Mechanical Systems and Signal Processing 190 (2023) 110099

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

Online fault classification in Connected Autonomous Vehicles using output-only measurements

Abdelrahman Khalil [a], Mohammad Al Janaideh [a,b,*], Deepa Kundur [c]

[a] Department of Mechanical Engineering, Memorial University, St. John's, NL A1B 3X5, Canada
[b] School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada
[c] Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada

ARTICLE INFO

Communicated by L. Mevel

Keywords:
Connected Autonomous Vehicles
Fault classification
Imitation learning
Transmissibility

ABSTRACT

Different health-monitoring techniques were considered in the literature to enhance the safety and stability of Connected Autonomous Vehicle (CAV) platoons. Mitigating these faults is faster and more reliable if the fault structure is known. In this paper, we propose using transmissibility operators, which are relationships that relate a set of velocities with another in the platoon, to classify the faults. Transmissibility operators were shown to be exceptional in signal estimation. However, it is also shown to be noncausal and thus can only be used offline. To this end, we propose using Data Aggregation (DAgger), an extension to imitation learning, to transfer the classification experience from transmissibility operators to a novice machine learning agent to be used online. The integration of transmissibility-DAgger gives the ability to adapt to new fault classes without the need to re-train the diagnosis model from the beginning. A heterogeneous CAV platoon was modeled with three different faults separately. These faults are actuator disturbances, false data injection attacks, and communication time delays. The classification scheme depends on estimating the faulty signal in the case of each fault class. Next, the measured faulty signal is compared with the three estimations, and the closer fault estimation to the measured one is considered the actual fault on the platoon. The proposed algorithm is then tested on the platoon model and then applied to an experimental setup that consists of three autonomous robots. The proposed results are compared with six different machine learning classification models. The overall classification accuracy achieved was 95.8% for the experiment using the proposed approach.

1. Introduction

To achieve a fully connected autonomous vehicle platoon, autonomous vehicles should be developed with the adequate ability to enhance platoon safety and conquer threats. This includes securing the platoon against faults and potential cyber-attacks on both physical and cyber layers. Many methods were considered in the literature for fault detection, localization, and mitigation, including but not limited to artificial neural networks [1] and model-based methods [2]. Considering the technology and complexity of connected autonomous vehicles evolving, a wide range of different classes of faults may occur [2]. Therefore, fault classification is necessary for reliable and fast recovery in different dynamic systems [3].

CAV platoon faults can be categorized according to the occurrence layer into physical and cyber faults. Physical faults occur in the vehicle dynamics, such as sensors and actuator faults [4]. This category of faults is affected by vehicle dynamics. The fault gets

* Corresponding author at: Department of Mechanical Engineering, Memorial University, St. John's, NL A1B 3X5, Canada.
E-mail address: mohammad.aljanaideh@utoronto.ca (M.A. Janaideh).

https://doi.org/10.1016/j.ymssp.2023.110099
Received 23 June 2022; Received in revised form 5 October 2022; Accepted 3 January 2023
Available online 30 January 2023

mixed with the vehicle dynamics and will be shaped as dynamic response. For example, assume a bounded disturbance fault in the velocity sensor used for the closed-loop Cruise Control (CC). The disturbance will be fed back into the controller and then the vehicle dynamics, which affects the power flow between the different physical components within the vehicle. Such a fault within the CC system can be easily mitigated by using, for example, the incremental twisting fault-tolerant control with partial vehicle model as in [5]. On the other hand, cyber faults are easier to recover as they affect the information transmitted through wireless communications without affecting the power propagation between the physical components. Cyber faults include, but are not limited to, spoofing, message falsification, false data injection (burst transmission), and denial-of-service [2]. Both fault categories can be avoided easily if the fault type is classified. For example, for communication time delay, a redundant communication link can be used. In another example, the event-triggered adaptive formation keeping, and interception introduced in [6] can be used to mitigate malicious cyberattacks. However, all fault classes result in similar corrupted signals, which makes it difficult for the vehicles to classify them. This wide range of possible faults requires a novel strategy to classify the fault class within the vehicle's network to enhance platoon safety.

In this paper, we consider first classifying faults using transmissibility operators first, which are mathematical relations that characterize the relationship between different sets of velocities. Transmissibilities are independent of the inputs acting on the underlying system and the dynamics of the underlying system. Transmissibilities will use one subset of the outputs to obtain an estimation of another subset. These operators re-arrange the system zeros to relate them with each other while excluding the system poles. The dependency on the system input fades with excluding the system poles. Although excluding the system poles makes it difficult to study the system dynamics directly (the system states), however, it showed great potential in estimating the system output. Transmissibility operators have been improved over the recent years, and have become one of the most beneficial output estimators while the system inputs are unknown, see for example [7]. In recent years, transmissibilities have been improved significantly, and the applicability range was developed. Time-domain transmissibilities were introduced in [7] by implementing the time differentiation operator $\mathbf{p} = \frac{\mathbf{d}}{\mathbf{d}t}$ instead of the Laplace complex variable s to count for non-zero initial conditions. Transmissibility operators have the privilege of precisely estimating the faulty velocity signal while the fault dynamics are unknown [7,8].

Transmissibility operators were shown to be independent of the excitation signals that act on the platoon. By formulating the fault dynamics as independent excitations, transmissibility operators can then overcome the problem of unknown fault dynamics and estimate the faulty velocity signal. See for example, the transmissibility potentials in dealing with systems under unknown fault dynamics [9]. However, transmissibility operators are noncausal in some cases and thus can only be used offline. In CAV platoons, health monitoring has to be done online immediately without any lags since the time between the fault and crash is very small and might be less than a second. To this end, this paper develops transmissibility operators to classify the platoon faults (expert policy), then transfers the classification knowledge to a set of deep neural networks (novice policy). Imitation learning was shown to be able to develop skills from direct experience, however, this approach suffers from data mismatch and compounding errors [10]. DAgger, which is an extension in the supervised learning that extracts the expert policy to enhance the novice policy training, can enhance the fault classifiers training [10].

DAgger was shown to solve many issues in imitation learning. These issues include failing when the neural networks that being trained encounters situations not adequately within the training dataset [11]. This is represented in inability to adapt with new fault classes in this paper. The goal of DAgger is to minimize the expert intervention while efficiently train the novice policy. DAgger allows the novice policy to be in charge, then takes the chance to correct and update the novice policy if it fails. This is done by firstly monitor the discrepancies between the novice and expert actions. If the novice deviates from the expert policy, DAgger will place the expert in charge, record the expert actions, and then train the novice on them. This will allow after to use the trained neural networks online for the purpose of fault classification.

One of the main advantages in this work is the ability to easily adapt with new fault classes without the need to re-train the classification model from the beginning. This is due to the integration of transmissibility-DAgger, and can be simply done by creating a new transmissibility operator for the new fault class. Next, DAgger will add the new experience to the previously trained classification model. Note that transmissibility operators are mathematical estimators (similar to transfer functions) that do not require any intense training or computations, which gives straightforwardness in including more fault classes to the classification model. Moreover, it is important to keep in mind that the proposed approach is applicable on deep neural networks, which is the most common machine learning method used in autonomous vehicles, see for example [12,13].

2. Related work

In [14], machine learning was used to classify the failure types in an autonomous robotic assembly architecture. The assembly strategy was chosen at the beginning according to the probability of achieving a successful assembly. This makes the classification algorithm not robust against different assembly strategies. Therefore, the technique in [14] is only applicable under specific operating conditions, that is, the faulty behavior was already known. This limits the applicability on new unknown faults that the classifier was not designed on. In [15], an approach was presented to exploit the phase structure of tasks to learn manipulation skills of robot arms. The robot learns a probabilistic model of phases, then the robot classifies normal or abnormal manipulation behaviors without classifying the abnormal behavior type. Applying this approach to CAV platoons might only detect the existence of faults, without the ability to classify the fault class. In [16], Support Vector Machine (SVM) was used to distinguish successful and unsuccessful automated parts assemblies using the force signature analysis, however, no information was given about the unsuccessful assembly to recover the failure. Similar to [15], the main disadvantage is represented in the inability to classify the fault class, not only detect it. In [17], the authors detect the failure in a robot picking operation, then the failure was classified according to the fault frequency.

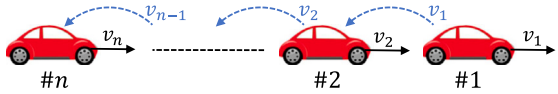

Fig. 1. CAV platoon with n vehicles. Each vehicle sends its velocity to the following vehicle using V2V communication links.

An assumption was also made in [17] that high frequency faults mean that something unusual occurred in the production line or the production process. The main disadvantage of this method is the dependency on the fault dynamics. This renders this method customized to the investigated system under a limited number of faults.

A novel threading task fault classification was introduced in [18] using the Convolution Neural Networks (CNN), and it was compared with SVM and Multilayer Perception (MLP). As shown in [18], CNN cannot be spatially invariant to the input data. SVM was shown to perform well when the fault class was clearly labeled, however, it is difficult to implement it on large data sets such as fault signals estimation as much higher training time is required [19]. Moreover, MLP requires a very high number of parameters, which makes the training more intense. A novel use of CNN was introduced in [20] for object classification using vision and light detection and ranging (LIDAR) fusion of autonomous vehicles in the environment. The work introduced in [20] uses image processing and each pixel depth data to classify objects. A novel fault classification approach using artificial neural networks was proposed in [21]. Although, CNN was shown to be used effectively in [20,21], the cost of the improvements in was more intense training. Another method that is well-proven in ML in the literature is random forest classifier. In random forest classifiers, including more fault classes needs more classification trees, which makes the algorithm slower and might cause problems in online estimation, see for example [22,23]. Thus, random fault classifiers are more suitable for offline classification.

This work is highly motivated by the DAgger results in cloning the behavior of different dynamic objects. In [24] DAgger was successfully used to learn the behavior of a monocular in reactive UAV control in cluttered natural environments. In [10] DAgger was used to imitate the behavior of a simple inverted pendulum and another results were presented for MuJoCo HalfCheetah OpenAI Gym environment. In [25] DAgger was used to imitate the behavior of a nonlinear MPC controller for flying robots. DAgger has shown many desirable properties, including online functionality and theoretical guarantees [10]. Different extensions were proposed to guarantee DAgger safety such as SafeDAgger as in [26], Human Gate DAgger [27], VanillaDAgger and EnsembleDAgger [10]. In this paper, the authors focus on imitating the transmissibility potentials in fault estimation, and implying the DAgger extensions is considered for future work.

On the other hand, estimating sensors measurements, also known as soft sensing, include many techniques from different fields [28]. The most common techniques in signals estimation of dynamic systems are observer-based techniques, see for example [2,29]. However, observer-based estimation requires the knowledge of the fault dynamics and faulty states to estimate the faulty signals. Many challenges introduced in [28] for signals estimation. These challenges include minimizing the assistance of human experts, such as in model selection, and filling the gap between the laboratory outcome and the industrial practice. Considering the soft sensing techniques in [28], some of these techniques require building a special hardware for the soft sensor [30], where other require knowledge of the fault dynamics or the dynamics of the system [31], or some information about the system like the impulse response parameters [32].

With regard to the signals estimation challenges in [28], DAgger is used in this paper to minimize the human interaction as the expert policy is cloned. The gap between the laboratory outcome and the industrial practice is minimized, as transmissibility operators are robust against external disturbances and any other external excisions on the platoon. No additional hardware components or additional sensors are required, and only available velocity sensors are used.

In this paper, we propose a classification scheme to classify both physical and cyber faults in CAV platoons using only velocity measurements. The proposed classification scheme considers first transmissibility operators for offline classification. Then, DAgger is used to transfer the transmissibility experience in classifying faults to a novice machine learning agent.

The main contributions of this work based on the literature above are as follows:

- Transmissibility operators are used for fault estimation. Transmissibilities are mathematical operators that do not require training, independent of the fault and platoon dynamics, and robust against external disturbances.
- DAgger is used to transfer the fault estimation knowledge of to a machine learning agent, which allows for online fault estimation. Moreover, this allows to include new fault classes directly to the machine learning agent.
- We develop an algorithm that uses output-only measurements available from the CAV platoon for fault classification without presuming previous knowledge of the type of fault. This algorithm can handle unknown fault dynamics at an unknown location. This renders the proposed algorithm applicable to a wide range of physical and cyber faults.

3. Heterogeneous CAV platoon modeling

Consider an autonomous heterogeneous vehicles platoon with n vehicles as in Fig. 1. Let $i \in \{1, \dots, n\}$ denote the vehicle order within the platoon and v_i denote the velocity of vehicle i. Each vehicle follows the velocity of its preceding vehicle such that $v_i^*(t) = v_{i-1}(t)$ where v_i^* is the desired velocity of the vehicle i. We consider three different CAV models within the platoon to enhance the DAgger training. Each vehicle is considered to have a closed-loop PI controller (cruise control) to track the desired velocity v_i^* and produce the control signal u_i.

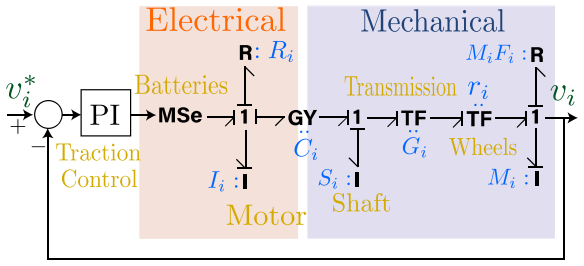

Fig. 2. Bond graph model of an electric powertrain vehicle that is considered to characterize CAVs.

Table 1
Platoon model Parameters description and values.

Symbol	Description	Value
τ_i	Time lag constant in (3.1)	0.5 s
R_i	Motor resistance	18 mΩ
I_i	Motor inductance	252 μH
C_i	Motor constant	0.26 rad/s A
S_i	Shaft moment of inertia	0.2 kg m^2
G_i	Transmission ratio	0.2
r_i	Wheel radius	0.3 m
M_i	Vehicle gross mass	1478 kg
F_i	Friction coefficient	0.6
σ_i	Headway gain in (3.3)	0.01
v_i	Relative velocity gain in (3.3)	0.28
$\dot{f}(h^*)$	Constant in (3.3)	0.1.389
$k_{P_1,i}, k_{P_2,i}, k_{P_3,i}$	Probational gain in (3.4)	0.75, 2.5, 0.75
$k_{I_1,i}, k_{I_2,i}, k_{I_3,i}$	Integral gain in (3.4)	0.01, 0.6, 0.25

3.1. CAV dynamics

The first model is based on an acceleration time lag filter to track the velocity of the preceding vehicle. The second model is created using the bond graph approach to simulate a realistic interaction between the different vehicle components, which allows more precise physical faults modeling. The third model is a leader–follower dynamic model that assures a safe spacing distance with the neighboring vehicles.

3.1.1. First CAV model

In the first CAV model, we adopt the dynamic vehicle model in [33, Chapter 5, Section 5.3] for the longitudinal motion of the vehicle. The vehicle is modeled as a first order time-lag filter between the desired acceleration a_i^* and the actual acceleration. Then the vehicle velocity is obtained by integrating the time-lag filter such that

$$v_i(t) = \frac{1}{\mathbf{p}(\tau_i \mathbf{p} + 1)} u_i(t), \tag{3.1}$$

where τ_i is the time-lag constant, $\mathbf{p} = \frac{\mathrm{d}}{\mathrm{d}t}$ is the differentiation operator, and $u_i = a_i^*$ is the tracking control signal.

3.1.2. Second CAV model

We model the electric powertrain topology introduced in [34] using the bond graph approach. Following [34], we consider the drive motor as a Brushless DC Motor that extracts power from the batteries based on the traction control signal. Fig. 2 shows the bond graph model of the vehicle considered, while parameters description and their numerical values are defined in Table 1. Following the formulation procedure in [35, Chapter 5], the bond graph model in Fig. 2 can be represented as a linear time-invariant model, as is Section 4.2.

3.1.3. Third CAV model

Following [4], by assuming constant spacing distances the leader–follower model can be represented as

$$\dot{h}_i(t) = u_i(t) - v_i(t), \tag{3.2}$$

$$\dot{v}_i(t) = \sigma_i \dot{f}(h^*) h_i(t) - (\sigma_i + \nu_i) v_i(t) + \nu_i u_i(t). \tag{3.3}$$

where h_i is the spacing distance between vehicles i and $i-1$, σ_i and ν_i are parameters that denote headway gain and relative velocity gain, respectively, where $\sigma_i > 0$ and $\sigma_i + \nu_i > 0$, and f denotes a range policy that is linearized at the constant spacing distance h^*.

3.2. Cruise control

The tracking control is represented in producing the control command u_i based on the desired velocity v_i^*, this can be done by using the cruise control, which is a simple closed-loop feedback with a proportional–integral (PI) controller given by

$$u_i(t) = (k_{\mathrm{P_j},i} + \frac{k_{\mathrm{I_j},i}}{\mathbf{p}})[v_i^*(t) - v_i(t)], \tag{3.4}$$

where $k_{\mathrm{P_j},i}$ and $k_{\mathrm{I_j},i}$ are the proportional and integrational gains of the ith vehicle that follows the jth CAV model, respectively.

4. Faulty platoon modeling

4.1. Fault scenarios

The following faults were investigated thoroughly in [36]. In this paper, same faults are considered to imply the proposed fault classification approach.

4.1.1. Actuator disturbances

Brushless DC motors that are used in electric vehicles are subjected to vulnerable operating conditions, including high magnetic force and severe weather conditions. Following [37], we introduce an additive fault to the motor's nominal value of the current-to-torque ratio after motor loss of effectiveness occurs. The faulty motor constant is then given by

$$\tilde{C}_i(t) = 0.8C_i + \delta_{C_i}(t), \tag{4.1}$$

where $\tilde{C}_i$ is the corrupted motor constant, and δ_{C_i} is the bounded deviation from the original motor constant after the loss of effectiveness occurs.

4.1.2. False Data Injection (FDI)

We consider false data injection in the communication link between vehicle i and vehicle $i+1$. For all $i \in \{1, \dots, n\}$,

$$\tilde{v}_i(t) = v_i(t) + \delta_{f,i}(t), \tag{4.2}$$

where $\tilde{v}_i$ represents the corrupted velocity of the vehicle i, and $\delta_{f,i}$ denotes the FDI disturbances.

4.1.3. Communication delay (Denial-of-service)

We consider the velocity of vehicle i that is received by vehicle $i+1$ to be time delayed. Then such a delay yields the corrupted signal,

$$\tilde{v}_i(t) = v_i(t - \delta_{v,i}(t)), \tag{4.3}$$

where $\tau_{v,i}$ is the time-variant communication delay in v_i. By Applying Taylor's theorem to (4.3) we can write the time delay as [38]

$$\tilde{v}_i(t) = v_i(t) + (-\tau_{v,i}\dot{v}_i(t) + \dots). \tag{4.4}$$

Note that (4.4) is still nonlinear, but only written in the additive form.

4.2. Faulty platoon model

The state space representation for a platoon of n vehicles can be given by

$$\dot{x}(t) = Ax(t) + B_v v_1^*(t) + B_{\delta,\mathrm{j}}\delta_i(t), \tag{4.5}$$

$$y(t) = Cx(t), \tag{4.6}$$

where $\delta_i(t)$ is bounded unknown signal due to a platoon fault. The aim of this paper is to classify the fault class that produced δ_i. Moreover,

$$A = \begin{bmatrix} A_1 & & \dots & 0 \\ B_2C_1 & A_2 & & \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & B_nC_{n-1} & A_n \end{bmatrix}, B_v = \begin{bmatrix} B_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$C = \mathrm{diag}(C_1, \dots, C_n), \qquad C_j = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},$$

$$A_i = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\gamma_i & -\beta_i & -\alpha_i \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ \epsilon_i \\ \gamma_i - \alpha_i\epsilon_i \end{bmatrix},$$

where $A \in \mathbb{R}^{n_p \times n_p}$, $B_v \in \mathbb{R}^{n_p \times 1}$, $B_\delta \in \mathbb{R}^{n_p \times 1}$, $C \in \mathbb{R}^{n \times n_p}$, and n_p is the platoon order. $\alpha_i, \beta_i, \gamma_i$, and ϵ_i are constants derived from the vehicle model and are different for each CAV model. For the first CAV model, $\alpha_i = \frac{1}{\tau_i}, \beta_i = \epsilon_i = \frac{k_{\mathrm{P}_1,i}}{\tau_i}$, and $\gamma_i = \frac{k_{\mathrm{P}_1,i}}{\tau_i}$. For the second CAV model, $\eta_j = \frac{S_j}{G_j^2 r_j^2 M_j}, \alpha_i = \kappa_i R_i[\eta_i + 1 + \frac{F_i I_i}{R_i}], \beta_i = \kappa_i[\frac{C_i^2 \eta_i}{S_i} + F_i + \zeta_i k_{\mathrm{I}_2,i}]$, $\gamma_i = \kappa_i \zeta_i, \epsilon_i = \gamma_i k_{\mathrm{I}_2,i}, \kappa_i = \frac{1}{I_i(\eta_i+1)}$, and $\zeta_i = \frac{C_i k_{\mathrm{P}_2,i}}{k_{\mathrm{I}_1,i} G_i r_i M_i}$. And for the third CAV model, $\alpha_i = \sigma_i + \nu_i + \nu_i k_{\mathrm{P}_3,i}, \epsilon_i = \dot{f}(h^*)\sigma_i k_{\mathrm{P}_3,i} + \nu_i k_{\mathrm{I}_3,i}, \beta_i = \sigma_i \dot{f}(h^*) + \epsilon_i, \gamma_i = \sigma_i k_{\mathrm{I}_3,i} \dot{f}(h^*)$.

Let j denote the fault class, for $\mathrm{j} \in \{\mathrm{m}, \mathrm{f}, \mathrm{d}\}$, $\mathrm{j} = \{\mathrm{m}\}$ indicates motor disturbances, $\mathrm{j} = \{\mathrm{f}\}$ indicates FDI fault, and $\mathrm{j} = \{\mathrm{d}\}$ indicates V2V time delay, then we can define

$$\delta_i(t) = \begin{cases} \delta_{C_i}(t), & \mathrm{j} = \{\mathrm{m}\}, \\ \delta_{f,i}(t), & \mathrm{j} = \{\mathrm{f}\}, \\ (-\tau_{v,i}\dot{v}_i(t) + \dots), & \mathrm{j} = \{\mathrm{d}\}, \end{cases} \tag{4.7}$$

$$B_{\delta,\mathrm{j}} = \begin{bmatrix} 0 & \dots & 1 & \dots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{4.8}$$

Note that the entry of the cell that carries the number 1 in $B_{\delta,\mathrm{j}}$ is different for each fault class.

5. Expert classification policy using transmissibility operators

Transmissibility operators are mathematical objects that characterize the relationship between outputs of an underlying system. In this section, we identify a transmissibility operator for each faulty scenario, then the fault can be classified based on which transmissibility operator correctly estimates the measured fault. Transmissibilities can obtain an accurate estimation of the faulty signal under unknown fault δ_i. However, for such an estimation the transmissibility operators must be noncausal, which results in a time-delayed fault classification and cannot be used for online applications. To this end, we implement the fault classification offline using transmissibilities, then use DAgger to transfer the classification experience to a new machine learning agent in Section 6.

Consider the platoon described by the state space model (4.5), (4.6). Then, define

$$y_{\mathrm{i}}(t) \triangleq \begin{bmatrix} v_{i-1}(t) \\ v_{i+1}(t) \end{bmatrix} = C_{\mathrm{i}} x(t) \in \mathbb{R}^p, \tag{5.1}$$

$$v_i(t) = C_{\mathrm{o}} x(t) \in \mathbb{R}, \tag{5.2}$$

to be two independent sets of noise-free velocity outputs, where $p = 2$ is the number of independent pseudo inputs,

$$C_{\mathrm{i}} = \begin{bmatrix} 0 & \dots & 0 & C_{i-1} & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & C_{i+1} & 0 & \dots & 0 \end{bmatrix}, \tag{5.3}$$

$$C_{\mathrm{o}} = \begin{bmatrix} 0 & \dots & 0 & C_i & 0 & \dots & 0 \end{bmatrix}, \tag{5.4}$$

$C_{\mathrm{i}} \in \mathbb{R}^{p \times n_p}$ and $C_{\mathrm{o}} \in \mathbb{R}^{1 \times n_p}$.

The relationships between y_{i} and u and v_i and u can be written as

$$\delta(\mathbf{p}) y_{\mathrm{i}}(t) = \Gamma_{\mathrm{i,j}}(\mathbf{p}) u(t), \tag{5.5}$$

$$\delta(\mathbf{p}) v_i(t) = \Gamma_{\mathrm{o,j}}(\mathbf{p}) u(t), \tag{5.6}$$

respectively, where

$$\Gamma_{\mathrm{i,j}}(\mathbf{p}) \triangleq C_{\mathrm{i}} \mathrm{adj}(\mathbf{p}\mathbf{I}_{n_p} - A) B_{\mathrm{j}}, \tag{5.7}$$

$$\Gamma_{\mathrm{o,j}}(\mathbf{p}) \triangleq C_{\mathrm{o}} \mathrm{adj}(\mathbf{p}\mathbf{I}_{n_p} - A) B_{\mathrm{j}}, \tag{5.8}$$

$$u(t) = \begin{bmatrix} v_1^*(t) & \delta_i(t) \end{bmatrix}^{\mathrm{T}}, \tag{5.9}$$

$B_{\mathrm{j}} = \begin{bmatrix} B_v & B_{\delta,\mathrm{j}} \end{bmatrix}$. Multiplying (5.5) by $\mathrm{adj}\Gamma_{\mathrm{i,j}}(\mathbf{p})$ from the left and using the fact that

$$\mathrm{adj}\Gamma_{\mathrm{i,j}}(\mathbf{p})\, \Gamma_{\mathrm{i,j}}(\mathbf{p}) = \det \Gamma_{\mathrm{i,j}}(\mathbf{p}) \mathbf{I}_{n_p} \tag{5.10}$$

yields

$$\delta(\mathbf{p}) \mathrm{adj}\Gamma_{\mathrm{i,j}}(\mathbf{p}) y_{\mathrm{i}}(t) = \det \Gamma_{\mathrm{i,j}}(\mathbf{p}) u(t), \tag{5.11}$$

where adj $\Gamma_{\mathrm{i,j}}$ denotes the adjugate matrix of $\Gamma_{\mathrm{i,j}}$ and $\mathbf{I}_{n_p}$ is the $n_p \times n_p$ identity matrix. Moreover, multiplying (5.6) by $\det \Gamma_{\mathrm{i,j}}(\mathbf{p})$ yields

$$\delta(\mathbf{p}) \det \Gamma_{\mathrm{i,j}}(\mathbf{p}) v_i(t) = \Gamma_{\mathrm{o,j}}(\mathbf{p}) \det \Gamma_{\mathrm{i,j}}(\mathbf{p}) u(t). \tag{5.12}$$

Next, substituting the left-hand side of (5.11) in (5.12) yields

$$\delta(\mathbf{p}) \det \Gamma_{\mathrm{i,j}}(\mathbf{p}) v_i(t) = \delta(\mathbf{p}) \Gamma_{\mathrm{o,j}}(\mathbf{p}) \mathrm{adj}\Gamma_{\mathrm{i,j}}(\mathbf{p}) y_{\mathrm{i}}(t). \tag{5.13}$$

Then the transmissibility whose pseudo input is y_{i} and whose pseudo output is v_i while vehicle i is under the fault class j satisfies [7]

$$v_i(t) = \mathcal{T}_{\mathrm{j}}(\mathbf{p}) y_{\mathrm{i}}(t), \tag{5.14}$$

where

$$\mathcal{T}_{\mathrm{j}}(\mathbf{p}) \stackrel{\Delta}{=} \frac{\delta(\mathbf{p})}{\delta(\mathbf{p})} \Gamma_{\mathrm{o,j}}(\mathbf{p}) \mathrm{adj} \Gamma_{\mathrm{i,j}}^{-1}(\mathbf{p}), \tag{5.15}$$

Note that (5.14) is a compact way of writing the differential equation (5.13). Cancellation of the common term $\delta(\mathbf{p})$ in (5.13) does not exclude any solutions of the differential equation (5.13), and thus is allowed. This cancellation yields a reduced order transmissibility operator. Although this result may seem straightforward to the reader, the proof has many technical details [7]. After canceling the common term $\delta(\mathbf{p})$, (5.13) becomes

$$\det \Gamma_{\mathrm{i,j}}(\mathbf{p}) v_i(t) = \Gamma_{\mathrm{o,j}}(\mathbf{p}) \mathrm{adj} \Gamma_{\mathrm{i,j}}(\mathbf{p}) y_{\mathrm{i}}(t), \tag{5.16}$$

which can be written as (5.14) with $\mathcal{T}_{\mathrm{j}}$ in (5.15) redefined as

$$\mathcal{T}_{\mathrm{j}}(\mathbf{p}) \stackrel{\Delta}{=} \Gamma_{\mathrm{o,j}}(\mathbf{p}) \Gamma_{\mathrm{i,j}}^{-1}(\mathbf{p}). \tag{5.17}$$

It is important to mention that the transmissibility operator $\mathcal{T}_{\mathrm{j}}(\mathbf{p})$ in (5.17) is independent of the excitation signals u that includes the fault dynamics, the initial condition $x(0)$, and the dynamics of the platoon denoted by the polynomial $\delta(\mathbf{p})$. Different faults will result in different B_{j} and therefore different transmissibility operator $\mathcal{T}_{\mathrm{j}}$. The faulty signal v_i can be estimated while the vehicle i is under unknown fault δ_i along with the velocity subset y_{i} from

$$\hat{v}_{i,\mathcal{T}}(t) = \mathcal{T}_{\mathrm{j}}(\mathbf{p}) y_{\mathrm{i}}(t). \tag{5.18}$$

Note that estimating $\hat{v}_{i,\mathcal{T}}$ depends on the velocity of the front vehicle v_{i+1} that occurs in a timely-manner after v_i. That is, estimating $\hat{v}_i$ depends on the future measurements of v_{i+1}, which results in a noncausal transmissibility operator $\mathcal{T}_{\mathrm{j}}$ (the second output channel in y_{i} will always have more zeros than the output channel of v_i). Define the discrepancy between the measured and estimated velocities

$$e_{\mathrm{j},\mathcal{T}}(t) = v_i(t) - \hat{v}_{i,\mathcal{T}}(t). \tag{5.19}$$

Consider discretizing $e_{\mathrm{j},\mathcal{T}}$ then compute

$$E_{\mathrm{j},\mathcal{T}}(k, w) \stackrel{\Delta}{=} \sqrt{\sum_{i=k}^{w+k} \| e_{\mathrm{j},\mathcal{T}}(i) \|}, \tag{5.20}$$

which represents the norm of the residuals over a sliding window of w steps width. The transmissibility-based fault classification algorithm depends on the discrepancy between the measured and estimated velocities. That is, if the fault class that affects the vehicle i is the same as the class $\mathcal{T}_{\mathrm{j}}$ was derived on, then the level of the normal of residual $E_{\mathrm{j},\mathcal{T}}$ is low, otherwise, the level of $E_{\mathrm{j},\mathcal{T}}$ will be high. Note that estimating $\hat{v}_{i,\mathcal{T}}$ depends on the velocity of the following vehicle v_{i+1} that occurs in a timely-manner after v_i. That is, estimating $\hat{v}_i$ depends on the future measurements of v_{i+1}, which results in a noncausal transmissibility operator $\mathcal{T}_{\mathrm{j}}$ (the second output channel in y_{i} will always have more zeros than the output channel of v_i). Therefore, transmissibility-based fault classification can only be used offline. To this end, we use DAgger in the rest of the paper to transfer the transmissibility experience in classifying the fault type to a supervised machine learning agent to classify platoon faults online.

5.1. Transmissibility identification

The transmissibility derivation introduced in Section 5 is based on the platoon model. However, in practice, the platoon model might be uncertain or even unknown. This section identifies the transmissibility in the structure of FIR models in case of unknown platoon model. Since sensor measurements are obtained in discrete time, we replace $\mathbf{p}$ by the forward shift operator $\mathbf{q}$. Accordingly, we consider identifying $\mathcal{T}$ in the $\mathbf{q}$ domain. The FIR model structure of $\mathcal{T}$ is given by

$$\mathcal{T}(\mathbf{q}) = \sum_{j=-d}^{r} H_j \mathbf{q}^{-j}, \tag{5.21}$$

where r, d denote the order of the causal and noncausal parts of the FIR model of $\mathcal{T}$, respectively, and $H_j \in \mathbb{R}^{1\times m}$ is the jth coefficient of the transmissibility operator $\mathcal{T}$. Let $\Theta = [H_{-d}, \dots, H_r]^{\mathrm{T}}$, then assume the system to be under healthy conditions for ϵ steps, then the least squares estimate of the transmissibility parameters Θ is given by

$$\hat{\Theta} = (\Phi\Phi^{\mathrm{T}})^{-1}\Phi\Psi, \tag{5.22}$$

$$\Psi = \begin{pmatrix} v_i(r) & \cdots & v_i(\epsilon - d) \end{pmatrix}^{\mathrm{T}}, \tag{5.23}$$

$$\Phi = \begin{pmatrix} \phi(r) & \cdots & \phi(\epsilon - d) \end{pmatrix}, \tag{5.24}$$

$$\phi(k) = \begin{pmatrix} y_{\mathrm{i}}(k + d) & \cdots & y_{\mathrm{i}}(k - r) \end{pmatrix}^{\mathrm{T}}. \tag{5.25}$$

where v_i and y_{i} are the platoon velocities as defined in Eqs. (5.1)–(5.2).

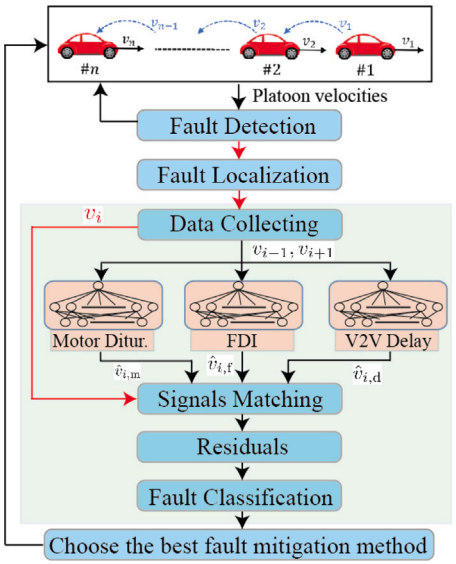

Fig. 3. Fault classification scheme. Three fault estimations are considered, each for a different fault class. Then the closest fault estimation to the measured one is considered as the actual fault. The DAgger algorithm in Algorithm 1 is used to train the estimators.

6. Novice classification policy training using imitation learning

As mentioned earlier, the difference between the expert and novice policies is the estimation method. The novice policy estimation is based on a neural network for each fault class as shown in Fig. 3, where the fault detection and localization are considered in [4]. Inspired by the work introduced in [10,27], we use DAgger to transfer the estimation experience from the transmissibility operators to the neural networks as in Algorithm 1, then we use the trained novice policy $\pi_{\mathcal{N}}$ for online classification as in Fig. 3. Moreover, DAgger will give the ability to adapt to new fault classes without the need to re-train the diagnosis model from the beginning.

The training procedure starts with a pre-recorded labeled data set D_0 and the initial novice policy $\pi_{\mathcal{N}_1}$ is trained on it. For epochs $i = 1, \dots, K_e$, DAgger is then implemented to update the novice policy from $\pi_{\mathcal{N}_1}$ to $\pi_{\mathcal{N}} = \pi_{\mathcal{N}_{K_e+1}}$. Each epoch represents an update in the novice policy such that epoch i updates $\pi_{\mathcal{N}_i}$ to $\pi_{\mathcal{N}_{i+1}}$. Each epoch consists of K_r rollouts (runs), each of these rollouts updates the training data set D. Each run is K_k time steps long. Dagger allows the novice classification policy to work as long as the novice decision taken about the fault class $d_{\mathcal{N}}$ is the same as the decision taken by the expert policy $d_{\mathcal{T}}$. Otherwise, the algorithm follows the expert classification policy and the decisions taken by the expert are recorded and added to the training data set D, and then the novice policy is trained on the updated training data set by the end of the epoch.

6.1. Data collection and neural networks structure

To collect the pre-recorded data set D, the platoon model in (4.5)–(4.6) was constructed with five vehicles such that the first and third vehicles follow the first CAV model, the second and fifth vehicles follow the second CAV model, and the fourth vehicle follow the third CAV model. The desired velocity of the platoon v_1^* was set to band-limited white noise with zero mean and maximum absolute amplitude of 30 m/s. The frequency of the desired velocity is constant during each run and varies randomly between $1-50$ s from one run to another. Only one fault class was emulated to a different vehicle in each run.

We recorded the faulty vehicle's velocity signal for 6000 runs, such that 2000 runs with each class of the three faults. Each signal was 50 s long with sampling time 0.1 s. Note that the steady transition platoon movement case is included when the velocity signal is 50 s long and the desired velocity frequency is 0.02 Hz. Moreover, the case where the platoon moves with oscillatory velocity is also included when the desired velocity frequency is relatively high. We create three deep neural networks, one for each fault class. Each neural network consists of six hidden layers, where each hidden layer consists of 128 neurons. The sixth layer is then followed by a scaling layer and a regression layer. Each neural network estimates the faulty signal v_i for a different fault class.

The integration between transmissibility and DAgger has the ability to adapt to new fault classes without the need to re-train a diagnosis model from the beginning each time we wish to include a new fault class. This can be simply done by creating a new transmissibility operator for the new fault class, then DAgger will add the new experience to the previously trained diagnosis model. Please note that transmissibility operators are mathematical estimators (similar to transfer functions) that do not require any intense training or computations. The following procedure shows how a new fault class can be added to the diagnosis model using the transmissibility-DAgger integration:

Algorithm 1: Novice policy training using DAgger.

$\mathcal{D} \longleftarrow D_0$
train $\pi_{\mathcal{N}_1}$ on $\mathcal{D}$
for epoch $i = 1 : K_e$ **do**
  for rollout $j = 1 : K_r$ **do**
    novice is in charge
    for time step $k = 1 : K_k$ **do**
      if $d_{\mathcal{N}} = d_{\mathcal{T}}$ **then**
        keep novice in charge
      else
        expert takes charge
        record expert labels into D_j
      end
    end
    $\mathcal{D} = \mathcal{D} \cup D_j$
  end
  train $\pi_{\mathcal{N}_{i+1}}$ on $\mathcal{D}$
end

1. Derive the new fault input vector $B_{\delta,\mathrm{j}}$ for the new fault class as in Eqs. (4.7) and (4.8).
2. Define a new transmissibility operator using Eq. (5.18) in the revised manuscript for the new fault. This can be simply done by substituting the new fault input vector $B_{\delta,\mathrm{j}}$ in Eqs. (5.7) and (5.8).
3. Run DAgger algorithm in Algorithm 1 of the revised manuscript. When it reaches the if statement, $d_{\mathcal{T}}$ will follow the new fault class and thus $d_{\mathcal{N}} \neq d_{\mathcal{T}}$. This will result in the new labels being recorded in D_j, and then the diagnosis model will be adapted with D_j.

6.2. Faults emulation

Following the model in (4.7), we set δ_i to band-limited white noise with zero mean and unit variance. The faults emulation differs in constructing $B_{\delta,\mathrm{j}}$ according to the models (4.1)–(4.4). The actuator fault is emulated in vehicle #2 as it follows the bond graph model. FDI and V2V delay faults were emulated in vehicles (#2 – #4) as these vehicles send their velocities through the V2V communication links.

6.3. Models training

After collecting a set of 6000 signals labeled from the expert, we used this set as an initialization for the novice policy training $\mathcal{D}_0$. Training the novice policy on $\mathcal{D}_0$ produces the initial novice policy $\pi_{\mathcal{N}_1}$, then we use DAgger to enhance $\pi_{\mathcal{N}_1}$. We set the number of DAgger epochs to $K_e = 40$, that is, the novice policy is enhanced 40 times from $\pi_{\mathcal{N}_1}$ to $\pi_{\mathcal{N}_{41}}$. The number of rollouts was set in each epoch to $K_r = 100$, that is, each epoch consists of 100 faulty signals. Each signal (rollout) is 50 s long with 0.1 s sampling time, that is, $K_k = 501$ steps. To conclude the training procedure, the novice policy was first trained on a set of 6000 labeled signals, and then the novice policy was enhanced 40 times over a set of 4000 signals.

6.4. Fault classification

The enhanced trained machine learning agent that consists of three neural networks is then used to obtain three estimations of the faulty signal, one for each fault class. Each of the estimations will be compared to the measured one, and the closest estimation to the measured behavior is assumed to be the fault class. This comparison represents the fault classification, which is determining what class of faults the measured fault belongs too. To distinguish the machine learning estimation from the transmissibility estimation, we denote the machine learning estimation with $\hat{v}_{i,\mathcal{N},\mathrm{j}}$ where $\mathrm{j} \in \{\mathrm{m}, \mathrm{f}, \mathrm{d}\}$. Similar to the transmissibility-based classification, define the discrepancy between the measured and estimated velocities

$$e_{\mathrm{j},\mathcal{N}}(t) = v_i(t) - \hat{v}_{i,\mathcal{N}}(t). \tag{6.1}$$

and compute the norm of residuals over a sliding window with w steps width

$$E_{\mathrm{j},\mathcal{N}}(k, w) \overset{\Delta}{=} \sqrt{\sum_{i=k}^{w+k} \|e_{\mathrm{j},\mathcal{N}}(i)\|}. \tag{6.2}$$

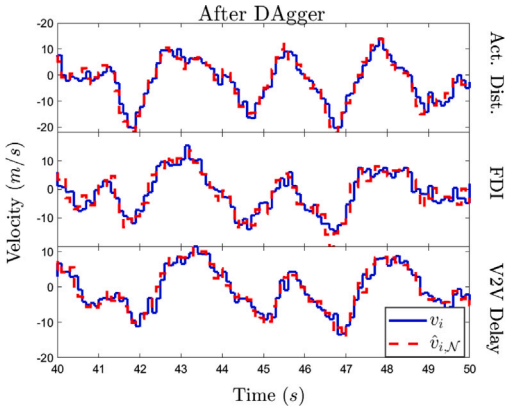

Fig. 4. Simulation Results: A simulation example of three measured faulty signals, each with a different fault class, and their online estimations using the DAgger trained machine learning agent.

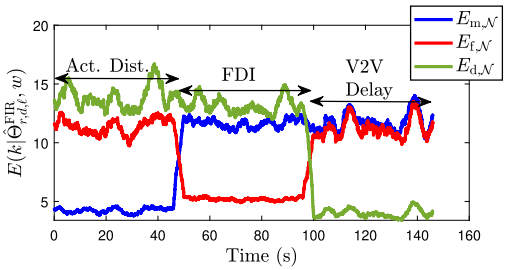

Fig. 5. Simulation Results: Norm of residuals over a sliding window using (6.2) where the emulated fault switches every 50 s. Note that the level of the norm of residual that is based on the same emulated fault is lower than the other two residuals.

The classification policy is then based on the level of norm or residuals. That is, if the norm of residual of the fault class that $E_{\mathrm{j},\mathcal{N}}$ was obtained on is the same as the actual fault acting on the system then the level of $E_{\mathrm{j},\mathcal{N}}$ should be low, otherwise the level of $E_{\mathrm{j},\mathcal{N}}$ will be high.

It is important to notice here that the proposed approach does not identify faults. The proposed approach is independent of the fault dynamics. Thus, even when the fault dynamics are unknown, the proposed approach does not require identifying them. This is due to the transmissibility's ability of estimating the faulty behaviors through the velocity measurements only.

7. Simulation example

The same platoon considered for the training in Section 6.1 is considered for simulation testing. All faults are emulated as in 6.2. The faulty velocity signals were recorded for the testing purposes by setting $\delta_{C,i}, \delta_{f,i}$, and $\delta_{v,i}$ to different random values. Fig. 4 shows an example of three measured faulty signals, each with a different fault class, and their online estimation using the DAgger trained machine learning agent $\hat{v}_{i,\mathcal{N}}$. We can see that by using DAgger, the novice policy was able to obtain a close estimation of the faulty signals under different fault classes. To show the online classification more clearly, an example run was conducted by running the platoon for 150 s. For the first 50 s, the motor disturbances fault was emulated, then we switched the emulated fault from motor disturbances to FDI for 50 s, and then switched it again from FDI to V2V delay for 50 s. The recorded signal was then compared with the three novice estimations as in Fig. 3, and the three residuals were obtained between the recorded signal and the three estimations. The norms of residuals over sliding windows are then obtained as in (6.2) with $w = 100$ steps for the three residuals as shown in Fig. 5. We can see from Fig. 5 that the level of the norm of residual that is based on the same emulated fault is lower than the other two residuals.

8. Experimental testing results

We consider the experimental setup shown in Fig. 6 consisting of three autonomous differential Quanser robots called Qbots 2e. Each Qbot consists of two coaxial wheels, where each wheel is driven by a DC motor that is controlled using a closed-loop inverse kinematic controller. If the desired angular velocity is zero, then both wheels velocities are equal and the Qbot moves forward or backward in a straight line. Qbot1 receives the excitation signal from a computer through wireless communication, and Qbot2

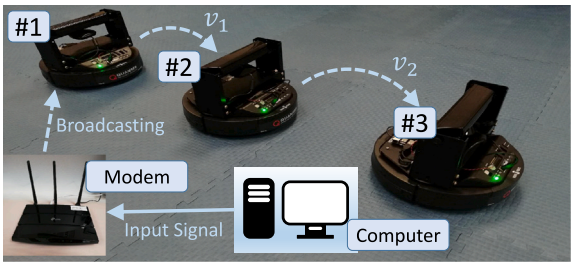

Fig. 6. The experimental setup. Qbot1 receives the desired velocity from the computer while Qbot2, and third Qbot3 receive the desired velocity from the preceding Qbot via V2V communication.

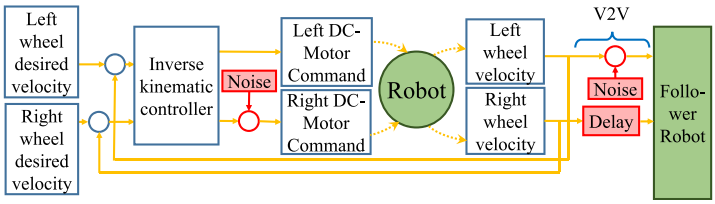

Fig. 7. Experimental emulation of the platoon faults. Three faults are considered separately, as represented by the red blocks. A physical fault is represented in actuator disturbances, and two cyber faults are FDI and V2V delay.

is connected with Qbot1 via a V2V communication channel. Similarly, Qbot3 is connected with Qbot2 via a V2V communication channel. We run the setup by setting the first Qbot desired velocity to band-limited white noise with maximum absolute velocity of 0.15 m/s.

8.1. Qbot internal noise emulation

We consider injecting Gaussian noise in the command signal of the DC-motor that derives the right wheel of the Qbot as shown in Fig. 7. This makes the velocities of the wheels in Qbot3 not equal, which results in a 2-D motion of Qbot3 (i.e. a physical fault). We set the injected noise to band-limited white noise with maximum absolute amplitude of 0.05 m/s. This fault was emulated in the second and third Qbots. We recorded 40 runs with Qbot internal disturbances, each is 50 s long with 0.1 s sampling time. The disturbances signal injected is adjusted to change its frequency after each run, such that the disturbances' frequency varies randomly between $1 - 0.02$ Hz.

8.2. FDI emulation

Following (4.2), we set $\delta_{f,i}$ to band-limited white noise with maximum absolute amplitude of 0.05 m/s as shown in Fig. 7. FDI fault was emulated in the first and second Qbots as these robots send their velocities through the V2V communication links. We recorded 40 runs with FDI, each is 50 s long with 0.1 s sampling time. The disturbances signal $\delta_{f,i}$ also changes its frequency after each run to vary randomly between $1 - 0.02$ Hz.

8.3. Communication time delay emulation

Following (4.3), we set $\tau_{v,i}$ to a constant value between $1 - 3$ s, as shown in Fig. 7. Similar to the FDI emulation, V2V communication time delay was also emulated in the first and second Qbots as these robots send their velocities through the V2V communication links. We recorded 40 runs with V2V delay fault, each is 50 s long with 0.1 s sampling time. The delay value $\tau_{v,i}$ changes from one run to another randomly between $1 - 3$ s.

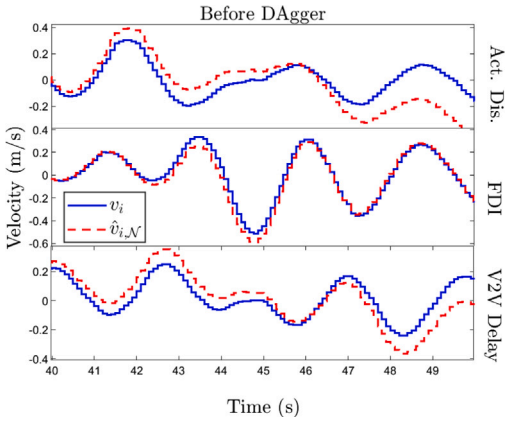

Fig. 8. Experimental Results: An experimental example of three measured faulty signals, each with a different fault class, and their online estimations using the novice policy before implementing DAgger.

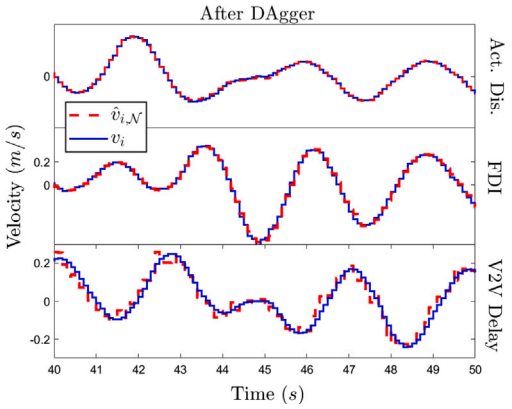

Fig. 9. Experimental Results: An experimental example of three measured faulty signals, each with a different fault class and their online estimations using the DAgger trained machine learning agent.

8.4. Fault classification

Fig. 8 shows test runs of three measured faulty signals, each with a different fault class, and their online estimation while the machine learning agent is only trained on $\mathcal{D}_0$. That is, the estimations in Fig. 8 are before enhancing the training using DAgger. Fig. 9 shows test runs of three measured faulty signals, each with a different fault class, and their online estimation using the DAgger trained machine learning agent from Section 6. Similar to the simulation example presented in Section 7, we run the setup for 270 s, actuator disturbances fault was emulated for the first 90 s, then we switch the emulated fault to FDI for 90 s, and then switch it to V2V delay for the last 90 s. Fig. 10 shows the norm of residuals over a sliding window computed using (6.2) with $w = 100$ steps, where the residuals are between the measured faulty signal and the three estimations. Note from Fig. 10 the level of the norm of residual that is based on the same emulated fault is lower than the other two residuals. Moreover, Fig. 11 shows the average of the norms of residuals for the offline classifier (transmissiblity), online classifier before DAgger, and the online classifier after using DAgger to enhance the online classification with the transmissiblity.

9. Results comparisons

We consider comparing the current results with nine classification trees. These classification trees are: Support Vector Machine (SVM), Naive Bayes (NB), Quadratic Discriminant (QD), and K-Nearest Neighbors (KNN). SVM is configured with Quadratic (Q-SVM)

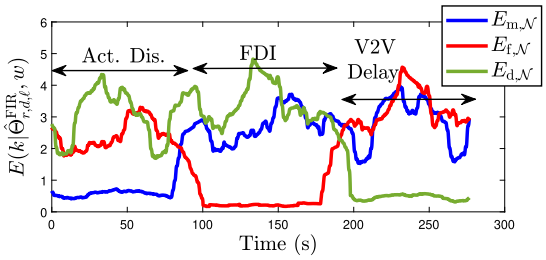

Fig. 10. Experimental Results: Norm of residuals over a sliding window using (6.2) where the emulated fault switches every 90 s. Note that the level of the norm of residual that is based on the same emulated fault is lower than the other two residuals.

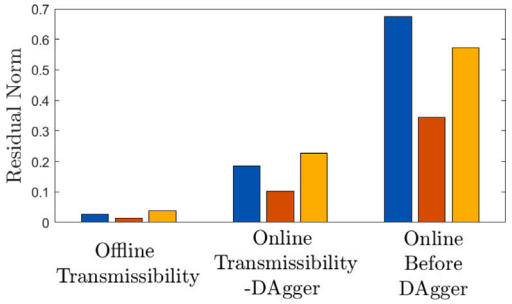

Fig. 11. Average of the norms of residuals for the offline classifier (transmissiblity), online classifier before DAgger, and the online classifier after using DAgger to enhance the online classification with the transmissiblity.

Table 2
Classification accuracy for the experimental test with the number of correctly classified signals.

Method	Act. Dist.	FDI	Delay	# Success runs	Accuracy
Transmissibility-DAgger	37/40	40/40	38/40	115/120	95.8%
Q-SVM	28/40	31/40	35/40	94/120	78.3%
C-SVM	33/40	25/40	34/40	92/120	76.6%
QD	38/40	37/40	28/40	103/120	85.8%
G-NB	36/40	29/40	31/40	96/120	80%
CE-KNN	29/40	34/40	33/40	96/120	80%
C-KNN	25/40	32/40	31/40	88/120	73.3%
RF	33/40	29/40	31/40	93/120	77.5%
BE	26/40	27/40	19/40	72/120	60%

and Cubic (C-SVM) Kernel functions separately in two separate models. NB is configured with Gaussian Kernel type (G-NB). KNN is configured with Coarse Euclidean (CE-KNN) and Cubic (C-KNN) distance metrics separately in two separate models. Moreover, two ensembles techniques are: Random Forest Classifier (FC), and Boosted Ensemble (BE). After comparing all model structures, the fault classification is shown to be most effective when Q-SVM is considered for the first classifier and QD for the second classifier. All of these models are first trained on the data set obtained in Section 6.2, and then used to classify the data recorded from the experimental setup.

We recorded 120 runs, 40 runs with each Qbot fault class. Each run is 50 s long with 0.1 s sampling time. The actuator disturbances signal injected is adjusted to change its frequency after each run, such that the disturbances' frequency varies randomly between $1-0.02$ Hz. Similarly, the FDI disturbances signal also changes its frequency after each run to vary randomly between $1-0.02$ Hz. The delay value $\tau_{v,i}$ for the V2V delay changes from one run to another randomly between $1-3$ s. We then use the DAgger trained classification scheme presented in Fig. 3 to classify the fault class in each of the 120 runs. The proposed approach is then compared with the six classification trees. Table 2 lists the successfully classified runs, where the overall classification success achieved using the proposed approach is 95.8%, while the second accuracy achieved was 85.8% using QD. Fig. 12 shows the average of the norms of residuals for each fault class of the methods in Table 2. The norm of residual was computed between the measured faulty velocity signals and their ML-based estimation.

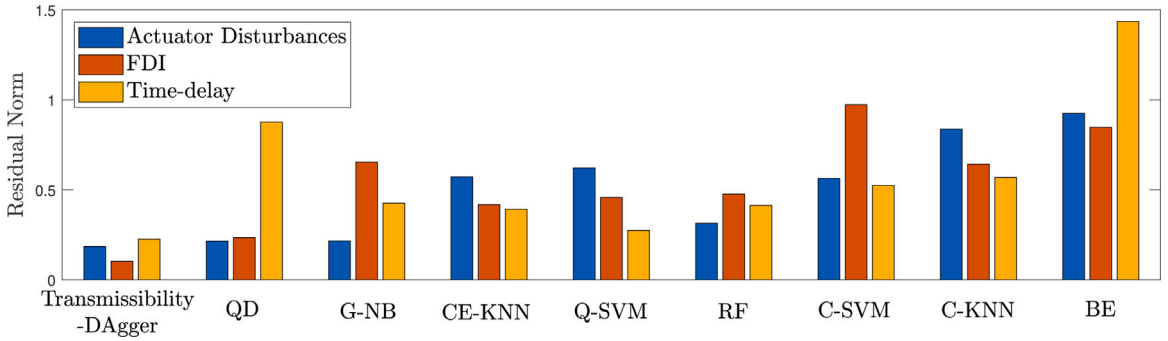

Fig. 12. Average of the norms of residuals for each fault class of the methods in Table 2. The norm of residual was computed between the measured faulty velocity signals and their ML-based estimation.

10. Conclusions

Fault mitigation techniques could be faster and more reliable with known fault class in CAV platoon. In this paper we used transmissibility operators, which are relationships that relate a set of velocities with another in the platoon, to classify the faults. However, transmissibility operators are shown to be noncausal and therefore can only be used offline. We then used DAgger, which is an extension in imitation learning that transfers experience from the expert agents (transmissibility operators) to novice agents (untrained neural network). A heterogeneous CAV platoon was modeled with three different faults separately. These faults are actuator disturbances, FDI attack, and V2V communication time delay. The classification scheme depends on estimating the faulty signal in case of each fault class. Next, the measured faulty signal is then compared with the three estimations and the closer fault estimation to the measured one is considered as the actual fault on the platoon. After using DAgger to train the novice agent, we test it on the simulation model, and then we apply it on an experimental setup that consists of three autonomous robots. The proposed results are compared with six different machine learning classification models. The overall classification accuracy achieved using the proposed approach was 95.8% for the experiment.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

All of the sources of funding for the work described in this publication are acknowledged below:
Natural Sciences and Engineering Research Council of Canada.

Data availability

No data was used for the research described in the article.

References

[1] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Detection of contact force transients in robotic assembly, in: IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 962–968.
[2] A. Petrillo, A. Pescape, S. Santini, A secure adaptive control for cooperative driving of autonomous connected vehicles in the presence of heterogeneous communication delays and cyberattacks, IEEE Trans. Cybern. 51 (2021) 1134–1149.
[3] A. Prasad, J.B. Edward, K. Ravi, A review on fault classification methodologies in power transmission systems: Part—I, J. Electr. Syst. Inf. Technol. 5 (2018) 48–60.
[4] A. Khalil, M. Al Janaideh, K.F. Aljanaideh, D. Kundur, Fault detection, localization, and mitigation of a network of connected autonomous vehicles using transmissibility identification, in: American Control Conference, ACC, 2020, pp. 386–391.
[5] T. Han, Q. Hu, H.-S. Shin, A. Tsourdos, M. Xin, Incremental twisting fault tolerant control for hypersonic vehicles with partial model knowledge, IEEE Trans. Ind. Inform. (2021).
[6] Y. Lu, R. Su, C. Zhang, L. Qiao, Event-triggered adaptive formation keeping and interception scheme for autonomous surface vehicles under malicious attacks, IEEE Trans. Ind. Inform. (2021).
[7] K.F. Aljanaideh, D.S. Bernstein, Time-domain analysis of sensor-to-sensor transmissibility operators, Automatica 53 (2015) 312–319.
[8] A. Magdaleno, A. Lorenzana, A transmissibility-based procedure to estimate the modal properties of an on-board tuned mass damper, Mech. Syst. Signal Process. 135 (2020) 106378.
[9] N.M. Maia, R.A. Almeida, A.P. Urgueira, R.P. Sampaio, Damage detection and quantification using transmissibility, Mech. Syst. Signal Process. 25 (7) (2011) 2475–2483.

[10] K. Menda, K. Driggs-Campbell, M.J. Kochenderfer, Ensembledagger: A bayesian approach to safe imitation learning, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2019, pp. 5041–5048.
[11] H. Daumé, J. Langford, D. Marcu, Search-based structured prediction, Mach. Learn. 75 (3) (2009) 297–325.
[12] A.R. Javed, M. Usman, S.U. Rehman, M.U. Khan, M.S. Haghighi, Anomaly detection in automated vehicles using multistage attention-based convolutional neural network, IEEE Trans. Intell. Transp. Syst. 22 (7) (2020) 4291–4300.
[13] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, F.-Y. Wang, Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey, IEEE Trans. Intell. Transp. Syst. 22 (6) (2021) 3234–3246.
[14] L.S. Lopes, L.M. Camarinha-Matos, Learning to diagnose failures of assembly tasks, in: Artificial Intelligence in Real-Time Control 1994, 1995, pp. 97–103.
[15] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, J. Peters, Towards learning hierarchical skills for multi-phase manipulation tasks, in: IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 1503–1510.
[16] A. Rodriguez, D. Bourne, M. Mason, G.F. Rossano, J. Wang, Failure detection in assembly: Force signature analysis, in: IEEE International Conference on Automation Science and Engineering, 2010, pp. 210–215.
[17] X. Song, H. Liu, K. Althoefer, T. Nanayakkara, L.D. Seneviratne, Efficient break-away friction ratio and slip prediction based on haptic surface exploration, IEEE Trans. Robot. 30 (2013) 203–219.
[18] G.R. Moreira, G.J. Lahr, T. Boaventura, J.O. Savazzi, G.A. Caurin, Online prediction of threading task failure using convolutional neural networks, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2018, pp. 2056–2061.
[19] J. Rojas, S. Luo, D. Zhu, Y. Du, H. Lin, Z. Huang, W. Kuang, K. Harada, Online robot introspection via wrench-based action grammars, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2017, pp. 5429–5436.
[20] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, D. Li, Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment, IEEE Trans. Ind. Inform. 14 (9) (2018) 4224–4231.
[21] A. Saxena, A. Saad, Evolving an artificial neural network classifier for condition monitoring of rotating mechanical systems, Appl. Soft Comput. 7 (1) (2007) 441–454.
[22] S.S. Roy, S. Dey, S. Chatterjee, Autocorrelation aided random forest classifier-based bearing fault detection framework, IEEE Sens. J. 20 (18) (2020) 10792–10800.
[23] V. Jackins, S. Vimal, M. Kaliappan, M.Y. Lee, AI-based smart prediction of clinical disease using random forest classifier and naive Bayes, J. Supercomput. 77 (5) (2021) 5198–5219.
[24] S. Ross, N. Melik-Barkhudarov, K.S. Shankar, A. Wendel, D. Dey, J.A. Bagnell, M. Hebert, Learning monocular reactive uav control in cluttered natural environments, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1765–1772.
[25] The MAt, Imitate nonlinear mpc controller for flying robot, [Online]. Available: https://www.mathworks.com/help/reinforcement--learning/ug/imitate--nonlinear--mpc--controller--for--flying--robot.html.
[26] J. Zhang, K. Cho, Query-efficient imitation learning for end-to-end autonomous driving, 2016, arXiv preprint arXiv:1605.06450.
[27] M. Kelly, C. Sidrane, K. Driggs-Campbell, M.J. Kochenderfer, Hg-dagger: Interactive imitation learning with human experts, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 8077–8083.
[28] Y. Jiang, S. Yin, J. Dong, O. Kaynak, A review on soft sensors for monitoring, control and optimization of industrial processes, IEEE Sens. J. 21 (2020) 12868–12881.
[29] X. He, E. Hashemi, K.H. Johansson, Distributed control under compromised measurements: Resilient estimation, attack detection, and vehicle platooning, 2020, arXiv preprint arXiv:2010.09661.
[30] Y. Hu, S. Zhang, Y. Yan, L. Wang, X. Qian, L. Yang, A smart electrostatic sensor for online condition monitoring of power transmission belts, IEEE Trans. Ind. Electron. 64 (9) (2017) 7313–7322.
[31] W. Yan, D. Tang, Y. Lin, A data-driven soft sensor modeling method based on deep learning and its application, IEEE Trans. Ind. Electron. 64 (5) (2016) 4237–4245.
[32] Z. Gao, X. Liu, M.Z. Chen, Unknown input observer-based robust fault estimation for systems corrupted by partially decoupled disturbances, IEEE Trans. Ind. Electron. 63 (4) (2015) 2537–2547.
[33] R. Rajamani, Vehicle Dynamics and Control, Springer Science & Business Media, 2011.
[34] B. Wang, M. Xu, L. Yang, Study on the economic and environmental benefits of different EV powertrain topologies, Energy Convers. Manage. 86 (2014) 916–926.
[35] D.C. Karnopp, D.L. Margolis, R.C. Rosenberg, System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems, John Wiley & Sons, 2012.
[36] A. Khalil, M. Al Janaideh, K.F. Aljanaideh, D. Kundur, Output-only fault detection and mitigation of networks of autonomous vehicles, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp. 2257–2264, 2020.
[37] G. Zhang, H. Zhang, X. Huang, J. Wang, H. Yu, R. Graaf, Active fault-tolerant control for electric vehicles with independently driven rear in-wheel motors against certain actuator faults, IEEE Trans. Control Syst. Technol. 24 (5) (2015) 1557–1572.
[38] W. Jeon, A. Zemouche, R. Rajamani, Resilient control under cyber-attacks in connected ACC vehicles, in: ASME 2019 Dynamic Systems and Control Conference, American Society of Mechanical Engineers Digital Collection, 2019.