Unsupervised Anomaly Detection for Automotive CAN Bus on the Intel Loihi

Rashedul Islam
Department of Electrical and Computer Engineering
University of Dayton
Dayton, Ohio, USA
islamm24@udayton.edu

Shahanur Alam
Department of Electrical and Computer Engineering
University of Dayton
Dayton, Ohio, USA
alamm8@udayton.edu

Chris Yakopcic
Department of Electrical and Computer Engineering
University of Dayton
Dayton, Ohio, USA
cyakopcic1@udayton.edu

Nayim Rahman
Department of Electrical and Computer Engineering
University of Dayton
Dayton, Ohio, USA
rahman12@udayton.edu

Simon Khan
Air Force Research Laboratory
Rome, NY, USA
simon.khan@us.af.mil

Tarek Taha
Department of Electrical and Computer Engineering
University of Dayton
Dayton, Ohio, USA
tarek.taha@udayton.edu

***Abstract*—Detecting anomalies and faults swiftly and accurately is essential in many applications, such as healthcare, infrastructure, industry, and security. In most of these applications, the available power for running an anomaly detection system is very limited. This is especially the case for modern smart and electric vehicles, which have an increasing amount of electronics. These vehicles have Electronic Control Units (ECUs) that communicate with each other through the Controller Area Network (CAN) bus. The Controller Area Network, however, is not very secure, so bad actors can do severe damage, putting the lives of drivers and passengers at risk. Thus, low-power anomaly detection is essential for both security and reliability. Deep learning networks have demonstrated proficiency in anomaly identification. However, their implementation on conventional Central Processing Units (CPUs) and Graphics Processing Units (GPUs) incurs significant energy consumption. To overcome this limitation, we propose a low-power approach for placing security directly in the network hardware. An autoencoder-based real-time anomaly detector trained through unsupervised learning is seamlessly mapped onto the Intel Loihi spiking neuromorphic processor. Our proposed anomaly detector used significantly less power than a CPU or GPU and showed the best accuracy and F-1 score to detect anomalies compared to the alternative spiking approaches. The proposed anomaly detection system consumed about 20,000 and 700 times less energy than a CPU and GPU respectively. To the best of our knowledge, this is the first low-power, unsupervised anomaly detection system using the Loihi or any other neuromorphic processor. The results of this work will apply to other anomaly detection applications, as well as other spiking neuromorphic processors, resulting in a universally applicable extreme low-power anomaly detection platform.**

Keywords—CAN bus, Intel Loihi, Unsupervised, Autoencoder, Low power, Anomaly detection

I. Introduction

Anomaly detection is a technique that concerns identifying patterns in data that deviate from the expected behavior. The accuracy of detection, precision, and faster diagnosis of such anomaly events are necessary for reducing significant damage and cost. Anomaly detection has a wide range of applications, including fraud detection in credit card transactions, loan processing, medical applications in health condition monitoring, detecting cyber security intrusions, and identifying faults in machinery. Machine learning algorithms (supervised and unsupervised) show tremendous success in anomaly detection operations, such as isolation forest [1], support vector machine [2], and k-means clustering [3]. In recent years, deep learning networks have shown more efficiency in detecting anomalies than traditional machine learning-based systems [4].

This paper focuses on developing a deep learning-based vehicular anomaly detection system. Modern automotive vehicles comprise up to 100 Electronic Control Units (ECUs) [5]. These ECUs control various car functions, such as steering, braking, engine, airbag, traction, and cruise control. ECUs share their information through a vehicle network called the Controller Area Network (CAN). The CAN bus is popular among major car manufacturers because it uses a standardized serial communication protocol for car control systems. It is known for being reliable, fast, and simple to setup [6]. The ECU gets updated data from sensors and executes different instructions to monitor the vehicle's status [7]. Since the CAN bus uses a broadcast protocol, the messages have no source and destination information. It does not support authentication and encryption. Therefore, it is easy for an attacker to attack the CAN bus and control the car through an external device or Bluetooth communication. Any attack on one ECU can immediately carry through to other ECUs. A corrupt ECU can seriously affect the passengers, drivers, and the surrounding environment. Therefore, anomaly detection must be integrated within the CAN environment to detect unwanted messages and enable a higher level of security.

Modern vehicles have many sensors that produce a large amount of data, and thus needs an anomaly detection system that is fast but consumes low power. Using CPUs or GPUs is for this would not be efficient as they are either not fast enough at network processing or consume too much power. For instance, a state-of-the-art GPU can consume about 200 W of power while computing at full load. A solution to this problem is to use neuromorphic processors to implement anomaly detection at extremely low power. Intel's Loihi is an example of a recent neuromorphic processor and consumes approximately 100 mW of power [8]. This computation efficiency is realized by its ability to mimic human cognitive systems for information processing. One of the critical features of neuromorphic processors is their ability to handle spiking neural networks (SNNs), which closely emulate the spiking behavior of the brain. This paper presents a spiking neural network-based anomaly detection algorithm using the Intel Loihi neuromorphic processor. We then evaluated the anomaly detection system for malicious packets on the automotive CAN bus.

To the best of our knowledge, this is the first unsupervised anomaly detection application implemented on the Loihi. We emulated our system using Nengo [9], which is a framework to solve neural network-based problems on neuromorphic hardware. This work is also the first-ever anomaly detection application implemented in neuromorphic hardware. Given the extreme advantage of novel spiking neural network hardware in terms of both energy efficacy and portability, this is very capable system to carry out security processes within complex but vulnerable in-vehicle computing networks.

The rest of the paper is organized as follows: related anomaly detection work in non-spiking and spiking systems is discussed in Section II. Section III discusses the background for this work. Section IV describes the dataset and the preprocessing required for our experiment. Section V introduces the autoencoder network. Section VI discusses the experimental setup for our system. Section VII analyzes the results, and Section VIII concludes the paper.

II. Related Works

In recent years, autonomous electric vehicles (EVs) have become very popular, and their use has increased rapidly. The rapid growth of EVs also increased the targets of cyberattacks on the CAN bus. Therefore, security of autonomous cars is crucial. Besides the automotive industry, the anomaly detection system has uses in finance, healthcare, and cybersecurity applications. Many research papers have looked at anomaly detection applications in recent years, most of which are pure software-based implementations without considering hardware mapping. In this section, some of the most relevant works are described briefly.

Hossain et al. [10] proposed a Long Short-Term Memory (LSTM) based anomaly detection technique that successfully classifies malicious messages from normal messages. They experimented on their own data with different hyper-parameter values and achieved around 99.99% detection accuracy. Seo et al. [11] demonstrate a GAN-based intrusion detection system (GIDS) consisting of a discriminator and generator. This model achieved an average of 98% accuracy in detecting unknown data. Song et al. [12] proposed a deep convolutional neural network (DCNN) based intrusion detection system in which they modified the Inception-ResNet model. The authors used a frame builder to extract a 29-bit identifier from a recent sequence of 29 CAN messages for training. They achieved a detection precision of 1.0 for DoS types of anomalies and 0.99 for other anomalies. Kwak et al. [13] highlight the significance of cosine similarity as a feature to detect and classify attacks from datasets. They achieved 98.93% and 99.18% accuracy for the KIA Soul and 99.43% and 99.49% for the HYUNDAI YF Sonata, respectively, in driving and stationary conditions. Jaoudi et al. [14] proposed a convolutional autoencoder to detect anomalies. The author converted the network into an SNN and showed that the network successfully detected anomalies.

Besides in-vehicle applications, there are many other anomaly detection applications in industrial, commercial, and healthcare institutions. Roth et al. [15] presented a method called PatchCore, which uses a maximally representative memory bank of nominal patch features to find defective parts in industrial manufacturing. The authors investigated the importance of locally aware patch features to show the changes in anomaly detection. Their methods achieved a 99.6% AUROC score on the MVTec AD benchmark. Vanini et al. [16] presented the development and validation of the risk management process using various fraud detection models. Alfardus et al. [17] presented a machine learning-based intrusion detection system with various algorithms like K-nearest neighbors, Random Forest, Support Vector Machine, and Multilayer perceptron and compared their accuracy. The accuracy of the proposed system is 100% for impersonation and DoS attacks and 99% for fuzzy attacks.

Zanatta et al. [18] proposed structural health monitoring (SHM) using spiking neural networks (SNNs) in MEMS data to detect infrastructural damages in a motorway bridge. They trained a Long Short-Term SNN (LSNN) with an e-prop algorithm and Backpropagation Through Time (BPTT). Dennler et al. [19] demonstrate spiking neural network (SNN) based vibration analysis, which is compatible with analog-digital neuromorphic circuits. They implemented a three-layer SNN on the Induced Bearing Fault (IBF) and Run-to-Failure Bearing Fault (R2F) datasets to detect anomalies from vibration data in an unsupervised fashion.

All the works described above are implemented in pure software only form for various anomaly detection applications. Only a few experiments have been reported that have been implemented in hardware. For example, Bauer et al. [20] proposed a real-time classification of ECG data on the Dynamic Neuromorphic Asynchronous Processor (DYNAP). They used a spiking recurrent neural network, which consists of three different neuronal populations. The authors converted ECG signals into sequential events by using the sigma-delta encoding scheme. Their system showed 91.3% anomaly detection on the test set and consumed less than 722.1 μW power. Chen et al.

[21] presented an autonomous anomaly recognition and detection (AnRAD) framework inspired by the human neocortex system to detect anomalies in traffic. They used a likelihood-ratio test to detect anomalies. Ussa et al. [22] proposed a hybrid neuromorphic framework for object tracking and classification. They embedded their system in an FPGA and interfaced with a TrueNorth (TN) chip. They compared their framework's performance with DART and SLAYER. They showed it also consumed low power, ranging from 5 to 14 mW. However, there is no neuromorphic hardware-based on an in-vehicle anomaly detection system available for automobiles.

III. Background

A. *Spiking Neural Network (SNN)*

In the human brain, neurons send signals using spikes of electrical activity through axons and collect signals from dendrites to exchange information. Neurons communicate through an extensive, interconnected network of synapses. A spiking neural network is a neural network model that mimics the human brain's computational mechanisms [23].

In a spiking neural network, each neuron actively monitors incoming spikes and assigns weights based on synaptic values. If the cumulative effect of these weighted spikes surpasses a threshold value, it will send a spike to connected neurons and initiate communication. After a spike, the neuron voltage is reset to a predetermined value. The training process in SNN involves adjusting the weights to optimize the network's ability to recognize the pattern and respond to the input spike. Both presynaptic and postsynaptic spikes play a crucial role in the training phase. Fig. 1 illustrates the structural components and functionality of a spiking neuron.

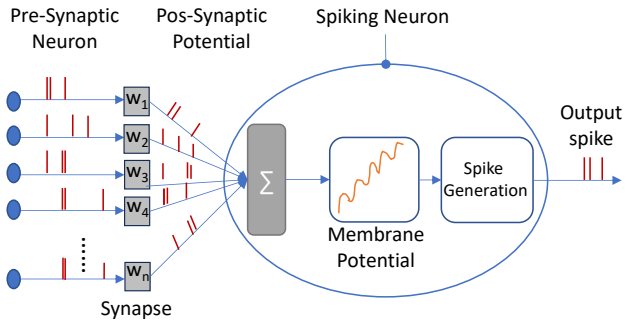

Fig. 1. Spiking Neural Network [24].

B. *Intel Loihi Neuromorphic Hardware*

Neuromorphic processors are specialized microprocessors. These processors are designed to mimic the structure and functionality of the human brain's neural networks. One of the critical features of neuromorphic processors is their ability to handle spiking neural networks (SNNs), which closely emulate the spiking behavior of neurons in the human brain. This event-driven approach allows for more efficient processing, as computations occur only when relevant spikes or signals are present.

Intel's Loihi was released in 2018 to support the properties of biological neural networks at extreme efficiency [25]. It is a 60 mm^2 neuromorphic chip fabricated using Intel's 14-nm FinFET process technology. The chip features include hierarchical connectivity, synaptic delay, and dendrite compartments. A total of 6.02 billion transistors and 128 neuromorphic cores constitute the Loihi chip. Each chip has three x86 processors to support Von Neumann processing. Each core contains 1024 neuronal compartments, and a 32-bit spiking-based message system holds communication between the cores, which enables efficient communication between them. It has a programmable synaptic learning system that supports various learning rules by evolving synaptic states. This learning engine has unique features and the power to implement spiking neural networks. The Loihi chip is available in multiple systems: Kapoho Bay, Nahuku, and Pohiki Springs [26].

IV. Dataset

A. *CAN Bus Dataframe*

The CAN is a standard communication protocol used in automotive vehicles. The data between the ECUs is shared through this network. Table I shows the standard format of a CAN data frame. The description of the data frame is as follows:

Table I: Format of CAN data frame [27].

	Field Name		Bit Size
	Bus Idle		
CAN Data frame	SOF		1
	Arbitration Field	RTR	1
		ID	11
	Control Field	DLC	4
		r0	1
		IDE	1
	Data Field		0,1,2,---,63
	CRC Field	CRC Seq.	15
		CRC Delimiter	1
	ACK Field	ACK	1
		ACK Delimiter	1
	Bus Idle		
	INT		3
	EOF		7

1. SOF – Start of Frame. It informs the start of messages.
2. Arbitration field – Consists of two values, ID and RTR. ID is the header of the frame. When multiple ECUs transmit data simultaneously, the data is prioritized using the ID. The lower the ID, the higher the priority. RTR stands for single Remote Transmission Request. When another node requires information, the RTR bit is dominant. All the nodes receive the request and the responding data.
3. Control field – The control field consists of three fields: IDE, r0, and DLC. The Identifier Extension (IDE) bit is dominant when transmitting a standard CAN identifier with no extension. A bit is reserved (r0) for future purposes. DLC holds 4-bit data containing the number of bytes (0 to 8) transmitted.

4. Data field – This field contains the data content being transmitted. It can hold up to 8 bytes of data.
5. CRC field – CRC stands for Cyclic Redundancy Check, which is used for error detection.
6. ACK field – This field acknowledges the transmitted error-free data the node receives.
7. EOF – End of the frame (EOF) indicates the end of the transmitted message [28].

This work uses the CAR hacking dataset the Hacking and Countermeasure Research Lab (HCRL) developed. This dataset includes five files. One is attack-free, and the four others contain normal messages in addition to injected messages. Injected messages are introduced by connecting the CAN protocol through an onboard diagnostic port (OBD-II) from a vehicle. The four attack types are DoS (Denial-of-Service), fuzzy, spoofing the drive gear, and spoofing the RPM gauge. CAN messages are sent every 0.3 ms for a DoS attack containing a CAN ID of 29 zero bits, the dominant CAN ID. The goal of these types of attacks is to consume network availability. The fuzzy attack is almost the same as the DoS. They injected random CAN ID and data values every 0.5 ms to construct a fuzzy attack. The fuzzy attack aims to cause the vehicle to malfunction. Spoofing of the drive gear and RPM gauge was performed by injecting messages every 1 ms of a specific CAN ID. These attacks can harm the drive gear and RPM gauge by deceiving the ECUs [29].

B. Data Pre-processing

The dataset has five attributes: Timestamp, the recorded message time in seconds; CAN ID, the CAN message identifier in HEX format; DLC, the length of data bytes up to 8; DATA [0~7], the transmitted payload data value; and flag, whether the data is normal or an attack. For our work, the unnecessary parts are removed from the dataset during preprocessing. We used 'arbitration' and data fields of CAN messages in our experiments as they captured the most valuable information of CAN messages. The data field consists of 8 bytes. If any one of the data fields is missing, then the information is padded with zero. Additionally, any null values within the data are substituted with zeros. The CAN bus data field is in hexadecimal format and converted into decimals for this experiment. The dataset is normalized by utilizing standard scores. We created a Python function to generate sequential data patterns for training and testing. The Python function uses time steps, ID, and data field values as inputs and converts them to sequences.

V. AUTOENCODER NETWORK MODELS

AutoEncoders (AE) are a generative neural architecture that can encode an input feature space to a bottleneck layer in the encoding state, and the decoding state learns to regenerate inputs in the output layer. A conceptual diagram of a simple autoencoder technique is presented in Fig. 2. The encoder section learns to compress the data samples into a lower-dimensional latent space at the bottleneck layer (Z). The decoder section learns to regenerate the input feature space at the output. Once the AE is trained, it becomes a very efficient machine for detecting anomalies. The AE network model is trained by backpropagation of error to minimize the reconstruction error of the decoder, according to the loss function in Eq. (1).

$$\ell(X, \hat{X}) = ||X - \hat{X}||^2 \quad (1)$$

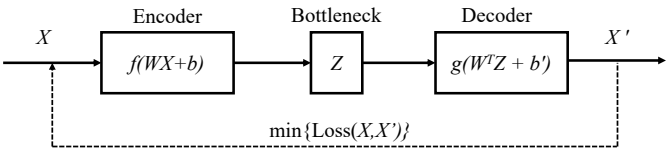

Fig. 2. Schematic block diagram of an autoencoder.

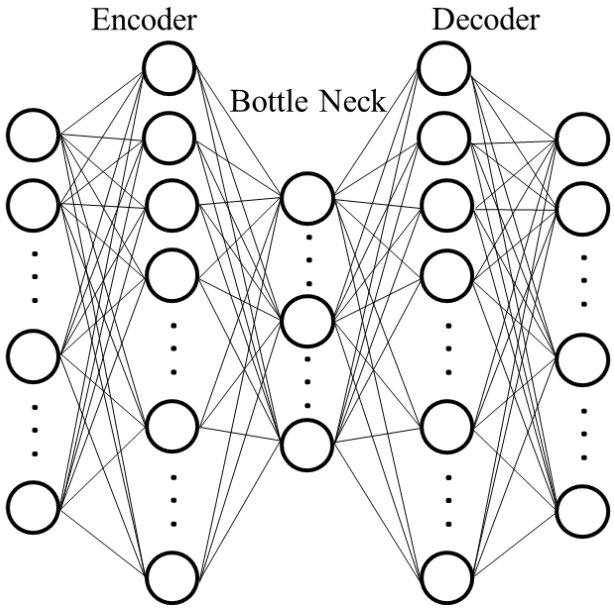

Fig. 3. Block diagram of a fully connected autoencoder.

The same technique can be applied for layer-wise data flow computation for autoencoders with multiple hidden layers. In this experiment, a fully connected Autoencoder (AE) is implemented in a spiking neural network for anomaly detection, as shown in Fig. 3. The inputs compressed in the encoder section produce the encoding at the bottleneck layer. Using the encoding at the bottleneck, the decoder section reconstructs the input.

During the training of an autoencoder, it learns the correlations among the features in the data. It can reconstruct a known or unknown dataset. The training dataset has lower reconstruction errors as the features become familiar over the training period. It creates higher reconstruction errors for unknown data. This work calculates the vector distance L between the input and predicted output using equation (2) for every sample, where a_i and b_i are the input and predicted output vector, respectively. We took the value of L as a threshold to detect anomalies.

$$L = max\,(|b_i - a_i|) \quad (2)$$

For anomaly detection, the incoming data passes through the network, and we calculate the vector distance. During inference operations, if the magnitude of the vector distance is greater than *L*, it identifies the test sample as an anomaly, and if the distance is less than *L,* it identifies it as normal.

VI. Experimental Setup

Fig. 4 presents the workflow of the proposed system for anomaly detection. At first, we trained the AE network with a training dataset. Next, the trained AE network is converted into a spiking AE network. After conversion, the spiking AE network was retrained with the same training dataset and trained weights. Finally, the required optimization was performed to map the trained spiking AE into the Loihi processor. The deep neural network model was first trained in Keras, then converted into spiking form by utilizing the NengoDL tool [30], and then mapped on the Loihi using NengoLoihi [31]. NengoDL and NengoLoihi are frameworks for building, testing, and deploying neural networks on different hardware.

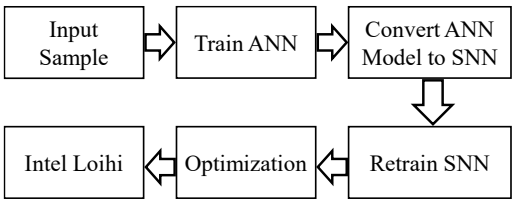

Fig. 4. Workflow of the proposed anomaly detection.

In this experiment, we implemented four models. The models are toe-to-toe, fully connected networks. Model-1 consists of six hidden layers as 128→64→16→16→64→128. The dimensions of hidden layers in Model-2 and Model-3 are 128→64→64→128 and 128→128 respectively. Model-4 is a non-spiking Artificial Neural Network (ANN) model that has the same architecture as Model-1.

The ANN training was performed using Keras in conventional backpropagation algorithms. We used 50,000 samples as a training dataset and 6,000 samples as a testing dataset. The attack-free normal dataset is used for training purposes, whereas the DoS, fuzzy, and gear data are used for testing. We used the optimizer RMSprop, loss of mean squared error, and learning rate of 0.0001 in the optimization process. The ReLU activation function is used in all layers of the AE network. To convert the Keras-based network into a spiking neural network, we used the NengoDL library. NengoDL captures information from the autoencoder (AE) during the conversion process. The NengoDL model is then transformed into an SNN-Autoencoder (SNN-AE) form, where adjustments are made to the weights and biases for adaptation. For mapping the SNN network on the Loihi chip, the ReLU neurons needed to be substituted with spike rate encoded Loihi-spiking ReLU neurons.

For SNN optimization, different firing rates were examined. For lower firing rates, the activities of Loihi neurons closely resemble those of ReLU and LIF neurons. However, at higher firing rates, there is a significant difference between ReLU and LIF neurons [32]. Nengo uses a synaptic value as a low pass filter for noise reduction and smoothing spikes in the output. We used a synaptic value of 0.05 in this experiment. We retrained the SNN-AE to map the network into the Loihi simulator. Finally, the retrained SNN-AE is ready to run on the Loihi chip using the Nengo Loihi library.

As mentioned earlier, our anomaly detection system is developed using the Nengo framework. We modified our application to run on the Loihi neuromorphic processor. Nengo can run applications on both CPU and neuromorphic hardware. If the Nengo framework doesn't detect the targeted hardware, it will simulate the entire process on the CPU. This simulation will provide the exact result if the application is running on neuromorphic hardware.

We utilized an estimation tool developed by Nengo for energy consumption evaluation. In addition to estimating Loihi's energy consumption, this tool can provide data for commercially available CPUs and GPUs for estimating energy for certain operations [33].

VII. Results and Discussion

A. Performance Analysis

We implemented four models (Model-1, Model-2, Model-3 as spiking, and Model-4 as non-spiking) for the anomaly detection experiment. The trained ANN model is converted to SNN using NengoDL. In the SNNs, the firing rates and synaptic values are varied to get optimal values for anomaly detection systems. Fig. 5 shows the impact of firing rates on the anomaly detection performance of the SNN-AE model. The F-1 score for the network model-2 is presented as an example in Fig. 5 for various firing rates. We see that for a firing rate of 50, the system shows the best F-1 score for different types of anomalies. Different synaptic values were also examined for the different models. Fig. 6 represents the F-1 score as the performance of the model-2 network for different synaptic values. The SNN-AE model exhibited the best performance on the F-1 score for Fuzzy, Gear, and DoS, respectively, with scores of 0.98, 0.78, and 0.78 once the synaptic value and neuron firing rate were set to 0.05 and 50 respectively. The rest of the experiments were carried out with a firing rate of 50 and a synaptic value of 0.05. Table II presents the Precision (P), Recall (R), and F-score (F-1) of various network models. All the models perform best for anomaly detection on fuzzy data. Among all the models, model-2 shows the optimal performance on precision, recall, and F-1 score for fuzzy, gear and DoS anomalies. The performance of the anomaly detection system changes if the number of hidden layers changes.

The sequence data was used to train all four network models regardless of spiking and non-spiking form. Using a sequence pattern in training improved the performance of SNN-AE significantly to achieve a similar performance as obtained in non-spiking model-4. Fig. 7 shows the F-1 score of all the models. From Fig. 7, we can see that model-2 shows a balanced F-1 score at a firing rate of 50.

Fig. 5. Comparison of the F-1 score according to neuron firing rates for model-2.

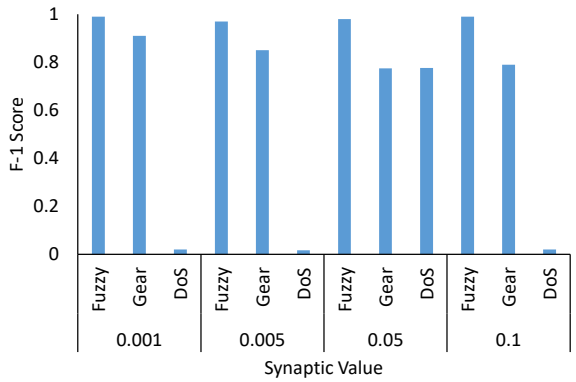

Fig. 6. Comparison of the F-1 score according to synaptic values for model-2.

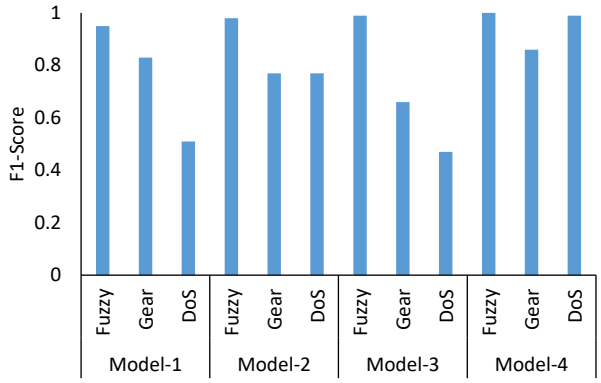

Fig. 7. Comparison of the F-1 score between the models for firing rate =50, and synaptic value =0.05.

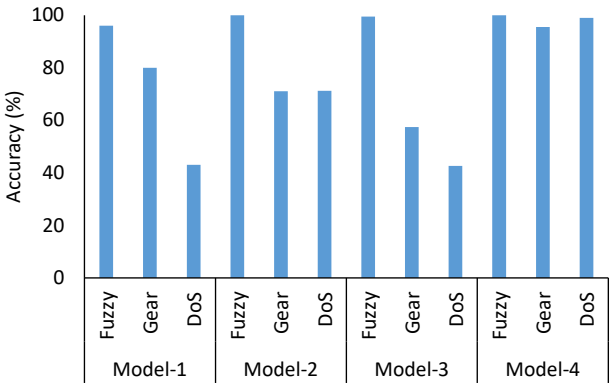

Fig. 8. Accuracy of various anomaly detection in different network models for firing rate 50, and synaptic value.

Table II: Precision, Recall and F-1 score of different anomaly detection models for firing rate 50 and synaptic value of 0.05.

		Fuzzy	Gear	DoS
Model-1	Precision	1.00	1.00	0.61
	Recall	0.91	0.71	0.45
	F-1	0.95	0.83	0.51
Model-2	Precision	1.00	1.00	1.00
	Recall	0.96	0.63	0.63
	F-1	0.98	0.78	0.78
Model-3	Precision	1.00	0.83	0.52
	Recall	0.99	0.54	0.43
	F-1	0.99	0.66	0.47
Model-4	Precision	1.00	1.00	1.00
	Recall	1.00	0.76	0.99
	F-1	1.00	0.86	0.99

Fig. 8 shows the accuracy of the anomaly detection system among the models. Model-4 shows anomaly detection accuracy in a non-spiking autoencoder. In the SNN-AE on Loihi, the detection accuracy for a fuzzy sample is 100%, equivalent to the non-spiking performance. However, for gear and DoS, the accuracy decreased in SNN-AE. Model-2 exhibited a balanced accuracy for various types of anomalies on the Loihi platform. The anomaly detection accuracy in model-2 is 100, 71.05, and 71.2%, respectively for fuzzy, gear, and DoS.

Our proposed anomaly detection systems outperformed the network shown in [14]. In [14], the authors achieved 0.82 and 0.64 F-1 scores for fuzzy and DoS scenarios, whereas in our system, the F-1 score was 0.98 and 0.78, respectively.

B. Energy Analysis in Anomaly Detection Operation

Table III shows the energy consumption per inference without the overhead (energy required to move data to and from the device) for all the models of the anomaly detection system on CPU, GPU, and Loihi. Nengo's power measurement library, 'Keras Spiking,' was used to measure the required lower bound energy without the overhead. From Table III, we can see that for all network models, the energy consumption for Loihi is around 0.17 µJ, which is about 20,000× and 700× less than the CPU and GPU respectively defined in the Nengo tools. The energy consumption among model-1, model-2, and model-3 changes slightly. The network in Model-4 is implemented in non-spiking. Thus the Loihi energy and estimation is not applicable here.

Table III: Energy consumption (μJ) per inference among all models.

	CPU	GPU	Loihi
Model_1	3670	128	0.18
Model_2	3620	127	0.17
Model_3	3600	127	0.17
Model_4 (Non-spiking)	2000	100	----

VIII. CONCLUSION

This is the first-ever implementation of an anomaly detection system on Loihi for in-vehicle applications. Using the Nengo framework, we convert the neural network to a spiking neural network and implement our system on the Loihi, which ensures low energy consumption. This work presented an autoencoder-based anomaly detection system on the CAN bus dataset. The results that the proposed anomaly detection system can detect anomalies successfully. The energy consumption in per anomaly detection operation on the Loihi is estimated at about 0.17 μJ, which is 20,000 and 700 times less than CPU and GPU processors respectively. Although the experiment was performed for vehicle anomaly detection, the same system can effectively be used in industrial and commercial anomaly and fraud detection. In future work, we plan to use online learning algorithms for on-chip training on the Loihi hardware as there is growing demand for online and on-chip learning for edge applications in industry and defense applications to achieve better performance and be more robust.

ACKNOWLEDGEMENT

This paper has been approved for public release by AFRL Case Number: AFRL-2024-1728.

REFERENCES

[1] Liu, F.T., Ting, K.M. and Zhou, Z.H., 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(1), pp.1-39.

[2] Chitrakar, R. and Chuanhe, H., 2012, November. Anomaly detection using Support Vector Machine classification with k-Medoids clustering. In *2012 Third Asian himalayas international conference on internet* (pp. 1-5). IEEE.

[3] Kumari, R., Singh, M.K., Jha, R. and Singh, N.K., 2016, March. Anomaly detection in network traffic using K-mean clustering. In *2016 3rd international conference on recent advances in information technology (RAIT)* (pp. 387-393). IEEE.

[4] Chalapathy, R. and Chawla, S., 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

[5] Kalutarage, H.K., Al-Kadri, M.O., Cheah, M. and Madzudzo, G., 2019, October. Context-aware anomaly detector for monitoring cyber attacks on automotive CAN bus. In *Proceedings of the 3rd ACM Computer Science in Cars Symposium* (pp. 1-8).

[6] Zhou, A., Li, Z. and Shen, Y., 2019. Anomaly detection of CAN bus messages using a deep neural network for autonomous vehicles. *Applied Sciences*, *9*(15), p.3174.

[7] C. Derse, M. el Baghdadi, O. Hegazy, U. Sensoz, H. N. Gezer and M. Nil, "An Anomaly Detection Study on Automotive Sensor Data Time Series for Vehicle Applications," 2021 Sixteenth International Conference on Ecological Vehicles and Renewable Energies (EVER), Monte-Carlo, Monaco, 2021, pp. 1-5, doi: 10.1109/EVER52347.2021.9456629.

[8] Blouw, P., Choo, X., Hunsberger, E. and Eliasmith, C., 2019, March. Benchmarking keyword spotting efficiency on neuromorphic hardware. In Proceedings of the 7th annual neuro-inspired computational elements workshop (pp. 1-8).

[9] Nengo Online Link: https://www.nengo.ai/keras-spiking/examples/model-energy.html

[10] Hossain, M.D., Inoue, H., Ochiai, H., Fall, D. and Kadobayashi, Y., 2020. LSTM-based intrusion detection system for in-vehicle can bus communications. *IEEE Access*, *8*, pp.185489-185502.

[11] Seo, E., Song, H.M. and Kim, H.K., 2018, August. GIDS: GAN based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)* (pp. 1-6). IEEE.

[12] Song, H.M., Woo, J. and Kim, H.K., 2020. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, *21*, p.100198.

[13] Kwak, B.I., Han, M.L. and Kim, H.K., 2021. Cosine similarity based anomaly detection methodology for the CAN bus. *Expert Systems with Applications*, *166*, p.114066.

[14] Jaoudi, Y., Yakopcic, C. and Taha, T., 2020, October. Conversion of an unsupervised anomaly detection system to spiking neural network for car hacking identification. In *2020 11th International Green and Sustainable Computing Workshops (IGSC)* (pp. 1-4). IEEE.

[15] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T. and Gehler, P., 2022. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14318-14328).

[16] Vanini, P., Rossi, S., Zvizdic, E. and Domenig, T., 2023. Online payment fraud: from anomaly detection to risk management. *Financial Innovation*, *9*(1), pp.1-25.

[17] Alfardus, A. and Rawat, D.B., 2021, December. Intrusion detection system for CAN bus in-vehicle network based on machine learning algorithms. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0944-0949). IEEE.

[18] Zanatta, L., Barchi, F., Burrello, A., Bartolini, A., Brunelli, D. and Acquaviva, A., 2021, June. Damage detection in structural health monitoring with spiking neural networks. In *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4. 0&IoT)* (pp. 105-110). IEEE.

[19] Dennler, N., Haessig, G., Cartiglia, M. and Indiveri, G., 2021, June. Online detection of vibration anomalies using balanced spiking neural networks. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 1-4). IEEE.

[20] Bauer, F.C., Muir, D.R. and Indiveri, G., 2019. Real-time ultra-low power ECG anomaly detection using an event-driven neuromorphic processor. *IEEE transactions on biomedical circuits and systems*, *13*(6), pp.1575-1582.

[21] Chen, Q., Qiu, Q., Li, H. and Wu, Q., 2013, November. A neuromorphic architecture for anomaly detection in autonomous large-area traffic monitoring. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 202-205). IEEE.

[22] Ussa, A., Rajen, C.S., Pulluri, T., Singla, D., Acharya, J., Chuanrong, G.F., Basu, A. and Ramesh, B., 2023. A hybrid neuromorphic object tracking and classification framework for real-time systems. *IEEE Transactions on Neural Networks and Learning Systems.*

[23] Zheng, S., Qian, L., Li, P., He, C., Qin, X. and Li, X., 2022. An introductory review of spiking neural network and artificial neural network: From biological intelligence to artificial intelligence. *arXiv preprint arXiv:2204.07519.*

[24] Dutta, S., Schafer, C., Gomez, J., Ni, K., Joshi, S., & Datta, S. (2020). Supervised Learning in All FeFET-Based Spiking Neural Network: Opportunities and Challenges. *Frontiers in Neuroscience*, *14*. https://doi.org/10.3389/fnins.2020.00634.

[25] Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S. and Liao, Y., 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, *38*(1), pp.82-99.

[26] Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G.A.F., Joshi, P., Plank, P. and Risbud, S.R., 2021. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, *109*(5), pp.911-934.

[27] Cook, J. A., and J. S. Freudenberg. "Controller area network (CAN)." *EECS* 461 (2007): 1-5.

[28] HPL, S.C., 2002. Introduction to the controller area network (CAN). *Application Report SLOA101*, pp.1-17.

[29] Song, H.M., Woo, J. and Kim, H.K., 2020. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, *21*, p.100198.

[30] Nengo Online Link: https://www.nengo.ai/nengo-dl/simulator

[31] Nengo Online Link: https://www.nengo.ai/nengo-loihi/

[32] Nengo Online Link: https://www.nengo.ai/nengo-loihi/v1.0.0/examples/keras-to-loihi.html

[33] B. Degnan, B. Marr and J. Hasler, "Assessing Trends in Performance per Watt for Signal Processing Applications," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 1, pp. 58-66, Jan. 2016, doi: 10.1109/TVLSI.2015.2392942.