



Towards reusable models in traffic classification

Jan Luxemburk 
FIT CTU & CESNET
Prague, Czech Republic
luxemburk@cesnet.cz

Karel Hynek 
FIT CTU & CESNET
Prague, Czech Republic
hynekkar@cesnet.cz

Abstract—The machine learning communities, such as those around computer vision or natural language processing, have developed numerous supportive tools. In contrast, the network traffic classification field falls behind, and the lack of standard datasets and model architectures holds the entire field back. This paper aims to address this issue. We introduce CESNET Models, a package comprising pre-trained deep learning models tailored for traffic classification. The included models are trained on public datasets for the task of web service classification. Using the new package, researchers and practitioners can skip model design from scratch and the collection of large datasets but instead focus on fine-tuning and adapting the models to their specific needs, thus accelerating the pace of research and development in network traffic classification.

Index Terms—Traffic classification, Machine learning, Neural networks, Pre-trained models, TLS, QUIC

I. INTRODUCTION

Machine learning (ML) is one of the core technologies used in network traffic classification (TC) for a broad scope of tasks. Nevertheless, ML use in TC still falls behind established ML fields such as natural language processing and computer vision due to a lack of supportive tools, model repositories, and benchmarks [1]. Therefore, progress in TC is slow since all experiments, models, and datasets often need to be created from scratch.

In our previous work [2], we introduced a DataZoo toolset for streamlining the work with large network traffic datasets. DataZoo implements configurable dataset processing, such as train-validation-test split or class selection, which minimizes the space for potential errors. However, it is limited to data handling, and support for other time-consuming tasks, such as model architecture design, is missing.

This paper introduces CESNET Models, a new package containing neural network architectures for traffic classification, together with their pre-trained weights and data transformations. The goal is to provide convenient access to pre-trained models, following a common practice in other ML domains. Our inspiration was the torchvision project that offers a suite of standard architectures for computer vision (e.g., ResNet) with pre-trained weights on popular datasets (e.g., ImageNet).

Using CESNET Models, researchers and practitioners can reproduce published results or improve the included state-of-

the-art classifiers without the need for extensive model development or training. Furthermore, the inclusion of pre-trained weights on large networking datasets enables transfer learning and model customization for specific TC tasks.

We acknowledge that reproducing the success of libraries such as torchvision is a challenging task, and this work and the DataZoo toolset are just the first steps in that direction. The following section discusses some of the challenging obstacles to more reusable models in TC.

II. OBSTACLES FOR REUSABLE MODELS IN TC

Reusing of ML models in the TC field has been minimal compared to other fields. One of the reasons is that researchers often do not publish source code for their experiments and model definitions, which is more common in other ML fields. Nevertheless, there are additional difficulties that complicate model reuse: differences in computer networks and their users, diverse production environments influencing the requirements in terms of false positive rates and inference speed, or general label differences worldwide [3].

However, we consider the lack of standard input feature format to be one of the main obstacles. In computer vision tasks, the inputs are images encoded as tensors of floats representing pixels. Sizes of images can differ, but there are standard methods for downscaling or upscaling with padding. In the TC field, there is no such standard format. Of course, there is full packet capture (PCAP), but it does not scale for even mid-sized datasets and can leak private information about network users. We received numerous emails asking for PCAP versions of our public flow-based datasets since each researcher would like to use its own toolsets to extract *their* feature set out of the original PCAP (which we do not have). We believe that if the TC community agreed on a standard set of features, it would bolster data and model sharing and would allow shifting the focus to more interesting tasks.

Despite the variance of model inputs used across the research works, we have identified three common types of features that have to be considered during the design phase of reusable models: flow statistics, packet sequences, and the first packet payload.

A. Common model inputs

Flow statistics represent common features describing network connection, such as the number of transmitted bytes and packets in both communication directions or the time duration

This work was supported by the Ministry of the Interior of the Czech Republic, grant No. VJ02010024: “Flow-Based Encrypted Traffic Analysis,” and also by the CTU in Prague, grant No. SGS23/207/OHK3/3T/18.

of the connection. Apart from those well-known features, there could be features related to the presence of individual TCP flags, histograms, or an indication of how the connection ended (e.g., with TCP FIN termination). An important distinction is whether flow statistics are computed from the entire connection, for example, when a TC model is deployed at a flow collector, or from the first N packets when the model is operating close to the monitored network for *early classification*.

Packet sequences describe the first N packets of the connection for each packet, including its size, direction, inter-packet time, and sometimes the size of the TCP window or the presence of the PUSH flag (these TCP-related fields are set to zero for UDP). The length of the sequence N is an important parameter that depends on operational requirements. Another important choice is whether to include packets with no payload, such as TCP ACKs, into the sequences.

The payload of the first packet is often used in TC literature. However, its usefulness is questionable. For unencrypted traffic, complex models learn to extract informative strings, which is a task more suitable for pattern-matching algorithms. For encrypted TLS traffic, models learn to extract fields from the ClientHello message, such as the Server Name Indication (SNI) domain. This SNI domain, however, is often used for labeling and, therefore, should be masked from the input to prevent "leaking" label information to the model. Moreover, the new Encrypted Client Hello extension will make this processing of TLS handshakes obsolete.

The variations in implementations of model inputs are summarized in Table I. The next section introduces models provided in the new package and the motivation for our design decision about their inputs.

TABLE I: Variations in model inputs across TC proposals.

Flow statistics	Packet size - entire vs. after transport headers Feature set Computed from an entire connection vs. the first N packets Differences in the flow creation process (timeouts, hashing, collisions, and more)
Packet sequences	Packet size - entire vs. after transport headers Included vs. excluded TCP SYNs, ACKs Sequence length Extra packet features, such as TCP window size or the presence of the PUSH flag
First packet payload	Size and start offset of the payload

III. CESNET MODELS

The goal of the CESNET Models package is to provide model architectures, pre-trained weights, and data transformations for encrypted traffic classification. The package is implemented in PyTorch, and the API is similar to the torchvision project. Each model (e.g., mm-CESNET V2) has a constructor function (`mm_cesnet_v2`) that accepts a weights argument from an enum of all available pre-trained weights for the given model (`MM_CESNET_V2_Weights`). Using the model constructor function without the weights argument will use random initial weights for training from scratch.

Data transformations are used as a preprocessing step before putting data into a model. The current version imple-

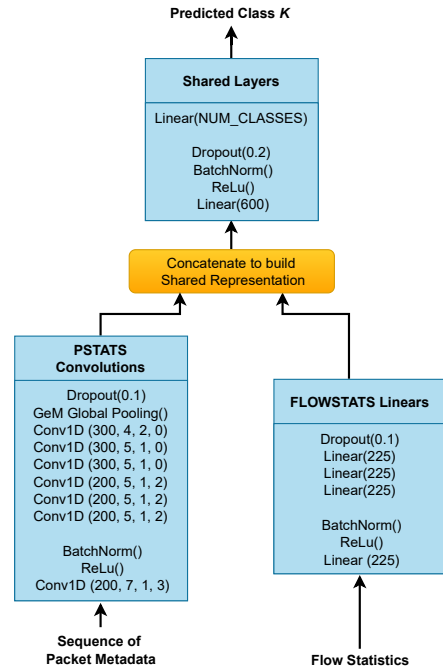


Fig. 1: The mm-CESNET V2 model. The parameters represent: Conv1D(#filters, kernel_size, stride, padding), Linear(#out_features). Some layers are omitted to save space.

ments three transformations—scaling and clipping of packet sequences and flow statistics, and normalization of packet histograms. If data scaling transforms were used for training a model, those transforms are included in the pre-trained weights (i.e., a loaded pre-trained model will use the same statistics for scaling the inputs as were used during training).

A. Network architectures

We developed our multi-modal architecture (mm-CESNET) in previous research focusing on TLS and QUIC traffic classification. It is similar to networks used in other TC works, such as MIMETIC [4] or the tripartite model used by Akbari et al. [5]. These networks have separate chains for processing input modalities. Outputs of those chains are concatenated to create a shared representation, which is further processed. An example of this multi-modal architecture is visualized in Figure 1. The main differences between the aforementioned networks are the input modalities and the types of "sub-networks" used for processing them (i.e., convolutional neural network (CNN), recurrent neural network (RNN), or multilayer perceptron (MLP)). The selected options for those networks are¹:

- MIMETIC [4] - CNN for payload and RNN for packet sequences. No processing of flow statistics.
- Tripartite model [5] - CNN for payload, RNN for packet sequences, and MLP for flow statistics.
- mm-CESNET [6], [7] - CNN for packet sequences and MLP for flow statistics. No processing of packet payload.

¹All three networks use MLP for processing the shared representation.

```

from cesnet_models.models import MM_CESNET_V2_Weights, mm_cesnet_v2

pretrained_weights = MM_CESNET_V2_Weights.CESNET_QUIC22_Week44
model = mm_cesnet_v2(weights=pretrained_weights, model_dir="models/")
transforms = pretrained_weights.transforms

dataloader = load_from_datazoo( # Load data from DataZoo using transforms from the pretrained model
    ppi_transform=transforms["ppi_transform"],
    flowstats_transform=transforms["flowstats_transform"],
    flowstats_phist_transform=transforms["flowstats_phist_transform"],
    ...
)
# Example tasks here
# - Fine-tuning
# - Reuse the model for some other task, i.e. transfer learning
# - Optimize the network with, for example, pruning
# - XAI techniques, such as SHAP or visualizations of convolutional filters

test_labels, preds = compute_model_predictions(model, dataloader) # Test the model

```

Listing 1: Example usage of the `cesnet_models` package. More complete examples at <https://github.com/CESNET/cesnet-tcexamples/>.

Our goal here is not to compare the models but to demonstrate options used in published research and to provide motivation for design decisions in our multi-modal architecture. We (1) opted to ignore packet payload because our focus is large-scale classification of TLS and QUIC traffic at a central flow collector. Transmitting the first packet payload of each connection to the collector could overload monitoring lines, and the added value is minimal, since we used the SNI domain for ground-truth labeling, and therefore, it would have to be masked from the model input. Moreover, we do not consider payload processing to be future-proof in the light of the new Encrypted Client Hello TLS extension. We (2) chose 1D CNN for processing packet sequences because recurrent networks tend to be slower than convolutional ones, and high processing speed is one of our main requirements. Also, we use packet sequences with a maximum length of 30 packets; thus, we cannot take much advantage of the fact that RNNs are capable of processing variable-length input (instead, we pad our sequences with zeroes for CNN processing). Also, we (3) use flow statistics input with MLP processing, which is the sensible option given that flow statistics are not sequence-like and are similar to tabular data. In our previous experiments on TLS, however, we found that using flow statistics is not that important. Omitting this input resulted in less than 1% decrease in performance [6].

B. Available models

The CESNET Models package currently provides two models with our multi-modal architecture. One older, mm-CESNET V1, from a TLS classification paper [6] and mm-CESNET V2 updated for a QUIC classification task [7], which is visualized in Figure 1. The differences between the two versions are in layer sizes, dropout rates, and added pooling operation in the CNN part processing packet sequences. The mm-CESNET V1 model was trained on the first week of the CESNET-TLS22 dataset [6], achieving 97% classification accuracy among 192 web service classes. The mm-CESNET V2 model was trained on the first week of the CESNET-QUIC22

dataset [8], achieving 86% test performance among 102 web service classes. Both test performance measurements were made using the traffic from a week immediately following the training period.

C. Usage

A simplified example of how the CESNET Models package can be used is provided in Listing 1. The package integrates well with our data handling DataZoo toolset. DataZoo was used to train the included models and to fit the scaling transformations that are provided together with the weights (the *transforms* dict in the example). The expected model input is in the format $\text{tuple}(\text{batch_ppi}, \text{batch_flowstats})^2$, which is the same format as provided by the DataZoo dataloader interface. The CESNET Models package is documented³ and can be installed from PyPI⁴ or GitHub⁵

IV. CONCLUSION

In this paper, we presented CESNET Models and discussed the challenges related to reusing models in the network traffic classification field. The TC community must address these challenges to catch up with other ML fields. We recognize that we cannot solve this task alone. Broader cooperation of research groups is needed to settle on the model input format and start sharing and reusing models more.

As future work, we plan to add data augmentations that can be beneficial for model training and are currently trending in TC [9], [10]. We plan to implement augmentations as data transformations in this package and allow their composing, which is not supported in the current version. We also plan to implement more TC model architectures.

²*ppi* stands for per-packet information, which is another name for packet sequences; *flowstats* stands for flow statistics. More details in the model reference https://cesnet.github.io/cesnet-models/reference_models/.

³<https://cesnet.github.io/cesnet-models/>.

⁴<https://pypi.org/project/cesnet-models/>.

⁵<https://github.com/CESNET/cesnet-models>.

REFERENCES

- [1] C. Wang, A. Finamore, L. Yang, K. Fauvel, and D. Rossi, "AppClassNet: A commercial-grade dataset for application identification research," *SIGCOMM Comput. Commun. Rev.*, vol. 52, no. 3, p. 19–27, sep 2022. [Online]. Available: <https://doi.org/10.1145/3561954.3561958>
- [2] 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*, ser. SAFE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 3–7. [Online]. Available: <https://doi.org/10.1145/3630050.3630176>
- [3] F. Pacheco, E. Exposito, M. Gineste, C. Baudoïn, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.
- [4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, p. 106944, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128619304669>
- [5] I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "A look behind the curtain: Traffic classification in an increasingly encrypted web," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, feb 2021. [Online]. Available: <https://doi.org/10.1145/3447382>
- [6] J. Luxemburk and T. Čejka, "Fine-grained TLS services classification with reject option," *Computer Networks*, vol. 220, p. 109467, Jan. 2023. [Online]. Available: <https://doi.org/10.1016/j.comnet.2022.109467>
- [7] J. Luxemburk, K. Hynek, and T. Čejka, "Encrypted traffic classification: the quic case," in *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023, pp. 1–10.
- [8] J. Luxemburk, K. Hynek, T. Čejka, A. Lukačovič, and P. Šiška, "CESNET-QUIC22: A large one-month QUIC network traffic dataset from backbone lines," *Data in Brief*, vol. 46, p. 108888, Feb. 2023. [Online]. Available: <https://doi.org/10.1016/j.dib.2023.108888>
- [9] C. Wang, A. Finamore, P. Michiardi, M. Gallo, and D. Rossi, "Data augmentation for traffic classification," in *International Conference on Passive and Active Network Measurement*. Springer, 2024, pp. 159–186.
- [10] R. Xie, J. Cao, E. Dong, M. Xu, K. Sun, Q. Li, L. Shen, and M. Zhang, "Rosetta: Enabling robust TLS encrypted traffic classification in diverse network environments with TCP-Aware traffic augmentation," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 625–642. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity23/presentation/xie>