Robust LLM-Based Website Fingerprinting under Dynamic Real-World Conditions

Xiyuan Zhao
Tsinghua University
Beijing, China
zhaoxy23@mails.tsinghua.edu.cn

Xinhao Deng
Ant Group
Hangzhou, China
Tsinghua University
Beijing, China
dengxinhao@mail.tsinghua.edu.cn

Tianyu Cui
Zhongguancun Laboratory
Beijing, China
cuity@zgclab.edu.cn

Yixiang Zhang
Tsinghua University
Beijing, China
zhangyix24@mails.tsinghua.edu.cn

Ke Xu
Tsinghua University
Beijing, China
Zhongguancun Laboratory
Beijing, China
xuke@tsinghua.edu.cn

Qi Li*
Tsinghua University
Beijing, China
Zhongguancun Laboratory
Beijing, China
qli01@tsinghua.edu.cn

Abstract

Website Fingerprinting (WF) attacks aim to infer the websites visited by Tor users by analyzing patterns in encrypted network traffic. However, most existing WF attacks are evaluated on traffic collected in controlled environments with fixed configurations, failing to reflect the complexity and variability of real-world conditions. In practice, traffic is far more dynamic and diverse due to heterogeneous network conditions, the large number of subpages within individual websites, and continuous evolution of website content. These factors increase intra-class variability and induce temporal feature drift, which ultimately degrades the long-term effectiveness of existing attacks. In this paper, we propose TraVerse, an LLM-based representation learning framework designed to achieve robust WF attacks under real-world conditions. TraVerse applies architectural adaptation and large-scale fine-tuning on diverse unlabeled traffic to learn generalizable and resilient representations that remain effective in dynamic and evolving environments. Furthermore, TraVerse integrates a lightweight classifier atop the LLM-derived representations, enabling accurate website identification and efficient few-shot adaptation with minimal model updates. We prototype TraVerse and conduct comprehensive evaluations using real-user traffic. Experimental results show that TraVerse improves Accuracy@3 by an average of 176.3% and weighted F1 by 343.3% over state-of-the-art baselines, while maintaining strong performance throughout a three-month longitudinal evaluation.

CCS Concepts

• **Security and privacy → Network security**.

Keywords

Tor; privacy; website fingerprinting; LLM fine-tuning; representation learning

ACM Reference Format:
Xiyuan Zhao, Xinhao Deng, Tianyu Cui, Yixiang Zhang, Ke Xu, and Qi Li. 2026. Robust LLM-Based Website Fingerprinting under Dynamic Real-World Conditions. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792288

1 Introduction

Tor [15] provides anonymity through randomized multi-hop routing and layered encryption. However, it remains vulnerable to Website Fingerprinting (WF) attacks [1, 2, 10–12, 28, 41, 44, 45, 55], where adversaries analyze encrypted traffic patterns to identify visited websites. These attacks extract meta-features like packet directions, sizes, and timings from traffic traces and train models to perform website classification.

Recent WF attacks are primarily evaluated under fixed settings, which fail to reflect the variability of real-world traffic [26]. In practice, network traffic exhibits pronounced intra-class variability and evolves continuously. While state-of-the-art methods can learn discriminative representations from stable datasets, their performance deteriorates in dynamic conditions, where heterogeneous traffic patterns and website evolution cause feature drift and poor adaptability [35, 49]. Although recent studies have attempted to address these issues [1, 6], notable limitations persist. For example, the data augmentation in [1] is environment-specific and lacks generality, whereas the online adaptation approach in [6] continuously retrains classifiers at the exit side but suffers from distortions between the entry and exit relays, resulting in degraded robustness [6, 27].

In this work, we focus on developing practical WF attacks tailored for real-world conditions, where network traffic is highly dynamic and diverse due to heterogeneous network environments, numerous subpages within individual websites, and continuously evolving website content. We introduce a Large Language Model (LLM)-based representation learning paradigm for robust website fingerprinting. LLMs have demonstrated strong capabilities

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WWW '26, Dubai, United Arab Emirates
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792288

in pattern mining and contextual modeling [7, 50], making them particularly well-suited for capturing complex dependencies in traffic sequences. Furthermore, their powerful generalization ability supports learning robust representations from heterogeneous and evolving traffic, a property that is critical for maintaining attack performance under real-world variability. Building upon these strengths, we leverage LLMs to derive generalizable traffic representations that sustain high identification accuracy and stable performance in dynamic, real-world environments.

However, leveraging LLMs for WF attacks presents several key challenges. First, most decoder-based LLMs adopt causal attention, which restricts each token to attend only to its predecessors and thus struggles to capture global dependencies across an entire traffic sequence. Second, pretrained LLMs lack traffic-domain knowledge, limiting their ability to learn robust and discriminative representations for dynamic real-world traffic. Third, real-world website identification requires continual adaptation to evolving traffic distributions, yet periodic LLM fine-tuning is computationally expensive and time-consuming, making long-term adaptation impractical.

To this end, we propose TraVerse, a robust and realistic WF framework that customizes LLMs for accurate website identification. Since real-world traffic is inherently dynamic and heterogeneous, with substantial variability even within the same website, TraVerse adapts the LLM architecture to capture robust and discriminative traffic features, enabling richer sequence understanding to support resilient fingerprinting. Moreover, to cope with temporal drift in traffic characteristics, TraVerse combines large-scale unsupervised fine-tuning for generalizable representations with lightweight adaptation that avoids frequent LLM fine-tuning, sustaining performance throughout long-term deployment.

To realize these objectives, TraVerse comprises three key components. First, we introduce a context-aware traffic profiling mechanism based on bidirectional attention to model full-sequence dependencies and facilitate the extraction of global patterns that remain stable under heterogeneous network conditions. Second, we conduct large-scale fine-tuning on massive unlabeled traffic data to imbue the model with domain-specific traffic knowledge, allowing TraVerse to learn generalizable representations across diverse traffic patterns. Finally, we perform few-shot supervised classification using a lightweight classifier built atop the LLM-derived representations to achieve accurate website identification. This classifier requires only a small number of labeled samples for adaptation, enabling efficient updates and long-term robustness under dynamic and evolving network environments.

In summary, this work makes the following contributions:

- We propose TraVerse, a WF attack framework that leverages LLM-based traffic representation learning to enable effective website identification under realistic and dynamic browsing scenarios.
- We adapt the LLM architecture and conduct large-scale unsupervised fine-tuning on massive traffic data, allowing the model to learn robust and generalizable traffic representations that remain effective under dynamic and evolving network conditions.
- We implement and evaluate TraVerse on real-user traffic in both the closed-world and open-world settings. Results from a three-month study show that TraVerse maintains long-term robustness and consistently outperforms existing methods.

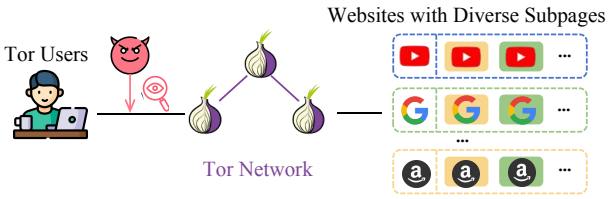


Figure 1: The threat model of TraVerse.

2 Background

WF attacks aim to identify the websites visited by users, even when traffic is encrypted on the Tor network. Since encryption prevents direct access to website content, attackers must instead infer websites by analyzing distinctive traffic patterns such as packet timings, direction sequences, and burst structures. Early works employ traditional classifiers like Random Forests and Support Vector Machines to distinguish websites [23, 36, 47], while recent works take advantage of deep learning to automate feature extraction and significantly improve attack performance [42, 44]. Building on these advances, research has extended WF attacks to more realistic settings, such as multi-tab browsing [8, 12, 21, 28, 51, 53], diverse network conditions [1], and fine-grained webpage identification [55]. In addition, efforts have focused on practical challenges, including limited training data [2, 35, 45], incomplete page loading [11, 13], and robustness against traffic obfuscation defenses [13, 17, 41, 43].

Despite these efforts, the majority of existing methods are still evaluated on simulated datasets with relatively homogeneous traffic characteristics. When deployed in real-world environments, their identification capability often deteriorates due to dynamic network conditions, high subpage diversity, and continuous feature drift. In this work, we focus on addressing real-world website identification, where traffic patterns are inherently diverse, evolving, and challenging to model using conventional approaches.

3 Threat Model

Figure 1 illustrates the threat model of TraVerse. Consistent with previous works [12, 28, 44, 55], we consider a passive attacker who has the ability to eavesdrop and collect traffic packets on the link between the user and the Tor entry relay. The attacker cannot decrypt, modify, or drop these packets, but can analyze traffic patterns to infer the websites being accessed.

Our threat model aims to reflect the complexities of real-world WF attacks, characterized by three key aspects. (i) Dynamic network conditions: accesses may occur from different locations and under diverse network conditions. Real-world fluctuations like packet delays, reordering, or loss introduce variations in traffic characteristics. (ii) Subpage diversity: users can visit arbitrary subpages within a target website, rather than being restricted to the homepage, resulting in high intra-website variability. (iii) Pattern drift: websites are constantly updated with new content and features, causing temporal drift in traffic patterns.

Following prior works [12, 28, 41, 44, 45, 55], we evaluate TraVerse in both the closed-world and open-world settings. In the closed-world setting, users browse a fixed set of monitored websites whose traffic is observed during training. In contrast, the

open-world setting allows users to access a wide range of unmonitored websites unseen during training, requiring the attacker to distinguish monitored visits from unmonitored background traffic.

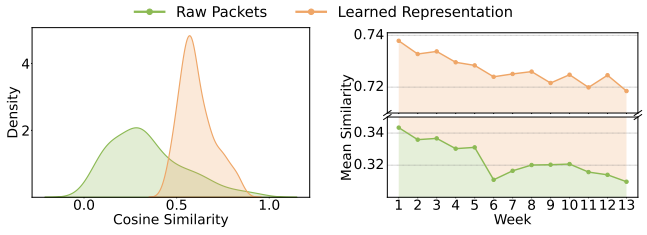
4 Design of TraVerse

In this section, we first present the key observation for the design of TraVerse, followed by a high-level overview of the framework.

4.1 Key Observation

Real-world traffic exhibits substantially greater variability than the controlled traces commonly used in prior studies. In particular, traces collected from real users display high intra-class diversity. We evaluate this diversity using a real-user traffic dataset spanning 13 weeks. As shown in Figure 2a, the intra-class cosine similarity computed from raw-packet direction sequences is skewed toward lower values, with most values concentrated around 0.3. We further analyze temporal dynamics by calculating the similarity of each website's traffic from Weeks 1–13 relative to Week 1. Figure 2b shows that the similarity displays a clear downward trend, reflecting temporal feature drift and the continuous evolution of traffic patterns. The fluctuation observed in Week 6 may be attributed to the sampling strategy during dataset construction. These findings underscore the necessity of learning robust traffic representations that can generalize across diverse and evolving conditions.

To address these challenges, we develop a traffic representation learning framework grounded in extensive LLM fine-tuning to learn robust representations. After applying traffic representation learning, the intra-class similarity becomes concentrated around 0.6, indicating enhanced consistency within individual websites. Furthermore, the cross-week similarity remains consistently higher after representation, demonstrating improved temporal robustness. Such improvements confirm that the learned representations are both robust and generalizable, enabling accurate and resilient website identification under dynamic real-world conditions.



(a) Intra-website similarity **(b) Similarity trend across weeks**

Figure 2: Analysis of representation learning on real-user traffic.

4.2 Design Overview

In this work, we propose TraVerse, a framework that leverages LLM-based traffic representation learning via model architectural adaptation and large-scale fine-tuning to enable practical WF attacks in real-world settings. Through extensive unsupervised fine-tuning on unlabeled traffic, TraVerse equips the adapted LLM with traffic-domain knowledge, allowing it to capture global, long-range dependencies across diverse browsing conditions and generate robust, highly generalizable representations for real-world traffic.

As illustrated in Figure 3, TraVerse comprises three main components. First, we propose a context-aware traffic profiling mechanism that captures global traffic patterns and improves robustness under dynamic network conditions. Specifically, we adapt the LLM architecture by introducing bidirectional attention, enabling comprehensive context aggregation across the entire traffic sequence. Second, we conduct unsupervised representation learning by extensively fine-tuning the adapted LLM on a large-scale traffic corpus using a masked pattern reconstruction objective. This objective injects domain-specific knowledge and strengthens the model's capacity to represent diverse traffic patterns. Despite the diversity and continual evolution of real-world traffic, the fine-tuned LLM produces robust and generalizable representations. Finally, leveraging the representations generated by the LLM, TraVerse performs few-shot supervised classification using a lightweight classifier for accurate website identification. By restricting retraining to this classifier while keeping the LLM frozen, TraVerse can efficiently adapt to distribution shifts with minimal labeled samples, ensuring long-term robustness under real-world drift.

5 Design Details

In this section, we present the details of TraVerse, comprising three key components: Context-Aware Traffic Profiling, Unsupervised Representation Learning and Few-Shot Supervised Classification.

5.1 Context-Aware Traffic Profiling

TraVerse leverages a context-aware traffic profiling mechanism to improve robustness under dynamic network conditions. Specifically, we adapt large decoder-based LLMs [3, 20, 52] for traffic profiling by modifying their attention mechanisms to enable holistic contextual modeling over traffic sequences.

In real-world environments, traffic patterns often exhibit local variations caused by latency fluctuations, packet reordering, and other transient network dynamics. Such variations disrupt the consistency of traffic patterns and degrade models that primarily rely on short-range dependencies. Therefore, we employ LLMs to capture long-range and global patterns in traffic sequences. However, decoder-based LLMs are designed for generation tasks and typically employ causal attention, where each token embedding attends solely to preceding tokens. This unidirectional attention can lead to incomplete profiling when applied to dynamic traffic sequences.

To address this limitation, TraVerse modifies the LLM architecture by introducing bidirectional attention, allowing each token to attend to both preceding and succeeding tokens over the sequence. This bidirectional attention, widely used in encoder-based models for sequence understanding, facilitates full-sequence interaction and comprehensive context aggregation throughout the traffic trace. As a result, TraVerse extracts global patterns that remain robust despite local variations caused by dynamic perturbations.

To enable context-aware traffic profiling, TraVerse constructs instruction-guided prompts that combine a natural language instruction with traffic feature inputs. The instruction specifies that the subsequent sequence represents a website browsing session, guiding the model to interpret the trace in the intended context.

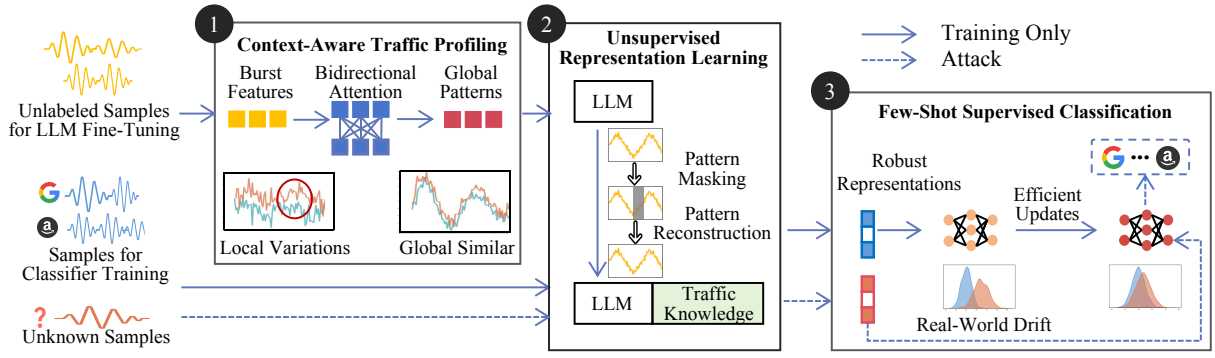


Figure 3: Design overview of TraVerse.

Traffic features are extracted as burst sequences, which capture higher-level semantics associated with website resources, such as text and images [1], therefore preserving salient structural information critical for traffic modeling. A burst is defined as consecutive packets in the same direction, represented by its length and direction. Finally, the instruction and burst inputs are concatenated into a unified input using the format: `<flow>: <burst sequence>`. Prompt templates are provided in Appendix B.

After prompt construction, the prompt is tokenized into a token sequence of length l: $L = (L_1, L_2, \ldots, L_l)$. The LLM then transforms this token sequence into token embeddings $T = [T_1, T_2, \ldots, T_l] \in \mathbb{R}^{l \times d}$, where d denotes the embedding dimension. A linear projection is subsequently applied to obtain the query, key, and value matrices:

$$Q = TW_Q, \quad K = TW_K, \quad V = TW_V, \tag{1}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_o}$ are trainable matrices and $Q, K, V \in \mathbb{R}^{l \times d_o}$ are the projected representations, with d_o denoting the projection dimension. The bidirectional attention score between token i and token j is then formally defined as:

$$\alpha_{ij} = \frac{\exp\left(\frac{Q_i K_j^\top}{\sqrt{d_o}}\right)}{\sum_{k=1}^{l} \exp\left(\frac{Q_i K_k^\top}{\sqrt{d_o}}\right)}, \tag{2}$$

where the denominator aggregates over all tokens in the sequence, ensuring full-sequence contextual aggregation. After the stacked attention layers, TraVerse applies position-weighted pooling over the resulting feature token embeddings to aggregate information and produce the final traffic representation. Specifically, earlier tokens are assigned higher weights because the front of a traffic trace typically contains more discriminative features [18].

Overall, the combination of bidirectional attention and position-weighted aggregation enables context-aware traffic profiling that captures global traffic patterns and remains resilient to local variations under dynamic network conditions.

5.2 Unsupervised Representation Learning

TraVerse leverages unsupervised representation learning to better model diverse traffic patterns. In practice, users access not only index pages but also numerous subpages, increasing intra-website variability. Moreover, websites continuously evolve, causing feature drift that degrades model stability over time. By fine-tuning the LLM on large-scale unlabeled traffic traces, TraVerse exploits the strong generalization capability of LLMs to capture diverse patterns and generate robust and transferable representations that remain effective under dynamic real-world conditions.

Figure 4 illustrates the details of the unsupervised representation learning. Specifically, TraVerse adopts a masked pattern reconstruction objective by randomly masking a fraction of pattern tokens and training the model to predict the masked pieces by leveraging contextual information in the sequence. We first construct prompts from the traffic corpus by combining a natural language description with traffic features, then tokenize the prompts and apply masking. The adapted LLM, equipped with bidirectional attention, produces contextualized representations for all positions. We train the model to predict masked tokens in a shifted decoder-style manner, where each target token is predicted from the contextualized representation at the previous position. This preserves the decoder-based pretraining paradigm while exploiting bidirectional context, enhancing the model's ability to interpret traffic patterns and facilitating domain adaptation.

Formally, let $M \subseteq \{1, \ldots, l\}$ denote the set of masked positions. The training objective is to maximize the likelihood of recovering the tokens L_i at masked positions $i \in M$. The loss is defined as:

$$\mathcal{L}_{\text{recon}} = -\sum_{i \in M} \log P(L_i \mid h_{i-1}; \theta), \tag{3}$$

where θ denotes the model parameters and h_{i-1} is the context-aware representation at position $i-1$ produced by multiple bidirectional attention layers over the masked input sequence. The masked tokens are replaced with a special token.

We adopt Low-Rank Adaptation (LoRA) [24], which introduces low-rank trainable matrices into the transformer layers for parameter-efficient fine-tuning. This significantly reduces the number of trainable parameters and memory costs, enabling efficient adaptation of large pretrained LLMs to traffic-specific tasks without full-scale fine-tuning. The LoRA fine-tuning is detailed in Appendix A.

By fine-tuning on extensive unlabeled traffic, TraVerse learns generalizable representations from diverse browsing conditions, forming a strong foundation for accurate website identification.

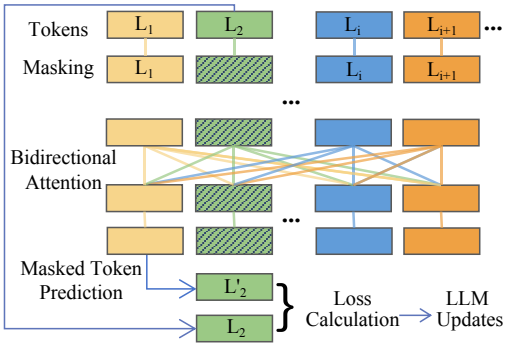


Figure 4: Details of the unsupervised representation learning.

5.3 Few-Shot Supervised Classification

TraVerse employs supervised classification on top of the LLM using few-shot labeled samples to achieve accurate website identification. While unsupervised fine-tuning enables generalization across diverse traffic patterns, real-world website identification further requires long-term adaptability to evolving traffic distributions. Leveraging robust LLM-generated representations, TraVerse trains a lightweight downstream classifier with a small labeled set, enabling efficient updates without frequent LLM fine-tuning.

Specifically, TraVerse constructs prompts combining instructions and traffic features, obtains representations from the fine-tuned LLM, and feeds them into a lightweight classifier for website identification. The classifier consists of two fully connected layers with ReLU activation, and incorporates batch normalization to stabilize training. During deployment, the trained classifier predicts the corresponding website for each traffic sample and can be updated using a small labeled set when the traffic characteristics shift.

Formally, given the input matrix $X \in \mathbb{R}^{N \times d_e}$, where N denotes the number of samples and d_e denotes the dimension of the LLM generated representations, the classifier is defined as:

$$\tilde{Y} = \sigma(\mathrm{BN}(XW_1))W_2 \tag{4}$$

where $W_1 \in \mathbb{R}^{d_e \times d_h}$ and $W_2 \in \mathbb{R}^{d_h \times W}$ are the weight matrices of the linear layers, d_h is the hidden dimension, W is the number of websites, $BN(\cdot)$ denotes batch normalization, and $\sigma(\cdot)$ is the ReLU activation function. During website identification, the classifier outputs the website with the highest predicted probability.

By decoupling representation learning from classification, TraVerse fully capitalizes on the robust representations generated by the LLM while ensuring high accuracy and adaptability for website identification under dynamic real-world conditions. This lightweight classifier supports efficient few-shot updates, allowing rapid adaptation to new traffic distributions and evolving environments.

6 Evaluation

In this section, we evaluate the performance of TraVerse using public datasets collected from real user browsing, and compare it with state-of-the-art methods.

6.1 Experimental Setup

Implementation. We implement TraVerse using Python 3.10 and Transformers 4.44.2 and adopt Llama3-8B [20] as the backbone model. We fine-tune the LLM on a single NVIDIA GeForce RTX 4090 GPU, requiring approximately 65 GPU hours and 21 GB of peak memory for 30k steps on 200k traces. The default hyperparameter settings and prompt templates are provided in Appendix B.

Datasets. We evaluate TraVerse using GTT23 [26], a real-user traffic dataset that offers a more realistic and challenging setting than prior datasets collected under fixed settings [33, 35, 38, 42, 44]. It reflects dynamic network conditions, includes traffic beyond index pages, and spans 13 weeks to capture temporal drift in traffic patterns.

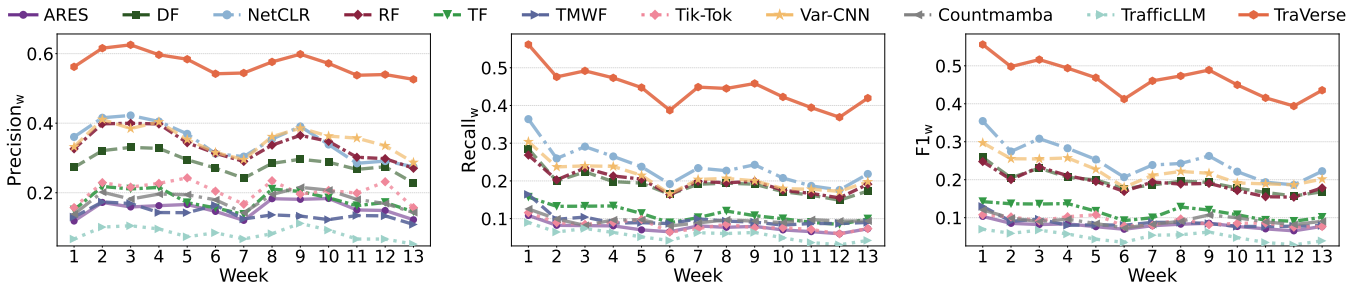
The dataset is collected at the exit relay, which differs from our threat model. However, due to Tor's design, collecting large-scale labeled entry-side traces from real users is difficult, and [27] demonstrates that training and testing on exit traces yields performance comparable to training and testing on transduced entry traces. This establishes GTT23 as a reliable benchmark for real-world WF evaluation without the cost of traffic transduction. For real-world deployment, TraVerse can be combined with the transduction methods in [25, 27] to approximate entry-side inputs.

We select 200,000 unlabeled traces for LLM fine-tuning to learn diverse traffic patterns, filtering out traces with fewer than 20 packets. For evaluation, we establish a monitored set consisting of 200 websites with sufficient samples, where these 200 websites are disjoint from those used for LLM fine-tuning. This separation creates a more challenging and realistic scenario, simulating an attacker with limited prior knowledge of the visited websites.

We use two datasets, $\mathbf{GTT_{CW}}$ and $\mathbf{GTT_{OW}}$, to evaluate TraVerse in both the closed-world and open-world settings. We train and validate on Week 1 and test on Weeks 1–13 to assess robustness under temporal drift. For $\mathbf{GTT_{OW}}$, we additionally include an unmonitored class consisting of traffic from disjoint unmonitored websites, using 2,000 and 100 unmonitored traces for training and validation, and 5,000 unmonitored traces per week for testing, while retaining the monitored samples. We ensure no unmonitored website overlap across LLM fine-tuning, classifier training, and testing. We also synthesize a 2-tab concurrent browsing dataset $\mathbf{GTT_{MT}}$ from GTT23 to evaluate TraVerse under the multi-tab setting since no real-user multi-tab dataset is publicly available. Details of the datasets are shown in Appendix C.

Baselines. We select nine WF attacks as baselines [1, 2, 12, 13, 28, 41, 43–45], where these baselines cover a representative range of WF attacks. For a fair comparison, all WF baselines are trained with a maximum of 100 epochs, and the best checkpoint is selected based on the validation performance. In addition, we include TrafficLLM [7] as an LLM-based baseline. TrafficLLM is trained for 5 epochs and uses the same prompt templates as TraVerse. Baseline details are provided in Appendix D.

Metrics. For the closed-world setting, following prior works [1, 11, 41, 44], we evaluate methods using Accuracy@3, $\mathrm{Precision_w}$, $\mathrm{Recall_w}$, and $\mathrm{F1_w}$. Accuracy@3 measures whether the ground-truth label appears among the top three predictions, while $\mathrm{Precision_w}$, $\mathrm{Recall_w}$, and $\mathrm{F1_w}$ jointly assess the correctness and completeness under real-world data distributions, where each class is weighted by its proportion in the dataset. For the open-world setting, we



(a) Precision$_w$ (b) Recall$_w$ (c) F1$_w$

Figure 5: Performance of various methods in the closed-world setting.

Table 1: Accuracy@3 of various methods in the closed-world setting.

Method	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
ARES	0.206	0.166	0.164	0.160	0.150	0.125	0.150	0.159	0.172	0.154	0.140	0.130	0.156
DF	0.419	0.331	0.350	0.334	0.324	0.266	0.306	0.337	0.311	0.279	0.277	0.256	0.287
NetCLR	0.472	0.367	0.396	0.383	0.363	0.288	0.336	0.355	0.348	0.328	0.307	0.274	0.334
RF	0.414	0.339	0.357	0.362	0.371	0.277	0.327	0.357	0.331	0.312	0.301	0.253	0.324
TF	0.239	0.188	0.191	0.194	0.177	0.143	0.160	0.190	0.168	0.161	0.153	0.152	0.168
TMWF	0.288	0.202	0.195	0.189	0.189	0.189	0.198	0.201	0.171	0.164	0.184	0.207	0.193
Tik-Tok	0.204	0.163	0.160	0.176	0.201	0.133	0.139	0.189	0.148	0.153	0.159	0.137	0.161
Var-CNN	0.440	0.358	0.357	0.363	0.341	0.265	0.323	0.337	0.301	0.272	0.289	0.297	0.319
Countmamba	0.205	0.165	0.143	0.154	0.157	0.128	0.143	0.155	0.157	0.147	0.149	0.140	0.145
TraVerse	**0.694**	**0.612**	**0.625**	**0.602**	**0.588**	**0.513**	**0.583**	**0.609**	**0.608**	**0.577**	**0.553**	**0.517**	**0.570**

follow [11, 46] and use r-precision with $r = 5$, where r denotes the ratio of visits to unmonitored versus monitored websites. For the multi-tab setting, following [55], we adopt Recall@10 and AP@2. Detailed definitions of all metrics are provided in Appendix E.

6.2 Closed-World Evaluation

We first evaluate the performance of TraVerse in the closed-world setting, where both training and test traffic originate from monitored websites. We use the closed-world dataset $\mathbf{GTT_{CW}}$ for evaluation, and report Accuracy@3, Precision$_w$, Recall$_w$ and F1$_w$. Note that TrafficLLM does not support computing Accuracy@3, so we do not report this metric for TrafficLLM.

Table 1 and Figure 5 present the results across 13 weeks. Week 1 yields the best performance since the training and test traffic share the highest similarity. In this case, TraVerse is the only method achieving Accuracy@3 close to 0.7, whereas all baselines remain below 0.48. Over time, although performance gradually decreases due to feature drift, TraVerse consistently maintains Accuracy@3 and F1$_w$ above 0.5 and 0.4 for most weeks, demonstrating strong temporal robustness. In general, TraVerse achieves an average Accuracy@3 improvement of 176.3% over the WF baselines across 13 weeks. It further improves Precision$_w$, Recall$_w$, and F1$_w$ by an average of 214.0%, 318.3% and 343.3%, respectively, compared with all baselines. These results confirm that the learned traffic representations generalize well under temporal drift.

We also evaluate TraVerse on $\mathbf{GTT_{MT}}$ under the multi-tab setting, using binary cross-entropy as the loss function. Excluding TF and TrafficLLM, which cannot be directly applied in this scenario, TraVerse improves Recall@10 and AP@2 by 92.8% and 168.4%, demonstrating strong effectiveness under multi-tab browsing.

Remark. TraVerse consistently outperforms all baselines in the closed-world setting, demonstrating resilient website identification under dynamic real-world conditions.

6.3 Open-World Evaluation

We then evaluate the performance of TraVerse in the open-world setting, where users can visit a large number of unmonitored websites. The attacker must accurately classify monitored websites while reliably grouping all unmonitored traffic into a single unmonitored class. Experiments are conducted on the open-world dataset $\mathbf{GTT_{OW}}$, with performance measured by r-precision with $r = 5$.

Table 2 reports the performance across 13 weeks. TraVerse achieves its best r-precision in Week 1 at approximately 0.55 and remains above 0.30 in all subsequent weeks. In contrast, all baselines except NetCLR perform poorly, reaching at most 0.26 in Week 1 and typically dropping below 0.20 afterward. NetCLR benefits from data augmentation and self-supervised learning, but it still underperforms TraVerse. Notably, in Week 13, TraVerse maintains r-precision above 0.30, whereas all other methods fall below 0.20. These results indicate that TraVerse yields robust traffic representations that generalize well in the presence of diverse unmonitored traffic.

Remark. TraVerse consistently outperforms prior methods in the open-world evaluation, where the learned representations enable reliable classification of monitored websites in the presence of unmonitored traffic.

6.4 Adaptability Analysis

In this section, we evaluate the real-world adaptability of TraVerse. We keep the LLM fine-tuned on Week 1 fixed and test classifier

Table 2: r-precision of various methods in the open-world setting.

Method	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
ARES	0.063	0.000	0.008	0.000	0.031	0.053	0.011	0.059	0.029	0.018	0.000	0.000	0.059
DF	0.159	0.194	0.053	0.186	0.179	0.077	0.078	0.251	0.071	0.112	0.192	0.280	0.190
NetCLR	0.414	0.268	0.279	0.218	0.194	0.151	0.234	0.213	0.205	0.215	0.210	0.200	0.174
RF	0.255	0.179	0.183	0.180	0.166	0.130	0.156	0.170	0.160	0.143	0.145	0.128	0.112
TF	0.064	0.058	0.055	0.061	0.034	0.028	0.040	0.050	0.038	0.039	0.030	0.029	0.016
TMWF	0.104	0.063	0.027	0.072	0.093	0.044	0.028	0.109	0.043	0.065	0.115	0.139	0.084
Tik-Tok	0.105	0.170	0.059	0.114	0.154	0.075	0.054	0.205	0.000	0.095	0.189	0.241	0.173
Var-CNN	0.060	0.098	0.025	0.114	0.168	0.071	0.085	0.155	0.019	0.093	0.106	0.156	0.144
Countmamba	0.101	0.012	0.016	0.027	0.008	0.018	0.002	0.007	0.012	0.006	0.013	0.000	0.008
TrafficLLM	0.063	0.026	0.027	0.022	0.019	0.017	0.026	0.019	0.026	0.018	0.017	0.017	0.017
TraVerse	**0.549**	**0.463**	**0.469**	**0.444**	**0.411**	**0.378**	**0.443**	**0.405**	**0.416**	**0.408**	**0.383**	**0.312**	**0.347**

performance on Weeks 2-13. For each week, we split the data into training, validation, and test sets, limiting the training set to fewer than 10 samples per website. Two evaluation settings are considered: (i) Pre-Update, where the classifier trained on Week 1 is directly tested on traffic from Weeks 2–13 without any update; (ii) Post-Update, where the classifier is retrained each week using that week's limited training data and tested on the corresponding test set. We report $\text{Precision}_\text{w}$, Recall_w and F1_w.

As shown in Figure 6, updating the classifier with a modest number of samples yields substantial performance gains. Specifically, $\text{Precision}_\text{w}$, Recall_w and F1_w improve by an average of 10.0%, 27.5%, and 24.2%, respectively. Moreover, the post-update performance is comparable to that of Week 1 reported in Section 6.2, indicating that the LLM-generated representations generalizes well to subsequent weeks. This demonstrates that TraVerse can effectively leverage a small number of labeled samples to retrain the lightweight classifier while keeping the LLM frozen, thereby enhancing adaptability under dynamic conditions. We further evaluate the impact of different sample sizes in Appendix F, where TraVerse maintains high identification accuracy with limited training data.

Remark. The adaptability analysis shows that the LLM-learned representations ensure temporal robustness as traffic evolves. The lightweight classifier supports efficient few-shot updates, enabling long-term adaptability and sustained real-world attack performance.

6.5 Ablation Study

We also conduct an ablation study to analyze the contribution of each design component. The comparison includes the full TraVerse, the variant without bidirectional attention, and the variant without unsupervised fine-tuning. All methods are evaluated on the closed-world dataset $\mathbf{GTT_{CW}}$, and we report Accuracy@3.

As shown in Table 3, the full TraVerse consistently outperforms the ablated variants. Removing bidirectional attention causes a substantial performance drop, with Accuracy@3 falling below 0.47 after Week 3. This underscores the importance of modeling both preceding and subsequent context for comprehensive representations of traffic patterns under dynamic conditions. Likewise, removing unsupervised fine-tuning leads to an even sharper degradation, with Accuracy@3 dropping to below 0.44 after Week 3, indicating that the model fails to learn diverse traffic patterns without large-scale domain adaptation. Overall, the full TraVerse improves by 34.5% and 39.5% over the variants without bidirectional attention and without unsupervised fine-tuning, respectively. These results confirm that bidirectional attention and traffic fine-tuning are complementary to support robust representation learning. We further evaluate TraVerse with different backbone sizes in Appendix G, where reducing the model size causes only modest degradation. This suggests that the performance gains stem from our design rather than relying solely on model scale.

Remark. Both bidirectional attention and unsupervised fine-tuning are indispensable to enable the generation of robust and generalizable representations for real-world website identification.

7 Discussion

Adaptability to Various LLMs. In this work, we adopt Llama3-8B [20] as the backbone model due to resource constraints. However, our framework is model-agnostic, where larger LLM backbones can be incorporated via the same architectural modification and fine-tuning procedure on the traffic-learning objective. Furthermore, TraVerse reduces the overhead of frequent LLM fine-tuning by decoupling LLM-based representation learning from lightweight classifier updates, enabling efficient adaptation to feature drift.

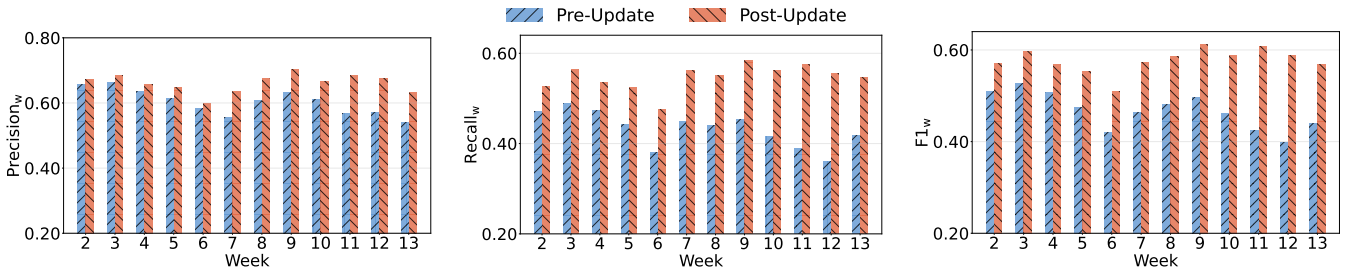
Multi-Tab Attacks. In this work, we use a synthetic multi-tab dataset to evaluate the performance of TraVerse, as no real-user multi-tab dataset is currently available. In this setting, we formulate multi-tab identification as a multi-label classification problem. However, synthetic data may deviate from real multi-tab browsing behavior. An important direction is to evaluate different methods on real-user multi-tab traffic and explore targeted design improvements for handling concurrent traffic from multiple tabs.

Defense Mechanisms. Numerous WF defenses have been proposed to protect user privacy [4, 5, 9, 16, 18, 19, 29, 32, 48]. Existing defenses obfuscate traffic patterns via packet delays or dummy padding, whereas TraVerse is designed to remain adaptable. When complex defenses are deployed, a modest amount of defense-specific traffic can be collected to retrain the lightweight classifier. Since the fine-tuned LLM produces robust and generalizable traffic representations, retraining only the classifier is expected to sustain strong identification performance under evolving adversarial conditions.

8 Related Work

8.1 WF Attacks

WF attacks analyze traffic patterns to identify the websites being accessed. Early approaches rely on hand-crafted fingerprints [23, 36, 47], while subsequent methods leverage deep learning to automate feature extraction and enhance performance [42, 44]. Recent



(a) Precision$_w$ (b) Recall$_w$ (c) F1$_w$

Figure 6: Adaptability analysis results of TraVerse.

Table 3: Ablation study results evaluated using Accuracy@3.

Method	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
w/o Bidirectional Attention	0.513	0.459	0.475	0.454	0.428	0.360	0.405	0.461	0.444	0.427	0.387	0.338	0.384
w/o Unsupervised Fine-Tuning	0.521	0.424	0.444	0.424	0.408	0.345	0.416	0.431	0.429	0.405	0.386	0.353	0.415
Full TraVerse	**0.694**	**0.612**	**0.625**	**0.602**	**0.588**	**0.513**	**0.583**	**0.609**	**0.608**	**0.577**	**0.553**	**0.517**	**0.570**

research focuses on enhancing attack efficiency and robustness. For instance, architectural innovations [2] and improved training strategies [35, 45] are proposed for data-limited settings, while other works strengthen traffic representations to enhance robustness against defenses [41, 43]. Multi-tab identification has been tackled via techniques such as traffic sectioning [8, 51, 53], attention-based mechanisms [12, 21, 28], and multi-label contrastive learning [55]. Efforts to tackle real-world complexities include NetCLR [1], which uses data augmentation to emulate dynamic conditions, and onlineWF [6], which performs continuous model updates using exit-relay traffic. Beyond these, Holmes [11] investigates early-stage attacks via spatio-temporal analysis, and Countmamba [13] unifies early-stage, multi-tab, and robustness settings by combining coarse-grained representations with fine-grained predictions.

TraVerse departs from prior WF attacks by targeting real-world identification on real-user traffic, addressing high intra-class diversity and poor adaptability under dynamic, evolving conditions.

8.2 LLM for Traffic Classification

LLMs have recently demonstrated remarkable capabilities in traffic classification. Existing efforts can be broadly categorized into pretraining-based and fine-tuning-based approaches. Pretraining-based methods use self-supervised learning over packet- or flow-level features to capture intrinsic traffic characteristics. For example, ET-BERT [31], TrafficFormer [56] and PTU [37] optimize traffic representations and design traffic-tailored pretraining tasks based on BERT [14], while NetGPT [34] and TrafficGPT [39] leverage GPT-2 [40] for traffic classification and generation. YaTC [54] and Flow-MAE [22] further employ Masked AutoEncoders (MAE) to achieve effective feature extraction. In contrast, fine-tuning-based approaches adapt pretrained models to downstream traffic tasks via task-specific fine-tuning. TrafficLLM [7] uses prompt tuning for traffic detection and generation, and ETooL [30] integrates flow-graph structures for robust detection under non-IID traffic.

However, existing methods primarily rely on packet payloads or datagrams, which provide limited information under the multi-layer encrypted traffic of Tor. In this work, we extend LLMs to WF using content-agnostic packet-sequence features, enabling robust modeling under the challenges of real-world website fingerprinting.

9 Conclusion

In this work, we develop TraVerse, a framework designed to achieve robust WF attacks under dynamic real-world conditions characterized by diverse network environments, substantial intra-website variability, and continuous temporal drift. To address these challenges, TraVerse customizes the LLM through model architectural adaptation and extensive fine-tuning, to learn robust and generalizable representations for real-world traffic. Moreover, the lightweight classifier design allows efficient updates with a modest number of labeled samples, sustaining long-term effectiveness under evolving traffic distributions. Experiments on 13 weeks of real-user traffic show that TraVerse improves Accuracy@3 by 176.3% over state-of-the-art WF baselines in the closed-world setting.

Acknowledgments

We thank our anonymous reviewers for their helpful comments and feedback. The work is supported in part by National Natural Science Foundation of China under Grant 62132011, National Science Foundation for Distinguished Young Scholars of China under Grant 62425201, the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under Grant 62221003, Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China under Grant JYB2025XDXM114, Beijing-Tianjin-Hebei Natural Science Foundation Cooperation Project under Grant 25JJJJC0003 and Ant Group Postdoctoral Programme.

References

[1] Alireza Bahramali, Ardavan Bozorgi, and Amir Houmansadr. 2023. Realistic website fingerprinting by augmenting network traces. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1035–1049.

[2] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. 2019. Var-CNN: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies* 2019, 4 (2019), 292–310.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[4] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. 121–130.

[5] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 227–238.

[6] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. 2022. Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world. In *31st USENIX Security Symposium (USENIX Security 22)*. 753–770.

[7] Tianyu Cui, Xinjie Lin, Sijia Li, Miao Chen, Qilei Yin, Qi Li, and Ke Xu. 2025. TrafficLLM: Enhancing Large Language Models for Network Traffic Analysis with Generic Traffic Representation. arXiv:2504.04222 [cs.LG] https://arxiv.org/abs/2504.04222

[8] Weiqi Cui, Tao Chen, Christian Fields, Julianna Chen, Anthony Sierra, and Eric Chan-Tin. 2019. Revisiting assumptions for website fingerprinting attacks. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 328–339.

[9] Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. 2020. Trafficsliver: Fighting website fingerprinting attacks with traffic splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1971–1985.

[10] Xinhao Deng, Jingyou Chen, Linxiao Yu, Yixiang Zhang, Zhongyi Gu, Changhao Qiu, Xiyuan Zhao, Ke Xu, and Qi Li. 2025. Beyond a Single Perspective: Towards a Realistic Evaluation of Website Fingerprinting Attacks. arXiv:2510.14283 [cs.CR] https://arxiv.org/abs/2510.14283

[11] Xinhao Deng, Qi Li, and Ke Xu. 2024. Robust and reliable early-stage Website fingerprinting attacks via spatial-temporal distribution analysis. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1997–2011.

[12] Xinhao Deng, Qilei Yin, Zhuotao Liu, Xiyuan Zhao, Qi Li, Mingwei Xu, Ke Xu, and Jianping Wu. 2023. Robust Multi-tab Website Fingerprinting Attacks in the Wild. In *IEEE Symposium on Security and Privacy (SP)*.

[13] Xianwen Deng, Ruijie Zhao, Yanhao Wang, Mingwei Zhan, Zhi Xue, and Yijun Wang. 2025. Countmamba: A Generalized Website Fingerprinting Attack via Coarse-Grained Representation and Fine-Grained Prediction . In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1419–1437. doi:10.1109/SP61157.2025.00154

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. doi:10.18653/v1/N19-1423

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[16] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE symposium on security and privacy*. IEEE, 332–346.

[17] Jiajun Gong, Wei Cai, Siyuan Liang, Zhong Guan, Tao Wang, and Ee-Chien Chang. 2024. WFCAT: Augmenting Website Fingerprinting with Channel-wise Attention on Timing Features. *arXiv preprint arXiv:2412.11487* (2024).

[18] Jiajun Gong and Tao Wang. 2020. Zero-delay lightweight defenses against website fingerprinting. In *Proceedings of the 29th USENIX Conference on Security Symposium*. 717–734.

[19] Jiajun Gong, Wuqi Zhang, Charles Zhang, and Tao Wang. 2022. Surakav: Generating Realistic Traces for a Strong Website Fingerprinting Defense. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1558–1573.

[20] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[21] Zhong Guan, Gang Xiong, Gaopeng Gou, Zhen Li, Mingxin Cui, and Chang Liu. 2021. BAPM: Block Attention Profiling Model for Multi-tab Website Fingerprinting Attacks on Tor. In *Annual Computer Security Applications Conference*. 248–259.

[22] Zijun Hang, Yuliang Lu, Yongjie Wang, and Yi Xie. 2023. Flow-mae: Leveraging masked autoencoder for accurate, efficient and robust malicious traffic classification. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 297–314.

[23] Jamie Hayes, George Danezis, et al. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX security symposium*. 1187–1203.

[24] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[25] Rob Jansen. [n. d.]. CELLSHIFT: RTT-Aware Trace Transduction for Real-World Website Fingerprinting. ([n. d.]).

[26] Rob Jansen, Ryan Wails, and Aaron Johnson. 2024. A measurement of genuine tor traces for realistic website fingerprinting. *arXiv preprint arXiv:2404.07892* (2024).

[27] Rob Jansen, Ryan Wails, and Aaron Johnson. 2024. Repositioning Real-World Website Fingerprinting on Tor. In *Proceedings of the 23rd Workshop on Privacy in the Electronic Society* (Salt Lake City, UT, USA) *(WPES '24)*. Association for Computing Machinery, New York, NY, USA, 124–140. doi:10.1145/3689943.3695047

[28] Zhaoxin Jin, Tianbo Lu, Shuang Luo, and Jiaze Shang. 2023. Transformer-based Model for Multi-tab Website Fingerprinting Attack. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1050–1064.

[29] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Dıaz, and Matthew Wright. 2015. WTF-PAD: toward an efficient website fingerprinting defense for tor. *CoRR, abs/1512.00524* (2015).

[30] Xinjie Lin, Gang Xiong, Gaopeng Gou, Wenqi Dong, Jing Yu, Zhen Li, and Wei Xia. 2025. Respond to Change With Constancy: Instruction-Tuning With LLM for Non-I.I.D. Network Traffic Classification. *IEEE Transactions on Information Forensics and Security* 20 (2025), 5758–5773. doi:10.1109/tifs.2025.3574971

[31] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. 2022. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*. 633–642.

[32] David Lu, Sanjit Bhat, Albert Kwon, and Srinivas Devadas. 2018. Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 109–113.

[33] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2023. Sok: A critical evaluation of efficient website fingerprinting defenses. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 969–986.

[34] Xuying Meng, Chungang Lin, Yequan Wang, and Yujun Zhang. 2025. NetGPT: Generative Pretrained Transformer for Network Traffic. arXiv:2304.09513 [cs.NI] https://arxiv.org/abs/2304.09513

[35] Se Eun Oh, Nate Mathews, Mohammad Saidur Rahman, Matthew Wright, and Nicholas Hopper. 2021. GANDaLF: GAN for data-limited fingerprinting. *Proceedings on privacy enhancing technologies* 2021, 2 (2021).

[36] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *NDSS*.

[37] Lingfeng Peng, Xiaohui Xie, Sijiang Huang, Ziyi Wang, and Yong Cui. 2024. Ptu: Pre-Trained Model for Network Traffic Understanding. In *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*. IEEE, 1–12.

[38] Tobias Pulls. 2020. Towards effective and efficient padding machines for Tor. *arXiv preprint arXiv:2011.13471* (2020).

[39] Jian Qu, Xiaobo Ma, and Jianfeng Li. 2024. TrafficGPT: Breaking the Token Barrier for Efficient Long Traffic Analysis and Generation. arXiv:2403.05822 [cs.LG] https://arxiv.org/abs/2403.05822

[40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[41] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. 2019. Tik-Tok: The utility of packet timing in website fingerprinting attacks. *arXiv preprint arXiv:1902.06421* (2019).

[42] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2017. Automated website fingerprinting through deep learning. *arXiv preprint arXiv:1708.06376* (2017).

[43] Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. 2023. Subverting website fingerprinting defenses with robust traffic representation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 607–624.

[44] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1928–1943.

[45] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1131–1148.

[46] Tao Wang. 2020. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 152–167.

Table 4: The default hyperparameter settings of TraVerse.

Module	Hyperparameter	Value
Unsupervised Representation Learning	Burst Dimension	300
	Max Input Length	512
	Masked Ratio	0.2
	LoRA Rank	8
	Max Training Steps	30,000
	Output Dimension	4,096
Few-Shot Supervised Classification	Hidden Dimension	1,024

[47] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*. 143–157.

[48] Tao Wang, Ian Goldberg, et al. 2017. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *USENIX Security Symposium*. 1375–1390.

[49] Yanbin Wang, Haitao Xu, Zhenhao Guo, Zhan Qin, and Kui Ren. 2022. snWF: Website Fingerprinting Attack by Ensembling the Snapshot of Deep Learning. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1214–1226. doi:10.1109/TIFS.2022.3158086

[50] Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. 2024. Netllm: Adapting large language models for networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 661–678.

[51] Yixiao Xu, Tao Wang, Qi Li, Qingyuan Gong, Yang Chen, and Yong Jiang. 2018. A multi-tab website fingerprinting attack. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 327–341.

[52] An Yang, Anfeng Li, Baosong Yang, et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[53] Qilei Yin, Zhuotao Liu, Qi Li, Tao Wang, Qian Wang, Chao Shen, and Yixiao Xu. 2021. An Automated Multi-Tab Website Fingerprinting Attack. *IEEE Transactions on Dependable and Secure Computing* 19, 6 (2021), 3656–3670.

[54] Ruijie Zhao, Mingwei Zhan, Xianwen Deng, Yanhao Wang, Yijun Wang, Guan Gui, and Zhi Xue. 2023. Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5420–5427.

[55] Xiyuan Zhao, Xinhao Deng, Qi Li, Yunpeng Liu, Zhuotao Liu, Kun Sun, and Ke Xu. 2024. Towards Fine-Grained Webpage Fingerprinting at Scale. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 423–436.

[56] Guangmeng Zhou, Xiongwen Guo, Zhuotao Liu, Tong Li, Qi Li, and Ke Xu. 2024. Trafficformer: an efficient pre-trained model for traffic data. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 102–102.

A Details of LoRA Fine-Tuning

Low-Rank Adaptation (LoRA) [24] is a widely used technique that introduces low-rank trainable matrices into transformer layers to approximate the updates of an LLM while keeping the original pretrained parameters frozen, thereby enabling parameter-efficient fine-tuning. Formally, for a weight matrix $W \in \mathbb{R}^{d\times k}$, instead of updating W directly, LoRA represents the update as a low-rank decomposition:

$$W' = W + \Delta W, \quad \Delta W = AB, \tag{5}$$

where $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$, and $r \ll \min(d,k)$. r is the rank hyperparameter that controls the number of additional trainable parameters. During training, only A and B are updated, while W remains frozen. This design substantially reduces the number of trainable parameters and memory consumption, allowing efficient adaptation of large pretrained LLMs to traffic-specific downstream tasks without full-scale fine-tuning.

B Implementation Details of TraVerse

We prototype TraVerse using Python 3.10 and Transformers 4.44.2, and adopt Llama3-8B [20] as the backbone model. Table 4 summarizes the default hyperparameter settings used throughout our evaluation. Figure 8 and Figure 9 present the prompt templates for unsupervised fine-tuning and supervised classification, respectively.

C Details of Datasets

Our datasets are derived from GTT23 [26] and cover closed-world, open-world, and multi-tab evaluations.

Closed-World Dataset $\mathbf{GTT_{CW}}$. For the closed-world evaluation, both training and test traffic are drawn from a total of 200 monitored websites. Traffic from Week 1 is used for training and validation, with 10 samples per website for training and 1 sample per website for validation. We also use traffic from Week 1 for testing, with 75 samples per website. To assess the robustness of TraVerse under temporal drift, we further evaluate on traffic from Weeks 2–13, where the test sets are substantially large. Unlike the training and validation sets, the sample number per website in these test sets is not fixed, reflecting the base rate and traffic distribution encountered in the real world. The precise number of test samples for each week is shown in Table 5.

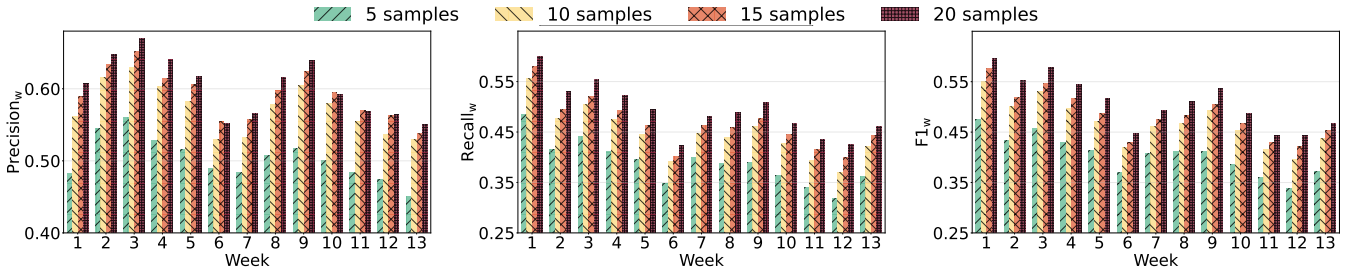
Open-World Dataset $\mathbf{GTT_{OW}}$. For the open-world evaluation, traffic can originate from a large number of unmonitored websites, which are collectively treated as the same unmonitored class. In addition to the monitored traffic used in the closed-world setting, we include 2,000 and 100 samples from unmonitored websites for training and validation, with each sample corresponding to a distinct website. For testing, we incorporate 5,000 unmonitored samples per week, also drawn from disjoint websites, ensuring that none of these websites overlap with those used for classifier training or LLM fine-tuning. The number of test samples for each week is shown in Table 5.

Multi-Tab Dataset $\mathbf{GTT_{MT}}$. Since no real-user multi-tab dataset is publicly available, we follow the method in [21] to synthesize a multi-tab dataset from GTT23. Specifically, we construct a 2-tab scenario and introduce a random inter-tab delay between 3 and 5 seconds. We use the training, validation, and test sets from Week 1 to generate the corresponding multi-tab training, validation, and test samples. The resulting multi-tab dataset contains 4,000 training samples, 200 validation samples, and 5,000 test samples.

D Details of Baselines

We select nine state-of-the-art WF methods and one LLM-based method to benchmark the performance of TraVerse.

- **Single-Tab WF Methods.** DF [44] employs deep neural networks composed of CNN blocks trained on packet direction sequences. Tik-Tok [41] extends DF by incorporating both packet direction and timestamp information. Var-CNN [2] improves the model architecture by introducing dilated convolution and other structural enhancements. TF [45] leverages the triplet network to capture inter-website differences for few-shot identification. RF [43] introduces a traffic representation mechanism, TAM, to improve robustness against defenses. NetCLR [1] integrates data augmentation with self-supervised learning to adapt to diverse network conditions.



(a) Precision$_w$ (b) Recall$_w$ (c) F1$_w$

Figure 7: Performance of TraVerse with different sample numbers.

Table 5: Number of test samples per week in the closed-world and open-world settings.

Scenario	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Closed-World	15,000	13,930	15,064	15,065	15,763	15,603	33,133	16,413	15,010	13,746	16,531	15,965	34,301
Open-World	20,000	18,930	20,064	20,065	20,763	20,603	38,133	21,413	20,010	18,746	21,531	20,965	39,301

Unsupervised Fine-Tuning Prompt Template

Input:
Traffic of browsing a website: <flow>: <burst sequence>: {burst feature}.

Figure 8: Prompt template used for unsupervised fine-tuning.

Supervised Classification Prompt Template

Instruction:
Given a traffic burst sequence as input, predict the website label.

Input:
<flow>: <burst sequence>: {burst feature}.

Figure 9: Prompt template used for supervised classification.

- **Multi-Tab WF Methods.** TMWF [28] is a Transformer-based approaches for multi-tab WF attacks. ARES [12] designs a multi-classifier framework for enhanced multi-tab identification. Countmamba [13] enhances feature representations and employs a state-space-oriented classifier to predict websites in the multi-tab setting.
- **LLM-Based Methods.** TrafficLLM [7] is an LLM-based method that uses dual-stage tuning pipeline for traffic classification.

E Metrics

For the closed-world setting, following [1, 11, 41, 44], we use four evaluation metrics: Accuracy@3, Precision$_w$, Recall$_w$, and F1$_w$. Accuracy@3 measures whether the ground-truth label appears among the top three predicted classes, defined per sample as:

$$\text{Accuracy@3} = \mathbb{I}\big(y_i \in \text{Top-3}(\hat{y}_i)\big), \tag{6}$$

where y_i is the ground-truth label of the i-th sample, Top-t($\hat{y}_i$) denotes the t classes with the highest predicted probabilities, and $\mathbb{I}(\cdot)$ is an indicator function equal to 1 if the condition holds. Precision$_w$, Recall$_w$, and F1$_w$ are the weighted averages of Precision, Recall, and F1-score, respectively, where the weights are proportional to class frequencies in the dataset.

For the open-world setting, following [11, 46], we adopt r-precision as the evaluation metric, which is defined per sample as:

$$\text{r-precision} = \frac{R_{TP}}{R_{TP} + R_{WP} + r \cdot R_{FP}}, \tag{7}$$

where R_{TP}, R_{WP} and R_{FP} denote true positive rate, wrong positive rate and false positive rate, respectively. In our evaluation, we set $r = 5$, where r denotes the ratio of visits to unmonitored websites relative to monitored websites.

For the multi-tab setting, following [55], we use Recall@k with $k = 10$ and AP@k with $k = 2$, defined per sample as:

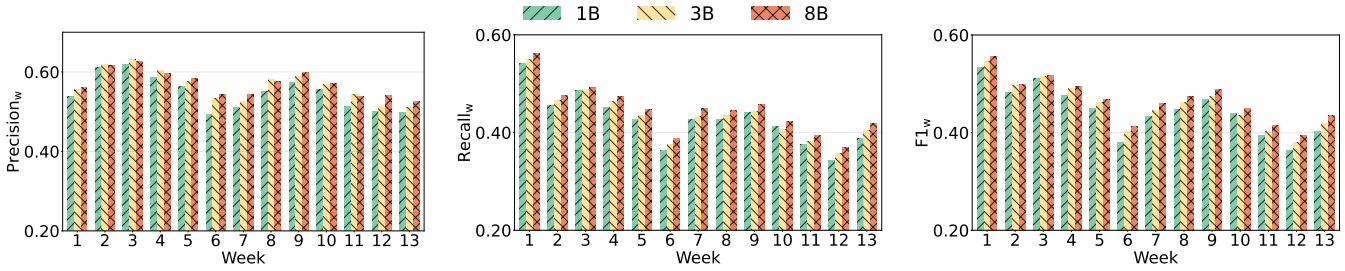
$$Recall@k = \frac{\|y_i \cap \text{Top-}k(\hat{y}_i)\|}{\|y_i\|}. \tag{8}$$

$$AP@k = \frac{1}{\min(k, \|y_i\|)} \sum_{t=1}^{k} \frac{\| y_i \cap \text{Top-}t(\hat{y}_i) \|}{t}. \tag{9}$$

F Performance with Various Sample Numbers

We evaluate the performance of TraVerse across different sample sizes to investigate its adaptability in real-world scenarios. As websites are constantly evolving, we test the model's ability to leverage a modest number of samples for rapid adaptation to feature drift. We conduct this experiment in the closed-world setting, varying the sample number per website from 5, 10, 15, to 20. We use Precision$_w$, Recall$_w$ and F1$_w$ as metrics.

The results in Figure 7 demonstrate the effectiveness of TraVerse with varying numbers of labeled samples. Specifically, increasing the sample number from 5 to 10, 10 to 15, and 15 to 20 yields F1$_w$ improvements of 15.5%, 3.7%, and 4.8%, respectively. The largest gain is obtained when increasing the sample number from 5 to



(a) Precision$_w$ **(b) Recall$_w$** **(c) F1$_w$**

Figure 10: Performance of TraVerse with various model sizes.

10, after which the improvements become markedly smaller, indicating diminishing returns beyond 10 samples. With only 10 samples, TraVerse achieves average Precision$_w$, Recall$_w$ and F1$_w$ of approximately 0.57, 0.45 and 0.47 across 13 weeks, respectively. This efficiency stems from the robust and transferable traffic representations learned by the fine-tuned LLM, which support accurate website identification with few labeled samples.

Remark. TraVerse requires only a modest number of samples to achieve accurate website identification, underscoring its ability to adapt efficiently under dynamic real-world conditions.

G Performance with Various Model Sizes

In this section, we study the impact of backbone model size on TraVerse to determine whether its improvements are primarily driven by scaling the LLM. Specifically, we instantiate TraVerse with three backbone sizes, including the 1B, 3B, and 8B variants. All experiments are conducted on the closed-world dataset $\mathbf{GTT_{CW}}$, and we report Precision$_w$, Recall$_w$ and F1$_w$.

Figure 10 shows that increasing model size yields only marginal improvements. In particular, the 8B variant achieves an average F1$_w$ gain of 4.8% over the 1B model and 2.2% over the 3B model. The same trend is observed for Precision$_w$ and Recall$_w$, indicating that the overall benefit from scaling is limited in this setting. These results suggest that the performance of TraVerse is not solely driven by larger model scale. Instead, the proposed domain-specific adaptation plays a critical role in enabling LLMs to generate robust representations for real-world traffic.

Remark. Overall, reducing the model size from 8B to 1B leads to only a modest performance degradation. This indicates that TraVerse is relatively insensitive to model scale and can retain competitive performance even when deployed with smaller backbones.