CityPM: Predictive Monitoring with Logic-Calibrated Uncertainty for Smart Cities

Meiyi Ma, John Stankovic
University of Virginia
{meiyi,stankovic}@virginia.edu

Ezio Bartocci
TU Wien
ezio.bartocci@tuwien.ac.at

Lu Feng
University of Virginia
lu.feng@virginia.edu

ABSTRACT

We present CityPM, a novel predictive monitoring system for smart cities, that continuously generates sequential predictions of future city states using Bayesian deep learning and monitors if the generated predictions satisfy city safety and performance requirements. We formally define a flowpipe signal to characterize prediction outputs of Bayesian deep learning models, and develop a new logic, named *Signal Temporal Logic with Uncertainty* (STL-U), for reasoning about the correctness of flowpipe signals. CityPM can monitor city requirements specified in STL-U such as "with 90% confidence level, the predicated air quality index in the next 10 hours should always be below 100". We also develop novel STL-U logic-based criteria to measure uncertainty for Bayesian deep learning. CityPM uses these logic-calibrated uncertainty measurements to select and tune the uncertainty estimation schema in deep learning models. We evaluate CityPM on three large-scale smart city case studies, including two real-world city datasets and one simulated city experiment. The results show that CityPM significantly improves the simulated city's safety and performance, and the use of STL-U logic-based criteria leads to improved uncertainty calibration in various Bayesian deep learning models.

KEYWORDS

Smart City, Predictive Monitoring, Uncertainty, Deep Learning

1 INTRODUCTION

The prevalence of the Internet of Things has enabled a variety of smart city applications which aim to improve citizens' safety, wellness, and quality of life [27]. In a typical smart city control center (e.g., Intel Traffic Control Center [17], BreezoMeter Air Pollution control [5], and Microsoft CityNext [31]), the decision system takes input of various sensor readings and output actions for actuators. Increasingly, deep learning techniques have been applied to predict the city's future states, e.g., air quality forecasting [24] and fire risk prediction [35]. However, existing works either focus on predictions only or conduct some simple check of the predicted results, none of which considers the continuous monitoring of predictions with respect to complex city safety and performance requirements [26], which can be found in city codes, standards, regulations, laws, etc. In addition, existing results rarely account for the uncertainty inherent in smart cities (e.g., sensing and environmental noise, unexpected events, accidents).

In this paper, we address these limitations by developing a novel predictive monitoring system, named CityPM, for smart cities. As illustrated in Figure 1, CityPM interacts with a smart city control center to continuously predict future city states and monitor if predictions satisfy city requirements. If CityPM forecasts a potential city requirement violation in a future state, it would support the decision system in a control center to choose actions (e.g., issuing alarms, controlling traffic signals) to prevent such a requirement violation. The challenges of developing a reliable predictive monitoring system raise from two aspects: (1) prediction, and (2) monitoring. In the following, we identify gaps in the state-of-the-art, and describe how CityPM addresses these challenges.

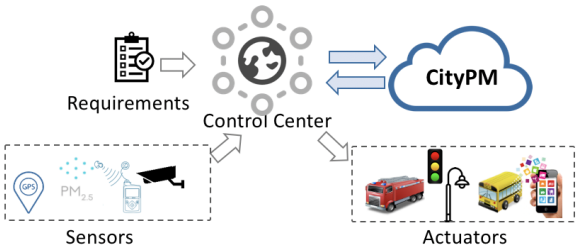


Figure 1: CityPM interacts with a smart city control center and enables the predictive monitoring of city states.

Prediction: Recurrent Neural Networks (RNNs) based sequential prediction has been popularly applied to smart cities [13, 25]. However, existing results mostly use deterministic RNNs which generate a single sequence of predictions and do not capture the uncertainty in smart cities. Recent advances such as Bayesian deep learning techniques [9] can adapt the prediction output stochastically as a sequence of posterior probability distributions over a finite discrete-time domain. But existing methods [10, 36] often use the loss functions of deep learning models (e.g., mean square error, negative log-likelihood, KL divergence) as the only metrics for the uncertainty estimation, which tend to over-estimate or under-estimate the uncertainty level. Furthermore, these metrics treat the uncertainty estimation of each individual value in a predicted sequence separately, and thus lack an integrated view about the uncertainty of sequential predictions. To address this challenge, we develop novel logic-based criteria to measure uncertainty. CityPM uses these logic-calibrated uncertainty measurements to select and tune the uncertainty estimation schema in deep learning models. To the best of our knowledge, this is the first work creating logic-based criteria to measure and train the uncertainty estimation schema for Bayesian RNN-based sequential prediction.

Monitoring: Signal Temporal Logic and its extensions have been applied for monitoring smart cities requirements [26, 28]. However, existing methods mostly focus on monitoring a single multi-variable signal and cannot be directly applied for monitoring the Bayesian sequential predictions. To address this challenge, we formalize the notion of a *flowpipe* signal to characterize prediction outputs of Bayesian deep learning, and develop a new logic, named *Signal Temporal Logic with Uncertainty* (STL-U), for reasoning about the correctness of flowpipe signals. STL-U can be used to specify city requirements with uncertainty, such as "with 90% confidence level,

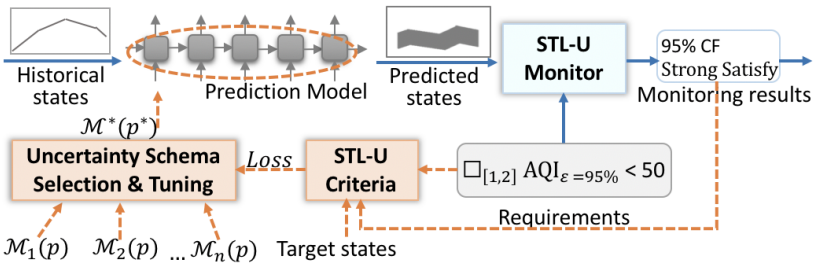
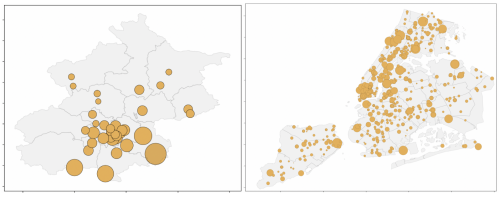


Figure 2: Overview of CityPM (The blue lines show the runtime, and the orange dash-lines illustrate the training time.)



(a) AQI in Beijing (b) Traffic in New York City

Figure 3: Maps of Uncertainty levels (The size of the yellow circle indicates the data uncertainty level at that location)

the predicated air quality index in the next 10 hours should always be below 100". We define *strong* and *weak* satisfaction relations for the Boolean semantics of STL-U to indicate whether *all* or *partial* values within a flowpipe confidence interval range satisfies a requirement, respectively. We also develop algorithms for computing the confidence level that guarantees a STL-U property is satisfied by the given flowpipe signals. Such results can provide smart city decision-makers with meaningful confidence guarantees about predictions of city future states satisfying city requirements.

In summary, the major **contributions** of this work include:

- conducted a data-driven study on real-world city datasets to analyze the uncertainty in smart cities and its influence in predictive monitoring;
- developed CityPM, a novel predictive monitoring system that predicts a city's future states accounting for the uncertainty, and monitors if the predictions violate city requirements;
- created novel STL-U logic-based criteria to measure and train uncertainty estimation schema for Bayesian RNN-based prediction models;
- implemented a prototype tool and applied it to three large-scale smart city case studies, including two real-world city datasets and one simulated city experiment. The results show that CityPM significantly improves the simulated city's safety and performance, and the use of STL-U logic-based criteria leads to improved uncertainty calibration in various Bayesian deep learning models.

For the rest of the paper, we describe an overview of CityPM system in Section 2, and a data-driven study about the uncertainty in smart cities in Section 3. We present STL-U monitor and logic-based criteria for uncertainty measurement in Section 4. We describe how to use STL-U criteria to select and tune the uncertainty estimation schema for Bayesian RNN-based sequential prediction in Section 5. We show the evaluation results in Section 6, discuss the related work in Section 7, and summarize the conclusion in Section 8.

2 SYSTEM OVERVIEW

The goal of CityPM system is to predict the future states in a smart city and monitor if the predictions satisfy the city requirements. Figure 2 illustrates an overview of the CityPM system. It first takes city's historical states (e.g., the Air Quality Index (AQI) in the past 5 hours) as inputs and returns city's future states (e.g., the predicted AQI in next 2 hours) via an RNN-based Bayesian sequential prediction model. The predicted future states are represented by a sequence of distributions. At each predicted time point, it shows a range of the potential values under a given confidence level. Then, the STL-U monitor takes the predicted states and the formalized city requirements as inputs, and returns the projected verification results over the future time interval. For example, a requirement on AQI "with 95% confidence, the AQI should never exceed 50 in the next two hours" is formalized as "$\square_{[1,2]}\text{AQI}_{\varepsilon=95\%} < 50$". The results are presented with a confidence guarantee, e.g., over the next two hours the requirement will be strongly satisfied with 95% confidence.

At *training* time (the flow marked by the orange dash-lines in Figure 2), CityPM conducts model selection and tuning using STL-U criteria to obtain a well-calibrated uncertainty estimation schema for the RNN-based Bayesian sequential prediction. Intuitively, the satisfaction degree of the predicted sequence (i.e., predicted future states) should be same as the satisfaction degree of the target sequence (i.e., the ground-truth values). STL-U criteria is designed to measure the loss based on the monitoring results and thus evaluates the quality of the uncertainty estimation schema. In this way, the uncertainty estimation schema with the smallest STL-U loss is selected. At *runtime* (the flow marked by the blue lines in Figure 2), CityPM outputs the current and future monitoring results to support the smart city control center. As a real-time operational scenario, CityPM runs as a continuous iterative process. For example, for predictive monitoring of AQI in a smart city, at time t, CityPM first predicts the AQI for the future 3 hours from time t and monitors if the predictions satisfy the city requirements; after a period Δt (e.g., 30 minutes), CityPM predicts the AQI for the future 3 hours from $t + \Delta t$ and checks if the new predictions satisfy the requirements. In this way, CityPM provides continuous predictive monitoring of city states for smart city decision-makers.

3 DATA-DRIVEN STUDY OF UNCERTAINTY

We analyse the uncertainty in smart cities based on real city datasets, and study its influence on predictive monitoring.

3.1 Datasets

In this paper, we use two large-scale real-world city datasets: (1) air quality data from four major cities in China, and (2) traffic volume data from New York City in the United States. The *air quality dataset* were collected by the Urban Computing Team from Microsoft Research from 2014/05/01 to 2015/04/30 [24]. This dataset covers 437 stations from 4 major Chinese cities (Beijing, Tianjin, Guangzhou

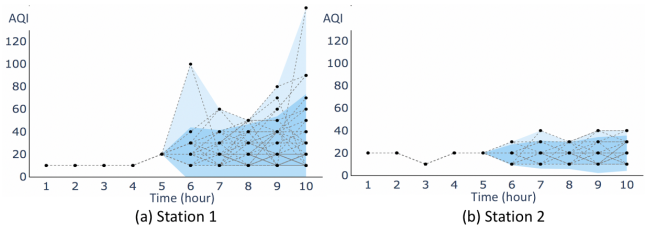


Figure 4: Uncertainty in the AQI data in Beijing

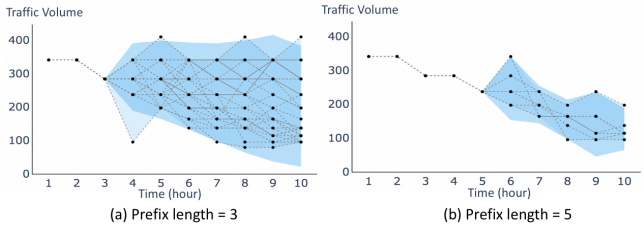


Figure 5: Uncertainty in the traffic volume data in NYC

and Shenzhen) and 39 adjacent cities within 300 kilometers to them. In total, there are 2,891,393 air quality records including PM2.5, PM10, NO2, CO, O3 and SO2. The *traffic volume dataset* were collected by the NYC Department of Transportation from 9/13/2014 to 4/5/2018 [32]. This dataset covers 1,490 street segments in New York City. In total, there are 514,776 records of traffic volume count.

3.2 Uncertainty Analytics

To start with, for both air quality and traffic volume datasets, we group every consecutive 10 hours data from one location as one instance. Then, at each location, we get all the instances with exactly the same l prefixes (i.e., the same levels of AQI or traffic volume for the first l hours), and compare their suffixes (i.e., the level of AQI or traffic volume for the following hours). Results of the air quality dataset and the traffic volume dataset are presented in Figure 4 and Figure 5, respectively. Here we utilize the range of 95% percentile to represent the uncertainty levels of the instances with the same prefix. The larger the range is (i.e. a higher standard deviation), the higher the uncertainty. The three sub-figures in Figure 4 compare the results from different locations in the air quality datasets, while the three sub-figures in Figure 5 compare the results with different prefix lengths. In addition, we calculate the average data uncertainty level (i.e., the standard deviation) of each location with prefix length equaling to 5 hours. Figure 3 (a) and (b) show the uncertainty levels of different locations in Beijing and New York City, respectively. The size of the yellow circle in the figure indicates the data uncertainty level at that location.

Observations: From these results, we made two observations.

(1) *Significant uncertainty exists in the city data.* For all locations from both datasets, we found instances with the same prefix leading to a different suffix. For example, Figure 4(a) presents a case from one of the air quality stations. With the prefix length equaling 5, we found 75 instances that have the same values in the first 5 hours. We draw the lines of all 75 instances in the figure, where we can see a very large range of data distribution. The 95% percentile data ranges are from 0 to 43 and from 0 to 62 at the 6th hour and 10th hour, respectively.

Table 1: Number of instances satisfying requirements with different threshold values of β

β	49	50	51	70	75	80
$\Box_{[0,10]}\text{AQI} < \beta$	2,731	2,807	2,895	3,614	4,011	4,443
$\Diamond_{[0,10]}\text{AQI} < \beta$	6,549	6,670	6,731	7,304	7,408	7,493
$\text{AQI} < 150\mathcal{U}_{[0,10]}\ \text{AQI} < \beta$	5,456	5,558	5,613	6,030	6,169	6,230
$\Box_{[0,10]}\text{Tra} < \beta$	1,122	1,241	1,359	3,090	3,332	3,532
$\Diamond_{[0,10]}\text{Tra} < \beta$	4,148	4,220	4,283	4,546	4,563	4,598

(2) *The levels of uncertainty vary by datasets and locations.* Comparing the three sub-figures in Figure 4, the overall uncertainty levels (i.e., the dark blue shadow) are different at three locations. Station 1 has the highest level of uncertainty while Station 3 has the lowest level. Meanwhile, the uncertainty levels from different datasets (e.g., Figure 4(a) vs. Figure 5(b)) also vary greatly. *Pre-knowledge (i.e., the prefix length) could also lead to different levels of uncertainty.* Comparing the three sub-figures in Figure 5 with prefix length equaling to 3, 4 and 5, respectively, we can see that the overall uncertainty level (i.e., the dark blue shadow) is highest when $l = 3$ and lowest when $l = 5$. The results are similar with the uncertainty levels at individual time (e.g., $t = 10$).

Most of the existing deterministic prediction models (e.g., ARIMA, RNN) predict future states based on past states' knowledge. Thus, given the same prefix from the same location (and potentially the same other features), the prediction models are likely to return the same results, which are inaccurate predictions based on the previous analysis on the datasets. Therefore, prediction with uncertainty is essential in cities. Moreover, since the uncertainty levels are different by dataset, selecting the uncertainty estimation schema based on the dataset is also important.

3.3 Influence on Predictive Monitoring

Safety requirements monitoring using formal specification languages such as Signal Temporal Logic (STL) and its extensions [3, 12, 26] are increasingly applied in smart cities. For example, to monitor the air quality, a requirement could be that the AQI should always be below a threshold β (e.g., $\beta = 50$ as Good, $\beta = 100$ as Medium), denoted as STL formula $\Box_{[0,10]}(\text{AQI} < \beta)$. Another example, to monitor the recovery of traffic after an accident, a requirement might be that the traffic volume should be less than β within 10 time units, denoted as STL formula $\Box_{[0,10]}(\text{Traffic} < \beta)$. In Table 1, we list five example requirements with β representing the parameter in the requirements. Then, we select different values for β and count the number of instances from one location satisfying the requirement. In total, there are 4,742 instances at one location for the first three AQI requirements, and 7,960 instances at one location for the last two traffic requirements. For example, when $\beta = 50$, there are 2,807 instances satisfying $\Box_{[0,10]}(\text{AQI} < 50)$.

Observation: From the results, we find that changing β slightly causes a big change of the number of satisfied requirements. For example, for the first requirement, when $\beta = 51$ (only adding 1 onto $\beta = 50$), the number goes up from 2,807 to 2,895 (i.e., 88); and changes 397 (from 3,614 to 4,011) comparing $\beta = 70$ and $\beta = 75$.

Even though the differences also vary by the type of requirements, the amount is still too large to ignore. From the perspective of the data, it shows that a small difference of the data could completely change the monitoring results. However, it is impossible to predict the data with 100% accuracy due to the existence of uncertainty, which makes the monitor results less effective to support decision making. It also indicates that the existing monitors are not robust enough for data with high uncertainty. It is therefore important to develop a monitor that can check the prediction results accounting for the uncertainty.

4 UNCERTAINTY MEASUREMENT: STL-U MONITOR AND CRITERIA

We first present a new logic *Signal Temporal Logic with Uncertainty* (STL-U) for specifying smart city requirements with uncertainty and monitoring the predicted city future states with confidence guarantees in Section 4.1. Then, we define a set of STL-U logic-based criteria to estimate the uncertainty for Bayesian sequential predictions in Section 4.2.

4.1 STL-U Monitor

We formally define new types of signals called *flowpipes*[1] that characterize the prediction results (i.e., the predicted future states of smart cities) of Bayesian deep learning with uncertainty.

Definition 1 (Flowpipe). *A flowpipe signal Ω is defined over a finite discrete time domain $\mathbb{T}$ such that $\Omega[t] = \Phi_t$ at any time $t \in \mathbb{T}$, where Φ_t is a Gaussian distribution $\mathcal{N}(\theta_t, \sigma_t^2)$.*

Given a confidence level $\varepsilon \in [0,1] \subseteq \mathbb{R}$, the flowpipe at time t is bounded by a confidence interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ with the lower bound $\Phi_t^-(\varepsilon) = \theta_t - \delta \cdot \frac{\sigma_t}{\sqrt{N}}$ and the upper bound $\Phi_t^+(\varepsilon) = \theta_t + \delta \cdot \frac{\sigma_t}{\sqrt{N}}$, where N is the number of samples that the Gaussian distribution is estimated from, and δ is a parameter obtained from the quantile function of Gaussian distribution Φ_t based on the confidence level ε. In the special case where the Gaussian distribution's variance $\sigma_t = 0$, the lower and upper bounds of the confidence interval coincide (i.e., $\Phi_t^-(\varepsilon) = \Phi_t^+(\varepsilon) = \theta_t$), thus the flowpipe becomes a single trace signal.

4.1.1 Syntax. We define the syntax of STL-U which is used to formalize city requirements on the city's future states. We denote by $\omega : \mathbb{T} \to \{\Omega\}^n$ a multi-dimensional flowpipe signal, where $\mathbb{T} = [0, d) \subseteq \mathbb{R}$ represents for a finite discrete time domain and $n = |X|$ for a finite set of (independent) *real* variables X. Each real variable $x \in X$ has a corresponding flowpipe Ω_x, whose value follows a Gaussian distribution $\Omega_x[t]$ at time t.

Definition 2 (STL-U Syntax). *The syntax of a STL-U formula φ over ω is defined by the grammar*

$$\varphi := \mu_x(\varepsilon) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2$$

where $I \subseteq \mathbb{R}^+$ is a time interval, and $\mu_x(\varepsilon)$ is an atomic predicate over a real variable x with confidence level ε whose value is determined by the sign of a function of an underlying flowpipe signal Ω_x, that is, $\mu_x(\varepsilon) \equiv f(x) > 0$ for $x \in [\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ with $\Phi_t = \Omega_x[t]$. (We assume that $f(x) = \lambda - x$ where $\lambda \in \mathbb{R}$ is a constant).

[1] To be noted, here we use the concept of flowpipes but define it in a new way.

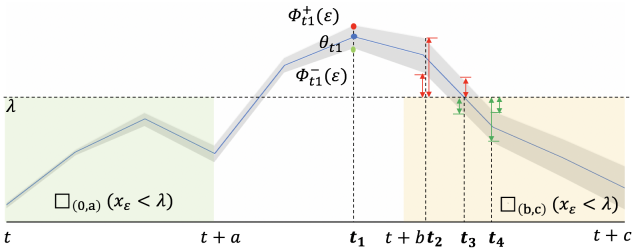


Figure 6: A flowpipe signal with confidence level ε.

The above syntactic definition of STL-U is minimal and includes only basic operators. We can derive other operators, for example, *eventually* denoted by $\Diamond_I \varphi \equiv \text{true}\ \mathcal{U}_I \varphi$, and *always* denoted by $\square_I \varphi \equiv \neg\Diamond_I \neg\varphi$. Using STL-U syntax, we can specify a requirement on AQI "with 95% confidence level, the AQI should never exceed 100 in the next 10 hours" as "$\square_{[0,10]}(\text{AQI}_{95\%} < 100)$".

4.1.2 Semantics. To verify the predicted flowpipe with its requirement, we define the the Boolean semantics of a STL-U formula φ over a multidimensional flowpipe signal ω at time t by two indices: *strong satisfaction*, denoted by $(\omega, t) \models_s \varphi$, and *weak satisfaction*, denoted by $(\omega, t) \models_w \varphi$.

Definition 3 (STL-U Strong Satisfaction).

$$\begin{aligned}
(\omega,t) &\models_s \mu_x(\varepsilon) && \Leftrightarrow f(\Phi_t^+(\varepsilon)) > 0\\
(\omega,t) &\models_s \neg\varphi && \Leftrightarrow (\omega,t) \not\models_w \varphi\\
(\omega,t) &\models_s \varphi_1 \wedge \varphi_2 && \Leftrightarrow (\omega,t) \models_s \varphi_1 \text{ and } (\omega,t) \models_s \varphi_2\\
(\omega,t) &\models_s \varphi_1 \mathcal{U}_I \varphi_2 && \Leftrightarrow \exists t' \in (t+I) \cap \mathbb{T}, (\omega,t') \models_s \varphi_2\\
& && \quad \text{and } \forall t'' \in (t,t'), (\omega,t'') \models_s \varphi_1
\end{aligned}$$

Definition 4 (STL-U Weak Satisfaction).

$$\begin{aligned}
(\omega,t) &\models_w \mu_x(\varepsilon) && \Leftrightarrow f(\Phi_t^-(\varepsilon)) > 0\\
(\omega,t) &\models_w \neg\varphi && \Leftrightarrow (\omega,t) \not\models_s \varphi\\
(\omega,t) &\models_w \varphi_1 \wedge \varphi_2 && \Leftrightarrow (\omega,t) \models_w \varphi_1 \text{ and } (\omega,t) \models_w \varphi_2\\
(\omega,t) &\models_w \varphi_1 \mathcal{U}_I \varphi_2 && \Leftrightarrow \exists t' \in (t+I) \cap \mathbb{T}, (\omega,t') \models_w \varphi_2\\
& && \quad \text{and } \forall t'' \in (t,t'), (\omega,t'') \models_w \varphi_1
\end{aligned}$$

Intuitively, the strong satisfaction means that all of the flowpipe signal values bounded within the confidence interval with a certain confidence ε should satisfy the requirement φ; while the weak satisfaction means that the flowpipe signal at least partially satisfies φ. In the smart city, the strong semantics can be applied to monitor safety critical requirements, e.g., the fire risk, accidents, and the weak semantics can be applied to monitor less-strict performance requirements, e.g., the noise levels, illumination of street lights.

Example 1. *The example flowpipe Ω shown in Figure 6 satisfies a STL-U formula $\square_{(0,a)}(x_\varepsilon < \lambda)$ at time t under both strong and weak Boolean semantics, because during the time interval $(t, t+a)$, the lower and upper bounds of the flowpipe's confidence interval are all below the threshold λ (see the green zone in Figure 6).*

Example 2. *Consider if the example flowpipe Ω shown in Figure 6 satisfies a STL-U formula $\square_{(b,c)}(x_\varepsilon < \lambda)$ at time t. At time t_2, both the lower and upper bounds of the flowpipe's confidence interval are above the threshold λ (i.e., $\lambda < \Phi_{t_2}^-(\varepsilon) < \Phi_{t_2}^+(\varepsilon)$), thus it neither strong nor weak satisfies $x_\varepsilon < \lambda$. At time t_3, the flowpipe signal weak*

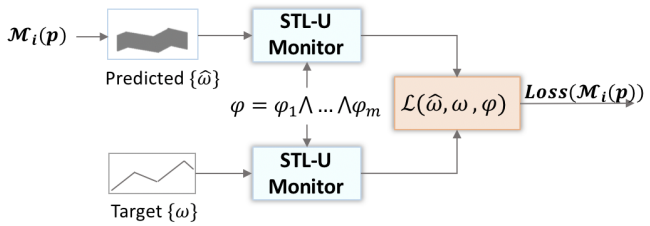


Figure 7: Illustration of STL-U Criteria

satisfies but not strong satisfies $x_\varepsilon < \lambda$, because $\Phi^-_{t_3}(\varepsilon) < \lambda < \Phi^+_{t_3}(\varepsilon)$. At time t_4, the flowpipe signal both strong and weak satisfies $x_\varepsilon < \lambda$, because $\Phi^-_{t_4}(\varepsilon) < \Phi^+_{t_4}(\varepsilon) < \lambda$. Therefore, the flowpipe neither strong nor weak satisfies the STL-U formula $\Box_{(b,c)}(x_\varepsilon < \lambda)$ at time t.

4.1.3 Confidence Guarantees. In the definition of Boolean semantics, we described how to monitor a flowpipe of city's future sates ω against a requirement φ for a given confidence level ε. However, it may not always be possible for city decision makers to have the confidence level ε specified *a priori*, instead they may want to query what is the appropriate confidence level ε for a predicted flowpipe ω to satisfy a requirement φ. In the following, we present a method for computing a confidence level that guarantees the predicted flowpipe signal ω strongly (*resp.* weakly) satisfies a STL-U formula φ, denoted by $\epsilon_s(\varphi, \omega, t)$ (*resp.* $\epsilon_w(\varphi, \omega, t)$). We give the formal definitions of calculating the confidence levels for strong and weak satisfaction in Definition 5 and Definition 6, respectively.

Definition 5. *Confidence calculation for strong satisfaction:*

$$
\begin{aligned}
\epsilon_s(\mu_x, \omega, t) &= \begin{cases} [0, \int_0^{2\theta} \Phi_t(x)dx], \text{if } \theta > 0 \\ \emptyset, \text{if } \theta \le 0 \end{cases} \\
\epsilon_s(\neg\varphi, \omega, t) &= \epsilon^c_w(\varphi, \omega, t) \\
\epsilon_s(\varphi_1 \wedge \varphi_2, \omega, t) &= \epsilon_s(\varphi_1, \omega, t) \cap \epsilon_s(\varphi_2, \omega, t) \\
\epsilon_s(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) &= \bigcup_{t' \in (t, t+I)} \{\epsilon_s(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_s(\varphi_1, \omega, t''))\}
\end{aligned}
$$

Definition 6. *Confidence calculation for weak satisfaction:*

$$
\begin{aligned}
\epsilon_w(\mu_x, \omega, t) &= \begin{cases} [\int_{2\theta}^0 \Phi_t(x)dx, 1], \text{if } \theta < 0 \\ [0, 1], \text{if } \theta \ge 0 \end{cases} \\
\epsilon_w(\neg\varphi, \omega, t) &= \epsilon^c_s(\varphi, \omega, t) \\
\epsilon_w(\varphi_1 \wedge \varphi_2, \omega, t) &= \epsilon_w(\varphi_1, \omega, t) \cap \epsilon_w(\varphi_2, \omega, t) \\
\epsilon_w(\varphi_1 \mathcal{U}_I \varphi_2, \omega, t) &= \bigcup_{t' \in (t, t+I)} \{\epsilon_w(\varphi_2, \omega, t') \cap (\bigcap_{t'' \in (t,t')} \epsilon_w(\varphi_1, \omega, t''))\}
\end{aligned}
$$

where ϵ^c_s and ϵ^c_w denote by the complement set of ϵ_s and ϵ_w within $[0, 1]$, respectively. At time t the value in a flowpipe follows a Gaussian distribution $\Phi_t \sim \mathcal{N}(\theta_t, \sigma_t^2)$, and $\Phi_t(x) = \frac{1}{\sqrt{2\pi}\sigma_t} e^{-\frac{(x-\theta_t)^2}{2\sigma_t^2}}$.

We include pseudo code of algorithms for computing confidence guarantees in the Appendix. Similarly to the process of monitoring algorithm, we first parse the requirement and assign the time interval for each condition. Then from the bottom of the parsing tree, we calculate the confidence level for the condition to be strong/weak satisfied at each time.

4.2 STL-U Criteria

The predicted flowpipes capture the level of uncertainty in the deep learning models. If the target sequence (i.e., the ground-truth of city states) satisfies (or violates) a certain set of requirements ($\varphi = \varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_m$), the predicted flowpipe should also satisfy (or violate) this requirement. Therefore, as illustrated in Figure 7, by calculating the discrepancy (i.e., Loss) between the predictive monitoring results of a predicted flowpipe and its target sequence, we can measure the quality of uncertainty estimated by schema $\mathcal{M}$. In this section, we define two STL-U criteria (loss functions) based on the STL-U Boolean semantics and confidence guarantees.

4.2.1 Criterion based on Boolean Semantics. We define the first criterion $\mathcal{L}_1$ based on the STL-U Boolean semantics. It mainly measures the quality of the uncertainty estimation by checking whether the predicted flowpipe has the same Boolean satisfaction with the target sequence. $\mathcal{L}_1$ is defined as a linear combination of three parts: $f_{\rho s}$ evaluates the accuracy on predicting the strong Boolean semantics of the requirement ϕ, $f_{\rho w}$ evaluates the accuracy on predicting the weak Boolean semantics of the requirement ϕ, and f_I evaluates the accuracy if the predicted flowpipe contains the target sequence .

$$
\begin{aligned}
f_{\rho_s} &= \mathbb{1}((\omega^{(i)} \models \varphi \wedge \hat{\omega}^{(i)} \models_s \varphi) \vee (\omega^{(i)} \not\models \varphi \wedge \hat{\omega}^{(i)} \not\models_s \varphi)) \\
f_{\rho_w} &= \mathbb{1}((\omega^{(i)} \models \varphi \wedge \hat{\omega}^{(i)} \models_w \varphi) \vee (\omega^{(i)} \not\models \varphi \wedge \hat{\omega}^{(i)} \not\models_w \varphi)) \\
f_I &= \mathbb{1}(\omega^{(i)} \in \hat{\omega}^{(i)})
\end{aligned}
$$

where, $\mathbb{1}(\phi)$ is an indicator function, $\mathbb{1}(\phi) = 1$ if ϕ = True, and $\mathbb{1}(\phi) = 0$ if ϕ = False. $\hat{\omega}^{(i)}$ and $\omega^{(i)}$ denoted by the ith predicted flowpipes and its corresponding target sequence, respectively, where the total number of validation samples denoted by N, thus, $\hat{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}$ denoted by the full set of predicted flowpipes and their target sequences, respectively. The goal is to maximize f_{ρ_s}, f_{ρ_w} and f_{ρ_I}, thus we have the loss function,

$$
\begin{aligned}
\mathcal{L}_1(\boldsymbol{\omega}, \hat{\boldsymbol{\omega}}, \varphi) = \; & \tfrac{1}{N} \textstyle\sum_{i=1}^{N} (1 - (\beta_1 f_{\rho_s}(\hat{\omega}^{(i)}, \omega^{(i)}, \varphi) + \beta_2 f_{\rho_w}(\hat{\omega}^{(i)}, \omega^{(i)}, \varphi) \\
& + (1 - \beta_1 - \beta_2) f_I(\hat{\omega}^{(i)}, \omega^{(i)})))
\end{aligned}
$$

β_1 and β_2 indicates the weights of strong and weak semantics, which evaluates two different aspects of the model, which should be set up based on the system in practice. If the system values more on the satisfaction of the whole uncertainty interval, the strong satisfaction formula f_{ρ_s} should be given a higher weight. Otherwise, the weak satisfaction formula f_{ρ_w} should be given a higher weight if the system values more on the existence of a sample in the interval.

4.2.2 Criterion based on Confidence Guarantees. We define the second criterion $\mathcal{L}_2$ based on the STL-U confidence guarantees. It mainly measures the quality of the uncertainty estimation by comparing the confidence level of satisfaction between the predicted flowpipe with the target sequence. $\mathcal{L}_2$ is also defined as a linear combination of two parts: the confidence level C_ϵ that guarantees the satisfaction (or violation) of the predicted flowpipe as the target sequence, and the confidence level C_I that guarantees the predicted flowpipe contains the target sequence. For C_ϵ, when the target sequence satisfies the requirement, the predicted flowpipe should be with higher confidence on strong satisfaction (i.e., a larger ϵ^+_s); or if the strong satisfaction is not possible (i.e., $\epsilon^+_s = 0$), it should be with a lower confidence on weak satisfaction (i.e., a smaller ϵ^-_w).

Then the confidence interval based loss is formulated as,

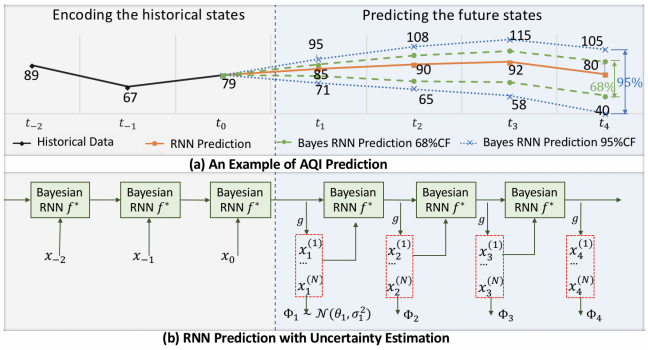


Figure 8: Prediction of AQI with Uncertainty Estimation

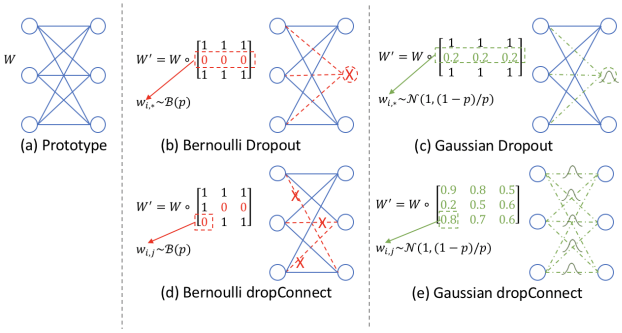


Figure 9: A visualization of Uncertainty Estimation Schemas on a Simple Neural Network

$$C_\epsilon = \begin{cases} 1 - \epsilon_s^+(\omega^{(i)}, \varphi, \omega_0^{(i)}) + \epsilon_w^-(\omega^{(i)}, \varphi, t)) & \omega_0 \models \varphi \\ \epsilon_w^+(\omega^{(i)}, \varphi, \omega_0^{(i)}) + 1 - \epsilon_s^-(\omega^{(i)}, \varphi, t)) & \omega_0 \not\models \varphi \end{cases}$$

We define $C_I(\omega, \hat{\omega})$ as the smallest possible confidence values c_I. For every time τ, $\hat{\omega}_\tau$ covers the ground-truth value ω_τ.

$$C_I = \min_{\tau \in T}\{c_I(\hat{\omega}_\tau^{(i)}, \omega_\tau^{(i)})\}$$

To minimize C_I and C_ϵ, we define the loss function as,

$$\mathcal{L}_2(\boldsymbol{\omega}, \boldsymbol{\omega}_0, \varphi, \mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}((1-\beta)C_\epsilon + \beta C_I)$$

Comparison: The two criteria are built on the strong/weak Boolean semantics and confidence level guarantees, respectively. They favor different aspects of the system. Users can select one or a combination of them based on the demands of the applications. In particular, $\mathcal{L}_1$ focuses on the satisfaction of the system. It fits the applications that only care about whether the prediction satisfy the requirements or not (rather than how much they satisfy). For example, the fire risk prediction and control service should apply $\mathcal{L}_1$ with the strong Boolean semantics since it has very strict requirements. Different from $\mathcal{L}_1$, $\mathcal{L}_2$ is more general since it does not require a pre-defined confidence level. In practice, $\mathcal{L}_2$ fits the applications that do not have a specific confidence level yet try to minimize the uncertainty, such as a recently deployed energy control service.

5 PREDICTION WITH LOGIC-CALIBRATED UNCERTAINTY

We describe the design of prediction models in CityPM, including how to use STL-U criteria to select and tune the uncertainty estimation schema for Bayesian RNN-based sequential prediction. Our method is agnostic to the choice of neural network structures.

Step 1: Build an RNN-based sequential prediction model. The first step is to build a deterministic prediction model using training data. We divide the prediction into two stages: (i) encoding the historical states from time t_{-2} to t_0, and (ii) predicting the future states from time t_1 to t_4. The model first encodes a part of the historical states. From t_1, at each time t, the cell inputs the current state x_t and outputs the next states x_{t+1}. x_t is decoded from the embedding states h_t, i.e. $x_t = g(h_t)$, and $h_t = f(x_t, h_{t-1}; W)$ where h_t is the hidden embedding and W are the learnable parameters of f. To predict the future AQI using a RNN predictor, when given a set of historical AQI states to the predictor, it returns estimated future states $\{x_1, x_2, x_3, x_4\}$. At this step, with the same set of input, the output will always be the same deterministic values.

Step 2: Cast the prediction model to a Bayesian model with uncertainty estimation. We cast the RNN predictor as a probabilistic model. We treat the learnable parameters W of RNN as a random variable W' by element-wise multiplying the learnable parameters W with a $n \times n$ mask w, where n is the number of neurons in one layer. Therefore, each element w_{ij} of mask w sampled from some probability distribution $\mathcal{M}(p)$ is multiplied by the connection between two neurons. Mathematically, this can be written as $W' = W \circ w$, where $w \sim \mathcal{M}(p)$. If $w_{ij} = 0$, the connection between neurons i and j will be dropped; if $w_{ij} = 1$, the connection remains the same; if $w_{ij} = \beta, \beta \in (0, 1)$, a weight β will be added to this connection. If $w_{i*} = 0$, it means all the connection to neuron i will be dropped, i.e. neuron i is dropped out. The Bayesian RNN formula is updated as, $x_{t+1}^* = g(h_t^*)$, and $h_t^* = f^*(x_t, h_{t-1}^*; w)$, where the h_t^* is a non-deterministic hidden embedding. To estimate the posterior probability distribution, we apply Monte Carlo estimate by repeating the prediction for N times. At each iteration, as shown in Figure 8(b), with the exact same input, we obtain a different set of outputs (i.e., a time series trace) $\{x_1^{(j)}, x_2^{(j)}, x_3^{(j)}, x_4^{(j)}\}$. In total, we have a set of traces ω containing N different traces. At a single time unit $t (t > 0)$, we obtain N observations of $x_t^{(j)}$ with $i = \{1, ..., N\}$. According to Bayesian rule, the prediction distribution is $\mathbb{P}(x_{t+1}|x_t, h_t) = \int_w \mathbb{P}(x_{t+1}|x_t, h_t, w)\mathcal{M}(p)dw$, where $\mathbb{P}(x_{t+1}|x_t, h_t)$ is commonly assumed to be subjected to Gaussian noise $\mathcal{N}(\theta_t, \sigma_t^2)$ [2]. Under this assumption, we estimate the expectation $\theta_t = \tilde{\mathbb{E}}[x_{t+1}|x_t, h_t] \approx \frac{1}{N}\sum_{i=1}^n x_t^{(i)}$ from the samples. Similarly, we have an estimator of the variance $\sigma_t = \widetilde{\mathrm{Var}}[x_{t+1}|x_t, h_{t-1}] \approx \frac{1}{N}\sum_{i=1}^n x_{t+1}^{(i)T} x_{t+1}^{(i)} - \tilde{\mathbb{E}}[x_{t+1}|x_t, h_{t-1}]^T \tilde{\mathbb{E}}[x_{t+1}|x_t, h_{t-1}]$. Therefore, at time t, we obtain the estimated results from the Bayesian RNN as a distribution, i.e. $\Phi_t \sim \mathcal{N}(\theta_t, \sigma_t^2)$.

Step 3: Uncertainty estimation parameter tuning and schema selection with STL-U criteria. In Step 2, the key element of building a good uncertainty estimation schema is the selection of $\mathcal{M}(p)$, i.e., the dropout approach and probability distribution used to generate the mask sample w. The goal of our design is to obtain the

best uncertainty estimation schema $\mathcal{M}^*(p^*)$ regarding to the city application and its requirements. We first apply a set of uncertainty estimation schemas $\mathcal{M}_1(p), \mathcal{M}_2(p), ..., \mathcal{M}_n(p)$, such as, Bernoulli dropConnect and dropout, and Gaussian dropConnect and dropout. To briefly explain the differences, as shown in Figure 9, for Bernoulli dropConnect and dropout, the mask is sampled from a Bernoulli distribution, which return a 1 with probability p and 0 with a probability $(1-p)$. In Bernoulli dropout, one Bernoulli variable is sampled for each row of the weight matrix, i.e., $w_{i,*} \sim \mathcal{B}(p)$. In Bernoulli dropConnect, each element of the mask is sampled independently, i.e., $w_{i,j} \sim \mathcal{B}(p)$. This sets $W_{i,j}$ to $W'_{i,j}$ with probability p and 0 with a probability $(1-p)$. For Gaussian dropConnect and dropout, the mask is sampled from a Gaussian distribution, often using a mean of 1 and a standard deviation of $\sqrt{(1-p)/p}$ [34]. In Gaussian dropout, each element in a row has the same random variable, i.e., $w_{i,*} \sim \mathcal{N}(1, (1-p)/p)$. In Gaussian dropConnect, each element of the mask is sampled independently, i.e., $w_{i,j} \sim \mathcal{N}(1, (1-p)/p)$.

On the validation set of the data, for each schema $\mathcal{M}()$, we tune its own parameter p using the loss function defined by STL-U criteria. With the optimized p^*, the uncertainty models are updated as $\mathcal{M}_1(p^*), \mathcal{M}_2(p^*), ..., \mathcal{M}_n(p^*)$. Then we compare the loss of each schema with its best parameter. The combination with the lowest loss is selected as the best uncertainty estimation schema $\mathcal{M}^*_n(p^*)$. To be noted, the schema candidates can be more than these four dropout approaches. As a measurement, STL-Criteria can easily adapt to newly developed schemas.

Discussion: Researchers have been applying dropout to estimate the uncertainty of deep learning models. However, for the same deep learning model trained by the same data sets, different schemas with different parameters will end up with different uncertainty estimation. How to select the schemas and tune its parameters (i.e, dropout rate p) to better capture the model uncertainty is still an open question. In general, most applications just pick a dropout technique and set up its dropout rate by experience without systematically evaluate its influence on the uncertainty estimation. There is a very limited types of criterion on how to train or evaluate the regularization parameters. Accuracy (e.g., RMSE between the real value and mean of the estimated distribution, checking if the real value is within the predicted range) is commonly used as the only metric to evaluate the performance of both deep learning model. For example, as shown in Figure 10 (a) and (b), if the ground-truth value x is within the predicted interval $[\Phi_t^-(\varepsilon), \Phi_t^+(\varepsilon)]$ (the right figure), it is considered as an accurate uncertainty estimate. Otherwise (the left figure), it is not accurate. The problem of this metric is that it will overestimate the uncertainty, since the larger the uncertain interval is, the more accurate the prediction is. For example, comparing Figure 10 (a) and (b), if only measuring by accuracy, $(\mathcal{M}_2, p_2)$ expands the predicted interval yet increases the model uncertainty. However, the system will always tend to select $(\mathcal{M}_2, p_2)$ since it has a higher accuracy.

In this paper, we propose a new way to measure the uncertainty interval by checking it against a requirement. As shown in Figure 10 (c), the dark area represents a requirement. Obviously, the real value x does not satisfy the requirement. Intuitively, when predicting a number, if the real value does not satisfy a requirement, then the estimated interval should also not satisfy the same requirement.

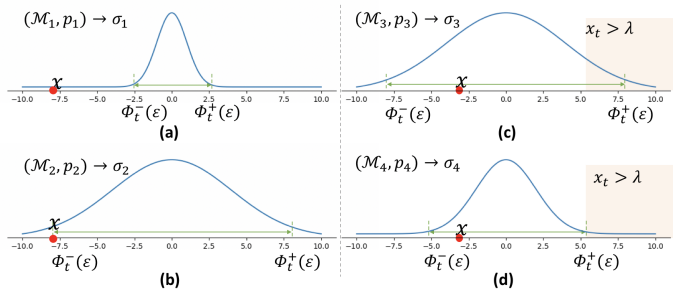


Figure 10: Comparison of Uncertainty Estimation Criteria

However, in Figure 10 (c), part of the predicted interval satisfies the requirement, but x could not be within $[\lambda, \Phi_t^+(\varepsilon)]$, i.e., this interval is invalid prediction. Therefore, the predicted uncertainty interval should be shorten. As shown in Figure 10 (d), the quality of uncertainty interval estimated by $(\mathcal{M}_4, p_4)$ is higher than $(\mathcal{M}_3, p_3)$. In addition, it is clearer that if the predictive flowpipe satisfies the requirement, which leads to a higher accuracy on the verification results. Decision makers are easier to tell if the predictive results satisfy the requirement or not.

6 EVALUATION

We conducted two set of experiments for evaluation: (1) using real city datasets to evaluate STL-U criteria for uncertainty estimation in Bayesian deep learning, and (2) using a simulated city case study to demonstrate the performance of CityPM for predictive monitoring. The experiments were run on a server machine with 20 CPUs, each core is 2.2GHz, and 4 Nvidia GeForce RTX 2080Ti GPUs. The operating system is Centos 7.

6.1 STL-U Criteria for Uncertainty Estimation

We apply STL-U criteria to train RNN models with real-world datasets: predictive monitoring of air quality in four big cities (Beijing, Tianjin, Guangzhou, and Shenzhen) in China, and traffic volumes in New York City, United States. The details of the datasets have been introduced in Section 3. With 80% data for model training, 10% data for uncertainty estimation training and 10% data for testing, the deep learning prediction generates 130,000 and 51,350 flowpipes for the air quality case and the traffic case, respectively.

6.1.1 Comparison among STL-U criteria. To show how STL-U criteria select and tune the uncertainty estimation schema, we first implement four schema (i.e., Bernoulli Dropout, Gaussian Dropout, Bernoulli DropConnect, and Gaussian DropConnect) with Monte Carlo method for LSTM predictor. In the training process, we specify the city requirements using STL-U, then convert it to a loss function, which later is used on the *validation set* to measure the quality of uncertainty estimation from different combinations of schemas and their parameter p. We show the intermediate products generated from two different criteria and two different requirements in Figure 11.

In all the sub-figures, the combinations of schema and parameters that achieve the best performance under the STL-U criteria

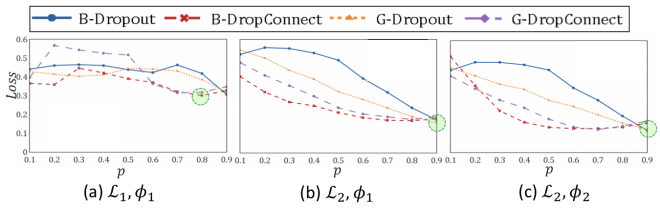


Figure 11: Intermediate products on the validation set generated from different criteria and different requirements. ($\phi_1 = \Box_I(x_\varepsilon < \lambda)$, $\phi_2 = \Diamond_I(x_\varepsilon < \lambda)$, the green circle highlighted the selected schema and parameter.)

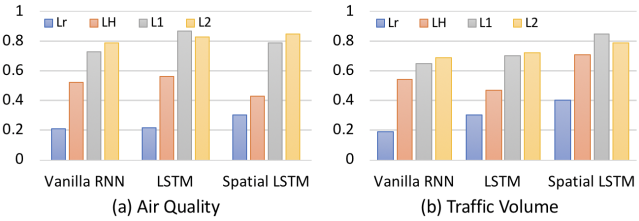


Figure 12: Comparison of F1-scores on the consistence of verification between predicted flowpipes and target sequences using different RNN-based prediction models with different loss functions

are highlighted in the green circle. We made the following observation. First, obviously, the loss from different combinations vary significantly, one SRT with good performance in one case does not always has the best performance in another case. Therefore, STL-U criteria training is a very important process to obtain the best schema. Secondly, comparing the results from two different loss functions (Figure 11 (a) and (b)), $\mathcal{L}_1$ and $\mathcal{L}_2$ select Bernoulli DropConnect with $p = 0.8$ and Gaussian Dropout with $p = 0.9$, respectively. Therefore, different criteria could end up with different results. Thirdly, comparing Figure 11 (b) and (c), which are trained with two different requirements and same criteria, ϕ_1 selects Gaussian Dropout with $p = 0.9$ while ϕ_1 selects Bernoulli Dropout with $p = 0.9$. However, the loss from these two combinations are very close in both cases. Therefore, different requirements could result in different models, but the results are shown to be relatively robust.

Summary : *STL-U criteria play an important role to support RNN models to select the best SRT and tune the parameter automatically for uncertainty estimation. The schema and parameter selected vary on different loss functions or requirements. Specifically, one can optimize the prediction module regards to different STL-U formulas and criteria based on the demands of the system to perform predictive monitoring.*

6.1.2 Comparison with the baseline criteria. We further compare the quality of uncertainty estimation on *testing set* between STL-U criteria ($\mathcal{L}_1$, $\mathcal{L}_2$) and the **baseline** approaches, including using four dropout approaches directly without uncertainty schema selection, which are the common approaches from state-of-the-art works, concrete dropout [10]; Two criteria without considering city requirements, the first one ($\mathcal{L}_r$) is the F1-score on checking if the predicted interval under a given confidence level covers the target sequence, and second one is Heteroscedastic loss [20] ($\mathcal{L}_H$), $\mathcal{L}_H = \frac{1}{N}\sum_{i=1}^{N} \frac{||y_i - \theta_i||^2}{2\sigma_i^2} + \frac{1}{2}\log\sigma_i^2$, which is a metric used in Bayesian deep learning to approximate heteroscedastic aleatoric uncertainty.

We evaluate the performance using four **metrics**, (1) F1-score of whether the predicted flowpipe has the same Boolean satisfaction over ϕ_3 as the target sequence, which is the requirement we used to train the uncertainty estimation schema (F1-ϕ_3); (2) F1-score of whether the predicted flowpipe has the same Boolean satisfaction over ϕ_4 as the target sequence, which is a new requirement that is not used in the training process (F1-ϕ_4). Here we have $\phi_3 : (x_\varepsilon < \lambda_1)\mathcal{U}_I(x_\varepsilon < \lambda_2)$ and $\phi_4 : \Box_I(x_\varepsilon < \lambda_3 \wedge x_\varepsilon > \lambda_4)$. (3) Accuracy of whether the predicted flowpipe covers the target sequence (Acc-range); (4) Heteroscedastic Loss (HeterLoss).

The results on the two datasets are shown in Table 2. From the table we can see that (1) all five dropout approaches with p selected based on prediction accuracy achieve a relatively high Acc-range, but very low F1-scores (about 0.25 on average) on requirements verification, which means the system can barely use the predicted flowpipes to check if the requirements are satisfied or not. It also indicates the shortcomings of metrics on individual values. (2) Comparing with the other two criteria, both STL-U based criteria achieve much higher F1 scores on both trained and non-trained requirements. It indicates that the models trained by STL-U requirements can really obtain a high quality of uncertainty estimation, not only for the trained requirements.

Summary : *Uncertainty estimation schemas trained by STL-U criteria achieve a similar accuracy in checking whether the target value is within the predicted interval. However, they achieve much higher performance on checking if the predicted flowpipe satisfies the same system requirement that the target sequence follows. It is significant for city decision-making in real-time based on the predictive monitoring results. Moreover, as a measurement, STL-U criteria can select uncertainty estimation schemas from a larger pool of candidates than the four dropout approaches in the evaluation. It can adapt to newly developed estimation schema easily.*

6.1.3 Performance on different RNNs. To test how STL-U criteria work on different RNN models, we apply it to three different types of Bayesian RNN models: the vanilla RNN, LSTM [15], and SpatialLSTM [21] To briefly introduce the models, the vanilla RNN is a classic RNN model, and the LSTM model is a special RNN model with forget gates, and the SpatialLSTM is built on LSTM, treating the geographic information as a feature to learn. For each RNN, we train the uncertainty estimation model using three STL-U loss functions, and compare the testing results with the baseline loss function.

First, we compare the results by checking if the target sequence is within the predicted flowpipes. We found that the F1 scores between different schemas using different criteria are very close. For example, for the air quality prediction, the highest F1 score in the air quality prediction is 0.835 when using LSTM predictor with $\mathcal{L}_r$ and lowest one is 0.783 when using Vanilla RNN predictor with $\mathcal{L}_2$. It indicates that STL-U criteria have a similar accuracy on the flowpipe coverage. Next, we compare F1-scores on checking the consistence

Table 2: Performance comparison with baseline approaches

	Air Pollution Index						Traffic Volume					
	Selected Schema	p	**HeterLoss**	**Acc-Range**	**F1-ϕ_3**	**F1-ϕ_4**	**Selected Schema**	p	**HeterLoss**	**Acc-Range**	**F1-ϕ_3**	**F1-ϕ_4**
B-Dropout	-	0.81	183.9	0.67	0.34	0.43	-	0.5	0.63	0.79	0.17	0.21
B-DropConnect	-	0.53	121.0	0.69	0.22	0.32	-	0.74	**0.2**	0.38	0.51	0.6
G-Dropout	-	0.45	152.8	0.76	0.10	0.29	-	0.5	0.66	0.79	0.17	0.19
G-DropConnect	-	0.58	129.4	0.78	0.12	0.28	-	0.54	0.247	0.56	0.44	0.55
Concrete Dropout	-	-	128.6	**0.91**	0.69	0.52	-	-	2.96	**1.00**	0.11	0.15
$\mathcal{L}_r$	B-DropConnect	0.53	121.0	0.69	0.22	0.19	B-DropConnect	0.74	0.23	0.38	0.51	0.32
$\mathcal{L}_H$	G-Dropout	0.5	**119.2**	0.81	0.65	0.48	G-Dropout	0.5	0.7	0.79	0.17	0.39
STL-U $\mathcal{L}_1$	G-DropConnect	0.81	154.1	0.80	**0.81**	**0.74**	B-DropConnect	0.58	0.24	0.51	0.67	**0.58**
STL-U $\mathcal{L}_2$	B-DropConnect	0.73	165.4	0.79	0.76	0.63	B-Dropout	0.9	0.3	0.78	**0.68**	0.39

of requirements verification between predicted flowpipes and target sequences using three RNNs trained by different loss functions, as shown in Figure 12. The results show that all STL-U criteria outperform the accuracy based criterion significantly. For example, averagely, F1 scores with STL-U criteria ($\mathcal{L}_1$, $\mathcal{L}_2$) are 2.33 times and 1.6 times more than $\mathcal{L}_r$ in air quality prediction and traffic volume prediction, respectively. In addition, the overall prediction performance is also affected by the RNN predictors. For example, LSTM and SpatialLSTM has a better performance than the other two when predicting traffic volume.

Summary : *STL-U criteria are agnostic to the choice of different RNN models. Nowadays, researchers are developing various RNN models to achieve better performance for different applications. It is of great significance that our STL-U criteria can improve the quality of uncertainty estimation for various RNNs.*

6.2 Real-time Predictive Monitoring for a Simulated Smart City

We set up a smart city simulation of New York City using the Simulation of Urban MObility (SUMO) [4] with the real city data [32], on top of which, we implement 10 smart services (S1: Traffic Service, S2: Emergency Service, S3: Accident Service, S4: Infrastructure Service, S5: Pedestrian Service, S6: Air Pollution Control Service, S7: PM2.5/PM10 Service, S8: Parking Service, S9: Noise Control Service, and S10: Event Service). We build a prototype implementation of STL-U monitor and apply it to conduct a real-time predictive monitoring and control. We use LSTM predictor to predict the future city states with actions requested by services. When a violation is detected, the system will try to reject or adjust the requested actions to avoid the violation. We trained the uncertainty estimation model using STL-U loss function $\mathcal{L}_1$. We specify 5 requirements to train the prediction model with uncertainty estimation and 390 real-time requirements on different variables (e.g., CO, traffic volume, vehicle waiting time, etc.) and locations to monitor. For the **baseline** approach, we use a deterministic LSTM predictor with a STL Monitor. We simulate the city running for 30 days in three control sets, one without any monitor, one with the STL monitor and one with the STL-U monitor.

6.2.1 Effectiveness and Efficiency. We check the predictive flowpipes against their requirements using strong and weak Boolean semantics with the confidence levels equaling to 0.95. Then we calculate the confidence levels that guarantee the strong/weak satisfaction. The individual monitoring results of each pair of flowpipe and their requirement are intuitive and helpful to support the decision-makers for air quality and traffic control. For example, the monitoring results in one case show (1) the requirement is weakly, but not strongly satisfied under 0.95 confidence level, and (2) the confidence levels for a strong satisfaction is lower than 0.86, and weak satisfaction is 1. The understanding from these results for the decision-makers are (1) at least one sequence from the flowpipe with 0.95 confidence level satisfies the requirement, but not all the sequences will satisfy; and (2) the flowpipe with 0.86 confidence level will completely satisfy the requirement, and at least one sequence from the flowpipe will satisfy the requirement as long as the confidence level is larger than 0. With the support of these results, the decision-maker can decide whether or not to take some actions to prevent the violation (e.g., release traffic congestion controls, control air pollution emission).

Our results show that CityPM is efficient in handling a large number of flowpipes and requirements. It only takes 417s to check 130,000 air quality flowpipes with prediction length of the flowpipe $T = 8$. The computing time is even shorter (281.3s for Strong Boolean) when checking the Boolean semantics, which will terminate early if the algorithm reaches the Boolean results before going through the whole flowpipe. Naively, using a CityPM to check the same number of predicted sequences from deep learning model independently, with the dropout sample sequences $N = 100$, it takes about 5 hours to check the air quality sequences.

Summary : *CityPM is effective in monitoring the flowpipes resulting from Bayesian deep learning models. It is efficient to monitor a large scale of flowpipes resulting from Bayesian deep learning models, e.g., on average, it only takes 417 seconds to monitor 130,000 flowpipes with an 8-time unites prediction against 390 requirements, which cannot be monitored by STL or its variants.*

6.2.2 Overall Performance. The overall results are shown in Table 3. We measure the city performance from the domains of transportation, environment, emergency and public safety using **metrics** including number of violations, average AQI, noise level, traffic volume, waiting time of emergency vehicles, other vehicles, and pedestrians.

The number of violations detected by CityPM is less than STL monitor. For the deterministic model, the monitor results are more likely to be affected by the inaccurate predicted sequence. As a

Table 3: Comparison of the City Performance with the STL Monitor and CityPM

	No Monitor	STL Monitor	**CityPM**
Number of Violation	undetected	267	**189**
Air Quality Index	67.91	57.22	**43.65**
Noise (db)	73.32	49.27	**48.21**
Emergency Waiting Time (s)	20.32	14.87	**10.65**
Vehicle Waiting Number	22	18	**15**
Pedestrian Waiting Time (s)	190.2	148.9	**121.1**
Vehicle Waiting Time (s)	112.12	89.77	**80.31**

result, more false violations are detected. While with CityPM monitoring on the predicted flowpipe, the confidence interval is less sensitive to the inaccurate prediction. On the other hand, some violations are not detected by STL monitor when the predictive result is only one sampled sequence. However, considering the flowpipe is generated from N sampled sequence, it also better reflects the real changing of the predicted states. In summary, the city's safety and performance is improved, for example, the air quality index is reduced by 23.7%, and emergency waiting time is reduced by 28.3% comparing to the monitor without considering uncertainty.

Summary : *CityPM reduces the number of false violation detection, and improves the safety and performance of a smart city comparing to the monitor without considering uncertainty.*

7 RELATED WORK

Predictive Monitoring of Smart Cities: Prediction and runtime monitoring play important roles in supporting smart city safety control and decision making. Deep learning models are popularly applied to predict city future states by services across different domains. For example, [24] forecasts the air quality with a linear regression-based temporal predictor and a neural network-based spatial predictor; [16] predicts users' future destination with their spatio-temporal behavior patterns and provide personalized GIS services to avoid congested roads; [35] builds a deep learning based framework for citywide fire risk forecasting. Several recent work have proposed temporal logic based solutions to monitor smart cities' operations. For example, [28] uses STL to support the specification-based monitoring of safety and performance requirements of smart cities; [12] monitors the city power grid. However, none of these existing work considers the predictive monitoring of smart cities, i.e., using temporal logic to monitor the prediction output of deep learning with uncertainty.

Uncertainty Estimation in Deep Learning: While most deep learning models do not offer the uncertainty of their predictions [8], works that capture the uncertainty (or confidence) of the prediction can be dated to the early development of neural networks in the 90s'. Bayesian Neural Network [29] represents a probabilistic model that infers a distribution as output. Bayesian Neural Network is known to be robust and resilient to overfitting. However, the hardness of inference prevents the prevalence of the model in practice. Following these directions, several works [11, 14] use variational inference to perform an approximated inference on Bayesian Neural Networks. Aside from variational inference, Monte Carlo Dropout is another approach to obtain uncertainty estimation of the model [9, 37]. By exploiting the dropout structure in the deep neural network, these approaches turn the original Neural Network model into a simple Bayesian Neural Network without changing the structure and apply approximated inference with the Monte Carlo approach. Existing works [20, 36, 37] mostly focus uncertain estimation on single-time classification or regression tasks. This paper focuses on the case of time series prediction. Moreover, in contrast to previous measures of uncertainty that are rather empirical, our work proposes a formal framework to model and define requirements to the output distribution. Our work can thus be used to give a confidence guarantee of the model prediction and evaluate the quality of the uncertainty estimation.

Temporal Logic Monitoring: In the last decade, there has been a great effort to develop tools and monitoring techniques [3] for Signal Temporal Logic [30] (STL) and its extensions. Examples of popular tools available for monitoring STL are Breach [6] and S-Taliro [1]. However, most of the literature focuses on monitoring a single multi-variables signal. This is a limiting factor when we need to monitor predictive models and to reason about uncertainty. Recent attempts to handle uncertainty focus on extending STL with the possibility to incorporate random variables in predicates and to express temporal and Boolean operations over such predicates. Examples are C2TL [19], PrSTL [33], StSTL [23] and StTL [22]. Our approach differs from these previous works, because instead of considering the probability to satisfy a constraint as atomic proposition, it reasons over the envelope of all the possible uncertain trajectories that we call *flowpipe*. The projection of a flowpipe for a single point of time is the interval of values that the possible trajectories can assume in that point. The notion of robustness for STL specification has been explored extensively in several previous papers where different quantitative semantics [7, 18] can be used to interpret the specification: a real value indicating how much a given signal satisfies/violates a given STL requirement according to a chosen metric. In our setting, the notion of robustness is associated to a measure of how much we can trust the prediction of satisfaction given the prediction. Similarly to other previous works on STL, we also show that our quantitative semantics is sound and correct with respect to the qualitative one.

8 CONCLUSION

We developed a novel predictive monitoring system (CityPM) for smart cities. CityPM consists of a RNN-based Bayesian prediction model that continuously generates sequential predictions of future city states, and a novel monitor that verifies if the generated predictions satisfy city safety and performance requirements. To obtain well-calibrated uncertainty estimation schema for predictive monitoring, we created a new logic (STL-U) for reasoning about the correctness of predicted flowpipes. The results show that STL-U criteria leads to improved uncertainty calibration in various Bayesian deep learning models, and CityPM significantly improves the simulated city's safety and performance.

REFERENCES

[1] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. 2011. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *Proc. of TACAS 2011 (LNCS, Vol. 6605)*. 254–257.
[2] Andreas Antoniou. 2016. *Digital signal processing.* McGraw-Hill.
[3] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan. 2018. Specification-based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In *Lectures on Runtime*

Verification. LNCS, Vol. 10457. Springer, 135–175. https://doi.org/10.1007/978-3-319-75632-5_5
[4] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011*. ThinkMind.
[5] Frank V Cespedes, Allison M Ciechanover, and Margot Eiran. 2018. BreezoMeter: Making Air Pollution Data Actionable. (2018).
[6] Alexandre Donzé. 2010. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Proc. of CAV 2010 (LNCS, Vol. 6174)*. Springer, 167–170.
[7] Georgios E. Fainekos and George J. Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.* 410, 42 (2009), 4262–4291. https://doi.org/10.1016/j.tcs.2009.06.021
[8] Yarin Gal. 2016. *Uncertainty in deep learning*. Ph.D. Dissertation. PhD thesis, University of Cambridge.
[9] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. 1050–1059.
[10] Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. In *Advances in neural information processing systems*. 3581–3590.
[11] Alex Graves. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*. 2348–2356.
[12] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Grosu, and Calin Belta. 2015. SpaTeL: a novel spatial-temporal logic and its applications to networked systems. In *Proc. of HSCC'15: the 18th International Conference on Hybrid Systems: Computation and Control*. IEEE, 189–198.
[13] Suining He and Kang G Shin. 2020. Towards Fine-grained Flow Forecasting: A Graph Attention Approach for Bike Sharing Systems. In *Proceedings of The Web Conference 2020*. 88–98.
[14] Geoffrey E Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*. 5–13.
[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[16] Ryo Imai, Kota Tsubouchi, Tatsuya Konishi, and Masamichi Shimosaka. 2018. Early destination prediction with spatio-temporal user behavior patterns. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–19.
[17] Intel. [n.d.]. *Paving the way forward: Intelligent Traffic Management System*.
[18] Stefan Jaksic, Ezio Bartocci, Radu Grosu, Thang Nguyen, and Dejan Nickovic. 2018. Quantitative monitoring of STL with edit distance. *Formal Methods in System Design* 53, 1 (2018), 83–112.
[19] Susmit Jha, Vasumathi Raman, Dorsa Sadigh, and Sanjit A. Seshia. 2018. Safe Autonomy Under Perception Uncertainty Using Chance-Constrained Temporal Logic. *J. Autom. Reasoning* 60, 1 (2018), 43–62. https://doi.org/10.1007/s10817-017-9413-9
[20] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision?. In *Advances in neural information processing systems*. 5574–5584.
[21] Dejiang Kong and Fei Wu. 2018. HST-LSTM: a hierarchical spatial-temporal long-short term memory network for location prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2341–2347.
[22] Panagiotis Kyriakis, Jyotirmoy V. Deshmukh, and Paul Bogdan. 2019. Specification Mining and Robust Design under Uncertainty: A Stochastic Temporal Logic Approach. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 96 (Oct. 2019).
[23] Jiwei Li, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli, Yugeng Xi, and Dewei Li. 2017. Stochastic Contracts for Cyber-Physical System Design under Probabilistic Requirements. In *Proc. MEMOCODE '17* (Vienna, Austria). 5–14. https://doi.org/10.1145/3127041.3127045
[24] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction.. In *IJCAI*. 3428–3434.
[25] Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin. 2019. Contextualized spatial–temporal network for taxi origin-destination demand prediction. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3875–3887.
[26] Meiyi Ma, Ezio Bartocci, Eli Lifland, John Stankovic, and Lu Feng. 2020. SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 51–62.
[27] Meiyi Ma, Sarah Preum, Mohsin Ahmed, William Tärneberg, Abdeltawab Hendawi, and John Stankovic. 2019. Data sets, modeling, and decision making in smart cities: A survey. *ACM Transactions on Cyber-Physical Systems* 4, 2 (2019), 1–28.
[28] Meiyi Ma, John A Stankovic, and Lu Feng. 2018. Cityresolver: a decision support system for conflict resolution in smart cities. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE Press, 55–64.
[29] David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472.
[30] O. Maler and D. Ničković. 2004. Monitoring Temporal Properties of Continuous Signals. In *Proc. of FORMATS/FTRTFT*. Springer, 152–166.
[31] Microsoft. 2020. Microsoft CityNext.
[32] NYC.gov. [n.d.]. *New York City Open Data*. https://nycopendata.socrata.com/.
[33] Dorsa Sadigh and Ashish Kapoor. 2016. Safe Control under Uncertainty with Probabilistic Signal Temporal Logic. In *Robotics: Science and Systems XII, 2016*.
[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
[35] Qianru Wang, Junbo Zhang, Bin Guo, Zexia Hao, Yifang Zhou, Junkai Sun, Zhiwen Yu, and Yu Zheng. 2019. CityGuard: Citywide Fire Risk Forecasting Using A Machine Learning Approach. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–21.
[36] Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7322–7329.
[37] Lingxue Zhu and Nikolay Laptev. 2017. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 103–110.

APPENDIX

In the appendix, we first present the monitoring algorithms for STL-U strong and weak Boolean semantics (Algorithm 1 and 2), then show the algorithms for calculating the confidence levels of strong and weak satisfaction (Algorithm 3 and 4), respectively. Then, we give an example to illustrate how to calculate the confidence levels.

We show the process of calculating the confidence levels for strong and weak satisfactions in Figure 13. In Figure 13(a), we query what confidence level that $\square_{[1,3]}(x_\varepsilon > 8)$ is strong (left part of the syntax tree) or weak (right part of the syntax tree) satisfied. First, at each time $t \in [1, 3]$, we calculate the confidence levels that guarantees the strong satisfaction of $(x_\varepsilon > 8)$, which are [0,0.38], [0,0.68], [0,0.68]. Next, to calculate $\square_{[1,3]}\phi$, we calculate the intersection of these intervals, which leads to [0,0.38]. Similarly, we get the confidence for strong and weak satisfaction of $\square_{[1,3]}(x_\varepsilon > 10)$ in Figure 13(b), which are [0,0] and [0.38,1], respectively.

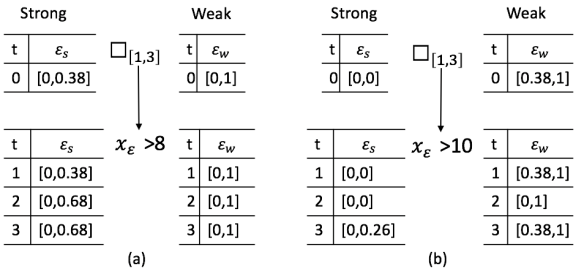


Figure 13: Calculating Confidence Levels

Algorithm 3: Confidence Level of Strong Satisfaction StrongConfidenceLevel(φ, ω, t)

Function StrongConfidenceLevel(φ, ω, t):
Input :STL-U Requirement φ, flowpipe $\omega = \{\Phi_t\}$, time t
Output:Confidence Level of Strong Satisfaction FloatInterval :$\epsilon_s \subseteq [0, 1]$
begin
switch φ **do**
Case μ_x
if $\theta_t > 0$ **then**
$\varepsilon \leftarrow \int_0^{2\theta} \Phi_t(x)dx$
$\epsilon_s \leftarrow [0, \varepsilon]$
return ϵ_s;
else
return $\emptyset$;
end
Case $\neg\varphi$
$\epsilon_s \leftarrow$ WeakConfidenceLevel^C
return ϵ_s;
Case $\varphi_1 \wedge \varphi_2$
return StrongConfidenceLevel$(\varphi_1, \omega, t) \cap$ StrongConfidenceLevel(φ_2, ω, t)
Case $\varphi_1 \mathcal{U}_I \varphi_2$
$\epsilon_s \leftarrow \emptyset$
for $t' \in (t + I)$ **do**
$\epsilon'_s \leftarrow$ StrongConfidenceLevel(φ_2, ω, t')
for $t'' \in [t, t']$ **do**
$\epsilon'_s \leftarrow \epsilon'_s \cap$ StrongConfidenceLevel(φ_1, ω, t'')
end
$\epsilon_s \leftarrow \epsilon_s \cup \epsilon'_s$
end
return ϵ_s;
end
end

Algorithm 4: Confidence Level of Weak Satisfaction WeakConfidenceLevel(φ, ω, t)

Function WeakConfidenceLevel(φ, ω, t):
Input :STL-U Requirement φ, flowpipe $\omega = \{\Phi_t\}$, time t
Output:Confidence Level of Weak Satisfaction FloatInterval :$\epsilon_w \subseteq [0, 1]$
begin
switch φ **do**
Case μ_x
if $\theta_t < 0$ **then**
$\varepsilon \leftarrow \int_0^{2\theta} \Phi_t(x)dx$
$\epsilon_w \leftarrow [\varepsilon, 1)$
return ϵ_w;
else
return $[0, 1]$;
end
Case $\neg\varphi$
$\epsilon_w \leftarrow$ StrongConfidenceLevel^C
return ϵ_w;
Case $\varphi_1 \wedge \varphi_2$
return WeakConfidenceLevel$(\varphi_1, \omega, t) \cap$ WeakConfidenceLevel(φ_2, ω, t)
Case $\varphi_1 \mathcal{U}_I \varphi_2$
$\epsilon_w \leftarrow \emptyset$
for $t' \in (t + I)$ **do**
$\epsilon'_w \leftarrow$ WeakConfidenceLevel(φ_2, ω, t')
for $t'' \in [t, t']$ **do**
$\epsilon'_w \leftarrow \epsilon'_w \cap$ WeakConfidenceLevel(φ_1, ω, t'')
end
$\epsilon_w \leftarrow \epsilon_w \cup \epsilon'_w$
end
return ϵ_w;
end
end

Algorithm 1: STL-U Strong Boolean monitoring algorithm StrongBoolean(φ, ω, t)

Function StrongBoolean(φ, ω, t):
Input :STL-U Requirement φ, flowpipe $\omega = \{\Phi_t\}$, time t
Output:Strong Boolean Satisfaction Boolean : StrongSat
begin
switch φ **do**
Case $\mu_x(\varepsilon)$
if $f(\Phi_t^+(\varepsilon)) > 0$ **then**
return True ;
else
return False ;
end
Case $\neg\varphi$
return $\neg$WeakBoolean(φ, ω, t);
Case $\varphi_1 \wedge \varphi_2$
return StrongBoolean$(\varphi, \omega, t) \wedge$ StrongBoolean(φ, ω, t)
Case $\varphi_1 \mathcal{U}_I \varphi_2$
StrongSat $\leftarrow$ True;
for $t' \in (t + I)$ **do**
if StrongBoolean(φ_2, ω, t') **then**
for $t'' \in [t, t']$ **do**
StrongSat $\leftarrow$ StrongSat $\wedge$ StrongBoolean(φ_1, ω, t'');
if $\neg$StrongSat **then**
break;
end
end
if StrongSat **then**
return True;
end
end
end
return False;
end
end

Algorithm 2: STL-U Weak Boolean monitoring algorithm WeakBoolean(φ, ω, t)

Function WeakBoolean(φ, ω, t):
Input :STL-U Requirement φ, flowpipe $\omega = \{\Phi_t\}$, time t
Output:Weak Boolean Satisfaction Boolean : WeakSat
begin
switch φ **do**
Case $\mu_x(\varepsilon)$
if $f(\Phi_t^-(\varepsilon)) > 0$ **then**
return True ;
else
return False ;
end
Case $\neg\varphi$
return $\neg$StrongBoolean(φ, ω, t);
Case $\varphi_1 \wedge \varphi_2$
return WeakBoolean$(\varphi, \omega, t) \wedge$ WeakBoolean(φ, ω, t)
Case $\varphi_1 \mathcal{U}_I \varphi_2$
WeakSat $\leftarrow$ True;
for $t' \in (t + I)$ **do**
if WeakBoolean(φ_2, ω, t') **then**
for $t'' \in [t, t']$ **do**
WeakSat $\leftarrow$ WeakSat $\wedge$ WeakBoolean(φ_1, ω, t'');
if $\neg$WeakSat **then**
break;
end
end
if WeakSat **then**
return True;
end
end
end
return False;
end
end