Fault Classification in Mixed Autonomous and Human-Driven Vehicle Platoons

by

Theodore Wu

Co-supervisors: Prof. Deepa Kundur &
Prof. Mohammad Al Janaideh

April 2023

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Connected Autonomous Vehicle (CAV) platoons are an emergent technology that promises increased efficiency in modern transportation infrastructure. However, the integration of cyber-communication with physical vehicle components, combined with the need for CAVs to integrate into a human-driver environment, creates vulnerabilities that can compromise platoon safety. Different faults can be introduced in all layers of the platoon system or even by impaired drivers, each requiring different fault resolution methods. Identifying the fault class becomes a critical fault management step that facilitates the selection of the best mitigation strategy. This paper introduces two Multi-Head Attention (MHA) machine learning models to perform fault classification on a set of five faults and abnormalities classes in mixed autonomous and human-driven vehicle platoons. CAVs can face actuator faults, False Data Injection (FDI) attacks, and Denial-of-Service (DoS) attacks, while human drivers could exhibit abnormal distracted or drunk behaviour. The proposed MHA networks are designed to classify faulty vehicle behaviour by identifying important time steps over long sequences of platoon velocity profiles. The MHA networks are trained on a mixed platoon simulation model and validated on multiple test sets containing noisy platoon velocity profile measurements. We demonstrate that the inclusion of attention allows our MHA approach to maintain an accuracy of 90% in high sensor noise environments, which vastly outperforms many benchmark models that do not use attention.

Acknowledgements

The author would like to thank Prof. Kundur and Prof. Al Janaideh for their guidance in developing this thesis work; Khalil and Satvick for their help in constructing the platoon simulation model; and his friends, family, and lots of coffee for their support in helping the author reach the end of his undergraduate degree.

Table of Contents

List of Figures

List of Tables

1 Introduction

Autonomous vehicle (AV) technology has long attracted research interest from a wide range of domains for its potential to alter modern transportation systems, promising numerous advantages in traffic safety, energy consumption, and congestion reduction compared to traditional human-driven vehicles [3]. While autonomous driving systems such as adaptive cruise control (ACC) and intelligent collision avoidance have been introduced in many in-market vehicles [4], these systems are inherently limited by their inability to make optimal decisions based on unobservable conditions of the external environment. For instance, a sudden braking maneuver initiated at the start of a traffic chain would propagate a sequence of rapid braking responses through the chain, resulting in unnecessary traffic slowdowns [5]. To enable greater environmental awareness, the most recent wave of development equips AVs with wireless communication capabilities to form connected autonomous vehicles (CAVs), in which a platoon of AVs can communicate with other connected vehicles as well as with surrounding roadside infrastructure. As the next generation of self-driving vehicles, CAVs are expected to further decrease traffic congestion, mitigate traffic jams, and enable efficient route planning [6]. However, introducing communication to AVs creates a complex cyber-physical system (CPS), which possesses a broader attack surface that leaves the system vulnerable to both cyber-attacks and physical system faults [7]. Further, the real-world operation of CAVs necessitates cooperation with unconnected and unpredictable human drivers, who add additional risks to the safe operation of CAVs. It thus becomes paramount to develop security measures that can diagnose and repair any system faults that may occur during CAV operation.

Many methods exist to both detect and mitigate faults in CAV systems for various cyber and physical faults. A fault detection technique can be classified as either model-based, which detects faults by modelling the system dynamics, or process-based, which detects faults using historical signal measurements [8]. These detection approaches are designed to be sensitive to any form of abnormal system behaviour. On the other hand, fault mitigation techniques are typically designed to mitigate a specific fault or abnormality class and may not be universally applicable [9]. To bridge the gap between generalizable fault detection and fault-specific risk mitigation approaches, fault classification is a critical task that maps a faulty CAV system to a specific fault type, thus enabling the selection of appropriate mitigation techniques [1]. Despite its importance, few works investigate the challenge of distinguishing detected faults in CAV platoons. There remains a research gap for an efficient fault classification mechanism to ensure the safety and stability of CAV platoons, especially when they are coexisting with unpredictable human-driven vehicles.

This work contributes a new multi-head attention (MHA) network fault classification algorithm based on transformer networks [10] for mixed human-driver and CAV platoons. Our work considers five potential fault and abnormality types: (i) actuator faults, (ii) false data injection (FDI), (iii) denial-of-service (DoS) attacks, (iv) distracted human drivers, and (v) drunk human drivers. These faults are selected for their full coverage of physical faults, cyber-communication attacks, and human driver abnormalities. We develop a platoon simulation model for these fault and abnormality classes and extract sequences of vehicle velocity measurements, which allows our classifier to be insensitive to the platoon model dynamics. This dataset of velocity profiles is used to train and evaluate the MHA approach for the classification of the five fault and abnormality classes. Our architecture is compared with existing machine learning-based fault classification approaches from similar domains on a test environment simulating the effect of sensor data anomalies and random external noise.

2 Literature Review

To position this work in the field of CAV security, this literature review provides an overview of historical developments in AV technology. We emphasize the paradigm shift from AVs to CAVs and highlight the resulting increase in the system's vulnerability. We document existing cyber-security methods for fault management of the CAV system, which indicates that the fault classification task is under-examined in the current literature. To conclude, we draw inspiration from existing fault classification approaches in related domains to motivate the design of our proposed machine learning (ML) method.

2.1 Introduction to Autonomous Vehicles

Our modern roadways and transportation infrastructure embody an impressive feat of engineering that supports millions of users each day. But as the demand for transportation continues to grow, new strategies for expanding traffic capacity will be necessary. Beginning in the late 1950s, autonomous vehicle (AV) technology was proposed as a solution for increasing the capacity and efficiency of existing roadways [11]. Since then, research in AVs has experienced resurgent waves of development, with the latest wave of scientific interest fueled by the DARPA grand challenge [12] and a public announcement of research investment from Google [5]. In 2016, automotive companies were estimated to spend over 100 billion CAD on fostering research and development in AV technology [3]. That estimate has since grown.

Broadly, AVs are vehicular systems that are equipped with a degree of driving function automation [5]. To enable these automated driving functions, an AV will deploy a combina-

Table 1: Overview of the SAE International J3016 taxonomy of driving automation [2]. Level 1 represents the most basic autonomous driving systems with limited automation capabilities, while Level 5 represents near-full automation. The highest level of automation achieved in a vehicle thus far is Level 3.

Level	**Summary**
0	No driving automation.
1	System controls one of lateral or longitudinal motion with driver control over other functions.
2	System controls full vehicle motion but requires the driver to monitor the road and supervise the automation system.
3	Sustained full control of the vehicle by the system with an expectation for the driver to respond to requests to intervene during failures.
4	Sustained full control of the vehicle by the system in a limited set of driving situations without expectation for the driver to respond.
5	Sustained full control of the vehicle by the system in all driving situations without expectation for the driver to respond.

tion of sensors (e.g. camera, lidar, radar, etc.) to observe the surrounding environment, after which a software system is used to map the input sensor data to appropriate vehicle control actions [3]. The Society of Automotive Engineers (SAE) International defines a five-level taxonomy [2] of an AV's driving automation capabilities, as summarized by Table 1. While the real-world impacts of AVs on the transportation system will vary depending on the level of automation achieved, AVs operating at an intermediate level of automation are expected to introduce reductions in collisions, vehicular congestion, travel time, and environmental impact [3].

Notwithstanding the proposed benefits, AVs face critical limitations caused by an inability to communicate with vehicles beyond the preceding vehicle. For instance, Milanes *et al.* [13] observe that a purely autonomous vehicle cannot anticipate traffic shock waves caused by sudden braking and can actually produce less stable car-following behaviour compared with human drivers. However, when inter-vehicular communication is enabled, as is the case with cooperative adaptive cruise control (CACC) [13], the effects of such traffic shock waves are significantly dampened. A white paper published by Siemens [14] offers an industry perspective that further corroborates the need for connectedness in enabling AV capabilities beyond Level 1-3 systems. Under the new "connected" paradigm, AVs communicate with their surroundings in one of three main configurations: (i) vehicle-to-vehicle (V2V) communication,

which enables cooperative traffic maneuvers; (ii) vehicle-to-infrastructure (V2I) communication, which enables advanced routing, planning, and access control; and (iii) vehicle-to-cloud (V2C) communication, which enables connection to virtually all devices, for example, phones carried by pedestrians [5].

This shift towards connected autonomous vehicles (CAV) represents the next generation of AVs and offers significant improvements over regular AV technology. Shladover *et al.* [15] consider a platoon of AVs in a CACC setting and measure substantial increases in traffic lane capacity simulations at moderate to high concentrations of CACC vehicles within the traffic composition. Ploeg *et al.* [16] provide empirical evidence for the reduction of traffic congestion enabled by a CACC configuration of test passenger vehicles. They also note the negligible effect of the non-connected ACC setting on improving traffic congestion. From a safety perspective, Ye *et al.* [17] observe reductions in the frequency of dangerous situations when CAVs are introduced to the traffic flow at even a meagre 25% adoption rate. Regarding energy efficiency, Jin *et al.* [18] show that AVs consume less energy when connected with V2V communication and can even produce energy savings in human vehicles following the platoon. CAVs clearly possess distinct advantages over unconnected AV technology and are thus expected to be the standard topology in future automotive markets [19].

2.2 Security Implications for Connected Autonomous Vehicles

The integration of communication protocols in CAVs advances vehicle automation capabilities, but at the same time introduces new complexity to the system. The physical components of the vehicle coupled with autonomous decision-making software and vehicular communication networks allow CAVs to be modelled as a cyber-physical system (CPS). However, the complex nature of such systems creates a threat surface with many potential vulnerabilities on both the physical and communication modules of CAVs [20, 21]. Pham *et al.* [22] conduct a broad survey of CAV attack models, identifying a taxonomy of eight possible attack targets whose impact spans vehicle control, environment sensing, and communication.

The range of avenues through which attacks can occur poses a serious threat to the safe operation of various autonomous driving functions. As an example, within the context of CACC, automated vehicles can only be considered safe and efficient if the platoon is string stable [23], which ensures that the effects of a traffic disturbance (e.g. a sudden braking maneuver) are not amplified along the chain of vehicles within the platoon. However, Alipour-Fanid *et al.* [24] show that jamming attacks, such as false data injection (FDI) [25] and denial-of-service (DoS) [26], made to the communication channel of a CACC configuration of CAV vehicles can compromise string stability and create unsafe vehicle following

distances when the attack is conducted close to the lead vehicle of the platoon. A further complication is introduced when we consider the need for such CAV platoons to be integrated with human drivers during real-world deployment. Dadras *et al.* [27] evaluate a connected vehicle platoon in the case of an adversarial leader vehicle and show that the platoon can be made to oscillate at a resonant frequency with the appropriate leader vehicle movements. If then, a malicious human driver was to assume the role of the adversarial platoon leader vehicle, they could trigger serious collisions between the follower vehicles of the platoon.

2.3 Fault Diagnosis in Connected Autonomous Vehicles

The high safety risks stemming from CAV vulnerabilities necessitate the development of attack counteraction solutions. While many works propose defence techniques for precluding successful attacks [22], the complexity of the system network in CAVs makes it impossible to proactively consider every attack case. Rather than seeking to prevent all attacks, a new research direction considers concurrently applying intrusion/fault detection techniques that can inform the CAV system that the system has been compromised and abnormal behaviour is occurring [9]. Following detection, the system may then correct behaviour through various fault mitigation techniques.

Borrowing from the literature on process fault detection, there are two prominent classes of quantitative methods: (i) model-based, which requires an understanding of the process dynamics, and (ii) process-based, which leverages process history data to identify abnormal behaviour [8, 28]. From a model-based perspective, many methods rely on observer-based controllers and parameter estimation to identify and mitigate faults. Ploeg *et al.* [29] use a continuous-time equivalent Kalman filter to produce preceding vehicle acceleration estimates in a CACC formulation. In the case that V2V communication is disturbed, these estimates can be used to preserve platoon string stability by replacing the acceleration data originally communicated through the network. For communication faults, Huang *et al.* [30] evaluate a reliable observer-based detector to detect FDI attacks on sensor output and actuator input signals in a CPS, on which a novel attack compensator is designed to mitigate the impact of the attack. Petrillo *et al.* [31] design an adaptive synchronization-based control algorithm for communication time delays and FDI. Biron *et al.* [32] consider a state estimation approach that uses a set of observers with a delay estimator to detect when a DoS attack is occurring, and subsequently applies the state estimate of the preceding vehicle and delay estimate of the DoS attack to switch to a modified CACC control strategy. For actuator faults, Zhang *et al.* [33] leverage a gain-scheduling observer to compare residuals between observed and estimated steering actuator states. Guo *et al.* [34] present an adaptive sliding-mode

controller to implement an improved quadratic spacing policy for CAV platoons that is tolerant to actuator faults.

Alternatively, process-based approaches adopt a data-centric perspective to tackle the fault detection and mitigation problem. These methods possess the advantage of generalizability to unknown vehicle controller dynamics. A recent trend has been to use machine learning (ML) to leverage the high volume of vehicle signal data emitted by CAVs [9]. Neural network-based observers are developed for detecting covert attacks [35] and FDI [36], which are integrated as part of a controller that initiates a fault resolution control scheme to adjust CACC following distance when an attack is detected. Fang *et al.* [37] adopt a hybrid signal-based and model-based approach using a one-class support vector machine (SVM) as a state fault detector and a Kalman filter as a trajectory deviation detector.

2.4 Fault Classification in Connected Autonomous Vehicles

In the existing CAV cyber-security and fault detection literature, it is assumed that the fault class is a known prior. The corresponding fault diagnosis algorithms are hence developed for these specific fault classes, but it is unknown whether they are universally generalizable to additional fault types. On the other hand, fault detection methods are potent in their ability to detect many types of abnormal system behaviour but do not often identify the source of the abnormality [9]. To bridge the gap between fault detection and resolution, a common intermediate task for CPS fault management processes is to introduce a fault classification mechanism that allows a triggered detection system to select an appropriate fault resolution method [1]. An overview of the standard fault management process is given in Figure 1.

Despite the importance of fault classification, few works examine this task in the context of CAVs, especially in the more realistic scenario of CAVs operating alongside human drivers. Similar to fault detection and mitigation, both model-based and signal-based solutions are examined. From the model-based perspective, Biron *et al.* [38] compute combinations of residuals from a sliding-mode observer to differentiate between velocity and range sensor faults in CACC vehicle systems. More popular is the use of signal-based ML techniques such as SVMs and neural networks to extract patterns from historical vehicle operation data. These approaches have been used to classify steering actuator faults [39] and vehicle sensor faults [40], or trajectory deviation types [37] for single autonomous vehicles. In the domain of CAVs, Khalil *et al.* [41] apply an SVM on the velocity signals of a faulty CAV platoon to differentiate between communication time delays, FDI disturbances, and physical engine bearing knock disturbances. van Wyk *et al.* [42] use a convolutional neural network (CNN) augmented with a Kalman filter to identify different types of anomalous sensor values

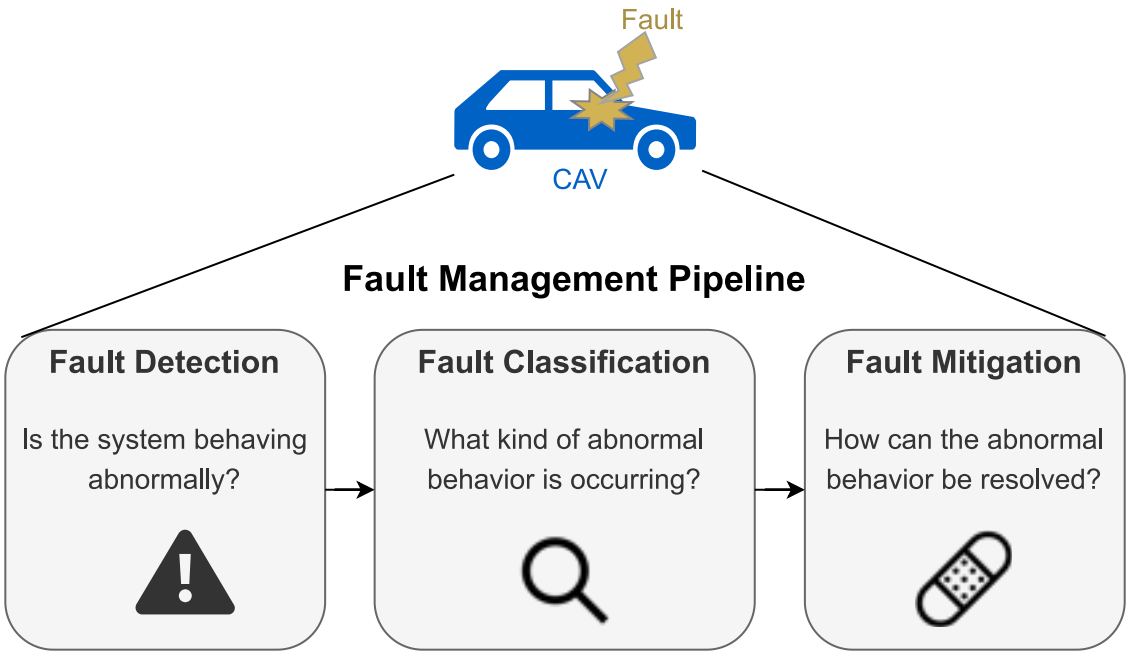


Figure 1: An overview of the standard fault management pipeline [1]. A generalized fault detection module first identifies whether abnormal system behaviour is occurring. A fault classification module then determines the specific type of abnormal behaviour. Finally, the appropriate fault mitigation procedure is selected based on the identified fault to restore normal system operation.

introduced in a simulated CAV environment. Alladi *et al.* [43] use a combination of long short-term memory (LSTM) networks and CNNs to distinguish between normal, faulty, and attack behaviour in a connected vehicular network. Javed *et al.* [44] fuse a CNN with an attention-based LSTM network to classify different sensor anomaly types and outperforms the approach used in van Wyk *et al.* [42], demonstrating the effectiveness of attention in identifying significant patterns that are representative of specific anomalies.

Deep learning solutions are also frequently employed for similar fault classification tasks in adjacent CPS domains given their ability to automatically extract patterns from large amounts of historical data. Xu *et al.* [45] use transfer learning to adapt a CNN model to diagnose industrial equipment faults through the analysis of time-series signal waveforms. Wu *et al.* [46] combine an LSTM with convolutional layers to perform imbalanced fault detection using sensor measurements from industrial plant cyber-physical systems. Cui *et al.* [47] show that a CNN augmented with multi-head attention learns to emphasize important features extracted from the CNN and outperforms feedforward neural networks and basic CNNs on an industrial system fault classification task.

From our survey of the existing literature, the fault classification problem in CAV systems

is understudied. Additionally, should CAVs be introduced to real-world environments, these systems will share the road with non-autonomous vehicles and must therefore account for unpredictable human driver behaviour. Such mixed-vehicle topologies introduce nonlinear dynamics and uncertain disturbances to CAV operation [48], which may result in fault-like sensor signals. To our knowledge, fault classification has not yet been examined in this mixed-vehicle platoon environment. Thus, there is a clear research gap for the development of a fault classification methodology that can not only distinguish between faults in both the cyber and physical layer of CAV systems, but also identify different types of abnormal human driver behaviour.

3 Faulty Platoon Model Design

Although the proposed machine learning technique does not require knowledge of platoon dynamics, a simulation model of the mixed vehicle platoon is required to generate training data. Section 3.1 presents the normal, healthy operation of the platoon model, while Section 3.2 modifies the healthy platoon model with the five fault and abnormality classes considered in this work.

3.1 Healthy Platoon Model

We consider a three-vehicle platoon model driving in a CACC configuration [13] that considers the platoon longitudinal drive only, where a human driver is introduced in between two CAVs as depicted in Figure 2. Our platoon represents the most fundamental model of the mixed-vehicle scenario since any reordering of the vehicles removes the human driver from the CAV platoon. Considering that fault detection techniques commonly provide fault localization [49, 50], our platoon model is also applicable to fault management processes enacted on a larger mixed-vehicle platoon. Preliminary detection and location could be used to first isolate the faulty set of three vehicles as represented by our model, on which our fault classification approach could then be applied.

3.1.1 CAV Model

For our platoon, the longitudinal drive of the CAVs is modelled as a set of brushless electric vehicles with an internal PI controller enacting a cruise control law. Following Khalil *et al.* [51], the CAV model is given by the following transfer function

$$\frac{V_i(s)}{V_i^*(s)} = \frac{\delta_i s + \epsilon_i}{s^3 + \alpha_i s^2 + \beta_i s + \gamma_i}, \tag{1}$$

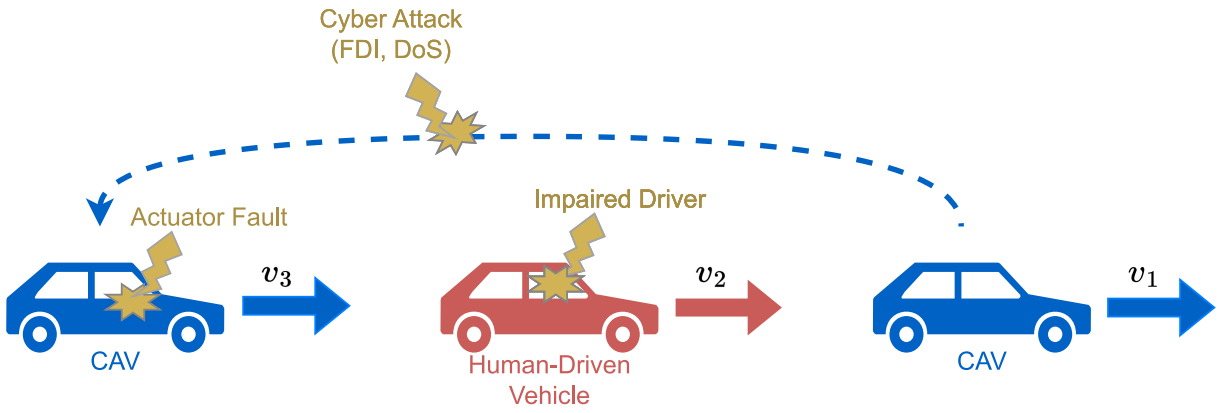


Figure 2: The mixed vehicle platoon fault model under consideration. The first and third vehicles are CAVs, while the second vehicle is human-driven. Actuator faults can occur in the physical components of the follower CAV, while cyber attacks such as FDI and DoS can occur in the communication link between the CAVs. Human drivers can be either distracted or drunk.

where for both the first and third vehicles $i \in \{1, 3\}$, V_i is the velocity of the i-th vehicle in the frequency domain, and V_i^* is the desired velocity of the i-th vehicle in the frequency domain. This model was implemented using a bond graph approach to produce realistic CAV behaviour. α_i, β_i, γ_i, δ_i, and ϵ_i represent the transfer function coefficients configured for the first and third vehicles $i \in \{1, 3\}$, with values given in Table 2.

3.1.2 Human Driver Model

For the second, human-driven vehicle, we adopt the human Intelligent Driver Model (IDM) from Treiber *et al.* [52] as an idealized example of healthy driver behaviour. This model of human driver behaviour is chosen for its frequent usage in AV literature and for the ease of interpretability of its parameters. The healthy IDM model is given by

$$a_2(t, v_2) = a_{\max} \left[1 - \left(\frac{v_2(t)}{v_2^*(t)} \right)^{\lambda} - \left(\frac{s_2^*(t, v_2)}{s_2(t)} \right)^2 \right], \tag{2}$$

where a_2 is the acceleration of the human-driven vehicle (second vehicle in the platoon), $a_{\max}$ is the maximum acceleration, v_2 and v_2^* are the measured and desired velocities of the human-driven vehicle, respectively. λ is a constant that controls the speed of the human response. s_2 is the actual spacing distance between the first and second vehicles, and s_2^* is the desired spacing distance and is given by

$$s_2^*(t, v_2) = s_0 + \max \left(0, v_2(t)T + \frac{v_2(t)\Delta v_2(t)}{2\sqrt{b^* a_{\max}}} \right), \tag{3}$$

where s_0 is the minimum spacing distance, T is a time constant, $\Delta v_2(t) = v_2(t) - v_1(t)$ is the approaching rate to the next vehicle, $a_{\max}$ is the maximum acceleration from (2), and b^* is the comfortable braking deceleration. The parameter values for the human-driven vehicle are given in Table 2.

Figure 3 shows the response of the platoon in Figure 2 when no faults occur. The platoon was constructed such that the first and third vehicles follow the model in (1), and the second vehicle follows the model (2)-(3). The second vehicle follows the velocity of the first vehicle, while the third vehicle follows the velocity average of the first two vehicles as a simple approach to establish platoon velocity consensus when faults are possible [53]. Note that the healthy human driver has a slower response compared with CAVs.

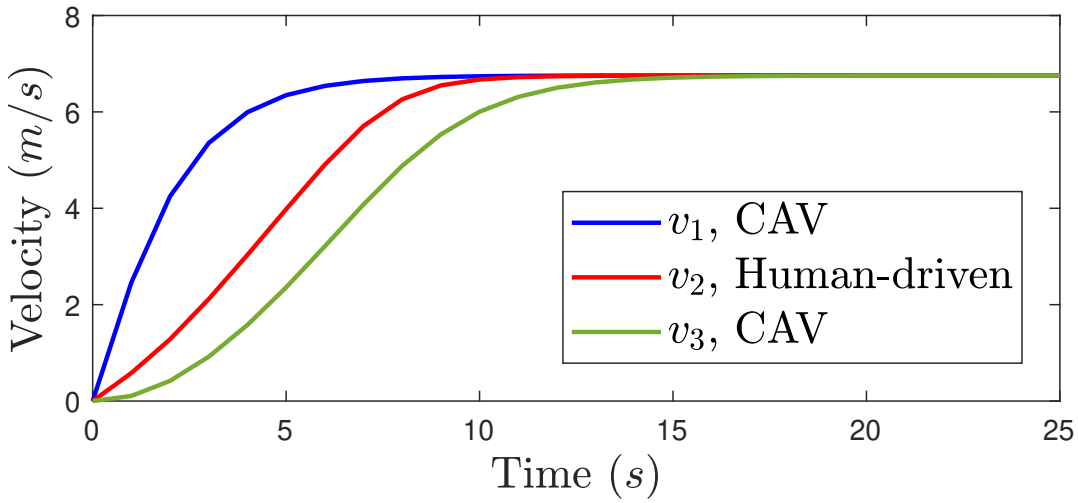


Figure 3: Healthy response of the three vehicles platoon in Figure 2, where the first and third vehicles are CAVs and the second vehicle is human-driven.

Table 2: Summary of parameter value selection in the healthy platoon model.

Model Parameter	Value
α_i	72.01
β_i	117.9
γ_i	46.72
δ_i	28.03
ϵ_i	46.72
λ	8
s_0	$2m$
T	$1.5s$
$a_{\max}$	$1m/s^2$
b^*	$3m/s^2$

3.2 Fault Models

To simulate the different fault classes, we alter the healthy platoon model in Section 3.1 with one of the following five fault classes: (i) actuator faults, (ii) false data injection (FDI), (iii) denial-of-service (DoS) attacks, (iv) distracted human drivers, and (v) drunk human drivers. These faults are selected for their full coverage of the physical fault, cyber-communication attack, and human-driver abnormality classes. Fault classes (i)-(iii) impact the behaviour of the CAVs, while abnormality types (iv) and (v) simulate impaired behaviour in the human-driven vehicle. Only one fault is assumed to occur at a time. The following modifications are made to the healthy platoon to simulate each of the fault classes.

(i) **Actuator Fault:** An actuator fault refers to a loss in the effectiveness of the third vehicle's motor. This fault occurs when the motor is subject to severe operating conditions such as high magnetic force and different weather conditions [54]. This loss in effectiveness results in a lower response amplitude, which is simulated by lowering the parameter ϵ_3 to a value of 41. This drops the actuator effectiveness by almost 80%.

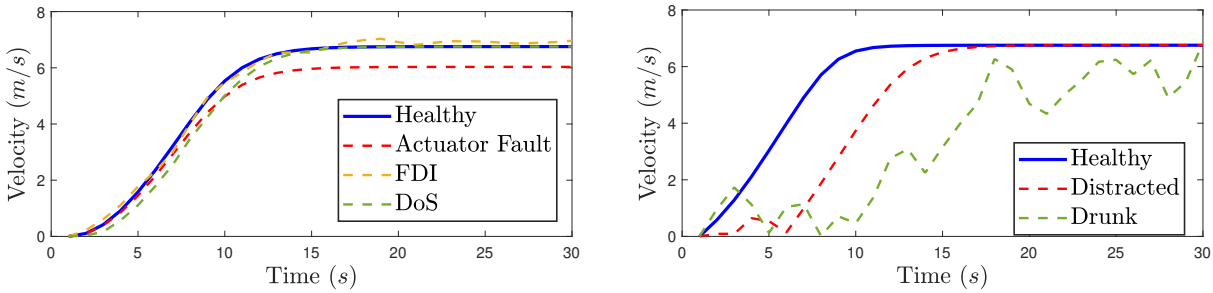
(ii) **False Data Injection:** FDI attacks refer to false vehicle velocity information being transmitted through communication channels [31]. We model this fault as noise injected in the velocity of the first vehicle that is received by the third vehicle. We simulate this by altering the velocity used in controlling the third vehicle to $\tilde{v}_1(t) = v_1(t) + \eta_{\text{FDI}}(t)$, where $\tilde{v}_1$ is the velocity of the first vehicle received by the third vehicle, and η_{FDI} is injected bounded white noise.

(iii) **Denial-of-Service Attack:** DoS attacks refer to malicious interference on the inter-vehicular communication channel, which results in delayed transmission of vehicle velocity information [31]. We model this fault as a time-variant communication delay in the velocity of the first vehicle that is received by the third vehicle under a no packet loss assumption. We alter the velocity of the first vehicle as $\tilde{v}_1(t) = v_1(t - \tau_{\text{delay}}(t))$, where τ_{delay} is a normally distributed variable time delay that captures the communication latency.

(iv) **Distracted Human Driver:** Distracted drivers tend to exhibit delayed responses to the external driving environment [55]. We model this abnormality as a delay in the response $\lambda = 5$ as well as in tracking the velocity of the front vehicle $v_2^*(t) = v_2^*(t - \tau_{\text{distracted}}(t))$, where $\tau_{\text{distracted}}$ is a normally distributed variable time delay that captures the delay in tracking the front vehicle.

(v) **Drunk Human Driver:** Drunk drivers are also considered alongside distracted drivers to represent a more extreme version of unstable human behaviour. Following the report published by the National Highway Safety Association [56], a moderately drunk driver exhibits (i) a decline in visual functions, (ii) reduced coordination, (iii) reduced ability to track moving objects, and (iv) decline in the ability to perform two tasks at the same time and reduced response to emergencies. We model these effects as (i) $\tilde{s}_2^*(t) = s_2^*(t) + \eta_{s^*}(t)$, (ii) $\tilde{s}_2(t) = s_2(t) + \eta_s(t)$, (iii) $\tilde{v}_1(t) = v_1(t - \tau_{\text{drunk}})$, and (iv) $\lambda = 3$, where $\tilde{s}_2^*, \tilde{s}_2, \tilde{v}_1$ refer to the corrupted desired spacing distance, actual spacing distance, and first vehicle's velocity used as the human driver's following velocity, respectively. η_{s^*}, η_s are bounded white noise, and $\tau_{\text{drunk}} = 2s$ captures a constant time delay in tracking the front vehicle.

Figure 4 shows the responses of the mixed vehicle platoon described in Section 3.1 after altering them with the faults and abnormalities introduced in Section 3.2. Figure 4a shows the third vehicle's response after altering it with the three CAV faults (i)-(iii), and Figure 4b shows the human-driven vehicle's velocity after altering it with the two human driver abnormalities (iv)-(v).



(a) **(b)**

Figure 4: Visualization of faulty vehicle responses under each fault class. (a) Third vehicle response v_3 in Figure 3 after emulating the faults (i)-(iii) introduced in Section 3.2, and (b) Second vehicle velocity v_2 in Figure 3 after emulating the impaired driver effects (iv)-(v) introduced in Section 3.2. Non-faulting vehicles are omitted for visualization clarity.

4 Machine Learning Models

The mixed vehicle platoon fault model simulation allows us to extract velocity profiles for each fault and abnormality class. To leverage this data, we take a signal-based fault classification approach by applying machine learning (ML), in particular deep learning (DL), to learn intrinsic patterns from high volumes of labelled input data. Section 4.1 presents multiple benchmark models from similar anomaly classification tasks in the autonomous vehicle domain. Section 4.2 introduces the motivation, architecture, and theoretical underpinning of our novel multi-head attention network approach.

4.1 Benchmark Models

While fault classification in the context of mixed vehicle platoons has not been directly studied in the literature, several works present ML solutions for similar classification tasks in AV, human driver, or CAV environments. This section reviews the design of these existing approaches and explains their implementation details.

4.1.1 Support Vector Machine

The Support Vector Machines (SVM) is a traditional supervised machine learning method designed for binary classification tasks [57]. The SVM applies a kernel function to transform input vectors to a higher dimensional space, after which it computes an optimal hyperplane that forms a linear decision boundary between the two classes in the higher dimensional space. The optimal hyperplane is one that maximizes its margin (i.e. distance) between

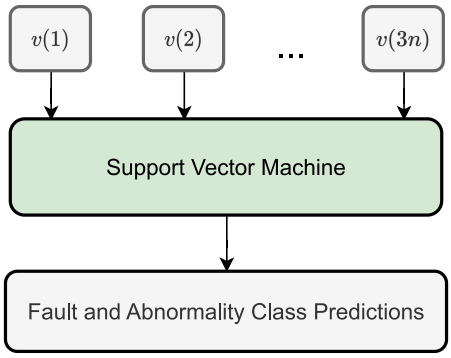


Figure 5: The SVM classifier benchmark model.

vectors of either class. The vectors that lie on this margin are known as support vectors, from which the SVM method derives its name. In the case that the training data used to form the decision boundary is not fully linearly separable, a soft-margin hyperplane can instead be derived to minimize the number of erroneously classified vectors. In both forms, the parameters of the optimal hyperplane are computed using quadratic programming methods. Burges [58] provides an excellent in-depth discussion of SVM concepts.

For the multi-class classification problem, SVMs can also be applied by learning a unique decision boundary for each class. Shi *et al.* [39] apply such an approach for fault diagnosis of steering actuator faults in autonomous vehicles. We adopt a similar methodology as a benchmark model for our mixed vehicle platoon fault classification task, given in Figure 5. Each velocity vector $\mathbf{v}_i \in \mathbb{R}^n$ for all vehicles $i \in \{1, 2, 3\}$ is concatenated to form an input vector of dimension $\mathbb{R}^{3n}$. These input vectors are then used to compute the decision boundaries for the SVM classifier. The SVM classifier uses a radial basis function kernel and includes an L2 regularization term to reduce overfitting.

4.1.2 Multi-Layer Perceptron with Time-Domain Features

The multi-layer perceptron (MLP) is a fundamental class of machine learning model consisting of multiple layers of fully connected neurons, where each layer of neurons apply a linear transformation $W\mathbf{x}+\mathbf{b}$ to an input vector $\mathbf{x}$, with W and $\mathbf{b}$ corresponding to a weight parameter matrix and bias parameter vector respectively. Typically, these network parameters are optimized through a gradient descent algorithm. These linear transformations are performed sequentially at each layer of the vector, with a non-linear activation function (e.g. sigmoid, ReLU) applied between each linear transformation. Together, this alternating sequence of activations allows MLPs to learn complex non-linear patterns from input data.

Safavi *et al.* [40] apply a 3-layer MLP to classify between different forms of sensor faults in an autonomous vehicle. To reduce the input sensor signals to a manageable size, the authors

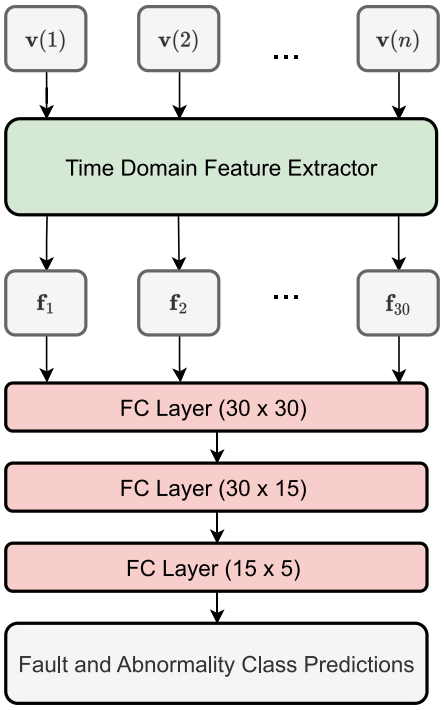


Figure 6: The MLP benchmark model.

extract a set of ten time-domain features from the input signal. The computation of these features is given in Table 3. We borrow this methodology and extract these ten features from the vehicle velocity profile emitted by each vehicle in our platoon, thus totalling 30 features. These 30 features are then used to represent each data sample in our MLP benchmark model shown in Figure 6. Between each fully-connected (FC) layer, the ReLU activation is applied to introduce non-linearities in the network.

Table 3: Time-domain features extracted from a vehicle velocity profile $\mathbf{x} \in \mathbb{R}^N$ for the MLP benchmark model. The i-th element of $\mathbf{x}$ is denoted as x_i. The sample mean is denoted as μ and the sample standard deviation is denoted as σ.

$f_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$	$f_{SRA} = \left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\lvert x_i\rvert}\right)^2$
$f_{KV} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^4$	$f_{SV} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^3$
$f_{PPV} = \lvert\mathbf{x}\rvert_{\max} - \lvert\mathbf{x}\rvert_{\min}$	$f_{CF} = \frac{\lvert\mathbf{x}\rvert_{\max}}{f_{RMS}}$
$f_{IF} = \frac{\lvert\mathbf{x}\rvert_{\max}}{\frac{1}{N}\sum_{i=1}^{N}\lvert x_i\rvert}$	$f_{MF} = \frac{\lvert\mathbf{x}\rvert_{\max}}{f_{SRA}}$
$f_{SF} = \frac{f_{RMS}}{\frac{1}{N}\sum_{i=1}^{N}\lvert x_i\rvert}$	$f_{KF} = \frac{f_{KV}}{\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{x_i^2}\right)^2}$

4.1.3 Convolutional Neural Network

The convolutional neural network (CNN) was popularized by Krizhevsky *et al.* [59] for its considerable improvement over existing methods at the time on image classification tasks, in particular, on the ImageNet grand challenge. Eschewing the interpretability of hand-crafted features, CNNs instead leverage huge volumes of training data and computational resources to allow the model itself to learn to extract salient features from the data. As some of the first "deep" networks, these networks had upwards of hundreds of thousands to millions of learnable parameters, distributed over many layers of computation. In contrast to the fully connected layer of MLPs, CNNs employ a small, fixed-size window known as a kernel and convolve this kernel over a data sample. By sharing a kernel throughout the data sample, CNNs can learn a translation-invariant feature representation for different elements of the input data. A pooling step is also frequently used to shrink the size of the input data as it traverses the network, which allows deeper-level layers to learn higher-order, aggregate features of the input data. The translation-invariant property created by these convolution and pooling layers makes CNNs a favourable choice for tasks such as image and signal processing.

van Wyk *et al.* [42] use a deep CNN to detect and identify sensor anomalies in CAV. We implement a similar network as a benchmark model by replacing the original input sensor signal sequences with vehicle velocity profiles. The resulting model architecture is given in Figure 7. Between each of the convolution and pooling layers, we apply the ReLU activation function to introduce non-linearities to the network. Dropout with probability 0.1 is also used between the two fully connected layers to reduce overfitting. Additionally, note that the fully connected layer requires a fixed-length vector, however, our input data can consist of a variable number of timesteps n. We address this discrepancy by introducing an average pooling layer to aggregate the final feature representation of the third convolution layer.

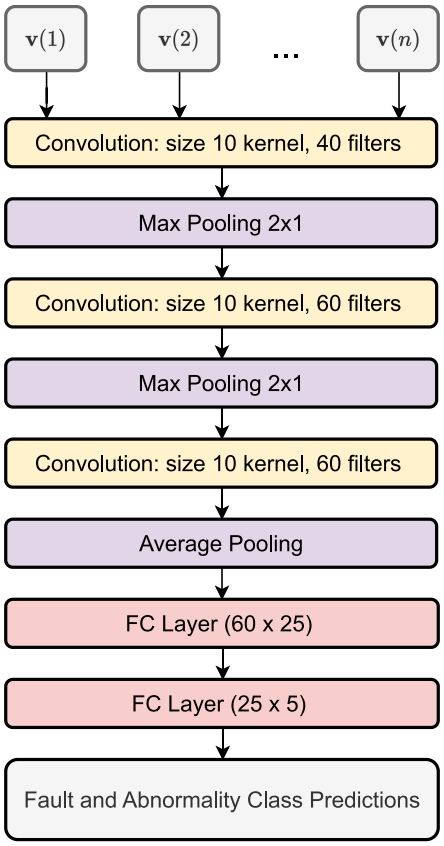


Figure 7: The CNN benchmark model.

4.1.4 LSTM Recurrent Neural Network

The recurrent neural network (RNN) is a form of neural network designed for sequence processing tasks. The RNN consists of multiple layers of cyclically connected nodes, where the same node is applied to compute an output for each time step of the input data sequence. By maintaining an internal state in all network nodes at each time step, the RNN is able to persist sequential information over the entire input data. The choice of the node for an RNN has a large impact on the effectiveness and stability of the network. One standard selection is the long short-term memory (LSTM) cell, which demonstrates superior performance in memorizing information over lengthy time sequences [60].

Girma *et al.* [61] leverage a two-layer LSTM network to identify human driver behaviour types given sequences of vehicle telemetrics data. Since our platoon vehicle velocity profiles consist of a similar data format, we adopt the same LSTM-based RNN architecture for a benchmark model, as presented in Figure 8. The input velocity data is first encoded at each time step using the same embedding layer structure as the multi-head attention network. Each input time step is then fed sequentially into two consecutive LSTM layers. The final

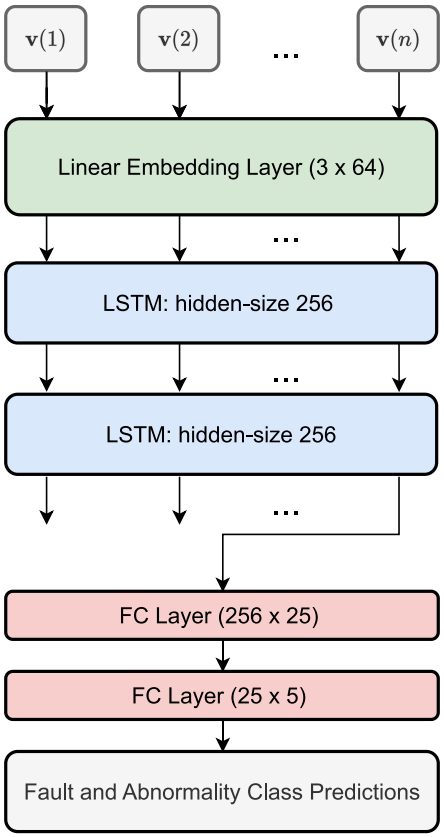


Figure 8: The LSTM benchmark model.

LSTM output is extracted and passed to a fully connected network with a ReLU activation function to produce fault class probabilities.

4.1.5 Multi-Stage Attention LSTM-CNN

Many custom ML architectures emerge from combining elements from these prior models to leverage their advantageous properties on particular tasks. Javed *et al.* [44] take such an approach by designing a novel multi-stage attention LSTM-CNN (MSALSTM-CNN) architecture for sensor anomaly identification in CAVs. Their proposed approach first uses a CNN to extract a feature map representation of the input sensor data. An LSTM layer is then applied to learn sequential patterns in the feature map representation. Finally, a content-based attention mechanism [62] derives a context vector from the sequential LSTM outputs, which learns to assign scores to particular areas of the sequence based on their significance. The MSALSTM-CNN approach is a unique example of the use of attention for classification tasks in the context of CAV fault identification, thus we adopt it as a benchmark model for our task, with the architecture of the network given in Figure 9.

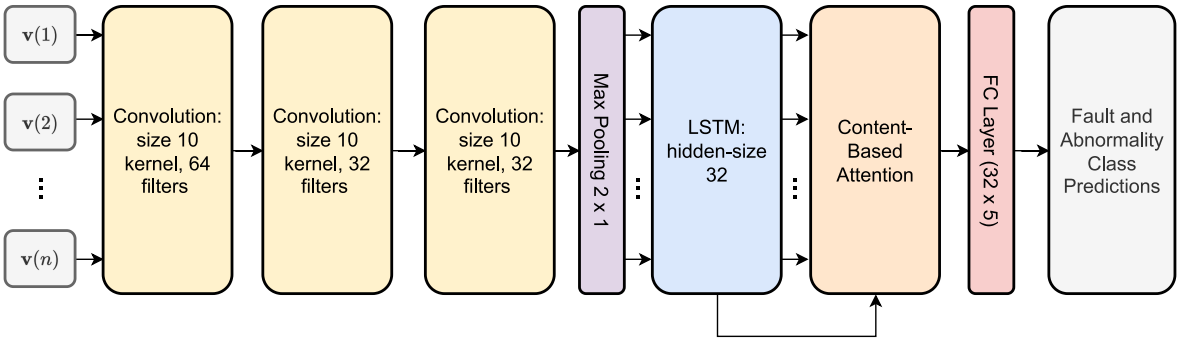


Figure 9: The MSALSTM-CNN benchmark model.

4.2 Multi-Head Attention Network

We introduce a new Multi-Head Attention (MHA) network for fault classification in mixed vehicle platoon topologies. The network structure is inspired by the transformer encoder proposed in [10], which was selected given its established performance and efficiency for long sequence processing. The MHA network consists of a stack of scaled dot-product attention computations that run in parallel. Each attention computation results in an independent output, where all outputs are then transformed linearly to the expected dimension. In this section, we provide a theoretical overview of the computation that is carried out by the MHA network.

For the discrete-time sample $k \in \{1, \ldots, n\}$, where n is the total number of samples, let the platoon velocities be collected in the row vector $\mathbf{v}(k) = \begin{bmatrix} v_1(k) & v_2(k) & v_3(k) \end{bmatrix} \in \mathbb{R}^3$. The goal is to identify the fault class using platoon velocities only. The MHA network is implemented as shown in Figure 10. We will first define the core processing layers and construct the forward computation using these layers.

4.2.1 Multi-Head Attention

Multi-head attention layers are the primary computation in the MHA network, as defined in the function given by

$$\mathcal{M}_m(X) = \begin{bmatrix} \mathcal{A}(XW_1^{Q^{[m]}}, XW_1^{K^{[m]}}, XW_1^{V^{[m]}}) \\ \vdots \\ \mathcal{A}(XW_H^{Q^{[m]}}, XW_H^{K^{[m]}}, XW_H^{V^{[m]}}) \end{bmatrix}^T W^{A^{[m]}}, \tag{4}$$

where $W_h^{Q^{[m]}} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_h^{K^{[m]}} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W_h^{V^{[m]}} \in \mathbb{R}^{d_{\text{model}} \times d_v}$ correspond to query, key, and value projection weight matrices for attention module stack $m \in \{1, \ldots, M\}$ and attention head h. M denotes the total number of attention module stacks in the network.

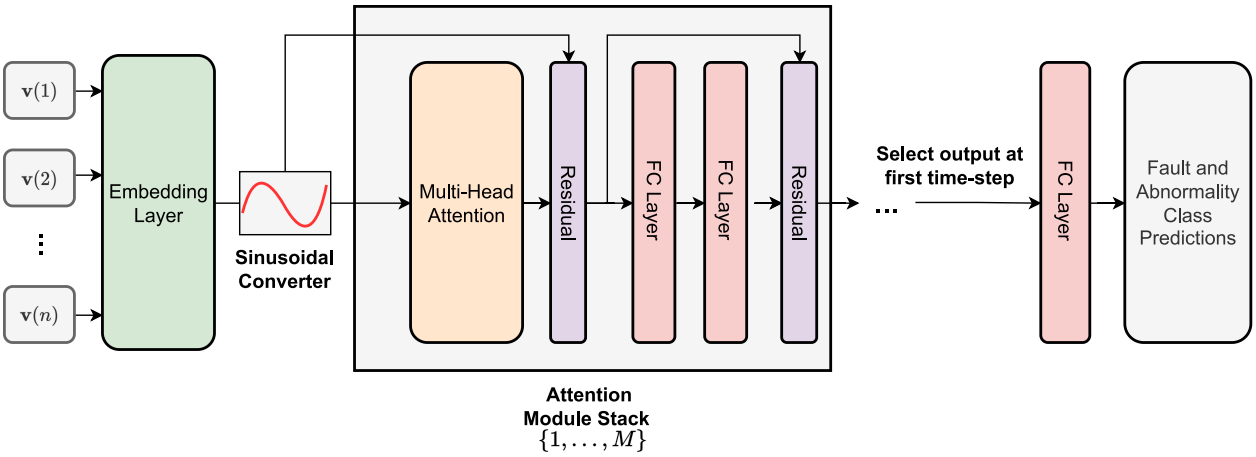


Figure 10: Proposed multi-head attention network architecture for mixed vehicle platoon fault classification.

$W^{A^{[m]}} \in \mathbb{R}^{Hd_v \times d_{\text{model}}}$ denotes the multi-head attention projection weight matrix for stack m. $X \in \mathbb{R}^{n \times d_{\text{model}}}$ represents the input matrix to the multi-head attention function. The scalars d_k and d_v denote the key and value projection sizes, respectively. d_{model} denotes the hidden size of the MHA network. H denotes the total number of heads in the multi-head attention layer. We consider M, H, d_{model}, d_k and d_v to be architectural hyperparameters that can be tuned to control the number of parameters used in each multi-head attention layer.

The multi-head attention function in (4) computes a stack of H independent calculations of the attention function $\mathcal{A}(Q, K, V)$, which represents the scaled-dot product attention function given by

$$\mathcal{A}(Q, K, V) = \mathcal{S}\left(\frac{1}{\sqrt{d_k}} QK^T\right) V, \tag{5}$$

where $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$, and $V \in \mathbb{R}^{n \times d_v}$ represent query, key, and value matrices. The scaling factor of $\frac{1}{\sqrt{d_k}}$ is necessary to counteract the vanishing gradient effect for large values of d_k [10]. Function $\mathcal{S}(\cdot)$ represents a matrix softmax function defined by

$$\mathcal{S}(Z) = \begin{bmatrix} \sigma(\mathbf{z}_1) \\ \vdots \\ \sigma(\mathbf{z}_n) \end{bmatrix}, \tag{6}$$

where

$$\sigma(\mathbf{z}_i) = \frac{1}{\sum_{j=1}^{n} e^{z_{i_j}}} \begin{bmatrix} e^{z_{i_1}} & \dots & e^{z_{i_n}} \end{bmatrix} \tag{7}$$

Here, $\mathbf{z}_i \in \mathbb{R}^n$ represents the i-th row vector of an input matrix $Z \in \mathbb{R}^{n \times n}$, with z_{i_j} denoting the j-th element of row vector $\mathbf{z}_i$. $\sigma(\mathbf{z})$ represents the vector softmax function computed on row vector $\mathbf{z}$ as per (7). The matrix softmax in (6) has the effect of computing

a probability distribution over the sequence dimension, which acts as a set of attention weights that prioritize specific timesteps of the input sequence.

4.2.2 Fully Connected Network

A simple fully connected network (FCN) introduces a non-linear transformation to the multi-head attention output. The network computation is captured by the function given by

$$\mathcal{F}_m(X) = \left([XW_1^{\text{ff}[m]}] \odot \mathbb{1}(XW_1^{\text{ff}[m]} > \mathbf{0})\right) W_2^{\text{ff}[m]}, \tag{8}$$

where $X \in \mathbb{R}^{n \times d_{\text{model}}}$ denotes an input matrix. $W_1^{\text{ff}[m]} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $W_2^{\text{ff}[m]} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ are the two weight parameter matrices in the FCN for attention module stack m. $\mathbb{1}(\cdot)$ denotes an element-wise indicator function which evaluates to 1 when the condition in the expression is true and 0 when the condition is false. $\odot$ denotes the Hadamard element-wise matrix product. Note that the expression $[XW_1^{\text{ff}[m]}] \odot \mathbb{1}(XW_1^{\text{ff}[m]} > \mathbf{0})$ corresponds to applying the rectified linear unit (ReLU) non-linear activation function to $XW_1^{\text{ff}[m]}$. We consider d_{ff} as an architectural hyperparameter that can be tuned to control the number of parameters in the FCN.

4.2.3 Residual

Residual connections are introduced to improve model stability and preserve information from the attention layer input. The function is defined by

$$\mathcal{R}(X, Z) = \mathcal{N}(X + Z), \tag{9}$$

where $X \in \mathbb{R}^{n \times d_{\text{model}}}$ denotes an input matrix and $Z \in \mathbb{R}^{n \times d_{\text{model}}}$ corresponding to some transformed version of X (i.e., after passing X as input to the multi-head attention layer or to the FCN). $\mathcal{N}(\cdot)$ represents the layer normalization function given by

$$\mathcal{N}(Y) = \frac{1}{\sqrt{\sigma^2 + \epsilon}}(Y - \bar{Y}) \odot \Gamma + B, \tag{10}$$

where $Y \in \mathbb{R}^{n \times d_{\text{model}}}$ is an input matrix with $y_{i,j}$ corresponding to the element at row i, column j. We define $\bar{Y} \in \mathbb{R}^{n \times d_{\text{model}}}$ as a matrix of element-wise means of Y such that $\bar{y}_{i,j} = \mu = \frac{1}{nd_{\text{model}}} \sum_{i=1}^{n} \sum_{j=1}^{d_{\text{model}}} y_{i,j}$ for all rows i, columns j in $\bar{Y}$. Note that $Y - \bar{Y}$ is equivalent to the element-wise subtraction of μ from each element in Y. We also define $\sigma^2 = \frac{1}{nd_{\text{model}}} \sum_{i=1}^{n} \sum_{j=1}^{d_{\text{model}}} (y_{i,j} - \mu)^2$ as the variance of Y element-wise. ϵ is a small constant introduced for numerical stability. $\Gamma \in \mathbb{R}^{n \times d_{\text{model}}}$ and $B \in \mathbb{R}^{n \times d_{\text{model}}}$ are learnable transformation weight parameters that allow rescaling and recentering of the normalized matrix [63].

4.2.4 Sinusoidal Converter

The MHA network uses a precomputed sinusoidal signal to inject sequential information, as per [10]. This sinusoidal signal allows the model to learn the relative positions of the sequence without relying on slower sequential processing from RNNs. The sinusoidal signal $P \in \mathbb{R}^{n \times d_{\text{model}}}$ is defined as

$$P = \begin{bmatrix} \mathbf{p}(1) \\ \vdots \\ \mathbf{p}(n) \end{bmatrix}, \tag{11}$$

where $\mathbf{p}(k) \in \mathbb{R}^{d_{\text{model}}}$ is defined with element $j \in \{1 \dots d_{\text{model}}\}$ of the row vector defined as

$$\mathbf{p}(k)_j = \begin{cases} \sin\left(\frac{k}{10000}^{2j/d_{\text{model}}}\right) & \text{when } j \text{ is even,} \\ \cos\left(\frac{k}{10000}^{2j/d_{\text{model}}}\right) & \text{when } j \text{ is odd.} \end{cases} \tag{12}$$

4.2.5 Forward Pass

Consider an input sequence of platoon velocities $\{\mathbf{v}(k)\}$, where $k \in \{1, \dots, n\}$ and $\mathbf{v}(k) \in \mathbb{R}^3$. The MHA network first computes a velocity embedding $E_0 \in \mathbb{R}^{n \times d_{\text{model}}}$ for the platoon velocities. We propose two types of embeddings. The first is a linear projection given by the expression

$$E_0 = \begin{bmatrix} \mathbf{e}_0(1) \\ \vdots \\ \mathbf{e}_0(n) \end{bmatrix} = \begin{bmatrix} \mathbf{v}(1) \\ \vdots \\ \mathbf{v}(n) \end{bmatrix} W_E + P, \tag{13}$$

where $\mathbf{e}_0(k) \in \mathbb{R}^{d_{\text{model}}}$ is the velocity embedding corresponding to platoon velocities $\mathbf{v}(k)$. $W_E \in \mathbb{R}^{3 \times d_{\text{model}}}$ is a weight matrix for the linear projection, and $P \in \mathbb{R}^{n \times d_{\text{model}}}$ is the sinusoidal signal defined in (11). We name this variant of the MHA network MHA-FCN.

Alternatively, a convolution-based embedding layer can be used to prime the MHA network with an input embedding sequence possessing a greater local context of surrounding

time steps in the vehicle velocity profiles. This embedding is given by the expression

$$E_0 = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{v}(1) \\ \vdots \\ \mathbf{v}(n) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} * K_E + P, \tag{14}$$

where $K_E \in \mathbb{R}^{d_{\text{kernel}} \times 3 \times d_{\text{model}}}$ represents a stack of d_{model} $\mathbb{R}^{d_{\text{kernel}} \times 3}$ kernels that are convolved over the input sequence of platoon velocities $\{\mathbf{v}(k)\}$, with $*$ representing the valid cross-correlation operation. Note that the input sequence $\{\mathbf{v}(k)\}$ is padded with zeros to ensure that the final embedding E_0 remains in $\mathbb{R}^{n \times d_{\text{model}}}$. Essentially, each $\mathbb{R}^{d_{\text{kernel}} \times 3}$ kernel computes a feature map of size $\mathbb{R}^{n \times 1}$. This operation is repeated, with the resulting feature maps concatenated along the column axis to form the embedding $E_0 \in \mathbb{R}^{n \times d_{\text{model}}}$. We name this variant of the MHA network MHA-CNN and include d_{kernel} as an additional architectural hyperparameter that controls the size of the kernel.

Next, for $m \in \{1, \ldots, M\}$, where M is the total number of attention module stacks we define for the network, let $E_m \in \mathbb{R}^{n \times d_{\text{model}}}$ denote the output of module stack m. The forward pass computation is carried out as follows

$$H_m = \mathcal{R}(E_{m-1}, \mathcal{M}_m(E_{m-1})) \tag{15}$$

$$E_m = \mathcal{R}(H_m, \mathcal{F}_m(H_m)), \tag{16}$$

where $H_m \in \mathbb{R}^{n \times d_{\text{model}}}$ denotes the hidden state of attention stack m between the multi-head attention layer and the FCN. $\mathcal{M}_m(\cdot)$ is the multi-head attention function for attention module stack m defined in (4), $\mathcal{F}_m(\cdot)$ is the FCN for attention module stack m defined in (8), and function $\mathcal{R}(\cdot, \cdot)$ is the residual add-and-normalize function defined in (9).

To compute the faults and abnormalities classification, we extract the first output time step of the final attention module stack E_M as performed in [64]. We define the resulting representation as

$$\bar{\mathbf{e}} = \mathbf{e}_M(1), \tag{17}$$

where $\mathbf{e}_M(k) \in \mathbb{R}^{d_{\text{model}}}$ denotes row k of E_M. The faults and abnormalities prediction probabilities, $\hat{\mathbf{y}} \in \mathbb{R}^5$, are then computed using a fully connected layer and softmax output function

given by

$$\hat{\mathbf{y}} = \sigma(\bar{\mathbf{e}} W_C), \tag{18}$$

where $W_C \in \mathbb{R}^{d_{\text{model}} \times 5}$ is a weight matrix that maps the pooled embedding $\bar{\mathbf{e}}$ to the five faults and abnormalities classes. $\sigma(\cdot)$ is the softmax function (7) used to compute classification probabilities. $\hat{\mathbf{y}}_i$ represents the probability that the input sequence exhibits the fault or abnormality class indexed by i. The MHA network proposes the fault class $\hat{y} \in 1, \ldots, 5$ with the highest predicted probability as the fault associated with the input velocity signal, as given by

$$\hat{y} = \underset{i \in [1,5]}{\text{argmax}}\ \hat{\mathbf{y}}_i, \tag{19}$$

5 Model Training

In this section, we introduce the data generation process, preprocessing techniques, and parameter optimization algorithm used to train the benchmark models and proposed MHA network.

5.1 Data Generation

To train the machine learning network, the healthy platoon model was constructed in MATLAB SimuLink with each fault class implemented individually in a separate run. The desired velocity of the platoon was set to change randomly every 30 seconds to elicit continuous responses from the platoon vehicles. Each run is 500 seconds long with a sampling period of 1 second. From each run, we record the three vehicles' velocities, resulting in 500 velocity measurements for each of the three vehicles in the platoon. Overall, 5000 runs were recorded, with 1000 runs per fault class, to form our simulation-based training dataset.

5.2 Data Preprocessing

Before training the machine learning models, the training data was normalized to improve the stability of the training process. We applied min-max normalization to re-scale each data sample to the range $[-1, 1]$. For all $k \in \{0, \ldots, n\}$, the scaled velocity of vehicle $i \in \{1, 2, 3\}$ is given by

$$\bar{v}_i(k) = 2 \left(\frac{v_i(k) - |\mathbf{v}_i|_{\min}}{|\mathbf{v}_i|_{\max} - |\mathbf{v}_i|_{\min}} \right) - 1, \tag{20}$$

where $\bar{v}_i$ is the scaled velocity of vehicle i, and $|\mathbf{v}_i|_{\max}, |\mathbf{v}_i|_{\min}$ are the upper and lower bounds over the complete training dataset for the velocity signal of the i-th vehicle $\mathbf{v}_i$, respectively. The min-max normalization was used rather than zero-mean feature standardization because

Table 4: Architectural hyperparameter values for the best multi-head attention network.

Architectural Hyperparameter	Variable	Value
Velocity embedding size	d_{model}	32
Attention key & value projection size	$d_k = d_v$	32
Number of attention heads	H	5
FCN hidden size	d_{ff}	64
Number of encoder stacks	M	3
Kernel size (used in MHA-CNN)	d_{kernel}	10

mapping to a single mean and variance would misrepresent the platoon's dynamics since the desired platoon velocity changes randomly throughout a run.

5.3 Network Optimization

We apply 5-fold cross-validation, which splits our dataset into five non-overlapping validation datasets each consisting of 20% of the simulation data. Cross-validation provides a better estimate of an ML model's generalization performance because it removes any bias towards a particular split of the training data. For each validation fold, the remaining 80% of the simulation data is used as a training dataset. The network parameters are learned on the training data using a cross-entropy loss function and Adam optimization [65]. We use a batch size of 100 and maintain default hyperparameter values for the optimizer. Each network is trained for 150 epochs with early stopping if the validation loss does not decrease for 5 consecutive epochs, which helps to avoid overfitting the training data. For the MHA network, a random search was conducted over 40 combinations of model architecture sizes to optimize the network structure. Values were selected from the following sets: $\{16, 32, 64\}$ for d_{model}, the velocity embedding size; $\{16, 32, 64\}$ for d_k, the key projection size; 1-5 for H, the number of attention heads; $\{16, 32, 64\}$ for d_{ff}, the FCN hidden size; and 1-5 for M, the number of attention module stacks. We also reduce the search complexity by setting the value projection size equal to the key projection size, $d_v = d_k$. The combination with the lowest validation loss was taken as the final network. Table 4 shows the architectural hyperparameter values of the best MHA network that were found through random search. Cross-entropy loss and accuracy curves on the training and validation datasets for each model are given in Figure 12 in Appendix A.

6 Model Validation

Although the network achieves promising results on the simulation-derived dataset, the ideal nature of simulations may limit the network's generalizability to real-world environments. In this section, we introduce an experimental method to test the robustness of our ML networks in real-world conditions and describe how performance will be compared.

6.1 Assessing Robustness to Noisy Sensor Conditions

The proposed fault classification approaches rely on measured velocity data collected from onboard vehicle sensors. In practice, these sensors are sensitive to defects and external environmental conditions that can introduce noise, potentially corrupting the measured velocity data [61]. To gain a better understanding of the generalizability of the proposed fault classification approaches, we must consider the effect of increasing levels of sensor measurement noise and evaluate its impact on model performance.

We begin by building a test dataset using the platoon simulation model. This test dataset contains a new set of 500 runs, with 100 runs for each fault class. Critically, the data in this test dataset is not seen by the model during training or validation, which allows predictions to be unbiased. We then simulate sensor measurement noise as a white Gaussian noise signal $\mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2. This noise is randomly distributed at each time step and is independent of the test data. The noise is injected into the test dataset post-normalization for each vehicle by summing the Gaussian noise vector with the velocity measurement. Different severities of sensor measurement corruption are induced by controlling the variance of the injected noise. We consider four distinct scenarios: (i) $\sigma^2 = 0$, representing the no noise scenario; (ii) $\sigma^2 = 0.05$, representing a low noise scenario; (iii) $\sigma^2 = 0.1$, representing a high noise scenario; and (iv) $\sigma^2 = 0.2$, representing a data corruption scenario.

6.2 Performance Evaluation Methods

Two classification performance measures, accuracy and F1-score, are used to evaluate model performance on the test dataset. Accuracy is computed at an aggregate level for all fault classes through the expression

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

F1-score is computed for each class through the expression

$$\text{F1} - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where precision is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

and recall is defined as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

with TP representing the number of true positive predictions for a given class, FP representing the number of false positive predictions for a given class, and FN representing the number of false negative predictions for a given class. Precision is used to assess the correctness of a model in predicting the positive class, while recall is used to assess how comprehensive a model is in predicting positive samples. F1-score is computed via the harmonic mean of precision and recall, which provides a balanced measure of the precision and recall for a model's predictions of a given class.

For each benchmark model and MHA network, an accuracy score is computed using predictions on the test dataset from the final trained network produced by each cross-validation fold. This results in five measurements of accuracy, on which the mean accuracy is computed with a 95% confidence interval. Similarly, we independently compute the F1-score for each of the fault classes. We additionally average the fault class-specific F1-scores to report a mean F1-score over all fault classes.

7 Results and Discussion

Using the test dataset presented in Section 6, we evaluate the predictive performance of each benchmark model, as well as both our MHA-FCN and MHA-CNN approaches. Section 7.1 presents the results of our noise injection experiment. Additionally, Section 7.2 showcases a unique probability visualization enabled by our attention-based approach that enhances the human interpretability of our model's predictions.

7.1 Model Robustness to Noise Injection

We present the performance of the five benchmark models and the two proposed MHA networks on different levels of injected noise in Tables 5-8. Across the experiment, we observe that the addition of stronger noise results in a decrease in classification performance for every model. The increased noise likely decreases the distinction between the fault classes, in particular between the FDI and DoS classes. From the class-specific F1 scores, we note that the F1 score for these two classes tends to decrease most rapidly in the higher noise scenario. From Figure 4a, we observe that these two faults result in very similar

responses even in the no-noise scenario, thus the addition of noise likely further obfuscates the differences between the two fault classes. On the other hand, the actuator fault class and drunk human driver abnormality are more easily distinguishable and retain high F1 scores for many models, even in the sensor data corruption environment ($\sigma^2 = 0.2$). We hypothesize that the velocity profiles emitted by platoons containing a drunk driver already resemble a noisy signal, as suggested by Figure 4b, and thus the additional noise is not misinterpreted by the models. The actuator fault class is also unique in that it is characterized by a lower response amplitude, as shown in Figure 4a. The addition of noise does not distort the amplitude of the signal by a constant amount, thus the model is able to retain higher scores in identifying the actuator fault class due to the distinctiveness of this characteristic.

At a model level, we observe individual differences in performance across the different noise injection levels that reflect each model's properties. The SVM approach achieves a low mean accuracy of 62.6% on the no-noise scenario but retains a similar performance across all levels of noise injection. Since SVMs establish a decision boundary based on a few key support vectors, the SVM likely learns a highly generalized decision rule that allows it to remain resilient to even noisy measurements. However, SVMs only compute convex loss functions, which inherently restricts their hypothesis space. For the complex task of fault classification using velocity profiles, a non-convex hypothesis space may likely find more optimal predictive boundaries.

We see the benefits of more complex non-convex classifiers when analyzing the performance of MLPs, CNNs, and RNNs. In the no-noise scenario, each of these benchmark models outperforms the SVM. The MLP approach improves the accuracy to 81.8%, although the performance may be affected by the dynamics of our vehicle platoon. Since the vehicles change their desired velocity throughout a run, certain time-domain features which use the sample mean will be misleading. The CNN approach performs very well and achieves a 99.3% accuracy on the no-noise test set. The strong performance of this model can be explained by the benefit of using convolutional kernels to aggregate data from time steps within a local window. Since the distinguishing feature between fault classes is often the characteristics of the vehicle responses, observing a local context may help the CNN learn to extract features related to changes in the velocity profile over a platoon response as opposed to characteristics of the velocity profile at some particular time step. We expect a similar performance from the RNN model, which should learn to propagate sequence information across input time steps. Yet, the RNN model was only able to achieve a 67.8% accuracy on average, because the model was unable to learn any meaningful representation of the data on two of the five cross-validation folds. This finding is represented by the wide confidence interval on the reported accuracy. We postulate that the RNN, although designed for sequence process-

ing, still has challenges retaining information and propagating gradient updates over highly lengthy sequences, which in our case is 500 time steps. In fact, no learning was observed for any of the RNN models until the data was truncated from a length of 500 to 150 time steps, an additional pre-processing step that was mandatory only for training the RNN.

The concept of attention is designed to counteract this phenomenon of amnesia over highly lengthy sequences in RNNs. Using attention, a model can learn to prioritize specific time steps in its predictions by computing a set of attention weights. These attention weights can further improve learning stability by providing short-circuit connections for gradients to propagate to the salient time steps of the input sequence. We observe the improvement enabled by attention in the benchmark MSALSTM-CNN model, MHA-FCN and MHA-CNN approaches. All of these models surpass 98% prediction accuracy, which matches the performance of the CNN. However, unlike CNN, attention-based models retain their high performance as increasing noise is injected. While other approaches are brittle and quickly degrade in performance with more noise, we find that the attention-based models are much less impacted. In the heavy noise scenario with $\sigma^2 = 0.1$, the MSALSTM-CNN model and MHA-CNN model are still able to maintain greater than 90% predictive accuracy. Only when $\sigma^2 = 0.2$ does the performance of these two models decrease significantly, but even so, they remain the top approaches among the examined models in this study. These results suggest that the attention mechanism is effective in reducing the impact of sensor noise on fault classification performance. By only focusing on highly informative time steps in the input velocity profile, the attention-based models are only impacted by noise at those specific informative time steps, whereas models without attention suffer from a compounding effect of noise over the entire data sample since each time step is treated equally.

Among our two proposed MHA approaches, the MHA-CNN consistently outperforms the MHA-FCN approach. By using the ability of CNNs to aggregate data from a local frame of time steps, the MHA-CNN possesses more informative embeddings compared to the MHA-FCN. The performance discrepancies between the MHA-FCN and MHA-CNN can likely thus be attributed to the quality of the embedding used by the multi-head attention modules. The MHA-FCN approach can only provide a one-to-one mapping between its attention weights and the input velocity profile time steps, while an attention weight computed in the MHA-CNN approach measures the importance of a particular segment of the input sequence. On the other hand, the benchmark MSALSTM-CNN approach is able to slightly outperform the MHA-CNN approach in all our experiments, albeit at a small performance differential. However, the use of an LSTM network in the MSALSTM-CNN model requires the processing of the velocity profile in sequence, whereas the MHA-CNN approach uses scaled-dot product attention to process all time steps simultaneously [10], which enables it to train more quickly

than the MSALSTM-CNN.

Table 5: Fault classification with $\sigma^2 = 0$ noise injection.

Model	Mean Accuracy	Mean F1	Actuator F1	FDI F1	DoS F1	Distracted F1	Drunk F1
SVM	0.626 ± 0.020	0.627	0.924	0.424	0.388	0.439	0.960
MLP	0.818 ± 0.030	0.817	0.993	0.944	0.595	0.560	0.993
CNN	0.993 ± 0.003	0.993	1.000	0.990	0.984	0.993	0.999
RNN	0.678 ± 0.440	0.630	0.615	0.628	0.535	0.634	0.737
MSALSTM-CNN	0.996 ± 0.004	0.996	1.000	0.998	0.993	0.991	0.998
MHA-FCN	0.982 ± 0.012	0.982	0.999	0.956	0.955	0.998	1.000
MHA-CNN	0.995 ± 0.004	0.995	1.000	0.989	0.988	0.999	1.000

Table 6: Fault classification with $\sigma^2 = 0.05$ noise injection.

Model	Mean Accuracy	Mean F1	Actuator F1	FDI F1	DoS F1	Distracted F1	Drunk F1
SVM	0.624 ± 0.023	0.626	0.921	0.419	0.392	0.437	0.961
MLP	0.375 ± 0.046	0.334	0.368	0.405	0.156	0.124	0.618
CNN	0.824 ± 0.156	0.807	0.997	0.638	0.594	0.832	0.975
RNN	0.609 ± 0.365	0.558	0.543	0.529	0.395	0.604	0.720
MSALSTM-CNN	0.987 ± 0.007	0.987	1.000	0.981	0.971	0.986	0.996
MHA-FCN	0.973 ± 0.015	0.973	0.999	0.931	0.941	0.992	1.000
MHA-CNN	0.986 ± 0.008	0.986	1.000	0.969	0.967	0.993	0.999

Table 7: Fault classification with $\sigma^2 = 0.1$ noise injection.

Model	Mean Accuracy	Mean F1	Actuator F1	FDI F1	DoS F1	Distracted F1	Drunk F1
SVM	0.619 ± 0.013	0.625	0.911	0.414	0.396	0.441	0.963
MLP	0.296 ± 0.013	0.243	0.261	0.349	0.093	0.059	0.453
CNN	0.595 ± 0.197	0.545	0.934	0.190	0.225	0.497	0.878
RNN	0.399 ± 0.155	0.326	0.272	0.255	0.119	0.374	0.608
MSALSTM-CNN	0.921 ± 0.063	0.918	0.978	0.827	0.839	0.951	0.995
MHA-FCN	0.832 ± 0.065	0.799	0.990	0.373	0.747	0.890	0.995
MHA-CNN	0.900 ± 0.050	0.897	1.000	0.787	0.777	0.923	0.999

Table 8: Fault classification with $\sigma^2 = 0.2$ noise injection.

Model	Mean Accuracy	Mean F1	Actuator F1	FDI F1	DoS F1	Distracted F1	Drunk F1
SVM	0.559 ± 0.026	0.573	0.839	0.326	0.415	0.406	0.880
MLP	0.246 ± 0.012	0.187	0.180	0.304	0.044	0.039	0.368
CNN	0.412 ± 0.177	0.342	0.574	0.047	0.106	0.255	0.728
RNN	0.228 ± 0.033	0.108	0.022	0.021	0.004	0.152	0.339
MSALSTM-CNN	0.6752 ± 0.181	0.650	0.794	0.403	0.467	0.642	0.943
MHA-FCN	0.4708 ± 0.169	0.381	0.230	0.029	0.381	0.475	0.792
MHA-CNN	0.646 ± 0.114	0.599	0.892	0.210	0.333	0.646	0.915

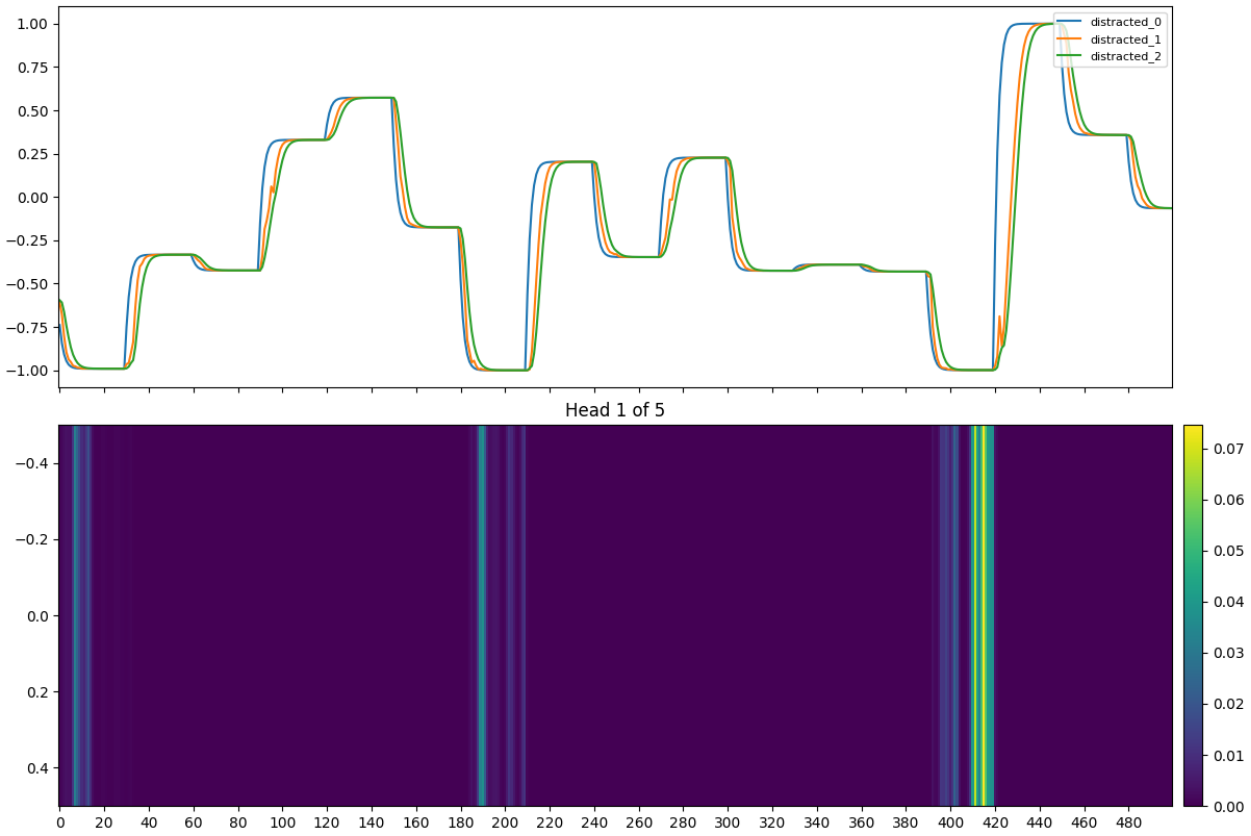


Figure 11: Example of a probability heatmap showing the relative strength of the attention weights emitted by the first head of a multi-head attention calculation for each time step of a sample velocity profile.

7.2 Attention Probability Visualization

The MHA-FCN uses a linear projection embedding approach that allows the model to compute attention weights that directly correspond to time steps in the input velocity profile. Although this embedding type results in slightly poorer classification performance, the one-to-one property of the attention weights enables interpretability of the MHA-FCN's predictions by a human observer. In contrast, it is more difficult to interpret the attention weights for the MHA-CNN and MSALSTM-CNN approaches because the initial CNN embedding layer aggregates elements of the input sequence and occludes these one-to-one relationships.

Figure 11 provides an example use case of the attention weights in the form of a probability heatmap of input time steps. From the example, we observe that for this particular sample, the MHA-FCN model tends to prioritize time steps around large changes in desired velocities when predicting the fault class. These larger velocity switches result in more prominent differences in platoon vehicle response characteristics, so the model has likely learned that greater amounts of information about the fault class can be extracted by observing the platoon behaviour at these points.

8 Conclusion

Fault classification is a critical task for the safe integration of CAV technology into modern transportation infrastructure. This work presents two novel deep-learning architectures based on multi-head attention, MHA-FCN and MHA-CNN, that can predict fault classes in CAVs with high accuracy. We further extend the fault classification task to consider human driver abnormalities through a mixed vehicle platoon formulation. Our approach is thoroughly compared with existing machine learning models from similar classification tasks in the literature. Our MHA-CNN variant achieves performance on par with current state-of-the-art approaches but with lower computational complexity. The benefit of including attention is also validated for real-world scenarios through a sensor noise injection experiment, where it is empirically demonstrated that attention-based models remain robust in high sensor noise environments. Further, the use of attention enables the creation of human interpretable visualizations of model predictions. We show this by using attention weights from the MHA-FCN model to generate a probability heatmap that clearly emphasizes important time steps used to construct the model's fault class prediction. Given these benefits, we encourage attention-based approaches as a gold standard for future works in CAV fault management.

We see numerous potential real-world applications for fault classification approaches such

as the one presented in this work. Primarily, a fault classification module could be a critical component of an onboard dynamic fault management algorithm present in CAVs. Sensor measurements and communication links could be used to construct the velocity profiles of surrounding vehicles in the platoon, which would then be monitored for potential faults. Alternatively, provided sufficient supporting infrastructure, fault classification could be used in a roadside platoon health monitoring system that communicates with CAV platoons in range. The responsible authorities could then be notified when a platoon exhibits certain types of faulty behaviour. In the case of CAV platoon collisions, fault classification also provides a method for the assignment of responsibility in automotive insurance. The sensor measurements recorded by the black box in each vehicle prior to the collision could be analyzed to determine the cause of the accident.

Future work is represented by validating our fault classification approach in different laboratory platoon environments, jointly tackling the fault classification and detection problem by including a healthy platoon class, exploring the fault classification problem in a multi-class setting where multiple faults may occur simultaneously, and introducing stochastic, time-varying effects to better simulate the real-world environment, such as through emulating weather or different road conditions. These extensions to the work aim to decrease the discrepancies between our current problem formulation and the conditions that could be encountered during real-world deployment.

References

[1] B. Dowdeswell, R. Sinha, and S. G. MacDonell, "Finding faults: A scoping study of fault diagnostics for industrial cyber-physical systems," *Journal of systems and software*, vol. 168, p. 110638, 2020.

[2] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles."

[3] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of modern transportation*, vol. 24, no. 4, pp. 284–303, 2016.

[4] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.

[5] S. E. Shladover, "Connected and automated vehicle systems: Introduction and overview," *Journal of Intelligent Transportation Systems*, vol. 22, no. 3, pp. 190–200, 2018.

[6] P. Kopelias, E. Demiridi, K. Vogiatzis, A. Skabardonis, and V. Zafiropoulou, "Connected & autonomous vehicles–environmental impacts–a review," *Science of the total environment*, vol. 712, p. 135237, 2020.

[7] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (cavs)," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[8] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[9] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "Roadmap for cybersecurity in autonomous vehicles," *IEEE Consumer Electronics Magazine*, 2022.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[11] J. G. Bender, "An overview of systems studies of automated highway systems," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 82–99, 1991.

[12] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[13] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Transactions on intelligent transportation systems*, vol. 15, no. 1, pp. 296–305, 2013.

[14] "Driving connectedness and leveraging capabilities in autonomous vehicle development," Siemens, Tech. Rep., 2021.

[15] S. E. Shladover, D. Su, and X.-Y. Lu, "Impacts of cooperative adaptive cruise control on freeway traffic flow," *Transportation Research Record*, vol. 2324, no. 1, pp. 63–70, 2012.

[16] J. Ploeg, A. F. Serrarens, and G. J. Heijenk, "Connect & drive: design and evaluation of cooperative adaptive cruise control for congestion reduction," *Journal of Modern Transportation*, vol. 19, no. 3, pp. 207–213, 2011.

[17] L. Ye and T. Yamamoto, "Evaluating the impact of connected and autonomous vehicles on traffic safety," *Physica A: Statistical Mechanics and its Applications*, vol. 526, p. 121009, 2019.

[18] I. G. Jin, S. S. Avedisov, C. R. He, W. B. Qin, M. Sadeghpour, and G. Orosz, "Experimental validation of connected automated vehicle design among human-driven vehicles," *Transportation research part C: emerging technologies*, vol. 91, pp. 335–352, 2018.

[19] C. Hendrickson, A. Biehler, and Y. Mashayekh, "Connected and autonomous vehicles 2040 vision," *Harrisburg, PA: Pennsylvania Department of Transportation*, 2014.

[20] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent transportation systems*, vol. 16, no. 2, pp. 546–556, 2014.

[21] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[22] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Computers & Security*, vol. 109, p. 102269, 2021.

[23] S. Feng, Y. Zhang, S. E. Li, Z. Cao, H. X. Liu, and L. Li, "String stability for vehicular platoon control: Definitions and analysis methods," *Annual Reviews in Control*, vol. 47, pp. 81–97, 2019.

[24] A. Alipour-Fanid, M. Dabaghchian, and K. Zeng, "Impact of jamming attacks on vehicular cooperative adaptive cruise control systems," *IEEE transactions on vehicular technology*, vol. 69, no. 11, pp. 12 679–12 693, 2020.

[25] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Preprints of the 1st workshop on Secure Control Systems*, vol. 1, 2010.

[26] Q. He, X. Meng, and R. Qu, "Survey on cyber security of cav," in *2017 Forum on cooperative positioning and service (CPGPS)*. IEEE, 2017, pp. 351–354.

[27] S. Dadras, R. M. Gerdes, and R. Sharma, "Vehicular platooning in an adversarial environment," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015, pp. 167–178.

[28] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[29] J. Ploeg, E. Semsar-Kazerooni, G. Lijster, N. van de Wouw, and H. Nijmeijer, "Graceful degradation of cacc performance subject to unreliable wireless communication," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 1210–1216.

[30] X. Huang and J. Dong, "Reliable control policy of cyber-physical systems against a class of frequency-constrained sensor and actuator attacks," *IEEE Transactions on Cybernetics*, vol. 48, no. 12, pp. 3432–3439, 2018.

[31] A. Petrillo, A. Pescape, and S. Santini, "A secure adaptive control for cooperative driving of autonomous connected vehicles in the presence of heterogeneous communication delays and cyberattacks," *IEEE transactions on cybernetics*, vol. 51, no. 3, pp. 1134–1149, 2020.

[32] Z. A. Biron, S. Dey, and P. Pisu, "Resilient control strategy under denial of service in connected vehicles," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4971–4976.

[33] H. Zhang and J. Wang, "Active steering actuator fault detection for an automatically-steered electric ground vehicle," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 3685–3702, 2016.

[34] G. Guo, P. Li, and L.-Y. Hao, "Adaptive fault-tolerant control of platoons with guaranteed traffic flow stability," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 6916–6927, 2020.

[35] F. Farivar, M. S. Haghighi, A. Jolfaei, and S. Wen, "On the security of networked control systems in smart vehicle and its adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3824–3831, 2021.

[36] A. Sargolzaei, C. D. Crane, A. Abbaspour, and S. Noei, "A machine learning approach for fault detection in vehicular cyber-physical systems," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 636–640.

[37] Y. Fang, H. Min, W. Wang, Z. Xu, and X. Zhao, "A fault detection and diagnosis system for autonomous vehicles based on hybrid approaches," *IEEE Sensors Journal*, vol. 20, no. 16, pp. 9359–9371, 2020.

[38] Z. Abdollahi Biron, S. Dey, and P. Pisu, "Sensor fault diagnosis of connected vehicles under imperfect communication network," in *Dynamic Systems and Control Conference*, vol. 50695. American Society of Mechanical Engineers, 2016, p. V001T16A003.

[39] Q. Shi and H. Zhang, "Fault diagnosis of an autonomous vehicle with an improved svm algorithm subject to unbalanced datasets," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 7, pp. 6248–6256, 2020.

[40] S. Safavi, M. A. Safavi, H. Hamid, and S. Fallah, "Multi-sensor fault detection, identification, isolation and health forecasting for autonomous vehicles," *Sensors*, vol. 21, no. 7, p. 2547, 2021.

[41] A. Khalil and M. Al Janaideh, "On fault classification in connected autonomous vehicles using supervised machine learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1198–1204.

[42] F. Van Wyk, Y. Wang, A. Khojandi, and N. Masoud, "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1264–1276, 2019.

[43] T. Alladi, V. Kohli, V. Chamola, and F. R. Yu, "Securing the internet of vehicles: A deep learning-based classification framework," *IEEE networking letters*, vol. 3, no. 2, pp. 94–97, 2021.

[44] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan, and M. S. Haghighi, "Anomaly detection in automated vehicles using multistage attention-based convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4291–4300, 2020.

[45] G. Xu, M. Liu, Z. Jiang, W. Shen, and C. Huang, "Online fault diagnosis method based on transfer convolutional neural networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 2, pp. 509–520, 2019.

[46] Z. Wu, Y. Guo, W. Lin, S. Yu, and Y. Ji, "A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems," *Sensors*, vol. 18, no. 4, p. 1096, 2018.

[47] W. Cui, X. Deng, and Z. Zhang, "Improved convolutional neural network based on multi-head attention mechanism for industrial process fault classification," in *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE, 2020, pp. 918–922.

[48] L. Zhang and E. Tseng, "Motion prediction of human-driven vehicles in mixed traffic with connected autonomous vehicles," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 398–403.

[49] R. A. Biroon, Z. A. Biron, and P. Pisu, "False data injection attack in a platoon of cacc: real-time detection and isolation with a pde approach," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[50] R. Loureiro, S. Benmoussa, Y. Touati, R. Merzouki, and B. O. Bouamama, "Integration of fault diagnosis and fault-tolerant control for health monitoring of a class of mimo intelligent autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 1, pp. 30–39, 2014.

[51] A. Khalil, K. F. Aljanaideh, and M. Al Janaideh, "On detecting drunk drivers in mixed autonomous platoons using vehicles velocity measurements," *IEEE/ASME Transactions on Mechatronics*, Early Access, 2022.

[52] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, pp. 1805–1824, 2000.

[53] M. Pirani, E. Hashemi, A. Khajepour, B. Fidan, B. Litkouhi, S.-K. Chen, and S. Sundaram, "Cooperative vehicle speed fault diagnosis and correction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 783–789, 2018.

[54] G. Zhang, H. Zhang, X. Huang, J. Wang, H. Yu, and R. Graaf, "Active fault-tolerant control for electric vehicles with independently driven rear in-wheel motors against certain actuator faults," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1557–1572, 2015.

[55] M. Emily Parcell, M. Shivani Patel, C. Severin, Y. Cho, and A. Chaparro, "Effect of driver distraction on vehicle speed control," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 65, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2021, pp. 958–962.

[56] N. H. T. S. A. (NHTSA), "The effects of blood alcohol concentration," pp. [Online]. Available: https://www.nhtsa.gov/risky–driving/drunk–driving.

[57] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[58] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[61] A. Girma, X. Yan, and A. Homaifar, "Driver identification based on vehicle telematics data using lstm-recurrent neural network," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 894–902.

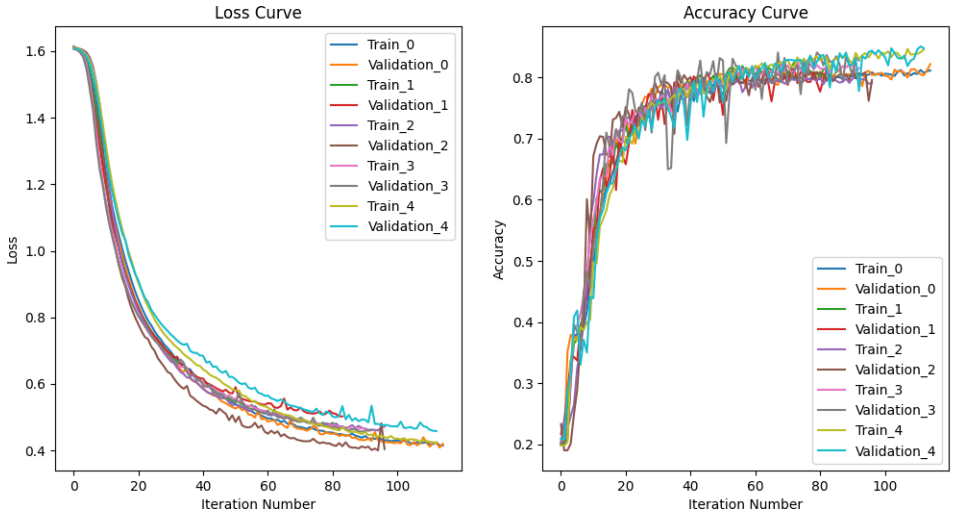
[62] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[63] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

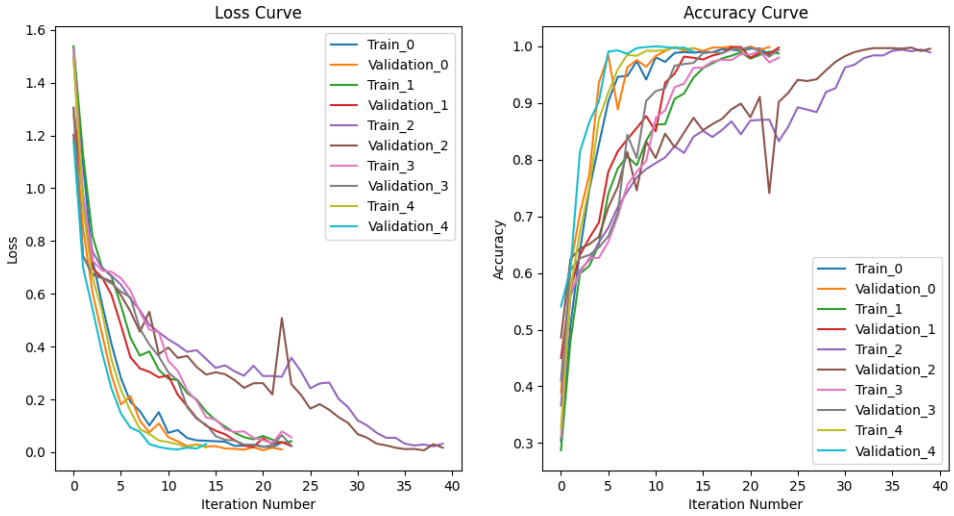
[64] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[65] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

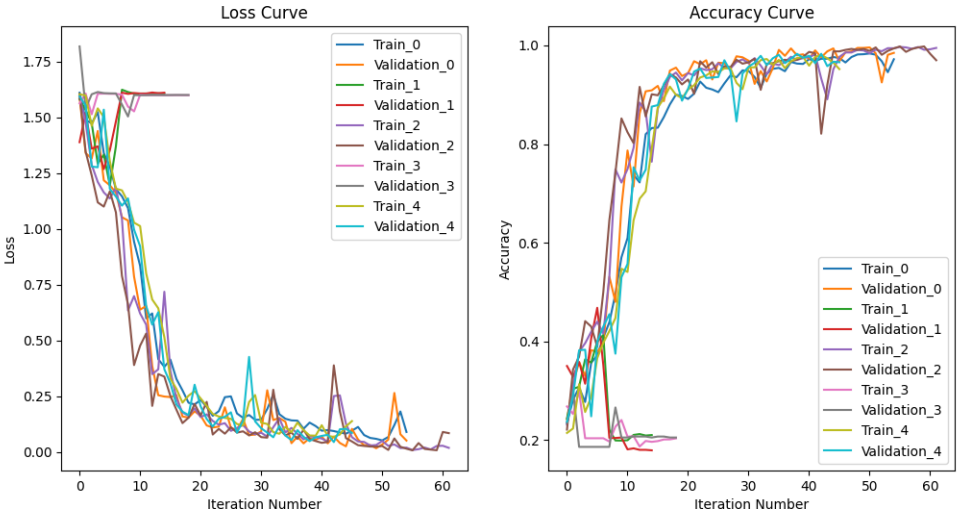
A Model Training Curves



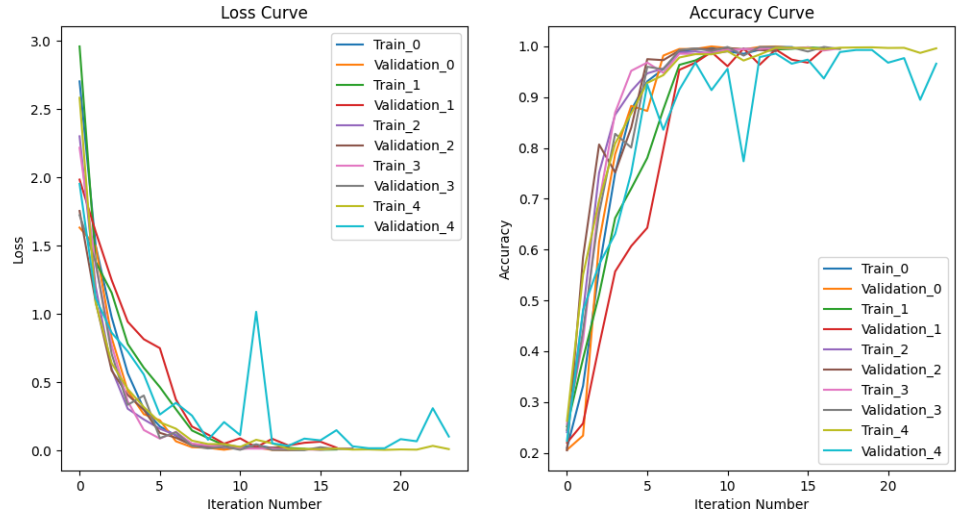
(a) Training curves for the MLP benchmark model.



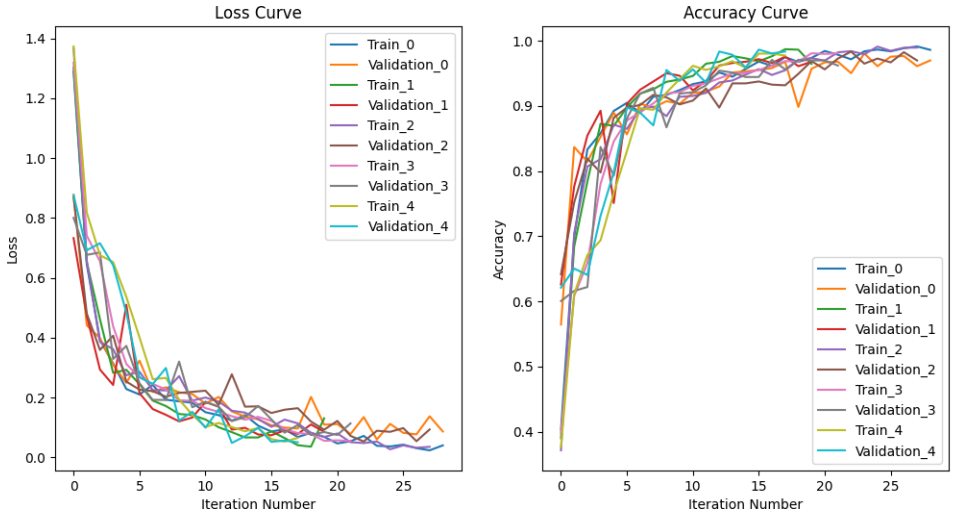
(b) Training curves for the CNN benchmark model.



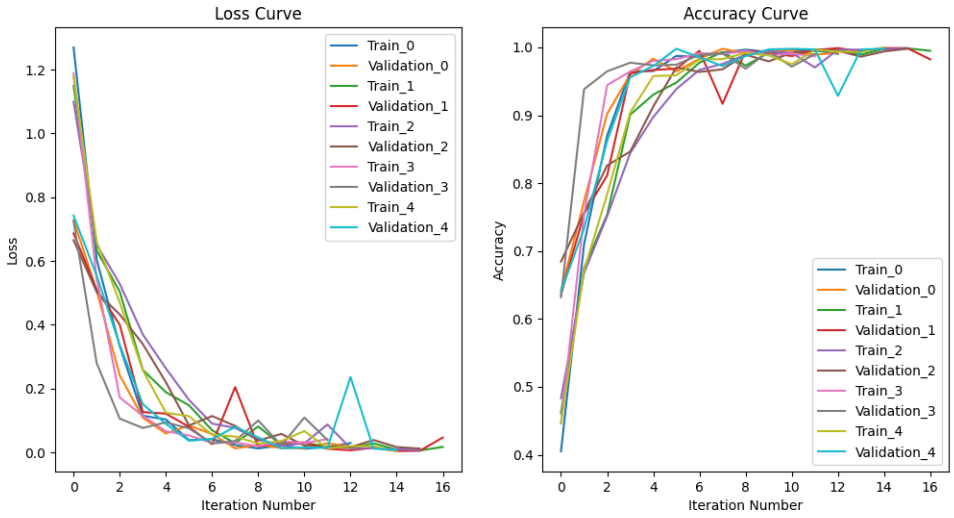
(c) Training curves for the RNN benchmark model.



(d) Training curves for the MSALSTM-CNN benchmark model.



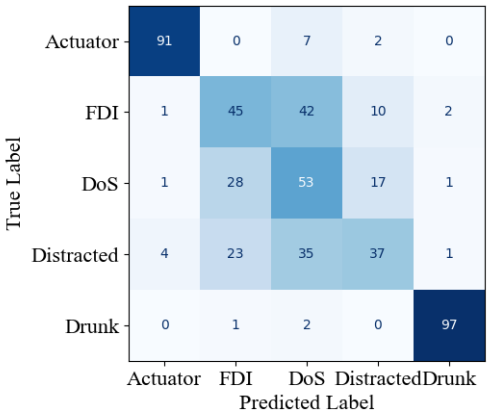
(e) Training curves for the proposed MHA-FCN model.



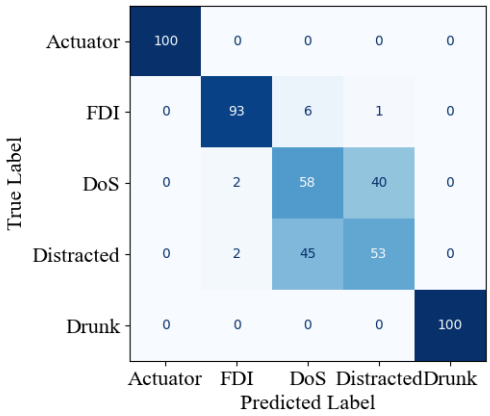
(f) Training curves for the proposed MHA-CNN model.

Figure 12: Training and validation loss and accuracy curves for each examined machine learning model on all five cross-validation folds. Note that the SVM is not represented because it is not trained using a gradient descent algorithm.

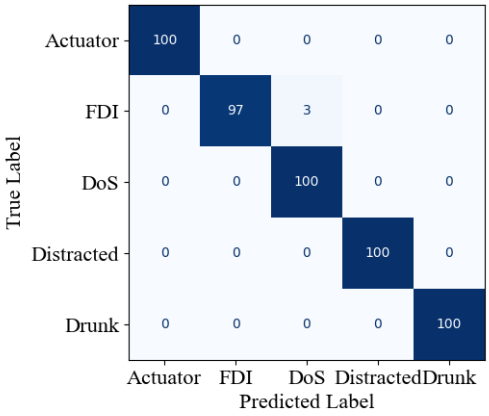
B Confusion Matrices of Model Predictions



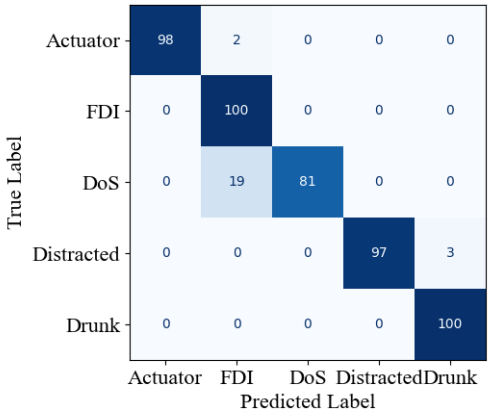
(a) SVM predictions.



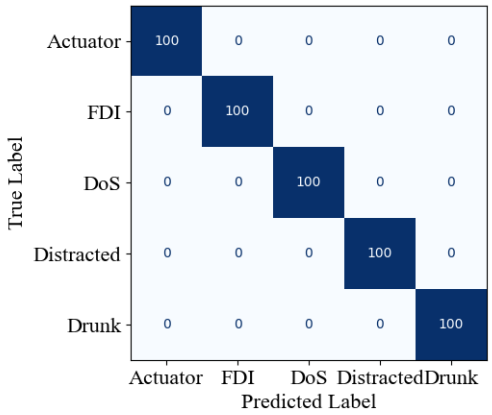
(b) MLP predictions.



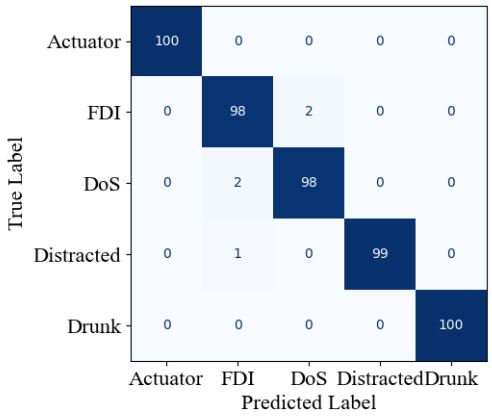
(c) CNN predictions.



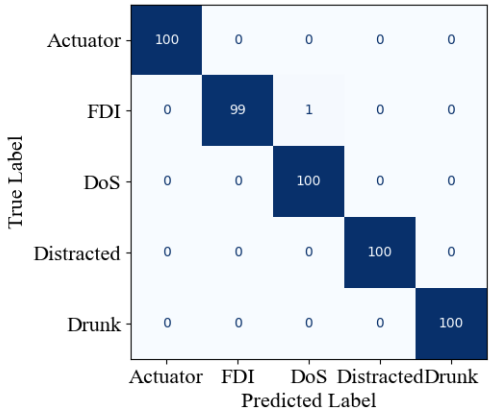
(d) RNN predictions.



(e) MSALSTM-CNN predictions.

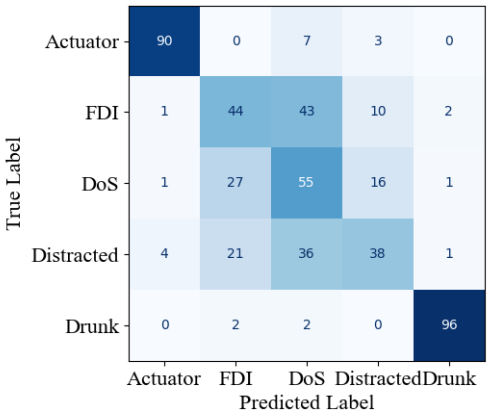


(f) MHA-FCN predictions.

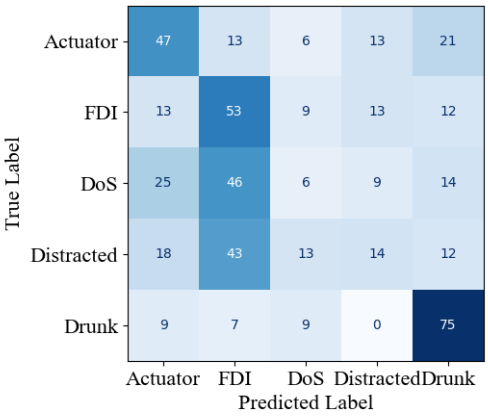


(g) MHA-CNN predictions.

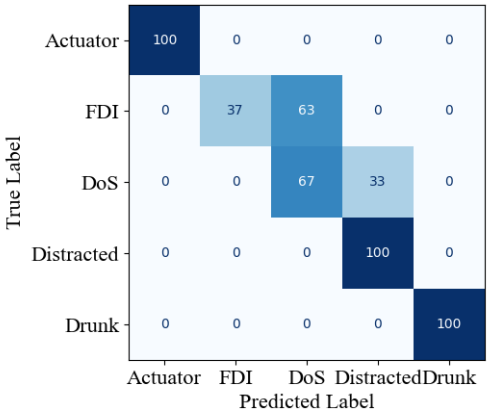
Figure 13: Confusion matrices of model predictions under the $\sigma^2 = 0$ noise level. For visual clarity, only predictions for the model trained using the first cross-validation fold are used.



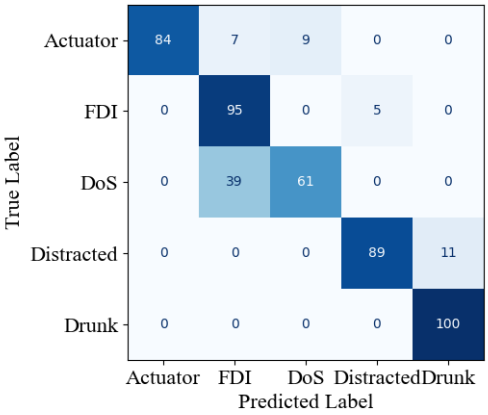
(a) SVM predictions.



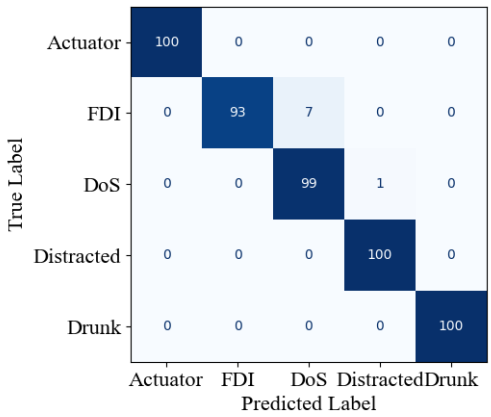
(b) MLP predictions.



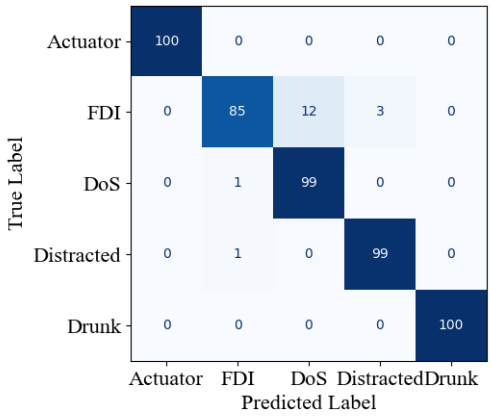
(c) CNN predictions.



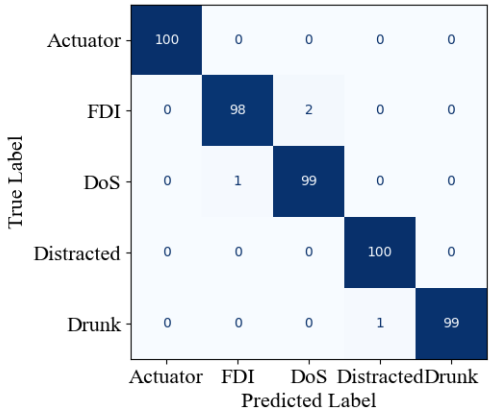
(d) RNN predictions.



(e) MSALSTM-CNN predictions.

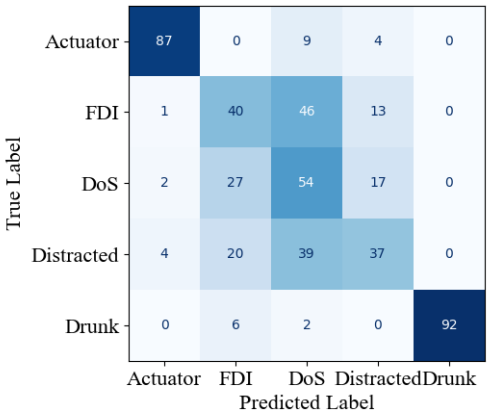


(f) MHA-FCN predictions.

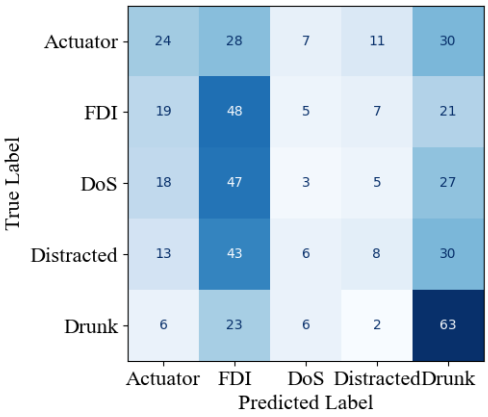


(g) MHA-CNN predictions.

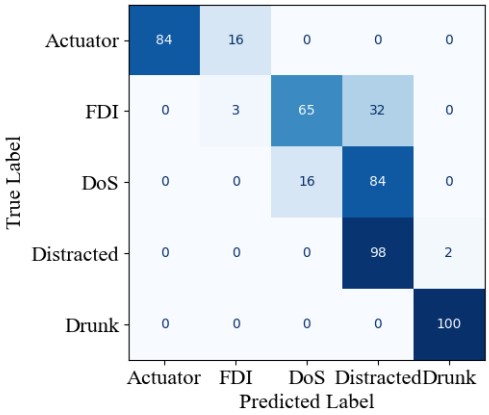
Figure 14: Confusion matrices of model predictions under the $\sigma^2 = 0.05$ noise level. For visual clarity, only predictions for the model trained using the first cross-validation fold are used.



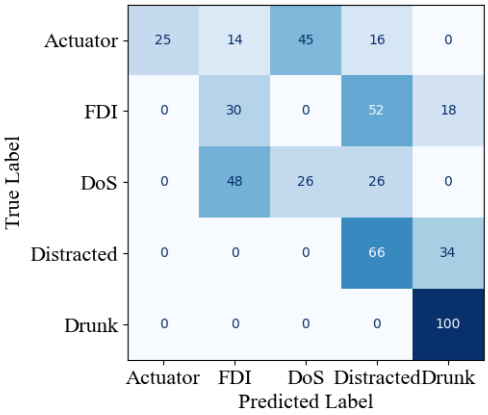
(a) SVM predictions.



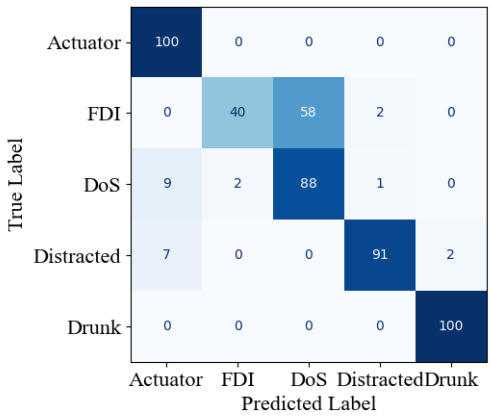
(b) MLP predictions.



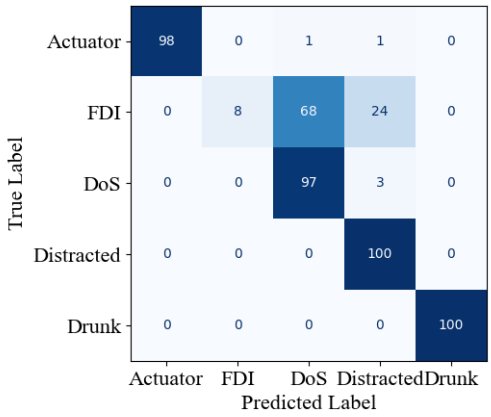
(c) CNN predictions.



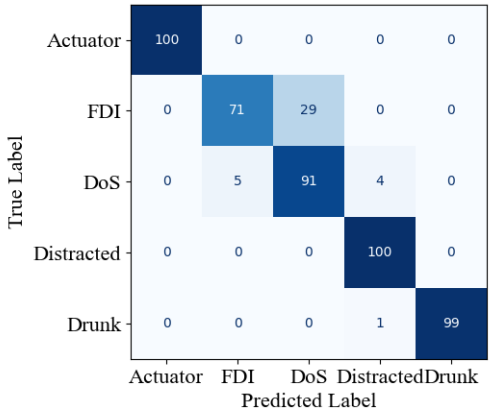
(d) RNN predictions.



(e) MSALSTM-CNN predictions.

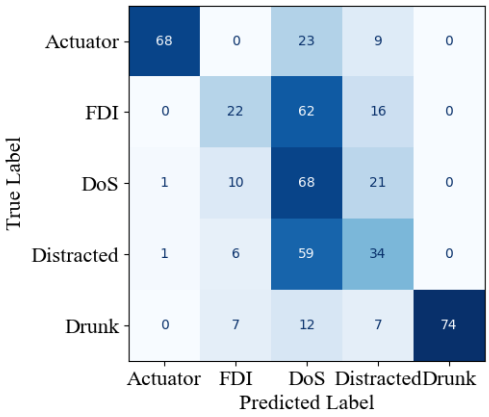


(f) MHA-FCN predictions.

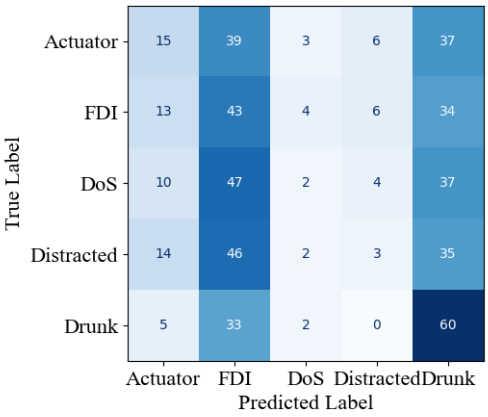


(g) MHA-CNN predictions.

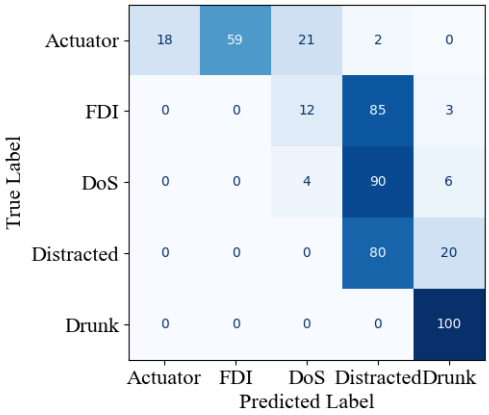
Figure 15: Confusion matrices of model predictions under the $\sigma^2 = 0.1$ noise level. For visual clarity, only predictions for the model trained using the first cross-validation fold are used.



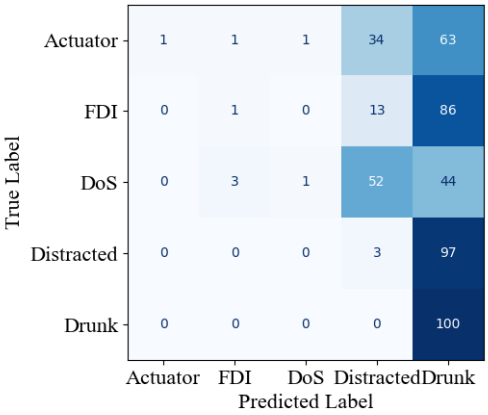
(a) SVM predictions.



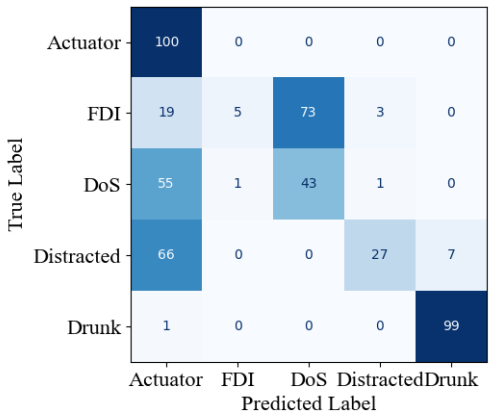
(b) MLP predictions.



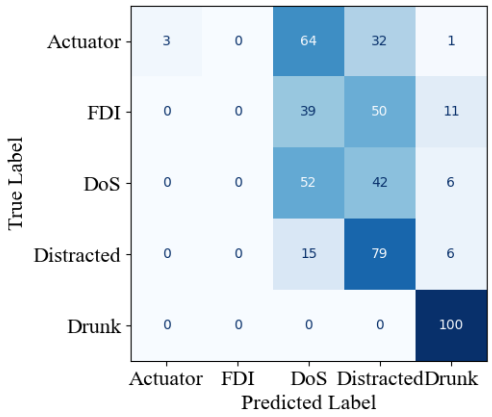
(c) CNN predictions.



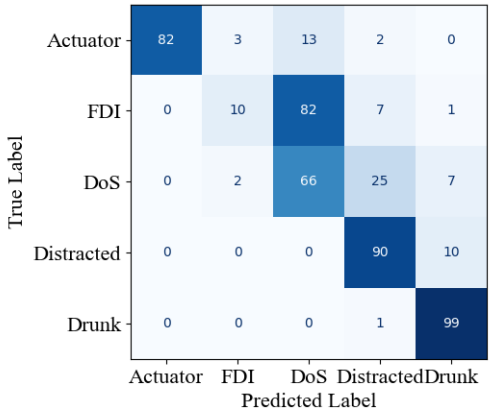
(d) RNN predictions.



(e) MSALSTM-CNN predictions.



(f) MHA-FCN predictions.



(g) MHA-CNN predictions.

Figure 16: Confusion matrices of model predictions under the $\sigma^2 = 0.2$ noise level. For visual clarity, only predictions for the model trained using the first cross-validation fold are used.

C Code Availability

All code used for machine learning model creation, training, and evaluation are publicly available at `https://github.com/theowu23451/fault-classification-attention-network`