Model-Free Neural Fault Detection and Isolation for Safe Control

Kunal Garg, *Member, IEEE*, Charles Dawson, *Graduate Student Member, IEEE*, Kathleen Xu, Melkior Ornik, *Senior Member, IEEE*, and Chuchu Fan, *Member, IEEE*

***Abstract*—A sudden actuator fault in a safety-critical system can cause safety violations and lead to severe consequences. Existing fault-tolerant control (FTC) approaches normally focus on maintaining system performance and do not consider system safety. Control Barrier Functions (CBFs) have emerged as useful tools from control theory for providing safety guarantees for control systems. However, existing applications of CBFs either do not consider actuator faults or only consider the special case where it is known which actuator is faulty or the case when redundant actuators are present to maintain controllability even under faults and failures. In this letter, we address the problem of safe recovery under a more realistic scenario where it is completely unknown which actuator is faulty and when the fault occurs. We develop a novel model-free learning framework for an output-based neural fault-detector that detects when a fault occurs and in which actuator. Based on the learned functions, we propose a switching framework for automatically detecting and recovering from faults. We evaluate our method on a case study involving a Crazyflie quadrotor with a motor failure.**

***Index Terms*—Fault detection, machine learning, neural networks.**

I. Introduction

SAFETY-CRITICAL systems are those where violation of safety constraints could result in loss of lives, significant property damage, or damage to the environment. In real-life applications, many cyber-physical control systems are safety-critical, including autonomous cars and aircraft. In this context, safe control requires finding a control policy that keeps the system within a predefined safe region at all times.

Manuscript received 10 May 2023; revised 11 July 2023; accepted 29 July 2023. Date of publication 7 August 2023; date of current version 21 August 2023. This work was supported in part by the NASA University Leadership Initiative under Grant 80NSSC22M0070, and in part by the National Science Foundation under Grant 2238030. The work of Charles Dawson was supported by NSF GRFP under Grant 1745302. The work of Kathleen Xu was supported by the Yao T. Li Fellowship from the Department of Aeronautics and Astronautics at MIT. Recommended by Senior Editor F. Dabbene. *(Corresponding author: Kunal Garg.)*

Kunal Garg, Charles Dawson, Kathleen Xu, and Chuchu Fan are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: kgarg@mit.edu; cbd@mit.edu; bfst@mit.edu; chuchu@mit.edu).

Melkior Ornik is with the Department of Aerospace Engineering, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: mornik@illinois.edu).

Digital Object Identifier 10.1109/LCSYS.2023.3302768

Designing and verifying safe control policies for complex autonomous systems is challenging because of the need to balance safety guarantees with other control objectives [1]. Control Barrier Functions (CBFs, [1], [2]) have been extensively used for certifying that a closed-loop system satisfies desired safety requirements. In recent years, CBF-based approaches have achieved promising results in many safety-critical control systems, ranging from self-driving cars [1] to aircraft [2], [3], [4].

Unfortunately, prior works on safe control using CBFs have paid little attention to the effects of actuator faults. While [2] proposes fault-tolerant control using CBFs, it is limited to systems with redundant actuators. Failures and faults without actuator redundancies have been studied in the field of fault-tolerant control (FTC), which has been applied extensively to applications such as aircraft [5], [6], and spacecraft attitude controls [7].

There is a plethora of work on fault-detection and identification (FDI); we refer the interested readers to the survey articles [8], [9], [10] that discuss various approaches of FDI used in the literature. In particular, the residual-based method has been used very commonly in prior work, where the expected output (under the commanded input and a known system model) and the actual output of the system are compared for fault detection. Such *residual* information requires the knowledge of the system model, and thus, is model-dependent. In this letter, we design a *model-free* FDI mechanism that only uses the actual output of the system and the commanded input to the system, and does not use the residual information. There is some work on using LSTM-based FDI, e.g., [11], [12], but it is limited to a very narrow class of faults. As noted in [13], prior work on neural network-based model-free FDI relies on the reconstruction of the model (e.g., [14]), or generating the residual information using Kalman filtering (see, e.g., [15]) or extended state-observers (see [16]). Koopman operator-based FDI techniques such as [14], [17] reconstruct a linear representation of the model for calculating the residual information. In such approaches, the approximations used for computing a finite-dimensional Koopman operator adversarially affect the performance of FDI. Another common data-driven approach of FDI is based on Principle Component Analysis (PCA), in particular, using Auto-associative neural network (AANN) [18], [19]). However, AANN-based methods are generally applicable for sensor-faults and its

fixed five-layer architecture limits its applications to higher dimensional systems. The method in [13] also estimates a reduced-order model of the system as an intermediate step. The main disadvantage of model-based FDI methods is that their performance can degrade significantly due to model uncertainties or imperfections in the used model for designing the FDI mechanism and the actual system model. To overcome this limitation, in this letter, we present a *truly* model-free approach, where we do not require to either learn the system model or create a reduced-order representation of the model. Instead, we use the system output and the commanded input as the features of a neural network, which directly predicts whether there is an actuator fault. We illustrate through numerical experiments that the model-free FDI mechanism performs at par (and even better in some cases than) the model-based mechanisms. We also illustrate the robustness of the proposed method against modeling uncertainties and demonstrate through numerical examples that while the performance of the model-based FDI mechanism drops significantly under model uncertainties, the performance of the designed model-free approach remains the same.

Notation: We denote by $\mathbb{R}$ and $\mathbb{R}_+$ the sets of real and nonnegative real numbers, respectively. $|x|$ denotes the Euclidean norm of a vector $x \in \mathbb{R}^n$. The Lie derivative of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ along a vector field $f : \mathbb{R}^n \to \mathbb{R}^m$ at a point $x \in \mathbb{R}^n$ is denoted as $L_f h(x) := \frac{\partial h}{\partial x}(x) f(x)$. A continuous function $\alpha : \mathbb{R}_+ \mapsto \mathbb{R}_+$ is class-$\mathcal{K}$ if $\alpha(0) = 0$ and α is strictly increasing.

II. Problem Formulation

We begin by considering a continuous-time nonlinear dynamical system of the form

$$\dot{x} = f(x) + g(x)u, \tag{1a}$$
$$y = \rho(x), \tag{1b}$$

for state $x \in \mathcal{X}$, control input $u \in \mathcal{U}$, with locally Lipschitz dynamics $f : \mathcal{X} \to \mathbb{R}^n$, $g : \mathcal{X} \to \mathbb{R}^{n\times m}$, and state and control sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$, respectively. Here, $\rho : \mathbb{R}^n \to \mathbb{R}^p$ denotes the output map of the system.

In this letter, we study the safety of $\mathcal{S}$ under actuator faults. Specifically, we consider an actuator fault occurring at some unknown time $t_f \geq 0$:

$$u(t,x) = \begin{cases} \pi(t,x) & \text{if} \quad t \leq t_F; \\ \text{diag}(\Theta)\ \pi(t,x) & \text{if} \quad t > t_F, \end{cases}, \tag{2}$$

where $\Theta = \{0,1\}^m \in \mathbb{R}^m$ is the vector denoting whether an actuator is faulty or not, and diag : $\mathbb{R}^m \to \mathbb{R}^{m\times m}$ maps a vector in $\mathbb{R}^m$ to a diagonal matrix in $\mathbb{R}^{m\times m}$. If the $i-$th actuator is faulty, then $\Theta_i = 0$ and the rest of each of the elements of Θ is 1. Another way to represent this model is:

$$u(t,x) = \pi(t,x) + \Delta u, \tag{3}$$

where $\Delta u_i = -\pi_i(t,x)$ and 0, otherwise. Here, the set of faulty signals can be collectively represented as $\Delta\mathcal{U} = \{\Delta u_1, \Delta u_2, \ldots, \Delta u_m\}$, where Δu_i represents the case when the $i-$th actuator is faulty. We can now state the problem studied in this letter. Consider system $\mathcal{S}$ with fault-model (2) for a given $\Delta\mathcal{U}$ and disjoint sets of safe and unsafe states $\mathcal{X}_{\text{safe}}, \mathcal{X}_{\text{unsafe}} \subseteq \mathcal{X}$, i.e., $\mathcal{X}_{\text{safe}} \cap \mathcal{X}_{\text{unsafe}} = \emptyset$. We assume that the fault signal is uniformly observable through the system output so that it is possible to detect the fault using system output. Given this context, we consider the following control synthesis problem:

Problem 1 (Fault-Tolerant Safe Control Synthesis Problem): Compute the largest possible subset $\mathcal{X}_0 \subset \mathcal{X}_{safe}$ and a control policy π such that the following safety property holds for all trajectories $x : \mathbb{R}_+ \to \mathbb{R}^n$ of the closed-loop dynamics under the policy π for all $\Delta u : \mathbb{R}_+ \to \Delta\mathcal{U}$ $x(0) \in \mathcal{X}_0 \implies x(t) \notin \mathcal{X}_{unsafe}\ \forall t \geq 0$.

Note that in some of the prior work (see, e.g., [20]), the safe and the unsafe regions are chosen as complementary sets, i.e., $\mathcal{X}_{safe} = \mathbb{R}^n \setminus \mathcal{X}_{unsafe}$. While it is possible to consider such a setup, we prefer a more general setup where there is a nonempty region $\mathcal{X} \setminus (\mathcal{X}_{safe} \cup \mathcal{X}_{unsafe})$. The reason for considering such a formulation is justified later in the letter, where we explain how it makes it easier to learn a CBF due to the presence of this *middle* region. We begin by reviewing the standard definition of CBFs in the fault-free case, then introduce our notion of fault-tolerant CBFs in the next section.

Definition 1 (Control Barrier Function (CBF) [1]): A function $h : \mathcal{X} \mapsto \mathbb{R}$ is a CBF for system $\mathcal{S}$ if there exists a class-$\mathcal{K}$ function α such that:

$$h(x) < 0\ \ \forall x \in \mathcal{X}_{unsafe}, \qquad h(x) \geq 0\ \ \forall x \in \mathcal{X}_{safe}, \tag{4}$$
$$\sup_{u\in\mathcal{U}} \{L_f h(x) + L_g h(x) u + \alpha(h(x))\} \geq 0\ \ \forall x \in \mathcal{X}_{safe}. \tag{5}$$

Given a CBF, we can define a set of admissible controls $K(x) = \{u \in \mathcal{U} \mid L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0\}$ so that any sufficiently smooth control input from this set is guaranteed to keep the system safe, per the following lemma.

Lemma 1 [1, Corollary 2]: If h is a CBF, then any locally Lipschitz continuous control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that $\pi(x) \in K(x)\ \forall x \in \mathcal{X}$ renders the closed-loop system $\mathcal{S}$ safe.

When $\mathcal{U}$ is a polytope, we can define a CBF-based controller using a Quadratic Program (QP) as in [21], where h is used to filter a nominal controller $\pi_{\text{nominal}} : \mathcal{D} \to \mathbb{R}^m$:

$$\pi_{\text{pre}}(x) = \underset{u\in\mathcal{U},\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2}\|u - \pi_{\text{nominal}}(x)\|_2^2 + \frac{1}{2}\alpha^2 \tag{6a}$$
$$\text{s.t. } L_f h(x) + L_g h(x) u \geq -\alpha\ h(x). \tag{6b}$$

Here, the class-$\mathcal{K}$ function is chosen as a linear function $\alpha(h(x)) = \alpha h(x)$, with α as a decision variable instead of being fixed to help with the feasibility of the QP [21]. In this architecture, the CBF filters the nominal controller to ensure safety even as the nominal controller pursues other objectives (e.g., reaching a goal location or tracking a reference trajectory). This approach is agnostic to the choice of π_{nominal} and guarantees to preserve safety under any choice of nominal controller [1]. The nominal controller can be designed to drive the system trajectories to a goal location $x_g \in \mathbb{R}^n$. In this letter, we use an LQR-based nominal controller, i.e., $\pi_{\text{nominal}}(x) := K_{\text{LQR}}(x - x_g)$, where the LQR gain matrix K_{LQR} is computed by linearizing the system (1) at the goal location x_g.

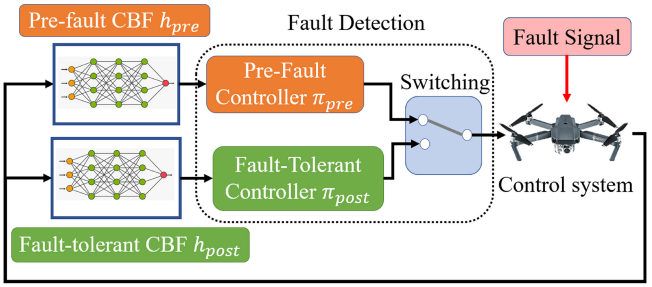

Fig. 1. Safe recovery using learned CBFs and a fault-detection mechanism.

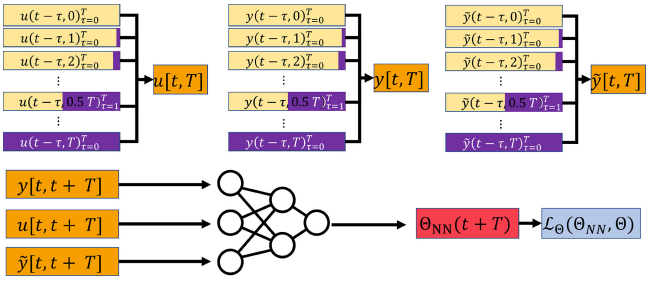

Fig. 2. General neural-network architecture for failure prediction. The training data includes all possible trajectories with different lengths of faulty input (the violet color represents the portion of the trajectory with faulty input).

Our approach to solving Problem 1 involves these steps:

1. Learn a fault-detector and a mechanism to switch the control policy from a pre-fault policy π_{pre} to a post-fault π_{post} in response.
2. Learn a *pre-fault* CBF h_{pre} such that the corresponding safety region $S_{pre} : \{x \mid h_{pre}(x) \geq 0\} \subset \mathcal{X}_{safe}$ and a fault-tolerant CBF h_{post} such that $S_{post} = \{x \mid h_{post} \geq 0\} \subseteq S_{pre}$ for the fault model (2) for a given $\Delta\mathcal{U}$.

First, we will combine these pre- and post-fault policies by deriving and proving the soundness of a CBF-based fault detection and switching mechanism. Then, we will provide the extension of the CBF theory to the fault-tolerant case and present our learning-based approach to finding pre- and post-fault CBFs h_{pre} and h_{post}, respectively. Finally, we will utilize these learned CBFs in a QP framework for synthesizing corresponding control policies. The resulting closed-loop system is depicted in Figure 1.

III. Neural Fault-Detection and Isolation

Model-free FDI: The faults must be detected correctly and promptly for the safe recovery of the system. We use a learning-based approach to design a fault-detection mechanism. Let $\Theta \in \{0,1\}^m$ denote the fault vector, where $\Theta_i = 0$ indicates that $i-$th actuator is faulty, while $\Theta_i = 1$ denotes it is not faulty. Let $\Theta_{NN} : \mathbb{Y} \times \mathbb{U} \to \mathbb{R}^m$ be the *predicted* fault vector, parameterized as a neural network. Here, $\mathbb{Y} = \{y(\cdot) \mid y(\cdot) = \rho(x(\cdot)),\ x(\cdot) \in \mathcal{X}\}$ is a function space consisting of trajectories of the output vector, and $\mathbb{U} = \{u(\cdot) \mid u(\cdot) \in \mathcal{U}\}$ is a function space consisting of input signals. To generate the residual data, the knowledge of the system model is essential, which makes the residual-based approach model dependent. This is the biggest limitation of this approach, as modeling errors can lead to severe performance issues in fault detection due to model uncertainties. To overcome this, we propose a model-free NN-based FDI mechanism that only uses the system input and output (y, u) as the feature data, i.e., it does not require the model-based residual information. For a given time length $T > 0$, at any given time instant $t \geq T$, the NN function Θ_{NN} takes a finite trace of the system trajectory $y(t-\tau)|_{\tau=0}^{T}$ and the *commanded* input signal $u(t-\tau)_{\tau=0}^{T}$ as input, and outputs the vector of predicted faults.

Model-based FDI: For model-based FDI mechanisms, the residual data is also required as an additional feature to the NN. The error vector $\tilde{y}$ is defined as the stepwise error between the actual output of the system with potentially faulty actuators and the output of the system assuming no faults, i.e., $\tilde{y}(t) = y(t) - \bar{y}(t)$ where y is the output of (1) with faulty input and $\bar{y}$ is the output of $\dot{\bar{x}} = f(\bar{x}) + g(\bar{x})u, \bar{y} = \rho(\bar{x})$, with $\bar{y}(k\tau) = y(k\tau)$, $k = 0, 1, 2, \ldots$, where $\tau > 0$ is sampling period for data collection. In most of the prior literature, only $\tilde{y}$ (or $\tilde{x}$, depending on whether the approach is output-based or state-based) information is used for designing FDI. In the numerical experiments, we compare the performance of an FDI with just $\tilde{y}$ trajectory as the feature, and $(y, \tilde{y}, u)$ trajectories as features.

Training data: For training, the trajectory data is collected under all one-actuator faults where one of the actuators is completely faulty, i.e., results in zero input. At each time instant $t \geq T$, it is possible that only a portion of the trajectory $y(\cdot)$ is generated under a faulty actuator. That is, the possible input to the system is $u(t-\tau, T_f)_{\tau=0}^{T} := [u(t-T), u(t-T+1), \ldots, u_f(t-T+T_f), u_f(t-T+T_f+1), \ldots, u(t)]$ (with the corresponding output trajectory $y(t-\tau, T_f)_{\tau=0}^{T}$ and the error trajectory $\tilde{y}(t-\tau, T_f)_{\tau=0}^{T}$), where $T_f \in [0, T]$ dictates the time instant when the fault occurs. Thus, the NN for fault prediction must be trained on all possible combinations of occurrences of fault. Hence, our training data includes $\bigcup_{T_f \in 0}^{T}(y(t-\tau, T_f)_{\tau=0}^{T}, \tilde{y}(t-\tau, T_f)_{\tau=0}^{T}, u(t-\tau, T_f)_{\tau=0}^{T})$ (see Figure 2). In every training iteration, we generate $N_{traj} = N_1 \times (m \times T_f + 2)$ trajectories, so that we have $N_1 > 0$ trajectories for faults in each of the m actuators with all possible lengths of trajectories under one faulty actuator in $[0, T_f]$, and $2 \times N_1 > 0$ trajectories without any fault. The loss function for training is defined as

$$\mathcal{L}_\Theta = \sum_{j=1}^{N_{taj}} \Big[||\Theta_{j,rNN}\big(y_j(\cdot), \tilde{y}_j(\cdot), u_j(\cdot)\big) - \Theta_j|| - \epsilon \Big]_+, \quad (7)$$

where $\Theta_j \in \{0,1\}^m$ is the fault vector used for generating the data for the $j-$th trajectory and $0 < \epsilon \ll 1$. In each training epoch, we generate $N = 250 \times m \times 100 + 25000$ trajectories of length T_f and maintain a buffer of 1.5 M trajectories. The trajectory data for training is generated by randomly sampling the initial conditions $\{x(0)\}_1^{N_1}$ from the safe set $\mathcal{X}_{safe}$ and rolling out the closed-loop system under an LQR input for both the fault and non-fault scenarios. We train the NN until the loss reduces to 10^{-3}. We use a Linear-Quadratic Regulator (LQR) input to generate the training data (since solving a CBF-based QP is very slow for collecting a sufficient amount of training data). In our experiments, we illustrate that the trained

NN is highly robust to the kind of input used for trajectory generation and can predict fault with the same accuracy for the trajectories generated by CBF-based QPs. During training, we optimize the loss function using stochastic gradient descent, and we train the pre- and post-fault networks separately. The number of trajectories in the buffer is capped at N_{buf} so that once the maximum number of trajectories are collected, the earlier trajectories are dropped from the buffer. The training is performed either till the number of iterations reaches $N_M > 0$, or the loss drops below 10^{-3} after at least $N_m < N_M$ training epochs. During each training epoch, we use a batch size of 5000 trajectories and perform 100 iterations of training on all the buffer data.

IV. Learning-Based Fault-Recovery

Next, we present a learning framework for designing a fault-recovery control law using CBFs. Before presenting the learning framework, we first extend the definition of CBFs to provide safety guarantees in the presence of actuator faults, and we present a theorem proving the soundness of *fault-tolerant CBF*-based control.

Definition 2 (Fault-Tolerant CBF): Consider a control-affine system $\mathcal{S}$ and disjoint sets $\mathcal{X}_{safe}, \mathcal{X}_{unsafe} \subseteq \mathcal{X}$, $\mathcal{X}_{safe} \cap \mathcal{X}_{unsafe} = \emptyset$. Assume that the fault vector Δu takes values from $\Delta\mathcal{U} = \{\Delta u_1, \ldots, \Delta u_m\}$. A function $h_{post} : \mathcal{X} \mapsto \mathbb{R}$ is a fault-tolerant CBF if there exists a class-$\mathcal{K}$ function α such that:

$$h_{post}(x) < 0, \ \forall x \in \mathcal{X}_{unsafe}, \tag{8}$$

$$h_{post}(x) \geq 0, \ \forall x \in \mathcal{X}_{safe}, \tag{9}$$

$$\sup_{u\in\mathcal{U}} \inf_{\Delta u\in\Delta\mathcal{U}} \left\{ L_f h_{post}(x) + L_g h_{post}(x)(u+\Delta u) \right\} \geq -\alpha(h_{post}(x)) \quad \forall x \in \mathcal{X}. \tag{10}$$

We define the admissible controls for a fault-tolerant CBF by $K_{post}(x) = \{u \in \mathcal{U} \mid L_f h_{post} + L_g h_{post}(u+\Delta u_j) + \alpha(h_{post}(x)) \geq 0, \ \forall j = 1, \ldots, m\}$ for the failed systems. In other words, the control input is admissible for the fault-tolerant CBF if it is admissible at each point of the set $\Delta\mathcal{U}$.

Theorem 1: If h_{post} is a fault-tolerant CBF, then any locally Lipschitz control policy with $\pi_{post}(x) \in K_{post}(x) \ \forall x \in \mathcal{X}$ renders the closed-loop system $\mathcal{S}$ safe for any $\Delta u \in \Delta\mathcal{U}$.

The proof follows from using the Definition 2 with Lemma 1, similar to the proof of [22, Th. 2], and is omitted in the interest of space.

The corresponding fault-tolerant CBF-based QP controller for a system with a loss of control authority is:

$$\pi_{\text{post}}(x) = \underset{u\in\mathcal{U},\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2}\|u - \pi_{\text{nominal}}(x)\|_2^2 + \frac{1}{2}\alpha^2 \tag{11a}$$

$$\text{s.t. } L_f h_{post}(x) + L_g h_{post}(x)u + L_g h_{post}(x)\Delta u_i \geq -\alpha \ h_{post}(x), \quad \forall i \in \{1, \ldots, m\}. \tag{11b}$$

We provide a result on the feasibility, regularity, and correctness of the solutions of the QPs (6) and (11) (see [21]).

Lemma 2: The QPs (6) and (11) are feasible for each $x \in \text{int}(\mathcal{X}_{safe})$. Furthermore, if the strict complementary slackness holds for (6) (respectively, (11)), then π_{pre} (respectively, π_{post}) is continuous on $\text{int}(\mathcal{X}_{safe})$.

One method to encode m constraints in (11b) via a single constraint is to use the following optimization formulation:

$$\pi_{\text{post}}(x) = \underset{u\in\mathcal{U},\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2}\|u - \pi_{\text{nominal}}(x)\|_2^2 + \frac{1}{2}\alpha^2 \tag{12a}$$

$$\text{s.t. } L_f h_{post}(x) + L_g h_{post}(x)u + \min_i L_g h_{post}(x)\Delta u_i \geq -\alpha \ h_{post}(x). \tag{12b}$$

It is easy to solve $\min_i L_g h_{post}(x)\Delta u_i$ by enumerating m options (this can be done before solving the QP since this term does not depend on any decision variables), and the resulting optimization in (12) is still a QP with just one inequality constraint. Note that the post-fault CBF constraint assumes the worst-case fault, and hence, learning one single post-fault CBF with the worst-case fault is sufficient to guarantee safe recovery from all possible faults in any one of the actuators.

In this letter, we use a linear class$-\mathcal{K}$ function $\alpha(h(x)) = \alpha \ h(x)$ with $|\alpha| \leq \alpha_M$ for some $\alpha_M > 0$, and modify (10) as

$$\sup_{|\alpha|\leq\alpha_M} \inf_{\Delta u\in\Delta U} \left\{ L_f h(x) + L_g h(x)(u+\Delta u) + \alpha \ h(x) \right\} \geq 0, \tag{13}$$

The satisfaction of this modified CBF condition implies the existence of a parameter $\alpha \in [-\alpha_M, \ \alpha_M]$ and $u \in \mathcal{U}$ for each faulty signal $\Delta u \in \Delta\mathcal{U}$ such that safety can still be guaranteed. Similarly, (5) can also be modified. Thus, we only need to learn the pre-and the post-CBFs.

The CBFs $h_{pre}, h_{post} : \mathcal{X} \to \mathbb{R}$ are parameterized as neural networks that are trained offline. Here, we also learn π_{pre} and π_{post} as witnesses that the feasible sets of the corresponding CBF QP controllers (6) and (12) will be non-empty. Once the CBFs are learned, we use the CBF and the nominal controller π_{nominal} online in a QP to find the safe control policy π. To learn these functions, we use an iterative learning procedure. At each step, we generate N training points $\mathcal{X}_I = \{x_i\}$ randomly sampled from $\mathcal{X}$. We then define an empirical loss for training the pre-fault CBF h_{pre} to satisfy the conditions in Definition 1:

$$\begin{aligned}\mathcal{L}_{pre} = {} & \frac{a_1}{N_{safe}} \sum_{x\in\mathcal{X}_I\cap\mathcal{X}_{safe}} [\epsilon - h_{pre}(x)]_+ \\ & + \frac{a_2}{N_{unsafe}} \sum_{x\in\mathcal{X}_I\cap\mathcal{X}_{unsafe}} [\epsilon + h_{pre}(x)]_+ \\ & + \frac{a_3}{N_{\text{train}}} \sum_{x\in\mathcal{X}_I} \Big[- \sup_{|\alpha|\leq\alpha_M} \big(L_f h_{pre}(x) \\ & + L_g h_{pre}(x)\pi_{pre} + \alpha h_{pre}(x) \big) + \epsilon \Big]_+ \end{aligned} \tag{14}$$

where $a_1, a_2, a_3 > 0$ are tuning parameters, $\epsilon > 0$ is a small parameter that allows us to encourage strict inequality satisfaction, $[\cdot]_+$ stands for the ReLU function, and N_{safe} and N_{unsafe} are the number of points in the training set that fall into $\mathcal{X}_{safe}$ and $\mathcal{X}_{unsafe}$, respectively. For post-fault CBF, we train m CBFs, one corresponding to a fault in each of the actuators. For $k-th$ post-fault CBF with $k \in \{1, 2, \ldots, m\}$, we modify the empirical loss by replacing $L_g h_{pre}(x)\pi_{pre}$ with $L_g h_{post}(x)\pi_{post} - L_{g_k} h_{post}(x)\pi_{pre,k}$ to account for zero actuation

from $k-$th actuator. We use a similar iterative training mechanism as described in Section III to train the CBFs, where instead of trajectories, we sample data points $x \in \mathbb{R}^n$.

Switching law: Based on the CBFs and FDI mechanism, we are ready to propose a switching-based control algorithm for input assignment. The control law is given as

$$\pi(t,x) = \begin{cases} \pi_{pre}(x) & \text{if} \quad t \le T; \\ \pi_{pre}(x), & \text{if} \quad t \ge T, \min \Theta_{NN}(t,y) > \Theta_{tol}; \\ \pi^{k^*}_{post}(x), & \text{otherwise}; \end{cases} \tag{15}$$

where $0 < \Theta_{tol} \ll 1$ is the prediction tolerance, and $\Theta_{k,NN}(t,y) = \Theta_{k,NN}(y(t-\tau)^T_{\tau=0}, u(t-\tau)^T_{\tau=0})$ denotes the $k-$th component of the predicted Θ vector. The predicted faulty actuator is given by $k^* = \arg\min \Theta_{k,NN}(t,y)$ if $\min \Theta_{NN}(t,y) < \Theta_{tol}$, and the control algorithm switches to post-fault CBF h_{post,k^*} for synthesizing $\pi^{k^*}_{post}$ for safe recovery.

V. Numerical Evaluations

The primary objective of our numerical experiments is to evaluate the effectiveness of our method in terms of fault detection. We consider an experimental case study involving the Crazyflie quadrotor with a fault in one of its motors. The 6-DOF quadrotor dynamics are given in [23] with $x \in \mathbb{R}^{12}$ consisting of positions, velocities, angular positions, and angular velocities, and $u \in \mathbb{R}^4$ consisting of the thrust at each of four motors. The output is chosen as $y = [p_x, p_y, p_z, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ which can be readily obtained using onboard GPS and IMU. In the case study, we consider the scenario when one of the motors is entirely faulty, i.e., produces zero input.

The state limit set is defined as $\mathcal{X} = \{x \mid |p_x|, |p_y|, |p_z| \le 25, |u|, |v|, |w| \le 10, |\phi|, |\theta|, |\psi| \le \frac{\pi}{3}, |p|, |q|, |r| \le 2\}$, the safe region is this case is defined as $\mathcal{X}_{safe} = \{x \in \mathcal{X} \mid 2 \le p_z \le 24, |w| \le 8\}$, where z is the altitude of the quadrotor. Similarly, the safe region for the post-fault case is defined as $\bar{\mathcal{X}}_{safe} = \{x \in \mathcal{X} \mid 1.9 \le z \le 24.1, |w| \le 8.1\}$ so that $\mathcal{X}_{safe} \subset \bar{\mathcal{X}}_{safe}$. The unsafe region is defined as $\mathcal{X}_{unsafe} = \{x \in \mathcal{X} \mid p_z \le 0.2$ or $p_z \ge 24.5$ or $w \le -9$ or $w \ge 9\}$, so that $\mathcal{X}_{safe} \cup \mathcal{X}_{unsafe} \ne \mathcal{X}$. This allows a non-empty region in $\mathcal{X}$, defined as $\mathcal{X}_{mid} = \mathcal{X} \setminus (\mathcal{X}_{safe} \cup \mathcal{X}_{unsafe})$ where there is no sign-requirement for the CBF. This helps improve the learning as it is generally hard to enforce that a NN has a specific zero level set, and this non-empty region $\mathcal{X}_{mid}$ allows the barrier function to smoothly decay from positive values in $\mathcal{X}_{safe}$ to negative values in $\mathcal{X}_{unsafe}$.

In the training, we use fully-connected NNs with tanh activation functions to parameterize the CBFs h_{pre} and h_{post}. At each learning iteration for h_{pre} (respectively, h_{post}), we generate 30,000 data points $\{x_i\}$, out of which 10,000 data points are sampled from the boundary of $\mathcal{X}_{safe}$ (respectively, $\bar{\mathcal{X}}_{safe}$), 10,000 from the safe set $\mathcal{X}_{safe}$ and 10,000 from the unsafe set $\mathcal{X}_{unsafe}$, and add these points to the buffer of the previously collected samples. The number of sampling points in the buffer is capped at 10^6 so that once the maximum number of samples are collected, the earlier samples are dropped from the buffer. The training is performed either till the number of iterations reaches 500, or the loss drops below 10^{-3}. For the post-fault CBF training, we assume that motor #1 is faulty for the CBF condition. During each training step, we use a batch size of 5000 samples and perform the training 10 times on all the data currently present in the buffer. Adam algorithm is used for optimization with learning rate 1×10^{-4}.

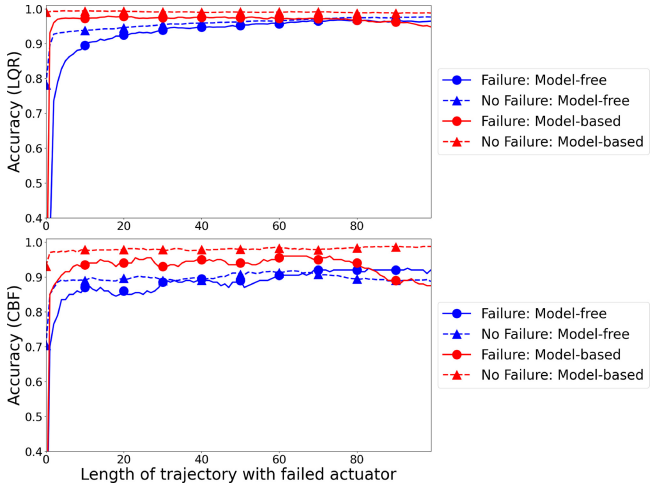

Fig. 3. Failure prediction accuracy for CBF-QP input (solid lines) and LQR input (dashed lines). The performance of model-free (Ours) FDI with data (y, u) is shown in blue, while the one with all the data $(y, u, \tilde{y})$ in red.

A fault is predicted if $\min_i \min_n \Theta_{i,NN}(\Phi_n(y)) < \Theta_{tol}$, where $\Theta_{tol} = 0.1$. The experiments are run to check the prediction accuracy of the NN-based FDI mechanism for various lengths of data with failed actuators between 0 and $T_f = 100$. We report the minimum of the prediction accuracy for fault detection when there is a fault as well as when there is no fault, across all 4 motors. Thus, a high overall prediction accuracy implies that FDI can correctly identify which actuator has a fault and when. We sample 1000 initial conditions randomly from the safe set $\mathcal{X}_{safe}$ to generate trajectories for test data, where 200 trajectories are generated for each of the faults and 200 trajectories are generated without any fault. Each trajectory is generated for 200 epochs with fault occurring at $t = 100$. We feed the moving trajectory data $(y(k-100,k), u(k-100,k), \tilde{y}(k-100,k)$ to the trained NN-based FDI starting from $k = 100$. For a given $k \in [100, 200]$, the portion of trajectory data with faulty actuator is $k - 100$.

We use a long-short-term-memory (LSTM)-based NN architecture for FDI where the LSTM layer is followed by 2 linear layers (as we observed superior performance of LSTM over multi-layer perceptron (MLP)). We first compare the prediction accuracy of the model-free NN-FDI ($\theta_{NN}(y,u)$) and model-based FDI. Figure 3 shows the prediction accuracy of the model-based FDIs, where it can be seen that the model-free FDI mechanism can perform at par with the model-based FDI mechanism. Based on this observation, we can infer that a model-free FDI mechanism can be used with very high confidence. We use $N \times 128$ as the size of the input layer with N being the size of the features, hidden layer(s) of size 128×128 followed by a hidden layer of size 128×64 and an output layer of size $64 \times m$. Note that $N = (2p + m) \times T_f$ for the FDI with

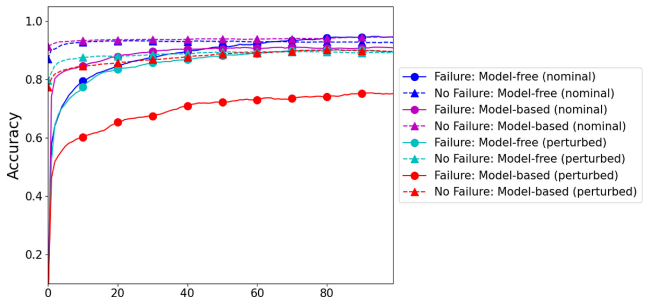

Fig. 4. Comparison of model-based and model-free FDI mechanisms with perturbation in system parameters.

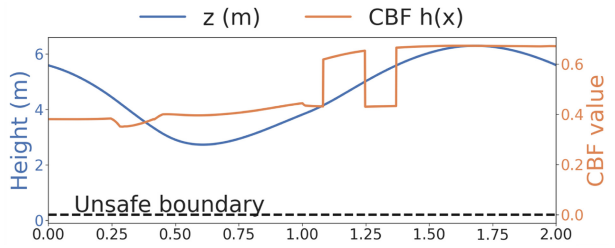

Fig. 5. Closed-loop plots under fault of actuator #2. The fault occurs at $t = 1.00$ sec.

all the data, $p \times T_f$ for the FDI with just the residual data, and $N = (p + m) \times T_f$ for the model-free FDI mechanism. Next, we also study the effect of change in model parameters (such as the inertia matrix, etc.) on the prediction accuracy of the FDI mechanisms. For this experiment, we changed the system parameters by more than 40%. As can be seen from Figure 4, the prediction accuracy of the model-free FDI mechanism changes only slightly due to changes in the model parameters, while that of the model-based FDI mechanism drops significantly. Thus, in the scenarios when a correct system model is not known or the system dynamics undergo changes during operation, a model-based FDI mechanism might not remain reliable.

Finally, the closed-loop performance with all the components integrated is illustrated in Figure 5. Here, the fault occurs in motor #2 at $t = 1.0$ sec, and the designed architecture can keep the system from crashing on the ground. The plot shows that the quadrotor maintains a safe altitude by switching to the post-fault CBF h_{post}. This illustrates that the proposed framework is capable of accurately identifying a fault and safely recovering the system from it.

VI. Conclusion

In this letter, we propose a learning method for effectively learning a model-free FDI and a switching mechanism for automatically detecting and recovering from a fault. The numerical experiments demonstrated that the applicability of a model-based FDI mechanism is very limited, while that of the proposed model-free is quite broad and general.

As part of future work, we will explore methods that can incorporate more general fault models where the faulty actuator can take any arbitrary signal, and more than one actuator can undergo failure simultaneously.

References

[1] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[2] H. Zhang, Z. Li, and A. Clark, "Safe control for nonlinear systems under faults and attacks via control barrier functions," 2022, *arXiv:2207.05146*.

[3] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *Proc. Int. Conf. Learn. Representations*, Jan. 2021, pp. 1–16.

[4] K. Garg, R. G. Sanfelice, and A. A. Cardenas, "Control barrier function based attack-recovery with provable guarantees," in *Proc. IEEE Conf. Decis. Control*, 2022, pp. 4808–4813.

[5] A. Eltrabyly, D. Ichalal, and S. Mammar, "Fault-tolerant model predictive control trajectory tracking for a quadcopter with 4 faulty actuators," *IFAC-PapersOnLine*, vol. 54, no. 4, pp. 141–146, 2021.

[6] B. Wang, Y. Shen, and Y. Zhang, "Active fault-tolerant control for a quadrotor helicopter against actuator faults and model uncertainties," *Aerosp. Sci. Technol.*, vol. 99, Apr. 2020, Art. no. 105745.

[7] X. Zhu, J. Chen, and Z. H. Zhu, "Adaptive learning observer for spacecraft attitude control with actuator fault," *Aerosp. Sci. Technol.*, vol. 108, Jan. 2021, Art. no. 106389.

[8] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 636–653, May 2010.

[9] R. Puchalski and W. Giernacki, "UAV fault detection methods, state-of-the-art," *Drones*, vol. 6, no. 11, p. 330, 2022.

[10] G. K. Fourlas and G. C. Karras, "A survey on fault diagnosis methods for UAVS," in *Proc. 2021 Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2021, pp. 394–403.

[11] J.-H. Park and D. E. Chang, "Data-driven fault detection and isolation of system with only state measurements and control inputs using neural networks," in *Proc. 21st Int. Conf. Control, Autom. Syst. (ICCAS)*, 2021, pp. 108–112.

[12] A. Bondyra, M. Kołodziejczak, R. Kulikowski, and W. Giernacki, "An acoustic fault detection and isolation system for multirotor UAV," *Energies*, vol. 15, no. 11, p. 3955, 2022.

[13] C. Alippi, S. Ntalampiras, and M. Roveri, "Model-free fault detection and isolation in large-scale cyber-physical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 1, pp. 61–71, Feb. 2017.

[14] M. Bakhtiaridoust, M. Yadegar, N. Meskin, and M. Noorizadeh, "Model-free geometric fault detection and isolation for nonlinear systems using Koopman operator," *IEEE Access*, vol. 10, pp. 14835–14845, 2022.

[15] M. Thirumarimurugan, N. Bagyalakshmi, and P. Paarkavi, "Comparison of fault detection and isolation methods: A review," in *Proc. 10th Int. Conf. Intell. Syst. Control (ISCO)*, 2016, pp. 1–6.

[16] J. Song, W. Shang, S. Ai, and K. Zhao, "Model and data-driven combination: A fault diagnosis and localization method for unknown fault size of Quadrotor UAV actuator based on extended state observer and deep forest," *Sensors*, vol. 22, no. 19, p. 7355, 2022.

[17] M. Yadegar, M. Bakhtiaridoust, and N. Meskin, "Adaptive data-driven fault-tolerant control for nonlinear systems: Koopman-based virtual actuator approach," *J. Frank. Inst.*, vol. 360, no. 11, pp. 7128–7147, 2023.

[18] M. Elnour, N. Meskin, and M. Al-Naemi, "Sensor fault diagnosis of multi-zone HVAC systems using auto-associative neural network," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, 2019, pp. 118–123.

[19] S. Ren, Y. Jin, J. Zhao, Y. Cao, and F. Si, "Nonlinear process monitoring based on generic reconstruction-based auto-associative neural network," *J. Frank. Inst.*, vol. 360, no. 7, pp. 5149–5170, 2023.

[20] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Control*, 2014, pp. 6271–6278.

[21] K. Garg, E. Arabi, and D. Panagou, "Fixed-time control under spatiotemporal and input constraints: A quadratic programming based approach," *Automatica*, vol. 141, Jul. 2022, Art. no. 110314.

[22] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural Lyapunov-barrier functions," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 1724–1735.

[23] C. Budaciu, N. Botezatu, M. Kloetzer, and A. Burlacu, "On the evaluation of the crazyflie modular quadcopter system," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2019, pp. 1189–1195.