NetGlyphizer: Labeled Network Traffic Generation Using Representation Learning and Transformers

Gabin Noblet[†*], Cédric Lefebvre[*], Philippe Owezarski[†], William Ritchie[*]
[†]*LAAS - CNRS, Université de Toulouse, CNRS, Toulouse, France*
[*]*Custocy, Toulouse, France*
{gabin.noblet, philippe.owezarski}@laas.fr
{gnoblet, clefebvre, writchie}@custocy.com

***Abstract*—Network security has become increasingly critical due to the growing frequency and sophistication of cyber-attacks. To enhance intrusion detection capabilities, emerging solutions leverage machine learning (ML) models. However, a significant challenge persists in acquiring sufficient high-quality training data. To overcome this challenge, we explored the use of generative neural networks to create realistic and synthetic network traffic data. This paper introduces *NetGlyphizer*, a novel approach to learn a discrete representation of network traffic using Vector Quantized-Variational Autoencoders (VQ-VAE). This model transforms network flows into a sequence of discrete tokens, referred to as *NetGlyphs*. This method enables the use of Transformer models to generate new *NetGlyphs* sequences, which can be decoded into real network traffic. The efficacy of this approach is evaluated using a dataset comprising both benign and malicious traffic flows. In comparison to a method employing a continuous representation, our model exhibits superior performances in accurately reconstructing the data and preserving the original distribution. Additionally, conditional generation facilitates the generation of labeled network traffic flows based on the specific network traffic class. The results demonstrate that our approach effectively preserves protocol compliances and usages, making it a promising solution for labeled network traffic generation.**

***Index Terms*—Representation Learning, Autoencoder, Transformer, Generative Model, Network Security**

I. Introduction

The increasing volume and complexity of network traffic, along with the growing prevalence of sophisticated cyber threats, underscores the critical importance of robust network security in today's digital landscape. This has led to a rising interest in employing Machine Learning (ML) techniques for network traffic analysis and security enhancement. While ML models have shown significant potential in areas such as classification and behavioral analysis, their development is often hindered by the limited availability of realistic, up-to-date, and diverse datasets. Privacy concerns make data collection challenging, and generating such datasets requires significant time and resources, including the deployment of enterprise-level infrastructures, simulation of attacks, and data collection.

Previous research has explored generative models like Generative Adversarial Networks (GANs) and Variational AutoEncoders (VAEs) to simulate realistic network traffic data. Existing approaches vary in their data representations, some have focused on flow metadata and metrics [1], [2], which restricts their capacity to precisely replicate flow packets. Others have focused on the generation of realistic network packets [3], [4], without considering flow context.

In this paper, we suggest representing network traffic as a collection of bidirectional flows, where each flow is a series of packets. This approach aims to enhance the realism of generated traffic data and facilitate accurate labeling.

We introduce *NetGlyphizer*, a model inspired by Vector Quantized-Variational AutoEncoders (VQ-VAE) [5], designed to transform network traffic flow into sequences of discrete *NetGlyphs*, an analogy to how tokenizers transform text into tokens for neural language processing tasks. These *NetGlyphs* enable efficient generation and reconstruction of network traffic patterns using a Transformer Decoder-Only architecture.

Our contributions can be summarized as follows:

- Feature engineering of network traffic flows, where each packet is represented by a set of different features allowing reconstruction of traffic in packet capture format (*Pcap*).
- Introducing *NetGlyphizer*, an analogy to tokenizers for text processing, to transform network traffic into a sequence of discrete tokens called *NetGlyphs*.
- Using a state-of-the-art *Decoder-Only* Transformer architecture to perform *NetGlyphs* sequences generation that can be decoded back into real traffic flows.
- Using conditional generation techniques to generate labeled network traffic flows.

The results show that the *NetGlyphizer* model better represents network data compared to a baseline model using a continuous representation. We also demonstrate that the learned representation of network traffic can be used to generate realistic and labeled network traffic flows with a Transformer *Decoder-Only* architecture conditioned on network traffic class.

II. Methodology

A. Network Traffic Representation

The proposed network traffic representation is dual, exposing both packet-level features and flow-level context for adaptability across security systems.

1) Motivation: Representing flows via metrics only may lack adaptability. A granular representation that captures packet distribution within a flow allows reconstruction into a packet capture file (*Pcap*), ensuring compatibility with security systems.

2) Flow Identification: Packets in a flow share a common *5-tuple* (source/destination IPs, source/destination ports, and protocol). The bidirectional nature of the flow allows us to account for the relationship between packets traveling in both directions.

3) Packet Distribution Features: To identify the packet distribution in a flow, we focused on three features common to all protocols. The inter-arrival time (IAT), that defines the time spent between packets, the packet direction in the flow, and the payload size. Since encrypted payloads limit feature extraction, only payload size is considered. The payload in generated flows is considered encrypted and will be randomly generated.

4) TCP Features: This study focuses on TCP protocol. This protocol ensures reliable delivery through various mechanisms like flags and counters. In order to reduce the amount of features used in our representation, without altering the quality of data, we filtered out some features. TCP flags as combinations (*Syn*, *Syn/Ack*, *Fin*, ...) are included to capture connection states. Sequence and acknowledgment numbers are not considered as they can be derived from payload size and flags. Congestion control features are omitted as congestion relates to the network infrastructure and not the applications in the endpoints. Finally, we assume well-formed traffic with no malformed packets or protocol misuses.

Table I summarizes the packet features used in this study.

TABLE I
PACKET FEATURES FOR NETWORK TRAFFIC REPRESENTATION

	Type	Example
IAT [1]	*Continuous*	$1.389s$
Payload size	*Numeric*	388
Direction	*Binary*	0 (forward)
Flags	*Categorical*	PA (Psh/Ack)

[1] Inter-Arrival Time

B. NetGlyphizer: Learning a Discrete Representation

We use a two-stage process for network traffic generation: first, a representation learning model is trained on the data, then a generator creates new instances in the learned latent space. This section compares our discrete representation learning approach with a baseline that learns a continuous representation.

1) Recurrent AutoEncoder (RAE) architecture: An autoencoder uses an encoder to project data into a latent space and a decoder to reconstruct the data. We focus on Recurrent AutoEncoders (RAE), which use Recurrent Neural Networks (RNNs) to process sequential data. Specifically, Long Short-Term Memory (LSTM) layers are used in both the encoder and decoder to capture long- and short-term dependencies.

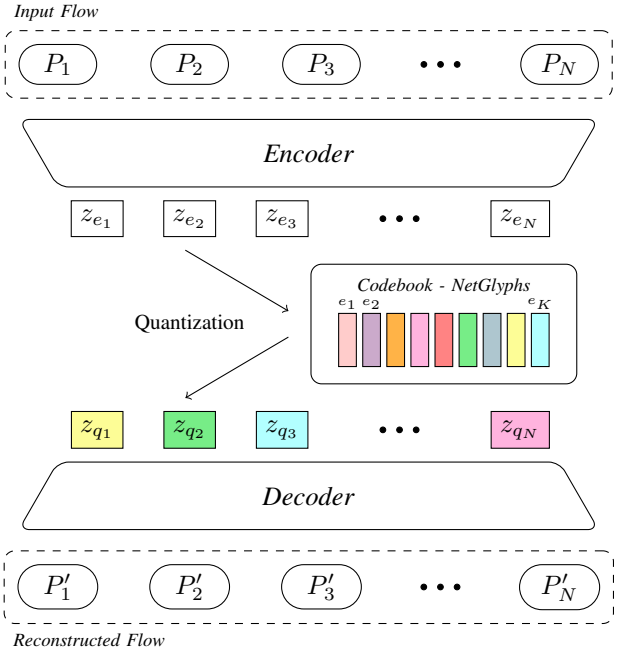

Fig. 1. *NetGlyphizer* model architecture. Input flows are cut up into individual packets, $F = \{P_1, P_2, \ldots, P_N\}$. The **Encoder** transforms this sequence of packets into a sequence of continuous vectors $\{z_{e_1}, z_{e_2}, \ldots, z_{e_N}\}$ corresponding to each packet. These vectors capture the dependencies between input packets. The **Quantization** layer then maps each continuous vector to the nearest vector from a codebook. Mapping continuous data to discrete data allows us to better generate flows that are by nature discrete. These discrete vectors are denoted as ***NetGlyphs***. The **Decoder** then reconstructs the input flow F' from the sequence of NetGlyphs using LSTM layers.

2) Baseline model: Following the approach of Shahid et al. [6], we use a Recurrent Autoencoder (RAE) to transform a sequence of packet features into a fixed-length latent vector. The encoder, composed of LSTM layers, processes the packet sequence and produces a final vector that encapsulates the sequence's context by aggregating information across all previous steps. The decoder then reconstructs the original packet sequence from this latent vector, performing a vector-to-sequence transformation using LSTM layers. However, since only the last output vector from the encoder is retained, information about the exact number of packets in the flow is lost.

3) NetGlyphizer: Although the inter-arrival time between packets is continuous, the majority of network features are composed of categorical and discrete features. Consequently, it is proposed that learning discrete data representations would be a more efficient approach than learning a continuous representation.

The proposed model is inspired by the Vector Quantized-Variational Autoencoder (VQ-VAE) model [5]. The encoder is modified from the baseline model to output a sequence of vectors rather than a single vector. The latent representation changes from a single vector to a sequence of vectors. Accordingly, the decoder is modified to be able to reconstruct a sequence of packets from a latent sequence. Furthermore, a

Vector Quantization (VQ) layer is added to discretize the latent sequence into a sequence of discrete vectors. These discrete vectors are referred to as *NetGlyphs*.

The model architecture is described in Fig. 1:

a) ***Encoder****:* For a flow F composed packets P_i, it produces a continuous latent sequence Z_e of corresponding vectors z_{e_i}. As we can see in Fig. 1,

b) ***Quantizer****:* Vector quantization is applied to discretize the latent sequence Z_e into sequence Z_q. Each vector z_{e_i} is mapped to the closest vector e_k in a codebook, s.t. $z_{e_i} \approx e_k$ as follows:

$$z_{q_i} = e_k, \quad \text{where} \quad k = \ argmin_j ||z_{e_i} - e_j|| \tag{1}$$

c) ***Decoder****:* The decoder reconstructs the flow F' as a sequence of packets P_i' from the quantized latent sequence Z_q.

The model is employed to transform a sequence of packets into a sequence of *NetGlyphs*. In accordance with this approach, the model is designated *NetGlyphizer*, a term derived from the tokenizer that transforms text into tokens.

Furthermore, the sequential latent representation enables the model to retain the sequence length information. This design addresses the issue of sequence length inconsistency present in the baseline model. However, this model architecture suffers from low codebook usage, i.e. it utilizes a limited number of *NetGlyphs* to represent network traffic. Such results are often due to their poor initialization. To tackle this issue, we implemented the improvement proposed by Yu et Al. [7]. They proposed to reduce the codebook vectors dimension and use a unit-normalization constraint.

C. Labeled Traffic Generation Using Transformers

We propose using conditional generative models to generate latent representations of network traffic flows related to a specific traffic class. Specifically, we apply a Conditional GAN (CGAN) to generate continuous latent representations, and a Transformer *Decoder-Only* model for generating sequences of *NetGlyphs* based on traffic classification.

1) Baseline for comparison: Conditional GANs (CGANs) [8] extend GANs by conditioning both the generator and discriminator on input labels. Shahid et al. [6] utilized Wasserstein GAN with Gradient Penalty (WGAN-GP) [9], which replaces traditional GAN loss with Wasserstein distance for improved stability. We propose to use a C-WGAN-GP that combines these approaches, conditioning generation on network traffic classes.

Training uses the continuous latent representations of network flows from the baseline encoder model, incorporating traffic class labels as one-hot vectors. During inference, the generator, conditioned on a traffic class, produces new latent representations, which are then decoded into network flows.

2) NetGlyphs Sequence Generation with Transformer: The *NetGlyphizer* encodes network traffic as sequences of learned *NetGlyphs*, analogous to tokenization in NLP. A Transformer model, conditioned on traffic class, generates new sequences by iteratively predicting the next *NetGlyph* index. The class is included as the first token, representing different traffic classes and 0 as the padding/end-of-sequence (EOS) token.

Training involves converting network traffic flows into sequences of *NetGlyphs*, prefixed with the class label. During inference, the Transformer generates sequences using only the class token as input, predicting subsequent tokens iteratively until the EOS token is predicted. The generated sequences are then decoded back into network flows using the *NetGlyphizer*.

III. Evaluation and Results

This section presents and discusses the results of the study on labeled network flow generation using representation learning and conditional generative models.

A. Dataset

The dataset used is sampled from the datasets provided by Stratosphere Laboratory at Czech Technical University in Prague [10]. We selected four classes of traffic. Three different malware traffic sources: *Emotet*, *Dridex*, and *Trickbot*. And benign captures of HTTPS traffic.

B. Network flow representation learning

The representation learning models are evaluated using various methods suited to each feature type (Table II). Reconstructions from the baseline model and *NetGlyphizer* are compared to the original distribution.

1) ***Earth Mover's Distance (EMD)****:* EMD is a distance used to measure distribution similarity. It is used to compare IAT and payload size sequences between original and reconstructed data. Due to large data volume we apply k-means clustering to approximate the EMD. The results are presented in Table II. Lower values indicate better reconstruction accuracy.

2) ***Error Rate****:* Binary and categorical feature reconstruction is evaluated by the error rate, computed as the fraction of incorrect reconstructions. A lower error rate indicates higher accuracy (Table II).

TABLE II
Representation learning evaluation on network features reconstruction

	Baseline	NetGlyphizer		
NetGlyphs Dim.	*N.A.*	4	3	2
IAT_{EMD}[1]	24.19	7.73	**6.41**	23.17
$Payload\ size_{EMD}$	1246.14	359.73	**226.88**	415.40
$Direction_{ErrorRate}$	1.72e−1	1.41e−6	**3.86e−7**	1.41e−6
$Flags_{ErrorRate}$	1.24e−1	1.61e−5	**1.17e−5**	3.41e−4

[1]Earth Mover's Distance

The results for all metrics presented in Table II show that the *NetGlyphizer* model with its discrete latent representation outperforms the previous state-of-the-art or baseline model that uses a continuous latent space. The accuracy on categorical features is higher for the *NetGlyphizer* than for the baseline model. The EM distance show that our model better preserves the original distribution of IATs and payload sizes.

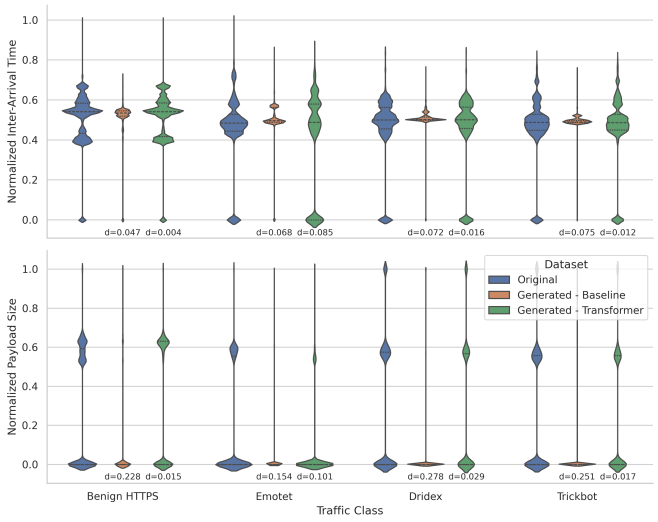

Fig. 2. Generated flows analysis across network traffic classes using violin plots. The violin plots show the density of IAT and payload size features across the traffic classes for the original dataset compared to the generated datasets. The baseline approach uses C-WGAN-GP to generate new continuous latent vectors that are decoded into packets using the baseline representation learning model. Our approach uses a Transformer *Decoder-Only* architecture to generate new sequences of *NetGlyphs* and decode them back to packets with the *NetGlyphizer* model. The Earth Mover's distance *d*, is computed between the analyzed and original distributions. The generation is conditioned on the network traffic class. Our approach shows better results for the considered features and traffic classes, except for the IAT of *Emotet* malware traffic.

C. Labeled network flow generation

The evaluation of generated traffic compares generated and original flow distributions.

1) ***Violin plots****:* They are used to visualize density differences across network traffic classes, and Earth Mover's Distance (EMD) quantifies similarity (Fig.2).

2) ***Protocol verification****:* Generated flows are checked for TCP compliance by analyzing handshake sequences and session termination. Ratios of correct/erroneous handshakes and connection terminations are compared (Table III).

TABLE III
PROTOCOL USAGE COMPARISON: ORIGINAL VS. GENERATED DATASETS

	Original	Baseline	*NetGlyphizer*
Connection established	**62.51%**	99.33%	**64.80%**
Connection refused	**28.42%**	0.11%	**26.97%**
Connection attempts	**8.88%**	0.37%	**8.03%**
Malformed 3-way handshakes	**0.19%**	**0.19%**	**0.20%**
Client reached *CLOSED* state	**88.38%**	27.18%	**89.00%**
Server reached *CLOSED* state	**84.12%**	26.01%	**84.99%**

3) ***Flow metrics****:* Flow metrics—packet count, duration, and byte transfer—are analyzed per traffic class. The EMD measures similarity between original and generated distributions, with lower values indicating better realism (Table IV).

The results show that our approach performs well in generation of network flows conditioned by the label. The violin plots (Fig. 2), along with the EM distances shows that our approach presents the best accuracy in most of the cases, except for the IAT of *Emotet* traffic type. Furthermore, an analysis of flow metrics showed that the generated data better respects the distribution of the original data (Table IV), except for benign class. Finally, the analysis of protocol shows that our approach outperforms the baseline models in protocol usages (Table III).

TABLE IV
COMPARISON OF FLOW METRICS DISTRIBUTION FOR GENERATED TRAFFIC

	Baseline C-WGAN-GP	*NetGlyphizer* Transformer	Class
Flow Length *EMD*[1]	**0.83**	0.90	Benign
	74.58	**0.08**	Emotet
	96.67	**0.20**	Dridex
	109.98	**0.24**	Trickbot
Flow Duration *EMD*	44.1	**7.48**	Benign
	15.56	**2.65**	Emotet
	15.11	**1.32**	Dridex
	31.57	**3.90**	Trickbot
Flow Bytes C2S *EMD*	**823.48**	4627.13	Benign
	163.01	**149.34**	Emotet
	997.95	**121.98**	Dridex
	754.36	**44.91**	Trickbot
Flow Bytes S2C *EMD*	10507.19	**4856.81**	Benign
	226.13	**62.08**	Emotet
	411.47	**112.28**	Dridex
	943.06	**102.68**	Trickbot

[1]Earth Mover's Distance

IV. CONCLUSION

We introduced *NetGlyphizer*, a model for learning discrete representations of network traffic using a VQ-VAE architecture adapted for sequential data. It converts packet sequences into discrete vectors, NetGlyphs, stored in a codebook. Our analysis shows *NetGlyphizer* outperforms a baseline continuous latent space model in capturing network data structure.

By pairing *NetGlyphizer* with a Transformer, we generate *NetGlyph* sequences that can be decoded back into packets, conditioned on traffic class. This approach better preserves feature distributions and generates data more similar to the original. Protocol compliance analysis confirms higher accuracy in capturing connection attempts, failures, and established flows. Additionally, generated traffic flows align better with real-world metrics like duration, packet count, and data volume. Our method also supports Pcap reconstruction from network traffic format for compatibility with security tools.

V. FUTURE WORKS

Future works should include more diverse data, additional protocols, and features along with other attack and benign scenarios. This work only focuses on the generation of network traffic flows, but additional work should be done on the context or attack scenario that links several individual flows together. Finally, the generation of payload content still remains a challenge and should be addressed further by taking into account unencrypted data or at least TLS handshakes or any other cryptographic protocols.

FUNDING

This work was supported by ICO, Institut Cybersécurité Occitanie, funded by Région Occitanie, France.

REFERENCES

[1] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using Generative Adversarial Networks," *Computers & Security*, vol. 82, pp. 156–172, May 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404818308393

[2] M. Rigaki and S. Garcia, "Bringing a GAN to a Knife-Fight: Adapting Malware Communication to Avoid Detection," in *2018 IEEE Security and Privacy Workshops (SPW)*. San Francisco, CA: IEEE, May 2018, pp. 70–75. [Online]. Available: https://ieeexplore.ieee.org/document/8424635/

[3] A. Cheng, "PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. Vancouver, BC, Canada: IEEE, Oct. 2019, pp. 0728–0734. [Online]. Available: https://ieeexplore.ieee.org/document/8936224/

[4] B. Dowoo, Y. Jung, and C. Choi, "PcapGAN: Packet Capture File Generator by Style-Based Generative Adversarial Networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Boca Raton, FL, USA: IEEE, Dec. 2019, pp. 1149–1154. [Online]. Available: https://ieeexplore.ieee.org/document/8999252/

[5] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," May 2018, arXiv:1711.00937 [cs]. [Online]. Available: http://arxiv.org/abs/1711.00937

[6] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, "Generative Deep Learning for Internet of Things Network Traffic Generation," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. Perth, WA, Australia: IEEE, Dec. 2020, pp. 70–79. [Online]. Available: https://ieeexplore.ieee.org/document/9320384/

[7] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu, "Vector-quantized Image Modeling with Improved VQGAN," Jun. 2022, arXiv:2110.04627 [cs]. [Online]. Available: http://arxiv.org/abs/2110.04627

[8] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 2014, arXiv:1411.1784 [cs]. [Online]. Available: http://arxiv.org/abs/1411.1784

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," Dec. 2017, arXiv:1704.00028 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1704.00028

[10] Stratosphere, "Stratosphere Laboratory Datasets," 2015. [Online]. Available: https://www.stratosphereips.org/datasets-overview