GENERATIVE TRANSFORMER FRAMEWORK FOR NETWORK TRAFFIC GENERATION AND CLASSIFICATION

DOI: 10.36724/2072-8735-2020-14-11-64-71

Radion F. Bikmukhamedov,
LLC "Factory5";
Kazan National Research Technical University named after A. N. Tupolev – KAI (KNRTU-KAI), Kazan, Russia,
radion.bikmukhamedov@pm.me

Adel F. Nadeev,
Kazan National Research Technical University named after A. N. Tupolev – KAI (KNRTU-KAI), Kazan, Russia,
afnadeev@kai.ru

Manuscript received 05 September 2020;
Revised 26 September 2020;
Accepted 21 October 2020

Keywords: *transformer, network traffic classifier, traffic generator, neural network, Random Forest, Markov chain, K-Means, transfer learning*

We introduced generative transformer-based network traffic model suitable for generating and classification tasks. Only packet size and inter-packet time sequences are used as flow features to unify the inputs for the two tasks. The source feature space is scaled and clustered with K-Means to form discrete sequences as model inputs. The model can be trained in two modes: (i) autoregressively, for network traffic generating, where the first token of training sequence represents a flow class, (ii) as a network flow classifier. The evaluation of generated traffic by means of Kolmogorov-Smirnov statistic demonstrated that its quality is on par with the first-order Markov chain, which was trained on each traffic class independently. The metric measured distances between source and generated empirical cumulative distributions of such parameters as packet size, inter-arrival time, throughput and number of packets per flow in directions to and from traffic origin. It was shown that enriching the dataset with external traffic from different domain improves quality of the generated traffic on target classes. The experiment results showed positive influence of generative pre-training on quality of the traffic classification task. In case of using the pre-trained model as a feature extractor for a linear algorithm, the quality was close to Random Forest trained on the raw sequences. When all model parameters are trained, the classifier outperforms the ensemble on average by 4% according to the F1-macro metric.

Information about authors:

Radion F. Bikmukhamedov, *LLC "Factory5", Machine learning engineer, PhD candidate at KNRTU-KAI, Kazan National Research Technical University named after A. N. Tupolev - KAI (KNRTU-KAI), Kazan, Russia*

Adel F. Nadeev, *Head of Radioelectronic and Telecommunication systems department, PhD, Kazan National Research Technical University named after A. N. Tupolev - KAI (KNRTU-KAI), Kazan, Russia*

Для цитирования:
Бикмухамедов Р.Ф., Надеев А.Ф. Генеративные модели на базе нейросетевой архитектуры Transformer для классификации и моделирования сетевого трафика // T-Comm: Телекоммуникации и транспорт. 2020. Том 14. №11. С. 64-71.

For citation:
Bikmukhamedov .F., Nadeev A.F. (2020) Generative transformer framework for network traffic generation and classification. *T-Comm*, vol. 14, no.11, pp. 64-71. (*in Russian*)

Introduction

A network flow classification system is the key component for network traffic control and monitoring tasks. The current trend aimed at increasing user privacy shifts development of network protocols and web applications towards encrypting plain text information, such as Domain Name Service (DNS) responses and requested services within Service Name Indicator (SNI) field of a TLS handshake, significantly limiting applicability of deep packet inspection. A viable alternative could be a machine learning based system for statistical discrimination of network sessions.

The steady growth of available compute power and presence of significant volume of data are the major catalysts for development of machine learning algorithms for a broad range of applications. A notable attention gained concept of transfer learning, when once trained model can be applied for a different task within the same application domain. This is closely followed by the idea of model pre-training in self-supervised mode on a large unlabeled corpus, which further allows applying the model on various supervised tasks and it is especially effective in scarce-label scenarios. Examples of such self-supervised approaches can be often found in natural language processing (NLP) domain within approaches aimed at transforming text to numerical representation, which started with Word2Vec embeddings [1] and have been significantly improved with models based on Transformer blocks [2]. The transformer architecture has become the cornerstone for numerous papers within NLP field, pushing state-of-the-art results for various tasks, such as text classification and sequences tagging and outperforming recurrent and convolutional models. The architecture was designed to process discrete sequence more efficiently, in part because of the built-in multi-headed self-attention mechanism. The architecture allowed significant speed-ups for pre-training on large data volumes and made possible to train larger than ever models. Several papers demonstrated effectiveness of auto-regressive models using transformer decoder blocks for text generating tasks [3], [4], whereas encoder blocks are preferred for building discriminative models, achieving superior results on various classifications tasks [5], but not suitable for sequence generation problems.

An application of approaches originally developed for natural language processing tasks is not a rarity for network traffic field. For example, one of the first works using Word2Vec for traffic classification domain was presented in [6], where the authors trained the model on domain names from DNS queries and further applied it as an extractor of a feature subset for network flow categorization. It appears, the authors of [7] were inspired by the papers devoted to text translation and presented a traffic classification model utilizing only packet sizes. Structurally, the model consists of packet embedding, followed by bi-directional recurrent encoder-decoders, whose output is branched off to the classification and the input reconstruction blocks. Thus, the tasks of classification and input sequence modeling are solved simultaneously during training. However, the results of the experiments showed when the share of reconstruction error in the total loss function increased, the classification performance deteriorated, which indicates the redundancy of the joint loss function for this task.

Taking into account the tendency to encrypt signaling information within flows and packets, very few statistical flow attributes for the discrimination task are left, namely, inter-packet time (IPT) and packet size (PS). Moreover, these parameters can be used to build statistical traffic generators of individual devices and flows. Taking into account the similarity of input data, the purpose of our work is to create a unified model synthesis approach, applicable for generation and classification of network flows of different categories. When synthesizing the approach, which uses decoder transformer blocks, it is necessary to give answers to the questions concerning (i) transformation of network flows into integer discrete sequences, (ii) peculiarities of creating a model, which takes into account the specifics of each task, (iii) training procedures and (iv) application of trained models.

The results of the work are as follows:

1. For the first time we show the generative framework for synthesis of single-model network traffic generators and classifiers that contain information about different traffic classes.
2. The proposed approach allows constructing a model that can be used as an independent generator of traffic of different classes, and the quality of generated data is comparable by objective metrics to a Markov-based model that is trained for each class separately.
3. Pre-training on the traffic generation task improves the classification quality. Also, the pre-trained model can be used as a feature extractor for classifiers.
4. We release the source code, datasets and generator checkpoints.

Proposed framework

The use of a unified input data representation for traffic generation and classification tasks allows to apply a common approach to model building. The input object is a bidirectional packet stream with packets sharing the same transport layer protocol and common [IP address, transport port] pairs.

Traditionally, transformer-based models have a limitation on the length of the input sequence, as the complexity of processing grows quadratically with its length L. The network stream itself may be represented as a matrix $\boldsymbol{F} \in L \times 2$:

$$\boldsymbol{F} = [\boldsymbol{p}_0, \boldsymbol{p}_1, \dots, \boldsymbol{p}_{L-1}]^T,$$

where $\boldsymbol{p}_i = [PS_i, IPT_i]$ is a vector with i-th packet features, where PS has negative values for direction opposite to the flow initiator, and inter-packet times are always positive.

Taking into account that transformer models operate with discrete inputs, it is necessary to perform conversion of the initial two-dimensional feature space into one-dimensional discrete one, which can be considered analogous to "tokenization" for NLP tasks.

One of the possible options is to use "soft" quantization approaches, when the source space is approximated by components of mixture distributions, for example, Gaussian ones. But mixture models have a significant number of parameters to train (including covariance matrices and vectors of mean values), which, given a significant amount of data (millions of network flows), makes them hardly applicable for this task.

On the other hand, "hard" quantization models, such as K-Means, are simpler and more accessible for learning on large volumes of information, and the number of total parameters corresponds to the number of selected clusters. Given that K-Means operates in Euclidean distance space, it is necessary to scale the initial features to correctly form the clusters. Thus, the packet sizes in bytes were linearly scaled by dividing by 1500 (as the maximum possible value of an IP packet), and the inter-packet intervals measured in microseconds were logarithmized (except for the values equal to 0).

The next step is to augment the cluster sequence with special tokens, which denote the beginning and end of the flow. When training a model for the traffic generation task, we assume that assigning the first token in accordance with the protocol/application of the stream will afterwards allow to sample the packets (encoded as cluster numbers) by submitting only the token of the required application. The generation process stops when the end of the flow token appears. It should be stressed that in order to generate class-specific flows, the training data has to contain flow labels. As for the classification task, we can either train model from scratch, or adapt a trained generator. When adapting pre-trained with class-specific tokens model, the first token has to be either masked (with the help of the attention mask) or put the same for all the flows. Additionally, if the number of packets is less than the limit , the vector is appended with special PAD tokens.

GPT-2 was chosen as the target architecture [3], which was successfully applied for a number of applications in the natural language processing domain. The basic configuration of GPT-2 consists of 12 decoder blocks, each having 12 attention heads, with the decoder dimension equal to 768. Such model has about 120 million parameters, which is quite expensive for training.

To reduce the training cost, the model configuration is optimized in terms of used parameters. Thus, in our model the number of decoder blocks was halved to 6 that corresponds to the configuration of distilled versions of transformers [8], while the decoder dimension was reduced to 512. Moreover, studies of the architecture showed that decreasing the number of attention heads to 8 practically does not affect the final performance [9].

As a result, the classifier training process can be divided into the following stages:

1. Flow assembling and packet feature extraction
2. Training the clustering model with the subsequent tokenization of packets in the flow.
3. Autoregressive model training (optional, allows using the model for traffic generation).
4. Training the model for the flow classification task.

The generic model is illustrated in

Fig. *1*. Note that corresponds to two special tokens with quantized packets, is equal to the number of flow classes within a dataset, is the number of cluster, and is the number of special tokens. The model configuration is fixed and does not change except for the input dimension of cluster embeddings block that depends on the number of classes during pre-training (which can widely differ across datasets), and the classification layer is affected by a training mode. Thus, during generative pre-training, that corresponds to the total number of tokens, or for the classification task.

Used data

We adapted NFStream [10] to assign flow labels and extract packet features from .pcap files, which in turn utilizes deep packet inspection module nDPI. The flow export timeout is set to one minute, thus, prioritizing the classification task during pre-processing. For long sessions, this can lead to multiple flows with the same key (IP addresses, ports and protocol), which are fragments of the first flow and require filtering when training models in the classification mode.

Table 1

Used traffic datasets

Name	Total .pcap size, GB	Flow number, thousands	Source
UNSWB	100	2048	[11]
ICSX		421	[12]
IoT		575	[13]
Local		484	[14]

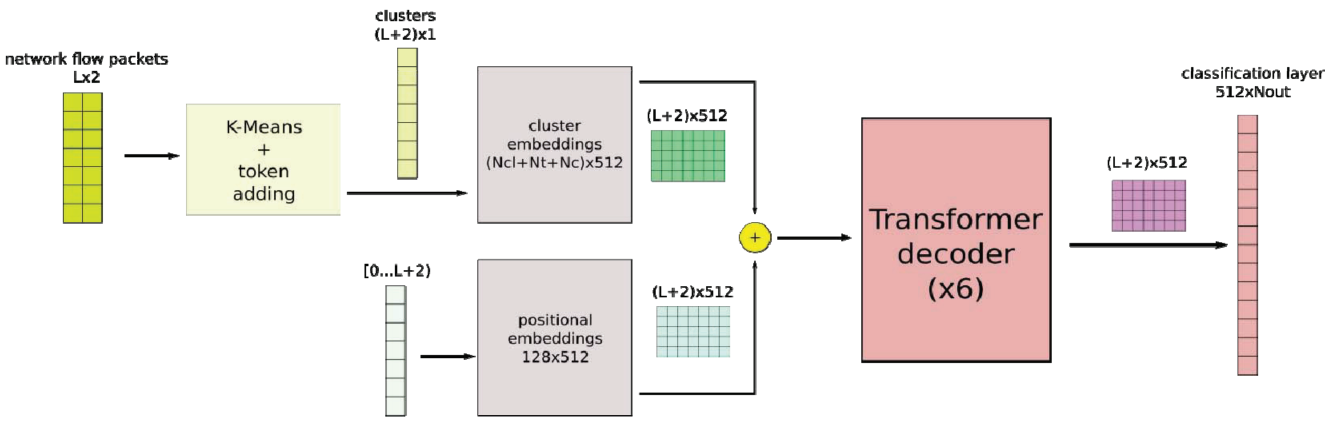

Fig. 1. Proposed network traffic model

The used for model evaluation datasets are listed in

Table *1*, where "Local" denotes our custom traffic dump that had been collected in the campus network for ten days, and "IoT" represents flows of IoT devices only. The artificial "UNSWB" and "ICSX" datasets are often used for evaluation of malware detectors that allow us to represent them as external. Thus, "Local" and "IoT" were chosen as target datasets for model performance analysis when "UNSWB" and "ICSX" were used as sources of additional (external) data for corresponding evaluation scenarios. Target traffic was split with ratio 3:1 to form a training and test sets, while preserving class proportions. The distribution of classes in the classification training set is shown in Fig. 2. The preprocessed datasets and model code can be found in [14].

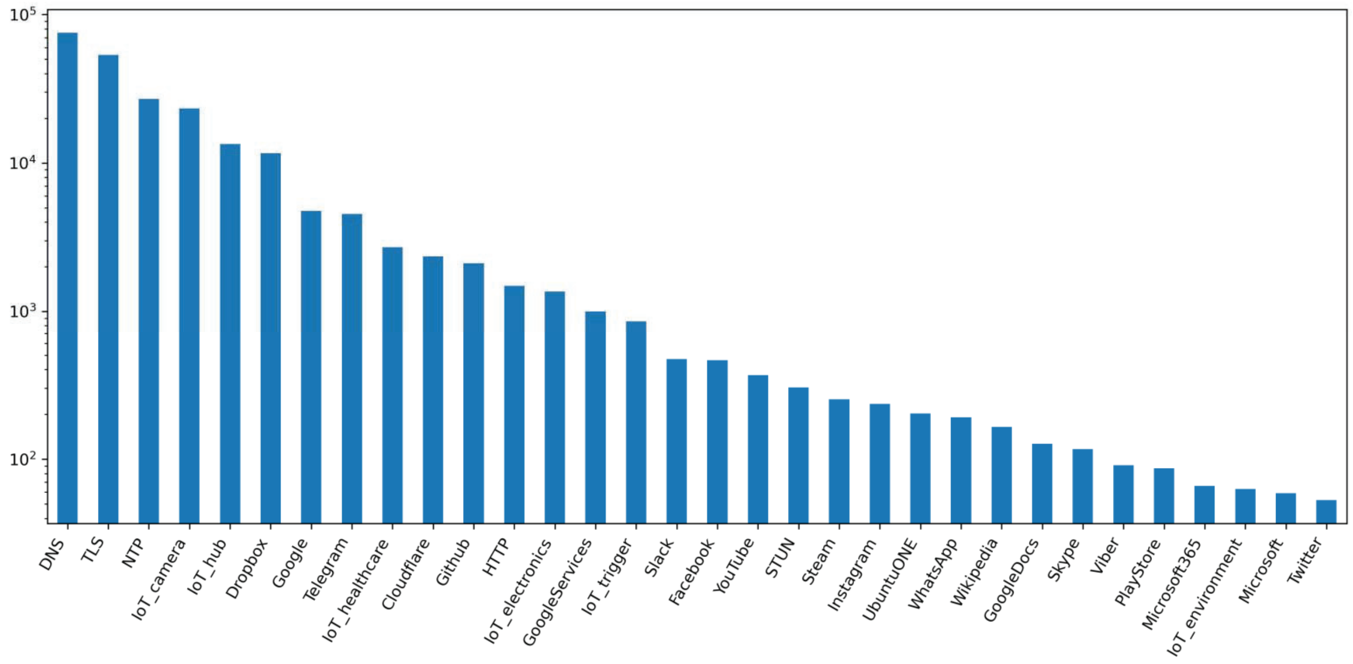

Fig. 2. Flow number distribution by classes within the training set for classifiers

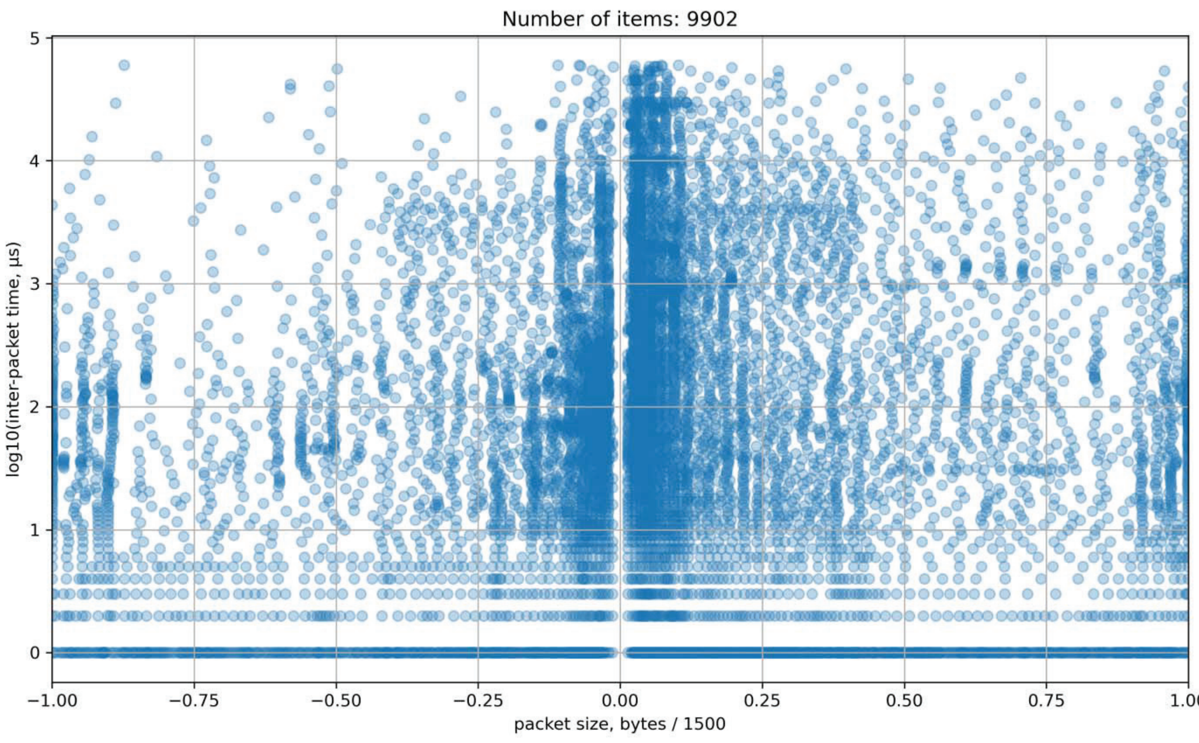

Fig. 3. K-Means clusters used for quantization of the source packet space

According to the analysis of extracted flows, the number of packets per flow in more than 99.9% of cases does not exceed 126, which makes possible to set the model's input dimension to 128, taking into account special tokens. As a result, the model without the output layer has about 24 million parameters, which is about 5 times less than a GPT-2 model with the basic configuration. In addition, the length of the input sequence depends on the learning mode.

When training the generator, the number of used packets is selected as $L = 126$ to stimulate accounting of all available flow statistics. In the classification mode, the important factor is the speed of decision-making, the quality of which, in turn, depends on the number of available packets. Considering the analysis of these relationships given in [15], $L = 20$ was selected.

Given the significant amount of data (hundreds of millions of packets), a library supporting CUDA accelerators, libKMCUDA, was used to train the K-Means clustering model. The maximum number of clusters was chosen empirically with logarithmic step in order to minimize data loss during quantization. As a result, during the initialization of the algorithm the number of clusters was selected equal to $2^{14} = 16384$ that yielded 9902 non-empty clusters after training.

The procedure was conducted once for all evaluation scenarios on the aggregated training set. As it is seen in Fig. 3, the source packet space is filled with cluster centroids quite densely, and a significant number of clusters are concentrated at the near-zero inter-packet interval, which is explained by the packet aggregation at the link layer.

To compare the quality of models, the following combinations of datasets were used:

1. External and target traffic. Allows to estimate the impact of using additional data during pre-training.
2. Target traffic. Aims to evaluate model performance when pre-training on data from the target network/domain.
3. External traffic. Evaluates applicability of the model when target domain/network differs from the pre-training one.

Evaluation results

Traffic generation

To check the quality of the proposed model-generator, auto-regressive training was conducted both on the target and mixed data scenarios. When training on external data only, set of target classes is not a complete subset of external classes, so this scenario was excluded from the evaluation to avoid distortions in the aggregate statistics.

The Markov chain was used as the baseline model, since it is widely applied for network traffic modeling tasks. In this work, the chain describes transitions between clusters of packets and, unlike the transformer model, it was trained on each traffic class individually and within the target dataset only.

In order to evaluate the quality of the generated traffic, it was decided to utilize the following parameters: distributions of packet size, inter-arrival time (IAT) and bitrate for directions from and to the connection origin [16]. The two-sample Kolmogorov-Smirnov metric is used to estimate proximity of empirical generated parameters distributions to the initial ones from the training set. The metric has values within [0, 1] range, measuring the maximum distance between two empirical cumulative distributions.

The aggregated by target classes metrics are presented in Fig. 4. As we see, training the transformer with class-specific first tokens indeed allows generating required traffic. In terms of "number of packets/flow" parameter, the Markov chain underperformed the proposed model, since it has no mechanism to control duration of the generated sequence and depends entirely on the state transition matrix.

On the other hand, the Markov property makes the training procedure simple, providing discrete packet cluster distribution quite close to the original. This positively affected the PS, IAT and bitrate parameters, which in some cases are marginally better than ones from a more complex transformer model. In general, the proposed model shows quality comparable by median values to individual Markov chains, which can be further improved and stabilized by adding external traffic.

Traffic classification

When creating a classifier, it is possible to reuse the model trained for the generation task by replacing the high dimensional dense output layer with the one having suitable for the classification task dimensionality $512 \times N_c$. There are also two different approaches to training: 1) all model parameters can be updated (fine-tuning), or 2) only the new output layer can be optimized, using the pre-trained part as a feature extractor.

The following pre-training scenarios were used for evaluation of the proposed classifier: 1) no pre-training, 2) with only external data, 3) only target and 4) mixture of the datasets.

Given pre-training with flow labels as first tokens, we evaluated two cases for each of the scenarios: 1) masking the first token, 2) setting a generic first token for all classes that was absent during pre-training.

Taking into the account multi-class nature of the task, the following evaluation metrics were used:

- Accuracy. It is the ratio of correctly classified flow to the total number.
- F1 micro. The harmonic mean of precision and recall, weighted according to the class share in the dataset.

F1 macro. As above, but the classes are weighted equally.

The evaluation results of the proposed model in different scenarios are shown in

Fig. **5**.

When optimizing all the model parameters, the pre-training procedure has a positive impact on the quality of classification, which practically does not change with the increase of data volume during pre-training, and the effect of masking the first token is insignificant. This can be explained by the fact that the model is fully optimized for the target task, and the pre-training stage is a way to initialize the model parameters.

The effect of masking the first token is the opposite when the pre-trained model is used as a feature extractor (i.e., the pre-trained part is not optimized when training the classifier) for a linear classifier. Now the masking improves the metrics by about 2%, but compared to the fully optimized models, the best result drops on average by 4%.

The model pre-trained exclusively on the target data demonstrates the best result (~0.86), whereas the use of an external dataset reduces the quality of the features. In the absence of any pre-training, the model becomes actually linear and shows expectedly poor performance.

We also performed a comparative analysis of the proposed model (with all parameters updated during training) with the ensemble algorithm Random Forest, which is widely used in network traffic classification tasks and shows good results [17].

We use the implementation of the algorithm from the scikit-learn library [18] with the default configuration and number of trees equal to 100.

As shown in Fig. 6, Random Forest has slightly inferior performance to transformer without pre-training, and the gap increases, reaching 4 points on F1 macro when using generative pre-training.

The pre-training procedure (even without class-specific first tokens) allows to train effective feature extractors. The latter can be combined with such non-parametric algorithms as K-NN to facilitate training the classifiers by several examples (few-shot learning), or with single-class classification methods to create detectors of malicious traffic.

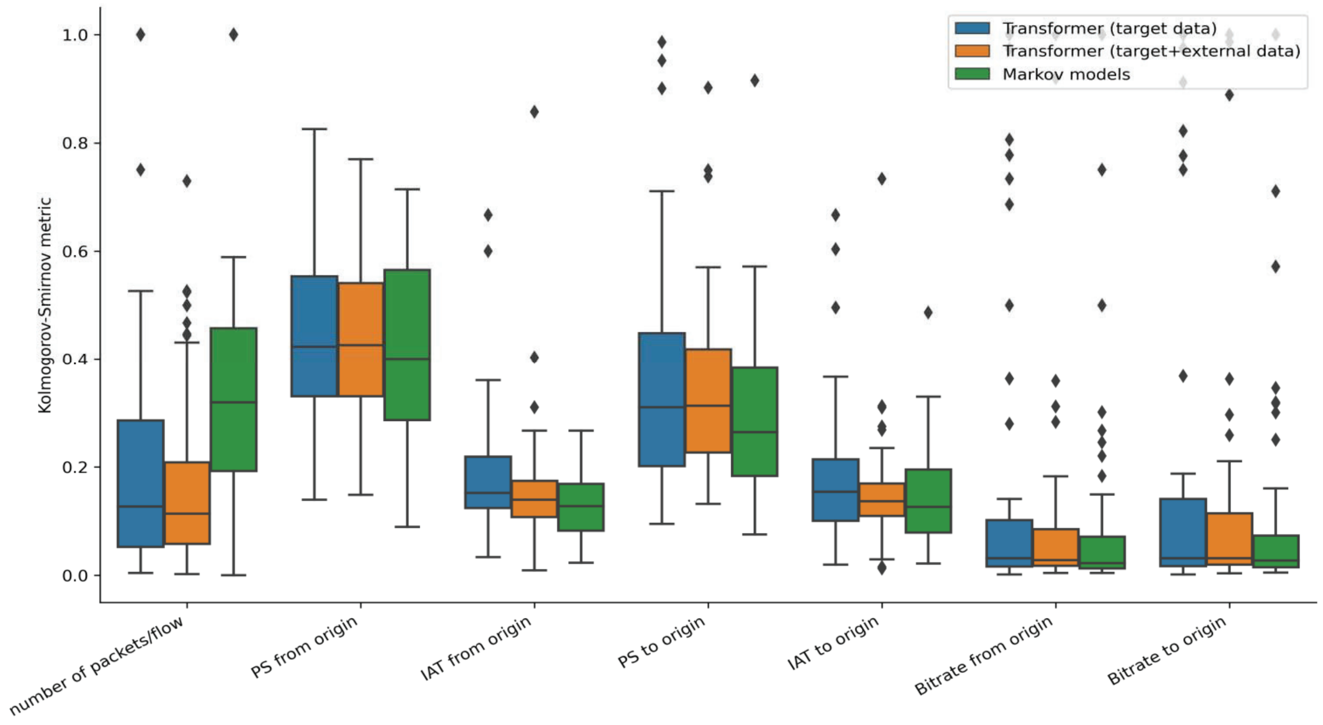

Fig. 4. Box-plots for Kolmogorov-Smirnov metric, measuring distances between original and generated parameters

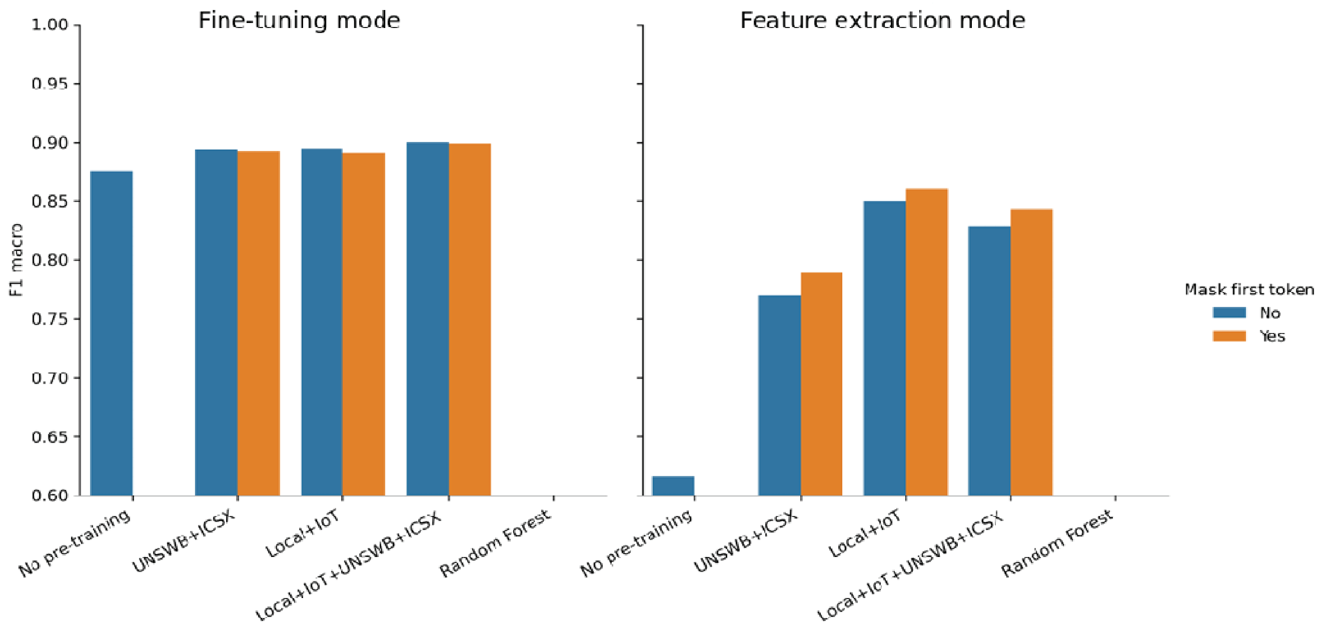

Fig. 5. F1 macro performance of the proposed classifier in different training scenarios

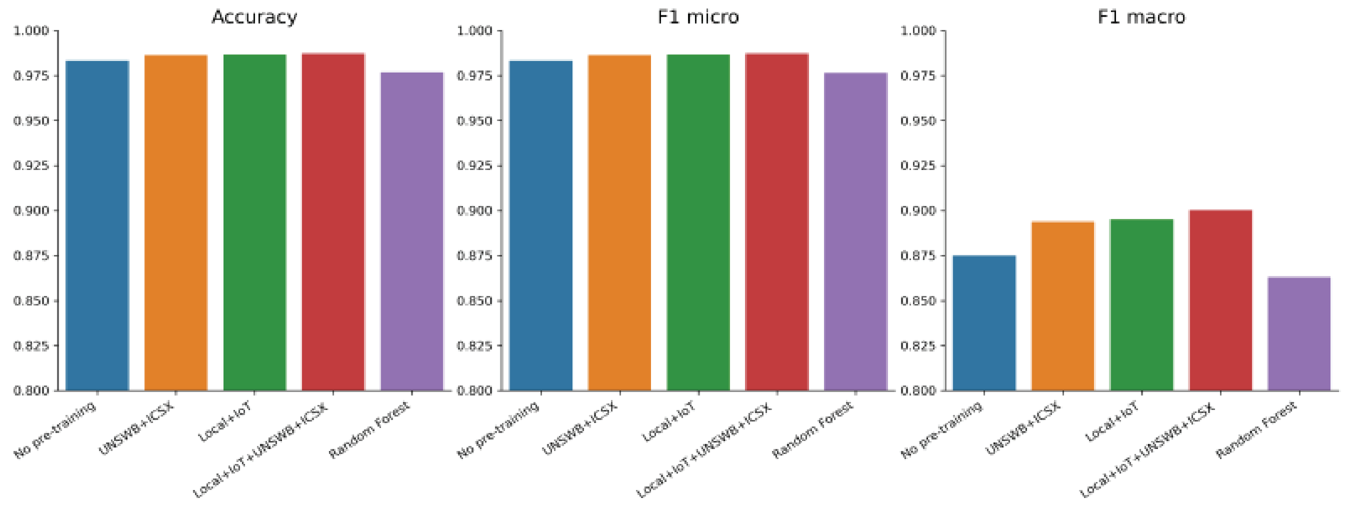

Fig. 6. Classification performance of fine-tuned models in comparison with Random Forest

Conclusions

In this paper, we proposed an approach to build neural network models based on the Transformer architecture for network flow classification and generation tasks, when only packet sizes and inter-packet intervals are used as features. For conversion of the initial two-dimensional feature space into one-dimensional discrete one, we implemented an option based on K-Means clustering. It was shown that it is possible to train a single model for different traffic classes when the first token of a flow sequence was class-specific. The quality of generated traffic was on par with Markov-based approach, where models were trained on each class separately. Moreover, augmenting training set with data from another domain had a positive impact on the quality of generated traffic.

As for the classification task, experiments showed positive impact of generative pre-training for cases with full and partial parameter optimization. It was found important to mask the first token of the input sequence when pre-trained models were used as feature extractors and trained in class-aware mode. Moreover, fine-tuned classifiers outperformed Random Forest model by objective metrics. All in all, the proposed approach allows to create models for generation and classification of network traffic with better or comparable to selected traditional methods quality.

References

1. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781. 2013.

2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. Attention Is All You Need. arXiv:1706.03762, 2017.

3. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. Language Models are Unsupervised Multitask Learners, 2019.

4. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *NeurIPS*. 2019.

5. Devlin, J., Chang, M., Lee, K., & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*. 2019.

6. Murgia, A., Ghidini, G., Emmons, S.P., & Bellavista, P. Lightweight Internet Traffic Classification: A Subject-Based Solution with Word Embeddings. *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2016. pp. 1-8.

7. Liu, C., He, L., Xiong, G., Cao, Z., & Li, Z. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. *IEEE INFOCOM 2019 – IEEE Conference on Computer Communications*, 2019. pp. 1171-1179.

8. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv:1910.01108. 2019.

9. Michel, P., Levy, O., & Neubig, G. Are Sixteen Heads Really Better than One? ArXiv:1905.10650. 2019.

10. NFStream: Flexible Network Data Analysis Framework URL: https://www.nfstream.org/ (10.09.2020).

11. Moustafa, Nour, and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS)*, 2015. pp. 1-6.

12. Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani. Characterization of Encrypted and VPN Traffic Using Time-Related Features. *2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*. 2016.

13. Sivanathan, A., Sherratt, D., Gharakheili, H., Radford, A., Wijenayake, C., Vishwanath, A., & Sivaraman, V. Characterizing and classifying IoT traffic in smart cities and campuses. *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017. pp. 559-564.

14. Network traffic classifier based on machine learning algorithms. Github: [2020]. URL: https://github.com/RadionBik/ML-based-network-traffic-classifier (23.09.2020).

15. Kostin D.V., Sheluhin O.I. Comparison of machine learning algorithms for encrypted traffic classification. *T-Comm*, Vol. 10, No. 9, 2016. pp. 43-52.

16. Molnár, S., Megyesi, P., & Szabó, G. How to validate traffic generators? *2013 IEEE International Conference on Communications Workshops (ICC)*, 2013. pp. 1340-1344.

17. Sivanathan, A., Gharakheili, H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., & Sivaraman, V.. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, No. 18, 2019. pp. 1745-1759.

18. Pedregos F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. Scikit-learn: Machine Learning in Python. ArXiv:1201.0490. 2011.

ГЕНЕРАТИВНЫЕ МОДЕЛИ НА БАЗЕ НЕЙРОСЕТЕВОЙ АРХИТЕКТУРЫ TRANSFORMER ДЛЯ КЛАССИФИКАЦИИ И МОДЕЛИРОВАНИЯ СЕТЕВОГО ТРАФИКА

Бикмухамедов Радион Фаритович, *ООО "М5"; Казанский национальный исследовательский технический университет им. А.Н. Туполева - КАИ, г. Казань, Россия,* ***radion.bikmukhamedov@pm.me***
Надеев Адель Фирадович, *Казанский национальный исследовательский технический университет им. А.Н. Туполева - КАИ, г. Казань, Россия,* ***afnadeev@kai.ru***

Аннотация
В статье разработана генеративная модель сетевых потоков на базе нейросетевой архитектуры Transformer. В качестве входных признаков для модели использовались размеры пакетов и межпакетные интервалы в потоке, которые были масштабированы и кластеризованы, что позволило сделать единообразными входные данные для задач классификации и генерации. Модель может обучаться в двух режимах: (i) как авто-регрессионная модель для генерации трафика, где первый символ обучающей последовательности кодирует тип моделируемого протокола/приложения, (ii) как классификатор сетевых потоков. Результаты оценки качества сгенерированного трафика показали, что предложенный подход демонстрирует качество сравнимое с моделью на основе Марковской цепи первого порядка, которая обучалась на каждом классе по отдельности. Качество оценивалось усредненной по классам метрике Колмогорова-Смирнова, показывающая расстояния полученных эмпирических распределений к исходным для следующих параметров: межпакетный интервал, размера пакета, пропускная способность, число пакетов в направлениях от и к источнику. Было также показано, что при обучении генератора добавление трафика из другого домена способно улучшить итоговое качество. Результаты экспериментов также показали позитивный эффект использования предварительно обученной модели для задачи классификации. Так, при использовании генератора как экстрактора признаков для линейного алгоритма, качество классификации приближается к результатам модели на основе алгоритма Случайный Лес, а при оптимизации всех параметров модели, качество по метрике F1-макро повышается на 5%, опережая ансамблевый алгоритм в среднем на 4%.

Ключевые слова: *transformer, классификатор трафика, генерация трафика, нейронная сеть, случайный лес, Марковская цепь, K-Means, перенос знаний.*

Литература

1. *Mikolov T., Chen K., Corrado G., Dean J.* Efficient Estimation of Word Representations in Vector Space // ArXiv:1301.3781. 2013.
2. aswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. Attention Is All You Need // arXiv:1706.03762, 2017.
3. *adford A., Wu J., Child R., Luan D., Amodei D., & Sutskever I.* Language Models are Unsupervised Multitask Learners, 2019.
4. *Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q.V.* XLNet: Generalized Autoregressive Pretraining for Language Understanding // NeurIPS. 2019.
5. *Devlin J., Chang M., Lee K., & Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // NAACL-HLT. 2019.
6. *Murgia A., Ghidini G., Emmons S.P., & Bellavista P.* Lightweight Internet Traffic Classification: A Subject-Based Solution with Word Embeddings // 2016 IEEE International Conference on Smart Computing (SMARTCOMP), 2016. pp. 1-8.
7. *Liu C., He L., Xiong G., Cao Z., & Li Z.* FS-Net: A Flow Sequence Network For Encrypted Traffic Classification // IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019. pp. 1171-1179.
8. *Sanh V., Debut L., Chaumond J., & Wolf T.* DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // ArXiv:1910.01108. 2019.9.
9. *Michel P., Levy O., & Neubig G.* Are Sixteen Heads Really Better than One? // ArXiv:1905.10650. 2019.
10. NFStream: Flexible Network Data Analysis Framework URL: https://www.nfstream.org/ та обращения: 10.09.2020).
11. *Moustafa, Nour, and Jill Slay.* UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) // Military Communications and Information Systems Conference (MilCIS), 2015. pp. 1-6.
12. *Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani.* Characterization of Encrypted and VPN Traffic Using Time-Related Features // 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016). 2016.
13. *Sivanathan A., Sherratt D., Gharakheili H., Radford A., Wijenayake C., Vishwanath A., & Sivaraman V.* Characterizing and classifying IoT traffic in smart cities and campuses // 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017. pp. 559-564.
14. Network traffic classifier based on machine learning algorithms // Github: [2020]. URL: https://github.com/RadionBik/ML- ed-network-traffic-classifier (Дата обращения: 23.09.2020).
15. *Kostin D.V., Sheluhin O.I.* Comparison of machine learning algorithms for encrypted traffic classification // T-Comm: Телекоммуникации и транспорт, Vol. 10, No. 9, 2016. pp. 43-52.
16. *Molner S., Megyesi P., & Szaby G.* How to validate traffic generators? // 2013 IEEE International Conference on Communications Workshops (ICC), 2013. pp. 1340-1344.
17. *Sivanathan A., Gharakheili H., Loi F., Radford A., Wijenayake C., Vishwanath A., & Sivaraman V.* Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics // IEEE Transactions on Mobile Computing, No. 18, 2019. pp. 1745-1759.
18. *Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Louppe G., Prettenhofer P., Weiss R., Dubourg V., VanderPlas J., Passos A., Cournapeau D., Brucher M., Perrot M., & Duchesnay E.* Scikit-learn: Machine Learning in Python // ArXiv:1201.0490. 2011.