SCALABLE NEURAL LEARNING FOR VERIFIABLE CONSISTENCY WITH TEMPORAL SPECIFICATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Formal verification of machine learning models has attracted attention recently, and significant progress has been made on proving simple properties like robustness to misclassification under perturbations of the input features. In this context, it has also been observed that folding the verification procedure into training makes it easier to train verifiably robust models. In this paper, we extend the applicability of verified training by extending it to (1) recurrent neural network architectures and (2) complex specifications that go beyond simple adversarial robustness, particularly specifications that capture temporal properties like requiring that a robot periodically visits a charging station or that a language model always produces sentences of bounded length. Experiments show that while models trained using standard training often violate desired specifications, our verified training method produces models that both perform well (in terms of test error or reward) and can be shown to be provably consistent with specifications.

1 INTRODUCTION

While deep neural networks (DNNs) have shown immense progress on diverse tasks (Sutskever et al., 2014; Mnih et al., 2015; Silver et al., 2016), they are often deployed without formal guarantees of their correctness and functionality. Their performance is typically evaluated using test data, or sometimes with adversarial evaluation (Carlini & Wagner, 2017; Uesato et al., 2018; Ebrahimi et al., 2018; Wang et al., 2019). However, such evaluation does not provide formal guarantees regarding the absence of rare but possibly catastrophic failures (Administration; Board; Ross & Swetlitz, 2018).

Researchers have therefore started investigating formal verification techniques for DNNs. Most of the focus in this direction has been restricted to feedforward networks and robustness to adversarial perturbations (Tjeng et al., 2017; Raghunathan et al., 2018b; Ko et al., 2019). However, many practically relevant systems involve DNNs that lead to sequential outputs (e.g., an RNN that generates captions for images, or the states of an RL agent). These sequential outputs can be interpreted as real-valued, discrete-time signals. For such signals, it is of interest to provide guarantees with respect to temporal specifications (e.g., absence of repetitions in a generated sequence, or that a generated sequence halts appropriately). Temporal logic provides a compact and intuitive formalism for capturing such properties that deal with temporal abstractions.

Here, we focus on Signal Temporal Logic (STL) (Donzé & Maler, 2010) as the specification language and exploit its quantitative semantics to integrate a verification procedure into training to provide guarantees with regard to temporal specifications. Our approach builds on recent work (Mirman et al., 2018; Gowal et al., 2018), which is based on propagating differentiable numerical bounds through DNNs, to include specifications that go beyond adversarial robustness. Additionally, we propose extensions to Mirman et al. (2018); Gowal et al. (2018) that allow us to train auto-regressive GRUs/RNNs to certifiably satisfy temporal specifications. We focus on the problem of verified training for consistency rather than post-facto verification. To summarize, our contributions are as:

- We present extensions to Mirman et al. (2018); Gowal et al. (2018) that allow us to extend verified training to novel architectures and specifications, including complex temporal specifications. To handle the auto-regressive decoder often used in RNN-based systems, we leverage differentiable approximations of the non-differentiable operations.

- We empirically demonstrate the applicability of our approach to ensure verifiable consistency with temporal specifications while maintaining the ability of neural networks to achieve high accuracy on the underlying tasks across domains. For supervised learning, verified training on the train-data enables us to provide similar verification guarantees for unseen test-data.
- We show that verified training results in robust DNNs whose specification conformance is significantly easier to guarantee than those trained adversarially or with data augmentation.

2 Related Work

Here, we discuss the most closely related approaches covering AI safety and DNN verification.

2.1 Neural Network Verification

There has been considerable recent progress on developing techniques for neural network verification, starting with the pioneering work of Katz et al. (2017), where a satisfiability-modulo-theories (SMT) solver was developed to verify simple properties for piecewise linear deep neural networks. Subsequently, several mature solvers that rely on combinatorial search (Tjeng et al., 2017; Dutta et al., 2017; Bunel et al., 2018) have helped scale these techniques to larger networks.

More recently, verification of neural networks using incomplete over-approximations – using dual based approaches (Dvijotham et al., 2018), and propagating bounds through the neural network (Gehr et al., 2018; Wang et al., 2018c; Weng et al., 2018a; Singh et al., 2019) – has emerged as a more scalable alternative for neural network verification. Raghunathan et al. (2018a); Wong et al. (2018) showed that folding the verification procedure into the training loop (called verified training) enables us to obtain stronger guarantees. Building on this line of work, Gowal et al. (2018) showed that training with simple interval bounds, with carefully chosen heuristics, is an effective approach towards training for verifiability. However, the focus of the above mentioned works on verified training is limited to adversarial robustness properties, and feedforward networks with monotonic activation functions. In this work, we build on Mirman et al. (2018); Gowal et al. (2018) to consider richer specifications that capture desired temporal behavior, and novel architectures with non-differentiable components. Table 1 compares different methods developed for verified training.

Table 1: Comparison of methods developed for training for consistency with specifications.

	Beyond ReLU	Continuous Input Features	Components with Gating	Auto-regressive	Temporal Specifications
Raghunathan et al. (2018a); Wong et al. (2018) Wang et al. (2018b)		✓			
Gowal et al. (2018); Mirman et al. (2018)	✓	✓			
Ghosh et al. (2018); Xiao et al. (2018) Li et al. (2017a)	✓		✓	✓	✓
Ours	✓	✓	✓	✓	✓

In independent and concurrent work, Jia et al. (2019) develop an approach for verifying LSTMs, CNNs and networks with attention-mechanism. They use a similar approach as developed in our paper to compute bounds through the softmax function, word-substitutions, and also extend bound-propagation to handle the gating mechanism. Their main focus is robustness to misclassification. In contrast, we consider complex temporal specifications, and auto-regressive architectures.

2.2 Satisfaction of temporal properties

Temporal Specifications While training networks to satisfy temporal logic specifications has been considered before, it has largely been from the perspective of encouraging RL agents to do so through a modified reward (Icarte et al., 2018b; Aksaray et al., 2016; Hasanbeig et al., 2018; Icarte et al., 2018a; Sadigh et al., 2014; Wen et al., 2017; Li et al., 2017a). The temporal logic specification is used to express the task to be performed by the agent, rather than as a verifiable property of the system. Ghosh et al. (2018) encourage specification conformance during training by regularizing with a loss arising from the desired specification. However, the specification is enforced on specific inputs and does not guarantee that the property holds across continuous regions of the input space (e.g., all

inputs in the neighborhood of a given image). In contrast, we train DNNs to verifiably satisfy rich temporal specifications over large/continuous sets of inputs. Further, we note that there is work on falsifying STL specifications using stochastic optimization(Annpureddy et al., 2011; Donzé, 2010), however our focus here is on verified training.

Safe RL Safe exploration methods (Garcıa & Fernández, 2015) consider explorations that do not visit unsafe states. Temporal logics, in general, permit richer specifications of desired temporal behaviors than avoiding unsafe states. Junges et al. (2016) synthesize a scheduler that limits the agent explorations to safe regions of the environment specified using probabilistic computation tree logic (PCTL). An alternative mechanism, called shields, monitors actions of an agent and restricts it to a safe subset ensuring conformance to specifications in linear temporal logic (Alshiekh et al., 2018) or PCTL (Jansen et al., 2018). Instead of using external mechanisms to restrict the agent choices, we incorporate temporal logic objectives into training and achieve verified training of our agents. Furthermore, our work is not restricted to training verifiable RL agents. We demonstrate the generality of our approach on image captioning and language generation tasks involving RNNs.

Verification using Interpretable Policies PIRL (Verma et al., 2018) and VIPER (Bastani et al., 2018) extract interpretable policies that are also amenable to verification. These approaches first learn DNN agents and then use imitation learning to extract interpretable/verifiable policies. Wang et al. (2018a) analyze RNNs by distillation to a Deterministic Finite Automaton (DFA), and demonstrate successful distillation on recognizing Tommika Grammar. However, this distillation is often difficult, and remains an open challenge for vision/language processing tasks. Further, there are no guarantees about the original DNN. In contrast, we focus on guarantees for the DNN itself.

3 Formulating Temporal Constraints on ML Models with Signal Temporal Logic

We consider the problem of verified training of a model with respect to a desired property. In what follows, we describe how signal temporal logic provides a formalism to describe properties of interest.

3.1 Preliminaries

We use Signal Temporal Logic (STL) (Donzé & Maler, 2010), an extension of Linear Temporal Logic (LTL) (Pnueli, 1977) that can reason about real-valued signals.

Syntax and Qualitative Semantics STL has the following syntax:

$$\varphi := \texttt{true} \,|\, q(s) \geq 0 \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \mathcal{U}_I \varphi_2 \tag{1}$$

where $\texttt{true}$ is the Boolean constant for truth, and $\neg$ and $\wedge$ are Boolean negation and conjunction operators. The symbol q is a quantifier-free non-linear real-valued arithmetic function over the vector-valued state denoted by s; the formula $q(s) \geq 0$ is called an atom. The formula $\varphi_1 \mathcal{U}_I \varphi_2$ is the *until* temporal operator, meaning φ_1 holds until φ_2 holds in the interval I. We define $\Diamond_I \varphi$ (meaning φ *eventually* holds in the interval I) as $\texttt{true}\,\mathcal{U}_I \varphi$ and $\Box_I \varphi$ (meaning φ *always* holds in I) as $\neg\Diamond_I \neg\varphi$.

In this work, we interpret STL formulae over a trace σ which is a discrete-time, vector-valued signal; σ_t denotes the value of the signal at time t. We write $(\sigma, t) \models \varphi$ to indicate that φ holds for σ at time t. An atom $q(s) \geq 0$ holds at time t if $q(\sigma_t) \geq 0$. Trivially, $(\sigma, t) \models \texttt{true}$ always holds. An until formula $\varphi_1 \mathcal{U}_I \varphi_2$, with $I = [a, b]$, holds at a time instance t if $(\sigma, t') \models \varphi_2$ for some time $t' \in [t + a, t + b]$ and $(\sigma, t'') \models \varphi_1$ for all $t'' \in [t, t']$. In the rest of the paper, we use $t + I$ to denote $[t + a, t + b]$ for $I = [a, b]$. A formula φ is said to be *satisfied* over a trace σ if $(\sigma, 0) \models \varphi$. In this work, we restrict I to be bounded-intervals of time. We refer the reader to Donzé & Maler (2010) for a detailed introduction to the semantics of STL specifications.

While bounded-time STL properties can be unrolled through time into logical properties using the Boolean conjunction and disjunction operators (Raman et al., 2015), STL provides a succinct and intuitive notation for expressing desired temporal properties. In contrast with prior work on verified training that only considers adversarial robustness (a linear constraint on the logits), we consider general specifications that assert temporal properties over input-output behaviors of neural networks. Section 3.2 lists several examples of relevant properties that can be expressed in bounded-time STL.

3.2 STL SPECIFICATIONS FOR LEARNING TASKS

To illustrate our approach, we consider three temporal properties that we want our networks to satisfy.

3.2.1 BOUNDING CAPTION LENGTH FOR IMAGE-CAPTIONING

Figure 1: MMNIST Image

Multi-MNIST images consist of non-overlapping MNIST digits on a canvas of fixed size (Figure 1). The number of digits in each image varies between 1 and 3. The task is to label the sequence of digits in the image, followed by an end of sequence token. Prior work on this task (Wang et al., 2019) has shown image-to-sequence models to be vulnerable to generating sequences longer than the true number of digits in the image, under small adversarially chosen perturbations. Here, we consider the task of training a DNN that does not output sequences longer than the desired length, while achieving similar nominal task performance.

Let $y := f(x)$ be the sequence of logits output by the RNN model when given input image x. For an image x, the termination specification is formalized as follows:

$$\forall \Delta x \in \{s : \|s\|_\infty \leq \epsilon\}.(f(x + \Delta x), 0) \models \varphi_x, \tag{2}$$

where $\varphi_x(y) := \Diamond_{[0,t_x^*]} \bigwedge_{i \neq e} (y[t]_e - y[t]_i) \geq 0$, t_x^* is the true number of digits in the image x, e is the label corresponding to the end of sequence token, $\epsilon > 0$ is the perturbation bound. Informally, this specification enforces that the end of sequence token is output no later than after the true number of digits have been output by the RNN, for all inputs within ϵ distance from a true-image.

3.2.2 VERIFYING THAT A ROBOT NEVER RUNS OUT OF CHARGE

To demonstrate our approach in the RL setting, we consider a task with a vacuum cleaning robot. We summarize this task here (See Appendix F for more details). The agent (robot) operates in a continuous domain with its location in $(x, y) \in [0, 25]^2$ (Figure 2, Appendix). The room is divided into discrete cells, and the agent gets a reward for visiting any "dirty" cell which has not been visited in the previous T_{dirt} time-steps. The agent must visit one of the recharge cells every T_{recharge} time-steps, or the episode is terminated with no further reward. The policy maps observations (of the agent location and a map of the room) to continuous velocity controls. We use f_θ to denote the result of applying the policy, parameterized by θ, followed by the environment update.

For this agent, we want to verify the specification: $\forall z \in S_\epsilon.(f_\theta(z), 0) \models \Box_{[0,T]} \Diamond_{[0,T_{\text{recharge}}]} \varphi_{\text{recharge}}$, where $\varphi_{\text{recharge}}$ corresponds to the agent being in one of the recharge cells. This specification ensures that, for a set of feasible starting positions S_ϵ, for every time-step t in $[0, T]$, the agent recharges itself at least once within T_{recharge} time-steps. See Appendix F.1 for a detailed description of S_ϵ.

3.2.3 VERIFYING GENERATED OUTPUTS FROM A LANGUAGE MODEL

A common failure mode for language models is their tendency to fall into degenerate loops, often repeating a stop-word (Wang et al., 2019). To illustrate the applicability of STL specifications in this setting, we show how to formalize the property that a GRU language model does not repeat words consecutively. We call this specification bigram non-repetition. More concretely, the desired specification is that the output sequence does not contain bigram repetition amongst the 100 most frequent tokens in the training corpus vocabulary. We want to verify this property over a large set of possible conditioning inputs for the generative model. Concretely, we define an input set S of roughly 25 million prefixes generated from a syntactic template (See Appendix G.1 for details). These prefixes are input to the LM, and then we evaluate the specification on the model output.

Now, consider a prefix x and the sequence of logits y output by the recurrent GRU network f (i.e. $y = f(x)$), with $y(t)_k$ referring to the logit corresponding to the kth most-frequent token in the vocabulary at time t. A compact formal specification φ_{bigram} ruling out bigram repetition is:

$$\varphi_{\text{bigram}} := \Box_{[0,T_{\text{sample}}]} \bigwedge_{i=1,2,\ldots,100} \left(\left(\bigwedge_{j \neq i} y(t)_i \geq y(t)_j \right) \rightarrow \Diamond_{[0,1]} \neg \left(\bigwedge_{j \neq i} y(t)_i \geq y(t)_j \right) \right) \tag{3}$$

where T_{sample} denotes the length of the generated sample, in our case 10. The RNN f is required to satisfy the specification $\forall x \in S.(f(x), 0) \models \varphi_{\text{bigram}}$.

4 VERIFIABLE DNN TRAINING FOR STL SPECIFICATIONS

We consider the problem of learning a trace-valued function f_θ to verifiably satisfy a specification of the form $\forall x \in S. (f_\theta(x), 0) \models \varphi$, where input x ranges over set S, and $f_\theta(x)$ is the trace generated by f_θ when evaluated on x, θ represents the trainable parameters, and φ is an STL specification. We drop θ for brevity, and simply denote $f_\theta(x)$ as $f(x)$. Formally, our problem statement is:

Given a set of inputs S, train the parameters θ of f_θ so that $\forall x \in S. (f_\theta(x), 0) \models \varphi$, where φ is a bounded-time STL specification.

4.1 OPTIMIZATION FORMULATION OF STL VERIFICATION

For an STL specification φ, its quantitative semantics can be used to construct a function $\rho(\varphi, f(x), t)$ whose scalar valued output is such that $\rho(\varphi, f(x), t) \geq 0 \iff (f(x), t) \models \varphi$ (Donzé & Maler, 2010). In terms of the quantitative semantics, the verification problem is equivalent to showing that $\forall x \in S. \rho(\varphi, f(x), 0) \geq 0$. This verification task can be written as the optimization problem of finding the sequence of inputs x such that the sequence of outputs $f(x)$ result in the strongest violation of the specification with regard to the quantitative semantics:

$$\min_x \rho(\varphi, f(x), 0) \text{ subject to } x \in S. \tag{4}$$

If the solution to equation 4 is negative, then there exists an input leading to the violation of φ.

4.2 BOUND PROPAGATION

The optimization problem in equation 4 itself is often intractable; even in the case when the specification is limited to robustness against perturbations in a classification task, it is NP-hard (Katz et al., 2017). There are tractable approaches to bounding the problem in equation 4 (Raghunathan et al., 2018a; Dvijotham et al., 2018), but the bounds are often too loose to provide meaningful guarantees. To obtain a tighter bound tractably, interval bound propagation – which by itself provides loose bounds, but is efficient to compute – can be leveraged for verified training to give meaningful bounds on robustness under l_∞ perturbations (Mirman et al., 2018; Gowal et al., 2018). Our general approach for doing bound propagation on the function f is to use standard interval arithmetic. While this is straightforward when f is a feedforward DNN (Gowal et al., 2018), here we extend bound propagation to a richer set of (temporal) specifications and architectures. First, we highlight the novel aspects of bound propagation required for (a) auto-regressive RNNs/GRUs, (b) STL specifications.

Bound Propagation through GRUs Computing bounds across GRU cells involves propagating bounds through a multiplication operation (as a part of gating mechanisms), which can be handled by a straightforward application of interval arithmetic (Hickey et al., 2001) (see Appendix H).

Bound Propagation through auto-regressive RNNs For language modeling and image captioning, we use GRU decoders with greedy decoding. Greedy-decoding involves a composition of the `one-hot` and the `argmax` operations. Both of these operations are non-differentiable. To overcome this and compute differentiable bounds (during training), we approximate this composition with a `softmax` operator (with a low temperature T). In the limit, as $T \to 0$, the softmax operator converges to the composition `one-hot(argmax(·))` For propagating bounds through the softmax operator, we leverage that the bounds are monotonic in each of the individual inputs. Formally, given a lower ($\underline{p}$) and upper ($\overline{p}$) bound on the input p to a softmax layer (i.e., $\underline{p} \leq p \leq \overline{p}$), the lower $\underline{w}$ and upper bound($\overline{w}$) on the output can be computed as:

$$\overline{s} = \sum_{i=1}^{N} \exp \overline{p}_i, \quad \underline{s} = \sum_{i=1}^{N} \exp \underline{p}_i, \quad \Delta_i = \exp \overline{p}_i - \exp \underline{p}_i, \quad \overline{w}_i = \frac{\exp \overline{p}_i}{\underline{s} + \Delta_i} \quad \underline{w}_i = \frac{\exp \underline{p}_i}{\overline{s} - \Delta_i},$$

where p_i is the ith coordinate of p and $p \in \mathbb{R}^N$. During evaluation, the `one-hot(argmax(.))` function is used as is. Given bounds on each coordinate of p (i.e., $\underline{p} \leq p \leq \overline{p}$) and $s =$ `one-hot(argmax(`p`))`, bounds on coordinate s_i can be computed as:

$$\underline{s}_i(x) = \begin{cases} 1 & \underline{p}_i \geq \overline{p}_j . \forall j \\ 0 & \text{otherwise.} \end{cases} \qquad \overline{s}_i(x) = \begin{cases} 0 & \exists j \neq i \text{ such that, } \underline{p}_j > \overline{p}_i \\ 1 & \text{otherwise.} \end{cases}$$

We discuss bound propagation for discrete inputs in Appendix C

Bound Propagation through the specification First, we extend the quantiative semantics for STL specifications (Donzé & Maler, 2010) to allow us to reason over sets of inputs. For a STL specification φ in negation normal form (NNF) (See Appendix A for details on the quantitative semantics and conversion to NNF), we first define a *lower bound* for the quantitative semantics of φ over the set S, which we denote by $\omega_{S,f}(\varphi, 0)$. We define this bound assuming we have lower bounds on all the atoms occurring in φ. Specifically, let $\Omega_{S,f}(q,t)$ be a lower bound on $q(f(x)_t)$ over all inputs $x \in S$; in other words, at each time t we have $\forall x \in S.\, \Omega_{S,f}(q,t) \leq q(f(x)_t)$. Now, we define the lower bound on a specification φ inductively as:

- $\omega_{S,f}(\texttt{true}, t) = +\infty, \qquad \omega_{S,f}(\neg\texttt{true}, t) = -\infty, \qquad \omega_{S,f}(q(s) \geq 0, t) = \Omega_{S,f}(q,t)$
- $\omega_{S,f}(\varphi_1 \wedge \varphi_2, t) = \min(\omega_{S,f}(\varphi_1, t), \omega_{S,f}(\varphi_2, t))$
- $\omega_{S,f}(\varphi_1 \vee \varphi_2, t) = \max(\omega_{S,f}(\varphi_1, t), \omega_{S,f}(\varphi_2, t))$
- $\omega_{S,f}(\varphi_1 \mathcal{U} \varphi_2, t) = \max_{t' \in t+I} \min\Big(\omega_{S,f}(\varphi_2, t'), \min_{t'' \in [t,t']} \omega_{S,f}(\varphi_1, t'')\Big)$.

Lemma 1. *For any time t, given lower bounds $\Omega_{S,f}(q,t)$ on all the atoms $q(s) \geq 0$ in φ, we have:*

$$\forall x \in S.\, \omega_{S,f}(\varphi, t) \leq \rho(\varphi, f(x), t)$$

Corollary 1. *If $\omega_{S,f}(\varphi, t) \geq 0$, then $\forall x \in S.(f(x), 0) \models \varphi$.*

See Appendix B for proof of Lemma 1. In order to compute the lower bounds $\Omega_{S,f}(q,t)$ required for Lemma 1, given bounds on the input x, we can first compute bounds on the outputs $f(x)_t$ at each time t. For the atoms $q(s) \geq 0$ appearing in φ, given bounds on the input s we can compute bounds on $q(s)$. These bounds can then be propagated through the specification inductively.

4.3 Verified Training for STL Specifications

In this section, we describe how to train a network to satisfy an STL specification φ. The quantitative semantics $\rho(\varphi, \sigma, 0)$ gives a degree to which σ satisfies φ. First, we compute lower bounds on the values of the atoms in φ at each instance of time. Then, by application of Lemma 1, we can compute the lower bound $\omega_{S,f}(\varphi, 0)$ satisfying $\forall x \in S.\, \omega_{S,f}(\varphi, 0) \leq \rho(\varphi, f(x), 0)$. Subsequently we optimize the lower bound $\omega_{S,f}(\varphi, 0)$ to be non-negative, thereby guaranteeing that the specification of interest holds: $\forall x \in S.\, \rho(\varphi, f(x), 0) \geq 0$.

Let L_{obj} be the loss objective corresponding to the base task, for example, the cross-entropy loss for classification tasks. Training thus requires balancing two objectives: minimizing loss on the base task by optimizing $L_{obj}(f_\theta)$, and ensuring the positivity of ω_{S,f_θ}. We can use gradient descent to directly optimize the joint loss: $L_{obj}(f_\theta) - \lambda \min\{\omega_{S,f_\theta}(\varphi, 0), \tau$, where λ is a scalar hyper-parameter, τ is a positive scalar threshold ($\tau \in \mathbb{R}_+$). The clipping avoids having to carefully balance the two losses. The quantitative semantics of an STL specification φ is a non-smooth function of the weights of the neural network, and is difficult to optimize directly with gradient descent. We find in practice that curriculum training, similar to Gowal et al. (2018), works best for optimizing the specification loss, starting with enforcing the specification over a subset $S' \subset S$, and gradually covering the entire S. Empirically, the curriculum approach means that the task performance (L_{obj}) does not degrade much.

5 Experimental Results

5.1 Sequential Captioning of Multi-MNIST Images

For this task, we perform verified training to enforce the termination specification φ_x (equation 4) on the training data as discussed in Section 4.3. Post training, for unseen test-set images, we evaluate the quantitative specification loss $\omega_{S_{x,\epsilon},f}(\varphi_x, 0)$. For an image x from the test-set, if $\omega_{S_{x,\epsilon},f}(\varphi_x, 0)$ is positive, it is guaranteed that there is no input within an l_∞ radius of ϵ around the current image that can cause the RNN to generate a longer sequence than the number of true digits in the image.

Table 2: Comparison of GRU training methods on the MMNIST task. We evaluate against the termination specification on different metrics, and also report nominal accuracy. '–' indicates a trivial verified accuracy of 0% obtained with bound propagation. The entries with *Verified Termination Accuracies* corresponding to 0.0 are those where we were able to generate adversarial examples (counter-examples) to the specification for every point in the test-set. We found that adversarial training is difficult because of the presence of the sigmoid & tanh activation functions commonly used in GRUs. To have a meaningful baseline, we performed adversarial training on an RNN (feedforward cells with ReLU activation). For $\epsilon = 0.1$, attacking the loss from Wang et al. (2019) to produce longer sequences performs better, while for the other ϵ values adversarial training with the STL quantitative loss performs better. Adversarial training performs well but is difficult to verify. At larger ϵ, verified training results in both better guarantees (specification conformance), and better nominal accuracies.

Perturbation ϵ	Training	Nominal Accuracy	Verified Termination Accuracy	Adversarial Termination Accuracy
0.1	Verifiable	94.9	**98.3**	**100.0**
	Adversarial	94.1	–	**100.0**
	Nominal	**95.9**	–	33.5
0.2	Verifiable	94.5	**98.7**	**100.0**
	Adversarial	93.3	–	**100.0**
	Nominal	**95.9**	–	20.94
0.3	Verifiable	94.4	**98.7**	**100.0**
	Adversarial	90.0	–	99.7
	Nominal	**95.9**	0.0	0.0
0.5	Verifiable	94.1	**99.0**	**100.0**
	Adversarial	75.6	–	100.0
	Nominal	**95.9**	0.0	0.0

In Tables 2 and 3, *verified termination accuracy* refers to the fraction of unseen data for which we can verify the absence of counter-examples to the termination property (equation 4). *Nominal accuracy* refers to the percentage of correctly predicted tokens – including the end of sequence token. Table 2 compares verified training with nominal and adversarial training. Verified training outperforms both adversarial and nominal training on both adversarial and verified termination accuracy metrics. The pixel values are scaled to be in the range $[0, 1]$. At perturbations of size $\epsilon = 0.5$, the images can be turned gray; however, the DNN remains robust to such large perturbations by predicting that the image has no more than a single digit at large perturbations, while maintaining nominal accuracy on clean data. This in contrast with robustness against misclassification, where it is not possible to be robust at large perturbations because the specifications for images from different classes conflict. Adversarial accuracy is evaluated with the iterative attack from Wang et al. (2019) (10000 steps).

Table 3: We train the RNN with ReLU activations from (Wang et al., 2019) to be verifiable with $\epsilon = 0.3$, and compare its verifiability with MILP based verification reported in Wang et al. (2019) at different perturbation radii. The nominal accuracy for the model trained to be verifiable is 93.9% and model trained in a standard manner is 96.4%. For larger perturbations, the MILP solver times out. '–' indicates that we were unable to certify robustness for any of the points in the test-set, for the given perturbation within the time-out window of 30 minutes.

Perturbation Radius ϵ	Training	Verification Method	Verified Termination Accuracy
0.002	Nominal	MILP	83.00
	Verifiable	Bound Prop.	**99.01**
0.02	Nominal	MILP	–
	Verifiable	Bound Prop.	**98.95**
0.3	Nominal	MILP	–
	Verifiable	Bound Prop.	**94.3**

Run-time Considerations As another baseline, we compare with verified termination accuracies from Wang et al. (2019)(Table 3). In Wang et al. (2019), the greedy-decoding and the specification are turned into a MILP-query solved with the SCIP solver (Gleixner et al., 2018). Further, we use ReLU RNNs here because GRUs are not amenable to MILP solvers. Verified training allows us to certify specification conformance for much larger perturbations (≈ 2 orders of magnitude larger).

5.2 An RL Mobile-Robot Agent

We consider the recharging specification $\varphi_{\text{recharge}}$ over a time-horizon of $T = 10$, for an agent starting within a l_∞ distance of ϵ from the center of the any of the cells (See Appendix F.1 for

details on feasible initial states). To regularize the DNN to be verifiable with regard to $\varphi_{\text{recharge}}$, the specification loss is obtained by rolling out the current policy through time, and propagating bounds through the rolled out policy and the dynamics. This assumes a deterministic dynamics model.

We compare our verifiably trained agent to both a vanilla RL agent, and an agent trained with reward shaping as in Li et al. (2017b). All agents achieve a similar reward, and we do not find specification violations for roll-outs from 10^6 random (feasible) initial states. To compare verifiability, we discretize a region within a distance of ϵ to each cell-center into 10^2 l_∞ balls, and verify with bound-propagation that the agent satisfies $\varphi_{\text{recharge}}$ for each sub-region. Agents trained with verified training are significantly more verifiable than agents trained otherwise, with little degradation in performance (Table 4), which is consistent with prior work in classification (Wong & Kolter, 2018)

Table 4: Mean/Variance performance (across 5 agents of each type) across different metrics. For each agent, reward is computed as mean across 100 episodes. ϵ is distance from the center of the grid cells, and for each ϵ we report the fraction of the cells for which we are able to certify that $\varphi_{\text{recharge}}$ holds.

Training	% of cells verified ($\epsilon = 1.0$)	% of cells verified ($\epsilon = 0.1$)	% of cells verified ($\epsilon = 0.01$)	% of cells verified ($\epsilon = 0.001$)	% of cells verified ($\epsilon = 0.0001$)	Reward
Verifiable	100.0/0.0	100.0/0.0	100.0/0.0	100.0/0.0	100.0/0.0	12.71/0.19
Standard	15.8/9.0	64.9/6.8	77.6/5.3	90.3/1.8	99.2/0.0	12.85/0.06
Reward Shaping	39.1/21.9	74.3/8.6	83.2/5.9	92.0/1.9	100.0/0.0	12.76/0.22

5.3 Language Generation

Our language model consists of a 2-layer GRU with 64 hidden nodes per layer, trained on the tiny Shakespeare corpus using a word embedding dimension of 32, and vocabulary truncated to the 2500 most frequent training words. We evaluate the model's ability to satisfy φ_{bigram}. We compare both a nominal model trained using log-likelihood, a model that randomly samples prefixes from the input space and penalizes violations to the specification, and verified training that covers the full input space (Details in Appendix G). We report test set perplexity and count of violations observed over the 25M prefixes (Table 5).

Table 5: Language model perplexity, number of failures during an exhaustive enumerative search over the 25M perturbations, and computational cost of verification (number of forward passes).

Training	Perplexity	# Failures	# Verification Cost
Verifiable	228.91	**0**	**≈2**
Sampled	174.89	**0**	2.57×10^7
Nominal	**153.63**	1.79×10^7	2.57×10^7

We find that while standard training achieves the best perplexity results, it also produces numerous specification failures. Sampling prefixes and regularizing them to avoid bigram repetition using $\rho(\varphi_{\text{bigram}}, f(x), 0)$ eliminates failures, but the overall evaluation cost of the exhaustive search is large. Verifiable training with bound propagation, by contrast, comes with a constant computational cost of ≈ 2 forward passes. This is because matrix multiplications form a significant majority of the computational cost during a forward pass, and propagating bounds through a layer of the form $y = \sigma(Wx + b)$, where σ is a monotonic activation function (e.g. ReLU, sigmoid, tanh), can be performed such that it only costs twice as much as a normal forward pass (Gowal et al., 2018).

Run-time Considerations Verification with propagating bounds can be performed in under **0.4 seconds** (including propagating bounds through the spec), while exhaustive search over 25M prefixes for specification violations takes over **50 minutes**. Further, as possible word substitutions increase, the cost for exhaustive search grows exponentially while that for bound propagation stays constant.

6 Conclusion

Temporal properties are commonly desired from DNNs in settings where the outputs have a sequential nature. We extend verified training to tasks that require temporal properties to be satisfied, and to architectures such as auto-regressive RNNs whose outputs have a sequential nature. Our experiments suggest that verified training leads to DNNs that are more verifiable, and often with fewer failures.

Future work includes extending verification/verified training to unbounded temporal properties. Another important direction is to develop better bound propagation techniques that can be leveraged for verified training. In the RL setting, an important direction is data-driven verification in the absence of a known model of the environment.

REFERENCES

US National Highway Traffic Safety Administration. Investigation pe 16-007. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. *CoRR*, abs/1609.07409, 2016. URL http://arxiv.org/abs/1609.07409.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In Parosh Aziz Abdulla and K. Rustan M. Leino (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 254–257, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19835-9.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, pp. 2494–2504, 2018.

US National Transportation Safety Board. Preliminary report highway: Hwy18mh010. https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf.

Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 4795–4804, 2018. URL http://papers.nips.cc/paper/7728-a-unified-view-of-piecewise-linear-neural-network-verification.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017.

Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In Tayssir Touili, Byron Cook, and Paul Jackson (eds.), *Computer Aided Verification*, pp. 167–170, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14295-6.

Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings*, pp. 92–106, 2010. doi: 10.1007/978-3-642-15297-9_9. URL https://doi.org/10.1007/978-3-642-15297-9_9.

Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106. Springer, 2010.

Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In Natasha Sharygina and Helmut Veith (eds.), *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pp. 264–279. Springer, 2013. doi: 10.1007/978-3-642-39799-8_19. URL https://doi.org/10.1007/978-3-642-39799-8_19.

Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks. *CoRR*, abs/1709.09130, 2017. URL http://arxiv.org/abs/1709.09130.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*, 2018.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2006. URL https://www.aclweb.org/anthology/P18-2006.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, May 2018. doi: 10.1109/SP.2018.00058.

Shalini Ghosh, Amaury Mercier, Dheeraj Pichapati, Susmit Jha, Vinod Yegneswaran, and Patrick Lincoln. Trusted neural networks for safety-constrained autonomous control. *CoRR*, abs/1805.07075, 2018. URL http://arxiv.org/abs/1805.07075.

Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, July 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018. URL http://arxiv.org/abs/1810.12715.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained neural fitted q-iteration. *CoRR*, abs/1809.07823, 2018. URL http://arxiv.org/abs/1809.07823.

T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *J. ACM*, 48(5):1038–1068, September 2001. ISSN 0004-5411. doi: 10.1145/502102.502106. URL http://doi.acm.org/10.1145/502102.502106.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation. *arXiv e-prints*, art. arXiv:1909.01492, Sep 2019.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2112–2121, 2018a.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pp. 452–461, 2018b. URL http://dl.acm.org/citation.cfm?id=3237452.

Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. Shielded decision-making in mdps. *arXiv preprint arXiv:1807.06096*, 2018.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified Robustness to Adversarial Word Substitutions. *arXiv e-prints*, art. arXiv:1909.00986, Sep 2019.

Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for mdps. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 130–146. Springer, 2016.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Ching-Yun Ko, Zhaoyang Lyu, Tsui-Wei Weng, Luca Daniel, Ngai Wong, and Dahua Lin. POPQORN: quantifying robustness of recurrent neural networks. *CoRR*, abs/1905.07387, 2019. URL http://arxiv.org/abs/1905.07387.

Xiao Li, Yao Ma, and Calin Belta. A policy search method for temporal logic specified reinforcement learning tasks. *CoRR*, abs/1709.09611, 2017a. URL http://arxiv.org/abs/1709.09611.

Xiao Li, Cristian Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 3834–3839, 2017b. doi: 10.1109/IROS.2017.8206234. URL https://doi.org/10.1109/IROS.2017.8206234.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 3578–3586, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pp. 46–57. IEEE Computer Society, 1977. doi: 10.1109/SFCS.1977.32. URL https://doi.org/10.1109/SFCS.1977.32.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018a.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *CoRR*, abs/1811.01057, 2018b. URL http://arxiv.org/abs/1811.01057.

Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M. Murray, and Sanjit A. Seshia. Reactive synthesis from signal temporal logic specifications. In Antoine Girard and Sriram Sankaranarayanan (eds.), *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, pp. 239–248. ACM, 2015. doi: 10.1145/2728606.2728628. URL https://doi.org/10.1145/2728606.2728628.

Casey Ross and Ike Swetlitz. Ibm's watson supercomputer recommended 'unsafe and incorrect'cancer treatments, internal documents show. *Stat News https://www. statnews. com/2018/07/25/ibm-watson-recommended-unsafe-incorrect-treatments*, 2018.

Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, pp. 1091–1096. IEEE, 2014. doi: 10.1109/CDC.2014.7039527. URL https://doi.org/10.1109/CDC.2014.7039527.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. volume abs/1602.07868, 2016. URL http://arxiv.org/abs/1602.07868.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, January 2019. ISSN 2475-1421. doi: 10.1145/3290354. URL http://doi.acm.org/10.1145/3290354.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*, 2018.

Chenglong Wang, Rudy Bunel, Krishnamurthy Dvijotham, Po-Sen Huang, Edward Grefenstette, and Pushmeet Kohli. Knowing when to stop: Evaluation and verification of conformity to output-size specifications. In *CVPR*, 2019.

Qinglong Wang, Kaixuan Zhang, Xue Liu, and C. Lee Giles. Verification of recurrent neural networks through rule extraction. *CoRR*, abs/1811.06029, 2018a. URL http://arxiv.org/abs/1811.06029.

Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mixtrain: Scalable training of formally robust neural networks. *CoRR*, abs/1811.02625, 2018b. URL http://arxiv.org/abs/1811.02625.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *32nd Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2018c.

Min Wen, Ivan Papusha, and Ufuk Topcu. Learning from demonstrations with high-level side information. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3055–3061, 2017. doi: 10.24963/ijcai.2017/426. URL https://doi.org/10.24963/ijcai.2017/426.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5276–5285, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018a. PMLR. URL http://proceedings.mlr.press/v80/weng18a.html.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018b.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. *arXiv preprint arXiv:1809.03008*, 2018.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *CoRR*, abs/1811.00866, 2018. URL http://arxiv.org/abs/1811.00866.

Appendices

A STL QUANTITATIVE SEMANTICS

In addition to the qualitative semantics discussed in the main text, STL formulae have quantitative semantics (Donzé et al., 2013; Donzé & Maler, 2010) defined inductively by the function ρ below. For a given trace σ, with σ_t indicating the value of the signal at time t, the quantitative semantics is given by:

- $\rho(\texttt{true}, \sigma, t) = +\infty$
- $\rho(q(s) \geq 0, \sigma, t) = q(\sigma_t)$
- $\rho(\neg\varphi, \sigma, t) = -\rho(\varphi, \sigma, t)$
- $\rho(\varphi_1 \wedge \varphi_2, \sigma, t) = \min\left(\rho(\varphi_1, \sigma, t), \rho(\varphi_2, \sigma, t)\right)$
- $\rho(\varphi_1 \, \mathcal{U}_I \, \varphi_2, \sigma, t) = \max\limits_{t' \in t+I} \min\left(\rho(\varphi_2, \sigma, t'), \min\limits_{t'' \in [t,t']} \rho(\varphi_1, \sigma, t'')\right)$

One can obtain the qualitative semantics from the sign of the quantitative semantics. Specifically, $(\sigma, t) \models \varphi \iff \rho(\varphi, \sigma, t) \geq 0$.

We can convert the formulae to their equivalent negation normal form by following the standard procedure until negations are only associated with atoms and Boolean constants. In particular, we interpret $\rho(\neg\texttt{true}, \sigma, t) = -\infty$ and use the disjunction operator defined as $\rho(\varphi_1 \vee \varphi_2, \sigma, t) = \max(\rho(\varphi_1, \sigma, t), \rho(\varphi_2, \sigma, t))$. The normal form for $\neg(\varphi_1 \mathcal{U}_I \varphi_2)$ is obtained by pushing the negation into the subformulae, and swapping $\min$ with $\max$. Finally, we turn $\neg(q(s) \geq 0)$ into $-q(s) > 0$ which we approximate by $-q(s) - \delta \geq 0$ for some small $\delta > 0$.

B PROOF OF LEMMA 1

Proof. We proceed by induction on φ. The base cases and the conjunction case are straightforward, and the atom case follows by assumption. The disjunction case requires us to show: $\omega_{S,f}(\varphi_1 \vee \varphi_2, t) \leq \min\limits_{x \in S}(\max(\rho(\varphi_1, f(x), t), \rho(\varphi_2, f(x), t)))$. Applying the max-min inequality, the right hand side is at least $\max(\min\limits_{x \in S}(\rho(\varphi_1, f(x), t)), \min\limits_{x \in S}(\rho(\varphi_2, f(x), t))$. Then using the inductive hypotheses, we know this is at least $\max(\omega_{S,f}(\varphi_1, t), \omega_{S,f}(\varphi_2, t))$, and the case follows. The case for the $\mathcal{U}$ operator has a similar proof based on the max-min inequality. □

C BOUNDS FOR DISCRETE INPUTS

Tasks with discrete inputs, such as language generation tasks, encode a prefix sentence as conditioning before decoding a follow-up sequence of words. Consider prefixes of the form $x = x_0, x_1, \ldots$ such that $x_i \in S_i$, where S_i is a finite set of tokens that can appear at position i in the input sequence. We can propagate perturbations in the prefix by first projecting the tokens S_i through the embedding layer E , and then considering the maximum and the minimum value along each embedding dimension to bound the output from E. Formally,

$$\underline{E}_j(x_i) = \min_{x_i \in S_i} E_j(x_i) \leq E_j(x_i) \leq \max_{x_i \in S_i} E_j(x_i) \leq \overline{E}_j(x_i). \tag{5}$$

Jia et al. (2019); Huang et al. (2019) also consider bound propagation for word substitutions.

	Beyond ReLU	Training for Spec. Satisfaction	Continuous Input Features	Components with Gating	Auto-regressive components	Temporal Specifications
Dvijotham et al. (2018) Zhang et al. (2018)	✓	✗	✓	✗	✗	✗
Raghunathan et al. (2018a); Wong et al. (2018) Wang et al. (2018b)	✗	✓	✓	✗	✗	✗
Gowal et al. (2018); Mirman et al. (2018)	✓	✓	✓	✗	✗	✗
Dutta et al. (2017); Tjeng et al. (2017)	✗	✗	✓	✗	✗	✗
Ghosh et al. (2018); Xiao et al. (2018) Li et al. (2017a)	✓	✓	✗	✓	✓	✓
Weng et al. (2018b)	✗	✗	✓	✗	✗	✗
Ko et al. (2019)	✓	✗	✓	✓	✗	✗
Ours	✓	✓	✓	✓	✓	✓

Table 6: Comparison of methods developed for enforcing/checking consistency with specifications.

D DETAILED COMPARISON WITH RELATED WORK ON VERIFICATION/VERIFIED TRAINING

E MMNIST

Figure 1 depicts a sample image from the MMNIST dataset. For this task, for the GRU decoder, we train a 2 layer-convolution network with 32 filters in each layer, and the GRU decoder has 2 cells, each with a latent space of 64 dimensions. Further, to help training with the specification stabilize, we normalize the weights in the linear-layers of the GRU with the l_1 norm (similar to the Salimans & Kingma (2016), where the weights are normalized with the l_2-norm. For the curriculum training, we first train on the original task for 5000 steps to get the full clean accuracy, and then train with the specification loss for another 300000 steps to regularize the model to satisfy the specification. During this phase, we gradually ramp up the perturbation radius ϵ to the desired value, and also simultaneously increase the coefficient λ. The normalizing prevents the bounds from getting large during the wamring-up phase of training.

For adversarial training, we use a 7-step PGD attack from Madry et al. (2017), and we found that a similar curriculum helps stabilize adversarial training. Adversarial training with the largest perturbation radius as in Madry et al. (2017) degraded the performance on the nominal task signficiantly, while curriculum based adversarial training degrades performance to a much lesser extent

For the test-train data split on this task, we use the same as in Wang et al. (2019).

F RL AGENT

F.1 INITIAL STATES OF THE RL AGENT

S_ϵ corresponds to the states (x, y) within a l_∞ distance of ϵ from the centre of each of the cells. We formally define this set below.

For a cell i (for Figure 3, $i \in \{1, 2 \ldots 25\}$) with center x_{c_i}, y_{c_i}, the ϵ-ball $S_{i,\epsilon}$ corresponds to the set of positions (x_a, y_a) for the agent such that $\|(x_a - x_{c_i}, y_a - y_{c_i})\|_\infty \leq \epsilon$. Formally,

$$S_{i,\epsilon} := \{(x_a, y_a) : \|(x_a - x_{c_i}, y_a - y_{c_i})\|_\infty \leq \epsilon\}.$$

We can then define $S = \bigcup_i S_{i,\epsilon}$ as the set of feasible initial states of the agent for which we wish to verify the property. Table 4 reports the fraction of cells i for which we are able to verify that the agent recharges on starting from $S_{i,\epsilon}$.

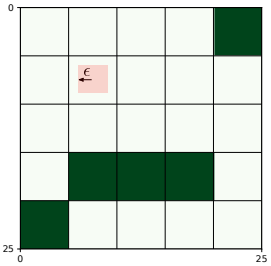

Figure 2: Domain for the robot. Recharge cells in green.

F.2 TASK AND TRAINING DETAILS

The agent's observation at each time-step t contains i) its own coordinates (x_t, y_t), ii) for each cell, the time remaining until that cell is dirty, and iii)

the (fixed) locations of the recharge cells. The agent learns a policy from these observations to a continuous control action $(a_{x,t}, a_{y,t}) \in \mathbb{R}^2$. The continuous part of the agent's state is updated as:

$$x_{t+1} = x_t + a_{x,t}, \qquad y_{t+1} = y_t + a_{y,t}. \tag{6}$$

The policy is represented in the parameters θ, and the result of applying the policy then the environment update is our function f_θ.

Here, we consider verifying an agents trained with Deep-Q learning in an environment with $T_{\text{recharge}} = 3$ and $T_{\text{dirt}} = 4$, i.e, every cell accumulates dust four time-steps after it was cleaned, and the robot needs to recharge itself every 4 time-steps. Additionally, the initial cell where the robot starts can have between $0 - 0.1\%$ uncertainty in the amount of dirt, the charge that can be acquired from the recharge station, and the initial battery (as another element of uncertainty in the initial position).

If the agent leaves the domain, it is clipped back into the problem domain. The agent's each have 4 discrete actions and are trained with a combination of vanilla Deep-Q learning, Deep-Q learning + reward-shaping, and Deep-Q learning + verifiable training. The actions corresponds to velocities in the 4 cardinal directions, i.e., $\{(0, 5), (5, 0), (-5, 0), (0, -5)\}$. For the agent trained to be verifiable, in addition to loss from Deep-Q learning, the agent is also trained to be verifiable with respect to the temporal specification presented in Section 3.2.2. The verification losses are optimized using the Adam optimizer Kingma & Ba (2015) with learning rate 10^{-3}. The weight of the verification loss anneals linearly between 0 and 1.5 during the first 70K steps. The model is trained to be verifiable starting at the center of the grids, and eventually covering the region around the centers, with $\epsilon = 0.015$ during the same 70k steps. We find that this regularizes the model to be verifiable in regions outside the region in which the model was trained to be verifiable.

G SHAKESPEARE

For the text generation task, we train a "seq2seq" model of Wu et al. (2016) with a GRU core. We use the standard test-train split for this task. The reconstruction and verification losses are optimized using the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 10^{-1}. The learning is decayed every 1K steps by a factor 10, till it reaches 0.001. The weight of the verification loss which is computed from the specification in Section 3.2.3 varies linearly between 20 and 1 during 100000 steps. Finally, the verification loss is clipped between below 10. The gradients during optimization are clipped at 0.1 to prevent exploding gradients from the differentiable approximations of `one-hot(softmax(.))`

G.1 PREFIX TEMPLATE

The Language Model prefixes are generated using the following syntax: `<pronoun>, <person>, <action-verb>, <connector>, <person>, <pronoun>, <action-verb>` , where:

`<pronoun>` = {*'my', 'your', 'his', 'her', 'our', 'their'*}

`<person>` = {*'sister', 'brother', 'father', 'mother', 'son', 'daughter', 'king', 'queen', 'knight', 'noble','lord','duke', 'duchess', 'cousin', 'palace', 'widow','nurse', 'marshal', 'archbishop', 'mayor', 'maid'*}

`<action-verb>` = {*['changed','despised', 'loved', 'married', 'accused', 'anointed', 'danced', 'rejoiced', 'killed', 'came', 'left', 'prayed', 'stood', 'read', 'consorted', 'denied', 'condemned', 'ruled', 'proved', 'parted' 'resolved', 'committed', 'raised', 'urged', 'painted', 'provoked', 'lived', 'charged', 'yielded', 'accursed', 'assured'],* }

`<connector>` = {*'but', 'while', 'yet', 'and', 'because']*}

The space of combinations holds 25779600 possibilities to condition the language model generation upon. An example prefix is :'*Our lord yielded and their king left*'.

H BOUND PROPAGATION THROUGH f

We describe how to perform bound propagation for a general recurrent neural network. The neural network takes as input x and produces a sequence of outputs y_τ for $\tau = 0, ..., K$ so the overall output is $(y_0, y_1, \ldots, y_K)$. We assume that we are given bounds on the input x

$$l_0 \leq x \leq u_0.$$

Our goal is to obtain bounds on y_τ given bounds on x for each τ. Each output is produced conditioned on the preceding outputs: y_τ depends on $y_0, \ldots, y_{\tau-1}$.

We proceed recursively, assuming that we have already computed bounds on $y_0, \ldots, y_{\tau-1}$. We stack the set of inputs to the computation as $(x, y_0, \ldots, y_{\tau-1}) \in [l_{\tau;0}, u_{\tau;0}]$. We study the computation graph mapping these inputs to the output y_τ. At each node in this computation graph, we perform a computation of the form

$$z_{\tau,i} = w_i^T h\left(z_{\tau;-i}\right) + \tilde{w}_i^T \tilde{h}\left(z_{\tau;-i}\right)^T z_{\tau;-i} + b_i$$

where $h, \tilde{h}$ are element-wise nonlinear operations (sigmoid, tanh, relu etc.) and $z_{\tau;-i}$ denotes the elements of computational graph that are ancestors of the node i. The second term represents multiplicative interactions (gating interactions) common in recurrent networks like LSTMs and GRUs. Suppose we have already computed lower and upper bounds $l_{\tau;-i}, u_{\tau;-i}$ on the preceding elements. Then, we have

$$\begin{aligned}
z_{\tau,i} &\geq \max\left(w_i, 0\right)^T h\left(l_{\tau;-i}\right) + \min\left(w_i, 0\right)^T h\left(u_{\tau;-i}\right) \\
&\quad + 1^T \min\begin{pmatrix} \tilde{w}_i \odot \tilde{h}\left(l_{\tau;i}\right) \odot \left(l_{\tau;i}\right), & \tilde{w}_i \odot \tilde{h}\left(u_{\tau;i}\right) \odot \left(l_{\tau;i}\right), \\ \tilde{w}_i \odot \tilde{h}\left(l_{\tau;i}\right) \odot \left(u_{\tau;i}\right), & \tilde{w}_i \odot \tilde{h}\left(u_{\tau;i}\right) \odot \left(u_{\tau;i}\right) \end{pmatrix} \\
z_{\tau,i} &\leq \max\left(w_i, 0\right)^T h\left(u_{\tau;-i}\right) + \min\left(w_i, 0\right)^T h\left(l_{\tau;-i}\right) \\
&\quad + 1^T \max\begin{pmatrix} \tilde{w}_i \odot \tilde{h}\left(l_{\tau;i}\right) \odot \left(l_{\tau;i}\right), & \tilde{w}_i \odot \tilde{h}\left(u_{\tau;i}\right) \odot \left(l_{\tau;i}\right), \\ \tilde{w}_i \odot \tilde{h}\left(l_{\tau;i}\right) \odot \left(u_{\tau;i}\right), & \tilde{w}_i \odot \tilde{h}\left(u_{\tau;i}\right) \odot \left(u_{\tau;i}\right) \end{pmatrix}
\end{aligned}$$

Setting $l_{\tau,i}$ to the lower bound above and $u_{\tau,i}$ to the upper bound, we have computed bounds on $z_{\tau,i}$. Thus, we can recursively compute bounds until we obtain bounds $l_\tau \leq y_\tau \leq u_\tau$, which can then be used to compute bounds on $y_{\tau+1}$. Proceeding recursively, we obtain lower and upper bounds on $(y_0, y_1, \ldots, y_K)$.

I SHAKESPEARE TEXT GENERATIVE MODEL SAMPLES

Method	Prefix	Generated Sequence
Verified Training	Very well; and	<eol> if we are like the king; and so the <eol> which of some other part
	Would you	<eol> could would be such other of the whole but is <eol> the a piece of
	If I must not,	<eol> as to be told general of the people, till he <eol> be the king to
	Soft! who comes	<eol> with the city but the whole prince and he <eol> are not them to the
Nominal Training	Very well; and	let <eos> 3 man <eos> will will not the new <eos> which the man of his
	Would you	<eol> shalt not the first to his face that you were not <eol> to the prince
	If I must not,	<eol> thing the matter but the man that he have <eol> not a piece of the
	Soft! who comes	<eol> these thing for the man that is the king <eol> <eol> is no matter of

Table 7: Samples of generated text for nominal and verifiably training GRUs on the Shakespeare corpus under greedy decoding. <eol> refers to the end of line.