MFWDD: Model-based Feature Weight Drift Detection Showcased on TLS and QUIC Traffic

Lukáš Jančička[1], Dominik Soukup[1], Josef Koumar[1], Filip Němec[1], and Tomáš Čejka[2]

[1]Czech Technical University in Prague, Prague, Czech Republic
[2]CESNET, a.l.e., Prague, Czech Republic
{janciluk, soukudom, koumajos, nemecfil}@fit.cvut.cz, cejkat@cesnet.cz

***Abstract*—Machine learning (ML) represents an efficient and popular approach for network traffic classification. However, network traffic inspection is a challenging domain and trained models may degrade soon after deployment. Besides biases present during data captures and model creation, data drifts contribute significantly to ML model degradation. This paper proposes a novel method called Model-based Feature Weight Drift Detection (MFWDD) for concept drift detection. It is a part of a public software framework suited for dataset drift analysis tailored to the domain of network traffic. This work addresses TLS and QUIC service classification problems, examines a variety of experiments analyzing the evolution of the respective distributions, and observes their degradation over time on different ML features. The MFWDD framework guided TLS and QUIC services classification models retraining throughout an extensive period and not only prevented model degradation but also improved its performance and consistency over time.**

***Index Terms*—Dataset quality, Traffic classification, Computer network, Concept drift**

I. Introduction

Network traffic monitoring provides an essential insight for maintaining services and security, and network traffic classification is its integral part. The security features, such as the *Encrypted Server Name Indication (ESNI)* for domain names encryption, forces the development of new means of network monitoring and analysis. Researchers have thus been focusing on indirect network inspection, identification of metadata, investigation of packet series and their temporal statistical properties, and analysis that does not rely on decryption.

For example, [1]–[6] targets on temporal network characteristics in Sequence of Packet Lengths and Times (SPLT) method, Koumar et al. [7] suggested extended IP flow NetTiSA for network traffic classification and Plny et al. [8] present a detection method based on multiple weak indicators.

Research of network traffic classification drives the need to deploy ML classifiers and maintain them continually. In real-world environments, the collected data often tends to change over time, e.g. reflecting state of CDNs, subtle protocol modifications through time etc. This phenomenon is reffered as concept or data drift and ideally, network traffic inspection would maintain its accuracy irrespective of such externalities. A possible solution is the concept of Active Learning (AL), that updates underlying datasets. However, due to the dynamicity of investigated network services, precise tuning of AL parameters is notoriously demanding.

To address these challenges, it is crucial to understand the behavior and importance of selected features in time, and the relationships between different features. This paper presents a method for well-timed drift detection and AL control. ML-model retraining is a computationally expensive task and our goal is to eliminate as many unnecessary retraining steps as possible. Therefore, we propose novel drift detection method MFWDD for ML-models based on feature weights. The method can detect model obsoletion without class labels for real-world deployment scenarios. Morover, in supervised scenarios, the technique brings descriptive analysis of behavioral changes in input features. Furthermore, we published MFWDD as an analytical tool and also a more detailed investigation of datasets available at Github repository[1].

II. Related works

There are several existing methods to detect a concept drift. A common example is *error rate-based drift detection*. These algorithms look at the error rate of a particular classifier. If the change of error rates over time is statistically significant a drift alarm is raised. Another common method is *distribution-based drift detection*. In this case, algorithms use distance metrics to detect changes between the original and new training data. Korycki et al. [9] point out detection limits in network traffic environment and introduced a drift detector called RBM-IM. It is based on a Restricted Boltzmann Machine, and it is designed for imbalanced and multi-class streams, which are a common classification task in the field.

To address the sensitivity limits of dataset drift detection methods, weighting or sampling is used to adjust for the differences in the data distributions. Soukup et al. [10] introduced the idea of dataset quality evaluation, which is affected by the metadata used by the ML classifier. This work focuses on ML-assisted network traffic classification problems and continuous model maintenance compensating for concept drifts. We attempted to leverage feature importance as weights to calculate drift severities, which yielded promising results. Multiple

This research was funded by the Ministry of Interior of the Czech Republic, grant No. VJ02010024: Flow-Based Encrypted Traffic Analysis and also by the Grant Agency of the CTU in Prague, grant No. SGS20/210/OHK3/3T/18 funded by the MEYS of the Czech Republic.

[1]https://github.com/FETA-Project/MFWDD

978-3-903176-66-9 ©2024 IFIP

existing statistical tests are available to calculate drift severity, for example the Wasserstein test, Kolmogorov–Smirnov test, or Maximum Mean Discrepancy (MMD) two-sample test [11], [12]. Another statistical test that has proven to be instrumental in several fields is the Page-Hinkley test (PHT) originally introduced in [13] addressing production quality problems, and inspection schemes optimization. While investigating network intrusion detection systems, Andresini et al. [14] successfully used PHT as a drift estimator in the Str-MINDFUL algorithm. We follow up on the existing state of the art and evaluate main metrics within the networking domain with the proposed feature weights. The introduced software tool is configurable and allows to change the statistical test if needed.

In [15], statistical properties of the CESNET-TLS-Year22 dataset with respect to weekdays were analyzed. It was confirmed that the network traffic statistical properties differ significantly on working days and weekdays. This experience underlines how important the appropriate choice of the time range for network dataset collection.

Žliobaitė et al. [16] introduced the value of combination AL and data drift detection. This direction is also confirmed by [17] from a recent study. This research motivated us to base a new drift detection method on drift severities corresponding to a given feature collection.

III. Datasets

Since we aim to find out how the distribution of various features evolves in time for network traffic classification datasets, we chose the one-year-long CESNET-TLS-Year22 [18] and four-weeks-long CESNET-QUIC22 [19] datasets. Both datasets were captured on an Internet Service Provider (ISP) backbone network and considered more than a hundred network traffic classes. The size of the datasets permits us to flexibly arrange diverse experimental settings, perform an offline evaluation of drift behavior concerning various research topics, and test developed algorithms in detail.

Both datasets are available using the CESNET-DataZoo [20] library. The collection and detailed feature description are available in the paper by Luxemburk et al. [2] for TLS traffic and Luxemburk et al. [21] for QUIC traffic. Datazoo uses natively Per Packet Information (PPI) or (SPLT), packet histograms (PHISTS), and flow-based features, such as the number of packets in a flow or the flow end reason.

IV. MFWDD: Model-based Feature Weight Drift Detection Methodology

In this section, we present details of the proposed method, MFWDD, for feature weighted detection of concept drift. Data drift and concept drift [22], [23]. Data drift occurs when the statistical properties of the input data change, leading to a potential degradation in model performance because the model was trained on a different data distribution. Concept drift, on the other hand, refers to changes in the target variable's definition or the relationship between input features and the target variable. Both types of drift pose significant challenges for predictive models in dynamic environments requiring continuous monitoring and model maintenance ensuring their effectiveness and relevance.

To detect a distribution drift presence at a given time, the drift detector compares samples from the initial and current distributions. The detector performs statistical tests and evaluates whether pairs of samples originate from different distributions. Upon detecting drift and its measure, a trigger prompting to retrain the classification model is raised. The measure should quantify dissimilarity between the two distributions and reflect the performance impact of network traffic classification. It is often referred to as drift severity s.

There are multiple methods to integrate various feature distributions into the final decision regarding the presence of a drift. A popular library in this field is Evidently [12], which utilizes the two-sample Kolmogorov-Smirnov (KS) test or the Wasserstein distance. Based on the results in [12], the Wasserstein distance is recommended for datasets with more than a thousand data instances. Otherwise, the KS test gives better results. Our experiments confirm the same best practice. Although the KS test was more sensitive and often identified more features as drifted for datasets with more than $\approx$ a thousand samples, it provided more precise results for smaller data collections. For class (supervised) level detection, we used the KS test with a threshold of $\alpha = 5\%$, since the number of samples per class per day was below one thousand. In unsupervised scenarios, the normlaized Wasserstein distance with a threshold of $\alpha = 5\%$ was used. Both the KS test and normalized Wasserstein distance use a single variable vector and, when applied on the i-th feature they yield severity s_i.

During our investigation, we also considered the MMD test, which operates with multiple variables and theoretically can detect more complex dependencies. Despite experimenting with various kernel settings and sensitivity levels, MMD did not prooved beneficial for our classification tasks. Moreover, analyzing drifts for specific features would be more complex with MMD. Thus we decided to build our method on the Wasserstein distance and KS test.

Considering a classification task with n features and given a statistical measure, i.e. two-sample Kolmogorov-Smirnov (KS) test or the Wasserstein distance, we may calculate individual drift measures $s_1, ..., s_n$ for each feature separately and then investigate mappings: $S: \ s_1, ..., s_n \ \mapsto \ \mathbb{R}_+$ called *severity* or *drift severity*. A straightforward enhancement may incorporate feature weights as an additional measure for the drift severity evaluation. The mapping S may be written as: $S: s_1, ..., s_n, w_1, ..., w_n \mapsto \mathbb{R}_+$.

Current drift detection methods often neglect the significance of individual features in affected models, and they are sensitive to reporting false positives since they neglect the target dataset context and externalities. This is particularly challenging when feature vectors contain hundreds of features, some of which provide minimal human-readable information.

To address this issue, statistical tests are conducted independently for each feature distribution, and their drift severities are stored for further analysis, for example: drift rates, extraction of features exhibiting severe drift, features with recurring drifts

etc. The total drift strength S is calculated as a weighted arithmetic mean of individual severity measures s_i, where the feature importances supplied by the classifier serve as the weights w_i. In our experiments, we used a normalization convention: $\sum_{i=1}^{n} w_i = 1$, where w_i are model feature weights. We propose to use this normalization convention for benchmarking compatibility; other choices require a drift detectionthreshold renormalization.

Thus, the overall drift severity is computed as by the equation $S = \frac{1}{n}\sum_{i=1}^{n} w_i s_i$ where n is the number of features, s_i i-th drift meaisre and w_i i^{th} drift weight. As a result, we arrive at drift strength representation incorporating intrinsic dataset statistical properties. We may state that a drift is detected when the total severity $S \geq s_t$, where s_t is a threshold, e.g. $s_t = 0.5$.

This method can be applied in both supervised and unsupervised scenarios. In unsupervised scenarios, it detects distribution changes that likely decrease model performance globally. Consequently, we can initiate an annotation process, employ strategies for updating the dataset and retrain the model. In supervised scenarios, we identify performance decrease using metrics such as accuracy or F1 score. However, the proposed drift analysis provides other useful data: which features exhibit drift, how severe the drifts are, are they gradual, recurring etc. Such insight helps us to react accordingly, for instance, by excluding features that frequently cause drifts.

V. Experiments and Analysis

The MFWDD method was evaluated on the CESNET-QUIC22 [19] and CESNET-TLS-Year22 [18] datasets and guided network classification model retraining over an extended period. In our experimental setting, we initialised the ML model M_1 based on the first week's data $D_1 = D(1, 7)$ with $N = \|D_1\|$ records as a reference and evaluated the rest of the dataset. Furthermore, we introduce the denote by $l = 7$ on the last retraining day. Upon drift detection on day n, the model's dataset is updated to $D_2 = [D_1 \cup D(l, n)]_N = [D(1, 7) \cup D(7, n)]_N$, where $D(i, j)$, denotes the initial dataset truncated to begin with i^{th} and ending with j^{th} day for $i \leq j$, and $[.]_N$: dataset $\mapsto$ dataset operation denotes truncation to last N records based on time ordering[2]. The new model M_2 stems from retraining on the D_2 dataset and so on: $D_n = [D_{n-1} \cup D(l, n)]_N$ and M_n is trained in the D_n dataset.

We use the XGBoost classifier with the default setting of hyperparameters as the model, i.e., XGBoost representative. The XGBoost was chosen due to its popularity in the field of network traffic classification.

Two different feature vectors were used for the evaluation: one supplied by the CESNET-DataZoo library based on PPI metadata and one based on NetTiSA features calculated from PPI metadata containing up to 30 packets. For both datasets, we used all available class labels.

Our experiments aim at three goals:

I The first goal is to evaluate if the MFWDD detector can work well on both feature vectors and both datasets.

II The second goal is to confirm the superiority of the newly trained model M_{n+1} compared to the previous model M_n after drift detection.

III The third goal is to investigate the behavior of different feature vectors on selected network traffic.

Experimental results are thoroughly described in the following subsections.

[2]In our case $\|D_n\| \leq \|D_{n+1}\|$ $\forall$n considered; we may define $[D]_N = D$ for D satisfying $\|D\| \leq N$.

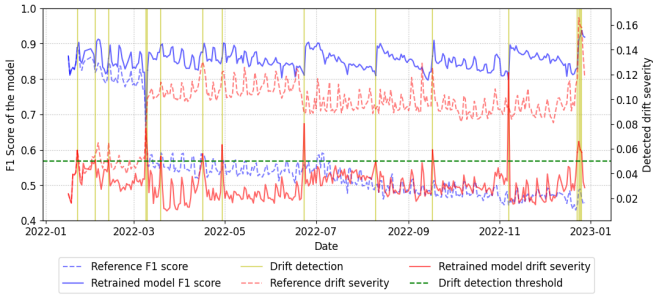
A. CESNET-TLS-Year22 Evaluation

This experiment confirmed that guided retraining controlled by MFWDD vastly improved ML-trained classificator accuracy over time. The maintained model even outperformed the initial model M_1 used as a baseline. Possible explanations are that the concept drift led to the simplification of some problems over time, some distributions might have became more stable or the newly gained datasets were more coherent. In the Fig. 1a and Fig. 1b we may judge how the model reacted to dataset updates for both versions of feature vectors. Vertical lines mark when the drift detection occurred, and the F1 score of the reference and the maintained models may be compared to illustrate model accuracy over time.

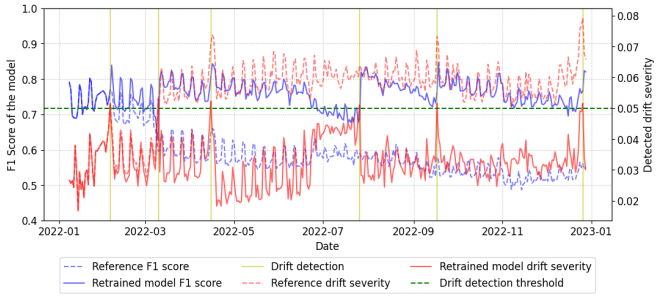
As can be seen in Fig. 1a, in some cases, a single retraining was sufficient to keep the model precise for several months. In other cases, several retrainings were needed to improve the model's performance significantly. This may be due to the incremental drift being present, where the change of concepts may be gradual. Note that these results confirm the importance of well-timidness for model retraining and its impact on the AL efficiency. A single retraining can only lead to the learning of the intermediate concept, and subsequent retrainings might be needed while the new concept stabilizes. The plot of reference drift severity in time helps us understand drift severity if the model was not updated. The periodic behavior [15] is also seen each weekend and sometimes the drift detection threshold is almost attained. Nevertheless, the weekends do not have a significant impact on the model performance, whis is in accordance with Jančička et al. [24].

After the initial model and PPI dataset stabilization during the first weeks, a significant drop in the F1 score and a peak in the drift strength occurred in March. The MFWDD method thus triggered two consecutive retrainings to adapt to new network conditions. The drift is attributed to changes in the network monitoring infrastructure and impacted the majority of classes. Then, we can see recurring drifts every few months, that are driven by *SIZE_4* and *DIR_4* features representing 4$^{\text{th}}$ packet in the network flow. At the end of the year, an incremental drift occurred and resulted in a series of retrainings to improve and stabilize the model performance.

The behavior of the NetTiSA feature vector is depicted in Fig. 1b. From a high-level perspective, the patterns observed agree with that of the PPI feature vector. However, thanks to more stable features, fewer retrainings resulted in a significant F1 improvement upon drift detection. The lengths of PPI in the CESNET-TLS-Year22 dataset, which contains at most 30 packets, probably cause this phenomenon. NetTiSA feature



(a) Simulated model evolution over multiple retrainings on the TLS dataset with PPI feature vector guided by the drift detection. In the case of drift detection, visualised by a yellow vertical line, a model is retrained. The model performance and drift severities may be compared between the reference and the continuously retrained model.



(b) Simulated classification model evolution over multiple retrainings on parts of the TLS dataset with NetTiSA feature vector guided by the MFWDD drift detection. Vertical lines show retraining trigger. The model performance and drift severities between the reference and the continuously updated model may be compared .

vector is calculated based on statistical features, and longer flows would be beneficial as described by Koumar et al. [7].

The average drift strength and amount of feature drift differ for both strategies: the updated model has a variance ranging from 4 % to 30 %, whereas with the no-retrain strategy, drifted features variance ranges from 20 % to 45 %.

In summary, the PPI feature vector usually performs with a higher F1 score but is more sensitive to drifts than NetTiSA. On the other hand, NetTiSA achieved higher stability, though the F1 score was lower. Drifted features showed variance $\approx$ 20 % for updated model, whereas for no-retrain strategy, the drift feature variance ratio was up to 36 %.

B. Outcomes and best practices

In this section, we briefly summarize leading outcomes and recommendations for best practices that stem from the experimental evaluation and analysis of data drifts and their detection by the MFWDD detector. The major outcomes and recommendations are described below.

It is always desirable to evaluate each feature drift severity before using it in the final model. If a feature that exhibits frequent drifts is present in the dataset, it will cause a disfavored performance decrease over time, and in extreme cases, its presence can lead to excessive model retraining. On the other hand, it is important to note that it is often the case that features exhibiting drifts encompass internal specifics for certain network traffic classes and are thus significant for the dataset. The offline setting of our experiments enabled us to evaluate the stability of each feature using the MFWDD. Yet, the MFWDD detector could be used online to guide supervised learning with an overview of the stability of each class.

Our experiments imply that for network traffic classification problems, models based on the SPLT are affected by a large number of detected data drifts throughout long time periods, for example, a year. Therefore, the basic XGBoost representative with SPLT as input features is not so stable. Moreover, from related works by Malekghaini et al. [3], [25], we knew that even highly optimized Deep Learning models are not stable in time. In the related works, this problem was handled using periodic retraining, which cannot be easily done in all classification problems. Therefore, we suggest concentrating not only on the highest precision of trained models but also on their long-term stability because today's models significantly lose precision in a week or two.

Even though periodic retraining is the most straightforward solution in some domains, it brings technical and theoretical questions, such as: How do we gather and manage training datasets? How to detect data obsolescence and how to filter undeserved records. What records should be added to a dataset, on the other hand? Generally, even if it is possible to use periodic retraining, we should avoid it for several reasons:

I Most of the retraining will be unnecessary because no changes in data appear.

II Frequent or periodic, ML model retraining is highly resource-intensive. Furthermore, for complex problems, this approach might be unmanageable or unfeasible [26].

III Frequent retraining without significant changes in data can cause fitting of the model on non-stable features because the model becomes too tailored to specific subsets of data, which often change to gain better performance by an insignificant percentage, reducing its generality [27].

IV It neglects some real-world effects, such as the weekend phenomenon described in [15] or other recurring events.

In real-world scenarios, a drift detector should be deployed side by side with the ML-model as a means for continuous monitoring, e.g., by a SOC team. The model based on the NetTiSA flow features supports this statement: it is a stable model with several drifts occuring throughout several months.

VI. Conclusions

We introduced a novel method called Model-based Feature Weight Drift Detection (MFWDD) to detect and investigate drifts in network datasets. The proposed method was evaluated using the large public datasets containing services in TLS and QUIC protocols, which are very common communication protocols in network traffic. The used datasets contain long-term traffic and allow for the identification of drifts. However, the MFWDD can be used with other classification tasks even

outside our network security domain because the computation of final drift severity does not depend on a particular dataset or types of the features.

The introduced drift detection method is ideal for a more detailed investigation of datasets, which we demonstrated in the domain of network traffic classification. We identified differences in the behavior of the PPI features and the NetTiSA flow feature vectors with respect to the selected dataset. The method was able to properly detect data drifts on both selected datasets and both selected feature vectors.

Furthermore, the MFWDD also works well in an unsupervised manner. Therefore, there is no need for labels to compare distributions. That means we can perform detection on problems for which it is very difficult/expensive to get labels (for example, by using active scanning or performing human annotation). Nevertheless, the detector can detect data drifts in real-world scenarios. When the detector detects a drift, the model should be retrained. First, the distribution changes can be evaluated, and the data records that may be deleted from the datasets and new records added to the dataset can be automatically evaluated. Second, the model is retrained on the created dataset. Lastly, the model is deployed instead of the original model.

We made the MFWDD framework containing both supervised and unsupervised methods publicly available as an open-source project in the GitHub repository. Therefore, the source code and examples are published for possible replication of the described results, for evaluation of additional datasets, or for follow-up on this research.

A. Future work

The developed MFWDD framework provides promising methods for network drift detection and, thanks to its extensibility, enables an additional use of statistical methods. In future work, we would like to focus on deeper analysis of datasets using the MFWDD framework and, from generated logs, classify dataset drifts in parallel with corresponding events in the network environment in the long-term measure. In related works, researchers observed several categories of drift patterns, sudden, gradual, incremental, and recurrent, Gama et al. [22], and Lu et al. [23], but more research on drift types and their impact to a dataset of ML model quality is needed.

We would like to leverage the knowledge of the ML model and classification performance score to get a more detailed identification of events (average F1 score improvement/degradation with drift, class changes, periodicity events, drift strength statistics, etc.). These metrics will help to identify dataset quality and better understand selected dataset use case [10]. Due to the complexity of collecting network traffic datasets, there is a demand for proactive analysis to avoid errors that can lead to fake results [28], [29].

REFERENCES

[1] G. Aceto *et al.*, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

[2] J. Luxemburk and T. Čejka, "Fine-grained tls services classification with reject option," *Computer Networks*, vol. 220, p. 109467, 2023.

[3] N. Malekghaini *et al.*, "Deep learning for encrypted traffic classification in the face of data drift: An empirical study," *Computer Networks*.

[4] J. Dai *et al.*, "Glads: A global-local attention data selection model for multimodal multitask encrypted traffic classification of iot," *Computer Networks*, vol. 225, p. 109652, 2023.

[5] J. Luxemburk *et al.*, "Encrypted traffic classification: the quic case," in *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023, pp. 1–10.

[6] I. Akbari *et al.*, "A look behind the curtain: traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 1, pp. 1–26, 2021.

[7] J. Koumar *et al.*, "NetTiSA: Extended IP flow with time-series features for universal bandwidth-constrained high-speed network traffic classification," *Computer Networks*, p. 110147, 2024.

[8] R. Plný *et al.*, "DeCrypto: Finding Cryptocurrency Miners on ISP Networks," in *NordSec 2022*.

[9] L. Korycki *et al.*, "Concept drift detection from multi-class imbalanced data streams," in *2021 IEEE ICDE*, Los Alamitos, CA, USA.

[10] D. Soukup *et al.*, "Towards evaluating quality of datasets for network traffic domain," in *CNSM 2021*.

[11] I. Goldenberg *et al.*, "Survey of distance measures for quantifying concept drift and shift in numeric data," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 591–615, 2019.

[12] O. Filippova. Which test is the best? we compared 5 methods to detect data drift on large datasets. [Online]. Available: https://www.evidentlyai.com/blog/data-drift-detection-large-datasets

[13] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[14] G. Andresini *et al.*, "A network intrusion detection system for concept drifting network traffic data," in *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*. Springer, 2021, pp. 111–121.

[15] L. Jančička *et al.*, "Analysis of statistical distribution changes of input features in network traffic classification domain," in *NOMS 2024 IEEE Network Operations and Management Symposium*. IEEE.

[16] I. Žliobaitė *et al.*, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 27–39, 2014.

[17] W. Liu *et al.*, "Multiclass imbalanced and concept drift network traffic classification framework based on online active learning," *Engineering Applications of Artificial Intelligence*.

[18] K. Hynek *et al.*, "CESNET-TLS-Year22: A year-spanning TLS network traffic dataset from backbone lines," Feb. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10608607

[19] J. Luxemburk *et al.*, "Cesnet-quic22: a large one-month quic network traffic dataset from backbone lines," *Data in Brief*, p. 108888, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340923000069

[20] J. Luxemburk and K. Hynek, "Datazoo: Streamlining traffic classification experiments," in *Proceedings of the 2023 on Explainable and Safety Bounded, Fidelitous, Machine Learning for Networking*, 2023, pp. 3–7.

[21] J. Luxemburk *et al.*, "Encrypted traffic classification: the quic case," in *TMA*. IEEE, 2023.

[22] J. a. Gama *et al.*, "A survey on concept drift adaptation," vol. 46, no. 4, 3 2014. [Online]. Available: https://doi.org/10.1145/2523813

[23] J. Lu *et al.*, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, 2019.

[24] L. Jančička *et al.*, "Analysis of statistical distribution changes of input features in network traffic classification domain," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, 2024.

[25] N. Malekghaini *et al.*, "Data drift in dl: Lessons learned from encrypted traffic classification," in *2022 IFIP Networking Conference (IFIP Networking)*. IEEE, 2022, pp. 1–9.

[26] X. Zhu and D. Klabjan, "Continual neural network model retraining," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 1163–1171.

[27] F. Chen *et al.*, "Lara: A light and anti-overfitting retraining approach for unsupervised time series anomaly detection," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 4138–4149.

[28] L. Liu *et al.*, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*, 2022, pp. 254–262.

[29] M. Lanvin *et al.*, "Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes," 2023.