Zero Stability Well Predicts Performance of Convolutional Neural Networks

Liangming Chen[1, 2, 3], **Long Jin**[1, 3*], **Mingsheng Shang**[1]

[1] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
[2] Chongqing School, University of Chinese Academy of Sciences
[3] School of Information Science and Engineering, Lanzhou University
{lmchen, jinlongsysu}@foxmail.com, msshang@cigit.ac.cn

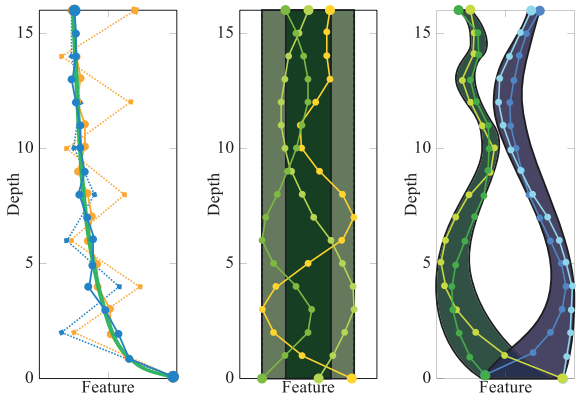
Abstract

The question of what kind of convolutional neural network (CNN) structure performs well is fascinating. In this work, we move toward the answer with one more step by connecting zero stability and model performance. Specifically, we found that if a discrete solver of an ordinary differential equation is zero stable, the CNN corresponding to that solver performs well. We first give the interpretation of zero stability in the context of deep learning and then investigate the performance of existing first- and second-order CNNs under different zero-stable circumstances. Based on the preliminary observation, we provide a higher-order discretization to construct CNNs and then propose a zero-stable network (ZeroSNet). To guarantee zero stability of the ZeroSNet, we first deduce a structure that meets consistency conditions and then give a zero stable region of a training-free parameter. By analyzing the roots of a characteristic equation, we theoretically obtain the optimal coefficients of feature maps. Empirically, we present our results from three aspects: We provide extensive empirical evidence of different depth on different datasets to show that the moduli of the characteristic equation's roots are the keys for the performance of CNNs that require historical features; Our experiments show that ZeroSNet outperforms existing CNNs which is based on high-order discretization; ZeroSNets show better robustness against noises on the input. The source code is available at https://github.com/logichen/ZeroSNet.

Introduction

The structure of a convolutional neural network (CNN) significantly affects its performance (He et al. 2016; Xie et al. 2019; Liu et al. 2021). However, there is no clear clue about determining the importance of historical features and current activations (e.g., a sequence consisting of ReLU, convolutional layer, and batch normalization layer). A promising direction for structure determination is the ordinary-differential-equation-inspired design (Lu et al. 2018; Zhu, Chang, and Fu 2018). We seek the answer from the perspective of zero stability which is a concept originating from numerical analysis.

There are several types of stabilities in different fields. We provide Fig. 1 to illustrate three of them: Absolute stability (A-stability), bounded input bounded output (BIBO) stability, and zero stability. In the following content, we discuss stability from a CNN structure design perspective. As shown in Fig. 1(a), absolute stability (A-stability) means that the numerical solution approaches the exact solution, regardless of the step size, as $t \to \infty$ (Atkinson, Han, and Stewart 2011; Luo et al. 2021). For CNNs, A-stability means that for a network with a given depth, the feature map magnitude does not increase from the input to the output (Haber et al. 2019). BIBO stability ensures that the output feature map is within a bound if the input is bounded (Zhang and Schaeffer 2020), as in Fig. 1(b). Although these stabilities benefit the learning ability and prevent the model from collapsing (Jin,



(a) A-stability (b) BIBO stability (c) Zero stability

Figure 1: Illustrations of A-stability, BIBO stability, and zero stability. (a) Blue lines denote an A-stable method: Regardless of the step size, the method approaches the exact solution (the solid green curve). Orange lines represent a non-A-stable method, which can only approach the exact solution if the step size is small. Note that dotted lines have a large step size. (b) The light shade represents the bound of the input; The dark shade represents the bound of the output. (c) The shades represent possible ranges of the difference magnitude between features with different initial values. It means that similar inputs generate similar outputs. In this work, we focus on zero stability and connect it with robustness and generalization.

*Corresponding Author

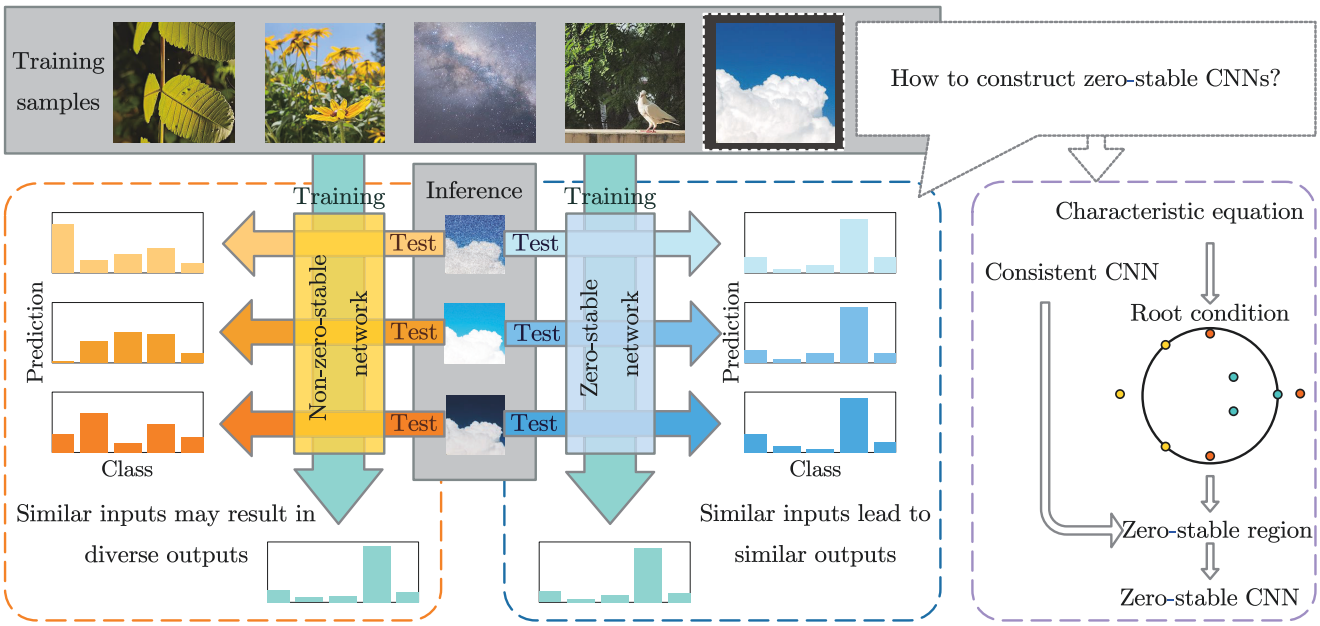


Figure 2: Left: Illustrations of the connection between zero stability and generalization/robustness. Right: Our approach to construct a zero-stable CNN, namely ZeroSNet.

Wei, and Li 2022; Luo et al. 2022), they give no clear clues about the CNNs' generalization and robustness.

We show two meanings of zero stability in the context of numerical analysis and connect them with the CNNs' generalization and robustness, respectively. As shown in Fig. 1(c), the first meaning of zero stability is that for two similar initial values, the corresponding states at time t are also similar (Gautschi 1997). We borrow this idea to analyze the generalization of CNN: For a well-trained CNN, zero stability means that if the test sample is slightly different from the training one, the feature map changes slightly compared with that of the training sample, and then the network output also changes slightly. Since this CNN is well-trained as aforementioned, a correct prediction for the test sample should be given. Another meaning of zero stability is that if an initial value is perturbed, the fluctuation of the output is bounded (Atkinson, Han, and Stewart 2011), as in Fig. 1(c). We add the noise on the input feature as the perturbation on the initial value, thereby building a bridge between zero stability and the robustness of CNNs.

To illustrate the insight of zero stability in the context of CNN, we give an example on the left side of Fig. 2. Both zero-stable and non-zero-stable CNNs classify the cloud image correctly. When similar samples are fed into the two CNNs, the non-zero-stable CNN gives diverse predictions, while the zero-stable one gives similar predictions and thus succeeds in this task. To achieve zero stability, we propose a zero-stable network (ZeroSNet) with a general form to ensure consistency (which tightens the upper bound of zero stability) of the ZeroSNet. Based on the characteristic equation of the ZeroSNet, we apply the root condition and then obtain a zero-stable region for a flexible coefficient. The right side of Fig. 2 describes the process of constructing the ZeroSNet.

Our contributions in this paper are summarized as follows:

- We are the first to find that zero stability well predicts the model performance (we provide preliminary observations and more convincing evidence). Based on the finding, we provide corresponding explanatory analyses.
- We propose a CNN named ZeroSNet with theoretical proofs on its consistency and give a stability region of a training-free parameter. Besides, we deduce optimal coefficients for historical features and the current activations.
- ZeroSNet with theoretically optimal coefficients achieves advanced performance and outperforms the existing high-order-discretization CNNs.
- Our experiments show that involved zero-stable CNNs are robust against noises that are injected on the input, while non-zero-stable ones reveal a dramatical degradation.

Preliminaries

We slightly extend the initial value problem (Atkinson, Han, and Stewart 2011; Chen et al. 2018) and get an initial values problem with more initial steps.

Definition 1 (Initial values problem)**.** *An initial values problem is defined as*

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{y}(t)}{\mathrm{d}t} &= \boldsymbol{f}(t, \boldsymbol{y}(t)), \quad s \le t \le e, \\ \boldsymbol{y}(s+qh) &= \boldsymbol{y}_{s+q}, \quad q = 0, 1, \ldots, d, \end{aligned} \tag{1}$$

where $\boldsymbol{y}(t) \in \mathbb{R}^p$ is a p-dimensional feature vector; s and e are the start and the end of the time t, respectively; h is

the step size with q denoting the qth step; $\boldsymbol{y}_{s+q-1}$ are given initial states.

In the context of the deep learning, $\boldsymbol{y}_s$ can be seen as the input of a neural network, and the training process determines the optimal $\boldsymbol{f}(t, \boldsymbol{y}(t))$ (Lu et al. 2018). To discretize (1), the definition of the dth-order discretization is given as follow.

Definition 2 (dth-order discretization)**.** *A dth-order discretization for an initial value problem is defined as*

$$\begin{aligned}\boldsymbol{y}(t_{n+1}) =&\alpha_0\boldsymbol{y}_n + \alpha_1\boldsymbol{y}(t_{n-1}) + \ldots + \alpha_{d-1}\boldsymbol{y}(t_{n-d+1}) \\ &+ h\beta\boldsymbol{f}(t_n, \boldsymbol{y}(t_n)), \\ &n = 0, 1, \ldots, \quad d = 1, 2, \ldots,\end{aligned} \tag{2}$$

where α_0, α_1, ..., α_{d-1}, and β are given coefficients; $t_n = s + nh$.

We can interpret α_0, α_1, ..., α_{d-1} and β as the weight of each historical featuremap and the current activations, respectively. If $d = 1$ and $\alpha_0 = \beta = 1$, equation (2) reveals the Euler discretization, and it gives the pre-activation ResNet (PreResNet) (He et al. 2016). Moreover, equation (2) can be regarded as a special case of the multilstep method. For $d = 2$, if $\alpha_0 = 1 - k_n$, $\alpha_1 = k_n$, and $\beta = 1$, one gets the linear multilstep (LM) architecture (Lu et al. 2018). In this work, we build higher-order-discretization-based CNNs, of which zero stability and consistency are guaranteed.

Assumption 1 (Lipschitz continuous sequence after normalization)**.** *Consider an $\boldsymbol{f}$ which consists of a sequence of layers (e.g., ReLU and convolutional layers) and a normalization layer in the end, and $\boldsymbol{f}$ is Lipschitz continuous. That is, for two arbitrary $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^p$,*

$$\|\boldsymbol{f}(t, \boldsymbol{y}) - \boldsymbol{f}(t, \hat{\boldsymbol{y}})\| \leq \ell \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|, t \in [s, e], \tag{3}$$

where ℓ is the Lipschitz constant; $\|\cdot\|$ denotes the 2-norm of a vector.

Usually, once the normalization layer (e.g., batch normalization, layer normalization) is involed as the last layer in $\boldsymbol{f}$, condition (3) is meet for CNNs. This is because no matter how large the original feature values are, after a normalization layer, these values are forced to follow a controled distribution. An example of such a sequence in $\boldsymbol{f}$ is $\boldsymbol{f}'(t, \boldsymbol{y}) = \boldsymbol{n}_2(ReLU(\boldsymbol{\theta}_2 \star \boldsymbol{n}_1(ReLU(\boldsymbol{\theta}_1 \star \boldsymbol{y}))))$, where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are both batch normalization layers; symbol $\star$ denotes the convolution operator.

Definition 3 (Zero stablility (Gautschi 1997))**.** *For two grid functions $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ on $[a, b]$, a dth-order discretization is zero-stable if the following inequality holds for a sufficient-small step size h:*

$$\|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\|_\infty \leq c(\max_{m\in[0,d-1]} \|\boldsymbol{y}_m - \hat{\boldsymbol{y}}_m\| + \|\boldsymbol{r}(\boldsymbol{y}_n) - \boldsymbol{r}(\hat{\boldsymbol{y}}_n)\|_\infty), \tag{4}$$

where $\boldsymbol{r}(\boldsymbol{y}_n) := 1/h \sum_{i=0}^{d-1} \alpha_i \boldsymbol{y}_{n+i} - \beta \boldsymbol{f}(t_n, \boldsymbol{y}(t_n))$; $\boldsymbol{r}(\hat{\boldsymbol{y}}_n) := 1/h \sum_{i=0}^{d-1} \alpha_i \hat{\boldsymbol{y}}_{n+i} - \beta \boldsymbol{f}(t_n, \hat{\boldsymbol{y}}(t_n))$; c is a constant; $\|\cdot\|_\infty$ is the infinity norm.

Definition 4 (Consistency (Atkinson, Han, and Stewart 2011))**.** *For an exact solution $\boldsymbol{y}(t)$, a dth-order discretization is consistent if*

$$\max_{t_n\in[t_{d-1},e]} \left\| \boldsymbol{y}(t_{n+1}) - \left(\sum_{i=0}^{d-1} \alpha_i \boldsymbol{y}_{n-i} + h\beta \boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) \right) \right\| / h \to 0 \text{ as } h \to 0. \tag{5}$$

If $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are two solutions with different initial values, $\boldsymbol{r}(\boldsymbol{y}_n)$ and $\boldsymbol{r}(\hat{\boldsymbol{y}}_n)$ are truncation errors exactly (Gautschi 1997). If the dth-order discretization (2) is consistent and the step size h is sufficient-small , we have $\lim_{n\to\infty} \boldsymbol{r}(\boldsymbol{y}_n) \to \boldsymbol{0}$ and $\lim_{n\to\infty} \boldsymbol{r}(\hat{\boldsymbol{y}}_n) \to \boldsymbol{0}$ (Gautschi 1997). Under the consistency condition (5), it follows that

$$\|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\|_\infty \leq c \left(\max_{m\in[0,d-1]} \|\boldsymbol{y}_m - \hat{\boldsymbol{y}}_m\| \right). \tag{6}$$

Criterion for Zero Stability

Root condition, a well-known criterion for zero stability, is given here and is further as a practical tool to verify zero stability and predict the performance of CNNs later.

Condition 1 (Root Condition (Ascher and Petzold 1998))**.** *The root condition means that the roots of a characteristic equation $r(\rho) = \rho^d - \sum_{i=0}^{d-1} \alpha_i \rho^{d-1-i}$ satisfy $|\rho_i| \leq 1$, and if $|\rho_i| = 1$ then ρ_i is a simple root, where $|\cdot|$ denoting to take the modulus of a complex number.*

The empirical observations in the next section show that some existing CNNs can be interpreted as first- and second-order discretizations. After that, we construct a higher-order CNN to further verify the relationship between model performance and zero stability.

Observations from Existing CNNs

Involving historical feature maps benefits the CNNs' representation ability (Huang et al. 2017). Meanwhile, a visualization study suggests that historical features may smooth the loss landscape. However, the importance of each historical feature for the CNNs' performance remains unclear: We adjust the coefficients (weights) of historical feature maps and current activations in the provided preparatory experiments.

An Observation from PreResNet

As discussed earlier, PreResNet can be deemed as an Euler discretization. Extending the Euler discretization slightly by involving a flexible coefficient α for the current feature $\boldsymbol{y}_n$ gives

$$\boldsymbol{y}_{n+1} = \alpha \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n). \tag{7}$$

Let us see what happens if we change the value of α from Table 1. According to (Gautschi 1997), we can check the stability quickly. As shown in Table 1, there is a significant gap in the test accuracy between zero-stable and non-zero-stable models. Besides, the original PreResNet ($\alpha = 1$) outperforms other models with the same structures but different coefficients α.

Model	α	Z. S.	Test acc. (%)
PreResNet-32	2	No	79.13±0.30
PreResNet-32	1.5	No	87.07±0.14
PreResNet-32	0.5	Yes	92.52±0.42
PreResNet-32	0.7	Yes	93.16±0.13
PreResNet-32	1	Yes	**93.19**±0.17

Table 1: Test accuracies (mean ± standard deviation) obtained by setting different feature weight α on CIFAR-10 dataset. "Z. S." and "acc." denote zero stability and the accuracy, respectively. Once the zero stability region is exceeded, the performance shows a clear degradation.

Model	k	Z. S.	Test acc. (%)
LM-ResNet-44	-1.5	No	81.43±0.19
LM-ResNet-44	1.5	No	89.46±0.30
LM-ResNet-44	-0.5	Yes	92.95±0.24
LM-ResNet-44	0.5	Yes	**93.69**±0.21

Table 2: Test accuracies (mean ± standard deviation) obtained by setting different k on CIFAR-10 dataset. "Z. S." and "acc." denote zero stability and the accuracy, respectively. Similar with the first-order discretization, the second-order discretization's performance can also be predicted by zero stability.

We are still not sure whether the phenomenon is caused by the forward propagation or the backward propagation (from the backward propagation perspective, PreResNet may also benefit from the residual connection when applying the chain rule, as discussed in Section 3 of (He et al. 2016)). We consider a second-order situation in the following subsection.

An Observation from LM-Architecture

The LM-architecture in (Lu et al. 2018) can be seen as a second-order discretization. We make a modification on β with a sharing k for all layers to ensure consistency and then obtain

$$\boldsymbol{y}_{n+1} = (1-k)\boldsymbol{y}_n + k\boldsymbol{y}_{n-1} + (2k+1)h\boldsymbol{f}(t_n, \boldsymbol{y}). \quad (8)$$

The characteristic equation of equation (8) is

$$r(\rho) = \rho^2 + (k-1)\rho - k. \quad (9)$$

We set several k and check zero stability by applying the root condition (Atkinson, Han, and Stewart 2011) for equation (9), and then obtain Table 2. The detailed experiment settings are described in Experiment section.

Zero Stability for CNN

For CNNs, the meaning of equation (6) is as follows. First, the backpropagation determines $\boldsymbol{f}$. When the network training process is done, we obtain $\boldsymbol{f}$, which fits the training data. We use $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$ to represent the inputs from the training set and the test set, respectively. From equation (6), if the inputs $\boldsymbol{y}_0$ and $\hat{\boldsymbol{y}}_0$ are similar, the predictions $\boldsymbol{y}_n$ and $\hat{\boldsymbol{y}}_n$ are similar, too. Assume that for a well-trained network, feeding $\boldsymbol{y}_0$ to the equation (2) gives the correct prediction $\boldsymbol{y}_n$. Then, we could conclude that for a test sample, which is similar to one of the training samples, the prediction result is close to the correct answer. It means that the zero-stable neural networks are robust against perturbations, and its generalization is well for value-based differences.

To further investigate whether zero stability well predicts the model performance, we deduce a consistent and zero-stable model named ZeroSNet in the next section and practically compare it with its non-zero-stable counterparts in the Experiment section.

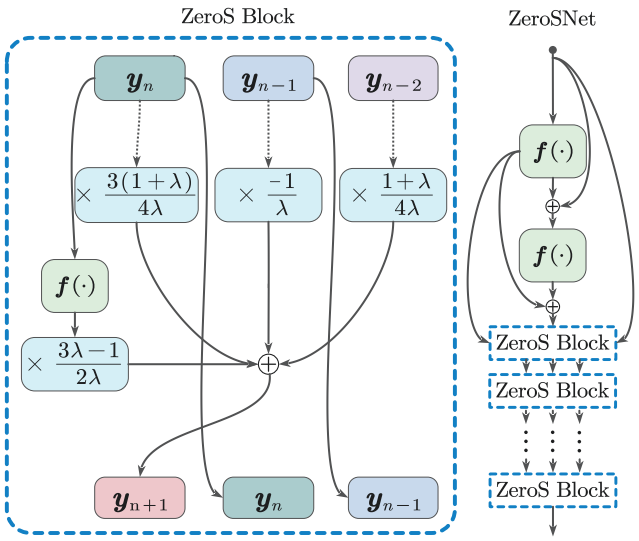


Figure 3: An overall structure of the ZeroSNet. Left: Two types of blocks with given coefficients that guarantee consistency and zero stability. The coefficients are given by equation (10) and are training-free. Note that each block takes three inputs and gives three outputs. The dotted arrow means performing downsampling if dimensions are mismatched. Left: A zero-stable block (ZeroS block). Right: Following the two basic start blocks, we stack ZeroS blocks to build the deep structure. $\boldsymbol{f}(\cdot)$ in the basic ZeroS block consists of two BN-ReLU-convolution triplets, and $\boldsymbol{f}(\cdot)$ in the ZeroS bottleneck block consists of three BN-ReLU-convolution triplets.

Zero-Stable Network (ZeroSNet)

In this section, we deduce a CNN named Zero-Stable Network (ZeroSNet), which is automatically consistent, and we give the range of a flexible and training-free parameter to ensure zero stability.

Description of ZeroSNet

In (Li, Zhang, and Mao 2019), a numerical method named general square-pattern discretization is presented. With the aid of this numerical method, we construct the ZeroSNet. Due to the space limitation, a mathematical derivation of the ZeroSNet is given in the Appendix. Directly, we give the

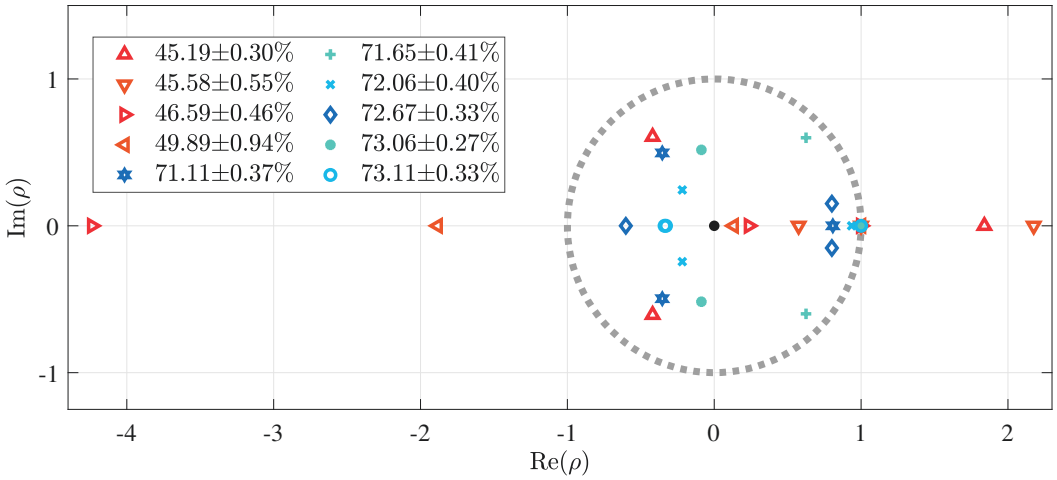


Figure 4: Test accuracies (mean ± standard deviation) of third-order-discretization-based models with different roots. The models all contains 56 layers and are evaluated on CIFAR-100 dataset. It is clear that models which satisfy the root condition (Condition 1) outperform the ones with large roots.

formal description of the ZeroSNet here:

$$\begin{aligned}\boldsymbol{y}_{n+1} =& \frac{3(1+\lambda)}{4\lambda}\boldsymbol{y}_n - \frac{1}{\lambda}\boldsymbol{y}_{n-1} + \frac{1+\lambda}{4\lambda}\boldsymbol{y}_{n-2} \\ &+ \frac{3\lambda-1}{2\lambda}h\boldsymbol{f}(t_n, \boldsymbol{y}_n),\end{aligned} \tag{10}$$

where $\lambda \neq 0$ is a scalar.

To illustrate the structure of ZeroSNet, we provide Fig. 3. We borrow ideas from (He et al. 2016) and (Zhang et al. 2017): Building a deep network by stacking blocks iteratively.

Note that each block takes three inputs and gives three outputs. In Fig. 3, the dotted arrow means to perform a downsampling if dimensions of $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}$, and $\boldsymbol{y}_{n-2}$ are mismatched. Three arrows pointing to the "ZeroS Block-/Bottlenecks" module are respectively $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}$, and $\boldsymbol{y}_{n-2}$; Arrows starting from the "ZeroS Block/Bottlenecks" module are respectively $\boldsymbol{y}_{n+1}, \boldsymbol{y}_n$, and $\boldsymbol{y}_{n-1}$. As will be discussed, Blocks in Fig. 3 ensure consistency according to the following Theorem 1. Moreover, if we follow Theorem 2 to set λ, the neural network is zero-stable. Compared to the PreResNet, the additional parameters of ZeroSNet are only for downsampling. We provide a comparison of the number of parameters in the Experiment section. By setting λ_n as a trainable parameter for the nth block ($n \geq 2$), we obtain a trainable version of the ZeroSNet, and we call it ZeroSNet-Tra. Theorems 1 and 2, as well as other properties of the ZeroSNet, will be introduced in the next section.

Properties of ZeroSNet

Compared to the PreResNet and the LM-ResNet, ZeroSNet involves more historical information and smoothly transmits the low-level features. However, (Gautschi 1997) suggests that high-order discretizations are easily to be non-zero-stable. As shown in equation (6), consistency removes the $\|\boldsymbol{r}(\boldsymbol{y}_n) - \boldsymbol{r}(\hat{\boldsymbol{y}}_n)\|$ term in equation (4) and thus gives a tighter upper bound of zero stability. Based on equation (6), zero stability has an ability to predict the model performance. In this part, we give Theorem 1 to ensure consistency of the ZeroSNet and then provide a zero stability region of parameter λ.

Theorem 1 (ZeroSNet (10) is consistent)**.** *Suppose that $\boldsymbol{y}(t)$ is continuously differentiable, ZeroSNet* (10) *meets the consistency condition.*

Proofs are deferred to the Appendix. By using the root condition (Atkinson, Han, and Stewart 2011), we investigate zero stability of ZeroSNet (10).

Theorem 2 (Zero stability region of the ZeroSNet (10))**.** *For a continuously differentiable function $\boldsymbol{y}(t)$, if $\lambda \in (-\infty, -1) \cup (1/3, +\infty)$, the ZeroSNet* (10) *is zero-stable.*

Proofs are deferred to the Appendix. Based on Theorem 2, we show optimal coefficients of historical features and the current activations $\boldsymbol{f}$.

Theorem 3 (Optimal coefficients of the ZeroSNet (10))**.** *From the perspective of zero stability, optimal coefficients of $\boldsymbol{y}_n, \boldsymbol{y}_{n-1}, \boldsymbol{y}_{n-2}$, and $\boldsymbol{f}(t_n, \boldsymbol{y}_n)$ in the ZeroSNet* (10) *are $1/3, 5/9, 1/9$, and $16/9$, respectively.*

Proofs are deferred to the Appendix. In addition to theoretical results on zero stability and optimal coefficients, we conduct experiments to verify whether zero stability well predicts CNNs' performance and whether the theoretically optimal coefficients work well in practice.

Experiments

In this section, we conduct extensive experiments to verify if zero stability well predicts performance with the aid of 3rd-order-discretization-based CNNs, i.e., ZeroSNets with zero stability and others without zero stability. Besides, we build a trainable version of ZeroSNet for the comparison on several benchmarks. In addition, we add different types of

# Layers	Roots' moduli	Z. S.	Test acc. (%)
32	0.57, 1.00, 2.18	No	77.55±0.15
32	1.84, 0.74, 0.74	No	78.44±0.67
32	4.24, 0.24, 1.00	No	79.67±0.49
32	1.88, 0.13, 1.00	No	88.77±0.24
32	1.00, 0.87, 0.87	Yes	89.44±0.08
32	0.94, 0.33, 0.33	Yes	92.59±0.15
32	0.81, 0.61, 0.61	Yes	92.84±0.10
32	1.00, 0.52, 0.52	Yes	93.14±0.07
32	0.60, 0.82, 0.82	Yes	93.27±0.04
32	0.33, 0.33, 1.00	Yes	**93.28**±0.12
44	0.57, 1.00, 2.18	No	77.91±0.62
44	1.84, 0.74, 0.74	No	78.70±0.43
44	4.24, 0.24, 1.00	No	79.95±0.23
44	1.88, 0.13, 1.00	No	83.08±0.58
44	1.00, 0.87, 0.87	Yes	92.70±0.17
44	0.94, 0.33, 0.33	Yes	93.06±0.20
44	0.81, 0.61, 0.61	Yes	93.10±0.11
44	0.60, 0.82, 0.82	Yes	93.68±0.07
44	0.33, 0.33, 1.00	Yes	93.68±0.15
44	1.00, 0.52, 0.52	Yes	**93.72**±0.10
56	1.84, 0.74, 0.74	No	78.29±0.09
56	0.57, 1.00, 2.18	No	78.34±0.30
56	4.24, 0.24, 1.00	No	79.39±0.17
56	1.88, 0.13, 1.00	No	83.11±0.13
56	1.00, 0.87, 0.87	Yes	92.42±0.35
56	0.81, 0.61, 0.61	Yes	93.08±0.36
56	0.94, 0.33, 0.33	Yes	93.15±0.16
56	1.00, 0.52, 0.52	Yes	93.71±0.24
56	0.60, 0.82, 0.82	Yes	93.99±0.06
56	0.33, 0.33, 1.00	Yes	**94.04**±0.12

Table 3: Test accuracies (mean ± standard deviation) obtained by setting different coefficients (α_0, α_1, α_2, and β) on CIFAR-10 dataset. These coefficients give different moduli of roots. "# Layers", "Z. S.", and "acc." denote the number of layers, zero stability, and the accuracy, respectively. Similar with first- and second-order discretizations, the third-order discretization's performance can also be well predicted by zero stability.

noise to the images and observe the relationship between robustness and zero stability. Note that hyperparameters for CIFAR-10 and CIAFR-100 are the same as those in (Lu et al. 2018).

Predicting Performance by Zero Stability

In early parts of this aper, preliminary experiments imply that zero stability well predicts the model performance. To further verify this conjecture, we use many 3rd-order-discretization-based CNNs for evaluations. We carefully choose coefficients $\alpha_0, \alpha_1, \alpha_2$, and β to include more root patterns (see the Table 10 for the mapping of those coefficients and the moduli of roots). Then, we provide Table 3 to show the results for 32- to 56-layer models on CIFAR-10. It is clear that if the roots satisfy the root condition (i.e., the model is zero-stable), the model performs well; If the model is non-zero-stable, its performance is relatively poor. Figure 4 shows results of 56-layer models on CIFAR-100, and the performance gap between the zero-stable and non-zero-stable models is significant. In Fig. 4, the root condition (Condition 1) well predicts the model performance. Optimal coefficients given by Theorem 3 leads to a group of moduli of roots being $0.33, 0.33, 1.00$; These optimal coefficients are denoted in cyan hollow circles in Fig. 4. Combining Table 3 and Fig. 4, we find that the optimal coefficients given by Theorem 3 indeed outperform other coefficients in most cases. Empirically, zero stability well predicts model performance on different datasets with different discretization orders.

Model	# Layer	C100 (%)	C10 (%)
ResNet	20	69.46±0.15*	91.25†
LM-ResNet	20	69.32±0.33	91.67†
ZeroSNet-Opt	20	69.88±0.21	92.01±0.35
ZeroSNet-Tra	20	**69.90**±0.26	**92.32**±0.10
ResNet	32	71.30±0.20*	92.49†
LM-ResNet	32	71.32±0.36	92.82†
ZeroSNet-Opt	32	**71.25**±0.31	**93.28**±0.12
ZeroSNet-Tra	32	71.09±0.21	93.07±0.14
ResNet	44	72.36±0.23*	92.83†
LM-ResNet	44	72.05±0.25	93.34†
ZeroSNet-Opt	44	**72.49**±0.30	93.68±0.15
ZeroSNet-Tra	44	72.18±0.38	**93.69**±0.21
ResNet	56	72.56±0.08*	93.03†
LM-ResNet	56	72.94±0.19	93.69†
ZeroSNet-Opt	56	**73.11**±0.33	**94.04**±0.12
ZeroSNet-Tra	56	72.72±0.13	93.8±0.22
ResNet	110	72.24*†	93.63*†
LM-ResNet	110	74.13†	93.84†
ZeroSNet-Opt	110	74.50±0.28	**94.35**±0.14
ZeroSNet-Tra	110	**74.56**±0.23	94.30±0.02

Table 4: Test accuracies (mean ± standard deviation) on CIFAR-10 and CIFAR-100 datasets. "# Layer" denotes the number of layers. Best results of each # Layers are bold. "†" indicates results obtained from (Lu et al. 2018); "*" indicates ResNet with the pre-activation (PreResNet).

Comparison Experiments

In this part, we compare the ZeroSNet with existing high-order-discretization CNNs (LM-ResNets) and PreResNets (Lu et al. 2018; He et al. 2016) on CIAFR-10 and CIFAR-100 datasets. In addition, comparisons on ImageNet are also performed. Although ZeorSNet outperforms existing high-order CNNs and PreResNets, our major goal is not to beat the state-of-the-art model. Thus, we do not involve additional tricks. We provide Table 4 to show the test performance of 20- to 110-layer models on CIAFR-10 and CIFAR-100 datasets. In addition, we use ZeroSNet-Opt to represent a ZeroSNet with optimal coefficients (i.e., Theorem 3) in this comparison. By setting λ_n as a trainable parameter for the nth block ($n = 2, 3, \ldots$), we have a trainable ZeroSNet,

Model	# Layer	Top-1 (%)	Top-5 (%)
PreResNet	18	69.66	88.94
ZeroSNet-Opt	18	**69.84**	**88.97**
PreResNet	34	72.21	90.68
ZeroSNet-Opt	34	**72.69**	**90.83**
PreResNet	50	74.51	91.91
ZeroSNet-Opt	50	**74.88**	**92.03**

Table 5: Accuracies (top-1 and top-5) on ImageNet validation set with single-crop testing. "# Layer" denotes the number of layers. We apply the mixed-precision training for all models on ImageNet.

namely ZeroSNet-Tra. Table 4 shows that ZeroSNets outperform LM-ResNets (Lu et al. 2018) and PreResNets. Compared to ZeroSNet-Tra with several trainable λ_n, ZeroSNet-Opt with one training-free λ shared for all blocks is competitive. In addition, Table 5 shows that the ZeroSNet-Opt has the advantage of top-1 and top-5 accuracies on ImageNet. Note that we use mixed-precision training for all models on ImageNet.

Robustness

We verify the robustness of models on the test set. We store network parameters after the noise-free training. Then, we unnormalize the input images into $[0, 1]$. After feeding these input images into the stored models, the accuracies under perturbations are obtained. Three types of noises are involved: Uniform noise, Gaussian noise, and constant noise. Each type of noise is added to input images with different levels. Table 6 shows the test performance of 56-layer models under these three types of noises. The uniform noises are in [lower bound, upper bound]; The Gaussian noises are generated with standard deviation δ and a mean of zero; The constant noise is a grey image with pixel values of μ. As in Table 6, the non-zero-stable models' test accuracies decrease dramatically after injecting noises, while zero-stable models are robust. For example, under uniform noises distributed in $[-0.08, 0]$, test accuracies of non-zero-stable models decrease 12.38% on average, while for zero-stable models, this degradation is only $\mathbf{6.40}\%$. Similar phenomenons are clear in other noise-model pairs. We provide more experiment results on noises with different levels and some results with adversarial examples (i.e., fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2014) on MNIST and projected gradient descent (PGD) (Madry et al. 2018) on CIFAR-10) in the Appendix.

Generalization Gap

In addition to performance on the test set, we provide the experimental results of the generalization gap for ZeroSNets in Table 7. To facilitate comparison, all involved ZeroSNets have a root as 1 and two repeated roots. From tables 7, we can see that smaller moduli of roots (which imply better zero stability) generally lead to a smaller generalization gap. In general, the optimal coefficients given by Theorem 3 lead to the best generalization gap. To achieve sufficient training, we train all models for 500 epochs for generalization gap experiments (this is different from all other experiments in this paper).

Computation Efficiency

From the Experiment section, we can see that there are performance improvements brought by ZeroSNets. In this part, we evaluate the costs of such improvements. A comparison of the number of parameters is given in Table 8. ZeroSNets have a close number of parameters compared with PreResNets. Besides, we provide the runtime of PreResNet20 and ZeroSNet20 on CIFAR-10 (Table 9). Table 9 shows that time consumption of ZeroSNet20 is close to PreResNet20, especially for large batch sizes. When we perform the runtime experiments, we remain only one task on a server.

Experiment Settings

We provide detailed experiment settings as follows. We use Pytorch 1.8.1 framework and run our experiments on a server with 10 RTX 2080 TI GPUs and 2 RTX 3090 GPUs.

CIFAR. Hyperparameters for CIFAR-10 and CIAFR-100 are the same as those in (Lu et al. 2018). We conduct all experiments with stochastic gradient descent (SGD) optimizer. On the CIFAR, we use a batch size of 128 with an initial learning rate of 0.1, the momentum of 0.9, and weight decay 0.0001. Models in generalization gap experiments (Table 7) are trained for 500 epochs to achieve sufficient training. Except for the generalization gap experiments, all models on CIFAR-10 and CIFAR-100 are trained for 160 and 300 epochs, respectively. We apply the step decay to train all models on CIFAR and divide the learning rate by 10 at half and three-quarters of the total epoch. We report the "mean $\pm$ standard deviations" accuracies based on three individual runs. For the trainable version of ZeroSNet (i.e., ZeroSNet-Tra), all λ_n are initialized as 1. The data augmentations are the random crop with a 4-pixel padding and random horizontal flip, as in (Lu et al. 2018).

ImageNet. Our training script is based on https://github.com/13952522076/Efficient_ImageNet_Classification and remains all default hyperparameters. To improve the training efficiency on ImageNet, we use a mix-precision strategy provided by NVIDIA apex with distributed training. We apply the cosine decay with a 5-epoch warmup to train models for 150 epochs. The weight decay and the momentum are 4×10^{-5} and 0.9, respectively. Following the adjustment guidance of the learning rate and the batch size (Goyal et al. 2017; Jastrzebski et al. 2018), we set them according to the GPU memory. Specifically, for 18-layer models, we use an initial learning rate of 0.2 and a batch size of 128; for 34-layer models, we use an initial learning rate of 0.1 and a batch size of 64; for 50-layer models, we use an initial learning rate of 0.05 and a batch size of 32. For ImageNet, we apply 8-GPU distributed training on a single server.

Robustness. The random seeds of PyTorch for generating the uniform and Gaussian noises are both 1. In the standard training phase, we store three individual models for each group of α_0, α_1, α_2, and β. Then, we use the three models to evaluate the average robustness and report the result in the "mean $\pm$ standard deviations" format. Finally, we map the

Roots' moduli	Z. S.	Noise-free	$[-0.08, 0]$	$[0, 0.08]$	$\delta = 0.01$	$\delta = 0.02$	$\delta = 0.04$	$\mu = 0.3$
1.84, 0.74, 0.74	No	78.29±0.09	67.01±1.08	67.14±1.32	77.17±0.28	70.65±0.78	50.76±3.94	66.82±1.36
0.57, 1.00, 2.18	No	78.34±0.30	66.16±1.81	66.99±1.67	76.88±0.48	69.75±1.38	48.72±2.82	67.81±0.22
4.24, 0.24, 1.00	No	79.39±0.17	67.06±1.83	67.06±2.02	77.41±0.66	70.19±1.84	51.23±0.39	69.31±0.97
1.88, 0.13, 1.00	No	83.11±0.13	69.38±0.40	69.16±0.51	81.25±0.22	73.23±0.37	50.2±0.96	73.88±0.96
1.00, 0.87, 0.87	Yes	92.42±0.35	83.89±1.55	83.98±0.81	91.21±0.43	86.73±0.94	64.69±2.35	86.75±0.77
0.81, 0.61, 0.61	Yes	93.08±0.36	86.91±0.31	86.92±0.29	92.00±0.28	88.66±0.30	74.19±0.39	87.43±0.48
0.94, 0.33, 0.33	Yes	93.15±0.16	88.05±0.23	**88.16**±0.29	92.24±0.18	89.51±0.19	**76.74**±0.13	87.73±0.32
1.00, 0.52, 0.52	Yes	93.71±0.24	87.10±0.38	87.23±0.27	92.47±0.30	88.84±0.29	73.96±0.47	**88.90**±0.35
0.60, 0.82, 0.82	Yes	93.99±0.06	87.76±0.31	87.84±0.42	92.75±0.14	89.46±0.29	74.97±0.84	88.82±0.29
0.33, 0.33, 1.00	Yes	**94.04**±0.12	**88.31**±0.33	87.64±0.43	**92.96**±0.12	**89.67**±0.29	74.49±1.25	88.78±0.29

Table 6: Test accuracies (mean ± standard deviation) on CIFAR-10 under uniform noise ([lower bound, upper bound]), zero-mean Gaussian noise (with standard deviation δ), and constant noise (with magnitude μ). Note the input imgages are normalized into an interval of $[0, 1]$. "Z. S." denotes zero stability.

Model	Roots' moduli	Training	Test	Gap
ZeroSNet44	1, 0.52, 0.52	99.31	72.25	27.06
ZeroSNet44	1, 0.87, 0.87	98.04	71.59	26.45
ZeroSNet44	1, 0.33, 0.33	99.34	72.91	**26.43**
ZeroSNet56	1, 0.87, 0.87	99.31	71.07	28.24
ZeroSNet56	1, 0.52, 0.52	99.67	72.55	27.12
ZeroSNet56	1, 0.33, 0.33	99.65	72.96	**26.69**
ZeroSNet110	1, 0.87, 0.87	99.90	73.62	26.28
ZeroSNet110	1, 0.52, 0.52	99.90	75.15	**24.75**
ZeroSNet110	1, 0.33, 0.33	99.92	75.00	24.92
ZeroSNet164	1, 0.87, 0.87	98.77	73.24	25.53
ZeroSNet164	1, 0.52, 0.52	99.92	77.77	22.15
ZeroSNet164	1, 0.33, 0.33	99.92	78.15	**21.77**
ZeroSNet326	1, 0.87, 0.87	99.47	73.38	26.09
ZeroSNet326	1, 0.52, 0.52	99.95	78.63	21.32
ZeroSNet326	1, 0.33, 0.33	99.95	79.26	**20.69**

Table 7: Generalization gap (%) on CIFAR-100. We use difference of the training and test accuracies (i.e., "training acc. − test acc.") to measure the generalization gap. Generally, as the moduli of roots decrease, generalization ability of the corresponding model improves.

# Layers	LMResNet	PreResNet	ZeroSNet
20	0.27M	0.28M	0.28M
32	0.47M	0.48M	0.48M
44	0.66M	0.67M	0.67M
56	0.86M	0.87M	0.87M
110	1.14M	1.74M	1.74M

Table 8: Parameter amount of ResNets, LMResNets, PreResNets, and ZeroSNets. Note that the parameter amount of the ZeroSNet is close to PreResNet.

Model	BS=128	BS=512	BS=4096
PreResNet20 (training)	1532	829	913
ZeroSNet20 (training)	1662	829	928
PreResNet20 (test)	142	140	212
ZeroSNet20 (test)	151	143	220

Table 9: Training and test runtime (second) of PreResNet20 and ZeroSNet20 on CIFAR-10.

pixels from $[0, 255]$ to $[0, 1]$. After noise injection, we clip the dirty data (negative-valued input pixels or values exceed 1) within $[0, 1]$.

Table 10 gives the mapping from coefficients to moduli of roots.

Related Work

Robustness of neural ODEs: Hanshu et al. gives a loss term to minimize the upper bound of the difference between end states and find that neural ODEs with continuous representation perform well on the robustness. Zhang et al. study the robustness through the lens of step size, and they find that small step size benefits both forward and backward propagation. Embedding Gaussian processes into a neural ODE improves the robustness, as in (Anumasa and Srijith 2021). By training multiple noise-injected ResNets to approximate the Feynman-Kac formula, a robust model is constructed in (Wang et al. 2019). Differently, we consider the robustness of discrete CNNs and bridge it with the network structure through zero stability.

Stability of CNNs: A-stability of CNNs is investigated in (Haber and Ruthotto 2017). The insight that the features should be well-posed in (Haber and Ruthotto 2017) is important for keeping the representation ability and away from explosions. Although the generalization is mentioned, the connection between it and A-stability is not clear in (Haber and Ruthotto 2017). Since A-stability does not involve perturbation, it may be irrelevant to the generalization. (Weinan 2017; Lu et al. 2018; Chen 2019) give the interpretation of deep neural networks from an ordinary differential equation (ODE) perspective. Based on those works, (Ruthotto and Haber 2020) further studies stability from a perspective of the partial differential equation (PDE). (Ruthotto and Haber 2020) constructs parabolic and hyperbolic CNNs, and proves that under certain assumptions (e.g., weight symmetry, special activation), the parabolic and hyperbolic CNNs are stable. Different from (Ruthotto and Haber 2020), we construct zero-stable CNNs based on high-order discretization and show that zero stability can predict performance well. (Zhang and Schaeffer 2020) studies the stability of sev-

α_0	α_1	α_2	β_0	Module of the 1st root	Module of the 2nd root	Module of the 3rd root	Z. S.
1.0000	1.0000	1.0000	1.0000	1.84	0.74	0.74	No
3.7500	-4.0000	1.2500	-0.5000	0.57	1.00	2.18	No
-3.0000	5.0000	-1.0000	4.0000	4.24	0.24	1.00	No
-0.7500	2.0000	-0.2500	2.5000	1.88	0.13	1.00	No
2.2500	-2.0000	0.7500	0.5000	1.00	0.87	0.87	Yes
0.1000	0.2000	0.3000	0.4000	0.81	0.61	0.61	Yes
0.5000	0.3000	0.1000	0.1000	0.94	0.33	0.33	Yes
0.8250	-0.1000	0.2750	1.4500	1.00	0.52	0.52	Yes
1.0000	0.3000	-0.4000	1.0000	0.60	0.82	0.82	Yes
0.3333	0.5556	0.1111	1.7778	0.33	0.33	1.00	Yes

Table 10: Mapping from coefficients to moduli of roots. "Z. S." denotes zero stability. Note that the theoretically optimal coefficients (1/3, 5/9, 1/9, and 16/9) are in decimal forms here (0.3333, 0.5556, 0.1111, and 1.7778).

eral ResNet-like networks, and it gives upper bounds of the output feature maps and the sensitivity bound. Differently, we use zero stability in numerical analysis (Gautschi 1997) and then provide guidance to construct high-order structures.

Structure based on high-order discretization: After interpreting some well-performed CNNs as ODEs, Lu et al. give the LM architecture. We interpret the LM architecture as a second-order discretization and use it as a tool for our preliminary observation on how zero stability affects model performance. Unlike the LM architecture, ZeroSNet in our work has a theoretical guarantee to be consistent and zero-stable. In our experiments, following the same settings of hyperparameters, ZeroSNet outperforms the LM-ResNet in (Lu et al. 2018).

Discussion

The well-performed ZeroSNet is somehow just a by-product for investigating the nature of CNNs. To speed up the training, we use plain settings for all experiments and apply the mixed-precision training on ImageNet, and our results cannot beat the state-of-the-art ones on the leaderboard. Besides, due to the space limitation, we only discuss the first- to third-order discretizations, but we believe the connection between performance and zero-stability is clear. A general theory for leading the structure designing is beyond this paper's scope, and it requires further exploration. The precise understanding of deep neural networks still needs more effort, and our work only takes a little step to this big problem's answer.

Conclusion

In this work, we first observe that zero stability well predicts the performance of PreResNets and LM-ResNets. Based on these preliminary observations, we construct a high-order CNN named ZeroSNet to further verify the prediction ability of zero stability. Theoretically, we prove ZeroSNet's advantages on consistency and zero-stability, with a group of optimal coefficients for historical features and the current activations deduced. Four groups of experiments are carried out in this paper. First, we compare ZeroSNets with their non-zero-stable counterparts, and the results clearly show that zero-stable models outperform non-zero-stable ones on generalization. Second, we evaluate the theoretically optimal coefficients on different datasets, and the results demonstrate that they are also optimal in practice. Then, ZeroSNet with the theoretically optimal coefficients and ZeroSNet with trainable parameters are employed for comparison. Results show that ZeroSNets outperform previous advanced CNNs on CIFAR-10, CIFAR-100, and ImageNet. Finally, experiments on test images injected with noise verify the superiority of zero-stable CNNs on the robustness.

Acknowledgements

This work was supported in part by the CAS "Light of West China" Program, in part by the Natural Science Foundation of Chongqing (China) under Grant cstc2020jcyj-zdxmX0028, in part by the National Nature Science Foundation of China under Grant 62176109, Grant 62072429, and Grant 61902370, in part by the Key Cooperation Project of Chongqing Municipal Education Commission under Grant HZ2021017 and Grant HZ2021008, and in part by the Chongqing Entrepreneurship and Innovation Support Program for Overseas Returnees under Grant CX2021100.

References

Anumasa, S.; and Srijith, P. 2021. Improving robustness and uncertainty modelling in neural ordinary differential equations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 4053–4061.

Ascher, U. M.; and Petzold, L. R. 1998. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. SIAM.

Atkinson, K.; Han, W.; and Stewart, D. E. 2011. *Numerical solution of ordinary differential equations*, volume 108. John Wiley & Sons.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6572–6583.

Chen, X. 2019. Ordinary differential equations for deep learning. *arXiv preprint arXiv:1911.00502*.

Gautschi, W. 1997. *Numerical analysis*. Springer Science & Business Media.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goyal, P.; Dollar, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Haber, E.; Lensink, K.; Treister, E.; and Ruthotto, L. 2019. IMEXnet: A forward stable deep neural network. In *International Conference on Machine Learning*, 2525–2534. PMLR.

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

Hanshu, Y.; Jiawei, D.; Vincent, T.; and Jiashi, F. 2019. On robustness of neural ordinary differential equations. In *International Conference on Learning Representations*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Jastrzebski, S.; Kenton, Z.; Arpit, D.; Ballas, N.; Fischer, A.; Bengio, Y.; and Storkey, A. 2018. Width of minima reached by stochastic gradient descent is influenced by learning rate to batch size ratio. In *International Conference on Artificial Neural Networks*, 392–402. Springer.

Jin, L.; Wei, L.; and Li, S. 2022. Gradient-based differential neural-solution to time-dependent nonlinear optimization. *IEEE Transactions on Automatic Control*, in press with DOI: 10.1109/TAC.2022.3144135.

Li, J.; Zhang, Y.; and Mao, M. 2019. General square-pattern discretization formulas via second-order derivative elimination for zeroing neural network illustrated by future optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3): 891–901.

Liu, M.; Chen, L.; Du, X.; Jin, L.; and Shang, M. 2021. Activated gradients for deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, in press with DOI: 10.1109/TNNLS.2021.3106044.

Lu, Y.; Zhong, A.; Li, Q.; and Dong, B. 2018. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, 3276–3285. PMLR.

Luo, X.; Liu, Z.; Jin, L.; Zhou, Y.; and Zhou, M. 2022. Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 33(3): 1203–1215.

Luo, X.; Qin, W.; Dong, A.; Sedraoui, K.; and Zhou, M. 2021. Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning. *IEEE/CAA Journal of Automatica Sinica*, 8(2): 402–411.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Ruthotto, L.; and Haber, E. 2020. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3): 352–364.

Wang, B.; Yuan, B.; Shi, Z.; and Osher, S. J. 2019. ResNets ensemble via the Feynman-Kac formalism to improve natural and robust accuracies. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 1657–1667.

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

Xie, S.; Kirillov, A.; Girshick, R.; and He, K. 2019. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1284–1293.

Zhang, J.; Han, B.; Wynter, L.; Low, B. K. H.; and Kankanhalli, M. 2019. Towards robust ResNet: A small Step but a giant leap. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 4285–4291.

Zhang, L.; and Schaeffer, H. 2020. Forward stability of ResNet and its variants. *Journal of Mathematical Imaging and Vision*, 62(3): 328–351.

Zhang, X.; Li, Z.; Change Loy, C.; and Lin, D. 2017. PolyNet: A pursuit of structural diversity in very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 718–726.

Zhu, M.; Chang, B.; and Fu, C. 2018. Convolutional neural networks combined with Runge-Kutta methods. *arXiv preprint arXiv:1802.08831*.