INL/CON-22-67651-Revision-1

Machine Learning Models for Network Traffic Classification in Programmable Logic

January 2023

Brendan Glen Jacobson, Denver Shane Conger, Bryton John Petersen, Matthew William Anderson, Matthew R Sgambati

INL/CON-22-67651-Revision-1

Machine Learning Models for Network Traffic Classification in Programmable Logic

Brendan Glen Jacobson, Denver Shane Conger, Bryton John Petersen, Matthew William Anderson, Matthew R Sgambati

January 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Machine Learning Models for Network Traffic Classification in Programmable Logic

Brendan Jacobson*†, Denver Conger*†, Bryton Petersen*†,
Matthew Anderson*, Matthew Sgambati*
* Idaho National Laboratory
Idaho Falls, ID
† Brigham Young University Idaho
Rexburg, ID

Abstract—**Network traffic classification via machine learning on network packet payloads has emerged as an active area of research for network security due to the high accuracy machine learning models have achieved in classifying payloads. For effective deployment as part of network security, these machine learning models must not only classify malicious packet payloads accurately, they must also identify anomalous payloads and perform inference at speeds generally faster than 10,000 packets per second to be effective. This work explores the inference speeds and accuracy of several neural network models implemented in programmable logic on various field programmable gate arrays (FPGA), including the Xilinx VC1902 and Xilinx Zynq Ultrascale+. This work also presents the design and performance of both an autoencoder and variational autoencoder programmed on the FPGA for identifying anomalous packet payloads. The performance benefits of the FPGA implementation for this type of packet payload inspection driven by machine learning are compared against graphics processing unit (GPU) inference implementations run on two state-of-the-art datacenter GPU devices, the NVIDIA V100 and A100. The model accuracy difference between the FPGA and GPU implementations was 4% or less while the Xilinx VC1902 outperformed both the NVIDIA V100 and A100 for inference speeds on all the models explored except the variational autoencoder.**

I. Introduction

Network traffic classification via machine learning on network packet payloads has emerged as an active area of research for network security [7, 13, 12, 16, 9, 5, 15, 11, 8]. Machine learning models are already able to both classify packet payloads with high accuracy as well as identify anomalous payloads to protect against zero-day attacks. For effective deployment as part of network security, these machine learning models must not only classify packets accurately, they must also correctly identify anomalous packets and perform inference at speeds generally faster than 10,000 packets per second as driven by 5G latency standards. However, of the various state-of-the-art methods for packet classification recently reviewed in [5], classification performance per packet was nearly a factor of 20 or more slower than this for each of the 11 methods examined. To explore how deep learning models for network payload perform in programmable logic, this work explores packet payload classification and anomaly detection using several types of machine learning models implemented on field programmable gate arrays (FPGA), including a convolutional neural network (CNN), a residual neural network (ResNet), and the autoencoder architecture. Each of these have been successfully applied to network packet payload classification before: ResNet in [7], CNN in [7, 15], and autoencoder in [11]. In addition to these, this work describes and implements a variational autoencoder [6] for anomaly detection in network packet payloads and deploys this architecture to FPGA. The performance and accuracy for each is examined using publicly available datasets, including NDSec-1 [3] and CTU-13 [4], with FPGA results compared against those using an advanced graphics processing unit (GPU) for inference. The accuracy difference between FPGA and GPU implementations for all models is 4% or less while the Xilinx VC1902 outperformed the state-of-the-art NVIDIA A100 GPU in all cases except the variational autoencoder and easily exceeded the 10,000 packet per second threshold in all models except the variational autoencoder.

This work is structured as follows: in Section II, the prior art for packet payload classification and anomaly detection is reviewed. In Section III, the FPGAs, GPUs, the raw packet datasets, and the network structure for the machine learning models are described. In Section IV, the results including performance, recall, precision, and F1 score for packet classification and anomaly detection are reported. In Section V, we present our conclusions and identify research questions left for future research.

II. Related Work

Deep learning approaches have been successfully applied in both packet classification and anomaly detection. Supervised learning approaches for packet classification have leveraged CNNs [16, 15, 7, 8], ResNet [7], and recurrent neural networks [8, 15] while unsupervised approaches have leveraged autoencoders [11]. Other approaches to anomaly detection have included $n-$gram approaches [14] and word2vec [10] applied to packet payloads [5]. While performance was evaluated for each of those approaches, none exceeded the 10,000 packet per second threshold, and the evaluated models were not implemented in programmable logic in any of these examples.

This work leverages some of the best deep learning approaches evaluated in these prior efforts for both classification (CNN, ResNet) and anomaly detection (autoencoder) and evaluates these models on state-of-the-art datacenter GPUs for comparison with state-of-the-art FPGA implementations. For comparison purposes, two of the same datasets used in [5] have been used for the model evaluations in this work both on GPUs and FPGAs.

III. Methods

A. Experimental setup and dataset

The NDSec-1 [3] dataset was collected in 2016 by the University of Applied Sciences Fulda (Germany). The raw NDSec-1 data was created by collecting normal traffic packets and simulated attack packets. The flows relative to each of the packets were classified in a separate dataset with three different labels. The primary label specified the type of attack as either Web Attack, BotNet, Bruteforce, DoS, Exploit, Malware, Misc, Probe, or Normal, if the flow did not contain attack traffic. The secondary label contained additional info regarding the flow, in the case of Normal traffic, or additional info regarding the type of attack, in the case of attack traffic. The tertiary label contained comments and additional attributes regarding each flow. For our purposes, the secondary and tertiary labels were dropped. From NDSec-1, a new representation was created in which individual packets were labeled according to their flows. Because of severe imbalances in the data where some attack labels contained as few as 208 packets and others contained over 300,000, attacks were omitted if their packets numbered less than 1,000. This left us with Web Attack, BotNet, Bruteforce, DoS, Exploit, Malware, Misc, Probe and Normal packets to be trained and tested.

The CTU-13 dataset was created in 2011 by CTU University (Czech Republic). The data were created by capturing a large amount of botnet traffic mixed with normal and background traffic in thirteen different simulated scenarios. The packets were labeled by the scenario in which they were collected. Because of security concerns, the normal and background traffic was omitted in the public release of the CTU-13 dataset. To compensate for this, the CTU-13 dataset was augmented with normal traffic as part of this study. This consisted of web browsing traffic, video streams, downloads, and get-requests to websites.

For both the modified NDSec-1 and CTU-13 datasets, packets with empty payloads were omitted. Training and testing was based purely on the payload of each packet. For standardization purposes, payloads were either buffered or truncated at 400 bytes. The hex was converted to bytes and then stored as a numpy array with the shape [400, 1, 1].

The FPGAs used in the testing were the Xilinx Zynq Ultrascale+ FPGA on the ZCU104 evaluation board and the Xilinx VC1902 FPGA on the VCK190 evaluation board. The GPUs were the NVIDIA V100 with 32 GB of memory and A100 with 40 GB of memory. The experimental setup is summarized in Figure 1. For each model explored, a standardized training and testing set was used based on a 70/30 split. Inference performance tests were run 10 times, the fastest and slowest numbers were eliminated, and the average of the remaining eight runs was reported.

B. Network structure

Four different supervised learning model structures were made, tuned, and tested on the VCK190 and ZCU104 FPGA evaluation boards and the A100 and V100 GPUs. The four models were a basic CNN (Figure 2) referred to here as CNN-1, a deeper CNN referred to here as CNN-4, a basic ResNet model referred to here as ResNet-1, and a deeper ResNet model (Figure 3) referred to here as ResNet-3. The specifics of each model are detailed below. After making the basic structure of each model, several different hyperparameters were tested. The search space of these hyperparameters is summarized in Table I. Not all hyperparameters are used in each model; for example, the basic CNN has no dropout layers, so variations of dropout rate were not tested for that case. Table II summarizes the results of the optimal hyperparameter search.

Hyperparameters	Search Space
Activation function	Sigmoid, Swish, Leaky ReLU, ReLU, ELU, SELU, tanh
Dropout rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
Learning rate	0.0001, 0.001, 0.01, 0.1, 0.02, 0.3, 1
Batch size	1, 10, 50, 100, 256, 500
Number of filters	8, 16, 32, 64, 128
Kernel size	1, 3, 5, 7
Optimizer	Adam, RMSprop, Stochastic gradient descent

TABLE I: Hyperparameter search space for the supervised learning models used for network packet classification.

	CNN-1	CNN-4	ResNet-1	ResNet-4
Dropout rate	N/A	0.3	0.03	0.6
Learning rate	0.01	0.001	0.0001	0.0001
Batch size	100	100	50	50
Number of filters	128	128	64	64
Kernel size	7	3	2	2
Optimizer	RMSprop	Adam	Adam	Adam
Resnet blocks	N/A	N/A	1	3
Activation	ELU	ReLU	ReLU	RelU

TABLE II: Optimal hyperparameter values for the supervised learning models used for network packet classification.

1) Autoencoder: Both an autoencoder and variational autoencoder were implemented on the programmable logic of the FPGA due to the ability of the autoencoder architecture to detect anomalous network packets and potential zero-day attacks. Both the autoencoder and variational autoencoder were trained on just normal network packet payloads collected from web browsing traffic, video streams, downloads, and get-requests to websites as described in Section III-A. The models follow general autoencoder architectures of bottle-necking and encoding the data down to a small latent space, which is then decoded to produce a generated packet representative of and similar to the original.

While autoencoders and variational autoencoders are often used for generating new data, in this work, they are used to

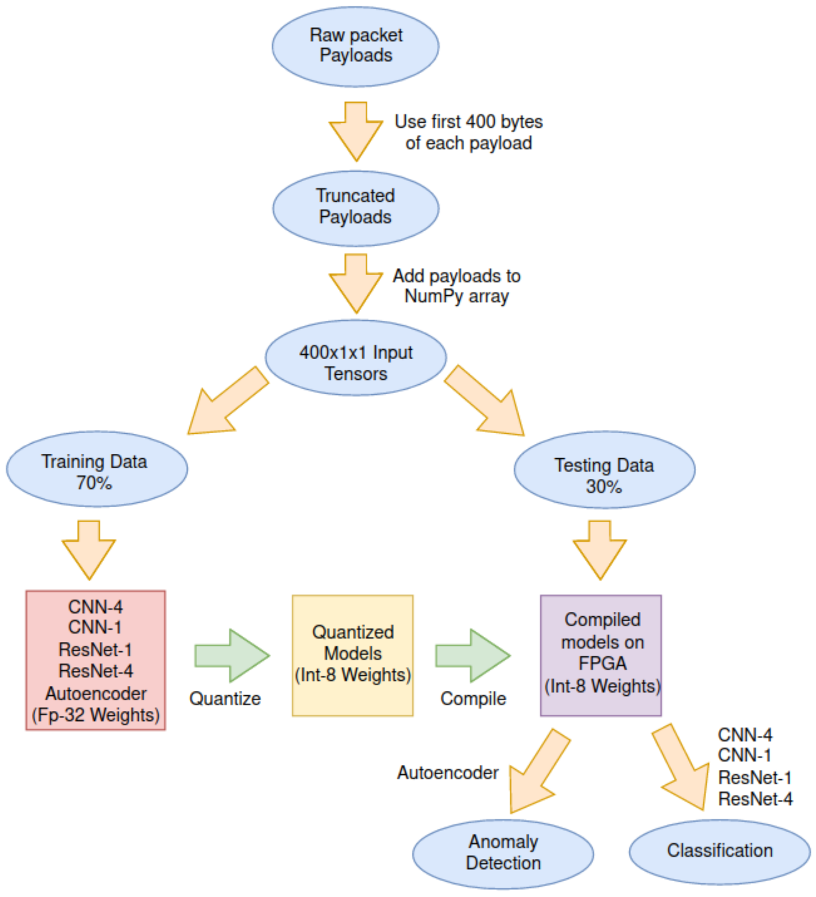
Raw packet Payloads
Use first 400 bytes of each payload
Truncated Payloads
Add payloads to NumPy array
400x1x1 Input Tensors
Training Data 70%
Testing Data 30%
CNN-4 CNN-1 ResNet-1 ResNet-4 Autoencoder (Fp-32 Weights)
Quantize
Quantized Models (Int-8 Weights)
Compile
Compiled models on FPGA (Int-8 Weights)
Autoencoder
CNN-4 CNN-1 ResNet-1 ResNet-4
Anomaly Detection
Classification

Fig. 1: An overview of the experimental setup is shown here. The packet payloads in the datasets were divided into a 70/30 training/testing split. Model training took place on the GPU. After training, each model was quantized using the Vitis toolset [1] and compiled for the FPGAs where accuracy and performance metrics were evaluated.

detect anomalous network packet payloads. These generative models were trained on normal network packet payloads in order to create an encoded space that does not know how to generate an attack payload in any meaningful way. When inferring, the reconstruction loss will be higher for an anomalous network packet payload, thereby allowing for identification. With a baseline loss comparison of normal and now attack payloads, a threshold value can manually be created that represents if the model has been trained on similar data before. The autoencoder and variational autoencoder architecture used in this work are summarized in Figures 4 and 5, respectively.

2) CNN-1: The basic CNN consisted of just four hidden layers. The first layer was the same reshape layer included in each model. Then, a single 2D convolutional layer is used. The other two layers were a max-pooling layer and a global-average pooling layer. In the end, the model had a total of 7,174 trainable parameters. In Table 1, we can see that the ELU activation function was the optimal choice of activation functions for this model. However, it should be noted that the ReLU function was used in the end. This is due to the fact that ELU is not supported by the Vitis-AI [1] compiler. This model was primarily created as a way of better understanding how reducing the number of layers might affect the speed and accuracy of a model.

3) CNN-4: The normal CNN had a total of 12 hidden layers. There were four convolutional layers, paired with max-pooling layers and dropout layers of 0.5. This model contained 56,949 trainable parameters.

4) Resnet-1: The basic Resnet model was created by iterating through a ResNet block consisting of five convolutional layers, two residual addition layers, two ReLU activation functions, and finally a MaxPooling2D layer. After iterating through the ResNet block, the model is made up of one convolutional layer, one GlobalAveragePooling layer, and two dense layers followed by two dropout layers of 0.5. The model ended with a final dense layer of size two and a Softmax activation function in order to provide a binary probability. In total, the model was made up of 97,410 trainable parameters.

5) Resnet-3: The ResNet-3 model was designed exactly the same as the ResNet-1 model but it iterated through the ResNet block three times instead of one. After iterating through the ResNet block, the model followed the same pattern as the ResNet-1 model. The ResNet-3 model contained 236,130 trainable parameters in total.

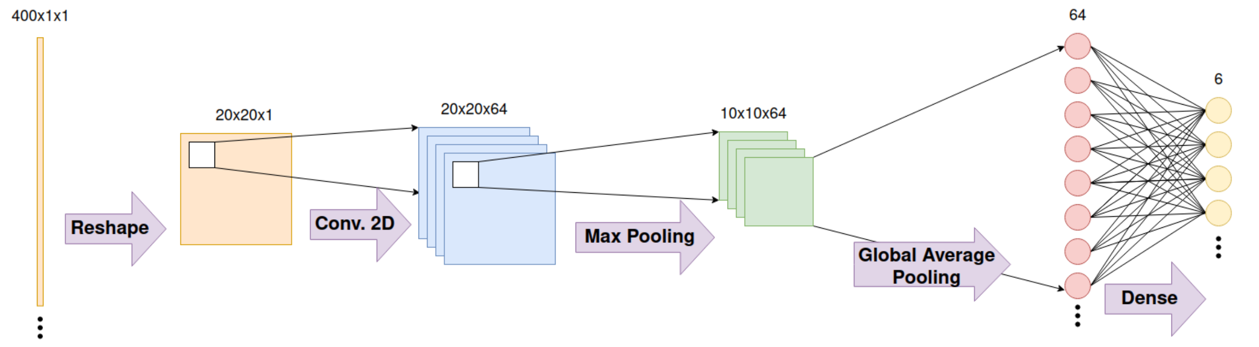

Fig. 2: A summary of the CNN-1 model used for network packet payload classification.

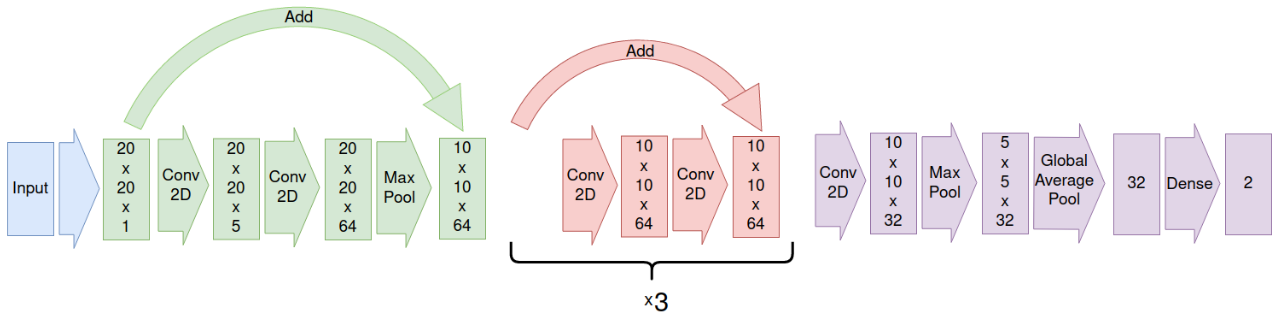

Fig. 3: A summary of the ResNet-3 model used for network packet payload classification.

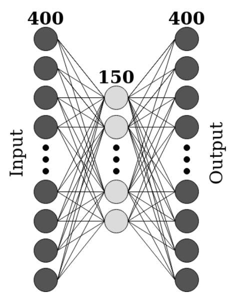

Fig. 4: Summary of the autoencoder model with a simple three-layer encoder-decoder.

IV. Results

The performance and accuracy results for the four supervised learning models and two unsupervised learning models explored on both GPU and FPGA are presented here. The VCK190 evaluation board hosting the VC1902 FPGA outperformed the best GPU (NVIDIA A100) in all cases except for the variational autoencoder. The accuracy difference between the FPGA and GPU implementations was less than 4%. The variational autoencoder FPGA implementation required a hybrid approach where some of the computations were executed on the CPU rather than the FPGA, resulting in the slower performance. The performance comparison between the FPGA implementations and the GPU implementations is shown in Table III. In these experiments, even the small Zynq Ultrascale+ FPGA hosted on the ZCU104 evaluation board nearly met 10,000 packets per second for all models except the variational autoencoder.

The F1 score, accuracy, precision, and recall for the supervised learning model cases for the NDSec-1 dataset are shown in Table IV. The accuracy lost by moving the model to the FPGA implementation was less than 4% in all cases and remained above the 90% threshold. While fine tuning the FPGA implementation can often reduce this accuracy loss, we note that no fine tuning of any of the models was done for the FPGA implementations in these results.

The CTU-13 dataset has been identified in previous work [5] as challenging for some packet payload classifier models. The F1 score, accuracy, precision, and recall of the supervised and unsupervised models in this work for the CTU-13 dataset are presented in Table VI. We found that ResNet-3 did the best on the GPU; the confusion matrix for that model on the CTU-13 dataset is shown in Table V to illustrate which scenarios were more likely to be misidentified.

Unlike the supervised learning models (CNN-1, CNN-4, ResNet-1, ResNet-3), the unsupervised models (autoencoder, variational autoencoder) were not trained on any attack packet payloads and were only trained on normal traffic. When these models were tested against the attacks in the CTU-13 dataset to see if they could distinguish an attack packet payload from a normal packet payload, the accuracy was not as high as the supervised learning models for both GPU and FPGA implementations as seen in Table VI. The variational autoencoder had an accuracy nearly 8% better than the autoencoder and shows promise for future investigations. Because the variational autoencoder has a combined mean squared error or reconstructed loss as well as the Kullback–Leibler divergence loss [2] used to map the deviations of the latent space, the

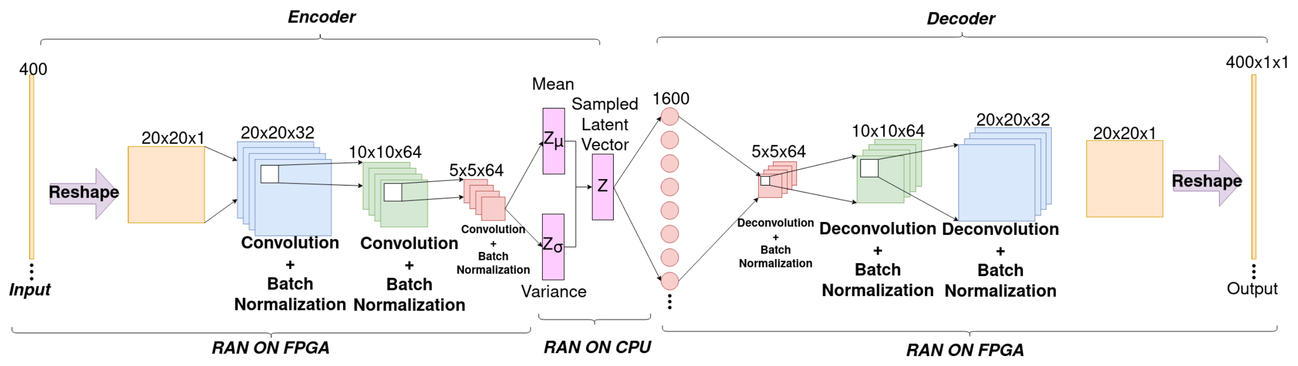
Encoder
Decoder
400
Reshape
20x20x1
20x20x32
10x10x64
5x5x64
Convolution + Batch Normalization
Convolution + Batch Normalization
Convolution + Batch Normalization
Input
Mean
Zμ
Sampled Latent Vector
Z
Zσ
Variance
1600
5x5x64
10x10x64
20x20x32
20x20x1
Reshape
400x1x1
Output
Deconvolution + Batch Normalization
Deconvolution + Batch Normalization
Deconvolution + Batch Normalization
RAN ON FPGA
RAN ON CPU
RAN ON FPGA

Fig. 5: Summary of the variational autoencoder model that utilizes a Kullback–Leibler divergence and reconstruction loss in the latent vector [2].

Model	FPGA ZCU104 Packets/Sec	FPGA VCK190 Packets/Sec	GPU V100 Packets/Sec	GPU A100 Packets/Sec
CNN-1	8.14k	25.8k	14.2k	23.7k
CNN-4	7.58k	24.9k	13.0k	22.1k
ResNet-1	8.02k	24.7k	12.7k	22.2k
ResNet-3	7.99k	26.3k	11.6k	20.8k
Autoencoder	7.81k	25.6k	14.4k	23.4k
Variational autoencoder	1.68k	4.14k	9.98k	23.6k

TABLE III: A comparison of the speed and accuracy of various machine-learning models run on GPU or FPGA.

		Resnet-3	Resnet-1	CNN-4	CNN-1
FPGA	F1	.921	.902	.931	.912
	Accuracy	.920	.902	.934	.912
	Precision	.931	.904	.954	.913
	Recall	.921	.901	.930	.911
GPU	F1	.939	.935	.954	.931
	Accuracy	.937	.935	.955	.931
	Precision	.953	.941	.955	.937
	Recall	.939	.917	.945	.930

TABLE IV: Accuracy, precision, recall, and F1 score for the supervised models explored here on both GPU and FPGA for the NDSec-1 dataset.

variational autoencoder provides a better distinction between trained normal payloads and anomalous payloads than what the basic autoencoder provides.

V. Conclusion

This work has explored the performance and accuracy of four supervised learning models and two unsupervised learning models for network packet payload classification and anomaly detection. Using two publicly available datasets, NDSec-1 and CTU-13, FPGA implementations of the machine learning models were compared against GPU implementations in terms of performance and accuracy. The performance of FPGA implementations exceeded the GPU implementations in all cases except for the variational autoencoder, which relied on some computations on the CPU to fully implement the variational autoencoder. The difference in accuracy between the FPGA and GPU implementations for all models was less than 4% without any fine tuning. These results suggest programmable logic implementations of machine learning models can achieve the speed and accuracy needed to reliably classify 10,000 packets per second or higher for both supervised and unsupervised learning modalities.

This work has also presented a variational autoencoder implemented for the FPGA. The variational autoencoder gave superior accuracy results over the autoencoder and suggests a clear path to detecting anomalous packet payloads associated with zero-day attacks. Future work will explore this variational autoencoder while also removing the few components that were computed on the CPU for a complete FPGA implementation.

VI. Acknowledgements

This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517. This manuscript has been authored by Battelle Energy Alliance, LLC under

Data Type	Normal	Scenario 1	Scenario 2	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Scenario 9	Scenario 12	Scenario 13
Normal	**405**	23	9	279	5	0	115	12	765	45	450
Scenario 1	66	**1324**	1	299	20	0	85	7	155	14	117
Scenario 2	32	8	**1919**	33	1	0	7	3	45	8	31
Scenario 4	157	53	18	**975**	18	0	166	44	315	35	329
Scenario 5	49	54	2	171	**1492**	0	71	10	133	8	131
Scenario 6	0	0	0	0	0	**2151**	0	0	0	1	0
Scenario 7	97	39	1	353	13	0	**1031**	46	194	15	246
Scenario 8	14	5	0	82	9	0	102	**892**	24	187	78
Scenario 9	338	18	2	256	0	0	117	11	**827**	34	415
Scenario 12	49	4	6	58	4	0	10	172	80	**1667**	67
Scenario 13	234	46	3	418	25	0	156	15	504	25	**676**

TABLE V: Confusion matrix for CTU-13 using the ResNet-3 model.

		ResNet-3	ResNet-1	CNN-4	CNN-1	Autoencoder	Variational Autoencoder
F	F1	.807	.822	.817	.771	0.579	0.675
P	Accuracy	.808	.820	.821	.770	0.617	0.694
G	Precision	.830	.836	.862	.774	0.685	0.736
A	Recall	.807	.807	.816	.771	0.617	0.694
G	F1	.832	.824	.817	.812	0.585	0.687
P	Accuracy	.835	.823	.824	.801	0.619	0.700
U	Precision	.854	.838	.870	.850	0.698	0.739
	Recall	.834	.823	.820	.808	0.619	0.700

TABLE VI: Accuracy, precision, recall, and F1 score for the CTU-13 dataset for the supervised and unsupervised models explored here on both FPGA and GPU.

Contract No. DE-AC07-05ID14517 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

References

[1] Vitis unified software platform, https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html. Accessed: 06/08/2022.

[2] Andrea Asperti and Matteo Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. *IEEE Access*, 8:199440–199448, 2020.

[3] Frank Beer, Tim Hofer, David Karimi, and Ulrich Bühler. A new attack composition for network security. In Paul Müller, Bernhard Neumair, Helmut Raiser, and Gabi Dreo Rodosek, editors, *10. DFN-Forum Kommunikationstechnologien*, pages 11–20, Bonn, 2017. Gesellschaft für Informatik e.V.

[4] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers Security*, 45:100–123, 2014.

[5] Mehedi Hassan, Md Enamul Haque, Mehmet Engin Tozal, Vijay Raghavan, and Rajeev Agrawal. Intrusion detection using payload embeddings. *IEEE Access*, 10:4015–4030, 2022.

[6] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[7] Hyun-Kyo Lim, Ju-Bong Kim, Joo-Seong Heo, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han. Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 046–051, 2019.

[8] Hongyu Liu, Bo Lang, Ming Liu, and Hanbing Yan. Cnn and rnn based payload classification methods for attack detection. *Knowledge-Based Systems*, 163:332–341, 2019.

[9] Manuel Lopez-Martin, Belén Carro, Antonio Sánchez-Esguevillas, and Jaime Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5:18042–18050, 2017.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[11] Baskoro Pratomo, Pete Burnap, and George Theodorakopoulos. Unsupervised approach for detecting low rate attacks on network traffic with autoencoder. pages 1–8, 06 2018.

[12] Ola Salman, Imad H. Elhajj, Ayman I. Kayssi, and Ali Chehab. A review on machine learning-based approaches for internet traffic classification. *Ann. des Télécommunications*, 75:673–710, 2020.

[13] Muhammad Shafiq, Xiangzhan Yu, Asif Ali Laghari, Lu Yao, Nabin Kumar Karn, and Foudil Abdessamia. Network traffic classification techniques and comparative analysis using machine learning algorithms. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 2451–2455, 2016.

[14] Mayank Swarnkar and Neminath Hubballi. Ocpad: One class naive bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, 64:330–339, 2016.

[15] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792–1806, 2018.

[16] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*, pages 712–717, 2017.