Dynamically Computing Adversarial Perturbations for Recurrent Neural Networks

Shankar A. Deka, Member, IEEE, Dušan M. Stipanović, Senior Member, IEEE, and Claire J. Tomlin, Fellow, IEEE

Abstract—Convolutional and recurrent neural networks (RNNs) have been widely used to achieve state-of-the-art performance on classification tasks. However, it has also been noted that these networks can be manipulated adversarially with relative ease, by carefully crafted additive perturbations to the input. Though several experimentally established prior works exist on crafting and defending against attacks, it is also desirable to have rigorous theoretical analyses to illuminate conditions under which such adversarial inputs exist. This article provides both the theory and supporting experiments for real-time attacks. The focus is specifically on recurrent architectures and inspiration is drawn from dynamical systems’ theory to naturally cast this as a control problem, allowing dynamic computation of adversarial perturbations at each timestep of the input sequence, thus resembling a feedback controller. Illustrative examples are provided to supplement the theoretical discussions.

Index Terms—Adversarial examples, control synthesis, dynamical systems, recurrent neural network (RNN).

I. INTRODUCTION

ADVERSARIAL attacks on neural networks and techniques to robustify against such attacks have been a topic of continued interest in the machine learning communities. A wide variety of methodologies have been presented in the past, for crafting adversarial input disturbances. These broadly fall under one category or another based on the information available to the attacker regarding the neural network being attacked. For example, widely known methods, such as fast gradient sign method (FGSM) [1], Carlini and Wagner (C&W) attack [2], Jacobian-based saliency map attack [3], to name a few, were developed considering a white-box attack model. Others like [4]–[7] work for black-box attacks. “Transfer attack” models like [8] do not assume knowledge of the model but require the training data. As far as the applications are concerned, the majority of prior works focus on computer vision [1], [9], [10], wherein given a feed-forward network [usually a convolutional neural network (CNN)] and an input image, an adversarial disturbance is crafted to fool the network on image classification tasks. A few, like [11], have considered attacks on recurrent neural networks (RNNs), although their algorithm requires unfolding the RNN first, thus resembling a feed-forward network in practice.

Observations drawn from these prior empirically supported algorithms and results have led to different explanations regarding adversarial examples. For instance, the research in [12], [13] speculates that adversarial examples are confined to tight “pockets” due to the highly nonlinear data manifold that the network represents, whereas in contrast, the authors of [1] have argued that “models being too linear” is what gives rise to adversarial examples. This motivates the need for a more rigorous mathematical formulation of the adversarial perturbation problem, which would hopefully shed light on how the input perturbations impact the network output from nominal input, and under what conditions adversarial perturbations are guaranteed to exist. From myriad experimentation in previous work, it appears that any given input can be perturbed adversarially using some simple gradient-based disturbance. A complementary line of work, on defense methodologies against adversarial inputs, has been considered by [14]–[16], with the simplest approach being augmentation of training data with adversarial examples.

It is also worthwhile to note some interesting practical experimentation-driven works surrounding adversarial attacks and RNNs. For example, [17] describes an audio input attack on a keyword spotting (KWS) system [18]. In contrast to our article, their real-time attack is not targeted toward an RNN classifier, but rather an RNN is trained to learn to generate adversarial inputs. The training data for this RNN itself are obtained using a traditional iterative FGSM method. With a similar idea, [19] adopts imitation learning approach to train an RNN model to generate real-time adversarial sound inputs to attack a voice command system. [20] proposes a targeted attack on a CNN–RNN-based model for image captioning. The perturbations are made on input image pixels, such that the semantic embedding of the perturbed image is close to that of the target image. [11], [21] demonstrate attacks on RNN text classification, by iteratively changing the original text, one word at a time, based on the Jacobian of the entire sequence of inputs. In each iteration, the input word corresponding to the largest element of the Jacobian matrix is altered, and the process is repeated for the new input sequence, until the sequence is misclassified.

Manuscript received 9 June 2021; revised 4 January 2022; accepted 9 March 2022. Date of publication 9 May 2022; date of current version 21 October 2022. Recommended by Associate Editor M. Abbaspazdah. This work was supported in part by NIFA; in part by DARPA Assured Autonomy Program; and in part by ONR BRC Program in Multibody Systems. (Corresponding author: Shankar A. Deka.)

Shankar A. Deka and Claire J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: deka.shankar@berkeley.edu; tomlin@eecs.berkeley.edu).

Dušan M. Stipanović is with the Coordinated Science Laboratory and the ISE Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: dusan@illinois.edu). Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2022.3165980.

Digital Object Identifier 10.1109/TCST.2022.3165980
In this article, we consider white-box attacks specifically on RNN models and demonstrate how their recurrent nature can also be used to compute disturbances dynamically. This is the key difference from the limited prior works on adversarial attack algorithms applicable to RNNs. The contributions of our article are as follows.

1) We present an algorithm to \textit{sequentially} generate adversarial perturbations for RNNs. In other words, the \textit{k}th input \( \tilde{x}(k) \) of the adversarial sequence is generated in constant time using just the \textit{k}th input \( x(k) \) of the nominal sequence and the \textit{k}th hidden states. This means that sequential inputs can be adversarially perturbed one timestep at a time, rather than collecting the entire sequence. This point is illustrated by Fig. 1. This in principle means that adversarial noise can be injected into real-time signals like speech commands. Our method is also amenable for scenarios with very long input sequences, as it would \textit{scale linearly} with sequence length.

2) We present a control-theory-based analysis to explain how the proposed disturbance sequence, viewed as a feedback control law, steers the state trajectories of the adversarially perturbed system away from the nominal, unperturbed states. Also, viewed in this manner as a dynamical system, we show how the problem of finding adversarial perturbations can be formulated as an optimal control problem, which enables us to use an off-the-shelf control toolbox for crafting adversarial examples.

3) Finally, a constructive proof regarding the existence of adversarial perturbations is provided. In particular, we present conditions under which one can always find an adversarial perturbation to a given nominal input. Since these results are analytical in nature, the conditions may be used in robust training of RNNs by directly imposing constraints on the training parameters, for example, the matrix measure condition in Theorem 1 or Example 1.

To better elucidate the dynamical system analysis in the sections to follow, we begin with a concrete example of a sequence classification task in Section II and how the RNN classifier can be seen as a time-varying nonlinear system, with disturbance acting as a control input. The formal analysis results will be presented in Section III. Later, in Section IV, we shall experimentally demonstrate our approach of crafting adversarial perturbations to a variety of RNN classification examples\(^1\) covering different network architectures and sizes, namely: 1) frequency discrimination using gated recurrent unit (GRU) architecture; 2) MNIST digit classification using vanilla RNN; 3) human activity recognition (HAR) using GRU; and 4) IMDb movie review classification using long short-term memory (LSTM) architecture. Finally, we present our concluding remarks and future directions in Section V.

\section{Time-Series Classification Task}

One of the key applications of RNNs is sequence classification where the input–output map is “many-to-one.” Commonly used examples in the literature include sentiment analysis (IMDB reviews or Twitter), MNIST digit recognition, HAR, handwriting recognition (IAM on-line handwriting database), and urban sound classification. The example we use to motivate the discussion to follow is the “frequency discrimination task” \cite{22, 23}. In this task, the RNN must classify sinusoidal input sequences based on their time period \( T \) into two classes, as shown in Fig. 2. The time period of class 0 inputs lies in the interval \((5, 6)\), while those of class 1 inputs is in the interval \((0, 5) \cup (6, 100)\). Details on the model and training can be found in Section IV, but in this section, we shall refer to this toy example to build a basic background.

\subsection{Notation}

We would like to highlight some key notational subtleties between control and machine learning communities. We follow standard notations from machine learning literature in this article. For example, the symbol \( x \) denotes the \textit{input} of our dynamical system (rather than the \textit{state}, for which we use \( h \) instead). Subscripts are used to denote various elements of the same set, while superscripts are used to denote components of a vector. Time dependence is shown via function argument \((k \text{ for discrete time and } t \text{ for continuous})\), although time

\(^1\)Code available at https://github.com/dekovski/Adversarial-attack-RNN
dependence is not always explicitly shown, for notational brevity. Thus, \( v^i_j(k) \) represents the \( i \)th component of vector \( v_j \) at time \( k \), unless stated otherwise. In case of continuous time, \( \dot{v} \) indicates derivative of \( v \) with respect to time.

### B. Perturbing the Inputs for Misclassification

The goal of an adversarial attack is to modify the inputs in a way that is indistinguishable to a human, but lead the neural network to have a very different behavior. Let us start with the compact representation of an RNN

\[
\begin{align*}
  h(k+1) &= \tilde{f}(h(k), x(k)), \quad h(0) = \text{fixed} \tag{1} \\
  y(k) &= \tilde{g}(h(k)) \tag{2}
\end{align*}
\]

where \( h(k) \in \mathbb{R}^n \) is the hidden state vector and \( x(k) \in \mathbb{R}^m \) is the RNN input. The output vector \( y(k) \in \mathbb{R}^l \) for an RNN classifier is interpreted as a probability distribution over \( l \) classes. The functions \( \tilde{f} \) and \( \tilde{g} \) are globally bounded, smooth functions with bounded derivatives. To keep the notations readable, we shall only consider single-layer RNNs. But the work in this article can be very easily applied to the multilayer case without any loss of generality—by simply stacking together the states of each layer at the \( k \)th timestep into a single vector \( h(k) \), and by taking \( y(k) \) to be the output of the final layer only.

For the classification task in Fig. 2, for example, the function \( \tilde{f} \) represents a GRU layer and \( \tilde{g} \) represents a softmax layer, with \( n = 2, m = 1 \), and \( l = 2 \). The phase plot of the network \( (h(k) \text{ trajectories}) \) corresponding to input signals, belonging to the two classes, is shown in Fig. 3(a). Plotted in Fig. 3(b) are the respective classification probabilities of signals (i.e., the first component of output vector \( y(k) \) for class 0 signals and second component for class 1 signals), which appear to be uniformly lower bounded by some monotonically increasing function.

We convert our RNN equation (1) into a continuous-time system for the convenience of analysis as

\[
\frac{dh}{dt} = \frac{1}{\Delta}(\tilde{f}(h, x) - h) = f(h, x), \quad h(0) = h_{\text{initial}} \tag{3}
\]

where the state \( h(t) \in \mathbb{R}^n \), the control input \( x(t) \in \mathbb{R}^n \), and \( \Delta \) is a small positive constant. We can now begin to setup our problem in a control theoretic framework and use Lyapunov-like functions to describe RNN classification task. Let us say the RNN is trained to classify input sequences belonging to set \( \mathcal{U} \) into two classes. Furthermore, let there be two sets of input signals, denoted by \( \mathcal{U}_a \) and \( \mathcal{U}_b \), s.t., \( \mathcal{U}_a \cup \mathcal{U}_b = \mathcal{U} \), and open sets \( \mathcal{S}_a \text{and} \mathcal{S}_b \in \mathbb{R}^n \) such that if \( x_a(t) \in \mathcal{U}_a \), then the corresponding state trajectory \( h_a(t) \) of system (3) converges to the interior of the set \( \mathcal{S}_a \). Similarly, \( h_b(t) \) corresponding to any \( x_b(t) \in \mathcal{U}_b \) converges to \( \mathcal{S}_b \). In our frequency discriminator example, the sets \( \mathcal{S}_a, \mathcal{S}_b \) can be taken to be the two half-planes whose boundaries are the “0.99” and “0.11” lines, respectively, as shown by the shaded regions in light green (top-left corner) and cyan (bottom-right corner) in Fig. 3(a). In addition, motivated by Fig. 3(b), let us define a Lyapunov-like function \( V(h) \) supported on the complement of set \( \mathcal{S}_a \) (denoted by \( \mathcal{S}'_a \)), such that \( W(t) > V(h_a(t)) > 0 \) whenever \( h_a(t) \notin \mathcal{S}_a \), where \( W(t) \) is a monotonically decreasing function. We formulate our problem of finding adversarial input perturbations to RNNs, with the help of these sets and functions defined here, in the following section.

### C. Formulation and Preliminary Analysis

Given input \( x_a(t) \in \mathcal{U}_a \), our goal is to find a perturbation \( d(t) \) to \( x_a(t) \) such that the corresponding trajectory no longer converges to \( \mathcal{S}_a \), and possibly converges to \( \mathcal{S}_b \). In other words, if we consider solution \( h(t) \) given by

\[
\dot{h}(t) = f(h(t), x_a(t) + d(t)), \quad h(0) = h_{\text{initial}} \tag{4}
\]

then we want \( V(h(t)) > \epsilon \) for all times \( t > T \), where \( \epsilon \) and \( T \) are some positive constants. To lead the network to misclassification, we need a stronger condition, i.e., given a \( x_a(t) \in \mathcal{U}_a \) we want \( d(t) \) subject to \( x_a(t) + d(t) \in \mathcal{U}_b \).

We start with

\[
\begin{align*}
  \dot{h}_1(t) &= f(h_1(t), x_a(t)) \\
  \dot{h}_2(t) &= f(h_2(t), x_a(t) + d(t)) \\
  h_1(0) &= h_2(0)
\end{align*} \tag{5}
\]
along with

$$0 < V(h_1(t)) < W(t).$$

Then, along the trajectory $h_2(t)$, we have

$$\dot{V}(h_2(t)) = \left(\nabla_h V(h_2)\right)^T \dot{h}_2(t)
\begin{equation}
= \left(\nabla_h V(h_2)\right)^T f(h_2, x_a + d)
= \left(\nabla_h V(h_2)\right)^T (f(h_2, x_a)
+ (\nabla_x f(h_2, x_a))d + o(d))
\simeq (\nabla_h V(h_2))^T f(h_2, x_a)
+ (\nabla_h V(h_2)) \nabla_x f(h_2, x_a) d.
\end{equation}

Let us define the term $g(h, u) \doteq (\nabla_h V(h))^T f(h_2, x_a)$, so the first term in the last line of (6) is equal to $g(h_2, x_a)$. Now, using the mean-value theorem for multivariable functions [25], we get

$$g(h_2, x_a) = g(h_1, x_a) + (\nabla_h g(\tilde{h}, x_a))^T (h_2 - h_1)$$

for some $\tilde{h} = h_1 + \alpha(h_2 - h_1)$ and diagonal matrix $\alpha$ with entries belonging to $[0, 1]$. Substituting this into (6) yields

$$\dot{V}(h_2(t)) \simeq (\nabla_h V(h_1))^T f(h_1, x_a)
+ \left[\nabla_h^2 V(\tilde{h}) \cdot f(\tilde{h}, x_a) + \nabla_h f(\tilde{h}, x_a) \cdot \nabla_h V(\tilde{h})\right]^T (h_2 - h_1)
+ (\nabla_h V(h_2))^T (\nabla_x f(h_2, x_a))d
\leq 0
\begin{equation}
\sum_{t = 0}^{N} \left[\nabla_h^2 V(\tilde{h}) \cdot f(\tilde{h}, x_a) + \nabla_h f(\tilde{h}, x_a) \cdot \nabla_h V(\tilde{h})\right]^T (h_{t+1} - h_t)
+ (\nabla_h V(h_2))^T (\nabla_x f(h_2, x_a))d
\end{equation}

At this point, we note that choosing

$$d = d(t) = \alpha(\text{sign} ((\nabla_h V(h_2))^T \nabla_x f(h_2, x_a)))$$

for some small positive $\alpha$ would be an appropriate choice in making $V(h_2(t))$ increase (instantaneously) at time $t$. Indeed, we can make our frequency discriminator RNN classify and assign very high probability to the incorrect class by choosing such a disturbance, as shown in Figs. 4 and 5.

Figs. 4 and 5 show how the addition of the adversarial disturbance given by (8) (with $\alpha = 0.15$) to the nominal input impacts the network. The Lyapunov-like function $V$ shown in Fig. 4 is chosen (among many other ways) as follows:

$$V(h) = \max \left(0, \begin{bmatrix} 1 \\ 0 \end{bmatrix} y(h) - \tilde{y}\right)^2$$

$$\tilde{y} = \text{threshold} = 0.9.$$

Since only the direction of $\nabla_h V$ is relevant, we can use any affine transformation of $V(h)$ given in (9), as indicated in Fig. 5.

Before we move to the main result and continue with the analysis in Section III, it is important to note at this point that such a gradient-based attack on RNNs is different from previous work on adversarial attacks on feed-forward NNs, like CNNs for image classification. This is because the adversarial input disturbances are applied sequentially at each time, and the disturbance at time $k$ would depend on the
disturbance at previous times implicitly through its dependence on the state trajectory $h(k)$ (as they are influenced by past input disturbances). In previous work regarding attacks in CNNs, for example, the disturbance to each pixel is computed simply based on a gradient around the current value of the pixels, and therefore these pixel-wise disturbances do not affect one another. Another practical difference in our disturbance computation for RNNs compared with previous methods is that disturbances at time step $k$ are computed without any knowledge of future inputs. Previous attacks are limited in the sense that one would require to collate the inputs at all timesteps before computing the disturbance, thus prohibiting “real-time” injection of disturbance to the inputs.

In the next section, we shall see the worst case effect of the input disturbance given by (8). The main result of this article is also presented—we show some sufficient conditions under which adversarial input perturbations exist, as well as how they can be constructed.

III. EXISTENCE OF ADVERSARIAL DISTURBANCE

As mentioned earlier, in this section, we present the existence results for adversarial perturbations in RNNs. But let us first look at how the effect of disturbance (8) can be upper bounded. Robustification against adversarial attacks can then be translated to making this upper bound smaller.

Thus, returning back to (7), let us simplify the expressions by considering $V(h) = a^T h + b$, i.e., to be linear, so we get $V = a$ and $V^2 = 0$. One can then integrate both sides of (7) and obtain

$$a^T h_2(t) = a^T h_1(t) + \int_0^t (\nabla_h f(\bar{h}, x_a)a)T(h_2(s) - h_1(s)) ds$$

Using the Bellman–Grönwall inequality, we then obtain

$$|a|_{\min} \cdot \|h_2(t) - h_1(t)\| \leq \|a\| \int_0^t \|\nabla_h f(\bar{h}, x_a)\| \|h_2(s) - h_1(s)\| ds$$

$$+ \|a\| \int_0^t \|\nabla_x f(h_2(a), x_a)\|_1 ds$$

$$\implies \frac{|a|_{\min}}{|a|} \|h_2(t) - h_1(t)\| \leq \gamma(t) \int_0^t \beta(s) ds$$

(11)

where $|a|_{\min}$ is the minimum over the absolute values of the elements of vector $a$. Here, the term $\beta(t)$, which enters exponentially, is more critical than $\gamma(t)$ which impacts the separation of the two trajectories only linearly, as is the case in linear systems. In this sense, the weights associated with the states can more critically impact the state trajectories compared with the weights associated with the input. We aim to tighten this upper bound next.

A. Lower Bounding Trajectory Separations Using Matrix Measures

Previously, we presented how a gradient-based disturbance may reasonably lead the perturbed system to diverge from the nominal system. The upper bound in (11) defines the largest divergence that the input disturbance can produce, but since such estimates are norm-based, they are usually conservative. Matrix measures provide a tighter estimation of system trajectories [26], since they are not norm-based and can therefore be positive or negative. Thus, our results on sufficient conditions for existence of adversarial perturbations shall make use of matrix measures.

The matrix measure of a matrix $A$, with induced norm $\|\cdot\|$, is defined as

$$\mu(A) = \lim_{k \to 0^+} \|I + \theta A\| - \|I\|$$

and this limit always exists. Given a system

$$\dot{h} = A(t)h(t)$$

one may bound its solutions using Coppel's inequality as

$$\|h(t)\| \geq \|h(0)\| \exp \int_0^t -\mu(-A(s)) ds$$

$$\|h(t)\| \leq \|h(0)\| \exp \int_0^t \mu(A(s)) ds.$$  (12)

To be able to apply this to our system, we first start by writing (5) in terms of $e(t) = h_2(t) - h_1(t)$ as

$$\dot{e}(t) = \dot{h}_2(t) - \dot{h}_1(t) = f(h_2, x + \delta) - f(h_1, x)$$

$$= f(h_2, x) - f(h_1, x) + f(h_2, x + \delta) - f(h_2, x)$$

$$= \left[ \int_0^1 \nabla_h f(h_1 + \lambda e, x) d\lambda \right] e(t)$$

$$+ \left[ \int_0^1 \nabla_h f(h_1 + e, x + \lambda \delta) d\lambda \right] \delta(t)$$

$$= A(t)e(t) + B(t, e, \delta)\delta(t)$$

(13)

and $e(0) = 0$. Once $e$ becomes non-zero, a sufficient condition for the trajectories $h_1$ and $h_2$ to continue diverging even without disturbance (i.e., $\delta$ set to zero) is when the matrix measure $\mu(-A(t, e))$ is negative for all $t, e$. This sufficient condition is easy to obtain in analytical form, as illustrated by the following example. Having an analytical, closed-form condition provides us with a clear view of how the network parameters impact its susceptibility/robustness to input perturbations. This can ultimately lead to robust networks, by constraining the set of learnable parameters during the training phase, although we shall not consider robust training in this article.

Example 1: Let us consider the vanilla RNN architecture given by

$$h(k + 1) = \tanh(Uh(k) + Wx(k) + b)$$

which is approximated in continuous time as

$$\dot{h} = \frac{1}{T}(\tanh(Uh + Wx + b) - h).$$

Then, the term $A(t, e)$ is given by

$$\frac{1}{T} \int_0^1 \left( \text{diag}[I - \tanh^2((1 - \lambda)Uh_1 + \lambda Uh_2 + Wx + b)] \times U - I \right) d\lambda.$$
We now choose our vector norm to be the infinity norm, and define the matrix norm and the matrix measure using this norm. Then, we get

\[ 0 > \mu(-A(t, e)) \]

\[ \iff 0 > \mu\left( I - \text{diag}\left[ 1 - \tanh^2(U_h + W_X + b) \right] \right) \]

\[ \forall h \in [-1, 1]^n \]

\[ \iff 0 > \mu\left( \text{diag}\left[ 1 - \tanh^2(U_h + W_X + b) \right]^{-1} - U \right) \]

\[ \forall h \in [-1, 1]^n. \quad (14) \]

We then obtain \[ 1 - \tanh^2(U_i h + W_i x + b_i) \] and plugging this \( \delta \) as the disturbance in (13) gives us

\[ \dot{e}(t) = A(t, e) e(t) + B(t, e, \delta) \]

\[ = A(t, e) e(t) + a B(t, e, \delta) B(t, e, \delta)^T e(t) \]

\[ = [A(t, e) + a B(t, e, \delta) B(t, e, \delta)^T] e(t) \]

\[ \dot{e}(t) \equiv K(t, e) e(t). \quad (17) \]

Clearly, the second term in (17) with the adversarial disturbance \( \delta \) leads \( e(t) \) to diverge. This is because we exclude the first term, we get

\[ e(t) = a B(t, e, \delta) B(t, e, \delta)^T e(t) \]

\[ \Rightarrow e(t)^T \dot{e}(t) = a \| B(t, e, \delta) B(t, e, \delta)^T \| e(t) \]

\[ \Rightarrow \frac{d}{dt} \| e(t) \|^2 = 2 a \| B(t, e, \delta) B(t, e, \delta)^T \| e(t) \| e(t) \| ^2 \geq 0. \]

We can now summarize the discussions in this section into the following theorem:

**Theorem 1:** Let us consider an RNN represented by the \((d/dt)h(t) = f(h(t), x(t))\), with fixed initial state \(h(0)\). For any given nominal input \(x_1(t)\) and the corresponding state trajectory \(h_1(t)\), there exists some input perturbation \(\delta(t)\) such that the state \(h_2(t)\) of the perturbed system corresponding to input \(x_1(t) + \delta(t)\) monotonically diverges if the matrix measure satisfies \(\mu(-A(t, e)) < 0\).

The procedure to compute this adversarial disturbance is given in Algorithm 1. We highlight again in our algorithm that the input is read sequentially at each timestep (in line 3), as it outputs the corresponding adversarial disturbance (in line 16) without *a priori* knowledge of the entire input sequence. We would like to point out through Theorem 1 that the existence of adversarial examples in context of RNNs is better explained through the matrix measure condition and model contraction property [27], [28] rather than taking a simplified viewpoint of model linearity versus nonlinearity. Irrespective of the model nonlinearities, if \(\mu(A(t, e)) < 0\) along a nominal trajectory, then nearby trajectories converge, i.e., the model is (locally) contractive, and adversarial perturbations are less effective in steering the system away from its nominal trajectories unless one applies a perturbation of a large enough magnitude.

Finally, let us draw a connection between the “fixed-point” adversarial disturbance proposed in this section and the gradient-based disturbance (8) described in Section II-C. It can be shown that one is the limiting case of the other, as stated in the following proposition:

**Proposition 2:** The perturbation \(\tilde{\delta}\), computed using (16), approaches the gradient-based perturbation

\[ \tilde{\delta}(t) = (\nabla_x f(h_2, x_0))^T e(t) \]
Algorithm 1 “Fixed-Point” Input Disturbance

**Input:** Original input sequence $x_1$, RNN classifier $\hat{f}$, and $f$ obtained using equation (3), initial hidden state $h(0)$ for the classifier $\hat{f}$, parameters MAXITER, $\alpha$, and STEP.

**Output:** Adversarial disturbance sequence $\delta$.

1. Initialize: $k \leftarrow 0$, $h_1$, $h_2 \leftarrow h(0)$.
2. while Not end of sequence $x_1$ do
   3. Get input $x_1(k)$ at the current timestep $k$;
   4. $\delta \leftarrow 0$;
   5. $e \leftarrow h_2 - h_1$;
   6. if $e$ equals 0 then
      7. $e \sim U(0, 1)$, // Set a random value
   8. end
   9. // Fixed-pt iteration to solve (16)
   10. for $i \leftarrow 0$ to MAXITER or until $\delta$ converges do
      11. $B \leftarrow 0_{n\times n}$;
      12. for $\lambda = 0$ to 1 by STEP do
      13. $B \leftarrow B + \nabla_x f(h_2, x_1(k) + \lambda \delta) \times \text{STEP}$;
      14. end
      15. $\delta \leftarrow a B^T e$;
      16. $\delta(k) \leftarrow \delta$; // Adversarial disturbance at current timestep $k$
      17. $h_1 \leftarrow f(h_1, x_1(k))$, $h_2 \leftarrow f(h_2, x_1(k) + \delta)$;
      18. $k \leftarrow k + 1$;
   19. end
   20. $N \leftarrow k$; // Length of sequence
21. return Sequence $\delta(i)$, $i = 0, 1, ..., N - 1$.

in direction, i.e., $\delta \parallel \delta$.

**Proof:** We start by looking at the direction of $\delta$ satisfying (16) as $\alpha$ approaches zero. Clearly, since $B(t, e, \cdot)^T e(t)$ is a bounded function, $a \to 0 \implies \delta \to 0$ which by continuity of $B$ implies $B(t, e, \delta)^T e(t) \to B(t, e, 0)^T e(t)$, and this further implies $\delta \to aB(t, e, 0)e(t)$.

Next, we note that by definition, $B(t, e, 0) = \int_0^1 \nabla_x f(h_2, x) d\lambda = \nabla_x f(h_2, x)$. Thus, we get $a \to 0 \implies \delta \to a \nabla_x f(h_2, x)^T e(t)$. This means $\delta \parallel \nabla_x f(h_2, x)^T e(t)$ as $a \to 0$.

This to a certain extent helps experimentally corroborate the proofs and results in Proposition 1 and Theorem 1, since gradient-based perturbation in Section II-C was already shown to be an effective adversarial attack. However, more rigorous experiments and comparisons shall follow later in Section IV.

The next section focuses on simplifying the computation of the fixed-point disturbance and presents some analysis on how this new disturbance converges to the manifold given by (16), while leading $\|e\|$ to grow.

C. Dynamic Computation of Adversarial Disturbance

The dynamics of error $e(t)$ driven by the input disturbance $d(t)$ is described by a continuous-time differential equation (13), whereas the input disturbance (16) needs to be computed at each (continuous) time instant $t$ by solving the implicit algebraic equation (16). Although this computation can be done in a straightforward manner using the globally converging discrete-time difference equation (15) if performed at a timescale faster than the error dynamics, this discrepancy between continuous versus discrete dynamics needs to be handled in a more formal manner.

Our aim in this section is to derive an ordinary differential equation (ODE) $(d\delta/dt) = \phi(t, e, \delta)$ that determines the evolution of adversarial disturbance in continuous time, consistent with the continuous error dynamics. We would ideally want the solution of this ODE to evolve in the manifold given by (16). One way to achieve this is perhaps with sliding mode controller $u = \phi(t, e, \delta)$ that takes $\delta(t)$ to the manifold (16) in finite time and keeps it on that manifold thereafter. However, such a controller would not be very encouraging as far as practical implementation is concerned, since we would need to compute gradients of each element of matrix $B$.

We start with differential equation (13), along with the ODE

$$\frac{d\delta}{dt} = aB(t, e, \delta)^T e - \delta$$

where $\epsilon$ is a positive constant. Let $\delta(t) = h(t, e)$ be the explicit solution of the (16), and $\delta(t, e)$ be the solution of (18). Note that by setting $\epsilon = 0$, one obtains what is called in control theory a “singularly perturbed” system [24] described by the original set of (13) and (16). So, it is reasonable to expect that disturbance $\delta(t, e)$ would also act as an adversarial disturbance for sufficiently small values of $\epsilon$. We can now introduce a new variable $y(t, e) = \delta(t, e) - h(t, e)$. Then, the following holds.

**Theorem 2:** Given the systems (13) and (18)

$$\dot{e}(t) = A(t, e)e(t) + B(t, e, \delta)\dot{\delta}(t)$$

$$\epsilon\dot{\delta}(t) = aB(t, e, \delta)^T e - \delta$$

the term $y(t, e)$ satisfies the exponentially decaying bound $\|y(t, e)\| \leq k_1 \exp(-(k_2/k_3) t) \|y_0\| + k_3 \epsilon$ for all $t > 0$, globally, i.e., for any initial condition $\|y_0\|$, and uniform constants $k_1, k_2, k_3 > 0$.

**Proof:** See Appendix.

**Corollary 1:** Given any time $T > 0$, $\eta > 0$ and initial condition $y_0$, there exists a $c'$ such that $\|y(t, e')\| \leq \eta$ for all $t > T$.

Let us now look at how disturbance $\delta(t, e)$ affects term $\|e(t)\|$. We recall that our goal is to find adversarial disturbances to the input such that the two trajectories $h_1$ and $h_2$ diverge. Focusing just on the disturbance term, we have

$$\frac{1}{2} \frac{d}{dt} \|e\|^2 = e^T \dot{e} = e^T B(t, e, \delta(t, e)) \dot{\delta}(t, e)$$

$$= e^T B(t, e, \delta(t, e)) [aB(t, e, \delta(t, e))^T e + \delta(t, e) - aB(t, e, \delta(t, e))^T e]$$

$$= e^T B(t, e, \delta(t, e))$$

$$\times [aB(t, e, \delta(t, e))^T e + y(t, e)$$

$$+ h(t, e) - aB(t, e, y(t, e) + h(t, e))^T e].$$

From the definition of $h(t, e)$, we know that $h(t, e) = aB(t, e, h(t, e))^T e$. Furthermore, we choose a uniform $a$ small
enough so that \( a\|((dB(t, e, z)^T e)/dz)\| \leq (1/2) \) (such a choice is always possible, following the proof of Proposition 1). One can then obtain the bound following the multivariate mean-value theorem [25]:
\[
\|aB(t, e, h(t, e))^T e - aB(t, e, y(t, e) + h(t, e))^T e\|
\leq \max_{z \in \mathbb{R}^n} a \left| \frac{dB(t, e, z)^T e}{dz} \right| \cdot \|y(t, e)\|
\leq \frac{1}{2} \|y(t, e)\|.
\]
(20)

Thus, substituting \( h(t, e) \) into the last line of (19) and using the bound in (20), we obtain
\[
\frac{1}{2} \frac{d}{dt}\|e\|^2
= e^T B(t, e, \delta(t, e)) \left[ aB(t, e, \delta(t, e))^T e + y(t, e) + h(t, e) \right]
= e^T B(t, e, \delta(t, e))
\times \left[ aB(t, e, \delta(t, e))^T e + y(t, e) + aB(t, e, h(t, e))^T e + h(t, e) \right]
\geq \alpha \|B(t, e, \delta(t, e))^T e\|^2 - \|B(t, e, \delta(t, e))^T e\| \cdot \|y(t, e)\|
- \|B(t, e, \delta(t, e))^T e\| \cdot \|y(t, e)\|
- \|B(t, e, \delta(t, e))^T e\| \cdot \left( \frac{1}{2} \|y(t, e)\| \right)
= \|B(t, e, \delta(t, e))^T e\| \left( aB(t, e, \delta(t, e))^T e - \frac{3}{2} \|y(t, e)\| \right).
\]
(21)

We want the right-hand side of the inequality above to be non-negative and remain so after some point of time. From Corollary 1, the term \( \|y(t, e)\| \) can become arbitrarily small at an exponential decay rate, if we choose \( \epsilon \) small enough. This decay of \( y(t, e) \) is independent of \( e(t) \). So if there exists some \( \mu_1, \mu_2 > 0 \) such that \( \|e\| > \mu_1 \Rightarrow \|aB(t, e, \delta(t, e))e\| \leq \mu_2 \), then perhaps we can find an invariant set of the form \( \{t \mid \|e\| > \mu_1, t > t' \} \) with \( \|e\| \) increasing on this set. Indeed, such an argument can be made if we assume that the following condition is satisfied:

**Assumption 1**: \( B(t, e, 0)^T e = f_s(h_1(t) + e, x(t))^T e \) is non-zero for all \( t > 0 \) and \( e \neq 0 \).

Then, we can assure that the right-hand side of (21) eventually becomes strictly positive because of the following. Let us say there exist a time interval \([t', t'']\) and \( \mu_1 > 0 \) (independent of parameter \( e \)), such that \( \|e(t)\| \geq \mu_1 \forall t \in [t', t''] \). We recall that \( \|e(t)\| \) and \( \|\dot{e}(t)\| \) are bounded from above due to RNN dynamics, and we denote these bounds by \( e_{\text{max}} \) and \( e'_{\text{max}} \), respectively. Let us define \( \mu_2 \) on a compact set \( \mathcal{C} = \{(t, e) \mid t \in [t', t'' + \tau], \|e\| \in [\mu_1, e_{\text{max}}]\} \) as
\[
\mu_2 \doteq \min_e \left\{ \|\delta\| \mid \delta = aB(t, e, \delta)^T e \right\}.
\]

Note that by Assumption 1, \( \mu_2 \) has to be strictly greater than zero. Thus, we have \( a\|B(t, e, h(t, e))^T e\| = \|h(t, e)\| \geq \mu_2 > 0 \) in the set \( \mathcal{C} \). Now, using triangular inequality with (20), it follows that if \( a\|B(t, e, h(t, e) + y(t, e))^T e\| \geq \|B(t, e, h(t, e))^T e\| - (1/2)\|y(t, e)\| \). Thus, in the compact set \( \mathcal{C} \), we have
\[
a\|B(t, e, h(t, e) + y(t, e))^T e\| \geq \mu_2 - \frac{1}{2} \|y(t, e)\|.
\]

From Corollary 1, we can always choose an \( \epsilon \) small enough such that \( \|y(t, e)\| < (1/4)\mu_2 \forall t > t' \). With such a choice of \( \epsilon \), we obtain \( a\|B(t, e, y(t, e))^T e\| - (3/2)\|y(t, e)\| \geq \mu_2 - 2\|y(t, e)\| \geq (\mu_2/2) > 0 \) in the compact set \( \mathcal{C} \). From the last line of (21), this means that
\[
\frac{1}{2} \frac{d}{dt}\|e(t)\|^2 > 0
\]
whenever \( e(t) \) is inside the compact set \( \mathcal{C} \), which means that the term \( \|e\| \) increases monotonically inside \( \mathcal{C} \). Thus, one can show that when \( \|e(t)\| \geq \mu_1 \) for \( t \in [t', t''] \), it remains \( \|e(t)\| \geq \mu_1 \) for the entire interval \( t \in [t', t'' + \tau] \) and increases monotonically in this time interval. We concisely present the preceding analysis as the following theorem.

**Theorem 3**: If 1) Assumption 1 holds, i.e., \( B(t, e, 0)^T e = 0 \) only when \( e = 0 \) and 2) matrix measure \( \mu(-A(t, e)) < 0 \).

Then, for every \( T > 0 \), there exists an \( \epsilon > 0 \) such that \( \|e(t)\| \) strictly monotonically diverges for a time interval of length \( T \), once it leaves the equilibrium point \( e = 0 \).

**Remarks on Assumption 1**: The condition that \( B(t, e, 0)^T e = f_s(h_1(t) + e, x(t))^T e \) only for \( e = 0 \) is mild in the case when the input dimension of the system is greater than or equal to the state dimension. For example, in the vanilla RNN architecture of Example 1, we have
\[
B(t, e, 0)^T e = \frac{1}{T} W^T \text{diag}(I - \tanh^2(Uh_2 + Wx + b))^T e.
\]
In that case, Assumption 1 can be easily verified to hold if the weight matrix \( W \) is of full rank. However, when the input dimension is smaller than the state dimension (this may happen easily in multi-layer RNNs, for example), verifying this assumption is not as straightforward, and one may need to rely on numerical simulations (such as random sampling) instead.

Algorithm 2 shows how to practically compute this disturbance in real time. Equation (18) is discretized (first-order Euler) with timestep \( \Delta T \), which is lumped with the parameter \( \epsilon \) to obtain the new parameter \( \tilde{\epsilon} \equiv (\epsilon/\Delta T) \) (in line number 12). Fig. 6 illustrates a comparative example, using the frequency discrimination task. A nominal input from class 0 is injected with disturbances computed using Algorithms 1 and 2, causing the network to misclassify it as class 1. In terms of runtime, “dynamic fixed-point attack” (18) was over four times faster than the “fixed-point attack” (16) in our Python implementation. The initial condition for ODE (18) was chosen to be zero, and the corresponding solution is denoted by \( \delta_0(t, e) \) in the figures. One can verify that the bound on \( \|y(t, e)\| \), described in Theorem 2, indeed holds globally by comparing \( \delta_0(t, e) \) to the solutions of ODE (18) starting at different initial conditions, as shown in Fig. 7.
**Algorithm 2 “Dynamic Fixed-Point” Input Disturbance**

**Input:** Original input sequence $x$, RNN classifier $f$, and $f$ obtained using equation (3), initial hidden state $h(0)$ for the classifier $f$, parameters $\bar{\epsilon}$, $\alpha$, and STEP.

**Output:** Adversarial disturbance sequence $\delta$.

1. Initialize: $k \leftarrow 0$, $h_1, h_2 \leftarrow h(0)$, $\delta \leftarrow 0$.
2. **while** Not end of sequence $x_1$ **do**
3. Get input $x_1(k)$ at the current timestep $k$;
4. $e \leftarrow h_2 - h_1$;
5. **if** $e$ equals 0 **then**
6. $e \sim U(0, 1)^n$; // Set a random value
7. **end**
8. $B \leftarrow 0$; // Compute $B$ using eq. (13)
9. // Num integration STEP = 0.1
10. **for** $\lambda \leftarrow 0$ to 1 by STEP **do**
11. $B \leftarrow B + \nabla_x f(h_2, x_1(k) + \lambda \delta) \ast$ STEP;
12. **end**
13. // Discretize equation (18)
14. $\delta(k) \leftarrow (1 - \bar{\epsilon}) \ast \delta + \bar{\epsilon} \ast (\alpha B^T e)$;
15. $\delta \leftarrow \delta(k)$; // Adversarial disturbance
16. $h_1 \leftarrow \bar{f}(h_1, x_1(k))$, $h_2 \leftarrow \bar{f}(h_2, x_1(k) + \delta)$;
17. $k \leftarrow k + 1$;
18. **end**
19. $N \leftarrow k$; // Length of sequence
20. **return** Sequence $\delta(i), i = 0, 1, ..., N - 1$.

### D. Optimal Adversarial Disturbance Computation

We have already demonstrated in Figs. 4 and 5 with our simple frequency discriminator example that the gradient-based disturbance (8) can adversarially perturb inputs from both the classes. But it may be of interest to explore optimally computed disturbances to evaluate how the approaches proposed in Sections II-C, III-B, and III-C compare. Computation-wise, our proposed approaches are clearly cheaper, and disturbances can be injected in “real time” since they only require the states and nominal input at the current timestep instead of the entire nominal state trajectory and input sequence. However, it is not clear how conservative our proposed method is in terms of the size of the disturbance needed to fool the classifier. Toward that end, we present a formulation for computing disturbances optimally and demonstrate it using the frequency discrimination task of Fig. 2.

We first remind ourselves that the adversarial attack problem for RNNs can be reframed as a control synthesis problem: Given a nominal input sequence $x_n(t)$ for which we desire to generate an adversarial input $d(t)$, we can consider the system (4) as a nonlinear, time-varying control system with disturbance $d(t)$ as the control input. With such a system, we can use well-developed off-the-shelf computational tools from optimal control theory to generate adversarial disturbances. More concretely, we can solve the following:

$$\min_{\delta(t)} - V(h(T)) + \int_0^T \delta(s)^T R(s)\delta(s)ds$$

s.t.:

$$\frac{dh}{dt} = \bar{f}(h, \delta) = f(h(t), x_n(t) + \delta(t))$$

---

**Fig. 6.** (a) “Fixed-point” disturbance (dashed black line) computed using (16) versus the dynamically computed disturbance (solid red line) using (18). (b) Solution of ODE (18) converges to the solution of the algebraic equation (16) as described in Theorem 2. (c) Green line shows that the class 0 nominal signal is correctly assigned a low confidence for class 1 ($\approx 0.05$) but assigned to class 1 with high confidence ($>0.99$) after adding the input disturbances (dashed-black line = “Fixed-point disturbance” and solid-red line = “Dynamic fixed-point disturbance”).

**Fig. 7.** Solutions of ODE (18) are computed for 1000 randomly sampled initial condition from a uniform distribution $U(-1, 1)$. All these $\delta(t, \epsilon)$ were able to fool the network.
To force the adversarial disturbance to be bounded within some prescribed $\epsilon > 0$, we reparameterize the perturbation term as $\epsilon \tanh(\delta(t))$ in place of $\delta(t)$.

Picking $R = 1$ and $\epsilon = 0.15$, we obtain these optimal perturbations for misclassifying a randomly generated input signal belonging to class 1 as class 0 by solving problem (22) using the iterative linear quadratic regulator (ILQR) technique [29] for nonlinear optimal control. The corresponding gradient-based perturbation given by (8) is also computed. Fig. 8 shows the results of the comparison. The sign change in the gradient-sign-based perturbation appears to coincide with the optimal perturbation and performs equally well in leading the network to misclassify. It is interesting to note in Fig. 8(c) that the gradient-sign-based perturbation is more greedy in driving the network to misclassify, as one might expect.

This concludes our presentation on control-system-rooted methodologies for constructing adversarial input disturbances. In Section IV, we demonstrate these approaches on a few different applications involving RNN-based classifiers.

### IV. Experiments

#### A. Frequency Discrimination Task

We now revisit the frequency discrimination task [22], [23] that we briefly introduced in Section II. In that task, a sine wave is provided as an input to the RNN which then classifies it based on its time period. Specifically, the sine inputs belong to one of the two classes: class 0 if the time period is in the interval $(5, 6)$ or class 1 if the time period lies in $(0, 5) \cup (6, 100)$. Every sine input with period $T$ has a random phase shift drawn from $U(0, T)$. The sequence length is 100, and the sinusoids are sampled at intervals of 0.1.

These inputs enter a GRU layer with hidden states of dimension 2 followed by a fully connected softmax layer with output dimension 2, which represents the classification probabilities for the two classes. This network is trained using the standard cross-entropy loss, with equal number of generated training examples for each class, and has a test accuracy of $\approx 97\%$.

Throughout Section III, we have seen in action various attacks (gradient-based, “fixed-point,” “dynamic fixed-point,” and optimal) on this model. Therefore, we shall instead compare our proposed methods here. We test our attacks on 1000 randomly generated input signals from each class. Adversarially perturbing nominal input from class 0 seems to be much easier than perturbing class 1 inputs as indicated by Fig. 9(a) and (b), where a higher attack success rate is achieved with smaller disturbances for class 0 inputs. Fig. 9(c) shows comparison of the runtimes of the three methods. The gradient-based method is the fastest requiring only a single Jacobian computation at each timestep. The dynamic fixed-point method requires a single evaluation of the $B(t, e, \delta)$ matrix, which in turn requires multiple computations of the Jacobian matrix for numerical integration, and thus takes longer than the gradient-based attack. Finally, the fixed-point attack requires multiple evaluations of the $B(t, e, \delta)$ matrix in the fixed-point iteration given by (15) and thus is the slowest, as expected.

### TABLE I

| Perturbation Method   | $\delta(k + 1)$ depends on | Compute time | Attack strength | $\delta(k)$ |
|-----------------------|---------------------------|--------------|----------------|-------------|
| Gradient based        | $B(k + 1, e(k + 1), 0)$   | ★★★          | ★★★           | ★☆☆☆        |
| Fixed-point disturbance| $B(k + 1, e(k + 1), \delta(k + 1))$ | ☆☆☆          | ★★★           | ★☆☆☆        |
| Dynamic Fixed-Point    | $B(k, e(k), \delta(k))$   | ★★★☆         | ★★★☆          | ★★★★        |

Fig. 8. (a) Nominal input sequence belonging to class 1 is shown in green. The red and black dashed lines correspond to the gradient-based and optimally perturbed inputs, respectively. (b) Perturbations added to the nominal input are shown in blue (gradient sign) and black (optimal). (c) Nominal input is correctly classified with $>99.8\%$ confidence (green), whereas perturbed inputs are misclassified with very high confidence of $>96\%$ (red = gradient sign, black = optimal perturbation).
A summary based on these results is presented in Table I. The comparisons are made based on the experiments on the frequency discrimination task, which is the simplest among all the examples in this article, and therefore the gradient-based disturbance performs on par with the other two approaches. A higher “attack strength” means a higher attack success rate, whereas the last column indicates the size of the computed perturbation in terms of the $l_2$ norm (smaller is better).

### B. MNIST Digit Classification

We shall use MNIST handwritten digit recognition as a second example, where the goal is to add small disturbances to the pixels of a greyscale image of a digit between zero and nine so that the RNN classifier misclassifies it. Although RNNs are not a traditional choice of neural networks for this task, they can nevertheless be used for image classification tasks, by inputting the image one row of pixels at a time thus resembling a sequence of inputs.

#### Table II

| Classification Accuracy on Adversarial Inputs |
|---------------------------------------------|
| Perturbation size | 0 | 0.05 | 0.1 | 0.2 | 0.3 |
| Accuracy          | 96% | 61.34% | 29.76% | 12.83% | 10.29% |

The model consists of an input layer of size 28 representing rows of a $28 \times 28$ pixels image, followed by a vanilla RNN layer with hidden state dimension of size 56. This layer then connects to a dense softmax layer of output dimension 10 corresponding to the ten classes representing digits 0–9. The network is trained on 55k images and achieves an accuracy of 96% on 10k test images.

Fig. 10 shows how the addition of the perturbations to each pixel causes the network to misclassify the images. The magnitude of the perturbation added to each pixel is upper bounded by 0.07, and the attack is untargeted.

As we increase the magnitude of pixel perturbations, we see that the network deteriorates in accuracy. Table II shows the classification accuracy on a set of 10k images obtained by adversarially perturbing the original test dataset, where the classifier accuracy drops with increase in the disturbance size.

### C. Human Activity Recognition

HAR involves the identification of actions carried out by a person using observations of themself and their environment. This typically involves inertial information from wearable sensors, and the set of activities to be identified may include walking, sitting, lying down, etc. In this experiment, we use the HAR dataset from the UCI Machine Learning Repository [30] to train a single-layer GRU network to recognize six different activities, namely, walking, walking upstairs, walking downstairs, standing, sitting, and lying down.

The input to the network is a 9-D time-series data containing filtered sensor signals from tri-axial accelerometer and gyroscope. The network consists of a single GRU layer with hidden state dimension of 50, followed by a dropout layer with a rate of 0.5. This is followed by a dense ReLU layer with an output dimension of 50, and finally a dense softmax layer with a 6-D output representing a distribution over the six activity classes. The trained network achieves a test accuracy of 93.41%.

To illustrate the fixed-point-based adversarial attack on this network, we consider a targeted attack wherein an input sequence corresponding to the activity “standing” is perturbed...
the problem setup and analysis pertained to untargeted adversarial perturbation. where we were concerned primarily with increasing $\|h_2 - h_1\|$. For targeted attacks, we steer $h_2$ toward very specific regions of the state space. With only slight modifications, the presentation in Section III can be extended to targeted attacks, as illustrated in this HAR example. Since we now are interested in the outputs $y_1$ and $y_2$ corresponding to states $h_1$ and $h_2$, the error dynamics must be multiplied on the left by the gradient of the output component $y_2$ with respect to $e$, if the target class is $j$, as follows:

$$
\left( \frac{\partial y_2}{\partial e} \right)^T \dot{e} = \left( \frac{\partial y_2}{\partial e} \right)^T (A(t, e)e + B(t, e, \delta)\delta)
$$

which leads us to the corresponding disturbance $\delta$ given by

$$
\delta = B(t, e, \delta)^T \frac{\partial y_2}{\partial e}.
$$

Table III shows targeted attack success for 300 nominal input signals sampled randomly from the test dataset. We only consider those nominal signals for attack that the network correctly classified without any perturbations, and hence the diagonal entries are left out since they are by default 100%. The untargeted success rates for each of the classes 0–5 are, respectively, 88.71%, 84.61%, 86.11%, 100%, 100%, and 20.97% (an untargeted attack is considered successful if the network misclassifies the nominal input to any one of the remaining classes). It is interesting to note the high success rates of targeted attacks “standing” to “sitting,” and “sitting” to “standing,” (even after considering error rates of 4.1% and 18.5%, respectively, for unperturbed inputs). This is consistent with the fact that sitting and standing are very close to one another, as physical activities.

**D. IMDb Review Sentiment Analysis**

Sentiment analysis aims at interpreting the subjective information such as sentiment expressed within textual input data and classifying them into classes such as positive, negative, or neutral. In this final example, we use the Stanford Large Movie Review Dataset [31] consisting of highly polar positive and negative movie reviews on IMDb (25,000 labeled data for training and testing).

Each textual input consists of a sequence of words and special characters which are converted into real-valued vectors of dimension 50, using the pre-trained GloVe word embedding, with 6 billion tokens and vocabulary size of 400k [32]. These embedding vectors are then inputted to a single-layer LSTM with a state dimension of 128 (64 hidden and 64 memory states). A dropout level of 0.75 is applied to the output of this layer, which is followed by a final softmax layer with an output dimension equal to 2. The test accuracy of the network is 82.7%.

Before we proceed, we would like to highlight an implementation challenge that separates this particular example from the previously demonstrated examples. Although the inputs to the LSTM are real-valued vectors, the raw input at each timestep belongs to a finite set of words, which means the word embedding vectors belong to a discrete and finite set $\mathcal{W} \subset \mathbb{R}^50$.

Therefore, at any given timestep $k$, the computed adversarial input $\tilde{x}(k)$ corresponding to nominal input $x(k) \in \mathcal{W}$ to the LSTM layer may not belong to set $\mathcal{W}$. To overcome this issue, we may use a heuristic similar to [11], wherein after computing $\tilde{x}(k)$, we pick the adversarial input to be equal to

$$
\arg \min_{z \in \mathcal{V}} \left( \|z - x(k)\| \right)
$$

where $\mathcal{V} = \{v \in \mathcal{W} | (v - x(k))^T (\tilde{x}(k) - x(k)) \geq 0 \}$, which simply means that we pick an embedding vector closest to the
Fig. 12. Obtaining the set of “replacement words” for each word in the original review.

computed adversarial input that aligns with the perturbation direction. A second more important challenge, however, is to ensure the new adversarial input, when finally mapped back to a sentence, remains reasonably readable and fits well with the rest of the sentence. We achieve this by incorporating the language model GPT-2 [33], as well as part-of-speech (POS) tagging provided within the NLTK package [34] to refine the set \( V \) so that newly generated adversarial sentence appears linguistically correct. Fig. 12 explains how the set of possible replacement words at each timestep \( k \) is generated. The set \( S \) consists of words that are similar to and contextually fitting replacements for the current word \( x(k) \), given \( \tau \) previous words \( x(k-1), x(k-2), \ldots, x(k-\tau) \) in the original review. We choose \( \tau \) to be equal to 4 for our experiments. Furthermore, since words in the set \( S \) are sorted by relevance, we can pick the top \( M \) words in \( S \), denoted by \( S_M \). \( M \) is chosen to be 20 in our experiment. Given the computed \( \hat{x}(k) \) obtained from (16), we then pick the replacement word to the original word \( x(k) \) to be

\[
\arg\min_{z \in V} \{ \|z - x(k)\| \}
\]

where the set \( V \) now is \( \{ v \in S_M \mid (v - x(k))^T(\hat{x}(k) - x(k)) \geq 0 \} \). Note that for this particular application, since the set of admissible perturbations \( S_M \) is a finite set, one may directly search for the best replacement word for \( x(k) \) by looking at the effect that each of the words in \( S_M \) has on the current output \( k \) of the classifier. Our attack success rate is 47% with 22.5% of the words changed on average.

Tables IV and V illustrate adversarial text generation for sentiment classification. The original reviews are correctly classified with a very high confidence. The perturbations in this case are the replacement words, shown as orange-colored text in the original review and blue-colored text in the perturbed review. The perturbed review is then classified as negative, with very high confidence.

Looking closer at the original and perturbed reviews in Table IV, the first occurrence of the word “bad” in the original negative review is replaced by “fantastic;” however, the attack does not simply replace “negative” sounding words with positive words as indicated the other two occurrences of the word “bad” in the original review. The perturbed review still retains negative sounding parts of the original review, and yet is able to fool the LSTM classifier.

In these experiments, we have chosen to create our perturbations by replacing words in the original review by other, similar words that fit the context of the overall sentence. There still seems to be instances where our choice of replacement words does not lead to meaningful sentences, however. Other works in literature, such as [35], [36], have considered introducing perturbations into the reviews by incorrectly spelling out the original words. Since all the misspelled words get mapped via word embedding to the same vector representing a “unknown token” (denoted as “unk”), the goal is then to find positions...
in the original review where we can place this “unk” vector to adversarially impact the network. The original words in those positions are then simply replaced by their misspelled versions. Such perturbations can, however, be easily detected by any spell-checking software, but still are an interesting type of textual perturbation.

We would like to stress our algorithm iterates through the original review only once and replaces words in the original review (if at all) as it sees them in sequential order, unlike the algorithm proposed in [11], which iterates through the entire sentence, changing one word per iteration until the review is misclassified.

Remarks on Robustification: Due to linear scaling of our algorithms with the length of input sequences, one can efficiently augment the training dataset to include adversarial examples and re-train the network to make it more robust. This is a well-known robustification technique, referred to as adversarial training [37], [38]. Alternatively, one may impose explicit constraints on the weights of the network during regular training, so that the matrix measure μ(−A) is large, thereby making adversarial attacks less effective in accordance with Theorem 1. An interesting question that we want to pursue in our future work is whether adversarial training achieves robustness by implicitly encouraging the matrix measure of μ(−A) to be large.

V. Conclusion

Adversarial examples for neural-network-based classification have received continued interest among the learning community and yet have lacked adequate analytical treatment. In this article, we have provided sufficient conditions for existence of adversarial perturbations to any given input sequence to RNNs. Such perturbations can be constructed easily, and under a limiting condition are equivalent to a closed-form, gradient-based perturbation. Furthermore, since our formulation and analysis are inspired by control theory, our proposed adversarial perturbations can be constructed dynamically, at each time step, as the RNN gets its input sequentially, which is advantageous in two major ways. First, we do require to know the entire input sequence a priori for computing these perturbations. Second, our proposed method for crafting these adversarial additive perturbations to an input sequence scales linearly with the length of the sequence. In addition, we take advantage of our dynamical system-based approach to show how optimal control may be used for computing adversarial perturbations. We illustrate this with some classification examples with varying complexities, in terms of architecture, and size of inputs and hidden states.

For our next work, we are interested in applying the results of this article toward developing robustification techniques for RNNs, through adversarial training with examples generated efficiently with our approach, as well as using sufficient conditions from Theorem 1 directly to impose constraints on the network weights and biases during training. Along similar lines, we would like to rigorously experiment across different networks (i.e., with different architectures, hidden state dimensions, number of layers, regularization, initial weight/random seed, etc.) to test whether or not some network structures are inherently more robust than others, and if so, why. Finally, we wish to pursue the question of generalization of our attacks to different networks, which is not an uncommon observation [1], [12]. Our attacks currently rely on complete knowledge of the network, and therefore, such a generalization could allow practical extension of our work to black-box or partially known models.

APPENDIX

Proof of Theorem 2: We start by writing the dynamics of variables y(t, e) = δ(t, e) − h(t, e) and e(t) as follows:

\[
\frac{de}{dt} = F(t, e, y) \triangleq A(t, e)e(t) + B(t, e, y + h(t, e))(y + h(t, e))
\]

\[
\frac{dy}{dt} = G(t, e, y, e)
\]

\[
\delta(t, e, y, e) \triangleq aB(t, e, y + h(t, e))T e - y - h(t, e) - \epsilon \frac{\partial h}{\partial t}(t, e) - \epsilon \frac{\partial h}{\partial y}(t, e)F(t, e, y).
\]

Let us consider t and e(t) appearing in G(t, e, δ, e) to be “frozen parameters” and look at the evolution of y on a fast timescale τ = e t

\[
\frac{dy}{d\tau} = G(t, e, y, 0).
\]

If we now take the Lyapunov function V(y) = (1/2)||y||^2, then along the solution y(τ) of (23), we have

\[
\frac{dV}{d\tau} = y(\tau)^T \frac{dy}{d\tau} = y(\tau)^T G(t, e, y, 0)
\]

\[
= -||y||^2 + y^T(aB(t, e, y + h(t, e))T e - h(t, e))
\]

\[
= -||y||^2 + y^T(aB(t, e, y + h(t, e))T e - aB(t, e, h(t, e))T e - aB(t, e, h(t, e))T e)
\]

\[
\leq -||y||^2 + ||y|| \cdot ||aB(t, e, y + h(t, e))T e||
\]

\[
\leq -\epsilon ||y||^2 + \frac{1}{2}||y||^2 = -\frac{1}{2}||y||^2 = -V
\]

which implies that y(τ) is uniformly exponentially stable and satisfies the inequality ||y(τ)|| ≤ ||y(0)|| exp(-1/2)τ. Using the (easily verifiable) fact that (∂/∂y)G(t, e, y, 0) has bounded partial derivatives with respect to t and e, along with the fact that G(t, e, 0, 0) = 0 for all t and e, we can obtain the following bound:

\[
\left| \frac{\partial G}{\partial y}(t, e, y, e) \right| = \left| \frac{\partial G}{\partial y}(t, e, y, e) - \frac{\partial G}{\partial y}(t, e, 0, 0) \right|.
\]

Thus, we can use [24, Lemma 9.8] on slowly varying systems, to conclude the existence of a Lyapunov function V(t, e, y) such that

\[
c_1||y||^2 \leq V(t, e, y) \leq c_2||y||^2
\]

\[
\frac{\partial V}{\partial y}G(t, e, y, 0) \leq -c_3||y||^2
\]
Taking the time derivative of $V(t, e, y)$, we get
\[
\frac{dV}{dt}(t, e, y) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial e} F(t, e, y, \epsilon) + \frac{1}{\epsilon} \frac{\partial V}{\partial y} G(t, e, y, \epsilon)
\]
\[
\leq \frac{\partial V}{\partial t} + \frac{\partial V}{\partial e} F(t, e, y, \epsilon) + \frac{1}{\epsilon} \frac{\partial V}{\partial y} G(t, e, y, 0) + \frac{1}{\epsilon} \frac{\partial V}{\partial y} (G(t, e, y, \epsilon) - G(t, e, y, 0)).
\]
Now, using the estimates $\|F(t, e, y, \epsilon)\| \leq k'$ and $\|G(t, e, y, \epsilon) - G(t, e, y, 0)\| \leq \epsilon L_3$, along with the norm bounds in (24), one obtains
\[
\frac{dV}{dt}(t, e, y) \leq c_5 \|y\|^2 + c_6 k' \|y\|^2 - c_3^2 \|y\|^2 + L_3 c_4 \|y\|/\epsilon.
\]
if we choose $\epsilon \leq (c_3/(2c_5 + 2c_6 k'))$. Thus, from the above inequality, one can bound $y(t, \epsilon)$ as follows:
\[
\|y(t, \epsilon)\| \leq \sqrt{\frac{c_1}{c_2}} \exp \left( -\frac{c_3}{4c_2} t \right) \|y_0\| + \frac{2c_2 c_4 L_3}{c_1 c_3} \epsilon.
\]
for all times $t > 0$ and initial condition $\|y_0\| \leq (c_1/c_2)^{1/2}$. This completes the proof. \(\square\)

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2014, arXiv:1412.6572.

[2] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2017, pp. 39–57.

[3] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), Mar. 2016, pp. 372–387.

[4] N. Narodytska and S. P. Kasiviswanathan, “Simple black-box adversarial perturbations for deep networks,” 2016, arXiv:1612.06299.

[5] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” 2017, arXiv:1712.04248.

[6] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, “Houdini: Fooling deep structured prediction models,” 2017, arXiv:1707.05373.

[7] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, “UPSET and ANGR: Breaking high performance image classifiers,” 2017, arXiv:1707.01159.

[8] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in Proc. ACM Asia Conf. Comput. Commun. Secur. (Asia CCS), New York, NY, USA: Association for Computing Machinery, Apr. 2017, pp. 506–519.

[9] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning,” in Proc. Int. Conf. Recurrent Neural Networks, 2019, arXiv:1901.00963.

[10] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” 2016, arXiv:1607.02533.

[11] N. Papernot, P. McDaniel, A. Swami, and R. Harang, “Crafting adversarial input sequences for recurrent neural networks,” in Proc. IEEE Mil. Commun. Conf. (MILCOM), Nov. 2016, pp. 49–54.

[12] C. Szegedy et al., “Intriguing properties of neural networks,” 2013, arXiv:1312.6199.