Reinforcing Neural Network Stability with Attractor Dynamics

Hanming Deng,[1] **Yang Hua,**[2] **Tao Song,**[*1] **Zhengui Xue,**[1]
Ruhui Ma,[1] **Neil Robertson,**[2] **Haibing Guan**[1†]
[1]Shanghai Jiao Tong University, {denghanmig, songt333, zhenguixue, ruhuima, hbguan}@sjtu.edu.cn
[2]Queen's University Belfast, {Y.Hua, N.Robertson}@qub.ac.uk

Abstract

Recent approaches interpret deep neural works (DNNs) as dynamical systems, drawing the connection between stability in forward propagation and generalization of DNNs. In this paper, we take a step further to be the first to reinforce this stability of DNNs without changing their original structure and verify the impact of the reinforced stability on the network representation from various aspects. More specifically, we reinforce stability by modeling attractor dynamics of a DNN and propose *relu-max attractor network* (RMAN), a light-weight module readily to be deployed on state-of-the-art ResNet-like networks. RMAN is only needed during training so as to modify a ResNet's attractor dynamics by minimizing an energy function together with the loss of the original learning task. Through intensive experiments, we show that RMAN-modified attractor dynamics bring a more structured representation space to ResNet and its variants, and more importantly improve the generalization ability of ResNet-like networks in supervised tasks due to reinforced stability.

1 Introduction

Numerous state-of-the-art deep learning tasks (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2015; Cho et al. 2014; Goodfellow et al. 2014; Graves 2012) are powered by successfully deployed deep neural networks (DNNs). One fundamental success of deep learning is that the large depth of DNNs provides sufficient capability to fit large and complicated datasets meanwhile they behave good generalization ability (Zhang et al. 2016). While this is counter-intuitive with the conventional theory about model complexity and generalization (Zhang et al. 2016), many new interpretations evolve to bring insights to the representation power of DNNs and provide better design principles (Zhang et al. 2016; Weinan 2017; Haber and Ruthotto 2018; Lu et al. 2018; Veit, Wilber, and Belongie 2016; Ruthotto and Haber 2018; Szegedy et al. 2013). One of these interpretations formulates DNNs as discretized differential equations, which is referred to as *dynamical systems* interpretation (Weinan 2017; Haber and Ruthotto 2018; Chang et al. 2017; Lu et al. 2018; Ciccone et al. 2018; Chen et al. 2018; Ruthotto and Haber 2018). Specifically, the residual block of ResNet (He et al. 2016) is interpreted as one step of forward Euler discretization. From the dynamical systems interpretation, the stability of a forward propagation in DNNs is related with the representation stability and the generalization ability of the network (Ciccone et al. 2018; Haber and Ruthotto 2018; Weinan 2017). In other words, a forward propagation is stable if similar inputs always converge to around some stable representations (i.e., equilibria in Ciccone et al. (Ciccone et al. 2018)) during the forward propagation, so that the output of the network is stable against small perturbations from the input. One example encountering the issue of stability is the adversarial attacks (Szegedy et al. 2013) to the DNNs, where small input perturbations lead to drastically different prediction.

Although the connection between the stability and generalization of a DNN has been established (Ciccone et al. 2018; Haber and Ruthotto 2018; Weinan 2017), how to properly model the distribution of these equilibria in a given task and quantify their impacts to the representation in a DNN are not yet answered (Bengio, Courville, and Vincent 2013). To address this, we examine the approach of *attractor dynamics* (Hopfield 1982; Kam and Cheng 1989; Bartlett and Sejnowski 1997; Wu et al. 2018), special forms of system dynamics that settle down a system from various different initial states to some steady states. These steady states that the system evolve towards are referred to as *attractors*, which can take many forms like a point or complex manifolds (Milnor 1985) [1]. As a widely existing dynamics in various dissipative system of the physical world, attractor dynamics are also used in neural networks for pattern retrieval and completion (Hopfield 1982; Kam and Cheng 1989; Bartlett and Sejnowski 1997; Wu et al. 2018) and biological memory retrieval (Poucet and Save 2005; Kanerva 1988). Compared with equilibria, attractors provide stronger convergence property. The convergence to attractors in neural networks can be directly analyzed with gradient descent through a probabilistic model (Wu et al. 2018)

[*]Tao Song is the corresponding author.

[†]This work was supported in part by National NSF of China (NO. 61872234, 61732010, 61525204) and Shanghai Key Laboratory of Scalable Computing and Systems.

[1]In this paper, we consider point attractors in the feature space.

or an energy function (Zemel and Mozer 2001).

Built upon the idea of attractor dynamics, we aim at modeling the attractor dynamics of a DNN to explore the impact of reinforced stable forward propagation on network performance. We propose a light-weight and practical module, named as *relu-max attractor network* (RMAN), which is ready to be deployed on state-of-the-art ResNet-like networks and improve their performance on supervised tasks. Firstly, we categorize the network equilibria as centers of attractors (Milnor 1985) in the feature space, which enables us to model their distribution and optimize them with gradient descent. Secondly, during training of a DNN, RMAN assigns accurate target attractor dynamically for each input through our proposed *relu-max* operation. Each target attractor can be composited by RMAN's learnable placement attractors. Thirdly, a quadratic energy function is proposed to quantify the attractor dynamics for each input. Without changing the architecture of a DNN, we can reinforce the stability of a ResNet-like network by minimizing the quadratic energy function during training. As illustrated in Figure 1, the modified attractor dynamics bring more stability in sense that not only initial states of the same class converge closer in the output feature space, but also the dynamics at the last few time steps are much smaller when the network is trained with RMAN.

By means of RMAN, we conduct extensive experiments to verify the impact of attractor dynamics on the representation of ResNet and its variants, including Neural ODE (Chen et al. 2018). Our first finding is that modifying the attractor dynamics and stability of ResNet-like networks does improve their generalization ability on supervised classification tasks. Second, through experiments on different datasets, we observe that the performance of RMAN is insensitive to the predefined number of placement attractors. Interestingly, the number of effective placement attractors that dominate the attractor dynamics are quite consistent across different network architectures on a given dataset, which suggests that the learned attractors successfully capture important factors about a dataset and thus modeling the attractor dynamics with RMAN can improve the generalization ability. Specifically, due to the structural design of RMAN, spurious placement attractors deteriorate to near zero norm, while target attractors are composited by different effective placement attractors to model more complicated dynamics. Third, we further evaluate the impact of modeling attractor dynamics on the distribution of learned representation. A noticeable effect is that RMAN significantly reduces the variance of distribution in ResNet output, suggesting that a more structured representation space is brought by RMAN. While testing on Neural ODE (Chen et al. 2018), we observe that a more complex input-output mapping is learned in order to embed attractor dynamics when training with RMAN.

To sum up, our contributions can be listed as follows:

- We propose to reinforce network stability of ResNet and its variants by modeling their attractor dynamics so that the network can learn a better distribution of stable representation.
- We design an effective yet simple module, relu-max attractor network (RMAN) to modify attractor dynamics in ResNets without changing their structure.
- Through extensive experiments, we examine the behavior of attractor dynamics in ResNets with our proposed RMAN and show reinforcing network stability with RMAN can improve network generalization ability on ResNet and its variants.

2 Background and Related Work

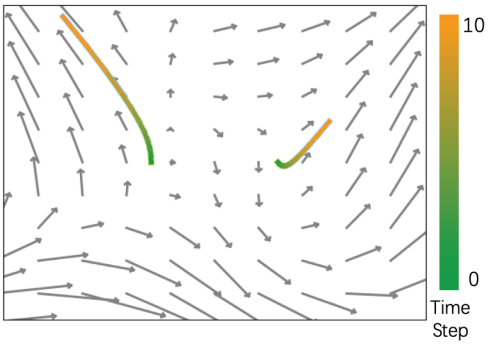
The dynamical systems perspective. Many recent approaches (Ciccone et al. 2018; Lu et al. 2018; Weinan 2017; Chang et al. 2017) interpret the forward propagation of deep neural networks as dynamical systems. Specifically, a deep neural network is considered as a discrete dynamical system that transforms different initial states (i.e., input patterns) into output state (i.e., representation) through several time steps, where the state dynamics at each time step are described by one layer of the neural network. One typical example is the ResNet-like structure that describes the transformation between states at time step t and $t+1$:

$$x_{t+1} = x_t + \Delta t f(x_t), \tag{1}$$

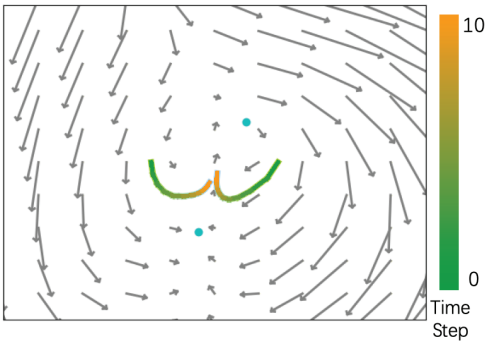
which is identical to one step of forward Euler discretization of ordinary differential equations (ODEs) $\dot{x}(t) = f(x(t))$ (Lu et al. 2018; Weinan 2017; Haber and Ruthotto 2018).

This interpretation provides a new perspective on understanding deep neural networks and sheds light on the principles of network architecture design. The relation between dynamical system and neural networks' forward pass has been analyzed in several papers (Haber and Ruthotto 2018; Ciccone et al. 2018; Lu et al. 2018; Chang et al. 2017). Weinan (Weinan 2017) proposed to view machine learning systems as continuous dynamical systems so that ideas from mathematics and physics can be leveraged. Haber and Ruthotto (Haber and Ruthotto 2018) further gave more instantiated links between ResNet and differential equations and enables stability modeling in simplified architectures. Chang et al. (Chang et al. 2017) carried lesion study on ResNet to support the dynamical systems property of deep neural networks. Lu et al. (Lu et al. 2018) drew links from the design of ResNet variants to dynamical systems, and proposed a new ResNet architecture based on the linear multi-step scheme from numerical ODEs (Ascher and Petzold 1998). Ciccone et al. (Ciccone et al. 2018) pointed out that the popular ResNet structures are autonomous and proposed a non-autonomous structure with proved stability. They showed that stable architectures have a significant reduction in generalization gap compared to the unmodified ResNet counterpart.

Attractor dynamics and attractor networks. Attractor arises from dynamical systems to describe the steady states where a system may finally settle (Milnor 1985). Attractor dynamics describe this convergence property and have been found to play an important role in robust memory retrieval (Poucet and Save 2005; Kanerva 1988). Attractor neural networks (Zemel and Mozer 2001) map inputs to steady attrac-



(a) Flow map of a trained ResNet without RMAN. The trajectories in the flow map diverge, resulting diverged distribution of representation.



(b) Flow map of a ResNet trained with RMAN. Blue points denote the placement attractors. The trajectories converge closely, resulting structured distribution of representation.

Figure 1: Flow map comparison without/with modifying attractor dynamics on a simplified ResNet. The plane denotes the 2-d feature space and each arrow denotes the dynamics of its tail position x in the feature space (the output of $f(x)$ in Eq. 1) at one time step. The detailed setup is described in Section 3. *Best viewed in color*

tor states and model the attractor dynamics into their networks. Different attractor networks are designed for associative memories (Kanerva 1988), noisy image reconstruction (Koch, Marroquin, and Yuille 1986), pattern completion (Zemel and Mozer 2001) and generative modeling (Wu et al. 2018).

In many attractor networks, the attractor dynamics are characterized by gradient descent in an energy function (Hopfield 1982; Zemel and Mozer 2001) and the steady states are where the energy is minimum (Hopfield 1982). The design of the energy function depends on the domain knowledge (Zemel and Mozer 2001). The simplest form of an energy function can be a quadratic function, because we can always approximate a smooth energy function with a quadratic function in a suitable neighborhood (Kass 1997).

Recent memory augmented neural networks (Graves, Wayne, and Danihelka 2014; Wu et al. 2018) potentiate the attractor mechanism into sequence modeling. Given a feature vector x, many approaches (Graves, Wayne, and Danihelka 2014; Wu et al. 2018; Zemel and Mozer 2001) model the attractor dynamics as follows:

$$f(x) = \Sigma_i \frac{\pi_i d(m_i, x)}{\Sigma_j \pi_i d(m_j, x)} m_i, \tag{2}$$

where m_i is referred to as center of attraction (Zemel and Mozer 2001) or memory (Graves, Wayne, and Danihelka 2014), π_i indicates the pull strength and $d(\cdot)$ describes the distance from each input x to the centers of attraction.

3 Reinforce Network Stability with Attractor Dynamics

From dynamical systems interpretation, the main concern in a supervised task is how to transform input state into subset of the feature space, so that the target task can exploit the learned representation satisfactorily (Weinan 2017). This subset can be an invariant set (Borrelli, Bemporad, and Morari 2017) to ensure a stable representation (Ciccone et al. 2018). Here, we further hypothesize that these sets are attractors (Milnor 1985), so that we can model the attractor dynamics of a deep neural network (DNN) to reinforce the stability of forward propagation. It is worth noting that, given the DNN's compositional structure and powerful approximation ability (Lin and Jegelka 2018), we aim to model the attractor dynamics of a DNN itself. Like Dropout (Srivastava et al. 2014), our module introduces no extra cost for the network during inference.

To model the attractor dynamics of a DNN, firstly, a target attractor needs to be assigned for each network output x. We propose a relu-max attractor network (RMAN) to dynamically compute a target attractor based on each x. Secondly, to induce the network to learn attractor dynamics, i.e., adjusting network weights to produce an output x closer to the target attractor, we propose to minimize a quadratic energy function together with the loss of the target task. The energy function guides each representation closer to the assigned attractor after each iteration of training.

Relu-max attractor network. The proposed relu-max attractor network g consists of h learnable *placement attractors* $m_1, \cdots, m_h \in \mathbb{R}^c$, where h is set as a hyper-parameter. Let $x_i \in \mathbb{R}^c, i = 1, \ldots, n$ denotes one feature vector at position i in a feature map of ResNet. Given x_i as an input, RMAN first evaluates the attraction strength between each placement attractor $m_j, j = 1, \cdots, h$ and x_i through *relu-max* operation. The *attraction strength* is defined as

$$a_j(x_i) = \frac{relu(m_j^T x_i)}{\Sigma_{k=0}^{h} relu(m_k^T x_i)}. \tag{3}$$

Then RMAN assigns a *target attractor* to x_i as combination of placement attractors by

$$g(x_i) = \Sigma_{j=0}^{h} a_j(x_i) m_j. \tag{4}$$

The assigned target attractor is where we want x_i to get close to, as a reflection of the attractor dynamics that we would like to model.

There are two noticeable differences between our proposed RMAN and the other attractor networks in the form of Eq. 2. The first one is that we consider the output of RMAN as the a target attractor. While in other methods (Zemel and Mozer 2001; Graves, Wayne, and Danihelka 2014), the outputs of the attractor networks are often used to modify one step of state dynamics. The second one is that we use our proposed *relu-max* in Eq. 3 instead of its common counterparts like softmax (Graves, Wayne, and Danihelka 2014) or Gaussian (Zemel and Mozer 2001). On the one hand, *relu-max* preserves better locality than softmax. This is reflected in the relu operation in Eq. 3, where if $m_j^T x_i < 0$ then the attraction strength $a_j(x_i) = 0$ and the j^{th} placement attractor has no effect when assigning a target attractor to x_i. On the other hand, softmax is monotonic, which means that for each input x_i there is a unique target attractor $g(x_i)$ if softmax is used. In contrast, different x_i can have the same $g(x_i)$ using relu-max, which is a favored property to enforce inputs of the same class to share similar target attractors.

Energy Function of Relu-Max Attractor Network. In order to model the attractor dynamics of ResNet and induce the output of ResNet to converge closer to a target attractor in the forward propagation, we minimize the following quadratic energy function with gradient descent method,

$$E(x_i) = \|g(x_i) - x_i\|^2. \tag{5}$$

Note that, the outputs of a DNN are not forced to be exactly close to placement attractor m_j. Instead, they approach the assigned target attractor $g(x_i)$, which can be composited by many placement attractors as verified by our experiments.

End-to-End Training with Relu-Max Attractor Network. In a supervised classification task like image classification, for each input image u with class c, we minimize the energy function in Eq. 5 together with the cross-entropy loss to update the network weights including placement attractors through gradient descent method,

$$l(x) = -P(c|u)log(\tilde{P}(c|u)) + \frac{\alpha}{n}\Sigma_i^n E(x_i), \tag{6}$$

where $x_1, \cdots, x_n$ are from the t^{th} layer of ResNet that RMAN is inserted and α is a scaling factor. The real attractors of a system can be very complicated and more than simple point attractors (Milnor 1985), thus we multiply energy function by a small scaling factor α in Eq. 6 to weaken its impact in forming point attractors during training. As suggested in (Haber and Ruthotto 2018), if all inputs converge to few points during forward propagation, then the essential variations between inputs are eliminated in the output, resulting in an ill-posed learning problem.

A Visualization of Learned Attractor Dynamics. An illustration of the learned dynamics is given in Figure 2 to demonstrate the impact of applying RMAN to a ResNet. Figure 2a1 to Figure 2a4 shows the dynamics of ResNet without RMAN while Figure 2b1 to Figure 2b4 shows those trained with RMAN. For illustrative purpose, we construct a simple classification dataset with two classes distributed as in Figure 2a1 and Figure 2b1, where each data point $x \in \mathbb{R}^2$ belongs to one of the classes. A simplified ResNet with 10 identical layers is used to classify the data. The t^{th} layer is set as $x_{t+1} = x_t + f(x_t)$, where $f(x) = W_1 relu(W_2^T x)$ describes the state dynamic between two consecutive layer t and $t+1$, with $W_1, W_2 \in \mathbb{R}^{2\times 16}$. The state dynamic f is shared across all layers so that we are only approximating one flow map and can thus easily split and visualize every time step of dynamics in the same feature space. The learned flow map represented by f is shown in Figure 1, where each arrow represents the output of f taken the arrow's tail coordinate as input.

In Figure 2, the sub-figure of time step t illustrates all the outputs of the t^{th} layer. We can clearly see the dynamics of each x at different time steps, where it "moves" a little according to the flow map at each time step. Although both results can be linearly classified at 10^{th} time step, with RMAN the output distribution is more structured in sense that they are more concentrated within each class and the variance of the distribution is much smaller. Note how data points flows and finally get attracted together. On the contrary, the outputs of ResNet without RMAN diverge and span a much larger space than the input distribution. This can be generalized to complex deep networks with similar architectures where multiple flow maps and more complex dynamics are learned.

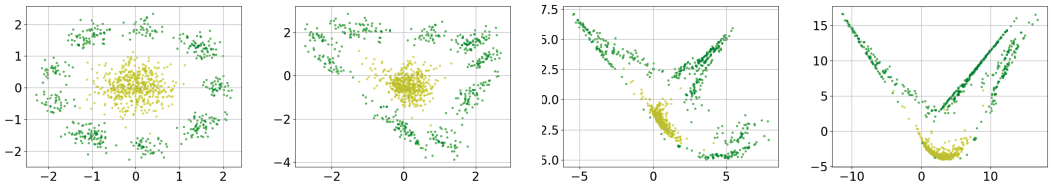
4 Experiments

4.1 Datasets and Implementation Details

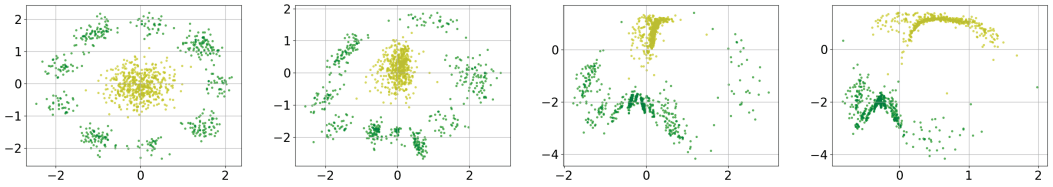
We conduct various experiments on the datasets for various tasks, including CIFAR10, CIFAR100 (Krizhevsky and Hinton 2009) for image classification and Google Commands (Warden 2017) for audio classification. On CIFAR10 and CIFAR100, we follow the data augmentation in (He et al. 2015; Lu et al. 2018), a random crop of 32×32 after a 4-pixel padding on each side of image and random horizontal flip. We use preact residual block (He et al. 2016) on ResNet and LM-ResNet implementations. For network depth $20, 32, 44, 56$, and 110, we use a two layer bn-relu-conv-bn-relu-conv structure and for networks with larger depth the bottleneck structure is adopted. Our implemented networks consist of a 3×3 conv layer, 3 residual layers, a batch normalization (Ioffe and Szegedy 2015), a global average pooling and a fully connected layer. We train all the networks using SGD with a batch size of 128 and momentum of 0.9. The learning rate starts with 0.1 and is devided by 10 at epoch 80/150, 120/225 and training terminates at epoch 160/300 on CIFAR10/CIFAR100, respectively. We apply a weight decay of 0.0005 for all networks. For all our experiments, we insert RMAN after the global average pooling layer. We set the scaling factor $\alpha = 0.0001$ and increase it by 0.0001 every epoch. The placement attractors are initialized with stand normal distribution.

4.2 Overall Performance on Classification Tasks and Ablation Study

Overall performance. The main results of testing with different numbers of placement attractors on different datasets and network architectures are presented in Table 1. Each experiment is tested three times and their mean and standard deviation are shown. We have tested on ResNet



(a1) Time step 0 (a2) Time step 2 (a3) Time step 6 (a4) Time step 10



(b1) Time step 0 (b2) Time step 2 (b3) Time step 6 (b4) Time step 10

Figure 2: Illustration of state dynamics in ResNet without/with RMAN. The upper row shows the results of ResNet without RMAN while the bottom row demonstrates the results of ResNet with RMAN. The time step t represents the outputs of the t^{th} layer. Specifically, time step 0 represents the input data. A 10-layer simplified ResNet is used, as described in Section 3. *Best viewed in color*

(He et al. 2016), LM-ResNet (Lu et al. 2018) and ResNext (Xie et al. 2017) of various depths. LM-ResNet is a ResNet variant that is directly inspired by linear multi-step method for solving differential equations (Ascher and Petzold 1998). The results show that RMAN can improve the classification accuracy. This suggests that RMAN modified attractor dynamics and reinforced stability indeed improve the performance of a deep neural network.

The average training/testing loss with/without RMAN are shown in Figure 3a. The data are collected from ResNet-44, and the RMAN results with different number of placement attractors are all shown. With RMAN, a better generalization gap is achieved where the gap between training loss (dotted curve) and testing loss (solid curve) is smaller than without RMAN, suggesting a better generalization gap. This is also observed on all the other models in Table 1. This shows that the reinforced network stability based on attractor dynamics is favourable for the network performance and it can improve the network's generalization ability.

Number of placement attractors. One thing worth noticing is that the performance gain is similar for different number of placement attractors as presented in Table 1. We attribute this to our design of RMAN in Eq. 3, where relu filters out attraction between a pair of placement attractor m_j and network output x_i whose dot product $m_j^T x_i < 0$. This leads to the result that spurious attractors deteriorate to zero norm with training going on and become ineffective in assigning target attractor, as represented in Table 2, while remaining placement attractors are able to assign sufficient target attractors for modeling attractor dynamics that leads to a performance gain.

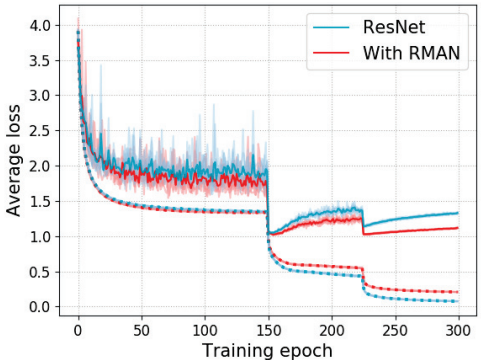
Most importantly, only small but constant number of effective placement attractor is needed in modeling attractor dynamics on a given dataset, as shown in Table 2. We define a placement attractor effective when its norm $\|m_i\|^2 > 0.01$ after training. The number of effective placement attractors is almost identical under different network architectures and different numbers of predefined placement attractors. In other words, these effective placement attractors capture important factors about the dataset, as also can be revealed from the difference in the average number of effective attractors between CIFAR10 and CIFAR100 datasets. On the one hand, this accords with the previous observation that distribution of these attractors embeds the knowledge from the dataset (Sudharsanan and Sundareshan 1991). On the other hand, this observation also suggests that while the attractor dynamics bring stability to the neural network, the end-to-end training of these attractors is also stable.

4.3 Evaluation of Learned Attractor Dynamics and Stability

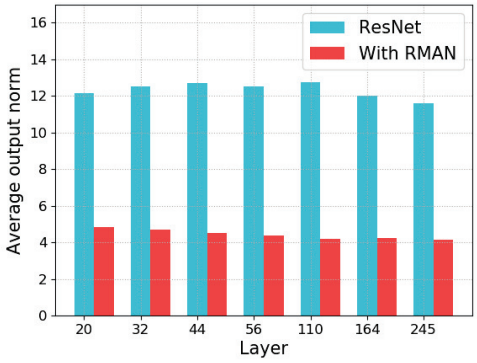
Interactions between learned placement attractors. While the number of effective placement attractors after training is almost constant, we find that their spatial relations are also similar across different network architectures and different number of placement attractors. Let vector m_j and m_k denote two effective placement attractors, in all our trained networks, we have $m_j^T m_k < 0$ except when $i = j$. Figure 4 illustrates this relationship between effective placement attractors, where different colors of circle at position (i, j) denote the value of dot product between m_i and m_j. We can see that dot products between different effective placement attractors are all below zero. This suggests that, if the output of ResNet x_i is close to any effective placement attractor m_j then it is likely to be affected solely by this attractor. Due to the relu operation used in Eq. 3, if $x_i \approx m_j$, then it is likely that $relu(m_k^T x_i) = 0$ for any $k \neq j$, which isolates x_i from the attraction of other placement attractors.

Dataset	Model	Layer	Number of Placement Attractors			
			h=0	16	64	256
CIFAR10	ResNet	20	$7.95_{\pm 0.08}$	$7.90_{\pm 0.05}$	$\mathbf{7.61}_{\pm 0.07}$	$7.68_{\pm 0.20}$
	ResNet	32	$6.95_{\pm 0.16}$	$\mathbf{6.50}_{\pm 0.17}$	$6.71_{\pm 0.16}$	$6.65_{\pm 0.07}$
CIFAR100	ResNet	44	$28.16_{\pm 0.21}$	$27.87_{\pm 0.09}$	$28.02_{\pm 0.12}$	$\mathbf{27.57}_{\pm 0.12}$
	ResNet	56	$27.28_{\pm 0.16}$	$27.12_{\pm 0.15}$	$26.90_{\pm 0.06}$	$\mathbf{26.67}_{\pm 0.37}$
	LM-ResNet	110	$25.46_{\pm 0.20}$	$24.76_{\pm 0.15}$	$24.58_{\pm 0.29}$	$\mathbf{24.36}_{\pm 0.36}$
	LM-ResNet	164	$22.44_{\pm 0.15}$	$22.02_{\pm 0.07}$	$22.09_{\pm 0.14}$	$\mathbf{21.86}_{\pm 0.18}$
	ResNext	29 (8x64d)	$19.59_{\pm 0.10}$	$18.33_{\pm 0.08}$	$18.43_{\pm 0.02}$	$\mathbf{18.22}_{\pm 0.11}$
	ResNext	29 (16x64d)	$18.64_{\pm 0.06}$	$17.62_{\pm 0.06}$	$17.45_{\pm 0.04}$	$\mathbf{17.38}_{\pm 0.02}$

Table 1: Comparison testing error (%) on different ResNet-like networks without/with RMAN and different number of placement attractors, where $h = 0$ denotes networks trained without RMAN.



(a) Generalization gap of ResNet-44. The gap between training (dotted curve) and testing (solid curve) loss of ResNet with RMAN (red curve) is smaller than ResNet without RMAN (blue curve), showing the modified attractor dynamics improve the generalization.



(b) The norm of ResNet outputs from the global average pooling layer on test set. For networks with different depths, there all exist a clear gap between the output norms with/without applying RMAN.

Figure 3: Illustration of the Generalization gap and the norm of ResNets outputs trained without/with RMAN on CIFAR100.

Impact of attractor dynamics on the output distribution. In Figure 4, the radius of each circle on the diagonal is equal to the average attraction strength $\Sigma_{i=0}^{n} a_j(x_i)/n$ over all n outputs of a ResNet on a dataset, where maximum radius is 1 and denotes the maximum attraction that all outputs are attracted solely by this placement attractor. For other circles off the diagonal, the radius denotes the average attraction strength over outputs x that are mostly attracted by both attractors i and j. First, the sum of the radius on the diagonal is approximately one, suggesting that almost all outputs are attracted by these effective placement attractors. Second, the variations of radius of circles off the diagonal show that many target attractors are composited by at least two effective placement attractors. This implies that the six effective placement attractors assign various and more complex distributed target attractors to capture the dynamics of ResNet on CIFAR100, which explains why with only six effective placement attractors RMAN can improve the performance of ResNet. Another observation to support the complex distribution of the target attractor is that among all the outputs on CIFAR100, only around 4% of outputs x_i are solely attracted (i.e., $a_j(x_i) > 0.9$) by one placement attractor m_j.

Figure 3b illustrates the impact of RMAN on the output norms of ResNet on CIFAR100. The output of different depths of ResNets in Table 1 are shown here. We display the norm of the output of global average pooling. A clear gap can be observed with/without RMAN. RMAN brings significantly variance reduction in the distribution of the output as the norms are now much smaller, suggesting that the distribution is more concentrated around the origin. We compare the norms because we find that the means of these outputs are all around $\mathbf{0}$, thus the norms reveal a rough distribution of these outputs since it is roughly the variance of each component in the outputs. This effect is more intuitive on the 2-d dataset in Figure 2.

Dataset	Model	Number of Placement Attractors						Avg.
		h=16	32	64	128	256	512	
CIFAR10	ResNet	5.25	5.5	6	5.25	6	5.75	5.63
CIFAR100	ResNet	5.71	5.57	6	6.29	6	6.57	6.07
	LM-ResNet	6.67	6	5.67	5.67	6	6.67	
	ResNext	6	6	6.5	6.5	5.5	7	

Table 2: Average number of effective placement attractors after training. Effective placement attractors have $\|m\|^2 > 0.01$. The number of effective placement attractors is almost identical under different network architectures and different predefined number of placement attractors.

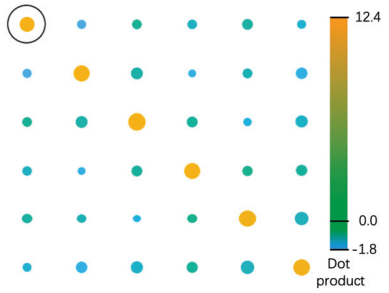


Figure 4: The color of circle on position (i, j) shows the dot product $m_i^T m_j$ between the i^{th} and j^{th} effective placement attractors. The radius of circle on position (i, j) shows the average attraction of each effective attractor (or combination of i^{th} and j^{th} attractor) to all network outputs. A reference for the size of unit radius (i.e., maximum attraction) is given at $(1, 1)$. The data is gathered on CIFAR100 test set using ResNet-110 trained with RMAN (256 placement attractors), where there are six effective placement attractors after training. *Best viewed in color*

Approximated function complexity. We apply RMAN on Neural ODE (Chen et al. 2018) and present the results in Table 3. Neural ODE (Chen et al. 2018) is a radically new design of neural network based on the idea of continuous ODEs. The conventional stacked layer structure of a neural network, analog to multiple discrete time steps of ODEs, is replaced with an ODE solver (Ascher and Petzold 1998) to directly approximate a continuous differential equation. Modern ODE solvers used in Neural ODE can monitor the approximation error and adapt the number of function evaluations (NFE) to achieve requested approximation accuracy. We also monitor NFE as reference to the function complexity approximated by the Neural ODE with/without modifying attractor dynamics. As shown in Table 3, after applying RMAN, NFE increases by 8.1 and 8.0 on CIFAR10 and CIFAR100 dataset, respectively. This suggests that a more complex function (i.e., more complicated dynamics) is learned by the Neural ODE to embed attractor dynamics when RMAN is applied. With RMAN in a supervised classification, it requires a network's final representation not only to be linearly separable, but also more structured because of the modified attractor dynamics, which lead to a more complicated trajectory and improved accuracy.

Dataset	Model	Num. of Attractor	Error (%)	NFE
CIFAR10	ODE-NET	0	12.09	26.3
	ODE-NET	8	11.64	32.4
CIFAR100	ODE-NET	0	41.06	26.4
	ODE-NET	16	40.54	32.4

Table 3: Performance of Neural ODE (Chen et al. 2018) without/with RMAN. Trained with RMAN, the number of function evaluations (NFE) increases—a more complex function is learned to embed attractor dynamics.

Layer	Number of Placement Attractors			
	h=0	16	64	256
18	$4.06_{\pm 0.07}$	$3.92_{\pm 0.01}$	$3.89_{\pm 0.01}$	$\mathbf{3.88}_{\pm 0.07}$
34	$4.02_{\pm 0.12}$	$3.87_{\pm 0.06}$	$3.86_{\pm 0.08}$	$\mathbf{3.78}_{\pm 0.06}$

Table 4: Testing result on Google Command dataset. Comparison testing error (%) on ResNets without/with RMAN and different number of placement attractors.

Experiments on audio recognition dataset. To examine the generalization of our method, we perform experiments on Google Commands (Warden 2017), a speech classification dataset. It contains 30 classes of utterances, corresponding to 30 classes of voice commands spoken pronounced different speakers. Following (Zhang et al. 2017), we extract normalized spectrogram from the waveform of each utterance at a sampling rate of 16 kHz and zero-pad each spectrogram to size 160×101. The result is shown in Table 4, where each experiment is tested three times and their mean and standard deviation are shown.

5 Conclusion

In this paper, we propose a novel relu-max attractor network (RMAN) to model the attractor dynamics to reinforce system stability of a deep neural network. RMAN is ready to be deployed on the state-of-the-art ResNets-like network and it is only required during training stage. We conduct extensive experiments to demonstrate that reinforcing network stability can improve network generalization ability on ResNet and its variants.

References

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

Bartlett, M. S., and Sejnowski, T. J. 1997. Viewpoint invariant face recognition using independent component analysis and attractor networks. In *NeurIPS*, 817–823.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8):1798–1828.

Borrelli, F.; Bemporad, A.; and Morari, M. 2017. *Predictive control for linear and hybrid systems*. Cambridge University Press.

Chang, B.; Meng, L.; Haber, E.; Tung, F.; and Begert, D. 2017. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *NeurIPS*, 6571–6583.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Ciccone, M.; Gallieri, M.; Masci, J.; Osendorfer, C.; and Gomez, F. 2018. Nais-net: stable deep networks from non-autonomous differential equations. In *NeurIPS*, 3025–3035.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPS*, 2672–2680.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Haber, E., and Ruthotto, L. 2018. Stable architectures for deep neural networks. *Inverse Problems* 34(1).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 1026–1034.

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

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79(8):2554–2558.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kam, M., and Cheng, R. 1989. Convergence and pattern-stabilization in the boltzmann machine. In *NeurIPS*, 511–518.

Kanerva, P. 1988. *Sparse distributed memory*. MIT press.

Kass, M. 1997. An introduction to physically based modeling: an introduction to continuum dynamics for computer graphics. *Computer Graphics Tutorial, ACM SIGRAPH* 60–66.

Koch, C.; Marroquin, J.; and Yuille, A. 1986. Analog "neuronal" networks in early vision. *Proceedings of the National Academy of Sciences* 83(12):4263–4267.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 1097–1105.

Lin, H., and Jegelka, S. 2018. Resnet with one-neuron hidden layers is a universal approximator. In *NeurIPS*, 6169–6178.

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

Milnor, J. 1985. On the concept of attractor. In *The theory of chaotic attractors*. Springer. 243–264.

Poucet, B., and Save, E. 2005. Attractors in memory. *Science* 308(5723):799–800.

Ruthotto, L., and Haber, E. 2018. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Sudharsanan, S. I., and Sundareshan, M. K. 1991. Equilibrium characterization of dynamical neural networks and a systematic synthesis procedure for associative memories. *IEEE Transactions on Neural Networks* 2(5):509–521.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Veit, A.; Wilber, M. J.; and Belongie, S. 2016. Residual networks behave like ensembles of relatively shallow networks. In *NeurIPS*, 550–558.

Warden, P. 2017.

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

Wu, Y.; Wayne, G.; Gregor, K.; and Lillicrap, T. 2018. Learning attractor dynamics for generative memory. In *NeurIPS*, 9379–9388.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*, 1492–1500.

Zemel, R. S., and Mozer, M. C. 2001. Localist attractor networks. *Neural Computation* 13(5):1045–1064.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.