STL Net

Ira A. Fulton Schools of Engineering

Arizona State University

Signal Temporal Logic (STL)

Signal Temporal Logic (STL)

- A temporal logic (like LTL or CTL)
  - Semantic structure assumes multiple worlds or states over time
- Allows for reasoning over time intervals (like MTL)
- Predicates represent *analog signals* meaning that they are equivalent to the value of a function exceeding a certain value (usually zero)

STL Syntax

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \ \mathbf{U}_{[a,b]} \ \psi$$

- $\perp = \neg\top$
- Eventually is $\mathsf{F}_{[a,b]} \ \varphi = \top \ \mathcal{U}_{[a,b]} \ \varphi$
- Always is $\mathsf{G}_{[a,b]}\varphi = \neg(\ \mathsf{F}_{[a,b]} \ \neg\varphi)$

STL adds an analog layer to MTL. Assume signals $x_1[t], \ x_2[t], \ \ldots, x_n[t]$, then atomic predicates are of the form:

$$\mu = f(x_1[t], \ldots, x_n[t]) > 0$$

$\square$, $\lozenge$ Are often used to denote "always" and "eventually"

From https://people.eecs.berkeley.edu/~sseshia/fmee/lectures/EECS294-98_Spring2014_STL_Lecture.pdf

STL Semantics

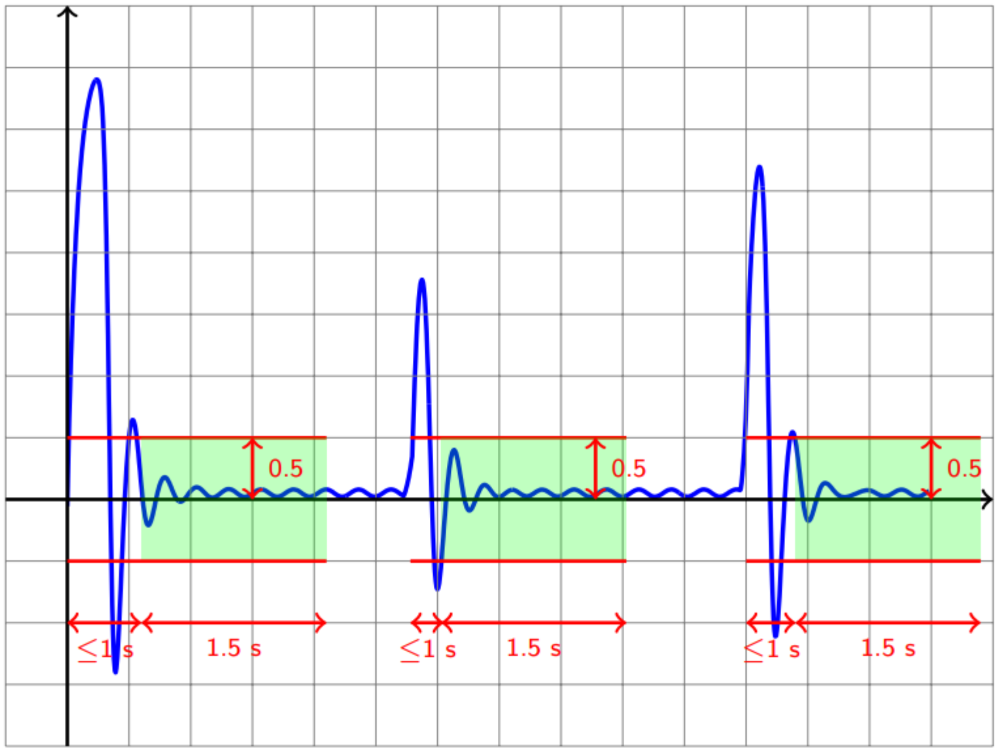
$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi \; \mathcal{U}_{[a,b]} \; \psi & \Leftrightarrow & \exists t' \in [t + a, t + b] \text{ such that } (x, t') \models \psi \wedge \\
 & & \qquad \forall t'' \in [t, t'],\ (x, t'') \models \varphi \}
\end{array}
$$

From https://people.eecs.berkeley.edu/~sseshia/fmee/lectures/EECS294-98_Spring2014_STL_Lecture.pdf

Example Specification

Always $|x| > 0.5 \Rightarrow$ *after 1 s,* $|x|$ *settles under 0.5 for 1.5 s*

$$\varphi := \mathsf{G}(x[t] > .5 \rightarrow \ \mathsf{F}_{[0,.6]} \ \ (\ \mathsf{G}_{[0,1.5]} \ x[t] < 0.5))$$



From https://people.eecs.berkeley.edu/~sseshia/fmee/lectures/EECS294-98_Spring2014_STL_Lecture.pdf

STL Satisfaction Function

The semantics can be defined as function $\chi^{\varphi}(x,t)$ such that:

$$x, t \models \varphi \Leftrightarrow \chi^{\varphi}(x,t) = \top$$

Considering Booleans $(\mathbb{B}, <, -)$ as an order with involution:

$$\begin{aligned}
\chi^{\mu}(x,t) &= f(x_1[t], \ldots, x_n[t]) > 0 \\
\chi^{\neg\varphi}(x,t) &= -\chi^{\varphi}(x,t) \\
\chi^{\varphi_1 \wedge \varphi_2}(x,t) &= \min(\chi^{\varphi_1}(x,t), \chi^{\varphi_2}(w,t)) \\
\chi^{\varphi_1 \ \mathcal{U}_{[a,b]} \ \varphi_2}(x,t) &= \max_{\tau \in t+[a,b]} (\min(\chi^{\varphi_2}(x,\tau), \ \min_{s \in [t,\tau]} \chi^{\varphi_1}(x,s))
\end{aligned}$$

From https://people.eecs.berkeley.edu/~sseshia/fmee/lectures/EECS294-98_Spring2014_STL_Lecture.pdf

Ira A. Fulton Schools of
Engineering
Arizona State University

Model Checking for Neural Networks

Setup

A sequence consists of *m* variables over *n* time steps.

$$\omega = \{\omega^1, \omega^2, \ldots, \omega^m\}$$

We are concerned with a prefix (first *i* time steps) and a suffix (the remaining time units) for each variable, so for a given variable *k* we represent it as follows.

$$\omega^k = x^k_{[0,i]} x^k_{[i+1,n]}$$

A parameterized model *f* accepts a prefix and predicts a suffix

$$(\hat{x}^1_{[i+1,n]}, \hat{x}^2_{[i+1,n]}, \ldots, \hat{x}^m_{[i+1,n]}) = f((x^1_{[0,i]}, x^2_{[0,i]}, \ldots, x^m_{[0,i]}); \theta)$$

Notation from Ma et al., NeurIPS 2020.

Setup

The sequence is required to satisfy a set of STL properties

$$\omega \models \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_\nu$$

Hence, we look to solve the following problem

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{\omega \leftarrow \mathbf{w}} \left[\mathcal{D}(\omega, \hat{\omega})\right]$$

$$s.t.\ \ \hat{\omega} \models \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_\nu$$

Where D is a distance function (e.g., loss, L-2 is used in this paper)

Notation from Ma et al., NeurIPS 2020.

What type of constraints are we interested in?

Property Type	Example	STL formula
Reasonable Range	The traffic volume on a road can never exceed the road capacity.	$\square_{[0,24]}(x_1 < \alpha_1) \wedge \cdots \wedge \square_{[0,24]}(x_n < \alpha_n)$
Consecutive Changes	The number of people in a shopping mall should not increase or decrease more than 1000 in 10 min if exits number is less than 5.	$y < 5 \to \square_{[0,10]}(\Delta x < 1000)$
Resource Constraint	The total energy distributed to all buildings should be less than e.	$\square_{[0,24]}\mathsf{sum}(x_1, \ldots, x_n) < e$
Variable and Temporal Correlation	For two consecutive intersections on a one-way direction road, if there are 10 cars passing intersection A, then there should be at least 10 cars passing intersection B within the next 5 minutes.	$(x_1 > 10 \to \lozenge_{[0,5]}(x_2 > 10)) \wedge \cdots \wedge (x_n > 10 \to \lozenge_{[0,5]}(x_{n+1} > 10))$
Existence	There should be at least 1 patrol car around school every day.	$\lozenge_{[0,24]} x_1 \geq 1 \wedge \cdots \wedge \lozenge_{[0,24]} x_n \geq 1$
Unusual Cases	If there is a concert on Friday, the number of people in the nearby shopping mall will increase at least 200 within 2 hours.	$x_{\mathsf{Event}} = \mathsf{True} \wedge x_{\mathsf{Day}} = \mathsf{Fri} \to \lozenge_{[0,2]}\Delta x > 200.$

Table from Ma et al., NeurIPS 2020.

Ira A. Fulton Schools of Engineering

Arizona State University

Student-Teacher Network Paradigm

Student-Teacher Network Paradigm

Intuition:

1. Build a **student network** (*p*) to predict the suffix based on the prefixes using a purely neural approach
2. Build a **teacher network** (*q*) generates a trace that satisfies the specification while having minimal distance from the result of *p.* Note that *q* is constructed by projecting *p* into a subspace constrained by the specification.
3. A special loss function is used for backpropagation to update the parameters in p. to propagate the loss.
4. Steps 2-3 are repeated iteratively until convergence

STL Loss Function

$$\mathcal{L}_{\mathrm{STL}} = \overbrace{\boxed{\beta}}^{\textit{hyperparameter}}\mathcal{L}(\hat{\omega}, \overbrace{\boxed{\omega}}^{\textit{ground truth}}) + (1-\beta)\mathcal{L}(\hat{\omega}, \overbrace{\boxed{\omega'}}^{\textit{teacher result}})$$

- The specialized loss function is used to compare the result from the neural network not only with the training data, but also with the result of the parent network
- Hyperparameter beta measures the trade-off between the two

Notation from Ma et al., NeurIPS 2020.

Two Models

- Note that upon converge, we have two models. The result of *p* will likely have better fit, but cannot be guaranteed to meet the specification, while the result of *q* will meet the specification
- In practice, the result from *q* performs well with respect to L2 loss

Architecture Overview

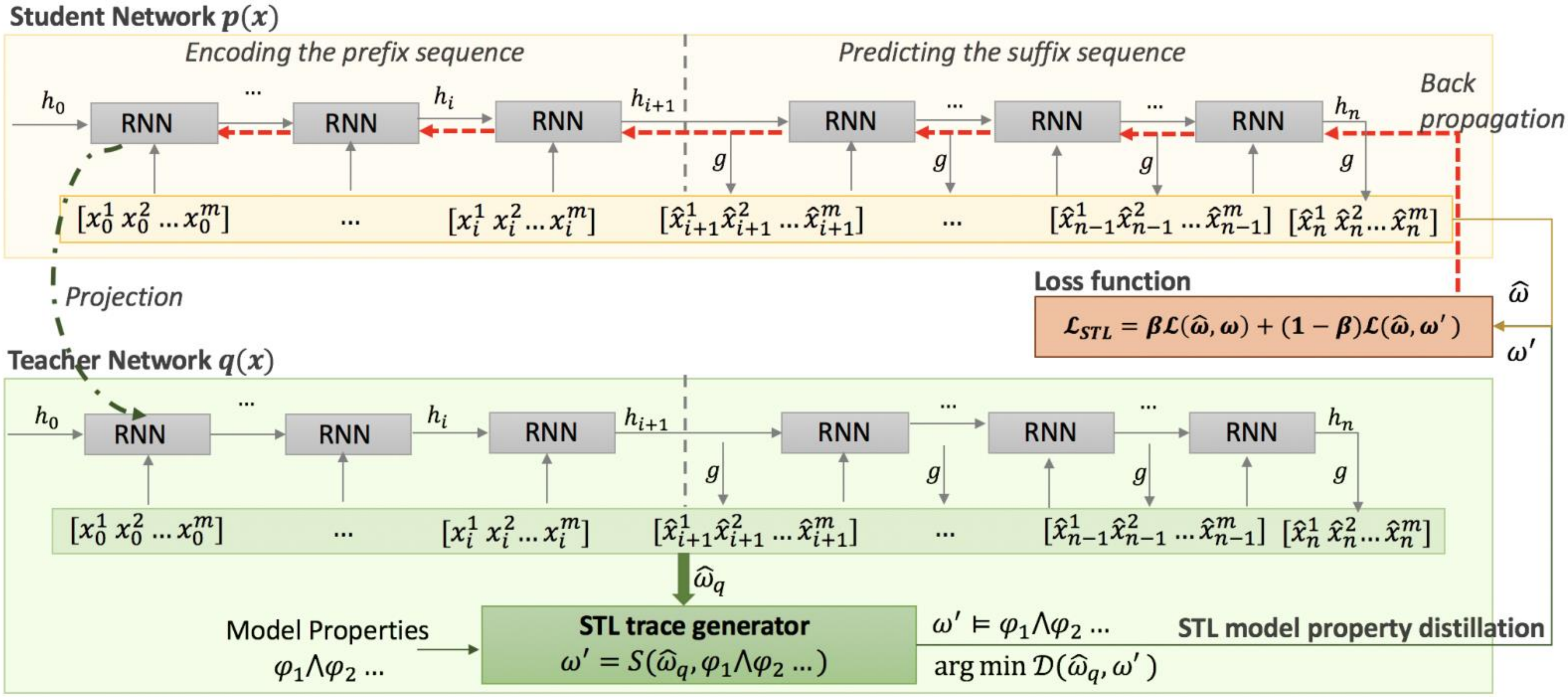


Figure from Ma et al., NeurIPS 2020.

Ira A. Fulton Schools of Engineering

Arizona State University

Trace Generation

Adjusting Neural Network Results to Meet a Specification

- We can think of the result of the neural model as a sequence of worlds over time (just note that the atoms in the worlds depend on analog values)
- If we can express a specification in a manner that allows us to simply compare worlds to the spec, we can update the trace in a straightforward manner

Key Idea: Converting Specification to DNF Form

- Turning the specification into DNF form (disjunction of conjunctions of literals) provides a few useful properties:
  1. Only one clause of the disjunction must be satisfied for the specification to be satisfied (so you can just iterate through clauses)
  2. The conjunctions of literals are very easy to compare with a world (as essentially you are just checking every atom and negation with each world in the sequence)
  3. Changing a sequence to meet a specification becomes trivial as you can simply modify the value associated with a specific predicate – and you can perform the modifications in such a way to be near the trace (via L1) as possible

Example Trace Generation

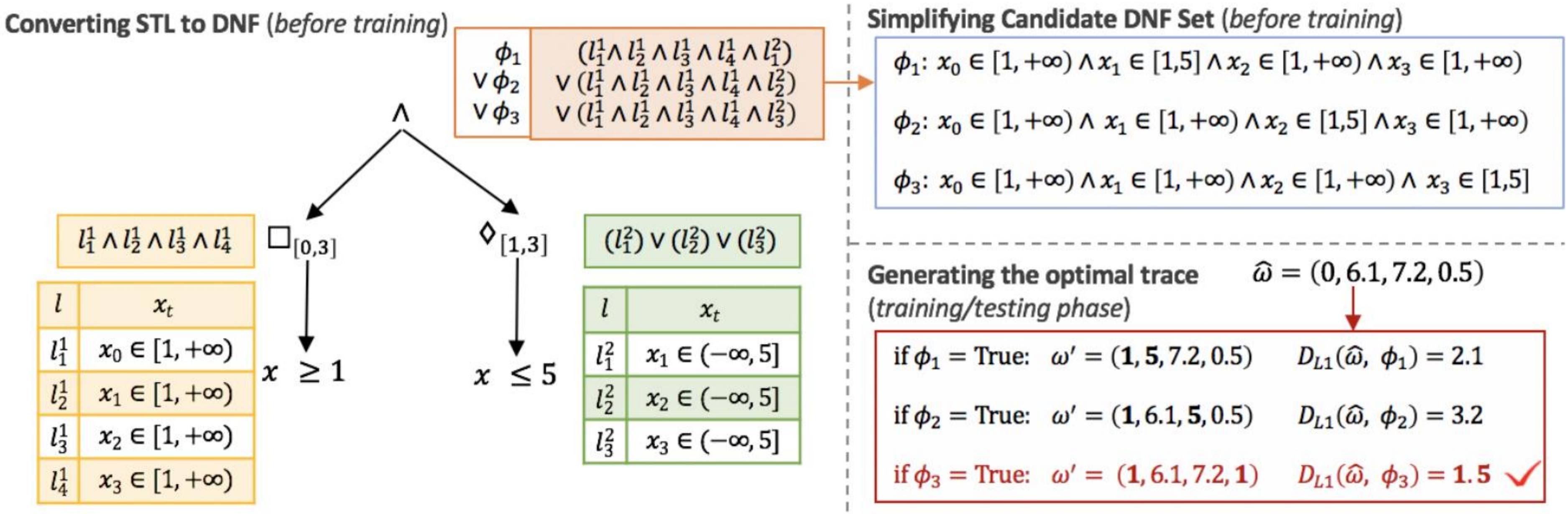


Figure 2: An example of STL trace generator ($\varphi = \square_{[0,3]} x \geq 1 \wedge \lozenge_{[1,3]} x \leq 5$)

Figure from Ma et al., NeurIPS 2020.

Key Result: Converting Specification to DNF Form

Proposition 4.1 (STL formula in DNF representation). *Every STL φ can be represented in the DNF formula $\xi(\varphi)$, where $\xi(\varphi)$ is a formula that includes several clauses ϕ_k that are connected with the disjunction operator, and the length of ϕ_k is denoted by $|\phi_k|$. Each clause ϕ_k can be further represented by several Boolean variables l_i that are connected with the conjunction operator. Finally, each Boolean variable l_i is the satisfaction range of a specific parameter.*

$$\begin{aligned} \xi(\varphi) &= \phi_1 \vee \phi_2 \vee ... \vee \phi_K \\ \phi_k &= l_1^{(k)} \wedge l_2^{(k)} \wedge .. \wedge l_{|\phi_k|}^{(k)} \quad \forall k \in \{1, 2..K\} \\ l_i^{(k)} &= \{x_t^j \mid f(x_t^j) \geq 0\} \; \textit{where} \; (t \in T), \forall i \in \{1, 2..|\phi_k|\} \end{aligned} \tag{2}$$

- The authors also describe how to reduce the number of clauses by eliminating clauses that are entailed by other clauses

Proposition from Ma et al., NeurIPS 2020.

Ira A. Fulton Schools of Engineering

Arizona State University

Experimental Evaluation

Metrics

- RMSE
- % of Specifications Satisfied
- ρ measure (distance from satisfaction, less than or equal to zero is satisfied)

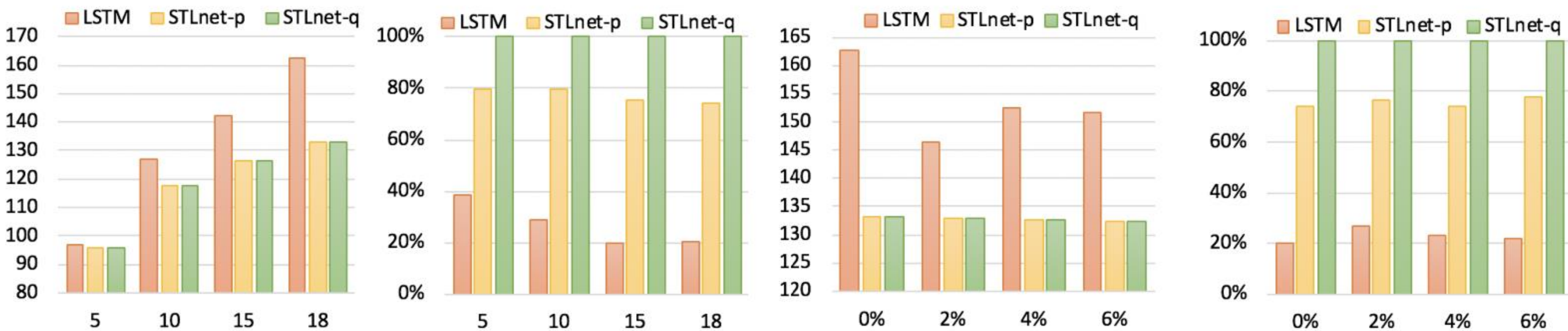
$$
\begin{aligned}
\rho(x \sim c, \omega, t) &= \pi_x(\omega)[t] - c \\
\rho(\neg\varphi, \omega, t) &= -\rho(\varphi, \omega, t) \\
\rho(\varphi_1 \wedge \varphi_2, \omega, t) &= \min\{\rho(\varphi_1, \omega, t), \rho(\varphi_2, \omega, t)\} \\
\rho(\Box_I \varphi, \omega, t) &= \min_{t' \in (t, t+I)} \rho(\varphi, \omega, t') \\
\rho(\Diamond_I \varphi, \omega, t) &= \max_{t' \in (t, t+I)} \rho(\varphi, \omega, t') \\
\rho(\varphi_1 \mathbf{U}_I \varphi_2, \omega, t) &= \sup_{t' \in (t+I) \cap \mathbb{T}} (\min\{\rho(\varphi_2, \omega, t'), \inf_{t'' \in [t, t']} (\rho(\varphi_1, \omega, t''))\})
\end{aligned}
$$

Results on Generated Data

	LSTM			LSTM STLnet-p			LSTM STLnet-q		
	RMSE	Sat Rate	Violateρ	RMSE	Sat Rate	Violateρ	RMSE	Sat Rate	Violateρ
φ_1	0.026	92.00%	-0.298	0.025	98.34%	-0.014	0.025	100.00%	0
φ_2	94.304	75.61%	-117.982	90.016	97.78%	-1.603	90.160	100.00%	0
φ_3	4.214	75.47%	-1.589	4.209	87.69%	-0.606	4.209	100.00%	0
φ_4	0.309	56.68%	-36.884	0.230	83.09%	-3.906	0.229	100.00%	0
φ_5	2.188	0.84%	-463.534	1.151	75.64%	-19.842	1.162	100.00%	0
φ_6	8.603	59.54%	-282.403	8.532	61.85%	-282.403	7.122	100.00%	0

	Transformer			Transformer STLnet-p			Transformer STLnet-q		
	RMSE	Sat Rate	Violateρ	RMSE	Sat Rate	Violateρ	RMSE	Sat Rate	Violateρ
φ_1	0.045	27.76%	-18.808	0.031	89.48%	-1.835	0.031	100.00%	0
φ_2	105.211	49.44%	-109.282	111.688	76.08%	-18.874	111.655	100.00%	0
φ_3	4.340	52.96%	-3.855	4.339	60.70%	-2.596	4.339	100.00%	0
φ_4	0.124	0.36%	-38.893	0.135	51.00%	-5.101	0.135	100.00%	0
φ_5	2.196	8.88%	-31.172	1.805	50.20%	-4.612	1.804	100.00%	0
φ_6	8.156	20.08%	-301.175	8.326	20.32%	-307.165	2.657	100.00%	0

Results on Air Quality Prediction Data



(a) RMSE-prediction lengths (b) Satisfaction Rate - prediction lengths (c) RMSE -missing data % (d) Satisfaction Rate - missing data %

ASU®

Ira A. Fulton Schools of
Engineering

Arizona State University

Comparison with Other Techniques

One Viewpoint: The Trace Generation as Refinement

- The neural network is producing a symbolic result
- A refinement process (in this case, the trace generation) adjusts the result
- We saw similar idea in DSR/DSP
- However, in this framework, the refinement results are passed back to the neural process

Another Viewpoint: Using a Neural Process to Approximate Results to Enable a Symbolic Process

- The student network is purely neural and does not provide any guarantees
- However, it is used as a starting point for more symbolically-oriented procedures

- We saw this idea with the “anchor model” in DSP
- This idea is also used in the “neural guided abduction” procedure of Tsamoura et al., (AAAI 2022)

Third Viewpoint: STL Net is just another way to add constraints to a neural architecture

- The hard constraints guaranteed by the result from the “teacher” network could be thought of an alternate process to frameworks like NeurASP or SLASH
- The softer constraints from the “student” network can be thought of as an alternative to frameworks like LNN, LTN, or NeuPSL

Ira A. Fulton Schools of Engineering

Arizona State University