can be observed from (3) that a reduced h gives extra control of the feature transformations in a ResNet by making them smoother, that is, smaller $\|\mathbf{x}_{n+1} - \mathbf{x}_n\|$. More importantly, as the features propagate forward through a deep ResNet, the adverse effect of the noise in the input features can be mitigated over a large depth. In particular, we will prove that a reduced h can help to stabilize the target output of a ResNet against noise in the input features.

Let $\mathbf{x}_0^\epsilon$ be a perturbation from $\mathbf{x}_0$ (i.e., $\|\mathbf{x}_0^\epsilon - \mathbf{x}_0\| \leq \epsilon$) and $\mathbf{x}_N^\epsilon$ be the corresponding transformed feature vector in the ResNet. Then, from (4),

$$\begin{aligned}\|\mathbf{x}_N^\epsilon - \mathbf{x}_N\| &= \left\|\mathbf{x}_0^\epsilon + h\sum_{i=0}^{N-1}\mathcal{F}(\mathbf{x}_i^\epsilon) - \mathbf{x}_0 - h\sum_{i=0}^{N-1}\mathcal{F}(\mathbf{x}_i)\right\| \\ &\leq \epsilon + h\sum_{i=0}^{N-1}\|\mathcal{F}(\mathbf{x}_i^\epsilon) - \mathcal{F}(\mathbf{x}_i)\|\,.\end{aligned} \tag{7}$$

It can be observed from (7) that the noise in the input features is amplified with an increasing depth of the ResNet. Fortunately, by introducing a reduced h, the noise amplification can be limited and its adverse effect can thus be mitigated.

Our next theoretical result analyzes the effect of small h on information forward propagation:

Proposition 2. *Consider the last residual block $N = D$. Let the noise in the last layer D be denoted by $\epsilon_D \triangleq \|\mathbf{x}_D^\epsilon - \mathbf{x}_D\|$. Suppose that $\|\mathcal{F}(\mathbf{x}_i^\epsilon) - \mathcal{F}(\mathbf{x}_i)\| \leq \mathcal{W}$ for $i = 0, \ldots, D-1$. Then,*

$$\epsilon_D \leq \epsilon + hD\mathcal{W} \,. \tag{8}$$

Its proof follows directly from (7). It can be observed from (8) that a small h can mitigate the adverse effect of noise that is accumulated over a large depth D. Hence, a deeper ResNet (i.e., larger D) requires a smaller h while a shallower ResNet allows for a larger h.

Note that for a given depth of ResNet, the step factor h cannot be reduced to be infinitesimally small. In the limiting case of $h = 0$, though the noise after the feature transformations would be perfectly bounded, there is no transformation of the initial input feature vector $\mathbf{x}_0$, that is, all feature transformations are smoothed out.

4 Experiments and Discussion

In this section, we conduct experiments on the vision-based CIFAR-10 dataset [Krizhevsky and Hinton, 2009] and the text-based AG-NEWS dataset [Zhang *et al.*, 2015]. We also employ a synthetic binary TWO-MOON dataset in Section 4.2 to illustrate how the adverse effect of noise in the input features is mitigated along the forward propagation. We fix our step factor $h = 0.1$ and compare it with the original ResNet (i.e., $h = 1$) in Sections 4.1 and 4.2. We will discuss how to select the step factor h in Section 4.3.

For the vision-based CIFAR-10 dataset, the residual block $\mathcal{F}$ contains two 2D convolutional operations [He *et al.*, 2016a]. For the text-based AG-NEWS dataset, the residual block $\mathcal{F}$ contains two 1D convolutional operations [Conneau *et al.*, 2017]. For the synthetic TWO-MOON dataset, 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 that in [He *et al.*, 2016a] for the CIFAR-10 dataset and in [Conneau *et al.*, 2017] for the AG-NEWS dataset by using convolutional layers with the kernel of size one to match the dimensions. Unless specified otherwise, the default optimizer is SGD with 0.9 momentum. We train a ResNet using the CIFAR-10 dataset for 80 epochs with an initial *learning rate* (LR) of 0.1 that is divided by 10 at epochs 40 and 60. We train another ResNet using the AG-NEWS dataset with a fixed LR of 0.1 for 15 epochs.

4.1 Small h Improves Training Robustness

Small h on Stable Gradients and Smaller Weights

To illustrate why a small h can give extra training robustness as compared with the original ResNet (i.e., $h = 1.0$), we train a ResNet with depth 218 on the CIFAR-10 dataset and collect statistics regarding the weights and corresponding gradients of layer 2, as shown in Fig. 2. Such statistics for ResNets with depths 110 and 218 over different layers are detailed in [Zhang *et al.*, 2019a].

Fig. 2 and the other figures in the appendix of [Zhang *et al.*, 2019a] show that in the early training iterations, for the case of large h (i.e., $h = 1$), there are *exploding gradients*

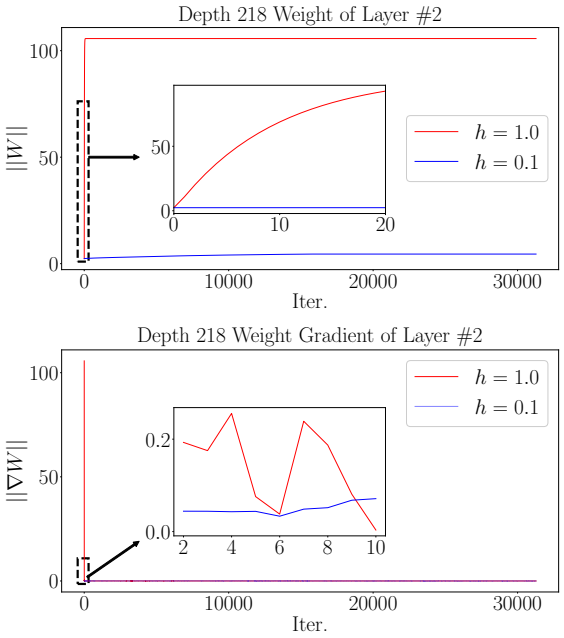

Figure 2: Dynamics of weights (top) and corresponding gradients (bottom) of layer 2 of ResNet with depth 218 over training iterations.

which make the weights quickly reach a plateau in a few initial iterations (red line). In the early training iterations, this causes violent feature transformations, which may amplify the noise in the training inputs, thus degrading the performance of ResNet. Furthermore, as shown in [Zhang *et al.*, 2019a], in the later training iterations, a ResNet with a larger h tends to have larger and bumpy gradients in its deep layers (red line); this may cause the issue of overshooting, thus adversely affecting its convergence.

The philosophy of characterizing a ResNet using a PDE is that the features should transform gradually over a large depth such that the transformed features in the last layer are linearly separable. So, we do not favor unstable gradients that cause the weights to change drastically, which in turn causes violent feature transformations.

It can also be observed from Fig. 2 that a larger h (e.g., $h = 1$) tends to encourage *larger weights* during the training procedure (red line). To show this, we calculate the averaged weights of layer 2 across all iterations. We repeat these experiments over two different random seeds. The case of $h = 1$ yields averaged weights of 51.12 and 105.63 while the case of $h = 0.1$ yields much smaller averaged weights of 4.54 and 4.10. As shown in [Ketkar, 2017], a trained model with large weights is more complex than that with small weights and tends to indicate overfitting to the training data. In practice, it is preferable to choose simpler models (i.e., Occam's razor principle). So, we favor models with smaller weights. In addition, as stated in [Reed and Marks II, 1999], large weights make the network unstable such that minor variation or statistical noise in the inputs will result in large differences in the target output.

To mitigate such an issue, a number of techniques have been proposed: (a) Warming up the training with a small

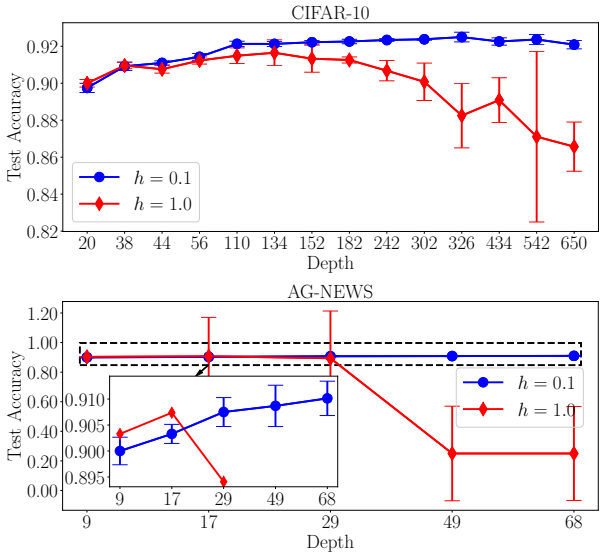

Figure 3: Comparison of training robustness of ResNets with increasing depths.

LR [He *et al.*, 2016a] and gradient clipping can counter the side effects of the issues of large and bumpy gradients, and (b) weight decay (e.g., L2 regularizer) can provide an extra penalty on large weights during training such that it encourages using a simpler DNN represented by smaller weights. However, these techniques require tedious fine-tuning by highly trained domain experts (e.g., how many iterations to run in the warm-up phase of training, how to set LR to be sufficiently small, how much to penalize weights during training). Furthermore, adding weight decay introduces extra computation to train a DNN and choosing a smaller LR will significantly affect training convergence, which requires more computational power for convergence. In contrast, without employing specialized techniques, our proposed use of a smaller h (e.g., $h = 0.1$) can serve to stabilize gradients over all training iterations; it encourages smaller weights for a deep ResNet, as shown in Fig. 2 (blue line). In addition, a small h does not introduce extra computation and is compatible with the above techniques.

Small h Enables Training Deeper ResNets

To verify the training robustness of a small h with an increasing depth of ResNet, Fig. 3 compares a ResNet with a reduced step factor $h = 0.1$ with the original ResNet (i.e., $h = 1$) over varying depths. Each configuration has 5 trials with different random seeds. We provide the median test accuracy with the standard deviation plotted as the error bar.

In both CIFAR-10 and AG-NEWS datasets, our ResNet with $h = 0.1$ outperforms that with $h = 1$ when their depths increase. In particular, as the depth of the ResNet increases, the original ResNet with $h = 1.0$ experiences a performance degradation while our ResNet with $h = 0.1$ has an increasing or stabilizing performance. It is also be observed that the blue shaded area (i.e., $h = 0.1$) is much thinner than the red one (i.e., $h = 1$). This shows that our ResNet with $h = 0.1$ has a smaller variance of the test accuracy over different random seeds. This demonstrates that a small h offers training robustness as well.

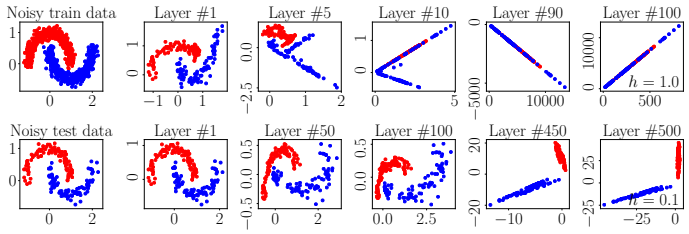

Figure 4: Illustration of feature transformations through ResNets with step factors $h = 1$ (top) and $h = 0.1$ (bottom).

To summarize, as proven in Section 3.2, a reduced h can prevent the explosion of backpropagated information over a large depth, thus making the training procedure of ResNet more robust to an increasing depth.

In terms of training robustness, a ResNet with a small h is also robust to larger learning rates and different types of optimizer. In fact, even without BN, a small h still helps to stabilize the training procedure while the performance of the original ResNet (i.e., $h = 1.0$) degrades significantly. All these results are reported in [Zhang *et al.*, 2019a].

4.2 Small h Improves Generalization Robustness

Synthetic Data for Mitigating Adverse Effect of Noise

To give insights on why a small h can improve the generalization robustness of ResNet, let us first consider a synthetic data example of using ResNet for a binary classification task, that is, by separating noisy "red" and "blue" points in a 2D plane. We train a vanilla ResNet (without BN) with $h = 1$ and a ResNet with $h = 0.1$ on the training data in the top left of Fig. 4 and perform feature transformations on the test data in the bottom left of Fig. 4 using the learned ResNets.

The series of figures at the top of Fig. 4 illustrate that the features are transformed through forward propagation in the vanilla ResNet (i.e., no BN, $h = 1$), which shows that the noise in the input features leads to the mixing of red and blue points, hence sabotaging the generalization capability of ResNet. The reason for this phenomenon is that with a large h, the features experience violent transformations between consecutive blocks due to larger weights and the adverse effect of noise is amplified over a large depth.

On the other hand, with a small h, the features experience smooth transformations at every consecutive residual block and the adverse effect of the noise is thus gradually mitigated, which entails correct classifications (see the series of figures at the bottom of Fig. 4). As mentioned in Section 3.3, a small h can help to mitigate the adverse effect of noise in the input features. With a small h, the noise amplification is effectively limited over a large depth.

Real-world Data for Mitigating Adverse Effect of Noise

To verify the effectiveness of a small h on the generalization robustness of ResNet, we train on noisy input data with varying noise levels; illustrations of noisy input are detailed in [Zhang *et al.*, 2019a]. We provide the test accuracy of ResNets with depths 218 and 49 on the clean input data of

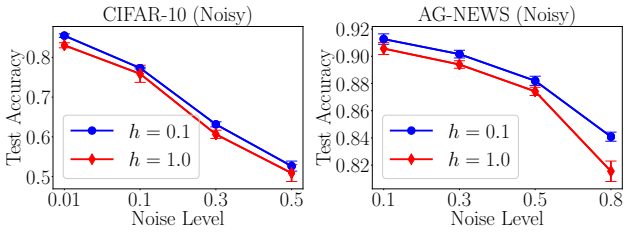

Figure 5: Effect of $h = 1$ and $h = 0.1$ on generalization robustness of ResNet.

CIFAR-10 and AG-NEWS datasets, respectively. For the AG-NEWS dataset, LR is fixed to 0.01. We compare test accuracy of the ResNet with reduced step factor $h = 0.1$ and the original ResNet (i.e., $h = 1.0$); other hyperparameters remain the same. Each configuration has 5 trials with different random seeds. The standard deviation is plotted as an error bar.

Fig. 5 shows that at different noise levels, our ResNet with reduced $h = 0.1$ consistently outperforms the original ResNet (i.e., $h = 1$). It can also be observed that our ResNet with reduced $h = 0.1$ has a smaller variance than its counterpart under different noise levels. Hence, our ResNet with $h = 0.1$ is robust to training on noisy input by mitigating the adverse effect of noise. By making smooth transformations, it gradually mitigates the adverse effect of noise in the input features along the forward propagation of ResNet. So, our ResNet with $h = 0.1$ offers better generalization robustness.

4.3 How to Select Step Factor h

We perform a grid search of h from 0.001 to 20 for the CIFAR-10 dataset and from 0.001 to 1.0 for the AG-NEWS dataset to optimize h. We train ResNets over different step factors h. We provide the median test accuracy with each configuration having 5 trials with different random seeds. The standard deviation is plotted as an error bar.

Fig. 6 shows that the optimal h is near 0.1 for ResNet-110 and ResNet-218 and in $[0.1, 2]$ for ResNet-20 for the CIFAR-10 dataset. For the AG-NEWS dataset, the optimal h is near 0.3 for ResNet-49 and 0.5 for ResNet-17.

When h is small, the error bar is small as well (i.e., performance variance is small). In comparison, when h is very large (e.g., $h \geq 5$ for ResNet-110 for the CIFAR-10 dataset), training becomes unstable and often fails. In ResNet-17 and ResNet-49 for the AG-NEWS dataset, when $h \geq 0.8$, 3 out of 5 training procedures fail in our experiments.

In addition, we observe that when h is very small (e.g., $h = 0.001$), generalization performance degrades even though the training variance is very small. This confirms our claim in Section 3.3 that an overly small h would smooth out the feature transformations.

To summarize, the guideline for the selection of h is that it should be small but not too small. Our experiments reveal that for a deep ResNet, h should be smaller (e.g., $h = 0.1$ for ResNet-218 for CIFAR-10 dataset) while for a shallow ResNet, a larger h (e.g., $h = 2.0$ for ResNet-20 for CIFAR-10 dataset) can be tolerated. In addition, once the depth of ResNet is fixed, the chosen h should not be too small (e.g., $h = 0.001$ and 0.01) so as not to smooth out the feature transformations.

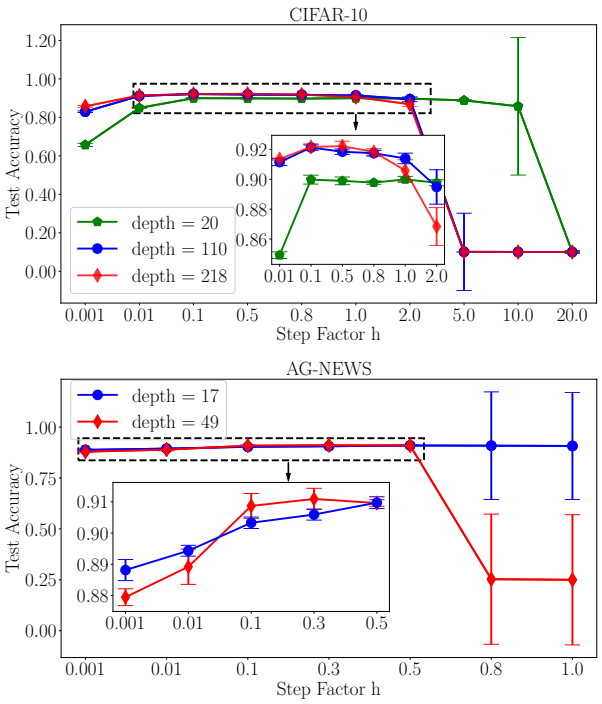

Figure 6: Selection of step factor h for ResNet with varying depths.

5 Conclusion

This paper describes a simple yet principled approach to boost the robustness of ResNet. Motivated by the dynamical system perspective, we characterize a ResNet using an explicit Euler method. This consequently allows us to exploit the step factor h in the Euler method to control the robustness of ResNet in both its training and generalization. We prove that a small step factor h can benefit its training and generalization robustness during backpropagation and forward propagation, respectively. Empirical evaluation on real-world datasets corroborates our analytical findings that a small h can indeed improve both its training and generalization robustness. For future work, we plan to explore several promising directions: (a) how to transfer the experience of a small h to other network structures (e.g., RNN for natural language processing [Cho *et al.*, 2014]), (b) how to handle the noisy target output labels [Han *et al.*, 2018], and (c) other means to choose the step size h (e.g., using Bayesian optimization [Dai *et al.*, 2019; Daxberger and Low, 2017; Hoang *et al.*, 2018; Ling *et al.*, 2016; Zhang *et al.*, 2017; Zhang *et al.*, 2019b]).

Acknowledgements

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative, RIKEN-AIP, and IBM Singapore.

References

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

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

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

[Chen *et al.*, 2018] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Proc. NeurIPS*, pages 6572–6583, 2018.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*, 2014.

[Conneau *et al.*, 2017] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proc. EACL*, 2017.

[Dai *et al.*, 2019] Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML*, pages 1496–1506, 2019.

[Daxberger and Low, 2017] Erik A. Daxberger and Bryan Kian Hsiang Low. Distributed batch Gaussian process optimization. In *Proc. ICML*, pages 951–960, 2017.

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

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

[Han *et al.*, 2018] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proc. NeurIPS*, pages 8536–8546, 2018.

[He *et al.*, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, pages 630–645, 2016.

[Hoang *et al.*, 2018] Trong Nghia Hoang, Quang Minh Hoang, Ruofei Ouyang, and Kian Hsiang Low. Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proc. AAAI*, pages 3231–3238, 2018.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *Proc. IEEE CVPR*, pages 2261–2269, 2017.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[Ketkar, 2017] Nikhil Ketkar. *Deep Learning with Python: A Hands-on Introduction*. Apress, 2017.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey E. Hinton. Learning multiple layers of features from tiny images. Master's thesis, Univ. Toronto, 2009.

[Lei Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. arXiv:1607.06450, 2016.

[Ling *et al.*, 2016] Chun Kai Ling, Kian Hsiang Low, and Patrick Jaillet. Gaussian process planning with Lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. In *Proc. AAAI*, 2016.

[Lu *et al.*, 2018] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proc. ICML*, pages 3282–3291, 2018.

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

[Reed and Marks II, 1999] Russell Reed and Robert J. Marks II. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999.

[Ruthotto and Haber, 2018] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. arXiv:1804.04272, 2018.

[Santurkar *et al.*, 2018] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Proc. NeurIPS*, pages 2488–2498, 2018.

[Weinan, 2017] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[Wu and He, 2018] Yuxin Wu and Kaiming He. Group normalization. In *Proc. ECCV*, pages 3–19, 2018.

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

[Zhang *et al.*, 2017] Yehong Zhang, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Mohan Kankanhalli. Information-based multi-fidelity Bayesian optimization. In *NIPS Workshop on Bayesian Optimization*, 2017.

[Zhang *et al.*, 2019a] Jingfeng Zhang, Bo Han, Laura Wynter, Kian Hsiang Low, and Mohan Kankanhalli. Towards robust ResNet: A small step but a giant leap. arXiv:1902.10887, 2019.

[Zhang *et al.*, 2019b] Yehong Zhang, Zhongxiang Dai, and Bryan Kian Hsiang Low. Bayesian optimization with binary auxiliary information. In *Proc. UAI*, 2019.