Highlights

Optimal Input Design for Guaranteed Fault Diagnosis of Nonlinear Systems: An Active Deep Learning Approach

Nathaniel Massa,Joel A. Paulson

- A novel optimal active fault diagnosis strategy for general nonlinear systems.
- Develop active learning method for efficiently finding minimally invasive separating inputs.
- Leverages state-of-the-art reachability tools to ensure guaranteed fault diagnosis.

Optimal Input Design for Guaranteed Fault Diagnosis of Nonlinear Systems: An Active Deep Learning Approach*

Nathaniel Massa[a], Joel A. Paulson[a,*]

[a]Department of Chemical and Biomolecular Engineering, The Ohio State University, Columbus, 43210, Ohio, United States

ARTICLE INFO

Keywords:
Active learning
Deep neural networks
Guaranteed fault diagnosis
Reachability analysis

ABSTRACT

The trend toward increasing complexity in many industries has made component malfunctions and other abnormal events increasingly frequent. These events, often referred to as faults, must be quickly and accurately diagnosed in order to ensure safe and reliable system operation. Active fault diagnosis (AFD) refers to methods that design particular input signals to be injected into a system that improve detectability of faults. In this work, we present a novel optimal AFD strategy focused on the design of minimally invasive input signals that guarantee safety (i.e., state constraints are not violated) while also providing a complete fault diagnosis (i.e., measurements are consistent with at most one model) for general nonlinear systems under uncertainty. Our approach is inspired from taking a data-driven perspective to this problem wherein we aim to learn its solution by querying an oracle that certifies if a given input sequence satisfies separability and safety constraints or not. Since the oracle is expensive to query in many cases, we develop an efficient active learning method that uses deep neural network models to sequentially identify a batch of informative input sequences to query at every iteration. We discuss strategies for practically evaluating upper and lower bounds on the oracle using over- and under-approximations of reachable state and output sets for the dynamic system. The effectiveness and generality of our proposed approach is demonstrated through multiple case studies including linear and nonlinear systems.

1. Introduction

In many critical industries, such as aerospace [15], chemical manufacturing [36, 34], and healthcare [37], "faults" (e.g., component malfunctions or other abnormal changes in system behavior) can lead to hazardous situations, potentially endangering lives, causing environmental harm, and reducing operational efficiency. Early detection and diagnosis of faults is important for achieving safe and reliable system operation, especially as systems are becoming increasingly complex. However, the fault detection and diagnosis (FDD) task is highly non-trivial to solve in practice due to several confounding factors including disturbances, measurement errors, and model structure and parameter uncertainty.

There has been a significant amount of work on FDD, with the vast majority of methods being categorized as either *passive* or *active*. Passive FDD methods focus on comparing available input-output system data to models and/or historical data [13, 52, 51]. A wide variety of techniques have been investigated for passive FDD including principal component analysis [31, 14], partial least squares [30], canonical variate analysis [21, 20], residual [27], observer [39, 10], set-based [23, 28], machine learning [29, 50, 3], and manifold learning [22, 12] methods; interested readers are referred to the following review paper [51] for a more comprehensive overview. The main challenge with passive FDD is that many faults may not be diagnosable without explicitly exciting the system [35, 9, 5, 18]. In other words, the collected system measurements may be consistent with many different fault scenarios, which is a problem often further exacerbated by the compensatory actions made by the control system in a passive setting. Active FDD methods, on the other hand, aim to inject specially designed input signals to improve diagnosability of faults. Both passive and active FDD methods can be further categorized as model-based [52] or model-free [13]; we focus on model-based approaches in this paper. Note that by "model-based", we mean these methods assume the existence of models for the nominal and faulty versions of the plant.

An optimization-based formulation of the (model-based) active FDD problem was first proposed in [35], which involved two linear time-invariant models subject to bounded disturbances and measurement errors: one representing the nominal (fault-free) system, and the other representing a system with a fault. For models defined by a set of dynamic linear equations, this work showed that the set of *separating inputs* (i.e., input signals that generate output signals that are consistent with at most one model, thus providing a complete fault diagnosis) is the complement of a projection of a high-dimensional polytope. Unfortunately, polytope projection operations can be computationally intensive and even numerically unstable in large dimensions. By minimizing the input cost/energy subject to being within the set of separating inputs, one can design so-called *optimal separating inputs* [9]; however, this type of method requires the solution to a complex bilevel optimization problem that is highly non-convex, even in the linear two-model case. To overcome numerical challenges with these methods, more recent work has explored the use of zonotopes and mixed-integer programming to improve the scalability of this framework [49, 48, 45]. However, such methods rely

*This work was supported by the National Science Foundation [grant number 2237616].

*Corresponding author
paulson.82@osu.edu (J.A. Paulson)
ORCID(s): 0000-0002-1518-7985 (J.A. Paulson)

on explicit characterization of reachable state and output sets, which is not possible for most system classes [1]. Thus, there has been very limited work on optimal active FDD for general nonlinear systems. A method for design of near-optimal separating inputs for polynomial/rational systems is presented in [43], which uses a convex relaxation strategy to compute outer approximations to the reachable sets. This method still requires one to solve a complex bilevel optimization problem, with computational cost quickly growing as more fault models are considered and the tightness of the relaxation is increased.

In this work, we propose a first-of-its-kind optimal active FDD method for nonlinear systems under bounded uncertainties with the following features: (i) applicable to a large number of fault models; (ii) systematically finds separating inputs with increasingly lower cost/energy (if they exist); and (iii) can easily incorporate worst-case state constraints on all models. The only assumption that we make about the system class is that we can query an oracle for a given input signal that provides a guaranteed yes or no answer to satisfaction of the separability and state constraint conditions. We show how such an oracle can be derived by computing reachable sets for the different models, and how upper and lower bounds on this oracle can be practically computed for a large class of nonlinear systems using existing state-of-the-art reachability algorithms (implemented in, e.g., [2]). Since we cannot compute a closed-form expression for this oracle, we instead propose to learn an approximation of it from data. The ability to efficiently learn optimal separating inputs across a much broader range of systems opens the door for active FDD to tackle previously inaccessible challenges. For instance, as digital twins become increasingly integrated into manufacturing processes, our method could be interfaced with a digital twin, allowing for the simulation of numerous "virtual" fault scenarios. This fast exploratory analysis with advanced plant emulators represents a significant advancement, as such capabilities are currently lacking in the optimal active FDD space. By enabling proactive identification and intervention, our approach could help prevent major system disruptions before they escalate, adding a powerful new tool to the modern predictive maintenance arsenal.

Although the idea of surrogate-assisted optimization is not new, the key observation here is that much of the input space is not informative for improving our knowledge of the optimal separating input. Thus, the learned surrogate model is not good at distinguishing separating and non-separating inputs in high dimensions without a huge amount of data. Since querying the oracle is expensive, we are operating in a relatively low-data regime and thus need a more intelligent sample selection policy. We derive a novel active learning policy for this problem that uses a combination of information theory (Shannon entropy) and optimization to sequentially identify informative batches of samples to query. At every active learning iteration, the current surrogate model (trained on all labeled data) is used to quantify how useful new samples could be if they are added into the training dataset. Lastly, to demonstrate the effectiveness of our proposed method, we apply it on two distinct case studies with different features including a highly nonlinear continuously stirred tank reactor system. We observe that our method is able to consistently find high quality solutions given a limited sample budget, and it also significantly outperforms existing sample selection strategies.

The remainder of this paper is organized as follows. Section 2 introduces the optimal active FDD problem of interest in this work. Section 3 provides a detailed overview of our proposed active learning method and how it can be practically implemented using reachable set approximations. We evaluate the performance of the method on numerical examples in Section 4 and provide some concluding remarks in Section 5.

2. Problem Statement

2.1. System dynamics

In this work, we consider nonlinear discrete-time systems indexed by time k with $k = 0$ as the initial time. In each interval $[k, k+1]$ for $k = 0, 1, 2, \ldots$, the system evolves according to one of n_m possible models, distinguished by an argument $i \in \mathbb{I} = \{1, \ldots, n_m\}$. For the sake of generality, we allow the model to change between intervals denoted by a subscript i_k that indicates the active model over $[k, k+1]$. Following a similar notation to [48], we denote the system evolution as follows

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k, i_k), \tag{1a}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k, i_k). \tag{1b}$$

Here, $\mathbf{x}_k \in \mathbb{R}^{n_x}$ are the states, $\mathbf{y}_k \in \mathbb{R}^{n_y}$ are the measured outputs, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ are the manipulated inputs, $\mathbf{w}_k \in \mathbb{R}^{n_w}$ are the disturbances, and $\mathbf{v}_k \in \mathbb{R}^{n_v}$ are the measurement errors. The model corresponding to $i_k = 1$ represents nominal operation, while all other models $i_k > 1$ denote the "abnormal" or "faulty" models. The choice of the set of considered models $\mathbb{I}$ is up to the user such that it may include, e.g., simultaneous faults if desired.

The functions $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_p} \times \mathbb{I} \to \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \times \mathbb{R}^{n_p} \times \mathbb{I} \to \mathbb{R}^{n_y}$, respectively, define the state transition and measurement process.

Note that we assume that all models in $\mathbb{I}$ share the same "global" state definition. If a model has a "local" state that does not directly appear in the other models, one can always introduce a dummy state that remains constant and does not influence the other states to arrive at a consistent system definition in (1). A similar argument can be made for the other unobserved system variables $\mathbf{w}_k$, $\mathbf{v}_k$, and $\mathbf{p}$. However, it is critical that the inputs $\mathbf{u}_k$ and outputs $\mathbf{y}_k$ share the same meaning since these quantities are physically changed/observed in the system.

2.2. Constraints

Since we are interested in discriminating between normal and faulty models, there must be some system uncertainty for this to be a non-trivial task. We model this

uncertainty as the unobserved variables taking on values in (known) sets, i.e.,

$$\mathbf{x}_0 \in X_0, \quad (\mathbf{w}_k, \mathbf{v}_k) \in W \times V, \quad \forall k \in \mathbb{N}, \tag{2}$$

where $\mathbf{x}_0$ denotes the initial state of the system. Note that we can also incorporate time-invariant parametric uncertainty by extending the definition of the state to include unknown parameters (see, e.g., [41, 42] for details). The state and input must also to satisfy pointwise-in-time constraints, i.e.,

$$(\mathbf{x}_k, \mathbf{u}_k) \in X \times U, \quad \forall k \in \mathbb{N}. \tag{3}$$

The set U represents bounded input constraints that limit the amount the inputs can be changed (e.g., valve opening can only range from 0% to 100%). These constraints can be directly enforced through proper selection of the input sequence. The set X, on the other hand, represents potential safety constraints on the states (e.g., temperature limits) and must be implicitly enforced through proper design of the input sequence. For simplicity, we further assume U and X are polytopes, though more complex (non-convex) set representations can be relatively easily incorporated into our proposed framework at additional computational cost.

2.3. Reachable set notation

We are interested in the reachable sets derived from the system (1) over various time intervals. We use tildes $\tilde{\cdot}$ to denote sequences. We further use subscripts to denote the range of time indices, i.e., $\tilde{\mathbf{a}}_{l:k} = (\mathbf{a}_l, \dots, \mathbf{a}_k)$ for any variable $\mathbf{a}$ and $0 \leq l \leq k$. A single subscript implies that the starting time index is 0, i.e., $\tilde{\mathbf{a}}_k = \tilde{\mathbf{a}}_{0:k}$. For any $k \in \mathbb{N}$, we define the solution mappings as

$$\begin{aligned}(\phi_k, \psi_k) : \mathbb{R}^{(k+1)n_u} \times \mathbb{I}^{k+1} \times \mathbb{R}^{n_x} \times \mathbb{R}^{kn_w} \times \mathbb{R}^{n_v} \\ \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_y},\end{aligned}$$

such that $\phi_k(\tilde{\mathbf{u}}_k, \tilde{i}_k, \mathbf{x}_0, \tilde{\mathbf{w}}_{k-1}, \mathbf{v}_k)$ and $\psi_k(\tilde{\mathbf{u}}_k, \tilde{i}_k, \mathbf{x}_0, \tilde{\mathbf{w}}_{k-1}, \mathbf{v}_k)$ correspond to the state and output in (1a) and (1b), respectively, given the specified inputs. In other words, these mappings represent the recursive application of the dynamic system (1) from time 0 up until time k, which depend on the collection of control input, model index, and disturbance sequences as well as the initial state and most recent measurement noise term. Note that the state mapping ϕ_k does not depend on $\mathbf{u}_k$, i_k, and $\mathbf{v}_k$, but they are included for notational convenience as ϕ_k does depend on $\mathbf{u}_{k-1}$, i_{k-1}, and $\mathbf{v}_{k-1}$. We further define sequences of these mappings as $\tilde{\phi}_{l:k} = (\phi_l, \dots, \phi_k)$ and $\tilde{\psi}_{l:k} = (\psi_l, \dots, \psi_k)$ where the arguments of these functions have been omitted for the sake of brevity. For each $\tilde{i}_k \in \mathbb{I}^{k+1}$ and $\tilde{\mathbf{u}}_k \in \mathbb{R}^{(k+1)n_u}$, we define the *reachable state and output sets* on the interval $[l, k]$ as $\tilde{\Phi}_{l:k}(\tilde{\mathbf{u}}_k, \tilde{i}_k)$ and $\tilde{\Psi}_{l:k}(\tilde{\mathbf{u}}_k, \tilde{i}_k)$. The reachable state and output sets at a particular time index k then correspond to $\Phi_k(\tilde{\mathbf{u}}_k, \tilde{i}_k) = \tilde{\Phi}_{k:k}(\tilde{\mathbf{u}}_k, \tilde{i}_k)$ and $\Psi_k(\tilde{\mathbf{u}}_k, \tilde{i}_k) = \tilde{\Psi}_{k:k}(\tilde{\mathbf{u}}_k, \tilde{i}_k)$. Note that $\tilde{\Phi}_{l:k}$ and $\tilde{\Psi}_{l:k}$ still depend on the full input and model index sequences since we always assume the system evolves from the initial condition X_0 starting at time 0.

Computing the exact reachable sets is not possible for general nonlinear systems such that we will rely on approximate computational procedures in practice. We discuss this further in Section 3.3, which describes how we can compute over- and under-approximations using established algorithms.

2.4. Optimal separating inputs

A "fault" (i.e., change in the behavior of the system) occurring at time k is modeled as a transition from one model in $\mathbb{I}$ to another such that $i_k \neq i_{k-1}$. Following a similar paradigm to [48], we define a *fault scenario* on an interval of $[0, N]$ (with $N > 0$ denoting the final time index) as a sequence of model indices $\tilde{i}_N \in \mathbb{I}^N$. We let $\tilde{\mathbb{I}} \subset \mathbb{I}^N$ denote the set of allowable fault scenarios on $[0, N]$. Note that, in general, a fault scenario could involve a different fault occurring at every time interval, which would correspond to multiple, simultaneous faults. As discussed in Remark 1, this would result in a combinatorial explosion of the number of fault scenarios that we need to check. However, in practice, many of these scenarios are not important such that we can explore a much smaller subset of all possible scenarios in $\mathbb{I}^N$. We are interested in finding feasible open-loop input sequences $\tilde{\mathbf{u}}_N$ over this interval such that: (i) the observed output sequence $\tilde{\mathbf{y}}_N$ is consistent with only a single fault scenario in $\tilde{\mathbb{I}}$ for all possible unobserved variables (2) and (ii) state safety constraints are satisfied for all all fault scenarios in $\tilde{\mathbb{I}}$. Input sequences that satisfy these conditions are called *separating inputs*. We can characterize the set of separating inputs $\mathscr{S}(\tilde{\mathbb{I}}) = S(\tilde{\mathbb{I}}) \cap C(\tilde{\mathbb{I}})$ as the intersection of the inputs that ensure separation of the fault scenarios $S(\tilde{\mathbb{I}})$ and the inputs that ensure state constraint satisfaction $C(\tilde{\mathbb{I}})$, which depend on the set of fault scenarios $\tilde{\mathbb{I}}$. We can express these sets directly in terms of the reachable state and output sets

$$\begin{aligned}S(\tilde{\mathbb{I}}) = \{\tilde{\mathbf{u}}_N : \tilde{\Psi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N) \cap \tilde{\Psi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{j}_N) = \emptyset, \\ \forall \tilde{i}_N, \tilde{j}_N \in \tilde{\mathbb{I}},\ \tilde{i}_N \neq \tilde{j}_N\}, \\ C(\tilde{\mathbb{I}}) = \{\tilde{\mathbf{u}}_N : \tilde{\Phi}_k(\tilde{\mathbf{u}}_N, \tilde{i}_N) \subseteq X, \forall k \in \mathbb{N}_1^N, \forall \tilde{i}_N \in \tilde{\mathbb{I}}\}.\end{aligned}$$

where $\mathbb{N}_1^N = \{1, 2, \dots, N\}$.

It is possible that many possible input sequences satisfy these constraints. As such, in this work, we are particularly interested in finding so-called *optimal separating inputs* that further minimize a cost function $J : U^{N+1} \to \mathbb{R}$ that measures the "energy" of the input sequence (e.g., average deviation from the nominal operating conditions) subject to the separability and safety conditions. An optimal separating input can be found by solving

$$\inf_{\tilde{\mathbf{u}}_N} \{J(\tilde{\mathbf{u}}_N) : \tilde{\mathbf{u}}_N \in \tilde{U}_N \cap \mathscr{S}(\tilde{\mathbb{I}})\}. \tag{4}$$

Solving this problem in practice, however, is very challenging mainly due to the difficulty in characterizing the set $\mathscr{S}(\tilde{\mathbb{I}})$. As shown in, e.g., [48], this set is non-convex even for simple linear time-invariant systems, with a cost that quickly grows with the number of states, vertices in polytopic representations of the sets, and number of fault scenarios $s = |\tilde{\mathbb{I}}|$. Although these costs can be partly controlled using zonotope

set representations, these ideas do not easily extend to general nonlinear systems. Here, we take a different approach that looks to sidestep the creation of an implicit form of $\mathscr{S}(\tilde{\mathbb{I}})$ that is likely to be computationally expensive (and must be tailored to the details of the dynamics). Instead, we opt for a general approach that aims to learn an explicit representation of $\mathscr{S}(\tilde{\mathbb{I}})$ from data in a sample-efficient manner. The details of our proposed method are described in the next section.

Remark 1. *The separability conditions in $\mathscr{S}(\tilde{\mathbb{I}})$ require checking that every distinct pair of scenarios $\tilde{i}_N, \tilde{j}_N \in \tilde{\mathbb{I}}$ are not overlapping, which results in $\binom{s}{2}$ with $s = |\tilde{\mathbb{I}}|$. The combinatorial growth of these constraints can be prohibitively expensive when either the number of models n_m or horizon N is large since $s = (n_m)^N$ scenarios are needed to represent all possible situations. However, as noted in [48], many of these scenarios are either irrelevant or extremely unlikely in practice. The size of s can also be greatly reduced by assuming a slow frequency of occurrence of faults (e.g., $i_0 = i_1 = \ldots = i_N$ over the horizon of interest). We assume that $\tilde{\mathbb{I}}$ is known, though its specific form is dependent on the application.*

Remark 2. *In this work, we focus primarily on the "offline" version of (4), where we aim to identify optimal input sequences $\mathbf{u}_N^\star$ for a variety of scenarios. For example, we may solve the problem for different values of X_0, $\tilde{\mathbb{I}}$, or $\tilde{U}_N$, with the results ready for online deployment in various contexts. In practice, these pre-computed sequences $\mathbf{u}_N^\star$ would be used once an online fault detection method indicates a potential fault. Consequently, there are no strict real-time constraints for solving (4). However, since we deal with complex, large-scale models across many conditions, efficient algorithms are essential to ensure that we can find reasonable solutions within a feasible computational budget. Note that there has been work on so-called "closed-loop active fault diagnosis," which does attempt to repeatedly solve (4) in a receding-horizon fashion, e.g., [45, 40]. Extending our method to this setting is an interesting direction for future work.*

3. Active Deep Learning of Optimal Separating Inputs

In this section, we describe our proposed approach. We first discuss how to view (4) as a "learning problem" using a separability oracle and then discuss an efficient active learning strategy for querying points that are particularly informative for learning optimal separating input sequences. We end this section by discussing how one can practically query over- and under-approximations of the separability oracle using state-of-the-art reachability methods.

3.1. Learning to distinguish separating and non-separating inputs

The core observation motivating our approach is that we can effectively take a "data-driven" view of (4) by learning an explicit (approximate) representation of the set of separating inputs $\mathscr{S}(\tilde{\mathbb{I}})$. This is achieved by defining a *separability oracle* $\mathcal{O} : \tilde{U}_N \to \{0, 1\}$ that maps an input sequence to a binary number

$$\mathcal{O}(\tilde{\mathbf{u}}_N) = \begin{cases} +1, & \text{if } \tilde{\mathbf{u}}_N \in \mathscr{S}(\tilde{\mathbb{I}}), \\ 0 & \text{if } \tilde{\mathbf{u}}_N \notin \mathscr{S}(\tilde{\mathbb{I}}), \end{cases} \tag{5}$$

where the explicit dependence on the fault scenarios $\tilde{\mathbb{I}}$ is omitted for brevity. Since $\mathcal{O}(\tilde{\mathbf{u}}_N) = +1$, $\forall \tilde{\mathbf{u}}_N \in \mathscr{S}(\tilde{\mathbb{I}})$ and $\mathcal{O}(\tilde{\mathbf{u}}_N) = 0$, $\forall \tilde{\mathbf{u}}_N \notin \mathscr{S}(\tilde{\mathbb{I}})$, we can interpret the set $\mathscr{S}(\tilde{\mathbb{I}})$ as the decision boundary separating the two classes defined by $\mathcal{O}$. Although we do not know the structure of $\mathcal{O}$, we can attempt to learn it from data as long as we can label input sequences. We defer a discussion on the labeling process to Section 3.2. For now, let us assume that we can obtain a set $\mathcal{L}$ of labeled samples

$$\mathcal{L} = \{(\tilde{\mathbf{u}}_N^{(1)}, \mathcal{O}(\tilde{\mathbf{u}}_N^{(1)})), \ldots, (\tilde{\mathbf{u}}_N^{(|\mathcal{L}|)}, \mathcal{O}(\tilde{\mathbf{u}}_N^{(|\mathcal{L}|)}))\}, \tag{6}$$

where $|\mathcal{L}| < \infty$. We can use $\mathcal{L}$ to train a surrogate model for the unknown function $\mathcal{O}$; we denote the model by $\mathcal{M}_\theta$ where θ are the set of model parameters. Although in principle we can use any surrogate model, we focus on deep neural network (DNN) models in this work since $\mathcal{O}$ can have a complex structure. DNNs with more than 1 hidden layer are postulated to be universal function approximators under relatively mild conditions [17] such that they can be flexibly applied to many different problems of interest. We investigate the choice of model type in the context of an example in Section 4. Here, we restrict the activation function of the output layer to be a sigmoid function such that the output of the DNN corresponds to the probability of the separating input class label being equal to +1. The full set of DNN parameters, which consists of the weights and bias terms in each layer of the network, are trained by minimizing the binary cross-entropy loss function [33]

$$\theta^\star(\mathcal{L}) = \text{argmin}_\theta -\tfrac{1}{|\mathcal{L}|} \textstyle\sum_{i=1}^{|\mathcal{L}|} \ell_i(\theta), \tag{7}$$

where $\ell_i(\theta) = y_i \log(p_i(\theta)) + (1 - y_i) \log(1 - p_i(\theta))$, $y_i = \mathcal{O}(\tilde{\mathbf{u}}_N^{(i)})$ denotes the true label (equal to +1 for separating inputs and 0 for non-separating inputs) for the ith sample point, and $p_i(\theta) = \mathcal{M}_\theta(\tilde{\mathbf{u}}_N^{(i)})$ denotes the predicted probability for the +1 (separating input) class for the ith sample point. In practice, (7) is solved approximately using local gradient-based optimization techniques such as stochastic gradient descent. Interested readers are referred to [17] for details on DNN training; we rely on standard methods throughout this work. We let $\mathcal{M}_\mathcal{L} = \mathcal{M}_{\theta^\star(\mathcal{L})}$ denote the final DNN model trained using labeled dataset $\mathcal{L}$.

This approach will be effective under the assumption that $\mathcal{L}$ is a sufficiently rich dataset for training $\mathcal{M}_\mathcal{L}$. This may not be the case if the input sample locations are chosen in a naive fashion such as uniformly at random. A more practical sampling method is to use active learning (AL) to select samples that are likely to be informative toward our end goal. This is particularly important in the application of interest in this work where querying $\mathcal{O}$ is expected to be relatively computationally expensive, meaning we do not

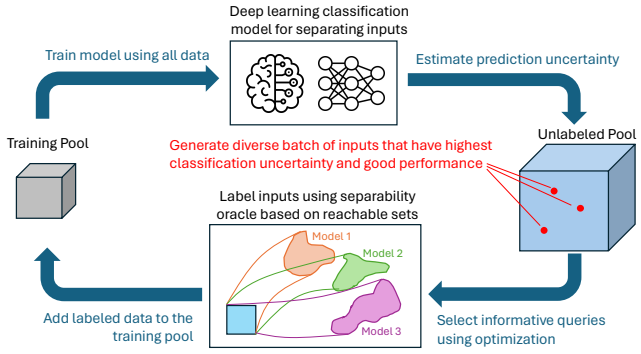


Figure 1: Illustration of proposed active deep learning framework for efficiently identifying near-optimal (low-energy) separating inputs that, when provided to a system, ensure that only a unique fault scenario is consistent with the measurements.

have the luxury to "waste" queries on samples that may not improve our knowledge about the solution to (4).

Remark 3. *DNN models have several hyperparameters that can be tuned in practice including the number of nodes and layers, the selected activation functions in the hidden layers, and the choice of parameters in the training procedure (e.g., learning rate). Here, we assume that these hyperparameters have been appropriately selected; however, this assumption can be straightforwardly relaxed in practice by incorporating automated hyperparameter tuning methods. There has been a substantial amount of recent work on this topic [19, 16, 55, 6] that can be directly leveraged by our approach (the auto-tuning method is simply folded into the model training process). The time required to complete the training process depends on several details including the specific training and hyperparameter tuning algorithms as well as the available computing resources. Since we are mainly focused on development of a general framework, these issues are not addressed in this paper; however, we believe a systematic exploration of this space is an interesting direction for future work. We do provide a discussion on the implementation details, for both the DNN and the oracle $\mathcal{O}$, in Appendix A.*

3.2. Active learning algorithm for optimal separating inputs

The main advantage of AL over traditional supervised learning methods is the ability to systematically utilize prior learners/models to iteratively improve performance. An illustrative figure of how AL can be used to solve (4) by learning how to classify separating input sequences using an oracle $\mathcal{O}$ is shown in Figure 1. A complete description of such an AL algorithm is also provided in Algorithm 1. The main idea is that, at every iteration t, one trains a classification model $\mathcal{M}_{\mathcal{L}_t}$ given the current labeled data $\mathcal{L}_t$. This model is then used to select a batch of new (unlabeled) "important" input sequences that are then labeled by querying the oracle, denoted by $\mathcal{B}_t$. This new data is appended to the current dataset $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \mathcal{B}_t$, and the entire process is repeated until the sampling budget has been exhausted.

Algorithm 1 Proposed active deep learning method for training classification surrogate model to efficiently identify optimal separating input sequences

1: **Inputs:** Separability oracle $\mathcal{O}$, initial data $\mathcal{L}_0$, budget N_s, batch size B, and sample selection function `batch_select`$(\cdot)$.
2: **Initialize:** Iteration $t \leftarrow 0$, train DNN model $\mathcal{M}_{\mathcal{L}_0}$ (Section 3.1).
3: **while** budget remaining, i.e., $|\mathcal{L}_t| < N_s$ **do**
4: Get batch $\tilde{\mathcal{U}}_N^{(t)} =$ `batch_select`$(\mathcal{M}_{\mathcal{L}_t}, \mathcal{L}_t, B)$.
5: Compute labels $\mathcal{O}(\tilde{\mathcal{U}}_N^{(t)})$.
6: Update data $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \mathcal{B}_t$ where $\mathcal{B}_t = \{(\tilde{\mathcal{U}}_N^{(t)}, \mathcal{O}(\tilde{\mathcal{U}}_N^{(t)}))\}$.
7: Train new DNN model $\mathcal{M}_{\mathcal{L}_{t+1}}$ (Section 3.1).
8: Increment iteration counter $t \leftarrow t + 1$.
9: **end while**
10: **return** Final model $\mathcal{M}_{\mathcal{L}_t}$ and labeled dataset $\mathcal{L}_t$.

Line 4 is a critical step in Algorithm 1, which identifies the next batch of "important" input sequence samples that are deemed worthy of spending part of our budget toward labeling them. Previous work has focused on the use of Shannon entropy to quantify the informativeness of new samples $\tilde{\mathbf{u}} \in \tilde{U}$, e.g., [11, 7] (note we drop the subscript N in this section for notational simplicity); however, these methods are focused on identifying samples that globally improve the classifier. Here, we are most interested in optimal (minimal energy) input sequences according to (4) and therefore want to prioritize finding samples that help us better distinguish between low-energy separating and non-separating input sequences. Thus, we propose a novel `batch_select` function that accounts for both the energy of the input sequence and the prediction uncertainty in the classifier, which is summarized in Algorithm 2. The algorithm starts by calculating the incumbent value η, which corresponds to the minimum cost separating input sequence contained in the training data $\mathcal{L}$. The quantity η serves as an upper bound on the solution to (4). It then generates a finite pool of candidate points by randomly sampling a large number of values in the set $\{\tilde{\mathbf{u}} : J(\tilde{\mathbf{u}}) < \eta\}$. Shannon entropy is still used to quantify informativeness of future input sequences; the expected gain in information by adding a new sample $\tilde{\mathbf{u}}$ to the existing labeled data is given by

$$I(\tilde{\mathbf{u}}; \mathcal{M}_{\mathcal{L}}) = -\mathcal{M}_{\mathcal{L}}(\tilde{\mathbf{u}}) \log(\mathcal{M}_{\mathcal{L}}(\tilde{\mathbf{u}})) - (1 - \mathcal{M}_{\mathcal{L}}(\tilde{\mathbf{u}})) \log(1 - \mathcal{M}_{\mathcal{L}}(\tilde{\mathbf{u}})). \tag{8}$$

Instead of maximizing $I(\tilde{\mathbf{u}}; \mathcal{M}_{\mathcal{L}})$ over the entire $\tilde{U}$ space, we maximize it over the finite pool $\mathcal{P}$ that only includes samples that improve upon the incumbent cost. The pool consists of a finite number of points that is typically generated by random sampling in the input space $\tilde{U}$. We avoid using continuous optimization in this step for two main reasons: it ensures there is at least some distance between possible inputs so that there is little chance of redundant queries and it reduces computational cost. It is common in the AL literature to

Algorithm 2 `batch_select`$(\mathcal{O}, \mathcal{M}, \mathcal{L}, B; \tilde{U}, J, \boldsymbol{\varepsilon})$

Inputs: DNN model $\mathcal{M}$, training data $\mathcal{L}$ that consists of input sequences and oracle labels as defined in (6), batch size B, input constraints $\tilde{U}$, input cost (energy) function J, and the number of optimization-based samples in a batch with $b < B$ probability levels $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_b)$.
Set batch of samples to empty $\tilde{\mathcal{U}} = \emptyset$.
Get incumbent $\eta = \min_{(\tilde{\mathbf{u}},o)\in\mathcal{L}}\{J(\tilde{\mathbf{u}}) : o = 1\}$ (cost of best separating input).
Generate candidate pool $\mathcal{P}$ by randomly generating input sequence values in $\tilde{U}$ with cost less than η.
for $i = 1, \dots, B$ **do**
  if $i > b$ **then**
    Solve the optimization given $I(\cdot)$ in (8):

$$\tilde{\mathbf{u}}^\dagger \leftarrow \operatorname{argmax}_{\tilde{\mathbf{u}}\in\mathcal{P}\setminus\tilde{\mathcal{U}}}\{I(\tilde{\mathbf{u}}; \mathcal{M})\}. \tag{9}$$

  else
    Solve the optimization problem:

$$\tilde{\mathbf{u}}^\dagger \leftarrow \operatorname{argmin}_{\tilde{\mathbf{u}}\in\tilde{U}}\{J(\tilde{\mathbf{u}}) : \mathcal{M}(\tilde{\mathbf{u}}) \geq \varepsilon_i\}. \tag{10}$$

  end if
  Append $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \{\tilde{\mathbf{u}}^\dagger\}$.
end for
return Selected batch of important input sequences $\tilde{\mathcal{U}}$.

have a fixed starting pool of unlabeled samples; however, the quality of the final incumbent will be limited by the quality of the starting pool. Since it is unlikely that the solution to (4) is inside $\mathcal{P}$, we also incorporate a step that runs a continuous optimization procedure in (10) to find the lowest cost input subject to a constraint on the predicted probability that the input is a separating input. Low (high) probability levels lead to more explorative (exploitative) sample selections. Since it is useful to have a mixture of exploration and exploitation, we introduce parameters $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_b)$ into the algorithm. Specifically, these parameters correspond to $b < B$ distinct probability levels that are used to generate samples through continuous optimization. We found that a default selection of $b = 4$, $\varepsilon_1 = 0.3$, $\varepsilon_2 = 0.4$, $\varepsilon_3 = 0.5$, and $\varepsilon_4 = 0.6$ gave consistently good results in our numerical examples for batch sizes $8 \leq B \leq 16$. We plan to study the impact of these choices more in future work, though we did not find the results were particularly sensitive to the number or exact values of $\boldsymbol{\varepsilon}$.

We provide a visual illustration of the proposed Algorithms 1–2 on a benchmark example in Appendix B.

3.3. Practical over- and under-estimation of reachable sets

In most cases of interest, we cannot compute exact reachable sets $\tilde{\Phi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N)$ and $\tilde{\Psi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N)$, which are needed to evaluate the true separability oracle $\mathcal{O}(\tilde{\mathbf{u}}_N)$. Instead, we rely on existing approximation methods that can be computed in a reasonable amount of time. Let superscripts I and O, respectively, denote inner and outer approximations of the reachable sets

$$\tilde{\Phi}^I_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N) \subseteq \tilde{\Phi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N) \subseteq \tilde{\Phi}^O_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N),$$
$$\tilde{\Psi}^I_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N) \subseteq \tilde{\Psi}_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N) \subseteq \tilde{\Psi}^O_{0:N}(\tilde{\mathbf{u}}_N, \tilde{i}_N).$$

The outer approximated reachable sets are particularly useful since they can be used to derive input sequences that guarantee the safety and separation constraints are satisfied, though the resulting cost (energy) might be suboptimal. The inner approximated sets, on the other hand, are useful for helping measure the suboptimality gap. To formalize the idea, we define the following notation

$$J^\star = J(\tilde{\mathbf{u}}^\star_N), \quad \tilde{\mathbf{u}}^\star_N = \operatorname*{argmin}_{\tilde{\mathbf{u}}_N\in\tilde{U}_N}\{J(\tilde{\mathbf{u}}_N) : \mathcal{O}(\tilde{\mathbf{u}}_N) = 1\},$$
$$J^\star_L = J(\tilde{\mathbf{u}}^L_N), \quad \tilde{\mathbf{u}}^L_N = \operatorname*{argmin}_{\tilde{\mathbf{u}}_N\in\tilde{U}_N}\{J(\tilde{\mathbf{u}}_N) : \overline{\mathcal{O}}(\tilde{\mathbf{u}}_N) = 1\},$$
$$J^\star_U = J(\tilde{\mathbf{u}}^U_N), \quad \tilde{\mathbf{u}}^U_N = \operatorname*{argmin}_{\tilde{\mathbf{u}}_N\in\tilde{U}_N}\{J(\tilde{\mathbf{u}}_N) : \underline{\mathcal{O}}(\tilde{\mathbf{u}}_N) = 1\},$$

where $J^\star$ denotes the energy of the exact optimal separating $\tilde{\mathbf{u}}^\star_N$ and $J^\star_L$ and $J^\star_U$ are, respectively, lower and upper bounds on $J^\star$ (i.e., $J^\star_L \leq J^\star \leq J^\star_U$) found from the approximate reachable sets. Notice how $\tilde{\mathbf{u}}^\star_N$ has been equivalently defined in terms of the separability oracle (equivalent to the solution of (4)). The input sequence $\tilde{\mathbf{u}}^L_N$, which leads to the lower bound on the cost, is defined similarly to $\tilde{\mathbf{u}}^\star_N$ with the main difference being that the uncomputable exact oracle $\mathcal{O}$ has been replaced by an upper bound $\overline{\mathcal{O}}$ constructed from the inner reachable set approximations $(\cdot)^I$. The function $\overline{\mathcal{O}}$ upper bounds $\mathcal{O}$ because inner reachable set approximations lead to outer approximations of the separability set $\mathscr{S}(\tilde{\mathbb{I}})$, which corresponds to a relaxation of the true feasible region of (4). Similarly, the input sequence $\tilde{\mathbf{u}}^U_N$ leads to an upper bound on the cost using a lower bound $\underline{\mathcal{O}}$ of $\mathcal{O}$ constructed from the outer reachable set approximations $(\cdot)^O$. Since the proposed AL framework applies to any form of $\mathcal{O}$, we can simply apply it to $\overline{\mathcal{O}}$ and/or $\underline{\mathcal{O}}$ depending on what information is needed. The quality of the upper and lower bounds for $J^\star$ will depend on the tightness of the reachable set approximations. Tighter approximations are more expensive to compute, which is the main rationale behind the use of AL in this context.

There has been a substantial amount of work on outer-approximated reachability analysis in recent years [4, 46, 54, 53]. In the numerical examples considered in this work, we focus on the use of COntinuous Reachability Analyzer (CORA) [2], which is a Matlab toolbox implementing an array of algorithms for reachability analysis of various system types including those with continuous, discrete, and hybrid dynamics. CORA supports many types of linear and nonlinear dynamics – the main restriction is that $\mathbf{f}$ and $\mathbf{h}$ satisfy some continuity conditions in at least some regions of the state space. The core functionality of CORA is also modular, so that new algorithms utilizing basic set types and operations can be easily implemented in the toolbox. Again, we note that future developments in reachability analysis

can be directly leveraged by our proposed AL method, as it is agnostic to the source of data. To construct inner approximations of the reachable sets, we can rely on the approach from [26] that extract them from the pre-computed outer-approximations.

4. Numerical Examples

In this section, we demonstrate the proposed optimal input design method (Algorithms 1–2) on three problems. The first is a benchmark linear system, which we use to illustrate some key aspects of our method such as the importance of the type of model and the sequential (active) query point selection strategy. The other two problems are a benchmark and an industrially-relevant continuously stirred tank reactor that are meant to showcase the generality and scalability of our method. To the best of our knowledge, our approach is the first to find low-energy input sequences that provide a guaranteed fault diagnosis in such complex systems.

Note the code used to generate these results can be found on Github: `https://github.com/PaulsonLab/Guaranteed_AFD_ADL`.

4.1. Illustrative linear system

To illustrate some of the key features of our proposed approach, we initially apply it to a simple benchmark linear system that is a variation of that studied in [47]. The system dynamics are of the form (1) where $\mathbf{f}$ and $\mathbf{h}$ are given by the following linear expressions

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k, i_k) = \mathbf{A}^{[i_k]}\mathbf{x}_k + \mathbf{B}^{[i_k]}\mathbf{u}_k + \mathbf{B}_w^{[i_k]}\mathbf{w}_k \quad (11a)$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k, i_k) = \mathbf{C}^{[i_k]}\mathbf{x}_k + \mathbf{D}_v^{[i_k]}\mathbf{v}_k, \quad (11b)$$

where $\mathbf{A}^{[i]}$, $\mathbf{B}^{[i]}$, $\mathbf{B}_w^{[i]}$, $\mathbf{C}^{[i]}$, and $\mathbf{D}_v^{[i]}$ are system matrices describing the interaction between two signals. In this case, we take a total of 3 fault scenarios over a horizon of $N = 1$ corresponding to one of the nominal or two faulty models being active. The nominal model system matrices and vectors are given by

$$\mathbf{A}^{[1]} = \begin{bmatrix} 0.6 & 0.2 \\ -0.2 & 0.7 \end{bmatrix}, \ \mathbf{B}^{[1]} = \begin{bmatrix} -0.3861 & 0.1994 \\ -0.1994 & 0.3861 \end{bmatrix},$$
$$\mathbf{C}^{[1]} = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.3 \end{bmatrix}, \ \mathbf{B}_w^{[1]} = \begin{bmatrix} 0.1215 & 0.0598 \\ 0.0598 & 0.1215 \end{bmatrix},$$
$$\mathbf{D}_v = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and the two fault models are defined as being the same as the nominal model except for the following modifications

$$\mathbf{B}^{[2]} = \begin{bmatrix} -0.3861 & 0 \\ -0.1994 & 0 \end{bmatrix}, \ \mathbf{B}^{[3]} = \begin{bmatrix} 0 & 0.1994 \\ 0 & 0.3861 \end{bmatrix}.$$

Models 2 and 3 correspond to a fault/failure in the second and first actuator, respectively. The uncertainty sets (2) are represented as follows

$$X_0 = \left\{ \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xi : \xi \in \mathbb{R}^2, \ \|\xi\|_\infty \leq 1 \right\},$$

$$W = V = \{\mathbf{w} \in \mathbb{R}^2 : \|\mathbf{w}\|_\infty \leq 0.1\},$$

We do not consider hard state constraints in this example, i.e., $X = \mathbb{R}^2$. The bounded input constraints are given by

$$U = \{\mathbf{u} \in \mathbb{R}^2 : \|\mathbf{u}\|_\infty \leq 10\}.$$

The cost function $J(\mathbf{u}_0) = \|\mathbf{u}_0\|_2$ corresponds to the 2-norm of the input vector.

4.1.1. Importance of deep neural network classifier

Note that, since the output does not depend on $\mathbf{u}_k$ in this case, the set of separating inputs $\mathscr{S}(\tilde{\mathbb{I}})$ only depends on $\tilde{\mathbf{u}}_{N-1}$. Further, due to the simplicity of the constraint sets and the system dynamics, we can exactly compute the reachable sets for this problem, which enables us to visualize $\mathscr{S}(\tilde{\mathbb{I}})$ by querying the oracle (5) on a fine 100×100 grid of points in the two-dimensional space $\tilde{\mathbf{u}}_0 = \mathbf{u}_0 \in U$; the results are shown in Figure 2 with $\mathscr{S}(\tilde{\mathbb{I}})$ being depicted in green. Notice how this set is disjoint and non-convex, even for such simple dynamics. We also see relatively sharp features that are non-trivial to capture with many existing classification model types. To study the impact of the choice of model, we compare the DNN (Section 3.1) to three alternative machine learning methods: (i) support vector classification (SVC), (ii) Gaussian process classification (GPC), and (iii) k-nearest neighbor (kNN) classification. We use the default implementations of these methods in the `scikit-learn` package in Python [44]. We use a DNN with 3 hidden layers, 20 nodes in the first layer, 13 in the second layer, 7 in the third layer, and ReLU activation functions, which is implemented using the PyTorch package [38]. The Adam optimizer [24] was used for training by minimizing the cross-entropy loss function (7) for 1000 epochs with a batch size of 200 and a learning rate of 8×10^{-4} (all inputs are scaled to be in the range of -1 to 1). We found these architectural choices to be robust across the examples considered in this work; these parameters can easily be modified in accordance with Remark 3.

A performance comparison of these models trained using labeled data collected by querying the oracle at 1024 and 2048 quasi-random Sobol samples from U is shown in Figure 3. We see that the DNN performs the best out of all models in both cases and is especially good at resolving the sharp transitions that occur near the optimal solution. The kNN model is the second-best model, though it does tend to struggle near the boundary. This issue is exacerbated in higher dimensions where the volume of the search space increases exponentially. DNNs are less susceptible to this issue and have more overall flexibility such that we prefer them over kNNs in this work. Both the GPC and SVC show a clear drop in performance compared to the DNN and kNN, as they assume a stronger degree of smoothness of the decision boundary that is not satisfied in this problem.

4.1.2. Performance comparison with other methods

We now compare our proposed method in Algorithms 1–2 to two alternative methods. The first is a standard "passive learning" approach that simply generates data uniformly at

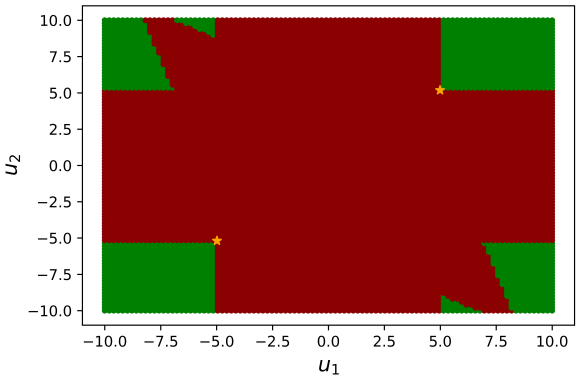


Figure 2: Illustration of the set of separating inputs for the linear system over a single-step time horizon $N = 1$. Input values that do and do not lead to separation of the output reachable sets are shown in green and red, respectively. Thus, any input in the green region provides a guaranteed fault diagnosis when given to the system. The optimal separating input values that minimize the cost function are shown with yellow markers.

random within the input constraint bounds $\tilde{U}_N$ (equal to U in this case). The second is a more traditional active learning method that follows the same skeleton as Algorithm 1 but uses a different `batch_select` function that maximizes the expected information gain in (8) over a fixed large pool of unlabeled samples (generated through random sampling in the full $\tilde{U}_N$ input space). All algorithms are run using 32 initial data points generated uniformly at random within the input constraint set, a batch size of $B = 8$, and a total budget of $N_s = 32 + 8 \times 30 = 272$ (equivalent to 30 total learning iterations). For our proposed method, we set ϵ as described in Section 3.2. Due to the inherent randomness of the selection of the initial labeled and unlabeled points, we perform 20 independent replicates of each algorithm (all algorithms share the same random seed to ensure a fair comparison). The best found value (i.e., the incumbent defined in Line 3 of Algorithm 2) for each algorithm versus the number of learning iterations averaged over the replicates is shown in Figure 4. The shaded regions correspond to confidence bounds computed from ± 1 standard deviation. We see that our method outperforms passive learning and traditional active learning for the given budget. Furthermore, it consistently gets close to the exact solution computed using the method in [47]. It is worth highlighting that we see advantages of our method even in such a simple case that only involves two independent input dimensions. As we study next, the observed improvement in sampling efficiency grows as the problem complexity increases.

4.2. Expanded linear system

We next consider a larger-scale extension of the illustrative example in (11). In particular, we incorporate two

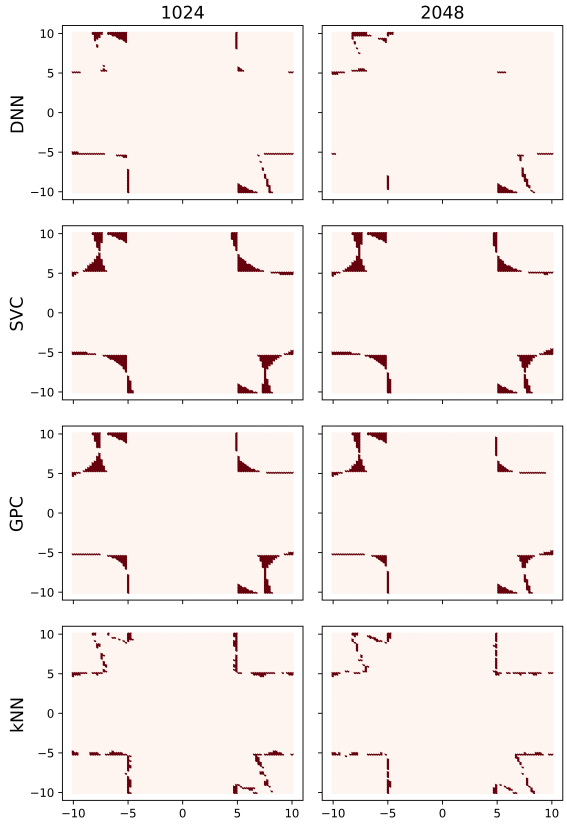


Figure 3: Classifier performance comparison using different model types trained using 1024 (left column) and 2048 (right column) quasi-random Sobol samples from the input domain. The dark red points indicate an incorrect prediction of the true class (in Figure 2), while the light points indicate a correct prediction.

additional system fault models

$$\mathbf{A}^{[4]} = \begin{bmatrix} 0.6 & 0 \\ -0.2 & 0.7 \end{bmatrix}, \; \mathbf{A}^{[5]} = \begin{bmatrix} 0.6 & 0.2 \\ 0 & 0.7 \end{bmatrix},$$

and we consider a longer time horizon of $N = 4$. In this case, the set of separating inputs now depends on an 8-dimensional vector corresponding to the 2 inputs over 4 time steps. For simplicity, we only require separation of the output sets at the final time step, which is technically an inner approximation of the constraints in $\mathscr{S}(\tilde{\mathbb{I}})$ that only require separation of the full output reachable sets over all time steps. We repeat the same performance comparison analysis from the previous section (only change is that 512 random initial points are used); the best found value for each algorithm as a function of number of learning iterations is shown in Figure 5. Here, we see a substantial improvement in the quality of the best found separating input sequence, achieving more than a 2x reduction in input cost over the course of 30 learning iterations. Furthermore, in nearly all 20 replicates, our method found a solution that was within $< 0.1\%$ of the exact solution.

To demonstrate the importance of finding a better separating input, we also plot the output reachable sets (and

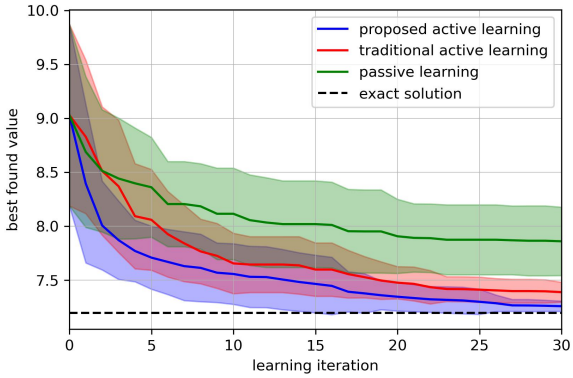


Figure 4: Performance of our proposed algorithm versus traditional active and passive learning on the illustrative linear problem. All algorithms are repeated 20 times from different random seeds; the solid line depicts average performance while the shaded region corresponds to ±1 standard deviation. The black dotted line is the exact solution computed with the method from [47].

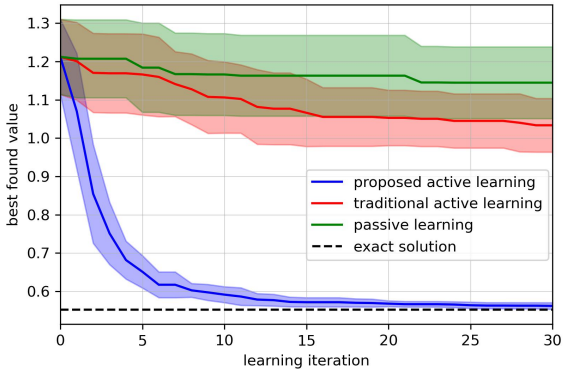


Figure 5: Performance of our proposed algorithm versus traditional active and passive learning on 8d version of the linear problem. All algorithms are repeated 10 times from different random seeds; the solid line depicts average performance while the shaded region corresponds to ±1 standard deviation. The black dotted line is the exact solution computed with the method from [47].

100,000 Monte Carlo samples of the outputs under random realizations of the uncertainty) at the final time for each of the five models given the best found solution for each method and the exact solution in Figure 6. We see that our proposed method closely matches the exact solution, while the other methods deviate substantially. Even though all methods yield disjoint output sets, implying the measurements taken on $[0, N]$ can be consistent with at most one model, our method ensures these sets are as close together as possible, meaning they create substantially less disruption to the operation. The separating input profiles for our method and the exact solution are shown in Figure 7, which highlights that a relatively small input perturbation can reveal the true status of the system.

Note that we also show that our method continues to achieve good performance when safety constraints (on states or outputs) are incorporated in Appendix C.

4.3. Nonlinear continuously stirred tank reactor

We now consider a highly nonlinear continuously stirred tank reactor (CSTR) model of an exothermic, irreversible reaction $A \rightarrow B$ with constant liquid volume. A model of this type of CSTR is given by [32, 8]

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = \frac{q}{V}(C_{Af} - C_A) - k_0 \exp\left(-\frac{E}{RT}\right) C_A, \quad (12a)$$

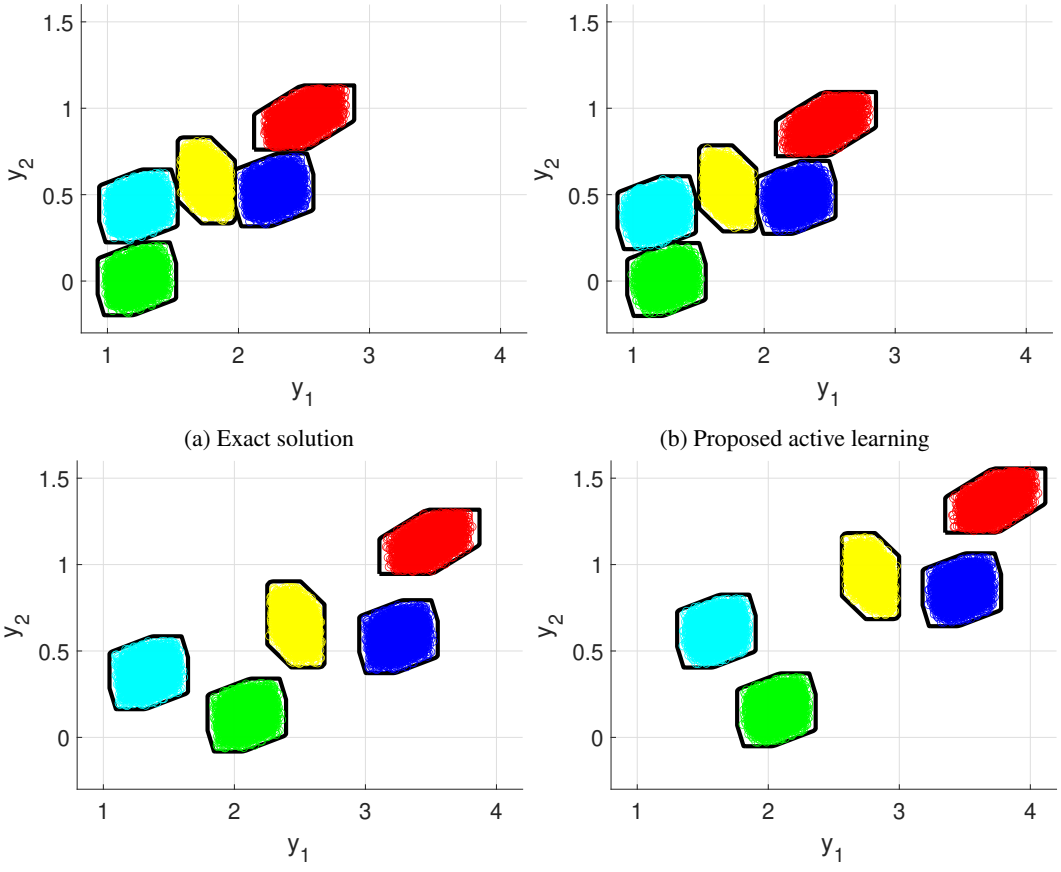
$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{q}{V}(T_f - T) - \frac{\Delta H k_0}{\rho C_p} \exp\left(-\frac{E}{RT}\right) C_A + \frac{UA}{\rho V C_p}(T_c - T), \quad (12b)$$

where C_A is the concentration of A in the reactor, T is the temperature of the reactor, T_c is the temperature of the coolant stream, and C_{Af} is the feed concentration. The model parameters are $\rho = 1000$ g/L, $C_p = 0.239$ J/(g·K), $\Delta H = -5 \times 10^4$ J/mol, $E/R = 8750$ K, $k_0 = 7.2 \times 10^{10}$ min^{-1}, and $U \cdot A = 5{\times}10^4$ J/(min·K). The nominal operating conditions are $q = 100$ L/min, $T_f = 350$ K, and $V = 100$ L. The steady-state operating condition for the nominal model is $C_{A,ss} = 0.5$ mol/L, $T_{ss} = 350$ K, $T_{c,ss} = 300$ K, and $C_{Af,ss} = 1.0$ mol/L. We define the states of the system in deviation form, i.e., $\mathbf{x} = [(C_A - C_{A,ss}), (T - T_{ss})]^\top$. The manipulated inputs $\mathbf{u} = [u_1, u_2]^\top$ specify the coolant temperature $T_c = T_{c,ss} + 50 \cdot u_1$ and the feed concentration $C_{Af} = C_{Af,ss} + 0.5 \cdot u_2$. To convert (12) to a discrete-time system of the form (1), we discretize it at a sampling time of 0.03 min and incorporate an additive process disturbance term $\mathbf{w}_k$ to the right-hand side.

We assume the states are directly measured such that $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k, i_k) = \mathbf{x}_k + \mathbf{v}_k$ for all models. The uncertainty sets (2) in this case are defined as follows

$$X_0 = \left\{ \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.05 \end{bmatrix} \xi : \xi \in \mathbb{R}^2,\ \|\xi\|_\infty \leq 1 \right\},$$
$$W = \left\{ \begin{bmatrix} 0.05 & 0 \\ 0 & 1 \end{bmatrix} \xi : \xi \in \mathbb{R}^2,\ \|\xi\|_\infty \leq 1 \right\},$$
$$V = \{[0, 0]^\top\}.$$

The state constraints are $X = \mathbb{R}^2$, bounded input constraints are $U = \{\mathbf{u} \in \mathbb{R}^2 : \|\mathbf{u}\|_\infty \leq 1\}$, and input cost function is $J(\tilde{\mathbf{u}}_{N-1}) = \|\tilde{\mathbf{u}}_{N-1}\|_2$. We assume a horizon of $N = 3$. In addition to the nominal system model described above, we consider three fault models: (i) fault resulting in loss of ability to manipulate cooling water such that $T_c = T_{c,ss} + 0 \cdot u_1$; (ii) fault resulting in loss of ability to control feed concentration such that $C_{Af} = C_{Af,ss} + 0 \cdot u_2$; and (iii)



(a) Exact solution

(b) Proposed active learning

(c) Traditional active learning

(d) Passive learning

Figure 6: The final output reachable sets and 100,000 randomly sampled outputs from the nominal and faulty models for the $N = 4$ linear model case study using the input profiles found using (a) the exact solution, (b) proposed active learning method, (c) traditional active learning method, and (d) passive learning method. Models 1, 2, 3, 4, and 5 correspond to dark blue, green, light blue, yellow, and red, respectively.

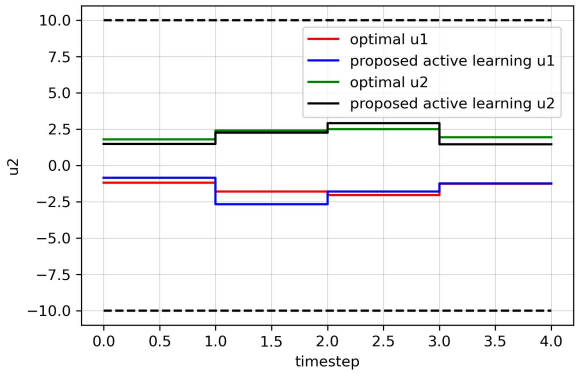


Figure 7: Best found (minimum cost) separating input profiles for $N = 4$ linear model case study using the exact optimal solution (first and second element of $\mathbf{u}$ in red and green, respectively) and our proposed active learning method (first and second element of $\mathbf{u}$ in blue and black, respectively).

fouling on the reactor walls that causes an increase in the heat transfer coefficient $U \cdot A = 6 \times 10^4$ J/(min·K).

Due to the highly nonlinear structure of the dynamics in (12), there is no known method that can provide even an approximate solution to (4). In fact, the subproblem of just calculating the reachable sets cannot be carried out exactly. As such, we use the CORA toolbox [2] to identify relatively tight outer-approximation of these reachable sets, as described in Section 3.3. Again, we only require separation of the full output reachable sets at the final time step for easier illustration. The cost of the best found separating input for each algorithm as a function of the number of learning iterations is shown in Figure 8. Similar to the previous case, our method significantly outperforms traditional passive and active learning, achieving a nearly 2x reduction in input cost (compared to the best found solutions with traditional active and passive learning over 30 iterations) within only 15 iterations. To demonstrate the value of our method, we also plot the CORA-computed output reachable sets and 100,000 Monte Carlo samples of the outputs over the considered time horizon by injecting a constant steady-state input profile (left column) and the optimal separating input found by our method (right column) in Figure 9. We see that, given a constant steady-state input, most of the faults yield reachable

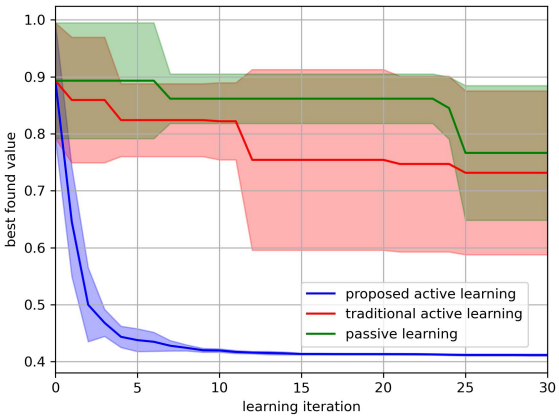


Figure 8: Performance of our proposed algorithm versus traditional active and passive learning on the CSTR case study. All algorithms are repeated 10 times from different random seeds; the solid line depicts average performance while the shaded region corresponds to ±1 standard deviation.

sets that overlap such that we cannot uniquely diagnose if a fault has occurred or not. The optimal separating input found by our method, on the other hand, yields enough perturbation to uniquely identify the fault, without causing the system to deviate too much from the steady-state condition.

4.4. Complex Industrial CSTR

Lastly, we consider a more complex (industrially-relevant) nonlinear CSTR adapted from [25]. This process is an exothermic irreversible set of reactions $A \rightarrow B \rightarrow C$ and $A \rightarrow D$ with a constant liquid volume where B is the desired product. The model is described by four state variables as follows

$$\frac{\mathrm{d}C_A}{\mathrm{d}t} = F(C_{A0} - C_A) - k_1 C_A - k_3 C_A^2, \tag{13a}$$

$$\frac{\mathrm{d}C_B}{\mathrm{d}t} = -FC_B + k_1 C_A - k_2 C_B, \tag{13b}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = F(T_0 - T) + \frac{k_W A_R}{\rho C_p V_R}(T_k - T) - \frac{k_1 C_A \Delta H_R^{AB} + k_2 C_B \Delta H_R^{BC} + k_3 C_A^2 \Delta H_R^{AD}}{\rho C_p}, \tag{13c}$$

$$\frac{\mathrm{d}T_k}{\mathrm{d}t} = \frac{1}{m_k C_{pk}}[\dot{Q}_k K_W A_R (T - T_k)], \tag{13d}$$

where C_A and C_B are, respectively, the concentration of A and B in the reactor and T and T_k are, respectively, the temperature of the reactor and the coolant. The temperature-dependent rate coefficients follow an Arrhenius law of the form

$$k_i(T) = k_{0i} \exp\left(\frac{-E_{Ai}}{R(T + 273.15)}\right). \tag{14}$$

Note that, for this system, the first and second reaction rate coefficients are equal, i.e., $k_1 = k_2$. The complete set

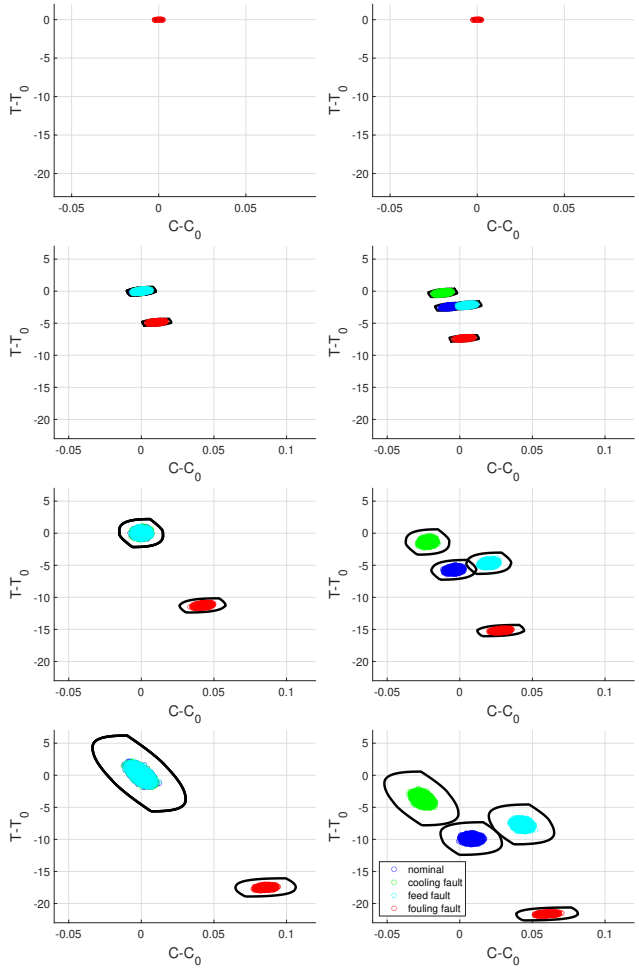


Figure 9: Outer-approximated output reachable sets and 100,000 randomly sampled outputs from the nominal and faulty models for the CSTR case study over $N = 3$ time horizon. The left column corresponds to keeping the input at their nominal value wherein we see multiple faults overlapping for all times. The right column corresponds to injecting the best separating input sequence found by our proposed method wherein we see clear separation and thus a guaranteed fault diagnosis.

of model parameters including the steady-state operating conditions are summarized in Table 1.

We define the states of the system in deviation form, i.e., $\mathbf{x} = [(C_A - C_{A,ss}), (C_B - C_{B,ss}), (T - T_{ss}), (T_k - T_{k,ss})]^\top$. The manipulated inputs $\mathbf{u} = [u_1, u_2]^\top$ specify the inlet flowrate $F = F_{ss} + 2.137u_1$ and the cooling rate $\dot{Q}_k = \dot{Q}_{k,ss} + 4105.9u_2$. To covert (13) to a discrete-time system of the form (1), we discretize it at a sampling time of 0.16 hours. Following the original work [25], $\mathbf{w} = [w_1, w_2, w_3]^\top$ act as time-varying uncertain parameters with $K_{0_1} = \bar{K}_{0_1} + (0.04 \times 10^{12})w_1$, $K_{0_2} = \bar{K}_{0_2} + (0.04 \times 10^{12})w_2$, and $K_{0_3} = \bar{K}_{0_3} + (0.27 \times 10^{9})w_3$ where $\bar{K}_{0_1}$, $\bar{K}_{0_2}$, and $\bar{K}_{0_3}$ are the nominal parameter values reported in Table 1. We assume only C_A and T are measured in this problem. The uncertainty

Table 1
Model parameters and steady state values for the industrially-relevant CSTR case study

Model Parameters		Nominal/Steady State
$K_{0_{1,2}} = 1.287 \times 10^{12}\,\mathrm{h}^{-1}$	$K_{0_3} = 9.043 \times 10^{9}\,\mathrm{l/molA \cdot h}$	$C_{Ass} = 3.712\,\mathrm{mol/l}$
$E_{A_{1,2}}/R = 9758.3\,\mathrm{K}$	$E_{A_3}/R = 8560.0\,\mathrm{K}$	$C_{Bss} = 0.60\,\mathrm{mol/l}$
$\Delta H_R^{AB} = 4.2\,\mathrm{kJ/molA}$	$\Delta H_R^{BC} = -11.0\,\mathrm{kJ/molB}$	$T_{ss} = 339.1\,\mathrm{K}$
$\Delta H_R^{AD} = -41.85\,\mathrm{kJ/molA}$	$\rho = 0.9342\,\mathrm{kg/l}$	$T_{k,ss} = 334.0\,\mathrm{K}$
$C_p = 3.01\mathrm{kJ/kg \cdot K}$	$C_{pk} = 2.0\mathrm{kJ/kg \cdot K}$	$F_{ss} = 2.137\ \mathrm{l/h}$
$A_R = 0.215\mathrm{m}^2$	$V_R = 10.01\ \mathrm{l}$	$\dot{Q}_{k,ss} = -4394.1\mathrm{kJ/h}$
$m_k = 5.0\mathrm{kg}$	$T_{in} = 403.15\mathrm{K}$	$C_{A0} = 5.1\mathrm{mol/l}$
$K_w = 4032.0\mathrm{kJ/h \cdot m}^2 \cdot \mathrm{K}$		

sets (2) in this case are defined as follows

$$X_0 = \left\{ \begin{bmatrix} 0.6 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 \\ 0 & 0 & 3.39 & 0 \\ 0 & 0 & 0 & 3.34 \end{bmatrix} \xi : \|\xi\|_\infty \leq 1 \right\},$$
$$W = \left\{ \xi \in \mathbb{R}^3 : \|\xi\|_\infty \leq 1 \right\},$$
$$V = \{[0,0]^\top\}.$$

The state constraints are $X = \mathbb{R}^4$, the inputs must satisfy $U = \{\mathbf{u} \in \mathbb{R}^2 : \|\mathbf{u}\|_\infty \leq 1\}$, and the input cost function is $J(\tilde{\mathbf{u}}_{N-1}) = \|\tilde{\mathbf{u}}_{N-1}\|_2$. We assume a horizon of $N = 5$. In addition to the nominal system model described above, we consider four fault models: (i) fault resulting in loss of ability to manipulate inlet flow such that $F = F_{ss} + 0 \cdot u_1$; (ii) fault resulting in loss of ability to control cooling rate such that $\dot{Q}_k = \dot{Q}_{k,ss} + 0 \cdot u_2$; (iii) high feed concentration with $C_{A0} = 5.7$ mol/l; and (iv) low feed concentration with $C_{A0} = 4.5$ mol/l.

Similarly to the previous example, the highly nonlinear structure of the dynamics in (13) means that no existing method can be provide an exact solution to (4) so we use the CORA toolbox [2] to identify relatively tight outer-approximation of these reachable sets, as described in Section 3.3. The cost of the best found separating input for our proposed method, traditional active learning, and passive learning as a function of the number of learning iterations is shown in Figure 10. We only ran a single replicate due to the cost of running the different methods. As in the previous cases, our method significantly outperforms traditional passive and active learning, achieving a more than a 2x reduction in input cost within only ~ 10 iterations. We also plot the final states of CORA-computed output reachable sets and 100,000 Monte Carlo samples of the outputs for each learning method in Figure 11. We see that the proposed method results in much less deviation from the normal operating conditions for all possible models. The resulting input sequence could then be used to obtain a complete, high-quality fault diagnosis with minimal disruption to the plant operation.

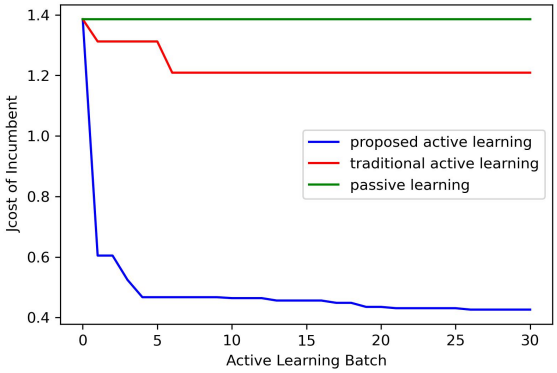


Figure 10: Performance of our proposed algorithm versus traditional active and passive learning on the industrially-relevant CSTR case study. All algorithms were ran a single time from the same random seed.

5. Conclusions

This paper presents a novel approach for near-optimal guaranteed active fault diagnosis of constrained nonlinear systems subject to bounded time-invariant and time-varying uncertainty. The main assumption about the system dynamics is that we are able to query an oracle that determines if the reachable output sets do not overlap for all fault models and the reachable state sets do not intersect with any unsafe regions of the state space. Even though such an oracle is not always directly available, we describe how upper and lower bounds on this oracle can be computed using state-of-the-art reachability algorithms. Since the oracle can be computationally expensive to evaluate, we propose an active learning approach for sequentially identifying the most informative input sequences to query. These inputs are designed in a way to tradeoff exploration and exploitation, so that we efficiently learn minimally invasive inputs that guarantee a safe fault diagnosis is achieved over a finite time horizon. We demonstrate the power of our approach on three case studies including an industrially-relevant nonlinear continuously stirred tank reactor system. Our results show that our approach can consistently identify high-quality separating input sequences given a limited computational budget.

There are several promising directions for future work. First, conducting a theoretical analysis of the method could provide a deeper understanding of its convergence properties, offering guidance on optimizing parameters such as batch size. Another important direction is the explicit

integration of this method with digital twins (high-fidelity plant emulators), enabling more comprehensive exploratory analyses of large sets of fault scenarios. Finally, while our approach currently focuses on either outer- or inner-reachable set approximations, a potential improvement would be to incorporate both sources of information into the active learning process, which could reduce performance gaps and enhance applicability in certain contexts.

A. Appendix: Implementation Guidelines

In this appendix, we provide implementation guidelines for each step of our proposed approach, including our envisioned problem setup, as well as how the results can be validated and deployed in practice. These guidelines are not intended to be exhaustive, but rather to help readers understand the motivation behind certain choices and facilitate building upon the results presented in this work. Additionally, we reiterate that the complete code for all examples is available on GitHub: https://github.com/PaulsonLab/Guaranteed_AFD_ADL, which we recommend referencing for a complete, line-by-line implementation.

A.1. Problem setup

When utilizing our proposed approach, the first step is to specify the relevant system models for the nominal and faulty versions of the system of interest, shown mathematically in (1). This requires specifying the state transition and output functions as well as the uncertainty sets in (2) and the state and input constraint sets in (3). The user has complete freedom in the choice of all of these parameters, along with the specific fault scenarios $\tilde{\mathbb{I}}$ of interest (particular sequence of model indices). A key source of motivation for efficiently solving (4) for optimal separating inputs is that, in practice, we imagine a user to want to solve this problem repeatedly under different choices of these parameters. For example, one may want to see the impact of increasing the number of fault scenarios or relaxing the state or input constraint sets.

A.2. Oracle selection

After defining a specific problem instance, the next step is to select an 'oracle,' as described in Section 3.3, which can compute outer-approximated reachable sets for each model. While our approach is agnostic to the choice of reachability tool, we recommend using CORA [2] due to its flexibility. However, note that this requires users to implement their models in CORA's language, which may involve some effort. Additionally, there are internal hyperparameters in CORA that users must configure, such as the time step and the choice of set representation. In our case studies, we largely used CORA's default settings, but we advise users to test a few random input sequences to ensure that the selected parameters do not result in overly conservative approximations. If significant conservatism is observed, adjusting the settings or exploring alternative methods may help prevent suboptimal results.

A.3. Initialization

An important input to our overall proposed approach (Algorithm 1) is the initial labeled training data $\mathcal{L}_0$. Since we do not assume any prior knowledge about how to sample the input space $\tilde{U}_N$, we recommend starting with quasi random (space filling) Sobol samples, typically with a number that roughly scales with D where $D = n_u N$ is the total number of inputs over the horizon N. We have found that $\max\{10D, 512\}$ works fairly well in practice. The most critical component of $\mathcal{L}_0$ is that it should not be too heavily imbalanced to one class or the other (0 or 1), i.e., there should be a reasonable mix of both positive and negative classes. If this is not the case after the first round of samples, we recommend generating an additional set of samples of the same size. If $\mathcal{L}_0$ has almost all 0 class labels still, it is likely that $\tilde{U}_N$ is too small such that we recommend doubling the size of $\tilde{U}_N$ until at least a reasonable fraction (5%-10%) is from the +1 class.

A.4. Algorithm 1

The main things to consider when implementing Algorithm 1 is the choice of batch size B and the DNN architecture and training procedure. We found that very small batch sizes often do not work well as they do not fully take into account the diversity of information provided by the DNN. As such, we recommend batch sizes of at least 8 in all situations. However, we expect that larger batch sizes might be needed for high-dimensional problems due to more possible interactions. Thus, similarly to the initialization, we would recommend a batch size that eventually scales roughly linear in $D = Nn_u$. We did not optimize the DNN architecture or optimization process; we found that a standard shrinking structure with 3 hidden layers, 20 nodes in the first layer, 13 in the second layer, 7 in the third layer, and a linear output layer (with ReLU activation functions throughout) consistently worked in all of our case studies. We also used the vanilla Adam optimizer with a fairly standard learning rate of 8×10^{-4}. There are a wide-variety of opportunities for further DNN optimization, as described in Remark 3.

A.5. Algorithm 2

The key tunable parameter in Algorithm 2 is the number of optimization-based samples, denoted by b. As discussed in the text, we found that setting $b = 4$ with corresponding probability levels $\epsilon_1 = 0.3$, $\epsilon_2 = 0.4$, $\epsilon_3 = 0.5$, and $\epsilon_4 = 0.6$ worked well across all of our examples. Our tests showed that very small or large values of ϵ almost always led to reduced performance, which is intuitive – extreme values of ϵ (near 1 or 0) suggest a high confidence in predicting the true class of the input sequence, contributing little useful information for future training steps. The choice of $b = 4$ was primarily due to the fact that nearby values of ϵ tend to yield similar input values. Since increasing b further did not significantly improve performance, we opted to fix it at 4.

A.6. Validation

To find optimal separating inputs, Algorithm 1 should be run iteratively until either the predefined oracle sample

budget is exhausted or progress stagnates after several iterations. Upon exiting the algorithm, we recommend selecting the best (lowest-cost) separating input for validation. One validation approach is to apply this input sequence to all possible fault scenarios, using a large number of Monte Carlo samples to account for uncertainties. This helps verify that the outputs generated by different models do not overlap. Since this step involves simulation rather than optimization, it can be executed much more efficiently. By running these tests, you can quickly assess the robustness of the input sequence across multiple scenarios, ensuring it is suitable for deployment in real-world systems.

A.7. Deployment

The final step is to deploy the validated separating input sequence on the real-world system when necessary, as described in Remark 2. In practice, users would pre-generate and store multiple low-energy separating inputs tailored to different potential fault scenarios. These inputs would be readily available for use in online settings. When an issue is detected – such as through established passive fault detection methods – a stored separating input can be applied to the system, enhancing the diagnosability of specific faults based on the operator's assessment of the plant's current status. The exact choice of which separating input to use will vary with each application. This proactive approach enables rapid response to evolving system conditions, helping minimizing downtime and ensuring continued operational safety.

B. Appendix: Illustration of Proposed Method

Here, we provide a visual illustration of our proposed method (Algorithms 1–2) on the illustrative linear system in Section 4.1. Figure B.1 shows the evolution of the proposed active learning process over iterations $t = 0, 1 \dots, 8$ with a batch size of $B = 8$. The candidates selected in Algorithm 2 using the expected information gain (Line 7) and the optimization method (Line 9) are shown with blue dots and black dots, respectively. The inputs predicted by the DNN model to lead and not to lead to separation of the output reachable sets are shown in green and red, respectively.

C. Appendix: Demonstration of Method with Safety Constraints

In this Appendix, we show results for a variation of the larger-scale linear system introduced in Section 4.2 that also includes safety constraints. In particular, we incorporate constraints on the output reachable set of the form

$$\Phi_k(\tilde{u}_N, \tilde{i}_N) \subseteq Y_{\text{safe}}, \quad \forall \tilde{i}_N \in \tilde{\mathbb{I}}, \quad \forall k \in \{0, \dots, N\},$$

where

$$Y_{\text{safe}} = \{\mathbf{y} = (y_1, y_2) \in \mathbb{R}^2 : y_2 \leq 1\},$$

is the set of allowable outputs, i.e., all outputs for all models must belong to Y_{safe} despite uncertainty. The best found feasible separating input sequence for each algorithm as a function of learning iterations over 10 replicates is shown in Figure C.1. Traditional passive and active learning barely make any improvement over the starting best value, while our proposed method reduces the input cost from ~ 1.2 to < 0.7 on average. We also plot the output reachable sets and 100,000 Monte Carlo samples of the outputs at the final time $N = 4$ when injecting the best found separating input under representative run of our proposed method in Figure C.2. As expected, our method finds an input that results in the output reachable sets for all models being disjoint (guaranteeing a complete fault diagnosis); however, these sets also satisfy the desired safety constraints in this case. A significantly different input signal is needed to ensure Model 5 does not enter the unsafe zone, which is easily identified by our approach.

CRediT authorship contribution statement

Nathaniel Massa: Methodology, Software, Writing - Original draft preparation. **Joel A. Paulson:** Conceptualization of this study, Methodology, Data curation, Writing - Original draft preparation.

References

[1] Althoff, M., Frehse, G., Girard, A., 2021. Set propagation techniques for reachability analysis. Annual Review of Control, Robotics, and Autonomous Systems 4, 369–395.

[2] Althoff, M., Kochdumper, N., 2016. CORA 2016 manual. TU Munich .

[3] Amini, N., Zhu, Q., 2022. Fault detection and diagnosis with a novel source-aware autoencoder and deep residual neural network. Neurocomputing 488, 618–633.

[4] Asarin, E., Dang, T., Girard, A., 2003. Reachability analysis of nonlinear systems using conservative approximation, in: International Workshop on Hybrid Systems: Computation and Control, Springer. pp. 20–35.

[5] Ashari, A.E., Nikoukhah, R., Campbell, S.L., 2012. Active robust fault detection in closed-loop systems: Quadratic optimization approach. IEEE Transactions on Automatic Control 57, 2532–2544.

[6] Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.L., et al., 2023. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 13, e1484.

[7] Bonzanini, A.D., Paulson, J.A., Makrygiorgos, G., Mesbah, A., 2022. Scalable estimation of invariant sets for mixed-integer nonlinear systems using active deep learning, in: Proceedings of the Conference on Decision and Control, IEEE. pp. 3431–3437.

[8] Bravo, J.M., Alamo, T., Camacho, E.F., 2006. Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. Automatica 42, 1745–1751.

[9] Campbell, S.L., Horton, K.G., Nikoukhah, R., 2002. Auxiliary signal design for rapid multi-model identification using optimization. Automatica 38, 1313–1325.

[10] Cen, Z., Noura, H., Susilo, T.B., Younes, Y.A., 2014. Robust fault diagnosis for quadrotor UAVs using adaptive Thau observer. Journal of Intelligent & Robotic Systems 73, 573–588.

[11] Chakrabarty, A., Danielson, C., Di Cairano, S., Raghunathan, A., 2020. Active learning for estimating reachable sets for systems with unknown dynamics. IEEE Transactions on Cybernetics 52, 2531–2542.

[12] Chang, T., Liu, T., Ma, X., Wu, Q., Wang, X., Cheng, J., Wei, W., Zhang, F., Liu, H., 2024. Fault detection in industrial wastewater treatment processes using manifold learning and support vector data description. Industrial & Engineering Chemistry Research .

[13] Chiang, L.H., Russell, E.L., Braatz, R.D., 2000. Fault Detection and Diagnosis in Industrial Systems. Springer Science & Business Media.

[14] Dong, D., McAvoy, T.J., 1996. Batch tracking via nonlinear principal component analysis. AIChE Journal 42, 2199–2208.

[15] Edwards, C., Simani, S., et al., 2019. Fault diagnosis and fault-tolerant control in aerospace systems. International Journal of Robust and Nonlinear Control 29, 5291–5292.

[16] Elsken, T., Metzen, J.H., Hutter, F., 2019. Neural architecture search: A survey. Journal of Machine Learning Research 20, 1–21.

[17] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

[18] Heirung, T.A.N., Mesbah, A., 2019. Input design for active fault diagnosis. Annual Reviews in Control 47, 35–50.

[19] Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration, in: International Conference on Learning and Intelligent Optimization, Springer. pp. 507–523.

[20] Jiang, B., Braatz, R.D., 2017. Fault detection of process correlation structure using canonical variate analysis-based correlation features. Journal of Process Control 58, 131–138.

[21] Jiang, B., Zhu, X., Huang, D., Paulson, J.A., Braatz, R.D., 2015. A combined canonical variate analysis and fisher discriminant analysis (cva–fda) approach for fault diagnosis. Computers & Chemical Engineering 77, 1–9.

[22] Jiang, Q., Jia, M., Hu, J., Xu, F., 2009. Machinery fault diagnosis using supervised manifold learning. Mechanical Systems and Signal Processing 23, 2301–2311.

[23] Kesavan, P., Lee, J.H., 2001. A set based approach to detection and isolation of faults in multivariable systems. Computers & Chemical Engineering 25, 925–940.

[24] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[25] Klatt, K.U., Engell, S., 1998. Gain-scheduling trajectory control of acontinuous stirred tank reactor. Computers & Chemical Engineering 22, 491–502.

[26] Kochdumper, N., Althoff, M., 2020. Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems, in: Proceedings of the Conference on Decision and Control, IEEE. pp. 2130–2137.

[27] Li, M., Yu, D., Chen, Z., Xiahou, K., Ji, T., Wu, Q., 2018. A data-driven residual-based method for fault diagnosis and isolation in wind turbines. IEEE Transactions on Sustainable Energy 10, 895–904.

[28] Lin, Y., Stadtherr, M.A., 2008. Fault detection in nonlinear continuous-time systems with uncertain parameters. AIChE Journal 54, 2335–2345.

[29] Lu, W., Li, Y., Cheng, Y., Meng, D., Liang, B., Zhou, P., 2018. Early fault detection approach with deep architectures. IEEE Transactions on Instrumentation and Measurement 67, 1679–1689.

[30] MacGregor, J.F., Jaeckle, C., Kiparissides, C., Koutoudi, M., 1994. Process monitoring and diagnosis by multiblock PLS methods. AIChE Journal 40, 826–838.

[31] MacGregor, J.F., Kourti, T., 1995. Statistical process control of multivariate processes. Control Engineering Practice 3, 403–414.

[32] Magni, L., Nijmeijer, H., Van Der Schaft, A., 2001. A receding–horizon approach to the nonlinear H∞ control problem. Automatica 37, 429–435.

[33] Mao, A., Mohri, M., Zhong, Y., 2023. Cross-entropy loss functions: Theoretical analysis and applications, in: International Conference on Machine Learning, PMLR. pp. 23803–23828.

[34] Md Nor, N., Che Hassan, C.R., Hussain, M.A., 2020. A review of data-driven fault detection and diagnosis methods: Applications in chemical process systems. Reviews in Chemical Engineering 36, 513–553.

[35] Nikoukhah, R., 1998. Guaranteed active failure detection and isolation for linear dynamical systems. Automatica 34, 1345–1358.

[36] Olivier-Maget, N., Hetreux, G., Le Lann, J.M., Le Lann, M.V., 2009. Model-based fault diagnosis for hybrid systems: Application on chemical processes. Computers & Chemical Engineering 33, 1617–1630.

[37] Pardeshi, K., Shaikh, J.A., Liyakat, K.S., 2022. Implementation of fault detection framework for healthcare monitoring system using iot, sensors in wireless environment. Telematique 21, 5451–5460.

[38] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems 32.

[39] Patton, R.J., Chen, J., 1997. Observer-based fault detection and isolation: Robustness and applications. Control Engineering Practice 5, 671–682.

[40] Paulson, J.A., Heirung, T.A.N., Braatz, R.D., Mesbah, A., 2018. Closed-loop active fault diagnosis for stochastic linear systems, in: 2018 Annual American Control Conference (ACC), IEEE. pp. 735–741.

[41] Paulson, J.A., Martin-Casas, M., Mesbah, A., 2017. Input design for online fault diagnosis of nonlinear systems with stochastic uncertainty. Industrial & Engineering Chemistry Research 56, 9593–9605.

[42] Paulson, J.A., Mesbah, A., 2019. An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems. International Journal of Robust and Nonlinear Control 29, 5017–5037.

[43] Paulson, J.A., Raimondo, D.M., Findeisen, R., Braatz, R.D., Streif, S., 2014. Guaranteed active fault diagnosis for uncertain nonlinear systems, in: Proceedings of the European Control Conference, IEEE. pp. 926–931.

[44] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[45] Raimondo, D.M., Marseglia, G.R., Braatz, R.D., Scott, J.K., 2016. Closed-loop input design for guaranteed fault diagnosis using set-valued observers. Automatica 74, 107–117.

[46] Rungger, M., Zamani, M., 2018. Accurate reachability analysis of uncertain nonlinear systems, in: Proceedings of the International Conference on Hybrid Systems: Computation and Control, pp. 61–70.

[47] Scott, J.K., Findeisen, R., Braatz, R.D., Raimondo, D.M., 2013a. Design of active inputs for set-based fault diagnosis, in: Proceedings of the American Control Conference, IEEE. pp. 3561–3566.

[48] Scott, J.K., Findeisen, R., Braatz, R.D., Raimondo, D.M., 2014. Input design for guaranteed fault diagnosis using zonotopes. Automatica 50, 1580–1589.

[49] Scott, J.K., Marseglia, G.R., Magni, L., Braatz, R.D., Raimondo, D.M., 2013b. A hybrid stochastic-deterministic input design method for active fault diagnosis, in: Proceedings of the Conference on Decision and Control, IEEE. pp. 5656–5661.

[50] Sun, W., Paiva, A.R., Xu, P., Sundaram, A., Braatz, R.D., 2020. Fault detection and identification using bayesian recurrent neural networks. Computers & Chemical Engineering 141, 106991.

[51] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., Yin, K., 2003a. A review of process fault detection and diagnosis: Part III: Process history based methods. Computers & Chemical Engineering 27, 327–346.

[52] Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N., 2003b. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Computers & Chemical Engineering 27, 293–311.

[53] Wetzlinger, M., Kochdumper, N., Bak, S., Althoff, M., 2023. Fully automated verification of linear systems using inner-and outer-approximations of reachable sets. IEEE Transactions on Automatic Control .

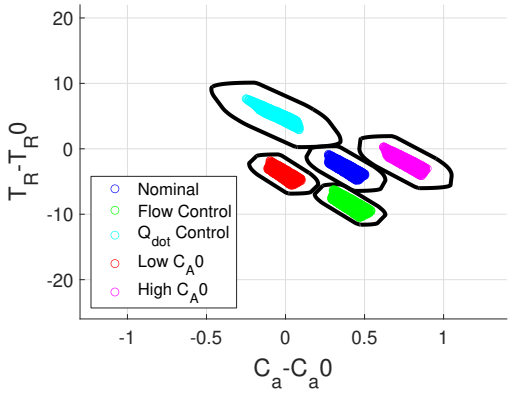
[54] Wetzlinger, M., Kulmburg, A., Althoff, M., 2021. Adaptive parameter tuning for reachability analysis of nonlinear systems, in: Proceedings

of the International Conference on Hybrid Systems: Computation and Control, pp. 1–11.

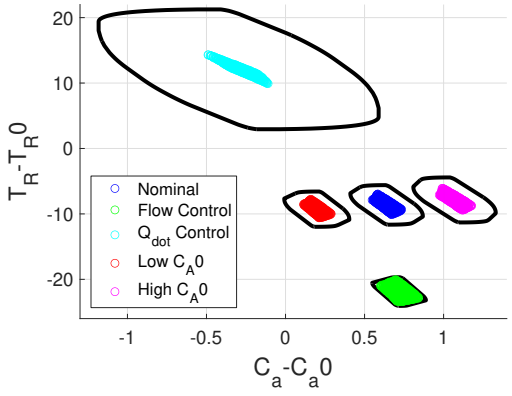
[55] Yang, L., Shami, A., 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing 415, 295–316.

Nathaniel Massa is a second-year PhD graduate student in Chemical Engineering at The Ohio State University currently being advised by Dr. Joel Paulson. His research interest are in the areas of fault detection and diagnosis and physics-informed machine learning. He received his B.S. in Chemical Engineering from the University of Iowa in 2023. During his undergraduate studies Nate worked in a high atmosphere aerosol experimental laboratory under Dr. Stanier working with an oxidation flow reactor to simulate the oxidative conditions of the upper atmosphere and their effects on cyclic sioloxanes.

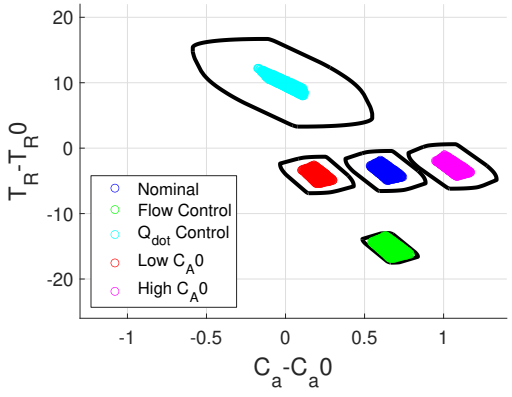
Joel Paulson is the H.C. Slip "Slider" Assistant Professor of Chemical and Biomolecular Engineering at The Ohio State University (OSU) where he is also a core faculty member of the Sustainability Institute (SI) and an affiliate of the Translational Data Analytics Institute (TDAI). He joined OSU in 2019 after completing his Ph.D. at the Massachusetts Institute of Technology (MIT) in Chemical Engineering and a subsequent postdoctoral appointment at the University of California, Berkeley in the area of systems and control theory. He has received several awards including the NSF CAREER Award, the Best Application Paper Prize from the 2020 IFAC World Congress, and the OSU Lumley Research Award. His research interests are mainly in the areas of data-driven optimization, physics-informed machine learning, and model predictive control.



(a) Proposed Active Learning



(b) Traditional Active Learning



(c) Passive Learning

Figure 11: Outer-approximated output reachable sets and 100,000 randomly sampled outputs from the nominal and faulty models for the CSTR case study at the final time step of the complex industrial CSTR case study for input trajectories found by (a) our proposed method, (b) traditional active learning and (c) traditional passive learning. We see our method is able to find a much lower energy input trajectory that still results in full separation of the reachable sets for all models.

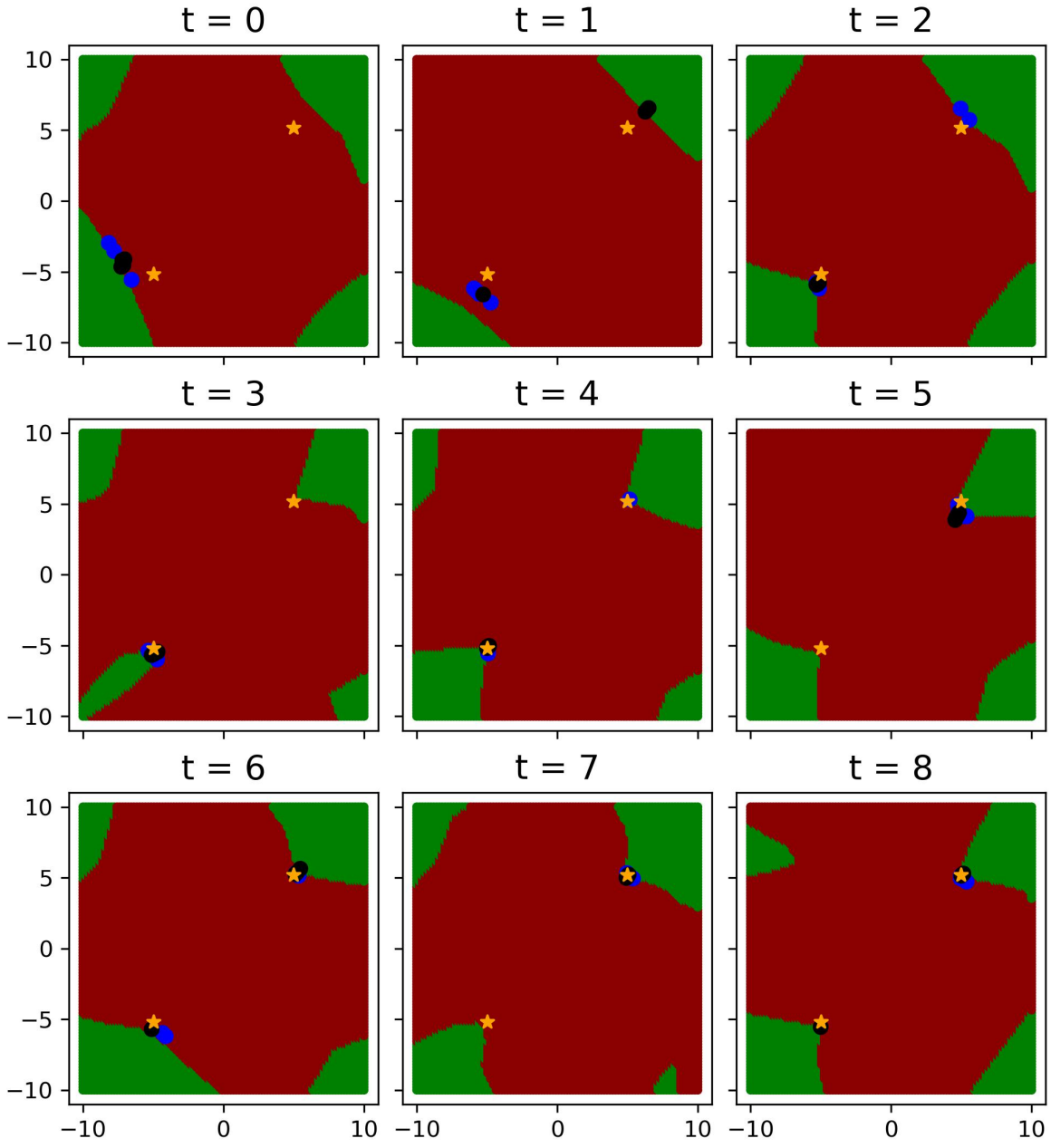


Figure B.1: Illustration of the first 9 iterations ($t = 0, 1, \ldots, 8$) of our proposed active learning method in Algorithms 1–2. Input values predicted by the DNN to (not) be separating inputs are shown in (red) green. The optimal separating input values are shown with yellow markers. The candidates selected in Algorithm 2 using the expected information gain (Line 7) and the optimization method (Line 9) are shown with blue dots and black dots, respectively.

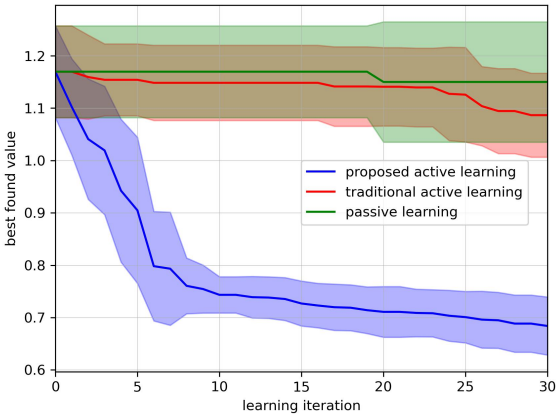


Figure C.1: Performance of our proposed algorithm versus traditional active and passive learning on 8d version of linear problem with output safety constraints incorporated. All algorithms are repeated 10 times from different random seeds; the solid line depicts average performance while the shaded region corresponds to ± 1 standard deviation.

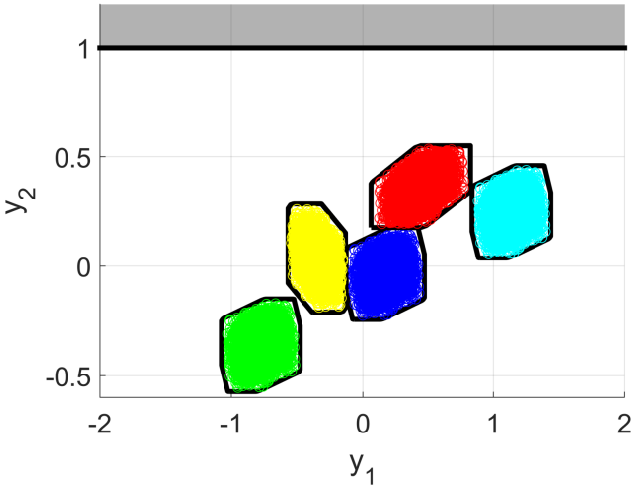


Figure C.2: The final output reachable sets and 100,000 randomly sampled outputs from the nominal and faulty models for the 8d version of the linear model case study ($N = 4$) with output safety constraints incorporated. The shaded gray region corresponds to the unsafe zone that is the complement of safety region Y_{safe}. Models 1, 2, 3, 4, and 5 correspond to dark blue, green, light blue, yellow, and red, respectively.