Anomaly-Based Intrusion Detection in IIoT Networks Using Transformer Models

Jorge Casajús-Setién
Computational Intelligence Group
Universidad Politécnica de Madrid
jcasajus@titaniumindustrialsecurity.com

Concha Bielza
IEEE Senior Member
Computational Intelligence Group
Universidad Politécnica de Madrid
mcbielza@fi.upm.es

Pedro Larrañaga
IEEE Senior Member
Computational Intelligence Group
Universidad Politécnica de Madrid
pedro.larranaga@fi.upm.es

2023 IEEE International Conference on Cyber Security and Resilience (CSR) | 979-8-3503-1170-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSR57506.2023.10224965

***Abstract*—With the increase of device connectivity in Industry 4.0, securing industrial networks to defend them against cyberattacks has become a primary concern. Motivated by the huge data generated by devices in industrial environments, artificial intelligence has emerged as a promising complement to traditional cybersecurity. In order to gain insight about the possibility of cyberattacks, we propose a novel methodology to analyze industrial network traffic in real time exploiting the sequence modelling capabilities of the transformer architecture, widely used by the GPT model family for sequential language generation. We demonstrate that our method provides state-of-the art performance with promising explainability potential.**

I. Introduction

Factory digitalization arrived in the mid 20th century with the introduction of programmable logic controllers (PLCs) into machinery, signaling the starting point for the third industrial revolution, following the pioneer use of steam power in the first one and the raise of oil, gas and electric power in the second one. Less than a hundred years later, we are living a fourth industrial revolution towards what is commonly referred to as Industry 4.0. By employing smart technologies, Industry 4.0 reached an unprecedented level of automation and efficiency in production [1].

The Internet of Things, hereon referred to as IoT, can be defined as a group of infrastructures interconnecting devices, and allowing their management and the access to data they generate [2]. The connected devices are not only computers, but also elements that are not usually considered to be such thing, namely actuators and sensors. Every connected object in an IoT scenario represents a node in a virtual network, within which it is capable to automatically (with minimal human intervention) communicate its state to other devices, continuously transmitting large volumes of data [3].

In the industrial setting, IoT turns into IIoT (Industrial IoT), aiming to optimize production by enabling real-time, autonomous exchange of information within the industrial environment, playing a central role in the development of Industry 4.0 [4]. IIoT allows machinery to intercommunicate and interoperate, earning factories the "smart" tag by empowering intelligent automation. Even though this shift in the industry paradigm has ultimately entailed enhanced productivity at reduced costs, it has also contributed to increase the attack surface of industrial systems and the amount of cyber-threats that these are exposed to [5]. Our goal is to mitigate this problem by introducing network monitorization techniques based on Artificial Intelligence (AI), as traditional security does not fit in IoT environments [6]. IIoT networks generate massive amounts of sensor and instruction data, making AI's excellent data-analysis competences a perfect match.

In order to provide a robust defense against unseen and evolving threats, network monitorization systems must steer towards an approach based on anomaly detection (AD). Moreover, as network traffic data is inherently sequential, it is a natural choice to turn to sequential AI models for this task. The transformer [7] is a sequence transduction model, i.e., it allows reasoning from a set of observed train sequences to a new set of test sequences. As the transformer is low-cost on training and highly parallelizable, it has become the *de facto* standard in Natural Language Processing (NLP) [8]. Additionally, the transformer model offers inherent interpretability properties [9], allowing for explaining algorithms to provide insight about how a certain result was achieved. These explanations are highly valuable, since they guarantee that the reasoning of the model is guided by truthful principles [10]. In cybersecurity, explanations can help decide the appropriate response to a detected attack by tracing back the relevance of each predictive variable in a result from the model.

We propose a new anomaly-based Network Intrusion Detection System (NIDS) using a transformer model. Prior to our work there were very few studies on the application of this architecture to cybersecurity. In 2021, Wang and Li [11] introduced for the first time a hybrid model combining a transformer and a convolutional neural network to detect distributed attacks. Marino *et al.* [12] also conducted a research on the application of transformer models in the cybersecurity context, using an intermediate graph model for interpretability purposes. The rest of the paper is structured as follows: Section II introduces the transformer model and the attention mechanism. Our methodology is provided in Section III, while the conducted experiments and their results are presented in Section IV. Conclusions are drawn in Section V.

II. The transformer-based model

This section is divided into two parts. The first one develops the concept of attention and the second one explains how the

original transformer model presented in [7] is built.

A. The attention mechanism and self-attention

An attention function is, by definition, just a weighted averaged of values [13]. In its most general form, an attention function maps a query, q, a set of keys, K, and its corresponding values, V, to an output.

The output of an attention function is built upon a similarity score between the query and each possible key, which defines the weights for the averaged sum of the values:

$$\text{Attention}(q, K, V) = \sum_i \text{Score}(q, k_i) v_i$$

Essentially, the attention mechanism captures how much a query relates to each key in a database, and then it scales the values according to these relations. In [7], the proposed attention function is the "Scaled Dot-Product Attention", for which the dot product is used as the similarity measure between the query and the keys. For this mechanism, both the query and the keys are vectors of the same dimension d_k, while each value is represented by a vector of dimension d_v. If we let Q be a matrix packing several queries, the Scaled Dot-Product Attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

As a means to improve the representational capabilities of the attention mechanism, the queries, keys and value matrices are put through a linear transformation before evaluating the attention function. Likewise, a linear transformation is applied to the output of the attention function, yielding the final result. In practice, these linear projections are implemented as linear layers of a neural network, so its weights are learnt during the training process by gradient backpropagation. Finally, in order to provide information from a variety of representation subspaces, the queries, keys and value matrices are linearly transformed h times in h parallel attention heads, instead of only once, with the output of the h different attention functions being concatenated before the final linear transformation. Subfigure 1a depicts the described multihead attention mechanism.

The term self-attention refers to the application of an attention mechanism to an input sequence $\mathbf{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{R}^{d_x}$ acting as the queries, keys and values at once [14]. A self-attention transformation of a sequence returns a new representation of it for which every element has been weighted by its relation to each other position in the sequence. Thus, self-attention mechanisms enable the capture of sequential dependencies without employing recurrent architectures.

B. The transformer structure

The transformer is a sequence transduction model comprised of two parts: an encoder and a decoder. The encoder maps a sequence of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ to a same length sequence of lower-dimensional representations $\mathbf{z} = (z_1, \ldots, z_n)$. The decoder uses $\mathbf{z}$ as input, mapping it to an output sequence of arbitrary length: $\mathbf{y} = (y_1, \ldots, y_m)$. In [7],

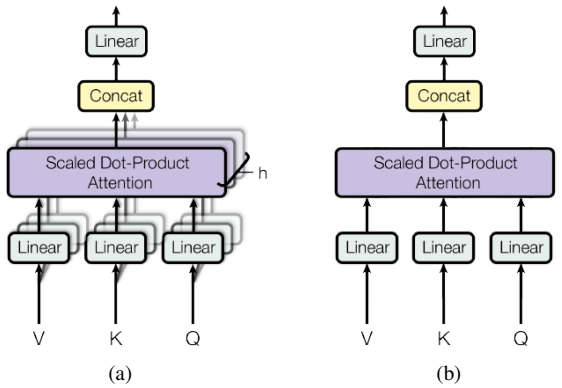


Fig. 1. Attention mechanisms. Subfigure 1a, taken from [7], is the attention mechanism presented in the same work, while Subfigure 1b has been modified to depicts the instance of it used in our model, with only one attention head.

both the encoder and the decoder are built as stacks of identical layers. The encoding layers are composed by:

- A self-attention step, using the input sequence as the query, the set of keys and the set of values.
- A feed forward step, applying a linear transformation upon its inputs, followed by an activation function.

On the other hand, each one of the decoding layers performs, sequentially:

- A masked self-attention step, using the target sequence as the query, the set of keys and the set of values. In order to avoid leakage of information from yet unseen positions of the target sequence during training, this self-attention step is said to be masked: each position of the sequence can only attend to the previous ones, and the forbidden attention values are set to $-\infty$.
- An attention step for which the queries and the keys are taken from the encoder output, while the input values come from the output of the masked self-attention step.
- A feed forward step, applying a linear transformation upon the output of the encoder-decoder attention layer, followed by an activation function.

The transformer is built upon a concatenation of N encoding layers, forming an encoder block, and N subsequent decoding layers, forming a decoder block. The encoder receives a sequence as an input, while the decoder receives a shifted version of the input sequence that is missing the last w positions. The decoder output is transformed by a final neural linear layer to provide a reconstruction of the original sequence from its shifted version and the encoding information.

A simplified scheme of the transformer model based on the one presented in [7] can be found in Figure 2. Note that:

- The full encoder and decoder blocks are built from N stacked up encondig or decoding layers, respectively.
- After each step in each encoding or decoding layer, residual addition and batch normalization is applied. This is, the output of every step is post-processed as:

$$\text{Sublayer}(x) = \text{Norm}(\text{Sublayer}(x) + x),$$

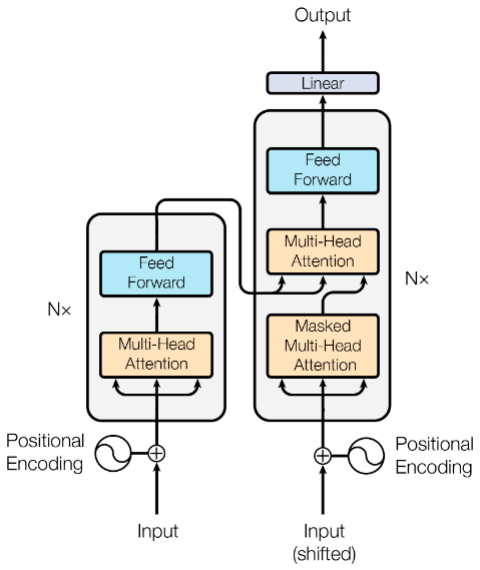


Fig. 2. Simplified scheme of the transformer, based on the diagram presented in [7]. The input order for the attention functions is inherited from Figure 1.

where x is the input of the step and $\mathrm{Sublayer}(x)$ represents the applied function.

- Both the encoder and the decoder inputs are positionally encoded, this is, each element of the sequence is summed to a certain vector that codifies information about its relative position within the sequence. This operation yields a new representation of the inputs that equips the model with a sense of order [15].

III. Methodology for sequential network AD

A. Network traffic data

Raw network traffic data in the form of IP packets can be captured in order to feed AI models. For this work, however, we used IP flow exporting to process packet data. Flow exporting groups packets exchanged in continuous time windows of traffic between the same source and destination IP and ports and through the same protocol. Each flow is described by a series of aggregated statistics derived from the packets it collects [16].

Our model will train and perform AD over recordings of flows in a given network that are sequentially ordered according to their ending time. Additionally, each flow will undergo a normalization process by standardization before being ran through the model.

B. Network AD using the transformer model

To the best of our knowledge, Marino *et al.* [12] approached for the first time the problem of network intrusion detection using transformers. Following their work, we also used a sliding-window approach to network monitoring: incoming flows are grouped in windows of a fixed length l_w to be inputted as sequences to the model.

Given a window $W = \{x_1, \ldots, x_{l_\mathrm{w}}\}$ of grouped flows, which constitutes the input of the encoder, a shifted version of it, made by removing the last l_s flows, is fed to the decoder. The goal, thus, will be for the decoder to reconstruct the original window, of length l_w, from its shifted version, of length $l_\mathrm{w} - l_\mathrm{s}$. Mathematically, this is achieved by means of setting a Mean Squared Error (MSE) loss function to be optimized by gradient backpropagation. The MSE loss for a given window can be expressed as:

$$\mathrm{Loss(W)} = \|W - \mathrm{D}(\mathrm{E}(W), W_{\mathrm{shifted}}))\|,$$

where $\|.\|$ stands for the Euclidean module, $\mathrm{E}(W)$ represents the output computed by the encoder for window W and $\mathrm{D}(\mathrm{E}(W), W_{\mathrm{shifted}})$ denotes the output of the decoder for the shifted window and the consequent encoder output.

AD is performed by measuring the loss function for new flow sequence samples. A given flow x_k is assigned an anomaly score corresponding to the loss function computed over the window $W_x = (x_{k-l_\mathrm{w}}, \ldots, x_k)$, this is:

$$\mathrm{Score(x)} = \|W_x - \mathrm{D}(\mathrm{E}(W_x), W_{x,\mathrm{shifted}}))\|,$$

where $W_{x,\mathrm{shifted}}$ is the shifted version of W_x, comprising the flows x_{k-l_w} to $x_{k-(l_\mathrm{w}-l_\mathrm{s})}$.

When the transformer model is trained following the optimization of this loss function, it can be used to capture the normal behaviour of a network by learning to predict benign sequences of flows traversing the network one flow at a time. Attack detection is based in the idea that flows containing an attack should represent a statistical anomaly when compared to normal traffic, so that the reconstruction of the flow sequence will be poor when presented with an attack, thus yielding a high loss value upon evaluation.

C. Configuration of the transformer model

The training and detection processes of the proposed system can be adjusted to the particularities of each network and its traffic data by varying the following hyperparameters:

- The number of encoding and decoding layers in the encoder and decoder blocks, respectively, N_{layers}.
- The length of the flow sequence window considered as an instance, l_w. This is the encoder input size.
- The length of the shift for the decoder input, l_s, which stands for the number of flows that the decoder will have to predict for a given window of length $l_w - l_s$.
- The number of attention heads, h.
- The number of training iterations, or epochs, N_{epochs}.
- The number of instances being fed to the model at the same time to exploit its parallelization features. This is referred to as the batch size of the training, b_s.

In practice, we've fixed l_s to one unit, but have experimented with different window lengths ranging from 25 to 100 flows, and exceptionally some tests with even longer windows. Our implementation of the transformer model uses only one attention head, since the number of attributes describing each flow is much smaller than that of data from the NLP field, for which the transformer was originally developed. Hence,

splitting attention into several heads makes no difference to the AD power of the system.

IV. Experiments and analysis of the results

To demonstrate the cyberattack detection capabilities of the proposed transformer-based model, a series of experiments have been conducted, baring in mind the double objective of empirically proving that our transformer model provides state-of-the art AD performance as a NIDS while comparing it against other well-established AD methods in the same industrial cybersecurity context.

The code for the implementation of the model and the experiments has been made publicly available, and can be found on GitHub in the following repository: https://github.com/jogecodes/transformerAD.

A. The WUSTL-IIoT-2021 dataset

As a platform to deploy our tests upon, we selected the WUSTL-IIoT-2021 dataset [17] for being a representative sample of a real IIoT network [18]. This dataset contains 1.194.464 pre-aggregated flows, each one described by 41 features, all of which are integer or float-point variables.

The importance of the features of the WUSTL-IIoT dataset (from now on we will ommit the 2021 label) has been previously tested, yielding the result that all of them are relevant enough as to be used in both training and AD.

For the experiment results to be consistent, we have split the WUSLT dataset in four approximate quarter parts of nearly 300.000 flows each. The first one, consisting exclusively of normal flows, will be used as training data. The following parts contain, 78.305 Denial of Service (DoS) attacks, 8.240 reconnaissance attacks, and, lastly, 259 command injection and 212 backdoor attacks. These types of attack have been selected to be representative of the industrial cyber-threat paradigm [17]. Reconnaissance attacks are aimed to gather control system network information, map the network architecture, and identify the characteristics of the devices in the network. DoS attacks against industrial systems attempt to stop the proper functioning of some portion of the cyber-physical system or to disable its totality. Command injection attacks insert false commands into a control system, overwriting machinery logic, code, or even remote terminal register settings [19]. Finally, backdoor attacks exploit vulnerabilities in the architecture perimeter that are forgotten, unnoticed, or disregarded, and allow to gain access to the system [20].

B. Performance report

The ratio of discovered attacks in the incoming traffic is, undoubtedly, the most important measure to assess the quality of a NIDS. However, in practice, every particular industrial cybersecurity scenario has a set of specific needs regarding aspects such as the tolerance for false alarms or the availability of alternative resources to palliate for the lacks of a given security system. If we think of AI models as another link in the chain of cybersecurity, it becomes clear that the optimal performance measure should be flexible enough to account for tuning of the model in each specific industrial context.

The Receiver-Operator Characteristic curve [21], or ROC curve, is a graphical representation of how the performance of a binary classifier responds to a variation of the threshold that splits the testing samples in its two possible classes according to a given score. By convention, the class assigned to the samples scored above the threshold is denoted as "positive" and those lying below the threshold are labelled "negative".

The ROC curve uses thresholds ranging from the lowest-given score from all testing samples to the highest one. In the lower end, all samples will be classified as positive, thus yielding a True Positive Rate (TPR, the ratio of correctly classified positive examples) of 1 and a False Positive Rate (FPR, the ratio of incorrectly classified negative examples) of 1 as well. When using the highest possible threshold, all of the samples will be labelled negative, so that both the TPR and the FPR will be 0. The ROC curve lies then on the FPR$-$TPR plane, and a random binary classifier would be represented as the diagonal from the origin of the plane to the $(1, 1)$ point.

The Area Underneath the ROC Curve (AUC) can be used as a performance measure for a binary classifier. However, this metric is not representative for the discrimination power of a model in a dataset with a highly imbalanced class ratio, as it is the case for the WUSTL-IIoT dataset. Real-life industrial networks will be subject to very little attack traffic compared with the traffic coming from normal behaviour.

To palliate this problem, we have computed an additional performance measure based on the precision-recall tradeoff. Precision (P) is defined as the of correctly classified positive examples to the total number of examples of examples labelled positive. Recall (R) is equivalent to the True Negative Rate (TNR, the ratio of correctly classified negative instances). The PR curve is formed by plotting on the $R - P$ plane the P and R values for a test set labelled according to incremental threshold values, ranging from the lowest given score to the highest one. For the PR curve, an unskilled classifier randomly assigning labels with equal probability would be represented by a horizontal line with P equal to the ratio of positive samples to the total number of samples. Derived from the PR analysis, the F-score is defined as the harmonic mean of P and R [22], $F = \frac{2PR}{P+R}$.

Both the ROC and the PR curves provide a visual interpretation of the performance of a classifier using different thresholds, leaving room for adjustment to a specific scenario.

C. Experimental results over the WUSLT-IIoT dataset

When deploying any AI model in any context, and in order to ensure proper fitting to the data, convergence during training must be studied; A stable learning process will have a steadily decreasing average loss throughout the training epochs.

In Figure 3, the evolution of the average training loss of various transformer models trained over the WUSTL-IIoT data using different combinations of the batch size and number of layers hyperparameters. It can be seen that convergence is always achieved, though using smaller batch sizes makes

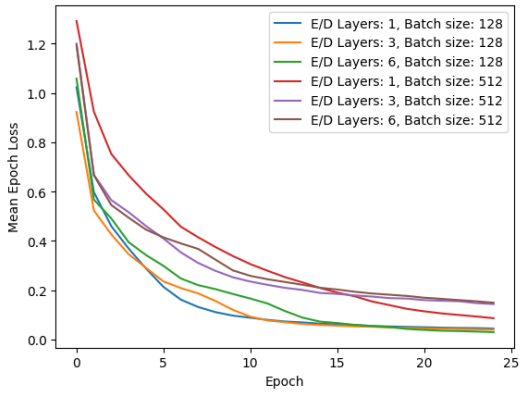


Fig. 3. Evolution of the mean training loss for six transformer models using different number of encoding/decoding layers and batch size. The experiments were carried out over 25 training epochs and a window length of $l_w = 50$.

TABLE I
MODEL SCORES ON THE WUSTL-IIOT DATASET DOS ATTACK DATA

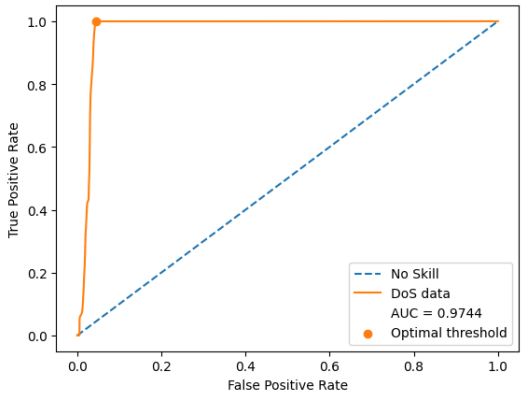
Measure	**Transformer**	**ECOD**	**DeepSVDD**	**RNN-AE**
AUC	**0.9744**	0.9637	0.9346	0.9309
F-score	**0.9431**	0.9251	0.9046	0.8104

convergence faster. As a downside, decreasing b_s reduces parallelization, thus increasing the time of computation. Increasing N_{layers} also has a negative impact over the computational time, since the number of trainable parameters is proportional to the number of layers in the encoding and decoding blocks. Although more parameters provide the model with more fitting power, as these tests were carried out using a window of $l_w = 50$, the models do not need more than one layer to properly adjust to the training data.

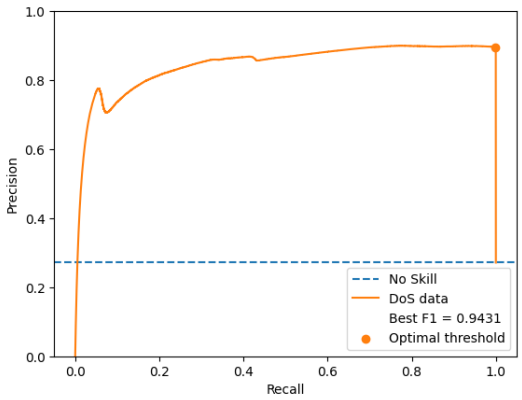
Figures 4 and 5 show the behaviour of the proposed transformer model over the different parts of the WUSTL-IIoT dataset. As the partial section of the WUSTL-IIoT dataset containing the DoS attacks is the only balanced one, both the ROC curve and the PR curve have been included in Figure 4. Our transformer model demonstrates very good anomaly discrimination power. Table I summarizes the performance of our system on this portion of the test dataset compared to three widely used alternative model-based AD methods:

- ECOD, a parameter-free, interpretable AD algorithm based on empirical cummulative distributions [23].
- DeepSVDD, a method based on training a neural network by minimizing the volume of the minimal hypersphere that encloses the network representations of the data [24].
- A RNN-based autoencoder (RRN-AE), which is another sequence transduction model for AD but powered by a recurrence mechanism, instead of an attention one [25].

The partition containing the reconnaissance attacks, as well as the one containing the command injection and backdoor threats, suffer from class imbalance, as they contain way less attacks than normal traffic. Consequently the ROC curve diagram has been ommitted for these data portions, and only the PR curves are shown in Figure 5. For the same reason, Table II



(a) ROC curve for the transformer model over the DoS data.



(b) PR curve for the transformer model over the DoS data.

Fig. 4. ROC and PR curves for an example transformer model over the first part of the WUSTL-IIoT dataset, representative of the DoS type of attacks.

TABLE II
MODEL SCORES ON THE WUSTL-IIOT RECONNAISSANCE (R) AND COMMAND/BACKDOOR (C/B) PARTITIONS

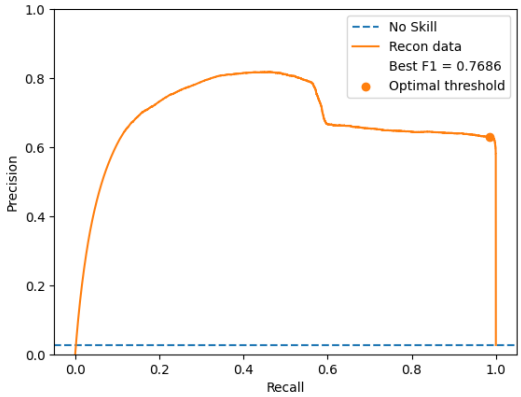
Measure	**Transformer**	**ECOD**	**DeepSVDD**	**RNN-AE**
F-score (R)	0.7686	0.7077	**0.7861**	0.419659
F-score (C/B)	**0.8909**	0.6810	0.8799	0.075891

only collects the F-score metric for the Reconnaissance and Command/Backdoor partition.

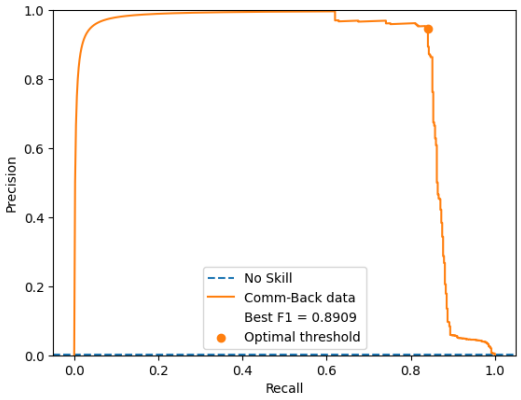
Our proposed transformer model provides the best detection results for the DoS attacks in terms of both the F-score and the AUC metrics. It also provides the best optimal F-score. It also obtains the higher F-score over the command injection and backdoor attacks, only being beaten by the DeepSVDD model on the reconnaissance attacks partition of the dataset.

V. CONCLUSION AND FUTURE RESEARCH

In this paper, we proposed an industrial NIDS framework based on AD using a transformer model, which proved to be a highly functional solution to the problem. Prior to the present study, the potential of this sort of models in this application had only been explored comprehensibly in [12]. The fundamental contributions we made are related to the simplification of the multi-head attention mechanism, including



(a) PR curve for the transformer model over the reconnaissance attack data.



(b) PR curve for the transformer model over the command injection and backdoor attack data.

Fig. 5. PR curves for an instance of the transformer AD model over the second and third part of the WUSTL-IIoT dataset, containing the reconnaisance and the command injection and backdoor attacks, respectively.

a detailed explanation of its role in sequential network flow analysis, and the application of the transformer model to the IIoT environment and its associated cyberattacks, elaborating further on the current state of the art of transformers in cybersecurity. Still, this is an ongoing work, and we have deemed the following to be the most interesting lines of research for its continuation:

- The development of a distributed AD system that also acts upon sequences of flows extracted from the traffic between two nodes of the network.
- Exploiting the intepretability capabilites of the transformer model to provide explanations of its results [9].

ACKNOWLEDGMENTS

This project was developed in collaboration with Titanium Industrial Security S.L. as part of the SLISE project (MIG-20211019), and has been partially supported by the Spanish Ministry of Science and Innovation (PID2019-109247GBI00, TED2021-131310B-l00 and RTC2019-006871-7 projects).

REFERENCES

[1] N. Ali, Z. Mohamed Isa, S. Abu Bakar, F. Ahmad Jali, and S. Shaharruddin, "Industrial revolution 4.0: Opportunities and challenges in online business," in *Proceedings*, vol. 82, no. 1. MDPI, 2022, pp. 85–93.

[2] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien, "Internet of things: A definition taxonomy," in *9th International Conference on Next Generation Mobile Applications, Services and Technologies*, 2015, pp. 72–77.

[3] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[4] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," *Journal of Manufacturing Systems*, vol. 47, pp. 93–106, 2018.

[5] A. Corallo, M. Lazoi, M. Lezzi, and A. Luperto, "Cybersecurity awareness in the context of the industrial internet of things: A systematic literature review," *Computers in Industry*, vol. 137, pp. 1–16, 01 2022.

[6] R. J. Raimundo and A. T. Rosário, "Cybersecurity in the internet of things in industrial management," *Applied Sciences*, vol. 12, no. 3, 2022.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[8] L. Tunstall, L. Von Werra, and T. Wolf, *Natural Language Processing with Transformers*. O'Reilly Media, Inc., 2022.

[9] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," *CoRR*, vol. abs/2012.09838, 2020.

[10] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[11] H. Wang and W. Li, "DDosTC: A transformer-based network attack detection hybrid mechanism in SDN," *Sensors*, vol. 21, pp. 5047–5061, 2021.

[12] D. L. Marino, C. S. Wickramasinghe, C. Rieger, and M. Manic, "Self-supervised and interpretable anomaly detection using network transformers," *CoRR*, vol. abs/2202.12997, 2022.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[14] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *CoRR*, vol. abs/1803.02155, 2018.

[15] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International conference on machine learning*. PMLR, 2017, pp. 1243–1252.

[16] P. Olivier and N. Benameur, "Flow level ip traffic characterization," in *Teletraffic Engineering in the Internet Era*, 2001, vol. 4, pp. 25–36.

[17] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research," 2021.

[18] M. Zolanvari, M. Teixeira, L. Gupta, K. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of Industrial IoT," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[19] T. Morris and W. Gao, "Industrial control system cyber attacks," in *Proceedings of the first International Symposium on ICS and SCADA Cyber Security Research*, 2013, pp. 22–29.

[20] D. Kuipers and M. Fabro, *Control Systems Cyber Security: Defense in Depth Strategies*. Idaho National Laboratory (INL), 2006.

[21] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proceedings of the Sixth International Workshop on Machine Learning*, 1989, p. 160–163.

[22] N. Chinchor, "MUC-4 evaluation metrics," in *Proceedings of the fourth conference on message understanding*, 1992, pp. 22–29.

[23] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen, "ECOD: Unsupervised outlier detection using empirical cumulative distribution functions," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–13, 2022.

[24] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 4393–4402.

[25] L. Bontemps, V. L. Cao, J. McDermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Third International Conference on Future Data and Security Engineering*, 2016, pp. 141–152.