Noisy Recurrent Neural Networks

Soon Hoe Lim
Nordita, KTH Royal Institute of Technology
and Stockholm University
soon.hoe.lim@su.se

N. Benjamin Erichson
University of Pittsburgh
School of Engineering
erichson@pitt.edu

Liam Hodgkinson
ICSI and Department of Statistics
UC Berkeley
liam.hodgkinson@berkeley.edu

Michael W. Mahoney
ICSI and Department of Statistics
UC Berkeley
mmahoney@stat.berkeley.edu

Abstract

We provide a general framework for studying recurrent neural networks (RNNs) trained by injecting noise into hidden states. Specifically, we consider RNNs that can be viewed as discretizations of stochastic differential equations driven by input data. This framework allows us to study the implicit regularization effect of general noise injection schemes by deriving an approximate explicit regularizer in the small noise regime. We find that, under reasonable assumptions, this implicit regularization promotes flatter minima; it biases towards models with more stable dynamics; and, in classification tasks, it favors models with larger classification margin. Sufficient conditions for global stability are obtained, highlighting the phenomenon of stochastic stabilization, where noise injection can improve stability during training. Our theory is supported by empirical results which demonstrate that the RNNs have improved robustness with respect to various input perturbations.

1 Introduction

Viewing recurrent neural networks (RNNs) as discretizations of ordinary differential equations (ODEs) driven by input data has recently gained attention [7, 27, 16, 49]. The "formulate in continuous time, and then discretize" approach [38] motivates novel architecture designs before experimentation, and it provides a useful interpretation as a dynamical system. This, in turn, has led to gains in reliability and robustness to data perturbations.

Recent efforts have shown how adding noise can also improve stability during training, and consequently improve robustness [35]. In this work, we consider discretizations of the corresponding stochastic differential equations (SDEs) obtained from ODE formulations of RNNs through the addition of a diffusion (noise) term. We refer to these as *Noisy RNNs* (NRNNs). By dropping the noisy elements at inference time, NRNNs become a stochastic learning strategy which, as we shall prove, has a number of important benefits. In particular, stochastic learning strategies (including dropout) are often used as natural regularizers, favoring solutions in regions of the loss landscape with desirable properties (often improved generalization and/or robustness). This mechanism is commonly referred to as *implicit regularization* [40, 39, 50], differing from *explicit regularization* where the loss is explicitly modified. For neural network (NN) models, implicit regularization towards wider minima is conjectured to be a prominent ingredient in the success of stochastic optimization [67, 28]. Indeed, implicit regularization has been linked to increases in classification margins [47], which can lead to improved generalization performance [51]. A common approach to identify and study implicit regularization is to approximate the implicit regularization by an appropriate explicit regularizer

[40, 39, 1, 6, 21]. Doing so, we will see that NRNNs favor wide minima (like SGD); more stable dynamics; and classifiers with a large classification margin, keeping generalization error small.

SDEs have also seen recent appearances in *neural SDEs* [59, 24], stochastic generalizations of *neural ODEs* [9] which can be seen as an analogue of NRNNs for non-sequential data, with a similar relationship to NRNNs as feedforward NNs do to RNNs. They have been shown to be robust in practice [35]. Analogously, we shall show that the NRNN framework leads to more reliable and robust RNN classifiers, whose promise is demonstrated by experiments on benchmark data sets.

Contributions. For the class of NRNNs (formulated first as a continuous-time model, which is then discretized), the following are our main contributions:

- we identify the form of the implicit regularization for NRNNs through a corresponding (data-dependent) explicit regularizer in the small noise regime (see Theorem 1);
- we focus on its effect in classification tasks, providing bounds for the classification margin for the deterministic RNN classifiers (see Theorem 2); in particular, Theorem 2 reveals that *stable RNN dynamics can lead to large classification margin*;
- we show that noise injection can also lead to improved stability (see Theorem 3) via a Lyapunov stability analysis of continuous-time NRNNs;
- we demonstrate via empirical experiments on benchmark data sets that NRNN classifiers are more robust to data perturbations when compared to other recurrent models, while retaining state-of-the-art performance for clean data. Research code is provided here: `https://github.com/erichson/NoisyRNN`.

Notation. We use $\|v\| := \|v\|_2$ to denote the Euclidean norm of the vector v, and $\|A\|_2$ and $\|A\|_F$ to denote the spectral norm and Frobenius norm of the matrix A, respectively. The ith element of a vector v is denoted by v^i or $[v]^i$, and the (i, j)-entry of a matrix A by A^{ij} or $[A]^{ij}$. For a vector $v = (v^1, \dots, v^d)$, diag(v) denotes the diagonalization of v with $\mathrm{diag}(v)^{ii} = v^i$. I denotes the identity matrix (with dimension clear from context), while superscript T denotes transposition. For a matrix M, $M^{\mathrm{sym}} = (M + M^T)/2$ denotes its symmetric part, $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ denote its minimum and maximum eigenvalue respectively, $\sigma_{\max}(M)$ denotes its maximum singular value, and $Tr(M)$ denotes its trace. For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ such that each of its first-order partial derivatives (with respect to x) exist, $\frac{\partial f}{\partial x} \in \mathbb{R}^{m \times n}$ is the Jacobian matrix of f. For a scalar-valued function $g : \mathbb{R}^n \to \mathbb{R}$, $\nabla_h g$ is the gradient of g with respect to the variable $h \in \mathbb{R}^n$ and $H_h g$ is the Hessian of g with respect to h.

2 Related Work

Dynamical Systems and Machine Learning. There are various interesting connections between machine learning and dynamical systems. Formulating machine learning in the framework of continuous-time dynamical systems was recently popularized by [62]. Subsequent efforts focus on constructing learning models by approximating continuous-time dynamical systems [9, 29, 48] and studying them using tools from numerical analysis [36, 64, 69, 68]. On the other hand, dynamical systems theory provides useful theoretical tools for analyzing NNs, including RNNs [60, 15, 34, 7, 16], and useful principles for designing NNs [23, 54]. Other examples of dynamical systems inspired models include the learning of invariant quantities via their Hamiltonian or Lagrangian representations [37, 22, 10, 71, 58]. Another class of models is inspired by Koopman theory, yielding models where the evolution operator is linear [56, 43, 17, 45, 33, 4, 3, 13].

Stochastic Training and Regularization Strategies. Regularization techniques such as noise injection and dropout can help to prevent overfitting in NNs. Following the classical work [5] that studies regularizing effects of noise injection on data, several work studies the effects of noise injection into different parts of networks for various architectures [25, 44, 35, 54, 2, 26, 66, 61]. In particular, recently [6] studies the regularizing effect of isotropic Gaussian noise injection into the layers of feedforward networks. For RNNs, [12] shows that noise additions on the hidden states outperform Bernoulli dropout in terms of performance and bias, whereas [18] introduces a variant of stochastic RNNs for generative modeling of sequential data. Some specific formulations of RNNs as SDEs were also considered in Chapter 10 of [41]. Implicit regularization has also been studied more generally than stochastic gradient based training of NNs [40, 39, 19, 11].

3 Noisy Recurrent Neural Networks

We formulate continuous-time recurrent neural networks (CT-RNNs) at full generality as a system of input-driven ODEs: for a terminal time $T > 0$ and an input signal $x = (x_t)_{t\in[0,T]} \in C([0,T];\mathbb{R}^{d_x})$, the output $y_t \in \mathbb{R}^{d_y}$, for $t \in [0,T]$, is a linear map of hidden states $h_t \in \mathbb{R}^{d_h}$ satisfying

$$\mathrm{d}h_t = f(h_t, x_t)\mathrm{d}t, \qquad y_t = Vh_t, \tag{1}$$

where $V \in \mathbb{R}^{d_y\times d_h}$, and $f : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}$ is typically Lipschitz continuous, guaranteeing existence and uniqueness of solutions to (1).

A natural stochastic variant of CT-RNNs arises by replacing the ODE in (1) by an Itô SDE, that is,

$$\mathrm{d}h_t = f(h_t, x_t)\mathrm{d}t + \sigma(h_t, x_t)\mathrm{d}B_t, \qquad y_t = Vh_t, \tag{2}$$

where $\sigma : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_h\times r}$ and $(B_t)_{t\geq 0}$ is an r-dimensional Brownian motion. The functions f, σ are referred to as the *drift* and *diffusion* coefficients, respectively. Intuitively, (2) amounts to a noisy perturbation of the corresponding deterministic CT-RNN (1). At full generality, we refer to the system (2) as a *continuous-time Noisy RNN* (CT-NRNN). To guarantee the existence of a unique solution to (2), in the sequel, we assume that $\{f(\cdot, x_t)\}_{t\in[0,T]}$ and $\{\sigma(\cdot, x_t)\}_{t\in[0,T]}$ are uniformly Lipschitz continuous, and $t \mapsto f(h, x_t)$, $t \mapsto \sigma(h, x_t)$ are bounded in $t \in [0,T]$ for each fixed $h \in \mathbb{R}^{d_h}$. For further details, see Section B in Supplementary Material (**SM**).

While much of our theoretical analysis will focus on this general formulation of CT-NRNNs, our empirical and stability analyses focus on the choice of drift function

$$f(h, x) = Ah + a(Wh + Ux + b), \tag{3}$$

where $a : \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous scalar activation function extended to act on vectors pointwise, $A, W \in \mathbb{R}^{d_h\times d_h}$, $U \in \mathbb{R}^{d_h\times d_x}$ and $b \in \mathbb{R}^{d_h}$. Typical examples of activation functions include $a(x) = \tanh(x)$. The matrices A, W, U, V, b are all assumed to be trainable parameters. This particular choice of drift dates back to the early Cohen-Grossberg formulation of CT-RNNs, and was recently reconsidered in [16].

3.1 Noise Injections as Stochastic Learning Strategies

While precise choices of drift functions f are the subject of existing deterministic RNN theory, good choices of the diffusion coefficient σ are less clear. Here, we shall consider a parametric class of diffusion coefficients given by:

$$\sigma(h, x) \equiv \epsilon(\sigma_1 I + \sigma_2 \mathrm{diag}(f(h, x))), \tag{4}$$

where the noise level $\epsilon > 0$ is small, and $\sigma_1 \geq 0$ and $\sigma_2 \geq 0$ are tunable parameters describing the relative strength of additive noise and a multiplicative noise respectively.

While the stochastic component is an important part of the model, one can set $\epsilon \equiv 0$ at inference time. In doing so, noise injections in NRNNs may be viewed as a learning strategy. A similar stance is considered in [35] for treating neural SDEs. From this point of view, we may relate noise injections generally to regularization mechanisms considered in previous works. For example, additive noise injection was studied in the context of feedforward NNs in [6], in which case a Gaussian noise is injected to the activation function at each layer of the NN. Furthermore, multiplicative noise injections includes stochastic depth and dropout strategies as special cases [36, 35]. By taking a Gaussian approximation to Bernoulli noise and taking a continuous-time limit, NNs with stochastic dropout can be weakly approximated by an SDE with appropriate multiplicative noise, see [36]. All of these works highlight various advantages of noise injection for training NNs.

3.2 Numerical Discretizations

As in the deterministic case, exact simulation of the SDE in (2) is infeasible in practice, and so one must specify a numerical integration scheme. We will focus on the explicit Euler-Maruyama (E-M) integrators [30], which are the stochastic analogues of Euler-type integration schemes for ODEs.

Let $0 := t_0 < t_1 < \cdots < t_M := T$ be a partition of the interval $[0, T]$. Denote $\delta_m := t_{m+1} - t_m$ for each $m = 0, 1, \ldots, M-1$, and $\delta := (\delta_m)$. The E-M scheme provides a family (parametrized by δ) of approximations to the solution of the SDE in (2):

$$h_{m+1}^{\delta} = h_m^{\delta} + f(h_m^{\delta}, \hat{x}_m)\delta_m + \sigma(h_m^{\delta}, \hat{x}_m)\sqrt{\delta_m}\xi_m, \tag{5}$$

for $m = 0, 1, \ldots, M-1$, where $(\hat{x}_m)_{m=0,\ldots,M-1}$ is a given sequential data, the $\xi_m \sim \mathcal{N}(0, I)$ are independent r-dimensional standard normal random vectors, and $h_0^{\delta} = h_0$. As $\Delta := \max_m \delta_m \to 0$, the family of approximations (h_m^{δ}) converges strongly to the Itô process (h_t) satisfying (2) (at rate $\mathcal{O}(\sqrt{\Delta})$ when the step sizes are uniform; see Theorem 10.2.2 in [30]). See Section C in **SM** for details on the general case.

4 Implicit Regularization

To highlight the advantages of NRNNs over their deterministic counterpart, we show that, under reasonable assumptions, NRNNs exhibit a natural form of *implicit regularization*. By this, we mean regularization imposed implicitly by the stochastic learning strategy, without explicitly modifying the loss, but that, e.g., may promote flatter minima. Our goal is achieved by deriving an appropriate explicit regularizer through a perturbation analysis in the small noise regime. This becomes useful when considering NRNNs as a learning strategy, since we can precisely determine the effect of the noise injection as a regularization mechanism.

The study for discrete-time NRNNs is of practical interest and is our focus here. Nevertheless, analogous results for continuous-time NRNNs are also valuable for exploring other discretization schemes. For this reason, we also study the continuous-time case in Section E in **SM**. Our analysis covers general NRNNs, not necessarily those with the drift term (3) and diffusion term (4), that satisfy the following assumption, which is typically reasonable in practice. We remark that a ReLU activation will violate the assumption. However, RNNs with ReLU activation are less widely used in practice. Without careful initialization [31, 57], they typically suffer more from exploding gradient problems compared to those with bounded activation functions such as $\tanh$.

Assumption A. The drift f and diffusion coefficient σ of the SDE in (2) satisfy the following:

(i) for all $t \in [0, T]$ and $x \in \mathbb{R}^{d_x}$, $h \mapsto f(h, x)$ and $h \mapsto \sigma^{ij}(h, x)$ have Lipschitz continuous partial derivatives in each coordinate up to order three (inclusive);

(ii) for any $h \in \mathbb{R}^{d_h}$, $t \mapsto f(h, x_t)$ and $t \mapsto \sigma(h, x_t)$ are bounded and Borel measurable on $[0, T]$.

We consider a rescaling of the noise $\sigma \mapsto \epsilon\sigma$ in (2), where $\epsilon > 0$ is assumed to be a small parameter, in line with our noise injection strategies in Subsection 3.1.

In the sequel, we let $\bar{h}_m^{\delta}$ denote the hidden states of the corresponding deterministic RNN model, satisfying

$$\bar{h}_{m+1}^{\delta} = \bar{h}_m^{\delta} + \delta_m f(\bar{h}_m^{\delta}, \hat{x}_m), \quad m = 0, 1, \ldots, M-1, \tag{6}$$

with $\bar{h}_0^{\delta} = h_0$. Let $\Delta := \max_{m \in \{0,\ldots,M-1\}} \delta_m$, and denote the state-to-state Jacobians by

$$\hat{J}_m = I + \delta_m \frac{\partial f}{\partial h}(\bar{h}_m^{\delta}, \hat{x}_m). \tag{7}$$

For $m, k = 0, \ldots, M-1$, also let

$$\hat{\Phi}_{m,k} = \hat{J}_m \hat{J}_{m-1} \cdots \hat{J}_k, \tag{8}$$

where the empty product is assumed to be the identity. Note that the $\hat{\Phi}_{m,k}$ are products of the state-to-state Jacobian matrices, important for analyzing signal propagation in RNNs [8]. For the sake of brevity, we denote $f_m = f(\bar{h}_m^{\delta}, \hat{x}_m)$ and $\sigma_m = \sigma(\bar{h}_m^{\delta}, \hat{x}_m)$ for $m = 0, 1, \ldots, M$.

The following result, which is our first main result, relates the loss function, averaged over realizations of the injected noise, used for training NRNN to that for training deterministic RNN in the small noise regime.

Theorem 1 (Implicit regularization induced by noise injection)**.** *Under Assumption A,*

$$\mathbb{E}\ell(h_M^{\delta}) = \ell(\bar{h}_M^{\delta}) + \frac{\epsilon^2}{2}[\hat{Q}(\bar{h}^{\delta}) + \hat{R}(\bar{h}^{\delta})] + \mathcal{O}(\epsilon^3), \tag{9}$$

as $\epsilon \to 0$, where the terms $\hat{Q}$ and $\hat{R}$ are given by

$$\hat{Q}(\bar{h}^\delta) = \nabla l(\bar{h}_M^\delta)^T \sum_{k=1}^{M} \delta_{k-1} \hat{\Phi}_{M-1,k} \sum_{m=1}^{M-1} \delta_{m-1} \boldsymbol{v}_m, \tag{10}$$

$$\hat{R}(\bar{h}^\delta) = \sum_{m=1}^{M} \delta_{m-1} \mathrm{tr}(\sigma_{m-1}^T \hat{\Phi}_{M-1,m}^T H_{\bar{h}^\delta} l \, \hat{\Phi}_{M-1,m} \sigma_{m-1}), \tag{11}$$

with $\boldsymbol{v}_m$ a vector with the pth component:

$$[v_m]^p = \mathrm{tr}(\sigma_{m-1}^T \hat{\Phi}_{M-2,m}^T H_{\bar{h}^\delta} [f_M]^p \hat{\Phi}_{M-2,m} \sigma_{m-1}), \tag{12}$$

for $p = 1, \ldots, d_h$. Moreover,

$$|\hat{Q}(\bar{h}^\delta)| \leq C_Q \Delta^2, \;\; |\hat{R}(\bar{h}^\delta)| \leq C_R \Delta, \tag{13}$$

for $C_Q, C_R > 0$ independent of Δ.

If the loss is convex, then $\hat{R}$ is non-negative, but $\hat{Q}$ needs not be. However, $\hat{Q}$ can be made negligible relative to $\hat{R}$ provided that Δ is taken sufficiently small. This also ensures that the E-M approximations are accurate.

To summarize, Theorem 1 implies that the injection of noise into the hidden states of deterministic RNN is, on average, approximately equivalent to a regularized objective functional. Moreover, the explicit regularizer is solely determined by the discrete-time flow generated by the Jacobians $\frac{\partial f_m}{\partial h}(\bar{h}_m^\delta)$, the diffusion coefficients σ_n, and the Hessian of the loss function, all evaluated along the dynamics of the deterministic RNN. We can therefore expect that the use of NRNNs as a regularization mechanism should reduce the state-to-state Jacobians and Hessian of the loss function according to the noise level ϵ. Indeed, NRNNs exhibit a smoother Hessian landscape than that of the deterministic counterpart (see Figure 3 in **SM**).

The Hessian of the loss function commonly appears in implicit regularization analyses, and suggests a preference towards wider minima in the loss landscape. Commonly considered a positive attribute [28], this, in turn, suggests a degree of robustness in the loss to perturbations in the hidden states [65]. More interesting, however, is the appearance of the Jacobians, which is indicative of a preference towards slower, more stable dynamics. Both of these attributes suggest NRNNs could exhibit a strong tendency towards models which are less sensitive to input perturbations. Overall, we can see that the use of NRNNs as a regularization mechanism reduces the state-to-state Jacobians and Hessian of the loss function according to the noise level.

5 Implications in Classification Tasks

Our focus now turns to an investigation of the benefits of NRNNs over their deterministic counterparts for classification tasks. From Theorem 1, it is clear that adding noise to deterministic RNN implicitly regularizes the state-to-state Jacobians. Here, we show that doing so also enhances an implicit tendency towards classifiers with large classification margin. Our analysis here covers general deterministic RNNs, although we also apply our results to obtain explicit expressions for Lipschitz RNNs.

Let $\mathcal{S}_N$ denote a set of training samples $s_n := (\boldsymbol{x}_n, y_n)$ for $n = 1, \ldots, N$, where each input sequence $\boldsymbol{x}_n = (x_{n,0}, x_{n,1}, \ldots, x_{n,M-1}) \in \mathcal{X} \subset \mathbb{R}^{d_x M}$ has a corresponding class label $y_n \in \mathcal{Y} = \{1, \ldots, d_y\}$. Following the statistical learning framework, these samples are assumed to be independently drawn from an underlying probability distribution μ on the sample space $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$. An RNN-based classifier $g^\delta(\boldsymbol{x})$ is constructed in the usual way by taking

$$g^\delta(\boldsymbol{x}) = \mathrm{argmax}_{i=1,\ldots,d_y} p^i(V \bar{h}_M^\delta[\boldsymbol{x}]), \tag{14}$$

where $p^i(x) = e^{x^i} / \sum_j e^{x^j}$ is the softmax function. Letting ℓ denoting the cross-entropy loss, such a classifier is trained from $\mathcal{S}_N$ by minimizing the empirical risk (training error), $\mathcal{R}_N(g^\delta) := \frac{1}{N} \sum_{n=1}^{N} \ell(g^\delta(\boldsymbol{x}_n), y_n)$, as a proxy for the true (population) risk (test error), $\mathcal{R}(g^\delta) =$

$\mathbb{E}_{(\boldsymbol{x},y)\sim\mu}\ell(g^\delta(\boldsymbol{x}),y)$, with $(\boldsymbol{x},y)\in\mathcal{S}$. The measure used to quantify the prediction quality is the generalization error (or estimation error), which is the difference between the empirical risk of the classifier on the training set and the true risk: $\mathrm{GE}(g^\delta) := |\mathcal{R}(g^\delta) - \mathcal{R}_N(g^\delta)|$.

The classifier is a function of the output of the deterministic RNN, which is an Euler discretization of the ODE (1) with step sizes $\delta = (\delta_m)$. In particular, for the Lipschitz RNN,

$$\hat{\Phi}_{m,k} = \hat{J}_m \hat{J}_{m-1} \cdots \hat{J}_k, \tag{15}$$

where $\hat{J}_l = I + \delta_l(A + D_l W)$, with $D_l^{ij} = a'([W\bar{h}_l^\delta + U\hat{x}_l + b]^i)e_{ij}$.

In the following, we let $\mathrm{conv}(\mathcal{X})$ denote the convex hull of $\mathcal{X}$. We let $\hat{\boldsymbol{x}}_{0:m} := (\hat{x}_0, \dots, \hat{x}_m)$ so that $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{0:M-1}$, and use the notation $f[\boldsymbol{x}]$ to indicate the dependence of the function f on the vector $\boldsymbol{x}$. Our result will depend on two characterizations of a training sample $s_i = (\boldsymbol{x}_i, y_i)$.

Definition 1 (Classification Margin)**.** The classification margin of a training sample $s_i = (\boldsymbol{x}_i, y_i)$ measured by the Euclidean metric d is defined as the radius of the largest d-metric ball in $\mathcal{X}$ centered at $\boldsymbol{x}_i$ that is contained in the decision region associated with the class label y_i, i.e., it is: $\gamma^d(s_i) = \sup\{a : d(\boldsymbol{x}_i, \boldsymbol{x}) \leq a \Rightarrow g^\delta(\boldsymbol{x}) = y_i \ \ \forall \boldsymbol{x}\}$.

Intuitively, a larger classification margin allows a classifier to associate a larger region centered on a point $\boldsymbol{x}_i$ in the input space to the same class. This makes the classifier less sensitive to input perturbations, and a perturbation of $\boldsymbol{x}_i$ is still likely to fall within this region, keeping the classifier prediction. In this sense, the classifier becomes more robust. In our case, the networks are trained by a loss (cross-entropy) that promotes separation of different classes in the network output. This, in turn, maximizes a certain notion of score of each training sample.

Definition 2 (Score)**.** For a training sample $s_i = (\boldsymbol{x}_i, y_i)$, we define its score as $o(s_i) = \min_{j\neq y_i} \sqrt{2}(e_{y_i} - e_j)^T S^\delta[\boldsymbol{x}_i] \geq 0$, where $e_i \in \mathbb{R}^{d_y}$ is the Kronecker delta vector with $e_i^i = 1$ and $e_i^j = 0$ for $i \neq j$, $S^\delta[\boldsymbol{x}_i] := p(V\bar{h}_M^\delta[\boldsymbol{x}_i])$ with $\bar{h}_M^\delta[\boldsymbol{x}_i]$ denoting the hidden state of the RNN, driven by the input sequence $\boldsymbol{x}_i$, at terminal index M.

Recall that the classifier $g^\delta(\boldsymbol{x}) = \arg\max_{i\in 1,\dots,d_y}[S^\delta]^i[\boldsymbol{x}]$, and the decision boundary between class i and class j in the feature space is given by the hyperplane $\{z = S^\delta : z^i = z^j\}$. A positive score implies that at the network output, classes are separated by a margin that corresponds to the score. However, a large score may not imply a large classification margin.

Following the approach of [52, 63], we obtain the second main result, providing bounds for classification margin for the deterministic RNN classifiers g^δ. We also provide a generalization bound in terms of the classification margin under additional assumptions (see Theorem 11 in **SM**).

Theorem 2. *Suppose that Assumption A holds. Assume that the score $o(s_i) > 0$ and*

$$\gamma(s_i) := \frac{o(s_i)}{C\sum_{m=0}^{M-1}\delta_m \sup_{\hat{\boldsymbol{x}}\in\mathrm{conv}(\mathcal{X})}\|\hat{\Phi}_{M,m+1}[\hat{\boldsymbol{x}}]\|_2} > 0, \tag{16}$$

where $C = \|V\|_2 \left(\max_{m=0,1,\dots,M-1}\left\|\frac{\partial f(\bar{h}_m^\delta,\hat{x}_m)}{\partial \hat{x}_m}\right\|_2\right) > 0$ is independent of s_i (in particular, $C = \|V\|_2[\max_{m=0,\dots,M-1}\|D_m U\|_2]$ for Lipschitz RNNs), the $\hat{\Phi}_{m,k}$ are defined in (15) *and the δ_m are the step sizes. Then, the classification margin for the training sample s_i:*

$$\gamma^d(s_i) \geq \gamma(s_i). \tag{17}$$

Now, recalling from Section 4, up to $\mathcal{O}(\epsilon^2)$ and under the assumption that $\hat{Q}$ vanishes, the loss minimized by the NRNN classifer is, on average, $\ell(\bar{h}_M^\delta) + \epsilon^2\hat{R}(\bar{h}^\delta)$, as $\epsilon \to 0$, with regularizer

$$\hat{R}(\bar{h}^\delta) = \frac{1}{2}\sum_{m=1}^{M}\delta_{m-1}\|\hat{M}_{M-1}\hat{\Phi}_{M-1,m}\sigma_{m-1}\|_F^2, \tag{18}$$

where $\hat{M}_M^T\hat{M}_M := H_{\bar{h}_M^\delta}l$ is the Cholesky decomposition of the Hessian matrix of the convex cross-entropy loss. The appearance of the state-to-state Jacobians in $\Phi_{m,k}$ in both the regularizer (18) and the lower bound (16) suggests that noise injection implicitly aids generalization performance. More precisely, in the small noise regime and on average, NRNNs promote classifiers with large classification margin, an attribute linked to both improved robustness and generalization [63]. In this sense, training with NRNN classifiers is a stochastic strategy to improve generalization over deterministic RNN classifiers, particularly in learning tasks where the given data is corrupted (c.f. the caveats pointed out in [53]).

Theorem 2 implies that the lower bound for the classification margin is determined by the spectrum of the $\hat{\Phi}_{M-1,m}$. To make the lower bound large, keeping δ_m and M fixed, the spectral norm of the $\hat{\Phi}_{M-1,m}$ should be made small. Doing so improves stability of the RNN, but may also lead to vanishing gradients, hindering capacity of the model to learn. To maximize the lower bound while avoiding the vanishing gradient problem, one should tune the numerical step sizes δ_m and noise level ϵ in NRNN appropriately. RNN architectures for the drift which help to ensure moderate Jacobians (e.g., $\|\hat{\Phi}_{M-1,m}\|_2 \approx 1$ for all m [8]) also remain valuable in this respect.

6 Stability and Noise-Induced Stabilization

Here we obtain sufficient conditions to guarantee stochastic stability of CT-NRNNs. This will also provide another lens to highlight the potential of NRNNs for improved robustness. A dynamical system is considered *stable* if trajectories which are close to each other initially remain close at subsequent times. As observed in [46, 42, 7], stability plays an essential role in the study of RNNs to avoid the *exploding gradient problem*, a property of unstable systems where the gradient increases in magnitude with the depth. While gradient clipping during training can somewhat alleviate this issue, better performance and robustness is achieved by enforcing stability in the model itself.

Our stability analysis will focus on establishing *almost sure exponential stability* (for other notions of stability, see **SM**) for CT-NRNNs with the drift function (3). To preface the definition, consider initializing the SDE at two different random variables h_0 and $h_0' := h_0 + \epsilon_0$, where $\epsilon_0 \in \mathbb{R}^{d_h}$ is a constant non-random perturbation with $\|\epsilon_0\| \leq \delta$. The resulting hidden states, h_t and h_t', are set to satisfy (2) with the same Brownian motion B_t, starting from their initial values h_0 and h_0', respectively. The evolution of $\epsilon_t = h_t' - h_t$ satisfies

$$\mathrm{d}\epsilon_t = A\epsilon_t \mathrm{d}t + \Delta a_t(\epsilon_t)\mathrm{d}t + \Delta\sigma_t(\epsilon_t)\mathrm{d}B_t, \tag{19}$$

where $\Delta a_t(\epsilon_t) = a(Wh_t' + Ux_t + b) - a(Wh_t + Ux_t + b)$ and $\Delta\sigma_t(\epsilon_t) = \sigma(h_t + \epsilon_t, x_t) - \sigma(h_t, x_t)$. Since $\Delta a_t(0) = 0$, $\Delta\sigma_t(0) = 0$ for all $t \in [0, T]$, $\epsilon_t = 0$ admits a trivial *equilibrium* for (19). Our objective is to analyze the stability of the solution $\epsilon_t = 0$, that is, to see how the final state ϵ_T (and hence the output of the RNN) changes for an arbitrarily small initial perturbation $\epsilon_0 \neq 0$. To this end, we consider an extension of the Lyapunov exponent to SDEs at the level of sample path [41].

Definition 3 (Almost sure global exponential stability)**.** The sample (or pathwise) Lyapunov exponent of the trivial solution of (19) is $\Lambda = \limsup_{t\to\infty} t^{-1} \log \|\epsilon_t\|$. The trivial solution $\epsilon_t = 0$ is *almost surely globally exponentially stable* if Λ is almost surely negative for all $\epsilon_0 \in \mathbb{R}^{d_h}$.

For the sample Lyapunov exponent $\Lambda(\omega)$, there is a constant $C > 0$ and a random variable $0 \leq \tau(\omega) < \infty$ such that for all $t > \tau(\omega)$, $\|\epsilon_t\| = \|h_t' - h_t\| \leq Ce^{\Lambda t}$ almost surely. Therefore, almost sure exponential stability implies that almost all sample paths of (19) will tend to the equilibrium solution $\epsilon = 0$ exponentially fast. With this definition in tow, we obtain the following stability result.

Theorem 3. *Assume that a is monotone non-decreasing, and $\sigma_1\|\epsilon\| \leq \|\Delta\sigma_t(\epsilon)\|_F \leq \sigma_2\|\epsilon\|$ for all nonzero $\epsilon \in \mathbb{R}^{d_h}$, $t \in [0, T]$. Then for any $\epsilon_0 \in \mathbb{R}^{d_h}$, with probability one,*

$$\phi + \lambda_{\min}(A^{\mathrm{sym}}) \leq \Lambda \leq \psi + L_a\sigma_{\max}(W) + \lambda_{\max}(A^{\mathrm{sym}}), \tag{20}$$

with $\phi = -\sigma_2^2 + \frac{\sigma_1^2}{2}$ and $\psi = -\sigma_1^2 + \frac{\sigma_2^2}{2}$, where L_a is the Lipschitz constant of a.

In the special case without noise ($\sigma_1 = \sigma_2 = 0$), we recover case (a) of Theorem 1 in [16]: when A^{sym} is negative definite and $\sigma_{\min}(A^{\mathrm{sym}}) > L_a\sigma_{\max}(W)$, Theorem 3 implies that (2) is exponentially stable. Most strikingly, and similar to [35], Theorem 3 implies that even if the deterministic CT-RNN is not exponentially stable, it can be stabilized through a stochastic perturbation. Consequently, injecting noise appropriately can improve training performance.

7 Empirical Results

The evaluation of robustness of neural networks (RNNs in particular) is an often neglected yet crucial aspect. In this section, we investigate the robustness of NRNNs and compare their performance to other recently introduced state-of-the-art models on both clean and corrupted data. We refer to Section G in **SM** for further details of our experiments.

Here, we study the sensitivity of different RNN models with respect to a sequence of perturbed inputs during inference time. We consider different types of perturbations: (a) white noise; (b) multiplicative white noise; (c) salt and pepper; and (d) adversarial perturbations. To be more concrete, let x be a sequence. The perturbations in consideration are as follows.

- *Additive white noise perturbations* are constructed as $\tilde{x} = x + \Delta x$, where the additive noise is drawn from a Gaussian distribution $\Delta x \sim \mathcal{N}(0, \sigma)$. This perturbation strategy emulates measurement errors that can result from data acquisition with poor sensors (where σ can be used to vary the strength of these errors). *Multiplicative white noise perturbations* are constructed as $\tilde{x} = x \cdot \Delta x$, where the additive noise is drawn from a Gaussian distribution $\Delta x \sim \mathcal{N}(1, \sigma_M)$.
- *Salt and pepper perturbations* emulate defective pixels that result from converting analog signals to digital signals. The noise model takes the form $\mathbb{P}(\tilde{X} = X) = 1 - \alpha$, and $\mathbb{P}(\tilde{X} = \max) = \mathbb{P}(\tilde{X} = \min) = \alpha/2$, where $\tilde{X}(i, j)$ denotes the corrupted image and $\min$ and $\max$ denote to the minimum and maximum pixel values. The parameter α controls the proportion of defective pixels.
- *Adversarial perturbations* are "worst-case" non-random perturbations maximizing the loss $\ell(g^\delta(X + \Delta X), y)$ subject to the constraint that the norm of the perturbation $\|\Delta X\| \leq r$. We consider the fast gradient sign method for constructing these perturbations [55].

We consider in addition to the NRNN three other RNNs derived from continuous-time models, including the Lipschitz RNN [16] (the deterministic counterpart to our NRNN), the coupled oscillatory RNN (coRNN) [49] and the antisymmetric RNN [7]. We also consider the exponential RNN [32], a discrete-time model that uses orthogonal recurrent weights. We train each model with the prescribed tuning parameters for the ordered (see Sec. 7.1) and permuted (see **SM**) MNIST task. For the Electrocardiogram (ECG) classification task we performed a non-exhaustive hyper-tuning parameter search. For comparison, we train all models with hidden-to-hidden weight matrices of dimension $d_h = 128$. We average the classification performance over ten different seed values.

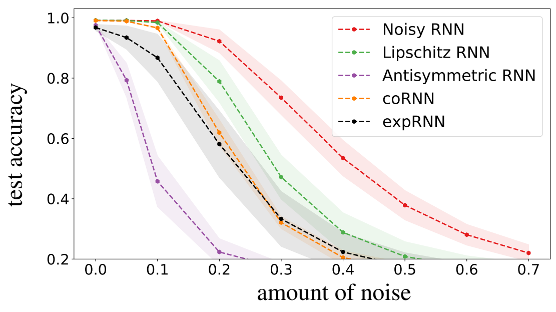
7.1 Ordered Pixel-by-Pixel MNIST Classification

First, we consider the ordered pixel-by-pixel MNIST classification task [31]. This task sequentially presents 784 pixels to the model and uses the final hidden state to predict the class membership probability of the input image. In the **SM** we present additional results for the situation when instead of an ordered sequence a fixed random permutation of the input sequence is presented to the model.

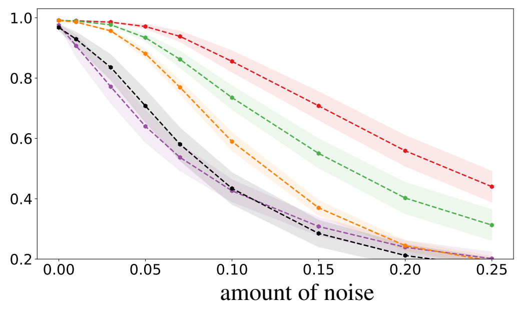
Table 1 shows the average test accuracy (evaluated for models that are trained with 10 different seed values) for the ordered task. Here we present results for white noise and salt and pepper (S&P) perturbations. While the Lipschitz RNN performs best on clean input sequences, the NRNNs show an improved resilience to input perturbations. Here, we consider two different configuration for the NRNN. In both cases, we set the multiplicative noise level to 0.02, whereas we consider the additive noise levels 0.02 and 0.05. We chose these configurations as they appear to provide a good trade-off between accuracy and robustness. Note that the predictive accuracy on clean inputs starts to drop when the noise level becomes too large.

Table 1: Robustness w.r.t. white noise (σ) and S&P (α) perturbations on the ordered MNIST task.

Name	clean	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\alpha = 0.03$	$\alpha = 0.05$	$\alpha = 0.1$
Antisymmetric RNN [7]	97.5%	45.7%	22.3%	17.0%	77.1%	63.9%	42.6%
CoRNN [49]	99.1%	96.6%	61.9%	32.1%	95.6%	88.1%	58.9%
Exponential RNN [32]	96.7%	86.7%	58.1%	33.3%	83.6%	70.7%	43.4%
Lipschitz RNN [16]	**99.2**%	98.4%	78.9%	47.1%	97.6%	93.4%	73.5%
NRNN (mult./add. noise: 0.02/0.02)	99.1%	**98.9**%	88.4%	62.9%	98.3%	95.6%	78.7%
NRNN (mult./add. noise: 0.02/0.05)	99.1%	**98.9**%	**92.2**%	**73.5**%	**98.5**%	**97.1**%	**85.5**%



(a) White noise perturbations.



(b) Salt and pepper perturbations.

Figure 1: Test accuracy for the ordered MNIST task as function of the strength of input perturbations.

Table 2 shows the average test accuracy for the ordered MNIST task for adversarial perturbations. Again, the NRNNs show a superior resilience even to large perturbations, whereas the Antisymmetric and Exponential RNN appear to be sensitive even to small perturbations.

Figure 1 summarizes the performance of different models with respect to white noise and salt and pepper perturbations. The colored bands indicate ± 1 standard deviation around the average performance. In all cases, the NRNN appears to be less sensitive to input perturbations as compared to the other models, while maintaining state-of-the-art performance for clean inputs.

7.2 Electrocardiogram (ECG) Classification

Next, we consider the Electrocardiogram (ECG) classification task that aims to discriminate between normal and abnormal heart beats of a patient that has severe congestive heart failure [20]. We use 500 sequences of length 140 for training, 500 sequences for validation, and 4000 sequences for testing.

Table 3 shows the average test accuracy (evaluated for models that are trained with 10 different seed values) for this task. We present results for additive white noise and multiplicative white noise perturbations. Here, the NRNN, trained with multiplicative noise level set to 0.03 and additive noise levels set to 0.06, performs best both on clean as well as on perturbed input sequences.

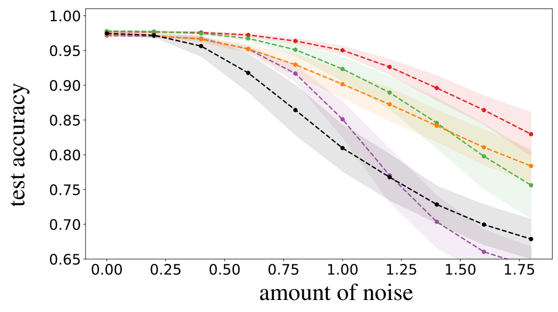
Figure 2 summarizes the performance of different models with respect to additive and multiplicative white noise perturbations. Again, the NRNN appears to be less sensitive to input perturbations as compared to the other models, while achieving state-of-the-art performance for clean inputs.

Table 2: Robustness w.r.t. adversarial perturbations on the ordered pixel-by-pixel MNIST task.

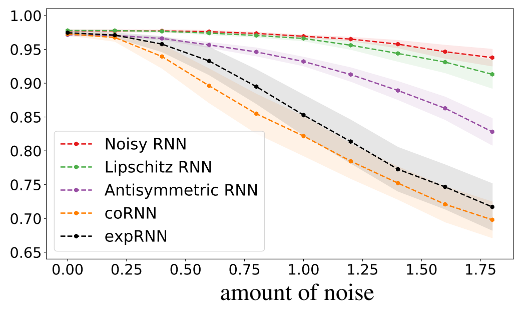
Name	$r = 0.01$	$r = 0.05$	$r = 0.1$	$r = 0.15$
Antisymmetric RNN [7]	79.4%	24.7%	11.4%	10.2%
CoRNN [49]	97.5%	85.5%	55.9%	35.1%
Exponential RNN [32]	94.5%	59.3%	19.7%	14.3%
Lipschitz RNN [16]	98.1%	85.7%	58.9%	37.1%
NRNN (mult./add. noise: 0.02/0.02)	**98.8**%	94.3%	79.6%	58.3%
NRNN (mult./add. noise: 0.02/0.05)	**98.8**%	**95.5**%	**86.8**%	**70.6**%

Table 3: Robustness w.r.t. white (σ) and multiplicative (σ_M) noise perturbations on the ECG task.

Name	clean	$\sigma = 0.4$	$\sigma = 0.8$	$\sigma = 1.2$	$\sigma_M = 0.4$	$\sigma_M = 0.8$	$\sigma_M = 1.2$
Antisymmetric RNN [7]	97.1%	96.6%	91.6%	77.0%	96.6%	94.6%	91.2%
CoRNN [49]	97.5%	96.8%	92.9%	87.2%	93.9%	85.4%	78.4%
Exponential RNN [32]	97.4%	95.6%	86.4%	76.7%	95.7%	89.4%	81.3%
Lipschitz RNN [16]	**97.7**%	97.4%	95.1%	88.9%	97.6%	97.0%	95.6%
NRNN (mult./add. noise: 0.03/0.06)	**97.7**%	**97.5**%	**96.3**%	**92.6**%	**97.7**%	**97.3**%	**96.5**%



(a) Additive white noise perturbations.



(b) Multiplicative white noise perturbations.

Figure 2: Test accuracy for the ECG task as function of the strength of input perturbations.

8 Conclusion

In this paper we provide a thorough theoretical analysis of RNNs trained by injecting noise into the hidden states. Within the framework of SDEs, we study the regularizing effects of general noise injection schemes. The experimental results are in agreement with our theory and its implications, finding that Noisy RNNs achieve superior robustness to input perturbations, while maintaining state-of-the-art generalization performance. We believe our framework can be used to guide the principled design of a class of reliable and robust RNN classifiers.

Our work opens up a range of interesting future directions. In particular, for deterministic RNNs, it was shown that the models learn optimally near the edge of stability [8]. One could extend these analyses to NRNNs with the ultimate goal of improving their performance. On the other hand, as discussed in Section 5, although the noise is shown here to implicitly stabilize RNNs, it could negatively impact capacity for long-term memory [42, 70]. Providing analyses to account for this and the implicit bias due to the stochastic optimization procedure [50, 14] is the subject of future work.

Acknowledgements

We are grateful for the generous support from Amazon AWS. S. H. Lim would like to acknowledge Nordita Fellowship 2018-2021 for providing support of this work. N. B. Erichson, L. Hodgkinson, and M. W. Mahoney would like to acknowledge the IARPA (contract W911NF20C0035), ARO, NSF, and and ONR via its BRC on RandNLA for providing partial support of this work. Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

References

[1] Alnur Ali, Edgar Dobriban, and Ryan Tibshirani. The implicit regularization of stochastic gradient flow for least squares. In *International Conference on Machine Learning*, pages 233–244. PMLR, 2020.

[2] Raman Arora, Peter Bartlett, Poorya Mianjy, and Nathan Srebro. Dropout: Explicit forms and capacity control. In *International Conference on Machine Learning*, pages 351–361. PMLR, 2021.

[3] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael W. Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020.

[4] Kaushik Balakrishnan and Devesh Upadhyay. Deep adversarial Koopman model for reaction-diffusion systems. *arXiv preprint arXiv:2006.05547*, 2020.

[5] Chris M Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

[6] Alexander Camuto, Matthew Willetts, Umut Simsekli, Stephen J Roberts, and Chris C Holmes. Explicit regularisation in Gaussian noise injections. In *Advances in Neural Information Processing Systems*, volume 33, pages 16603–16614, 2020.

[7] Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2019.

[8] Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. In *International Conference on Machine Learning*, pages 873–882. PMLR, 2018.

[9] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[10] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2019.

[11] Michal Derezinski, Feynman T Liang, and Michael W Mahoney. Exact expressions for double descent and implicit regularization via surrogate random design. In *Advances in Neural Information Processing Systems*, volume 33, pages 5152–5164, 2020.

[12] Adji Bousso Dieng, Rajesh Ranganath, Jaan Altosaar, and David Blei. Noisin: Unbiased regularization for recurrent neural networks. In *International Conference on Machine Learning*, pages 1252–1261. PMLR, 2018.

[13] Akshunna S. Dogra and William Redman. Optimizing neural networks via Koopman operator theory. In *Advances in Neural Information Processing Systems*, volume 33, pages 2087–2097, 2020.

[14] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K Fletcher. Implicit bias of linear RNNs. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 2982–2992. PMLR, 2021.

[15] Rainer Engelken, Fred Wolf, and LF Abbott. Lyapunov spectra of chaotic recurrent neural networks. *arXiv preprint arXiv:2006.02427*, 2020.

[16] N. Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W. Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2021.

[17] N Benjamin Erichson, Michael Muehlebach, and Michael W Mahoney. Physics-informed autoencoders for Lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

[18] Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[19] D. F. Gleich and M. W. Mahoney. Anti-differentiating approximation algorithms: A case study with min-cuts, spectral, and flow. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1018–1025, 2014.

[20] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[21] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. Maxup: Lightweight adversarial training with data augmentation improves neural network training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2474–2483, 2021.

[22] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

[24] Liam Hodgkinson, Chris van der Heide, Fred Roosta, and Michael W Mahoney. Stochastic continuous normalizing flows: Training SDEs as ODEs. In *Uncertainty in Artificial Intelligence (UAI)*, 2021.

[25] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[26] Kam-Chuen Jim, C Lee Giles, and Bill G Horne. An analysis of noise in recurrent neural networks: Convergence and generalization. *IEEE Transactions on Neural Networks*, 7(6):1424–1438, 1996.

[27] Anil Kag, Ziming Zhang, and Venkatesh Saligrama. RNNs incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *International Conference on Learning Representations*, 2020.

[28] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[29] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, volume 33, pages 6696–6707, 2020.

[30] Peter E Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*, volume 23. Springer Science & Business Media, 2013.

[31] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[32] Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pages 3794–3803, 2019.

[33] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional Koopman operators for model-based control. In *International Conference on Learning Representations*, 2019.

[34] Soon Hoe Lim. Understanding recurrent neural networks using nonequilibrium response theory. *J. Mach. Learn. Res.*, 22:47–1, 2021.

[35] Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. How does noise help robustness? Explanation and exploration under the neural SDE framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 282–290, 2020.

[36] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR, 2018.

[37] Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[38] Chao Ma, Stephan Wojtowytsch, and Lei Wu. Towards a mathematical understanding of neural network-based machine learning: What we know and what we don't. *arXiv preprint arXiv:2009.10713*, 2020.

[39] M. W. Mahoney. Approximate computation and implicit regularization for very large-scale data analysis. In *Proceedings of the 31st ACM Symposium on Principles of Database Systems*, pages 143–154, 2012.

[40] M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 121–128, 2011.

[41] Xuerong Mao. *Stochastic Differential Equations and Applications*. Elsevier, 2007.

[42] John Miller and Moritz Hardt. Stable recurrent models. *arXiv preprint arXiv:1805.10369*, 2018.

[43] Jeremy Morton, Freddie D Witherden, and Mykel J Kochenderfer. Deep variational Koopman models: Inferring Koopman observations for uncertainty-aware dynamics modeling and control. *arXiv preprint arXiv:1902.09742*, 2019.

[44] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pages 5109–5118, 2017.

[45] Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

[46] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR, 2013.

[47] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning III: Explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

[48] Alejandro F Queiruga, N Benjamin Erichson, Dane Taylor, and Michael W Mahoney. Continuous-in-depth neural networks. *arXiv preprint arXiv:2008.02389*, 2020.

[49] T. Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021.

[50] Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

[51] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Generalization error of deep neural networks: Role of classification margin and data structure. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 147–151. IEEE, 2017.

[52] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[53] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[54] Qi Sun, Yunzhe Tao, and Qiang Du. Stochastic training of residual networks: A differential equation viewpoint. *arXiv preprint arXiv:1812.00174*, 2018.

[55] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[56] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1130–1140, 2017.

[57] Sachin S Talathi and Aniket Vartak. Improving performance of recurrent neural network with ReLU nonlinearity. *arXiv preprint arXiv:1511.03771*, 2015.

[58] Peter Toth, Danilo J Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2019.

[59] Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[60] Ryan Vogt, Maximilian Puelma Touzel, Eli Shlizerman, and Guillaume Lajoie. On Lyapunov exponents for RNNs: Understanding information propagation using dynamical systems tools. *arXiv preprint arXiv:2006.14123*, 2020.

[61] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *Proceedings of the 37th International Conference on Machine Learning*, pages 10181–10192, 2020.

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

[63] Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

[64] Yibo Yang, Jianlong Wu, Hongyang Li, Xia Li, Tiancheng Shen, and Zhouchen Lin. Dynamical system inspired adaptive time stepping controller for residual network families. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6648–6655, 2020.

[65] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pages 4954–4964, 2018.

[66] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[67] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[68] Huishuai Zhang, Da Yu, Mingyang Yi, Wei Chen, and Tie-yan Liu. Stability and convergence theory for learning ResNet: A full characterization. 2019.

[69] Jingfeng Zhang, Bo Han, Laura Wynter, Bryan Kian Hsiang Low, and Mohan Kankanhalli. Towards robust ResNet: A small step but a giant leap. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[70] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do RNN and LSTM have long memory? In *International Conference on Machine Learning*, pages 11365–11375. PMLR, 2020.

[71] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ODE-Net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2019.