Neural Predictive Monitoring

Luca Bortolussi[1,4], Francesca Cairoli[1], Nicola Paoletti[2], Scott A. Smolka[3], and Scott D. Stoller[3]

[1] Department of Mathematics and Geosciences, Università di Trieste, Italy
[2] Department of Computer Science, Royal Holloway, University of London, UK
[3] Department of Computer Science, Stony Brook University, USA
[4] Modelling and Simulation Group, Saarland University, Germany

Abstract. Neural State Classification (NSC) is a recently proposed method for runtime predictive monitoring of Hybrid Automata (HA) using deep neural networks (DNNs). NSC trains a DNN as an approximate *reachability predictor* that labels a given HA state x as *positive* if an unsafe state is reachable from x within a given time bound, and labels x as *negative* otherwise. NSC predictors have very high accuracy, yet are prone to prediction errors that can negatively impact reliability. To overcome this limitation, we present *Neural Predictive Monitoring* (NPM), a technique based on NSC and *conformal prediction* that complements NSC predictions with statistically sound estimates of uncertainty. This yields principled criteria for the rejection of predictions likely to be incorrect, without knowing the true reachability values. We also present an active learning method that significantly reduces both the NSC predictor's error rate and the percentage of rejected predictions. Our approach is highly efficient, with computation times on the order of milliseconds, and effective, managing in our experimental evaluation to successfully reject almost all incorrect predictions.

1 Introduction

Hybrid systems are a central model for many safety-critical, cyber-physical system applications [2]. Their verification typically amounts to solving a hybrid automata (HA) reachability checking problem [14]: given a model $\mathcal{M}$ of the system expressed as an HA and a set of unsafe states U, check whether U is reached in any (time-bounded) path from a set of initial states of $\mathcal{M}$. Due to its high computational cost, reachability checking is usually limited to design-time (offline) analysis.

Our focus is on the online analysis of hybrid systems and, in particular, on the *predictive monitoring* (PM) problem [10]; i.e., the problem of predicting, *at runtime*, whether or not an unsafe state can be reached from the current system state within a given time bound. PM is at the core of architectures for runtime safety assurance such as Simplex [26], where the system switches to a safe fallback mode whenever PM indicates the potential for an imminent failure.

In such approaches, PM is invoked periodically and frequently, and thus reachability needs be determined rapidly, from a single state (the current system state), and typically for short time horizons. This is in contrast with offline reachability checking, where long or unbounded time horizons and sizable regions of initial states are typically considered.

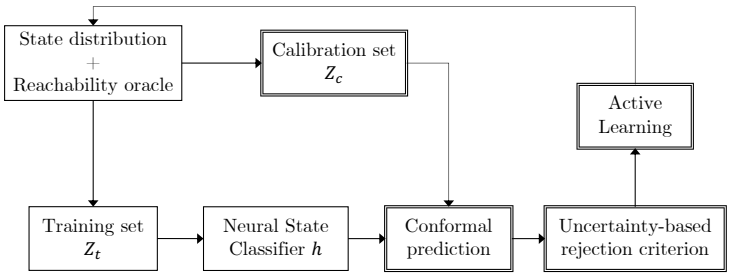

Fig. 1. Overview of the NPM framework. Double-bordered components denote extensions to the method of [21]. Training of the neural state classifier h and retraining via active learning are performed offline. The only components used at runtime are the classifier h and the rejection criterion.

PM also differs from traditional runtime verification in that PM is *preemptive*: it detects potential safety violations before they occur, not when or after they occur.

Any solution to the PM problem involves a tradeoff between two main requirements: *accuracy* of the reachability prediction, and computational *efficiency*, as the analysis must execute within strict real-time constraints and typically with limited hardware resources. In this work, we present *Neural Predictive Monitoring* (NPM), a machine learning-based approach to PM that provides *high efficiency and accuracy*, and crucially, *statistical guarantees on the prediction error*.

NPM builds on *Neural State Classification* (NSC) [21], a recently proposed method for approximate HA reachability checking using deep neural networks (DNNs). NSC works by training DNNs as state classifiers using examples computed with an oracle (an HA model checker). For any state x of the HA, such a classifier labels x as *positive* if an unsafe state is reachable from x within a given time bound; otherwise, x is labeled as *negative*. Executing a neural state classifier corresponds to computing the output of a DNN for a single input, and thus is extremely efficient. NSC has also demonstrated very high accuracy in reachability predictions, owing to the powerful approximation capabilities of DNNs. Some classification errors are, however, unavoidable, the most important being *false negatives*, in which positive states are misclassified as negative. Such errors may compromise the safety of the system.

NPM overcomes this problem by extending NSC with rigorous methods for quantifying the uncertainty of NSC predictions. NPM can consequently identify and reject predictions that are likely to produce classification errors. For this purpose, we investigate the use of *Conformal Prediction* (CP) [27], a method that provides statistical guarantees on the predictions of machine-learning models. Importantly, CP requires only very mild assumptions on the data[5], which makes it suitable for state classification of HA models.

Figure 1 provides an overview of the NPM approach. We sample from a distribution of HA states to generate a training set Z_t and a calibration set Z_c. An HA reachability oracle (a model checker or, for deterministic systems, a simulator) is used to label

[5] The only assumption is exchangeability, a weaker version of the independent and identically distributed assumption.

sampled states as positive or negative. A neural state classifier h (i.e., a DNN-based binary classifier) is derived from Z_t via supervised learning.

We use CP to estimate two statistically sound measures of prediction uncertainty, *confidence* and *credibility*. Informally, the confidence of a prediction is the probability that a reachability prediction for an HA state s corresponds to the true reachability value of s. Credibility quantifies how likely a given state is to belong to the same distribution of the training data.

Using confidence and credibility, we derive criteria for anomaly detection, that is, for rejecting NSC predictions that are likely to be erroneous. The rejection criterion is based on identifying, via support vector classification, confidence and credibility thresholds that optimally separate incorrect and correct predictions. The key advantage of such an approach is that predictions are rejected on rigorous statistical grounds. Furthermore, computation of CP-based confidence and credibility is very efficient (approximately 1 ms in our experiments), which makes our NPM method suitable for online analysis and PM.

Finally, our approach includes an active learning strategy to improve the reliability of the state classifier h. The idea is to employ the CP-based rejection criterion to identify HA states for which h yields uncertain predictions, and augment the training and calibration sets with those states. We then train a new state classifier with the augmented dataset, thus ensuring improved accuracy on the HA states where h performed poorly, and in turn, a reduced rejection rate. As opposed to simple random sampling of the state distribution, an advantage of our active learning strategy is its parsimony: by focusing on the states with uncertain predictions, it requires a significantly smaller number of additional re-training samples to achieve a given reduction in rejection rate, and thus significantly reduces the cost of re-training. The active learning procedure can be iterated, as shown in Fig. 1. We stress that these re-training iterations are part of the training process, which is performed offline and hence does not affect runtime performance.

In summary, the main contributions of this paper are the following:

- We develop Neural Predictive Monitoring, a framework for runtime predictive monitoring of hybrid automata that extends neural state classification with conformal prediction.
- We derive statistically sound and optimal criteria for rejecting unreliable NSC predictions, which leverage CP-based measures of prediction uncertainty.
- We develop an active learning method designed to reduce both prediction errors and the rejection rate.
- We evaluate the method on five case studies, demonstrating that our optimal rejection criteria successfully rejects almost all prediction errors (missing an average of only 1.4 errors out of 49.4 over a total of 50,000 samples), and that a single iteration of our active learning strategy reduces the range of prediction errors from 15.2–82 to 3.8–22.8, and the range of overall rejection rates from 3.46%–9.88% to 0.51%–2.74%. The ranges are taken over the set of case studies, and the results for each case study are averaged over 5 runs.

2 Problem Formulation

We describe the predictive monitoring problem for hybrid automata reachability and the related problem of finding an optimal criterion for rejecting erroneous reachability predictions. We assume that the reader is familiar with the definitions of HA and HA reachability. These definitions can be found in e.g. [21].

Problem 1 (Predictive monitoring for HA reachability). Given an HA $\mathcal{M}$ with state space X, time bound T, and set of unsafe states $U \subset X$, find a *predictor* h^*, i.e., a function $h^* : X \to \{0,1\}$ such that for all $x \in X$, $h^*(x) = 1$ if $\mathcal{M} \models \mathsf{Reach}(U,x,T)$, i.e., if it is possible for $\mathcal{M}$, starting in x, to reach a state in U within time T; $h^*(x) = 0$ otherwise.

A state $x \in X$ is called *positive* if $\mathcal{M} \models \mathsf{Reach}(U,x,T)$. Otherwise, x is *negative*.

The neural state classification method of [21] provides an approximate solution to the above PM problem[6], a solution based on deep neural networks (DNNs). NSC assumes a distribution $\mathcal{X}$ of HA states and derives a DNN-based reachability predictor h using supervised learning, where the training inputs are sampled according to $\mathcal{X}$ and labeled using a reachability oracle. Being an approximate solution, h can commit prediction errors: a state $x \in X$ is a *false positive* (FP) if $h(x) = 1$ but $\mathcal{M} \not\models \mathsf{Reach}(U,x,T)$; x is a *false negative* (FN) if $h(x) = 0$ but $\mathcal{M} \models \mathsf{Reach}(U,x,T)$. These errors are respectively denoted by predicates $fn(x)$ and $fp(x)$. Predicate $pe(x) = fn(x) \vee fp(x)$ denotes a generic prediction error.

A central objective of this work is to derive, given a predictor h, a rejection criterion R able to identify states x that are wrongly classified by h. Importantly, for runtime applicability, R should not require knowing the true reachability value of x, as computing it would be too costly at runtime. Further, R should be optimal, that is, it should ensure minimal probability of rejection errors w.r.t. the state distribution $\mathcal{X}$.

Problem 2. Given an approximate reachability predictor h, a state distribution $\mathcal{X} : X \to [0,1]$, and $e \in \{pe, fn, fp\}$, find an optimal rejection rule $R : X \to \{0,1\}$, i.e., such that it minimizes the probability $P_{x \sim \mathcal{X}}(e(x) \neq R(x))$.

Note that Problem 2 requires specifying the kind of prediction errors to reject. Indeed, depending on the application at hand, one might desire to reject only a specific kind of errors. For instance, in safety-critical applications, FNs are the most critical errors while FPs are less important.

As we will explain in Section 4, our solution to Problem 2 will consists in identifying optimal rejection thresholds for confidence and credibility, two statistically sound measures of prediction uncertainty based on CP. The statistical guarantees of our approach derive from using these uncertainty measures as the basis of the rejection criterion.

3 Conformal Prediction for Classification

Conformal Prediction (CP) associates measures of reliability to any traditional supervised learning problem. It is a very general approach that can be applied across all

[6] In [21], the PM problem is called "state classification problem", and its solution a "state classifier".

existing classification and regression methods [5]. Since we are interested in the analysis of the DNN-based state classifiers of NSC, we present the theoretical foundations of CP in relation to a generic classification problem.

Let X be the input space, $Y=\{y^1,...,y^c\}$ be the set of labels (or classes), and define $Z=X\times Y$. The classification model is represented as a function $h:X\to[0,1]^c$ mapping inputs into a vector of class likelihoods, such that the class predicted by h corresponds to the class with the highest likelihood. In the context of PM of HA reachability, X is the HA state space, $Y=\{0,1\}$ ($c=2$) indicates the possible reachability values, and h is the predictor[7].

Let us introduce some notation: for a generic input x_i, we denote with y_i the true label of x_i and with $\hat{y}_i$ the label predicted by h. Test points, whose true labels are unknown, are denoted by x_*.

The CP algorithm outputs *prediction regions*, instead of single point predictions: given a significance level $\varepsilon\in(0,1)$ and a test point x_i, its prediction region, $\Gamma_i^\varepsilon\subseteq Y$, is a set of labels guaranteed to contain the true label y_i with probability $1-\varepsilon$. The main ingredients of CP are: a *nonconformity function* $f:Z\to\mathbb{R}$, a set of labelled examples $Z'\subseteq Z$, a classification model h trained on a subset of Z', and a statistical test. The nonconformity function $f(z)$ measures the "strangeness" of an example $z_i=(x_i,y_i)$, i.e., the deviation between the label y_i and the corresponding prediction $h(x_i)$.

3.1 CP algorithm for classification

Given a set of examples $Z'\subseteq Z$, a test input $x_*\in X$, and a significance level $\varepsilon\in[0,1]$,CP computes a prediction region Γ_*^ε for x_* as follows.

1. Divide Z' into a training set Z_t, and calibration set Z_c. Let $q=|Z_c|$ be the size of the calibration set.
2. Train a model h using Z_t.
3. Define a nonconformity function $f((x_i,y_i))=\Delta(h(x_i),y_i)$, i.e., choose a metric Δ to measure the distance between $h(x_i)$ and y_i (see Section 3.2).
4. Apply $f(z)$ to each example z in Z_c and sort the resulting nonconformity scores $\{\alpha=f(z)\,|\,z\in Z_c\}$ in descending order: $\alpha_1\geq\cdots\geq\alpha_q$.
5. Compute the nonconformity scores $\alpha_*^j=f((x_*,y^j))$ for the test input x_* and each possible label $j\in\{1,...,c\}$. Then, compute the smoothed p-value

$$p_*^j=\frac{|\{z_i\in Z_c:\alpha_i>\alpha_*^j\}|}{q+1}+\theta\frac{|\{z_i\in Z_c:\alpha_i=\alpha_*^j\}|+1}{q+1},\tag{1}$$

   where $\theta\in\mathcal{U}[0,1]$ is a tie-breaking random variable. Note that p_*^j represents the portion of calibration examples that are at least as nonconforming as the tentatively labelled test example (x_*,y^j).
6. Return the prediction region

$$\Gamma_*^\varepsilon=\{y^j\in Y:p_*^j>\varepsilon\}.\tag{2}$$

   together with a vector of p-values, one for each class.

[7] We will interchangeably use the term "predictor" for the function returning a vector of class likelihoods, and for the function returning the class with highest likelihood.

Note that in this approach, called inductive CP [19], steps 1–4 are performed only once, while Steps 5–6 are performed for every test point x_*.

The rationale is to use a statistical test, more precisely the Neyman-Pearson theory for hypothesis testing and confidence intervals [15], to check if (x_*,y^j) is particularly nonconforming compared to the calibration examples. The unknown distribution of $f(z)$, referred to as $\mathcal{Q}$, is estimated applying f to all calibration examples. Then the scores α_*^j are computed for every possible label y^j in order to test for the null hypothesis $\alpha_*^j \sim \mathcal{Q}$. The null hypothesis is rejected if the p-value associated to α_*^j is smaller than the significance level ε. If a label is rejected, meaning if it appears unlikely that $f((x_*,y^j)) \sim \mathcal{Q}$, we do not include this label in Γ_*^ε. Therefore, given ε, the prediction region contains only those labels for which we could not reject the null hypothesis.

3.2 Nonconformity function

A nonconformity function is well-defined if it assigns low scores to correct predictions and high scores to wrong predictions. A natural choice for f, based on the underlying model h, is $f(z) = \Delta(h(x_i), y_i)$, where Δ is a suitable distance[8]. Recall that, for an input $x \in X$, the output of h is a vector of class likelihoods, which we denote by $h(x) = [P_h(y_1|x),...,P_h(y_c|x)]$. In classification problems, a common well-defined nonconformity function is obtained by defining Δ as

$$\Delta(h(x_i),y_i) = 1 - P_h(y_i|x_i), \tag{3}$$

where $P_h(y_i|x_i)$ is the likelihood of class y_i when the model h is applied on x_i. If h correctly predicts y_i for input x_i, the corresponding likelihood $P_h(y_i|x_i)$ is high (the highest among all classes) and the resulting nonconformity score is low. The opposite holds when h does not predict y_i. The nonconformity measure chosen for our experiments, Equation 3, preserves the ordering of the class likelihoods predicted by h.

3.3 Confidence and credibility

Observe that, for significance levels $\varepsilon_1 \geq \varepsilon_2$, the corresponding prediction regions are such that $\Gamma^{\varepsilon_1} \subseteq \Gamma^{\varepsilon_2}$. It follows that, given an input x_*, if ε is lower than all its p-values, i.e. $\varepsilon < \min_{j=1,...,c} p_*^j$, then the region Γ_*^ε contains all the labels. As ε increases, fewer and fewer classes will have a p-value higher than ε. That is, the region shrinks as ε increases. In particular, Γ_*^ε is empty when $\varepsilon \geq \max_{j=1,...,c} p_*^j$.

The *confidence* of a point $x_* \in X$, $1-\gamma_*$, measures how likely is our prediction for x_* compared to all other possible classifications (according to the calibration set). It is computed as one minus the smallest value of ε for which the conformal region is a single label, i.e. the second largest p-value γ_*:

$$1 - \gamma_* = \sup\{1-\varepsilon : |\Gamma_*^\varepsilon| = 1\}.$$

The *credibility*, c_*, indicates how suitable the training data are to classify that specific example. In practice, it is the smallest ε for which the prediction region is empty, i.e.

[8] The choice of Δ is not very important, as long as it is symmetric.

the highest p-value according to the calibration set, which corresponds to the p-value of the predicted class:

$$c_* = \inf\{\varepsilon : |\Gamma^\varepsilon_*| = 0\}.$$

Note that if $\gamma_* \leq \varepsilon$, then the corresponding prediction region Γ^ε_* contains at most one class. If both $\gamma_* \leq \varepsilon$ and $c_* > \varepsilon$ hold, then the prediction region contains *exactly* one class, i.e., the one predicted by h. In other words, the interval $[\gamma_*, c_*)$ contains all the ε values for which we are sure that $\Gamma^\varepsilon_* = \{\hat{y}_*\}$. It follows that the higher $1-\gamma_*$ and c_* are, the more reliable the prediction $\hat{y}_*$ is, because we have an expanded range $[\gamma_*, c_*)$ of significance values by which $\hat{y}_*$ is valid. Indeed, in the extreme scenario where $c_* = 1$ and $\gamma_* = 0$, then $\Gamma^\varepsilon_* = \{\hat{y}_*\}$ for any value of ϵ. This is why, as we will explain in the next section, our uncertainty-based rejection criterion relies on excluding points with low values of $1-\gamma_*$ and c_*. We stress, in particular, the following statistical guarantee: the probability that the true prediction for x_* is exactly $\hat{y}_*$ is at most $1-\gamma_*$.

In binary classification problems, each point x_* has only two p-values, one for each class, which coincide with c_* (p-value of the predicted class) and γ_* (p-value of the other class).

4 Uncertainty-based Rejection Criteria

Confidence and credibility measure how much a prediction can be trusted. Our goal is to leverage these two measures of uncertainty to identify a criterion to detect errors of the reachability predictor. The criterion is also required to distinguish between false-negative and false-positives errors.

The rationale is that every new input x is required to have values of confidence, $1-\gamma$, and credibility, c, sufficiently high in order for the classification to be accepted. However, determining optimal thresholds is a non-trivial task.

In order to automatically identify optimal thresholds, we proceed with an additional supervised learning approach. For this purpose, we introduce a *cross-validation strategy* to compute values of confidence and credibility, using Z_c as validation set. The cross-validation strategy consists of removing the j-th score, α_j, in order to compute γ_j and c_j, i.e. the p-values at $x_j \in X_c$, where $X_c = \{x \,|\, (x,y) \in Z_c\}$. In this way, we can compute confidence, $1-\gamma$, and credibility, c, for every point in the calibration set.

We now state our supervised learning approach to derive the optimal rejection thresholds. Starting from the calibration set, we construct two training datasets, $\mathcal{D}^{fn}_c$ and $\mathcal{D}^{fp}_c$, which will be used to learn thresholds specific to FN and FP errors, respectively. The inputs of dataset $\mathcal{D}^{fn}_c$ ($\mathcal{D}^{fp}_c$) are the confidence and credibility values of the calibration points, and these inputs are labelled with 1 or 0 depending on whether the classifier h makes a FN (FP) error on the corresponding calibration point. Formally,

$$\mathcal{D}^{fn}_c = \{((\gamma_j, c_j), l_j) \,|\, x_j \in X_c, l_j = I(\hat{y}_j = 0 \wedge y_j = 1)\}$$
$$\mathcal{D}^{fp}_c = \{((\gamma_j, c_j), l_j) \,|\, x_j \in X_c, l_j = I(\hat{y}_j = 1 \wedge y_j = 0)\},$$

where $I(pred)$ equals 1 if predicate *pred* is true and equals 0 otherwise.

For simplicity, let us now focus on one of the two cases, $\mathcal{D}^{fn}_c$. Analogous reasoning applies to $\mathcal{D}^{fp}_c$. We seek to find confidence and credibility values that optimally separate the points in $\mathcal{D}^{fn}_c$ in relation to their classes, that is, separate points yielding FN errors

from those that do not. We solve this problem by learning two linear Support Vector Classifiers (l-SVCs), trained on pairs $(1-\gamma,l)$ and (c,l), respectively. In this way, we identify individual confidence and credibility thresholds, denoted by $1-\gamma_\tau^{fn}$ and c_τ^{fn}, respectively[9]. Given a test point x_* with predicted label $\hat{y}_*$, confidence $1-\gamma_*$ and credibility c^*, the learned thresholds establish two rejection criteria: one for confidence, $R_\gamma^{fn}(x_*)=(1-\gamma_*<1-\gamma_\tau^{fn})$, and one for credibility, $R_c^{fn}(x_*)=(c_*<c_\tau^{fn})$.

Proposition 1. *Both R_γ^{fn} (R_γ^{fp}) and R_c^{fn} (R_c^{fp}) are the best approximate solutions of Problem 2, i.e., they are such that the probability of wrongly rejecting or accepting a FN (FP) prediction is minimal.*

Proof (Sketch). A SVC finds the maximum-margin hyper-plane that separates the classes, i.e. it maximizes the distance between the hyper-plane and the nearest point from either group. In general the larger the margin, the lower the generalization error. If the classes overlap the exact separation of the training data can lead to poor generalization. The SVC allows some of the training points to be misclassified, with a penalty that increases linearly with the distance from that boundary. The optimization goal is to maximize the margin, while penalizing points that lie on the wrong side of the hyper-plane (see Chapter 7 of [7] for a complete treatment). Therefore, the learned hyperplane optimally separates erroneous from non-erroneous predictions, that is, the probability, over the calibration set, that a prediction is wrongly rejected or wrongly accepted is minimal and so is the generalization error. Since we are cross-validating, i.e. we are approximating a sample from the data distribution, then the criterion is optimal for any input test point.

The final rejection criterion is a conservative combination of the four rejection criteria. A test point x_* is rejected if:

$$R(x_*)=\big(1-\gamma_*<\max(1-\gamma_\tau^{fn},1-\gamma_\tau^{fp})\big)\vee\big(c_*<\max(c_\tau^{fn},c_\tau^{fp})\big). \tag{4}$$

Alternatively, one can implement rejection criteria specific to FN (FP) errors by using only the thresholds $1-\gamma_\tau^{fn}$ and c_τ^{fn} ($1-\gamma_\tau^{fp}$ and c_τ^{fp}).

Tuning of SVC hyperparameters. In NSC, we deal with high-accuracy state classifiers. This implies that the datasets $\mathcal{D}_c^e$, with $e\in\{fp,fn\}$, are highly unbalanced, as they contain more examples of correct classifications (label 0) than of classification errors (label 1). In binary classification problems, such as our l-SVCs, accuracy can be misleading with imbalanced datasets, as any model that "blindly" assigns the label of the most frequent class to any input will have high accuracy.

A simple method to handle imbalanced classes in SVC is to design an empirical penalty matrix $\mathcal{P}_e$, which assigns different error penalties by class [6]. In particular, the (i,j)-th entry of $\mathcal{P}_e$ gives the penalty for classifying an instance of class i as class j. Of course, when $i=j$, the penalty is null. The penalty matrix for dataset $\mathcal{D}_c^e$ is defined as

$$\mathcal{P}_e=\begin{bmatrix} 0 & \frac{q}{2r_e(q-n_e)} \\ \frac{r_e q}{2n_e} & 0 \end{bmatrix}, \tag{5}$$

[9] As opposed to learning a linear combination of confidence and credibility, which is less interpretable.

where n_e is the number of points belonging to class 1 in dataset $\mathcal{D}_c^e$, and r_e is a parameter influencing how many errors of type e we are willing to accept. The term $\frac{r_e q}{2n_e}$, which represents the penalty for wrongly classifying an error of type e as correct, increases as n_e decreases. Note that, when $r_e = 1$ and the dataset is perfectly balanced ($q = 2n_e$), the penalties are equals: $\frac{r_e q}{2n_e} = \frac{q}{2r_e(q-n_e)} = 1$. Further, if $r_e > 1$, the penalty term increases, leading to more strict rejection thresholds and higher overall rejection rates. On the contrary, if $r_e < 1$, the penalty decreases, leading to possibly miss some errors of type e.

5 Active Learning

Recall that we are dealing with two combined learning problems: learning a prediction rule (i.e., a state classifier) using the training set Z_t, and learning a rejection rule using the calibration sets D_c^{fn} and D_c^{fp}. As the accuracy of a classifier increases with the quality and the quantity of observed data, adding samples to Z_t will generate a more accurate predictor, and similarly, adding samples to D_c^{fn} and D_c^{fp} will lead to more precise rejection thresholds. Ideally, one wants to maximize accuracy while using the least possible amount of additional samples, because obtaining labeled data is expensive (in NSC, labelling each sample entails solving a reachability checking problem), and the size of the datasets affect the complexity and the dimension of the problem. Therefore, to improve the accuracy of our learning models efficiently, we need a strategy to identify the most "informative" additional samples.

Our solution is *uncertainty sampling-based active learning*, where the re-training points are derived by first sampling a large pool of unlabeled data, and then considering only those points where the current predictor h is still uncertain. We develop an efficient query strategy that leverages the CP-based measures of uncertainty, and in particular, the rejection rule of Section 4, since rejected points are indeed the most uncertain ones. The proposed active learning method should reduce both the overall number of false-positive and false-negative predictions and the overall rejection rate.

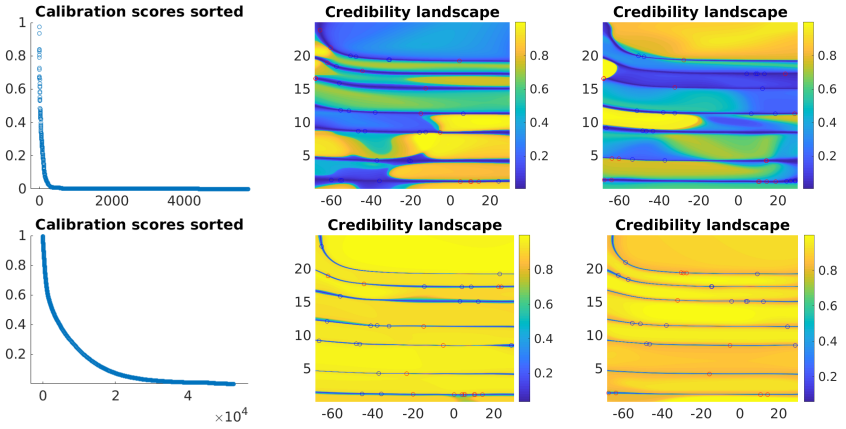

Fig. 2. Spiking Neuron: calibration scores (first column) and credibility landscapes using the initial calibration set Z_c (top line) versus the query set Z_Q (bottom line). The landscapes are obtained for different instances of the predictor h, trained on the same dataset Z_t.

5.1 Refining the query strategy

Sensitivity of the uncertainty measures. The distribution of calibration scores depends both on the case study at hand and on the trained classifier. If such a classifier h has high accuracy, then most of the calibration scores α_1,...,α_q will be close to zero. Each p-value p_*^j of an unseen test point x_* counts the number of calibration scores greater than α_*^j, the non-conformity score for label j at x_*. Credibility, which is the p-value associated with the class predicted by h, is expected to have a small score and therefore a high p-value. On the contrary, γ, which is the p-value associated to the other (non-predicted) class, is expected to have a larger score. However, given the high accuracy of h, the number of calibration scores significantly greater than zero is very small. Therefore, the fraction of calibration scores determining γ is not very sensitive to changes in the value of α_*, which is determined by $h(x_*)$. On the contrary, credibility is extremely sensitive to small changes in α_*. In general, the sensitivity of confidence with respect to α_* increases as the accuracy of h decreases, and vice versa for credibility. Figure 2 shows the credibility landscapes for two different training instances of model h on the same training set for a concrete case study. We observe that even if regions where misclassifications take place are always assigned low credibility values, outside those regions credibility values are subject to high variance.

This sensitivity results in a over-conservative rejection criterion, leading to a high rejection rate and in turn, to an inefficient query strategy. However, if we enrich the calibration set using additional samples with non-zero α-scores, we can reduce such sensitivity, thereby making credibility more robust with respect to retraining. This process is illustrated in Figure 2, where the additional non-zero α-scores (bottom) lead to a more robust credibility landscape, where low-credibility regions are now more tightly centred around areas of misclassification.

Since samples with uncertain predictions will have non-zero α-scores[10], we will use the original rejection rule to enrich the calibration set, thereby deriving a refined rejection rule and in turn, a refined and more effective query strategy for active learning. Notice that, once the model h has been retrained we must accordingly retrain the rejection rule as well, since values of confidence and credibility depend on the predictions of h.

5.2 Active learning algorithm

The active learning is divided in two phases. In the first phase, we refine the query strategy: we use the current rejection rule to select a batch of uncertain points, temporarily add these points to the calibration set, and obtain an updated rejection rule, which represents our query strategy.

In the second phase, using the refined query strategy, we sample another batch of points, divide it in two groups, and use them to augment training and calibration sets, respectively. The resulting predictor h_a, trained on the augmented set, is expected to be more accurate then h. Further, h_a is used to update the α-scores and the values of confidence and credibility for the augmented calibration set. This results in an updated rejection rule, for which a lower rejection rate is expected.

[10] Note indeed that the α-score of a sample (x_i,y_i) is zero only if h both correctly predicts y_i and the corresponding class likelihood $P_h(y_i \mid x_i)$ is 1.

We now describe in details our uncertainty sampling-based active learning algorithm, which given an initial training set Z_t, a prediction rule h trained on Z_t, an initial calibration set Z_c, a rejection rule R trained on Z_c using some rejection ratios r_{fn} and r_{fp}, computes an enhanced predictor h_a and enhanced rejection rule R_a as follows.

1. **Refining the query strategy:**
   - Randomly select a large number of input points, compute their confidence and credibility using h, and identify the subset Q of points rejected based on R.
   - Invoke the reachability oracle to label the points in Q and define a query set Z_Q by adding these points to Z_c.
   - Obtain an updated rejection rule R_Q from Z_Q using the method of Section 4 with rejection ratios r_{fn} and r_{fp}.
2. **Active phase:**
   - Randomly select a large number of input points, compute their confidence and credibility using h, and identify the subset A of points rejected based on R_Q.
   - Invoke the reachability oracle to label the points in A, divide the data into two groups and add them respectively to Z_t and Z_c, obtaining an augmented training set, Z_t^a, and an augmented calibration set, Z_c^a.
   - Train a new predictor h_a from Z_t^a.
   - Train new detection thresholds using the method of Section 4, with rejection ratios r_{fn} and r_{fp}, and obtain the enhanced rejection rule R_a.

Note that the above algorithm can be iterated, using Z_t^a, Z_c^a, h_a, and R_a as new inputs.

It is important to observe that, in order for the active learning algorithm to preserve the statistical soundness of conformal prediction, the augmented training and calibration sets Z_t^a and Z_c^a must be sampled from the same distribution. This is guaranteed by the fact that, in the active learning phase, we add new points to both the training and the calibration dataset, and these points are sampled from the same distribution (in particular, we apply the same random sampling method and same rejection criterion). The only caveat is ensuring that the ratio between the number of samples in Z_c and Z_t is preserved on the augmented datasets.

6 Experimental Results

In order to experimentally evaluate the proposed method, both the initial approach and the active learning approach have been applied to hybrid systems with varying degrees of complexity. We consider three deterministic case studies: the spiking neuron [21], which is a two-dimensional model with non-linear dynamics, the artificial pancreas (AP) [18], which is a nine-dimensional non-linear model, and the helicopter [21], a linear model with 29 variables. In addition, we analyze two non-deterministic models with non-linear dynamics: a cruise controller [21], whose input space has four dimensions, and a triple water tank (WT) [1], which is a three-dimensional model. For the AP model, the unsafe set U corresponds to hypoglycemia states, i.e., $U = BG \leq 3.9$ mmol/L, where BG is the blood glucose variable. The state distribution considers uniformly distributed values of plasma glucose and insulin. The insulin control input is fixed to the basal value. The time bound is $T = 240$. For the WT model, U is given by

states where the water level of any of the tanks falls outside a given safe interval I, i.e., $U = \vee_{i=1}^{3} x_i \notin I$, where x_i is the water level of tank i. The state distribution considers water levels uniformly distributed within the safe interval. The time bound is $T = 1$. Details on the other case studies are available in Appendix D of [20].

Experimental settings. The entire pipeline is implemented in MATLAB. Motivated by the results presented in [21], we define the state classifier as a sigmoid DNN. Each case study shares the same DNN architecture: 3 hidden layers, each consisting of 10 neurons with the Tan-Sigmoid activation function and an output layer with 1 neuron with the Log-Sigmoid activation function. In particular, the output of the DNN, which is our model h, is the likelihood of class 1, i.e., the likelihood that the hybrid automaton state is positive. Training is performed using MATLAB's `train` function, with the Levenberg-Marquardt backpropagation algorithm optimizing the mean square error loss function, and the Nguyen-Widrow initialization method for the NN layers. Training the DNNs takes from 2 to 39 seconds. For every model we generate an initial dataset $Z^{'}$ of 20,000 samples and a test set Z_{test} of 10,000 samples. The helicopter model is the only exception, where, due to the higher dimensionality, a dataset of 100,000 samples is used. The training and calibration sets are two subsets of $Z^{'}$ extracted as follows: a sample $z \in Z^{'}$ has probability 0.7 of falling into Z_t and probability 0.3 of falling into Z_c. We used the dReal solver [13] as reachability oracle to label the datasets for the non-deterministic case studies. For deterministic ones, we used an HA simulator implemented in MATLAB.

Computational performance. We want our method to be capable of working at runtime, which means it must be extremely fast in making predictions and deciding whether to trust them. We must clarify that the time required to train the method does not affect its runtime efficiency, as it is performed in advance only once. Learning the rejection criteria, which is also performed offline, requires the following steps: (i) train the state classifier, (ii) generate the datasets $\mathcal{D}_c^{fp/fn}$, which requires computing the p-values for each point in Z_c, and (iii) train four l-SVCs. Executing the entire pipeline takes around 10 seconds, if $|Z^{'}| = 20K$, and around 80 seconds if $|Z^{'}| = 100K$. Nonetheless, given a new input x_*, it takes from 0.3 up to 2 ms to evaluate the rejection criterion. This evaluation time does not depend on the dimension of the hybrid system, but it is affected by the size of the calibration set Z_c. Refining the uncertainty measures leads to an increase in the size of Z_c. Hence the aim of active learning is to improve the performance while keeping the technique as efficient as possible. Instead of adding random samples to Z_c, our active learning approach adds only samples that are extremely informative and brings a consistent improvement in the precision of the uncertainty measures. It carries two additional training costs: the time needed to compute confidence and credibility for a large pool of data, and the time the oracle needs to compute labels for the uncertain points. The latter dominates, especially for non-deterministic systems, since their oracles are more expensive. Therefore, if the rejection rate is relatively high and we consider a large pool of points, which allows for a good exploration, the procedure may be long. However, this time spent to optimally tune the performance improves the run-time behaviour of our method. This is another good reason to improve the query strategy before proceeding with the active learning approach. The time required to refine the query strategy depends on the size of the pool of data, the higher the initial rejection rate, the higher the number of queries. However,

the pool has to be large in order to find significant instances. Adding observations about uncertain samples results in a more precise rejection rule with a lower rejection rate. Therefore, points selected with the refined query strategy are fewer and more informative.

Experiments. We compare our uncertainty-based query strategy with a random sampling strategy. Both strategies add the same number of samples to Z_t and the same number of samples to Z_c. However, in the first case, referred to as active approach, these samples are selected according to the refined query strategy, whereas in the second case, referred to as passive approach, they are randomly sampled following a uniform distribution, the same distribution used to generate the initial datasets.

The duration of the active learning phase depends on the sizes of the sample pools. In our study, the pool used to refine the query strategy contains 100,000 samples (250,000 for the helicopter), whereas the pool used for the active learning phase contains 200,000 samples (500,000 for the helicopter). In particular, one iteration of the active learning procedure took around 10 minutes for the spiking neuron and the artificial pancreas, the simplest deterministic models, and around 70 minutes for the helicopter, due to the larger pools. For the non-deterministic models (triple water tank and cruise controller), it took around 2.25 hours. This time is expected to decrease for subsequent iterations, as the rejection rate will be lower (leading to fewer retraining samples). Note that retraining is performed offline and does not affect runtime performance of our approach.

Table 1 and Table 2 present the experimental performance of the rejection criterion obtained using the original method in Section 4 and the refined rejection criteria obtained using the active and passive approaches. All results are averaged over 5 runs; in each run, we resample Z_t and Z_c from $Z^{'}$ and retrain the DNN. Table 1 shows the performance obtained using the initial rejection rule on the test set Z_{test}. The accuracy of the NSC, averaged over the five case studies, is 99.5832%. The rejection criterion recognizes well almost all the errors (with average accuracy over the accepted predictions of 99.9956%), but the overall rejection rate is around 5%, a non-negligible amount. We see from Table 2 that the passive learning approach provides little improvement: the overall number of errors is similar to the initial one and the rejection rate is still relatively large. Table 2 also shows that the active approach provides much more significant improvements: the overall rejection rate and the number of errors made by the NSC fall dramatically, while preserving the ability of recognizing almost all of the errors, both false positives and false negatives, made by the predictor (average accuracy over the accepted predictions of 99.9992%). The overall rejection rates span between 3.46% and 9.88% when the initial rejection rule is applied. In contrast, the active learning approach achieves rejection rates between 0.51% and 2.74%. The overall number of errors reduces as well: the range of false-negative errors reduces from $7-33.2$ to $1.8-11.6$, while the range of false-positive errors reduces from $8.2-48.8$ to $2-11.2$.

In our analysis, parameters r_{fp} and r_{fn} are set to one, i.e., they do not influence the selection of rejection thresholds. If false negatives have severe consequences, one can design a stricter policy by assigning r_{fn} a value greater than one. On the contrary, we can relax the policy on false positives, assigning to r_{fp} a value smaller than one, and thus reducing the overall rejection rate. In general, it may be wise to first improve the performance of the predictor in recognizing both types of errors via active learning, and then decide to reduce the overall rejection rate by allowing some false positives.

	INITIAL			
Model	accuracy	**fp**	**fn**	rej. rate
Spiking Neuron (SN)	99.582%	24.4/24.6	17.2/17.2	5.68%
Artificial Pancreas (AP)	99.488%	30.4/30.6	20.6/20.6	6.23%
Helicopter (HE)	99.180%	47.4/48.8	33/33.2	9.88%
Water Tank (WT)	99.818%	8.6/8.6	9.6/9.6	5.97%
Cruise Controller (CC)	99.848%	8.2/8.2	7/7	3.46%

Table 1. Performance of the initial rejection criterion on the test set. Results are averaged over 5 runs. Accuracy is the percentage of points in the test set that are correctly predicted. The **fp** and **fn** columns show the ratio of false positives and false negatives, respectively, recognized by each criteria. The last column shows the percentage of point rejected over the entire test set.

		PASSIVE			ACTIVE			
Model	# samples	**fp**	**fn**	rej. rate	accuracy	**fp**	**fn**	rej. rate
SN	5748.2	18.2/18.2	10.6/10.8	3.91%	99.918%	2.8/2.8	5.4/5.4	1.16%
AP	6081.8	23/23.4	19.4/19.4	5.94%	99.892%	6.2/6.2	4.4/4.6	1.02%
HE	22014.6	31.4/31.6	26/26.6	7.21%	99.772%	11.2/11.2	10.4/11.6	2.74%
WT	4130.2	8.4/8.4	10.2/10.4	4.43%	99.962%	2.8/2.8	1/1	0.70%
CC	2280.6	6/6	6/6	5.15%	99.962%	2/2	1.8/1.8	0.51%

Table 2. Performance of rejection criteria obtained by refining the initial rejection criterion using the passive and active approaches. Results are averaged over 5 runs. Most of the columns have the same meaning as in Table 1. "# samples" is the number of samples added globally to Z_t and Z_c.

7 Related Work

A number of methods have been proposed for online reachability analysis that rely on separating the reachability computation into distinct offline and online phases. However, these methods are limited to restricted classes of models [10], or require handcrafted optimization of the HA's derivatives [4], or are efficient only for low-dimensional systems and simple dynamics [25].

In contrast, NSC [21] is based on learning DNN-based classifiers, is fully automated and has negligible computational cost at runtime. In [11, 24], similar techniques are introduced for neural approximation of Hamilton-Jacobi (HJ) reachability. Our methods for prediction rejection and active learning are independent of the class of systems and the machine-learning approximation of reachability, and thus can also be applied to neural approximations of HJ reachability.

The work of [3] addresses the predictive monitoring problem for stochastic black-box systems, where a Markov model is inferred offline from observed traces and used to construct a predictive runtime monitor for probabilistic reachability checking. In contrast to NSC, this method focuses on discrete-space models, which allows the predictor to be represented as a look-up table (as opposed to a neural network).

In [22], a method is presented for predictive monitoring of STL specifications with probabilistic guarantees. These guarantees derive from computing prediction intervals of ARMA/ARIMA models learned from observed traces. Similarly, we use CP which

also can derive prediction intervals with probabilistic guarantees, with the difference that CP supports any class of prediction models (including auto-regressive ones).

A related approach to NSC is smoothed model checking [9], where Gaussian processes [23] are used to approximate the satisfaction function of stochastic models, i.e., mapping model parameters into the satisfaction probability of a specification. Smoothed model checking leverages Bayesian statistics to quantify prediction uncertainty, but faces scalability issues as the dimension of the system increases. In contrast, computing our measure of prediction reliability is very efficient, because it is nearly equivalent to executing the underlying predictor.[11] In Bayesian approaches to uncertainty estimation, one often does not know the true prior distribution, which is thus often chosen arbitrarily. However, if the prior is incorrect, the resulting uncertainty measures have no theoretical base. The CP framework that we use is instead distribution-free and provides uncertainty information based only on the standard i.i.d. or exchangeability assumption. Avoiding Bayesian assumptions makes CP conclusions more robust to different underlying data distributions, which is also shown experimentally in [17].

A basic application of conformal predictors in active learning is presented in [16]. Our approach introduces three important improvements: a more flexible and meaningful combination of confidence and credibility values, automated learning of rejection thresholds (which are instead fixed in [16]), and refinement of the query strategy.

In [8], we presented a preliminary version of this approach. The present paper greatly extends and improves that work by including an automated and optimal method to select rejection thresholds, the active learning method, and an evaluation on larger HA benchmarks.

8 Conclusion

We have presented Neural Predictive Monitoring, a technique for providing statistical guarantees on the prediction reliability of neural network-based state classifiers used for runtime reachability prediction. To this purpose, we have introduced statistically rigorous measures that quantify the prediction uncertainty of a state classifier. We have employed these uncertainty measures to derive conservative rejection criteria that identify, with minimal error, those predictions that can lead to safety-critical state classification errors. We have further designed an active learning strategy that, leveraging such uncertainty-based rejection criteria, allow to increase the accuracy of the reachability predictor and reduce the overall rejection rate.

The strengths of our NPM technique are its effectiveness in identifying and rejecting prediction errors and its computational efficiency, which is not directly affected by the complexity of the system under analysis (but only by the complexity of the underlying learning problem and classifier). As future work, we plan to extend our approach to predict quantitative measures of property satisfaction (like the robust STL semantics [12]), which will require us to develop a regression framework for NPM.

[11] Evaluating our rejection criterion reduces to computing two p-values (confidence and credibility). Each p-value is derived by computing a nonconformity score, which requires one execution of the underlying predictor h, and one search over the array of calibration scores.

References

1. dReal – Networked Water Tank Controllers (2017), http://dreal.github.io/benchmarks/networks/water/
2. Alur, R.: Formal verification of hybrid systems. In: Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT). pp. 273–278 (Oct 2011)
3. Babaee, R., Gurfinkel, A., Fischmeister, S.: Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In: International Conference on Runtime Verification. pp. 187–204. Springer (2018)
4. Bak, S., Johnson, T.T., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. In: Real-Time Systems Symposium (RTSS), 2014 IEEE. pp. 138–148. IEEE (2014)
5. Balasubramanian, V., Ho, S.S., Vovk, V.: Conformal prediction for reliable machine learning: theory, adaptations and applications. Newnes (2014)
6. Batuwita, R., Palade, V.: Class imbalance learning methods for support vector machines (2013)
7. Bishop, C.M.: Pattern recognition and machine learning. springer (2006)
8. Bortolussi, L., Cairoli, F., Paoletti, N., Stoller, S.D.: Conformal predictions for hybrid system state classification. In: From Reactive Systems to Cyber-Physical Systems, to appear (2019)
9. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time Markov chains. Information and Computation **247**, 235–253 (2016)
10. Chen, X., Sankaranarayanan, S.: Model predictive real-time monitoring of linear systems. In: Real-Time Systems Symposium (RTSS), 2017 IEEE. pp. 297–306. IEEE (2017)
11. Djeridane, B., Lygeros, J.: Neural approximation of PDE solutions: An application to reachability computations. In: Proceedings of the 45th IEEE Conference on Decision and Control. pp. 3034–3039. IEEE (2006)
12. Donzé, A., Maler, O.: 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)
13. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: International Conference on Automated Deduction. pp. 208–214. Springer (2013)
14. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of computer and system sciences **57**(1), 94–124 (1998)
15. Lehmann, E.L., Romano, J.P.: Testing statistical hypotheses. Springer Science & Business Media (2006)
16. Makili, L.E., Sánchez, J.A.V., Dormido-Canto, S.: Active learning using conformal predictors: application to image classification. Fusion Science and Technology **62**(2), 347–355 (2012)
17. Melluish, T., Saunders, C., Nouretdinov, I., Vovk, V.: The typicalness framework: a comparison with the bayesian approach. University of London, Royal Holloway (2001)
18. Paoletti, N., Liu, K.S., Smolka, S.A., Lin, S.: Data-driven robust control for type 1 diabetes under meal and exercise uncertainties. In: International Conference on Computational Methods in Systems Biology. pp. 214–232. Springer (2017)
19. Papadopoulos, H.: Inductive conformal prediction: Theory and application to neural networks. In: Tools in artificial intelligence. InTech (2008)
20. Phan, D., Paoletti, N., Zhang, T., Grosu, R., Smolka, S.A., Stoller, S.D.: Neural state classification for hybrid systems. ArXiv e-prints (Jul 2018)
21. Phan, D., Paoletti, N., Zhang, T., Grosu, R., Smolka, S.A., Stoller, S.D.: Neural state classification for hybrid systems. In: Automated Technology for Verification and Analysis. Lecture Notes in Computer Science, vol. 11138, pp. 422–440 (2018). https://doi.org/10.1007/978-3-030-01090-4_25

22. Qin, X., Deshmukh, J.V.: Predictive monitoring for signal temporal logic with probabilistic guarantees. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 266–267. ACM (2019)
23. Rasmussen, C.E., Williams, C.K.: Gaussian processes for machine learning, vol. 1. MIT press Cambridge (2006)
24. Royo, V.R., Fridovich-Keil, D., Herbert, S., Tomlin, C.J.: Classification-based approximate reachability with guarantees applied to safe trajectory tracking. arXiv preprint arXiv:1803.03237 (2018)
25. Sauter, G., Dierks, H., Fränzle, M., Hansen, M.R.: Lightweight hybrid model checking facilitating online prediction of temporal properties. In: Proceedings of the 21st Nordic Workshop on Programming Theory. pp. 20–22 (2009)
26. Sha, L.: Using simplicity to control complexity. IEEE Software (4), 20–28 (2001)
27. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic learning in a random world. Springer Science & Business Media (2005)