Robust Neural Networks inspired by Strong Stability Preserving Runge-Kutta methods

Byungjoo Kim[1⋆], Bryce Chudomelka[2⋆], Jinyoung Park[1],
Jaewoo Kang[1†], Youngjoon Hong[2], and Hyunwoo J. Kim[1†]

[1] Department of Computer Science, Korea University, Seoul, Republic of Korea
{byung4329,lpmn678,kangj,hyunwoojkim}@korea.ac.kr
[2] Department of Mathematics and Statistics, San Diego State University, San Diego, California, USA
{bchudomelka,yhong2}@sdsu.edu

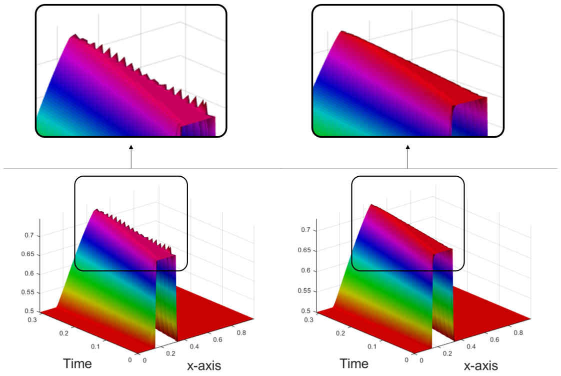
Abstract. Deep neural networks have achieved state-of-the-art performance in a variety of fields. Recent works observe that a class of widely used neural networks can be viewed as the Euler method of numerical discretization. From the numerical discretization perspective, Strong Stability Preserving (SSP) methods are more advanced techniques than the explicit Euler method that produce both accurate and stable solutions. Motivated by the SSP property and a generalized Runge-Kutta method, we proposed Strong Stability Preserving networks (SSP networks) which improve robustness against adversarial attacks. We empirically demonstrate that the proposed networks improve the robustness against adversarial examples without any defensive methods. Further, the SSP networks are complementary with a state-of-the-art adversarial training scheme. Lastly, our experiments show that SSP networks suppress the blow-up of adversarial perturbations. Our results open up a way to study robust architectures of neural networks leveraging rich knowledge from numerical discretization literature.

1 Introduction

Recent progress in deep learning has shown promising results in various research areas, such as computer vision, natural language processing and recommendation systems. In particular, on the ImageNet classification task [18], deep neural networks show state-of-the-art performance, e.g., residual networks (ResNet), which outperform humans in image classification [15]. Despite the success, deep neural networks often suffer from the lack of robustness against adversarial attacks [30]. ResNet, which is a widely used base network, also suffers from adversarial attacks which necessitates a more fundamental understanding of the architecture at hand.

One interesting interpretation of the ResNet architecture is that of the explicit Euler discretization scheme, i.e., $x(t_{k+1}) = x(t_k) + F(x(t_k))$, because it

⋆ Equal Contribution, † Corresponding Author.



(a) Euler method (b) 3rd order SSP

Fig. 1: We illustrate the difference between a forward Euler discretization and a third-order SSP discretization applied to the inviscid Burgers' solution. After computing numerical solutions, the solutions are filtered through the sigmoid function as an activation function. Evidently, in (a) the Euler scheme, i.e., a *ResBlock*, produces notable numerical errors while the SSP3 discretization in (b) shows a stable numerical approximation. For more details, please see the supplement.

allows one to view neural networks as numerical methods. The explicit Euler method is one of the simplest first-order numerical schemes but often leads to large numerical errors due to its low order. Thus, we would expect that applying advanced numerical discretizations would produce a more accurate numerical solution than the Euler method, such as an explicit high-order Runge-Kutta method. However, an arbitrary explicit high-order Runge-Kutta method can pose a stability problem if the numerical solution becomes unstable [27]. To tackle this issue, [9] and [27] introduce the notion of total variation diminishing (TVD); also called Strong Stability Preserving (SSP) methods. The strong stability preserving approach produces a more accurate solution of the differential equation than the Euler method. We would expect to obtain a more accurate solution of the underlying function with non-smooth initial data (shocks) compared to the Euler method without notable numerical errors, see Figure 1. This phenomenon is directly related to the problem of adversarial attack and robustness of neural networks [30].

Motivated by the advanced numerical discretization schemes, we propose novel network architectures with the SSP property that address robustness; SSP networks (SSPNets). The use of the SSP property consistently demonstrates that all of our proposed architectures outperform ResNet in terms of robustness. SSP architectural blocks do not increase the amount of model parameters compared to ResNet, can be easily implemented, and realized by a convex combination of existing ResNet modules. The parameters used in SSP blocks are *mathematically* *derived coefficients* from the advanced numerical discretization methods. In addition, starting from an explicit Runge-Kutta method with the SSP property, we propose novel Adaptive Runge-Kutta blocks with *learned coefficients* obtained by training. With these learned coefficients, we are able to improve robustness while retaining the natural accuracy of ResNet.

The simple architectural change, SSPNets, improve robustness and are complementary with adversarial training, which is the *de facto* state-of-the-art defensive methodology. Our **contributions** are summarized as follows:

- We propose multiple novel architectural blocks motivated by the Strong Stability Preserving explicit higher-order numerical discretization method.
- We demonstrate empirically that these proposed blocks improve the robustness of Strong Stability Preserving networks consistently; against adversarial examples and without any defensive methods.
- We further improve on robustness with a novel adaptive architectural block motivated by a generalized Runge-Kutta method and the SSP property.
- Last but not least, we show that Strong Stability Preserving Networks suppress the blow-up of adversarial perturbations added to inputs.

2 Background and Related Work

2.1 Neural Networks and Differential Equations

Neural networks such as ResNet [15], PolyNet [40] and recurrent neural networks share a common operation represented as $x_{t+1} = x_t + F(x_t; \Theta_t)$. Interestingly, a sequence of the operations (or equivalently the network architectures) can be interpreted as an explicit Euler method for numerical discretization [6, 7, 20, 24, 25]. For instance, ResNet can be written mathematically as

$$\begin{aligned} &x_0 = x, \\ &x_{k+1} = x_k + F(x_k; \Theta_k), \quad k \in \{0, 1, \dots, A-1\}, \\ &\hat{y} = f(x_A), \end{aligned} \tag{1}$$

where A denotes the number of layers in the network.

If we multiply the function F by Δt, i.e., $x_{k+1} = x_k + \Delta t F(x_k; \Theta_k)$, then ResNet can be seen as the explicit Euler numerical scheme discretization with an initial condition, $x(0)$, to solve the initial value problem given as

$$\begin{aligned} &x(0) = x, \\ &\frac{dx(t)}{dt} = F(x(t); \Theta(t)), \\ &\hat{y} = f(x(A)). \end{aligned} \tag{2}$$

The explicit Euler method is the simplest Runge-Kutta method and often suffers from low accuracy because it is a first-order method. In this regard, higher-order numerical methods are natural candidates to obtain a more precise numerical solution, but the higher accuracy from higher-order methods may come with the cost of instability, e.g., poor convergence behaviour on stiff differential equations compared to the first order Euler method [4]. Therefore, it is important to understand the trade-off between accuracy and stability when considering a numerical method.

Recently, some network architectures inspired by the computational similarity between ResNet and Euler discretization have been proposed, e.g., NeuralODE and FFJORD [6, 11]. Unlike ResNet, which requires the discretization of observation/emission intervals to be represented by a finite number of hidden layers, NeuralODE and FFJORD use numerical discretization methods in the forward propagation to define continuous-depth and continuous-time latent variable models. These require ODE solvers for training and inference, unlike our implementation of SSP networks. Since we changed only computational graphs and coefficients based on the numerical discretization theory, our methods perform the standard forward/backward propagation in the discrete space as ResNet.

Another approach to design new blocks/layers of neural networks is to make them have operations similar to advanced numerical discretization techniques that possess desirable properties [20, 25]. From the partial differential equation perspective, analysis on numerical stability of conventional residual connections lead to the development of new architectures: parabolic/hyperbolic CNNs to achieve better stability as parabolic/hyperbolic PDEs [25]. The models use theoretical assumptions on the function to achieve stability with a positive semidefinite Jacobian of the function resulting in constraints on convolutional kernels; alternatively, our networks do not require such constraints.

2.2 Robust Machine Learning and Adversarial Attacks

Stability and robustness of neural networks have been studied in the context of adversarial attacks after the success of deep learning [2, 30, 36]. Gradient-based adversarial attacks create adversarial examples solving optimization problems. One example is the maximization of loss against ground truth labels within a small ball, e.g., $\max_\delta \mathcal{L}(h_\theta(x+\delta), y),\ s.t.\ \|\delta\|_\infty \leq \epsilon$, where h_θ is a model parameterized by θ, x, y are the input (natural sample) and its target label respectively, and $\mathcal{L}$ is a loss function. The simplest procedure to approximate the solution is to use the fast gradient sign method (FGSM) [8]. It can be seen as an optimal solution to a linearized loss function, i.e., $\arg\max_{\|v\|_\infty \leq \alpha} v^T \nabla_\delta \mathcal{L}(h_\theta(x+\delta), y) = \alpha \cdot \mathrm{sign}(\nabla_\delta \mathcal{L}(h_\theta(x+\delta), y))$. Furthermore, the FGSM can be more powerful when it is used with iterative methods such as the projected gradient descent (PGD). PGD has been used in both untargeted and targeted attacks [21, 5].

One of the early attempts to defend against adversarial attacks is adversarial training using FGSM, a single-step method [8]. After that, various defensive techniques have been proposed [3, 22, 26, 29, 31]. Many of them were defeated by iterative attack methods [5] and Backward Pass Differentiable Approximation [1]. Adversarial training with stronger multi-step attack methods is still promising and shows state-of-the-art performance [21, 32, 35]. More recently, *provably* robust neural networks have been successfully trained by minimizing the lower bound of risk based on convex duality and convex relaxation [33, 34]. Most adversarial training methods above assume that attack methods are known *a priori*, i.e., a *white-box* attack, and generate augmented samples using the attacks. Another defensive technique is to alleviate the effect of perturbation by augmentation and reconstruction [23, 37], or denoising [35]. These methods alongside adversarial training achieved comparable robustness. Similarly, in this work we will introduce our approach and evaluate it with adversarial training.

3 Strong Stability Preserving Networks

In this section, we introduce the mathematical framework for the Strong Stability Preserving property and describe how to implement SSP blocks with *mathematically derived coefficients*. Next, we provide a variance analysis to compare high-order Runge-Kutta blocks with residual blocks. Lastly, we introduce adaptive Runge-Kutta blocks with *learnable coefficients* which possess the SSP property.

3.1 Motivation of strong stability preserving method

Our objective is to solve the *non-autonomous* differential equation given as

$$\frac{\partial u}{\partial t} = L(u(t), t), \quad t \in [t_0, ..., t_N], \tag{3}$$

where t_0, t_N are the initial and terminal time state respectively; a non-autonomous system permits a time varying solution, e.g., the learned function varies as the depth of the network increases. The function L is a linear (or nonlinear) function and $u(t_0)$ is given by the initial condition. The objective is to figure out the terminal state of the function u, i.e., $u(t_N)$.

A general high-order Runge-Kutta time discretization for solving the initial value problem (3) introduced in [28] is given as

$$\begin{aligned} u^{(0)} &= u^n, \\ u^{(i)} &= \sum_{k=0}^{i-1} \left(\alpha_{i,k} u^{(k)} + \Delta t \beta_{i,k} L(u^{(k)}) \right), \quad i \in \{1, \cdots, m\}, \\ u^{n+1} &= u^{(m)}, \end{aligned} \tag{4}$$

where $\sum_{k=0}^{i-1} \alpha_{i,k} = 1$ and $\alpha_{i,k} \geq 0$. For example, if $m = 1$, it becomes the first-order Euler method as in Equation (1) with $\alpha_{1,0} = \beta_{1,0} = 1$.

Shu et al. [27, 28] propose a TVD time discretization method that is called the SSP time discretization method; for more discussion on the TVD method, we refer the reader to [12, 13]. The procedure of TVD time discretization is to take the high-order method to decrease the local truncation error and maintain the stability under a suitable restriction on the time step. While applying the TVD scheme into the explicit high-order Runge-Kutta methods, there needs the assumption to hold it: *The first-order Euler method in time is strongly stable under a certain (semi) norm when the time step Δt is suitably restricted* [10]. More precisely, if we assume that the forward Euler time discretization is stable under a certain norm, the SSP methods find a higher-order time discretization that maintains strong stability for the same norm; improving accuracy.

Followed by this assumption, for a sufficiently small time step known as Courant-Friedrichs-Lewy (CFL) condition $\Delta t \leq \Delta t_{CFL}$, the total variation seminorm of the numerical scheme does not increase in time, that is,

$$TV(u^{n+1}) \leq TV(u^n), \tag{5}$$

where the total variation is defined by

$$TV(u^n) := \sum_j |u_{j+1}^n - u_j^n|, \tag{6}$$

where j is the spatial discretization. The explicit high-order Runge-Kutta discretization with the SSP property maintains a higher order accuracy with a modified CFL condition $\Delta t \leq c\Delta t_{CFL}$. In other words, the high-order SSP Runge-Kutta scheme improves accuracy while retaining its stability. This has been theoretically studied by the following Lemma 1.

Lemma 1. *If the forward Euler method is strongly stable under the CFL condition, i.e.* $||u^n + \Delta t L(u^n)|| \leq ||u^n||$*, then the Runge-Kutta method possesses SSP,* $||u^{n+1}|| \leq ||u^n||$*, provided that* $\Delta t \leq c\Delta t_{CFL}$*.*

We provide a sketch of the proof of Lemma 1 in the supplement. The full proof of the Lemma 1 can be found in [28]. Following this representation, we can figure out the specific coefficients $\alpha_{i,k}$ and $\beta_{i,k}$ in equation (4). In particular, the second and third order nonlinear SSP Runge-Kutta method was studied in [28].

Lemma 2. *An optimal second-order SSP Runge-Kutta method is given by,*

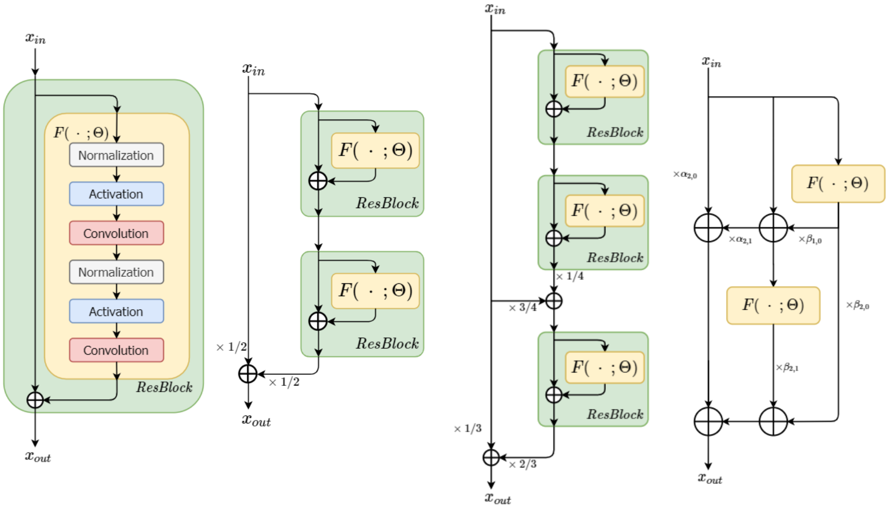
$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{n+1} &= \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t L(u^{(1)}), \end{aligned} \tag{7}$$

with a CFL coefficient $c = 1$*. In addition, an optimal third-order SSP Runge-Kutta method is of the form*

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}), \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}), \end{aligned} \tag{8}$$

with a CFL coefficient $c = 1$*.*

A sketch of the proof for Lemma 2 can be found in the supplement and for the detailed proof, we refer the reader to [9, 10, 28].



(a) (b) (c) (d)

Fig. 2: Network modules with *ResBlock* and SSP blocks. (a): *ResBlock*. (b): *SSP2-block* (c): *SSP3-block*, (d): *ArkBlock*

3.2 Strong Stability Preserving Networks

Next, we show how to incorporate the explicit SSP Runge-Kutta method into neural networks. Equation (7) and (8) can be implemented with standard residual blocks and simple operations, as shown in Figure 2.

Let *ResBlock* denote a standard residual block written as $\mathit{ResBlock}(x(t_k); \mathit{\Theta}(t_k)) = x(t_k) + F(x(t_k); \mathit{\Theta}(t_k))$, where $\mathit{\Theta}(t_k)$ are the parameters of $\mathit{ResBlock}(\cdot;\mathit{\Theta}(t_k))$. The function F is typically composed of two or three sets of normalization, activation, and convolutional layers, e.g., Figure 2 and [15, 16]. When the numbers of input and output channels differ, we use the expansive residual block *ResBlock-E*; this can be implemented with a 1×1 convolutional filter to expand the number of channels.

Using the standard modules in ResNet (*ResBlock* and *ResBlock-E*), SSPNets can be constructed. First, SSP blocks can be implemented using linear combinations of *ResBlock*s. As the Euler method interpretation of ResNet requires $\mathit{\Delta} t = 1$, we assume $\mathit{\Delta} t = 1$ in Equation (7), then the *SSP2-block* is given by,

$$\begin{aligned} x(t_{k+\frac{1}{2}}) &= \underbrace{x(t_k) + F(x(t_k); \mathit{\Theta}(t_k))}_{\mathit{ResBlock}(x(t_k);\mathit{\Theta}(t_k))}, \\ x(t_{k+1}) &= \frac{1}{2}x(t_k) + \underbrace{\frac{1}{2}x(t_{k+\frac{1}{2}}) + \frac{1}{2}F\left(x(t_{k+\frac{1}{2}}); \mathit{\Theta}(t_k)\right)}_{\frac{1}{2}\mathit{ResBlock}\left(x\left(t_{k+1/2}\right);\mathit{\Theta}(t_k)\right)}. \end{aligned} \tag{9}$$

Similarly, the third order SSP in Equation (8) (*SSP3-block*) is written as

$$\begin{aligned} x(t_{k+\frac{1}{3}}) &= \underbrace{x(t_k) + F\left(x(t_k); \Theta(t_k)\right)}_{ResBlock(x(t_k);\Theta(t_k))}, \\ x(t_{k+\frac{2}{3}}) &= \frac{3}{4}x(t_k) + \underbrace{\frac{1}{4}x(t_{k+\frac{1}{3}}) + \frac{1}{4}F\left(x(t_{k+\frac{1}{3}}); \Theta(t_k)\right)}_{\frac{1}{4}ResBlock\left(x\left(t_{k+1/3}\right);\Theta(t_k)\right)}, \\ x(t_{k+1}) &= \frac{1}{3}x(t_k) + \underbrace{\frac{2}{3}x(t_{k+\frac{2}{3}}) + \frac{2}{3}F\left(x(t_{k+\frac{2}{3}}); \Theta(t_k)\right)}_{\frac{2}{3}ResBlock\left(x\left(t_{k+2/3}\right);\Theta(t_k)\right)}. \end{aligned} \tag{10}$$

The SSP block schematic is presented in Figure 2 and SSP blocks are only used when the number of channels does not change.

The explicit SSP Runge-Kutta methods in Equation (7) and (8) use the same function L multiple times. Similarly, SSP blocks in Equation (9) and (10) apply the same *ResBlock* multiple times. Using the same *ResBlock* multiple times can be viewed as parameter sharing, which is a kind of regularization. In other words, without increasing the number of parameters, a SSP block implementation improves the robustness of neural networks by utilizing higher-order schemes.

Midpoint Runge-Kutta Second-Order Methods. For contrast, one may ask whether or not the stability preserving properties are key to the robustness against adversarial perturbation. We address this important question by training another network that utilizes a second-order midpoint Runge-Kutta method (mid-RK2) which does not have the strong stability preserving property [4, 10]. Recall that this method is implemented numerically as

$$x(t_{k+1}) = x(t_k) + F\left(x(t_k) + \frac{1}{2}F(x(t_k); \Theta(t_k)); \Theta(t_k)\right), \tag{11}$$

and does not have the SSP property. This network will provide a comparison of numerical discretization methods with regard to stability in attacked accuracy.

Variance Analysis of SSP networks. We analyze the variance increase of SSP blocks following previous works [14, 39], which compare the variance of input and output of functional modules. Next, we show that SSP blocks suppress the variance increase compared to *ResBlock*; as well as comparing the variance of the midpoint Runge-Kutta second-order numerical method for further justification.

Lemma 3. *If* $\mathtt{Var}[F(x)] = \mathtt{Var}[x]$, $\mathtt{Cov}[x, F(y)] = 0$ *then the variance increases by*

$$\begin{aligned} \mathtt{Var}[ResBlock(x)] &= 2\,\mathtt{Var}[x], & \mathtt{Var}[\textit{mid-RK2(x)}] &= \frac{9}{4}\mathtt{Var}[x], \\ \mathtt{Var}[\textit{SSP2-Block(x)}] &= \frac{7}{4}\mathtt{Var}[x], & \mathtt{Var}[\textit{SSP3-Block(x)}] &= \frac{29}{18}\mathtt{Var}[x]. \end{aligned} \tag{12}$$

The variance of SSP blocks is smaller than that of *ResBlock*. The variance adds to our argument that the SSP property is the reason for improved robustness; for more detailed derivation and proof, see the supplement.

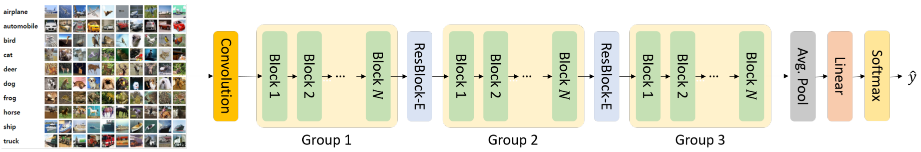


Fig. 3: The overall architecture of neural networks used in experiments. Each group has $N \in \{6, 10\}$ blocks and the block is either *ResBlock*, SSP blocks (2 or 3) or *ArkBlock*. The *ResBlock-E* is inserted between groups to expand the number of channels for all the architectures.

Adaptive SSP Networks. Also, we generalize Equation (4) with the second-order Adaptive Runge-Kutta block (*ArkBlock*) that has the SSP property by construction. These novel computational blocks slightly increase the number of parameters compared to *ResBlock* but also provide greater robustness and natural accuracy than *SSP2-Block* or *SSP3-Block*. Finally, we explore different computational architectures within each group to retain natural accuracy and further improve robustness.

A naive implementation of Equation (4) yields 5 additional parameters. We can retain the SSP property in *ArkBlocks* by reducing the number of parameters with Ralston's method [9]. Thus, the number of additional learned parameters per block, when compared with *ResBlock*, is 2 and is defined as

$$\begin{aligned} &\alpha_{1,0} = 1, \qquad \alpha_{2,0} = 1 - \alpha_{2,1}, \\ &\beta_{2,0} = 1 - \frac{1}{2\beta_{1,0}} - \alpha_{2,1}\beta_{1,0}, \qquad \beta_{2,1} = \frac{1}{2\beta_{1,0}}. \end{aligned} \tag{13}$$

We further improve performance by reducing the number of parameters by fixing $\alpha_{2,1}$ and simply learning $\beta_{1,0}$ in each block.

Adaptive SSP networks still maintain the same architecture, as in Figure 3, but are comprised of blocks that have the form

$$\begin{aligned} u^{(1)} &= u^n + \beta_{1,0} L(u^n), \\ u^{(n+1)} &= \alpha_{2,0} u^n + \beta_{2,0} L(u^n) + \alpha_{2,1} u^{(1)} + \beta_{2,1} L(u^{(1)}). \end{aligned} \tag{14}$$

We implement *ArkBlock*s with,

$$\begin{aligned} x(t_{k+\frac{1}{2}}) &= x(t_k) + \beta_{1,0} F\left(x(t_k); \Theta(t_k)\right), \\ x(t_{k+1}) &= \alpha_{2,0} x(t_k) + \beta_{2,0} F\left(x(t_k); \Theta(t_k)\right) \\ &\quad + \alpha_{2,1} x(t_{k+\frac{1}{2}}) + \beta_{2,1} F\left(x(t_{k+\frac{1}{2}}); \Theta(t_k)\right). \end{aligned} \tag{15}$$

The *ArkBlock*s are inspired by the generalized Runge-Kutta method in (14). However, the numerical scheme in Equation (14), keeps $\alpha_{2,1}$ and $\beta_{1,0}$ constant in all blocks, while *ArkBlock*s set those parameters as *learnable*; varying in each block. Such an adaptivity based on data and architectures cannot be obtained by mathematically derived coefficients. To our knowledge, this is the first attempt.

Model	Clean	FGSM	PGD$_{20}$	PGD$_{30}$
ResNet	0.9961	0.7674	0.5799	0.1773
SSP-2	0.9954	0.7984	0.5979	0.1850
SSP-3	0.9960	0.8022	0.6176	0.1930
SSP-adap	0.9946	**0.8586**	**0.7611**	**0.5102**

Table 1: The accuracy against adversarial attacks with standard training on the MNIST dataset; all models were trained with 6 blocks. Note that PGD$_i$ represents a projected gradient descent attack with i iterations and that all the SSPNets are more robust against adversarial attacks than ResNet.

4 Experiments

We evaluate the robustness of various SSP networks against adversarial examples. MNIST [19] and CIFAR10 [18] are used for evaluation; for results on other datasets, see the supplement. The robustness is measured by the classification accuracy on adversarial examples generated by FGSM [8] and PGD [21].

In this section, we empirically address the following three questions:

- Are deep neural networks with the SSP property more robust than ResNet when the models are trained with or without adversarial training?
- Can we further improve upon adversarial robustness and simultaneously retain the natural accuracy of ResNet?
- Do Strong Stability Preserving networks suppress the perturbation growth during forward propagation?

4.1 Experimental setup

ResNet and SSP networks. Each group has N blocks where each block can be either *ResBlock*, *SSP2-block*, *SSP3-block*, or *ArkBlock*, as seen in Figure 3. Networks are named after the type of blocks: ResNet, SSP-2, SSP-3, and SSP-adap. The blocks in each group have the same number of input/output channels. The convolutional layers in group 1, group 2, and group 3 have 16, 32, 64 channels respectively. The classification layer of our networks consist of an average pooling and softmax layer, in order to calculate the confidence score.

4.2 Evaluation on MNIST with standard training

We demonstrate that SSPNets are more robust than ResNet with standard training. Since MNIST has relatively low-resolution images compared to CIFAR10, we used a smaller architecture by skipping group 1 and 2 in Figure 3.

Experimental Details. We evaluate the models on MNIST. When training the models, samples are augmented by adding random noise δ drawn from a uniform distribution $Uniform(-\epsilon, \epsilon)$. We set the maximum perturbation magnitude $\epsilon = 0.3$ for both training and evaluation. For optimization, Adam [17] is used with learning rate 0.0001 and $(\beta_1, \beta_2) = (0.9, 0.999)$, minibatch size of 128. Models are trained for 100 epochs.

Robustness Comparison. The results in Table 1 show that all four models have high accuracy ($99.5 \sim 99.6\%$) in classifying clean samples. This means that SSP blocks do not lead to a significant loss of accuracy on clean samples. Further, the improvement by SSP compared to ResNet is consistently observed in different settings. SSP-2 improves the robustness by 3% against FGSM and 1% against PGD. SSP-3 shows larger improvement about 4% and 2% against FGSM and PGD. SSP-adap shows the largest improvement about 9% and 33% against FGSM and PGD. It is known that adversarial training on MNIST is sufficiently robust against FGSM and PGD. All models trained by adversarial training achieve $96 \sim 97\%$ on MNIST, which makes it hard to demonstrate the benefit of SSP networks with adversarial training compared to ResNet.

4.3 SSP with adversarial training

We analyze the robustness of SSP networks, on the CIFAR10 dataset. Our preliminary experiments show that all the models, e.g., ResNet, SSP-2, SSP-3, and SSP-adap trained without adversarial training are easily fooled by PGD attacks, but more analysis is needed on a more challenging dataset. For this reason, we focus on the adversarial training setting for CIFAR10. Please see supplementary materials for more analysis on SSP networks with adversarial training.

Adversarial Training. Before experimental results, we briefly summarize the adversarial training proposed by [21]. The objective of adversarial training is to minimize the adversarial risk given as,

$$R_{adv}(h_\theta) = \mathbb{E}_{(x,y)\sim D}\big[\max_{\delta\in\varDelta} \mathcal{L}(h_\theta(x+\delta), y)\big], \tag{16}$$

where the h_θ is a model parameterized by θ, $\mathcal{L}$ is a loss function, y is the label of corresponding image x, D is a true data distribution, and $\varDelta$ is a set of small perturbations satisfying $\|\delta\|_p \leq \epsilon$. In our experiments, the ℓ_∞ metric is used, i.e., $p = \infty$. Finding the exact solution to $\max\limits_{\delta\in\varDelta} \mathcal{L}(h_\theta(x+\delta), y)$ is intractable, so [21] approximate it with a sample generated by the PGD attack. PGD attack finds the adversarial example given as $x_{i+1} = \varPi(x_i + \alpha\nabla_{x_i}\mathcal{L}(h_\theta(x_i), y))$, where $i \in \{0, 1, \cdots, K-1\}$, K is the number of iterations of PGD attack, $\varPi$ denotes the projection to a small ball $\varDelta$ and a valid pixel range. In our experiment, x_0 is initialized with the input image augmented by adding the random perturbation δ_0 sampled from the uniform distribution $Uniform(-\epsilon, \epsilon)$.

To summarize, our adversarial training procedure works as follows: First, randomly perturb the image within the allowed perturbation range ϵ. Next, generate the candidate adversarial example by PGD attack. Finally, take the gradient descent step on a minibatch composed of only candidate adversarial examples. The adversarial training is closely related to the Frank-Wolfe Algorithm and two projections in the original adversarial training can be simplified to one projection to the intersection of two convex sets. The pseudocode of adversarial training and a detailed discussion of implementation are provided in the supplement.

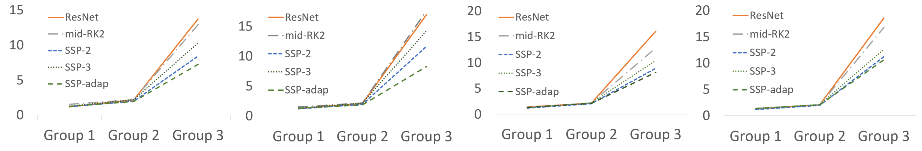
N	K	Model	Clean	FGSM	PGD_7	PGD_{12}	PGD_{20}
6	7	ResNet	0.8357	0.5116	0.4389	0.4215	0.4150
6	7	mid-RK2	**0.8407**	0.5156	0.4377	0.4193	0.4129
6	7	SSP-2	0.8257	0.5223	0.4577	0.4426	0.4368
6	7	SSP-3	0.8376	0.5165	0.4478	0.4305	0.4246
6	7	SSP-adap	0.8376	**0.5283**	**0.4640**	**0.4455**	**0.4403**
6	12	ResNet	**0.8010**	0.5304	0.4817	0.4691	0.4650
6	12	mid-RK2	0.7957	0.5326	0.4849	0.4740	0.4693
6	12	SSP-2	0.7899	0.5426	0.5073	0.4983	0.4961
6	12	SSP-3	0.7966	0.5440	**0.5092**	**0.4999**	**0.4976**
6	12	SSP-adap	0.7988	**0.5504**	0.5066	0.4964	0.4943
10	7	ResNet	**0.8516**	0.5225	0.4398	0.4188	0.4111
10	7	mid-RK2	0.8451	0.5146	0.4343	0.4122	0.4045
10	7	SSP-2	0.8437	**0.5373**	0.4714	0.4502	0.4427
10	7	SSP-3	0.8505	0.5350	**0.4719**	**0.4558**	**0.4497**
10	7	SSP-adap	0.8504	0.5308	0.4592	0.4376	0.4310
10	12	ResNet	0.8181	0.5467	0.4957	0.4799	0.4755
10	12	mid-RK2	**0.8198**	0.5522	0.4968	0.4818	0.4775
10	12	SSP-2	0.8144	0.5497	0.5074	0.4957	0.4932
10	12	SSP-3	0.8119	0.5507	0.5032	0.4929	0.4890
10	12	SSP-adap	0.8156	**0.5643**	**0.5166**	**0.5054**	**0.5016**

Table 2: CIFAR10 robustness evaluation against adversarial attacks. The column index N indicates the number of blocks in each group of Figure 3, K indicates the number of PGD iterations during training while PGD_i represents the attack with i iterations during attack. The SSP-adap model indicates an adaptive Runge-Kutta structure. All the SSP networks are more robust against adversarial attack than ResNet. Moreover, SSP-adap maintains the natural accuracy.

Experimental Details. We use the Stochastic Gradient Descent method with Nesterov momentum, learning rate of 0.1, weight decay of 0.0005, momentum 0.9, and a minibatch size of 128 samples. All models are trained for 200 epochs and in every 60, 100, 140 epochs, the learning rate decayed with a decaying factor 0.1. Both adversarial training and robustness evaluation, we set the maximum perturbation range $\epsilon = 8/255$. To evaluate the robustness, we use FGSM [8] and PGD [21]; similar to our MNIST experiments. We set the PGD attack parameters to $\alpha = 2/255$, and the number of iterations $K = 7, 12, 20$ in evaluation.

Robustness Comparison. The experimental results are shown in Table 2. Models are evaluated in four different settings varying both the number of blocks (6 or 10 in column N) for each group in Figure 3 and the number of iterations in PGD (7 or 12 in column K) to generate adversarial examples during training.

Before discussing about the effectiveness of SSP, we briefly show the relationship among robustness, the amount of model parameters, and the strength of attacks used in adversarial training. As shown in Table 2, the robustness of all the models is improved by stronger attacks during training (e.g., larger K in PGD). The same observation is reported in [32]. For instance, SSP-3 (N=6, K=12) shows higher accuracy than SSP-3 (N=6, K=7) against all the attacks and especially the improvement is about 7% against PGD with 20 iterations.



(a) N=6,K=7,p=1 **(b)** N=6,K=7,p=2 **(c)** N=10,K=12,p=1 **(d)** N=10,K=12,p=2

Fig. 4: Perturbation growth ratio in Equation (17) of clean samples and its adversarial counterparts. As the SSP networks suppress the perturbation growth during forward propagation, SSP-2, SSP-3 and SSP-adap have a lower ratio than ResNet. For full version of this figure, see the supplement.

Also, a bigger model size (e.g., larger N) increases the robustness against adversarial examples. This is closely related to the finding in [21] that increasing the number of channels in hidden layers often improves the robustness. Our experiments show that increasing the model size by adding more layers improves the robustness. For example, when $K = 7$, SSP-3 with $N = 10$ blocks show overall higher accuracy than SSP-3 with $N = 6$ blocks, the gain is about 2%. From the numerical discretization perspective, more blocks can be seen as a finer time discretization that leads to a more accurate numerical solution (or prediction).

All SSPNets, SSP-2, SSP-3 and SSP-adap, consistently outperform ResNet by, roughly, $1 \sim 3.9\%$ when ResNet and SSPNets have the same number of blocks, N, and iterations, K, in adversarial training. Note that we compare SSP-2 (and SSP-3) with ResNet, which has the same amount of parameters, and this is important to assure that the gain is not from an increased the amount of model parameters. Also, SSP-2, SSP-3 and SSP-adap have the same time discretization as ResNet. So, we conclude that the improvement in robustness against adversarial attacks solely comes from the strength of a higher-order numerical discretization. Table 2 shows one more interesting property of SSP networks. Unlike adversarial training and defensive methods that usually cause the label leaking effects [31, 38], SSP-2, SSP-3 and SSP-adap (our architectural changes) do not bring any additional loss of accuracy on natural samples.

On the other hand, Table 2 also shows that the mid-RK2 architecture does not outperform ResNet, SSP-2, SSP-3 or SSP-adap even though the mid-RK2 is derived from the second order numerical scheme. This gives credence to the implementation of SSPNets and implies that the robust performance is not a result of arbitrary high-order methods. In addition, SSP-adap achieves comparable natural accuracy as ResNet and improves robustness. Table 2 demonstrates the consistent improvement across various settings. For example, SSP-adap achieves nearly 4% absolute performance improvement for $N = 10, K = 7$. The improvement by SSP networks compared to ResNet and the performance difference between different SSP networks are relatively smaller than Table 1. Our conjecture is that this is due to the improvement by adversarial training. We believe that the Strong Stability Preserving property imposed by our architectural change allows the SSPNets to improve the robustness against adversarial attacks.

Perturbation Growth Ratio Comparison. We investigate how the distance between clean samples and adversarial examples evolves through networks by calculating the perturbation growth ratio between input/output of groups given by

$$\texttt{PGR}(f) = \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{x'\sim\mathcal{X}'}\left[\frac{\|f(x)-f(x')\|_p}{\|x-x'\|_p}\right]\right], \quad p\in\{1,2\} \tag{17}$$

where $f(\cdot)$ is a function of a group, x' is a corrupted sample from x and is an element of the set $\mathcal{X}'$, $\mathcal{X}'$ is a small neighborhood of x, and p defines a type of norm either ℓ_1 (related to TV in Equation (5)) or ℓ_2 (related to Lemma 3). Since each model has a different scale of feature maps, to compare, the distance needs proper normalization. So, we first measure the distance between a clean sample and its adversarial example before/after each group in Figure 3. x' is the adversarial example generated by PGD attack with 20 iterations for each model.

Figure 4 presents the perturbation growth ratio when $N = 6, K = 7$ and $N = 10, K = 12$ at each group in the models. Since the adversarial examples change the final predictions, the perturbation growth ratio increases in all the models. However, for SSPNets, the perturbation growth ratio is significantly lower than ResNet. This result supports that the proposed SSP blocks improve robustness of networks against adversarial attacks when compared to ResNet. We also conducted an experiment when x' is corrupted by adding a random perturbation to x and the result is consistent with Figure 4. For full version of Figure 4 and more discussion, see the supplement.

5 Conclusion

In this work, we leverage the Strong Stability Preserving property of numerical discretization in order to improve adversarial robustness. Inspired by the Strong Stability Preserving methods, we design a series of SSPNets by applying the same *ResBlock* multiple times with parameters derived from numerical analysis. All of the SSP networks provide robustness against adversarial attacks. In particular, SSPNets with the *ArkBlock* improve adversarial robustness while maintaining natural accuracy. The proposed networks are complementary with adversarial training and suppress the perturbation growth. Our work shows the way to improve the robustness of neural networks by utilizing the theory of advanced numerical discretization schemes. We believe that the intersection of numerical discretization and robust deep learning will provide new opportunities to study robust neural networks. [1]

Acknowledgements This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00533), National Supercomputing Center with supercomputing resources including technical support (KSC-2019-CRE-0186), National Research Foundation of Korea (NRF-2020R1A2C3010638), and Simons Foundation Collaboration Grants for Mathematicians.

[1] The codes are available at https://github.com/matbambbang/sspnet.

References

1. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: ICML. pp. 274–283 (2018)
2. Ben-Tal, A., El Ghaoui, L., Nemirovski, A.: Robust optimization, vol. 28. Princeton University Press (2009)
3. Buckman, J., Roy, A., Raffel, C., Goodfellow, I.: Thermometer encoding: One hot way to resist adversarial examples. In: ICLR (2018)
4. Butcher, J.C.: The numerical analysis of ordinary differential equations. A Wiley-Interscience Publication, John Wiley & Sons, Ltd., Chichester (1987), runge Kutta and general linear methods
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57 (2017)
6. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: NeurIPS. pp. 6572–6583 (2018)
7. Ciccone, M., Gallieri, M., Masci, J., Osendorfer, C., Gomez, F.: NAIS-Net: stable deep networks from non-autonomous differential equations. In: NeurIPS. pp. 3025–3035 (2018)
8. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
9. Gottlieb, S., Shu, C.W.: Total variation diminishing runge-kutta schemes. Mathematics of computation of the American Mathematical Society **67**(221), 73–85 (1998)
10. Gottlieb, S., Shu, C.W., Tadmor, E.: Strong stability-preserving high-order time discretization methods. SIAM Rev. **43**(1), 89–112 (2001)
11. Grathwohl, W., Chen, R.T., Betterncourt, J., Sutskever, I., Duvenaud, D.: FFJORD: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367 (2018)
12. Harten, A.: High resolution schemes for hyperbolic conservation laws. J. Comput. Phys. **49**(3), 357–393 (1983)
13. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.R.: Uniformly high-order accurate essentially nonoscillatory schemes. III. J. Comput. Phys. **71**(2), 231–303 (1987)
14. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
16. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV. pp. 630–645. Springer (2016)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2014)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
19. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
20. Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: ICML. pp. 5181–5190 (2018)

21. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
22. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 582–597. IEEE (2016)
23. Raff, E., Sylvester, J., Forsyth, S., McLean, M.: Barrage of random transforms for adversarially robust defense. In: CVPR (June 2019)
24. Rubanova, Y., Chen, R.T., Duvenaud, D.: Latent odes for irregularly-sampled time series. arXiv preprint arXiv:1907.03907 (2019)
25. Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations. arXiv preprint arXiv:1804.04272 (2018)
26. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: ICLR (2018)
27. Shu, C.W.: Total-variation-diminishing time discretizations. SIAM Journal on Scientific and Statistical Computing **9**(6), 1073–1084 (1988)
28. Shu, C.W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. Journal of computational physics **77**(2), 439–471 (1988)
29. Song, Y., Kim, T., Nowozin, S., Ermon, S., Kushman, N.: Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In: ICLR (2018)
30. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
31. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: ICLR (2019)
32. Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., Gu, Q.: On the convergence and robustness of adversarial training. In: ICML. pp. 6586–6595 (2019)
33. Wong, E., Kolter, Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: ICML. pp. 5283–5292 (2018)
34. Wong, E., Schmidt, F., Metzen, J.H., Kolter, J.Z.: Scaling provable adversarial defenses. In: NeurIPS. pp. 8400–8409 (2018)
35. Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: CVPR. pp. 501–509 (2019)
36. Xu, H., Caramanis, C., Mannor, S.: Robustness and regularization of support vector machines. Journal of Machine Learning Research **10**(Jul), 1485–1510 (2009)
37. Yang, Y., Zhang, G., Xu, Z., Katabi, D.: Me-net: Towards effective adversarial robustness with matrix estimation. In: ICML. pp. 7025–7034 (2019)
38. Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L.E., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: ICML. pp. 7472–7482 (2019)
39. Zhang, H., Dauphin, Y.N., Ma, T.: Residual learning without normalization via better initialization. In: International Conference on Learning Representations (2019)
40. Zhang, X., Li, Z., Change Loy, C., Lin, D.: Polynet: A pursuit of structural diversity in very deep networks. In: CVPR. pp. 718–726 (2017)