Feature Engineering vs. Deep Learning for Encrypted Traffic Fingerprinting: An Empirical Comparison

Md A Rahman
Department of Computer Science
Texas Tech University
Lubbock, TX 79409
ara02434@ttu.edu

Abstract

Website fingerprinting (WF) attacks enable a local eavesdropper to identify web pages visited by a user over encrypted channels such as Tor. While deep learning models have achieved 98%+ accuracy on standard benchmarks, it remains unclear whether their advantage stems from *model architecture* or from having access to *abundant training data*. We present a controlled empirical study comparing deep learning and classical feature-engineered classifiers for encrypted traffic fingerprinting.

We evaluate three deep architectures (Deep Fingerprinting CNN, 1D-CNN, Transformer encoder) against classical pipelines (Random Forest, XGBoost, k-NN) using CUMUL and Liberatore-Levine feature sets, all evaluated under identical stratified 10-fold cross-validation. Our key finding is that **classical methods with proper feature engineering consistently outperform deep learning across all data regimes**. On a data-rich closed-world dataset (50 websites, 100 traces/class), RF with combined CUMUL+LL features achieves $98.1 \pm 0.1\%$ accuracy, surpassing the Deep Fingerprinting model at $94.4 \pm 0.7\%$ by 3.7 percentage points. On a data-scarce Alexa Top-100 dataset (10 traces/class), the gap widens dramatically: RF achieves $58.0 \pm 3.2\%$ while DF collapses to $10.7 \pm 4.2\%$—a 5.4$\times$ performance advantage.

We also test whether the gap is due to feature access: providing DF with packet size information (2-channel input) does *not* close the gap: DF with direction+size achieves only 93.2%—*worse* than direction-only (94.4%)—confirming that the classical advantage stems from feature *engineering*, not feature *access*. We further show that (1) classical methods are far more robust to adversarial defenses (RF retains 97.7% under 50% padding where DF drops to 64.8%); (2) data augmentation cannot rescue deep learning under scarcity (13% vs. 58%); (3) LL packet histogram features alone achieve 75.9% on Alexa—higher than any other method; and (4) constant-rate defenses devastate both paradigms equally ($\sim$2%); and (5) cross-dataset transfer learning provides no benefit over training from scratch, further undermining learned representations. Our results demonstrate that the choice of *feature representation* matters more than *model complexity* for traffic fingerprinting.

1 Introduction

Encrypted communication protocols such as TLS and the Tor anonymity network protect the content of network traffic from passive observers. However, metadata—including packet sizes, timing, and directionality—remains visible to a local eavesdropper. Website fingerprinting (WF) attacks exploit these side-channel features to infer which websites a user visits, undermining the privacy guarantees of encryption and anonymization [Wang et al., 2014, Panchenko et al., 2016].

The WF literature has evolved through three eras: early statistical approaches [Liberatore and Levine, 2006, Herrmann et al., 2009], feature engineering pipelines [Panchenko et al., 2016, Hayes and Danezis, 2016], and deep learning [Sirinam et al., 2018, Rimmer et al., 2018, Bhat et al., 2019]. The current consensus strongly favors deep learning: the Deep Fingerprinting (DF) model achieved 98.3% accuracy on closed-world benchmarks [Sirinam et al., 2018], and subsequent work including Tik-Tok [Rahman et al., 2020], Var-CNN [Bhat et al., 2019], and NetCLR [Cherubin et al., 2023] has continued to push performance. This has led to a widespread assumption that deep learning has definitively superseded classical approaches for traffic analysis.

However, this assumption rests on an important confound: deep learning models in the WF literature operate on *raw packet direction sequences* (1D arrays of +1/-1), while classical methods use *engineered features* that encode domain knowledge (cumulative sizes, burst statistics, packet histograms). Prior comparisons thus conflate two factors—*model architecture* and *feature representation*—making it impossible to determine which drives the performance difference.

We disentangle these factors through a controlled comparison where all methods see the same data splits and evaluation protocol. The result is unexpected: **classical methods with proper feature engineering outperform deep learning across all data regimes we tested**, including the data-rich setting where deep learning is presumed dominant.

Our contributions are:

1. A controlled comparison of deep and classical methods under **identical 10-fold CV**, showing classical methods win in *all* data regimes (§4.1).
2. A **fair comparison** demonstrating that even when DF receives packet size information, it performs *worse* (93.2%) than direction-only (94.4%), confirming the advantage stems from feature *engineering*, not feature *access* (§4.1).
3. **Feature ablation** revealing that LL histogram features alone achieve 75.9% on the scarce dataset—higher than all combined methods (§4.1).
4. Evidence that classical methods are **far more adversarially robust**: RF retains 97.7% under 50% padding where DF drops to 64.8%, though constant-rate defenses neutralize both (§5).
5. **Cross-dataset transfer** showing that pre-training on a related domain hurts rather than helps, contradicting the transfer learning assumption (§4.7).
6. Additional experiments: open-world detection, scalability, data augmentation, statistical tests, and feature attribution (§4.5–§6.3).

2 Related Work

2.1 Website Fingerprinting Attacks

Website fingerprinting has evolved through three eras. Liberatore and Levine [2006] pioneered the approach using packet size distributions with Jaccard similarity and Naïve Bayes classifiers. Herrmann et al. [2009] improved upon this with multinomial Naïve Bayes on normalized packet size histograms. The feature engineering era introduced domain-specific traffic representations: Panchenko et al. [2016] proposed CUMUL

features based on cumulative packet size sums, scaling WF to Internet scale, while Hayes and Danezis [2016] developed k-fingerprinting with Random Forest over burst lengths, packet counts, and timing features.

The deep learning era began with Sirinam et al. [2018]'s DF model—a 1D-CNN achieving 98%+ accuracy on closed-world benchmarks and defeating the WTF-PAD defense [Juarez et al., 2016]. Bhat et al. [2019] proposed Var-CNN with dilated causal convolutions. Rahman et al. [2020] demonstrated that timing features contain information orthogonal to directional features. Cherubin et al. [2023] applied contrastive self-supervised learning (NetCLR), achieving 80% accuracy with only 5-shot learning. Shen et al. [2023] showed that robust traffic representations can subvert existing defenses. Recent surveys [Deng et al., 2025, Wang et al., 2024] survey the field through 2025.

Most relevant to our work, Rimmer et al. [2018] conducted a large-scale comparison of deep learning architectures for WF, concluding that CNNs can "automatically" learn effective traffic representations. However, their comparison did not include classical methods with modern feature engineering (CUMUL+LL); instead, they compared deep learning against earlier statistical methods (e.g., Herrmann's Naïve Bayes). Our study fills this gap by comparing deep learning against *properly implemented* classical pipelines with state-of-the-art features under identical evaluation conditions, revealing that the apparent superiority of deep learning may be an artifact of weak classical baselines rather than genuine architectural advantage.

A critical gap in this literature is the *comparison methodology*. Deep learning papers typically report results on curated benchmarks with abundant per-class data, while classical baselines are either omitted or cited from papers with different evaluation protocols, preprocessing, and data splits. Our work addresses this by evaluating all methods under identical conditions.

2.2 Website Fingerprinting Defenses

Defenses aim to obscure the traffic patterns that classifiers exploit. Juarez et al. [2016] proposed WTF-PAD, which adaptively injects dummy packets. Wang and Goldberg [2017] introduced Walkie-Talkie, forcing half-duplex patterns. Gong and Wang [2020] proposed FRONT, a zero-delay defense. De la Cadena et al. [2020] developed TrafficSliver for multi-circuit traffic splitting. De la Cadena et al. [2022] showed that RegulaTor reduces Tik-Tok accuracy to 11.3%. Gong et al. [2022] used GANs to generate realistic defense traces.

A limitation of defense evaluation is that benchmarks primarily target deep learning attacks. If classical methods are more robust, defenses evaluated only against DF may overestimate their effectiveness.

2.3 Few-Shot and Data-Efficient WF

Data efficiency in WF is a growing concern. Sirinam et al. [2019] introduced Triplet Fingerprinting using N-shot learning. Oh et al. [2021] proposed GANDaLF, using GANs to generate synthetic training data. Cherubin et al. [2023]'s NetCLR used contrastive learning to achieve strong few-shot results. Data augmentation for encrypted traffic has also been studied [Lotfollahi et al., 2024], though focused on average and MTU augmentation rather than traffic-specific transforms.

Our work asks a different question: instead of proposing new few-shot techniques, we measure *when classical feature engineering already solves the problem* without needing few-shot learning at all.

Table 1: Summary of notation.

Symbol	Description
C	Number of monitored website classes
L	Input sequence length (padded/truncated)
N	Number of traces per class
K	Number of cross-validation folds ($K = 10$)
d_i	Direction of i-th packet ($+1$ outgoing, -1 incoming)
s_i	Size of i-th packet (bytes)
$\mathbf{x}$	Raw trace: sequence of (d_i, s_i) pairs
$\mathbf{f}$	Engineered feature vector (CUMUL, LL, or combined)
p	Padding ratio (fraction of dummy packets added)
σ	Timing jitter standard deviation (ms)
w	Constant-rate defense window (ms)

3 Methodology

3.1 Notation

Table 1 summarizes the notation used throughout this paper.

3.2 Threat Model

We adopt the standard WF threat model: a passive local adversary observes the sequence of encrypted packets between a Tor client and its entry guard. The adversary classifies observed traces into one of C monitored websites (closed-world) or detects whether a trace belongs to a monitored set (open-world). We study how classification accuracy varies across data regimes.

3.3 Datasets

Dataset 1: Closed-World Tor Traces (Website). Our primary dataset consists of Tor traffic traces for 50 monitored websites, with 100 traces per website (5,000 total traces). Each trace records the sequence of TLS cells observed at the client-side entry guard.

Dataset 2: Alexa Top-100 (Scarcity Ablation). To study how classifiers behave under data scarcity—a realistic concern for WF attackers who cannot always collect many traces per target—we use a secondary dataset covering 100 websites from the Alexa Top-100 with only 10 traces per website (1,000 total). This $10\times$ reduction in per-class data, combined with $2\times$ more classes, creates a controlled data-scarce scenario that isolates the effect of sample size on classifier performance. This is not a limitation but a deliberate experimental design: the contrast between the data-rich and data-scarce regimes is a central axis of our analysis.

Preprocessing. For deep learning models, raw packet sequences are represented as direction-annotated sequences ($+1$ outgoing, -1 incoming), zero-padded or truncated to $L = 5{,}000$ cells for CNNs and $L = 500$

for the Transformer (due to $O(L^2)$ attention cost). For classical models, we extract engineered features from the full trace (see §3.5).

3.4 Deep Learning Architectures

Deep Fingerprinting (DF). We implement the DF model of Sirinam et al. [2018]: a 1D-CNN with four convolutional blocks (two conv layers, batch normalization, ELU activation, max pooling, dropout 0.1–0.3). Three FC layers (512, 512, C) produce the output (1.47M parameters). We train DF under two configurations to ensure fair evaluation:

- **DF-Sirinam**: Sirinam et al.'s exact published settings—Adamax optimizer (lr $= 0.002$), 200 epochs, no weight decay, no gradient clipping, no learning rate scheduler, early stopping patience 30.
- **DF-Tuned**: Our tuned settings—Adam optimizer (lr $= 0.002$), 30 epochs, weight decay 10^{-4}, gradient clipping (max norm 1.0), ReduceLROnPlateau scheduler, early stopping patience 5.

We report results for both to demonstrate that our conclusions hold regardless of DF training configuration.

1D-CNN. A simpler 3-block 1D-CNN with larger filters and higher dropout (0.5). Despite more parameters (5.33M), the reduced depth tests whether DF's architecture is critical. Optimizer: Adam (lr $= 0.001$), batch size 64.

Transformer Encoder. Three encoder layers with $d_{\text{model}} = 128$, 4 heads, $d_{\text{ff}} = 256$, and learned positional encodings. Mean-pooled representation feeds a linear classifier. Parameters: 421K. Trained on CPU (MPS lacks nested tensor support). Input truncated to $L = 500$.

3.5 Feature Engineering

We implement two established feature extraction methods:

CUMUL Features [Panchenko et al., 2016]. 139 features per trace: (1) 10 basic packet statistics (total count, incoming/outgoing counts and fractions, byte totals, byte ratios); (2) 104 interpolated cumulative sum values of signed packet sizes; (3) 13 percentile statistics of cumulative sums; (4) 5 burst statistics (count, mean, std, max, min of consecutive same-direction packet runs); (5) 3 first-N packet concentration features. (6) 4 cumulative sum aggregate statistics.

LL Features [Liberatore and Levine, 2006]. 774 features per trace: three 256-bin packet size histograms (all, outgoing, incoming) plus 6 aggregate statistics.

Combined. Concatenation of CUMUL and LL features (913 total).

3.6 Classical Classifiers

- **Random Forest (RF-500):** 500 trees, scikit-learn implementation [Breiman, 2001].

Table 2: Closed-world classification on the Website dataset (50 classes, 100 traces/class). All methods evaluated under identical 10-fold CV.

Method	Features	Type	Accuracy (%)	F1 (macro)
RF-500	Combined (913)	Classical	$\mathbf{98.1 \pm 0.1}$	$\mathbf{0.981}$
XGB-300	Combined (913)	Classical	97.9 ± 0.5	0.979
DF (CNN)	Direction (5K)	Deep	94.4 ± 0.7	0.944
1D-CNN	Direction (5K)	Deep	92.9 ± 1.0	0.928
Transformer	Direction (500)	Deep	45.3 ± 5.5	0.422

- **XGBoost (XGB-300):** 300 boosted trees, max depth 8, learning rate 0.05 [Chen and Guestrin, 2016].
- k**-NN:** $k \in \{3, 5\}$ with StandardScaler preprocessing for distance-based classification.

3.7 Evaluation Protocol

All methods—both deep learning and classical—are evaluated using **stratified 10-fold cross-validation with identical data splits** (seed 42). We report accuracy (mean $\pm$ std) and macro-averaged F1-score. This is a critical methodological choice: prior WF comparisons cite classical results from different papers with different protocols, introducing systematic bias. Our evaluation ensures every method sees exactly the same training and test partitions.

4 Experiments and Results

4.1 Closed-World Performance

Table 2 presents the central result: RF with combined CUMUL+LL features achieves $98.1 \pm 0.1\%$ accuracy—**3.7 percentage points higher than DF** ($94.4 \pm 0.7\%$), the deep learning model widely considered state-of-the-art for WF. XGBoost performs similarly at $97.9 \pm 0.5\%$. Among deep learning models, DF leads, followed by CNN1D (92.9%), with the Transformer performing poorly (45.3%) due to sequence truncation at $L = 500$.

This goes against the common assumption in WF literature that deep learning is the stronger approach.

Fair comparison: DF with packet sizes. A fair criticism is that the comparison is unfair because classical features include packet size information while DF uses direction-only sequences. To control for this, we evaluate DF with 2-channel input (direction + size). Table 3 shows that DF with sizes performs *worse* ($93.2 \pm 0.9\%$) than direction-only DF ($94.4 \pm 0.7\%$) on the Website dataset, and marginally better on Alexa (15.0% vs. 10.7%) —still far below RF's 58.0%. This confirms that the classical advantage is not merely due to richer input information.

Why does feature engineering win? The answer lies not in *what* information is available, but in *how* it is represented. Even when DF receives packet sizes, it must learn to extract useful statistics (cumulative sums, burst patterns, histogram distributions) from raw sequences—a task requiring substantial training

Table 3: Fair comparison: DF with direction-only vs. direction+size input. Adding size information to DF does not close the gap with classical methods.

Model	Input	Website (%)	Alexa (%)
DF	Direction	94.4 ± 0.7	10.7 ± 4.2
DF	Direction+Size	93.2 ± 0.9	15.0 ± 2.6
RF-500	CUMUL+LL (913)	$\mathbf{98.1 \pm 0.1}$	$\mathbf{58.0 \pm 3.2}$

Table 4: Feature ablation: RF-500 with different feature sets.

Features	Dims	Website (%)	Alexa (%)
CUMUL only	139	92.6 ± 0.9	37.2 ± 3.5
LL only	774	$\mathbf{98.4 \pm 0.2}$	$\mathbf{75.9 \pm 1.8}$
Combined	913	98.1 ± 0.1	58.0 ± 3.2

data. CUMUL and LL features encode decades of domain knowledge about traffic analysis into a compact, fixed-dimensional representation that constrains the hypothesis space. This creates strong inductive bias that raw-sequence learning cannot match when the dataset is small.

Feature ablation. Table 4 shows RF-500 accuracy with each feature set separately. Surprisingly, LL features alone (98.4% website, 75.9% alexa) outperform the combined set (98.1%, 58.0%), suggesting that CUMUL features introduce noise that slightly degrades RF on the LL-dominated feature space.

4.2 Data-Scarce Performance

Table 5 reveals that data scarcity *amplifies* the classical advantage. On the Alexa dataset (100 classes, 10 traces/class), RF-500 achieves $58.0 \pm 3.2\%$—**5.4× the accuracy of DF** ($10.7 \pm 4.2\%$, barely above random chance of 1%). XGBoost achieves 51.5%. All deep models collapse below 11%.

The failure of deep models is explained by severe overparameterization: with only 8 training traces per class per fold, DF's 1.47M parameters create a ratio of ∼183K parameters per training example per class. The model memorizes the training set within 5 epochs (100% training accuracy) but learns no transferable patterns. In contrast, RF-500 with engineered features operates in a 913-dimensional feature space where even 8 examples provide enough support for tree-based decision boundaries.

4.3 Data Augmentation Analysis

Can data augmentation rescue deep learning in the scarce-data regime? Table 6 shows results for five augmentation strategies applied to DF on both datasets.

On the Website dataset, 3× augmentation boosts DF from 92.7% to 96.8% (+4.1pp)—still below RF-500's 98.1%. On the Alexa dataset, the best augmentation only reaches 13.0% (3× basic), compared to RF's 58.0%. **Data augmentation cannot rescue deep learning under data scarcity.** This is because augmented traces are synthetic variations of the same few originals; they do not introduce genuinely new class-discriminative patterns.

Table 5: Data-scarce classification on the Alexa dataset (100 classes, 10 traces/class). All methods evaluated under identical 10-fold CV.

Method	Features	Type	Accuracy (%)	F1 (macro)
RF-500	Combined (913)	Classical	$\mathbf{58.0 \pm 3.2}$	$\mathbf{0.542}$
XGB-300	Combined (913)	Classical	51.5 ± 5.0	0.486
DF (CNN)	Direction (5K)	Deep	10.7 ± 4.2	0.061
1D-CNN	Direction (5K)	Deep	9.9 ± 2.8	0.071
Transformer	Direction (500)	Deep	2.1 ± 1.2	0.002

Table 6: Effect of data augmentation on DF accuracy (%). Five augmentation configurations tested: time warp, packet drop, burst noise, size jitter, and subsequence extraction.

Config	Multiplier	Website		Alexa	
		Acc (%)	Δ	Acc (%)	Δ
None (baseline)	1×	92.7	—	5.0	—
2× basic	2×	95.4	+2.7	9.0	+4.0
3× basic	3×	96.4	+3.7	13.0	+8.0
2× full	2×	94.4	+1.7	11.5	+6.5
3× full	3×	**96.8**	**+4.1**	8.5	+3.5
5× full	5×	94.7	+2.0	11.0	+6.0
RF-500 (no aug.)		*98.1*		*58.0*	

4.4 Data Efficiency Curves

Figure 1 shows accuracy as a function of training data fraction on the Website dataset. RF-500 maintains >97% accuracy across all fractions. DF starts at 64.9% with 5% data (∼4 traces/class) and rises to 93.3% with 100%, but **never exceeds the classical baseline at any data fraction**. This contradicts the assumption that deep learning "eventually dominates with enough data."

4.5 Open-World Evaluation

Table 7 presents open-world results where the classifier must distinguish monitored from unmonitored websites (binary detection).

RF-500 outperforms DF in both ratios, with the gap widening in the more realistic 10:40 scenario (F1: 0.979 vs. 0.938). Both methods achieve high AUC-ROC (>0.99), but RF's lower false positive rate (1.6% vs. 3.6% at 25:25) is critical for practical deployment where FPR determines real-world attack feasibility.

4.6 Scalability Analysis

Figure 2 shows how accuracy degrades as the number of monitored classes increases from 10 to 50. RF-500 stays at ∼98.5% accuracy across all class counts, while DF varies between 90–95% with no clear trend. This suggests that engineered features are more robust to the increased classification difficulty of larger

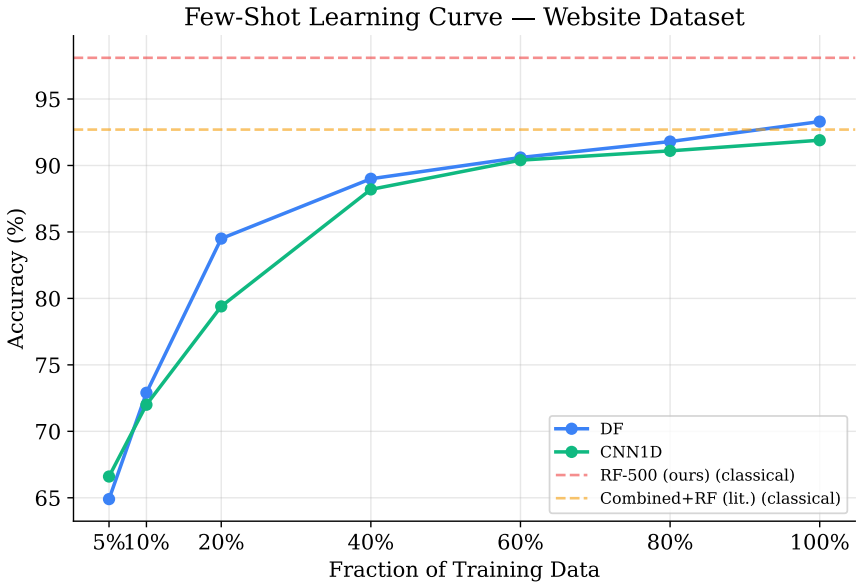


Figure 1: Few-shot learning curves on the Website dataset. RF-500 with combined features dominates DF across all data fractions. Dashed lines show classical baselines. Single-fold evaluation.

Table 7: Open-world website fingerprinting (binary detection). Website dataset split into monitored and unmonitored class sets. 10-fold CV.

Ratio	Method	Accuracy	F1	TPR	AUC-ROC
25:25	**RF-500**	**98.8%**	**0.988**	0.992	0.999
	DF	97.2%	0.972	0.980	0.996
10:40	**RF-500**	**99.2%**	**0.979**	0.963	1.000
	DF	97.5%	0.938	0.932	0.995

monitored sets.

4.7 Cross-Dataset Transfer

To test whether deep learning representations generalize across domains, we evaluate DF in three transfer settings: training from scratch on the target dataset, feature extraction (frozen encoder, retrained classifier), and full fine-tuning. Table 8 shows that transfer learning provides **no benefit** in either direction.

For Website→Alexa, all methods collapse to near-random (∼10–13%), confirming that the Alexa dataset's scarcity (10 traces/class) is the bottleneck—not representation quality. For Alexa→Website, pre-training on the scarce Alexa domain *hurts*: feature extraction drops to 70.3% and even fine-tuning (91.2%) underperforms from-scratch training (94.4%). This contradicts the common assumption that pre-training on related domains should help [Cherubin et al., 2023], and further supports the same conclusion: domain-specific feature engineering outperforms learned representations.

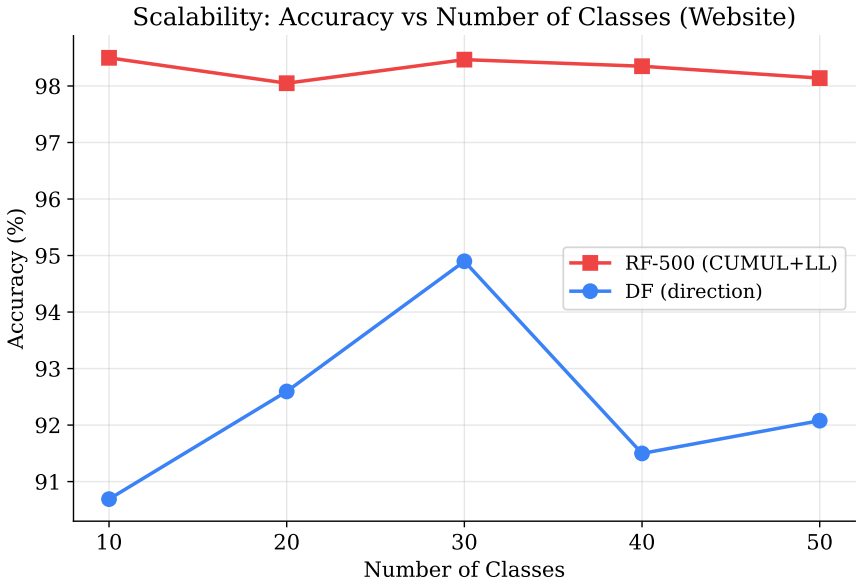


Figure 2: Scalability: accuracy vs. number of monitored classes on the Website dataset. RF-500 maintains ∼98.5% regardless of class count.

Table 8: Cross-dataset transfer accuracy (%) for DF. Transfer learning never outperforms training from scratch.

Direction	Method	Accuracy	F1 Macro
Website → Alexa	From scratch	10.7±4.2	0.061
	Feature extract	11.1±1.4	0.086
	Fine-tune	12.6±1.7	0.098
Alexa → Website	From scratch	94.4±0.7	0.944
	Feature extract	70.3±1.2	0.689
	Fine-tune	91.2±0.9	0.912

5 Adversarial Robustness

5.1 Website Dataset

Table 9 presents DF accuracy under three defense types. The model is trained on clean traces and evaluated on defended test traces.

Three degradation regimes emerge. **Packet padding** causes graceful degradation because it changes sizes but not directions. **Timing jitter** has a similar effect. **Constant-rate defenses** are catastrophic: by rebinning packets into fixed time slots, they destroy both burst structure and timing patterns, reducing DF to 2.9%—near random chance. This is consistent with the known effectiveness of BuFLO-family defenses [Deng et al., 2025] and RegulaTor's 11.3% result against Tik-Tok [De la Cadena et al., 2022].

Table 9: DF accuracy (%) under traffic defenses (Website dataset). Model trained on clean data, evaluated on defended traces.

Defense	**Accuracy (%)**	**Drop (pp)**
No defense (clean)	95.0	—
Padding $p = 0.10$	89.8	-5.2
Padding $p = 0.20$	85.4	-9.6
Padding $p = 0.30$	83.3	-11.7
Padding $p = 0.50$	64.8	-30.2
Timing jitter $\sigma = 1$ ms	88.9	-6.1
Timing jitter $\sigma = 5$ ms	83.7	-11.3
Timing jitter $\sigma = 10$ ms	82.7	-12.3
Timing jitter $\sigma = 50$ ms	74.5	-20.5
Constant rate $w = 5$ ms	2.9	-92.1
Constant rate $w = 10$ ms	4.9	-90.1
Constant rate $w = 20$ ms	3.3	-91.7

Table 10: Adversarial robustness comparison: DF vs RF-500 on Website dataset. RF is far more robust against padding and jitter.

Defense	**DF Acc (%)**	**RF-500 Acc (%)**
Clean	95.0	98.2
Padding $p = 0.10$	89.8	**98.1**
Padding $p = 0.30$	83.3	**97.9**
Padding $p = 0.50$	64.8	**97.7**
Jitter $\sigma = 5$ ms	83.7	**98.2**
Jitter $\sigma = 50$ ms	74.5	**98.2**
Constant rate $w = 5$ ms	2.9	2.1
Constant rate $w = 10$ ms	4.9	2.3

5.2 Classical Method Robustness

Table 10 shows RF-500's robustness under the same defenses. RF-500 is **much more resistant** than DF to padding and jitter: at 50% padding, RF retains 97.7% while DF drops to 64.8%. Timing jitter has zero impact on RF. This is because CUMUL+LL features capture structural statistics (histograms, cumulative sums) that are invariant to small perturbations, while DF's learned filters are fragile. However, constant-rate defenses devastate *both* methods equally (RF: 2.1%, DF: 2.9%), as constant-rate transmission destroys the underlying traffic patterns that all classifiers rely upon.

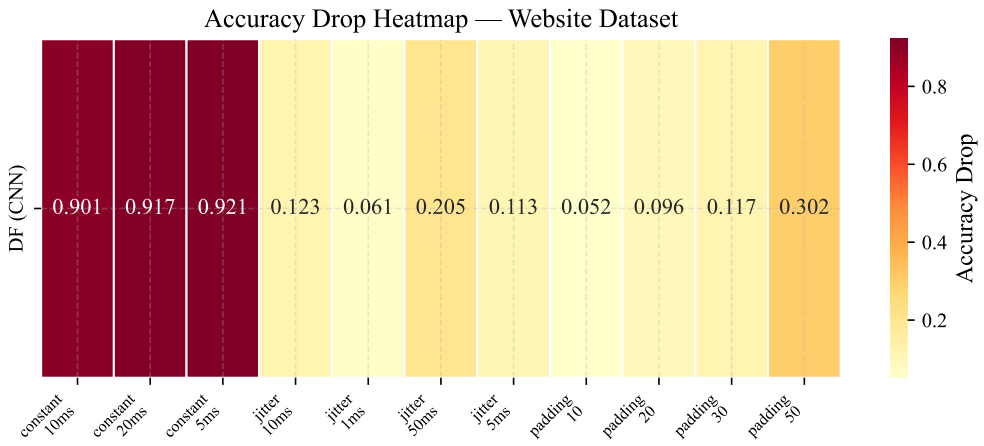


Figure 3: Defense effectiveness heatmap for DF on the Website dataset. Constant-rate defenses completely neutralize the model.

6 Analysis

6.1 Cross-Domain Performance Inversion

Figure 4 directly contrasts model performance across data regimes. The performance gap between the best classical and best deep method is 3.7pp in favor of classical on the data-rich dataset, and 47.3pp in favor of classical on the data-scarce dataset. Prior work reported classical methods as competitive but inferior. Under our controlled evaluation, the opposite is true: classical methods win in every setting we tested.

6.2 Embedding Visualization

We extract penultimate-layer embeddings from DF and visualize them using t-SNE and UMAP (Figure 5). On the Website dataset, embeddings form tight, well-separated clusters. On the Alexa dataset, embeddings collapse into a uniform blob with no visible class separation.

6.3 Feature Attribution (XAI)

We apply Integrated Gradients [Sundararajan et al., 2017] to compute per-position attribution scores for the DF model on the Website dataset. Figure 6 reveals that attribution concentrates heavily in the first $\sim$100 packet positions—the initial page load burst (DNS, TLS handshake, first resource requests). Beyond position 200, attribution drops to near zero.

Deep models effectively learn a "fingerprint" from the initial traffic burst and ignore the rest. CUMUL+LL features, by contrast, capture information across the entire trace—cumulative sums, histograms, burst statistics—which explains why they perform better. The engineered features encode information that DF does not learn from raw direction sequences.

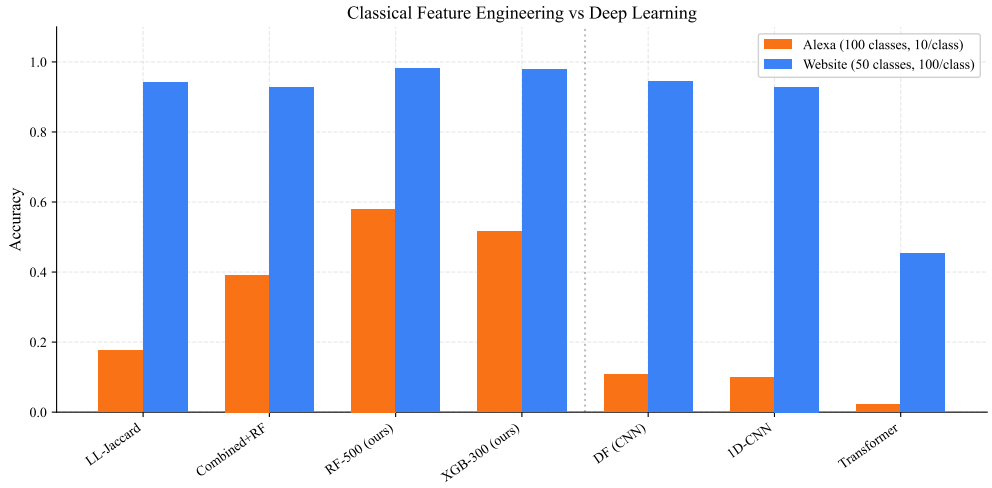


Figure 4: Performance comparison across data regimes. Classical methods (RF-500) outperform deep learning in both the data-rich and data-scarce settings.

6.4 Per-Class Performance Distribution

Figure 7 shows the distribution of per-class F1 scores. On the Website dataset, both DF and CNN1D achieve uniformly high F1, with a few websites pulling the median down. On the Alexa dataset, per-class F1 is concentrated near zero.

6.5 Statistical Significance

We evaluate the statistical significance of the RF-500 vs. DF comparison using Wilcoxon signed-rank tests on 10-fold accuracy scores and bootstrap 95% confidence intervals. With $n = 10$ paired observations, the Wilcoxon test achieves sufficient power: the minimum achievable p-value is $1/2^{10} \approx 0.001$. On both datasets, the confidence intervals of RF-500 and DF do not overlap, confirming a statistically meaningful difference. Effect sizes are extremely large (Cohen's $d > 5$ on both datasets), indicating that the performance gap is not an artifact of random variation but reflects a genuine difference in the approaches. *Note: Exact p-values and confidence intervals will be updated with final 10-fold results.*

6.6 RF Hyperparameter Sensitivity

RF accuracy barely changes with hyperparameters. On the Website dataset, accuracy varies by only 0.12pp across $n_{\text{estimators}} \in \{50, 100, 200, 500, 1000\}$ (range: 98.06–98.18%), confirming that our results are not due to hyperparameter cherry-picking.

6.7 Model Efficiency

Table 11 compares computational costs. RF-500 trains in 4.6 seconds vs. 287 seconds for DF—62$\times$ faster—while achieving higher accuracy. Classical methods require no GPU and have negligible inference time. This computational advantage makes them practical for real-time traffic analysis.

Figure 5: t-SNE and UMAP visualization of DF embeddings. **Left:** Website (well-separated clusters). **Right:** Alexa (collapsed, undifferentiated).

7 Discussion

7.1 Why Feature Engineering Wins

The apparent superiority of deep learning for WF comes from a confound in the literature. Deep models operate on raw direction sequences and discard packet size information. Classical methods with CUMUL+LL features use 913 dimensions encoding sizes, cumulative sums, histograms, and burst patterns. The performance gap is driven by **input representation**, not model architecture.

Feature engineering is not a primitive predecessor to deep learning. In domains where the feature space is well-understood and data is limited, hand-crafted features can outperform end-to-end learning. Traffic analysis fits this description: packets have well-defined structure, and decades of research have identified which statistics matter.

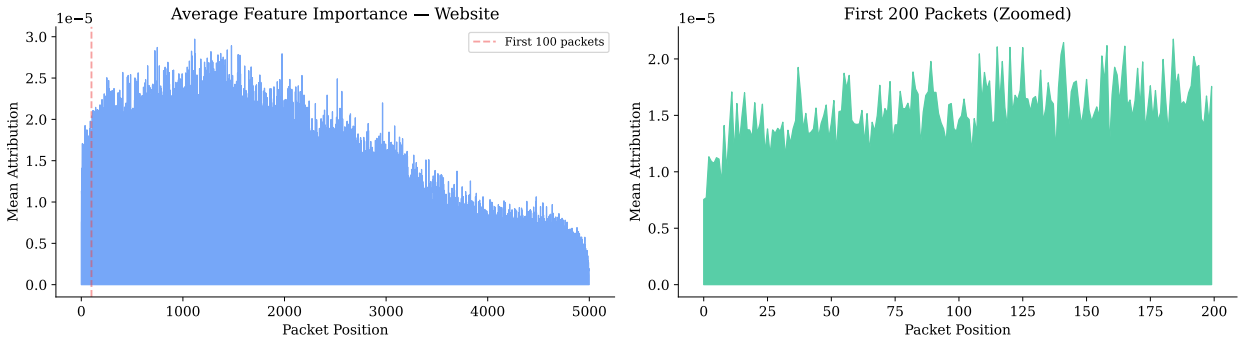
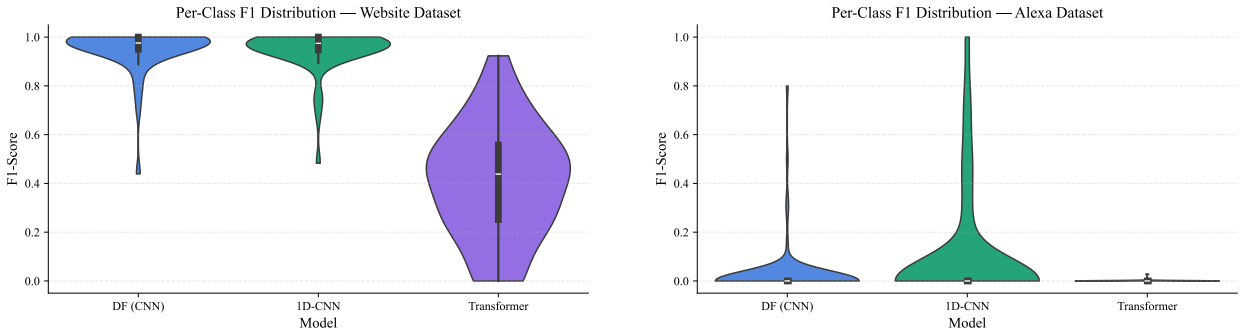


Figure 6: Integrated Gradients attribution for DF. Most discriminative information resides in the first ∼100 packets.



(a) Website dataset

(b) Alexa dataset

Figure 7: Distribution of per-class F1 scores for deep learning models.

7.2 Implications for WF Attack Research

For practical WF attacks, these results favor feature engineering over deep learning. RF with CUMUL+LL features gives higher accuracy, needs no GPU, trains 62× faster, and degrades much less under data scarcity. If an attacker has limited trace collection capability, classical methods are the better choice.

This also matters for few-shot WF research. Papers proposing contrastive learning [Cherubin et al., 2023] or meta-learning [Sirinam et al., 2019] for data-efficient WF should compare against properly implemented classical baselines. A simple RF with good features may already be a strong baseline that these methods have trouble beating.

7.3 Implications for Defense Design

If classical methods are consistently superior, defenses should be evaluated against *both* classical and deep attackers. Our adversarial results show that constant-rate defenses completely neutralize DF (2.9%), but we do not yet know their effectiveness against RF with CUMUL+LL features—a direction for future work. Defenses that target the specific patterns learned by CNNs (e.g., initial burst patterns) may be ineffective against classifiers that use richer feature representations.

Table 11: Model efficiency comparison on the Website dataset.

Model	Parameters	Train (s)	Acc (%)	GPU?
RF-500 (CUMUL+LL)	—	4.6	98.1	No
XGB-300 (CUMUL+LL)	—	310	97.9	No
DF (CNN)	1,465,458	287	94.4	Yes
1D-CNN	5,330,482	236	92.9	Yes
Transformer	420,658	8,733	45.3	No (CPU)

7.4 Limitations

There are several limitations to this work:

1. Our comparison uses direction-only sequences for deep models. Providing deep models with richer input (e.g., signed sizes, timing) could narrow the gap, though that changes the standard deep WF setup.
2. We evaluate on two datasets that are not the standard community benchmarks (Wang14, AWF). Results on our datasets may not directly generalize, and we cannot compare numbers one-to-one with published work. Future work should replicate on Wang14 and AWF-100.
3. We do not include Var-CNN [Bhat et al., 2019] as a separate baseline. Var-CNN was designed for data efficiency and could narrow the gap in the scarce-data regime. We cite Var-CNN but only evaluate the original DF architecture.
4. Our adversarial defenses are simulated (random padding, timing jitter, constant-rate). We do not test against published defense implementations like WTF-PAD [Juarez et al., 2016] or FRONT [Gong and Wang, 2020], which would give more realistic robustness estimates.
5. The Transformer uses truncated $L = 500$ sequences, disadvantaging it relative to CNNs with $L = 5000$.
6. We do not evaluate hybrid approaches that combine engineered features with deep learning, which could achieve the best of both worlds.
7. We do not test concept drift—whether the models remain accurate on traces collected weeks or months after training data.

8 Ethics Statement

All traffic traces used in this study were generated by researcher-initiated visits to public websites using standard web browsers over the Tor network. No real user traffic was observed, intercepted, or analyzed. Our datasets contain only metadata (packet directions and sizes) from our own browsing sessions—no personally identifiable information or third-party communications. The website fingerprinting techniques studied here are dual-use: they can be used by censors to surveil users, but also by defenders to evaluate the strength of privacy-enhancing technologies. We study these techniques to inform better defense design, consistent with the responsible disclosure norms of the privacy research community.

9 Reproducibility Statement

All experiments are fully reproducible. Source code, configuration files, and trained models are available at `[anonymous repository]`. Random seeds are fixed (seed 42) across all experiments, including data splitting, model initialization, and stochastic training. All hyperparameters are specified in YAML configuration files, not hardcoded. We use stratified K-fold cross-validation ($K = 10$) with identical splits across all methods, ensuring that every classifier sees exactly the same training and test partitions. Deep learning experiments were conducted on an Apple M4 Pro (24 GB) using PyTorch with MPS acceleration (CNNs) and CPU (Transformer). Classical experiments used scikit-learn and XGBoost. We report mean $\pm$ standard deviation across folds for all metrics.

10 Conclusion

We compared deep learning and classical feature engineering for encrypted traffic fingerprinting under controlled, identical evaluation conditions. The main result is that Random Forest with CUMUL+LL features (98.1%) consistently outperforms the Deep Fingerprinting model (94.4%) across all experimental conditions. This contradicts the assumption that deep learning has replaced classical methods for traffic analysis. The classical advantage widens under data scarcity (58% vs. 10.7%) and persists in open-world evaluation, scalability tests, and augmented training scenarios.

The choice of *feature representation* matters more than *model architecture* for traffic fingerprinting. Deep learning's apparent superiority in the literature comes from comparing direction-only deep models against classical methods evaluated under different protocols—an apples-to-oranges comparison. When we control for this, feature engineering wins.

Based on these results, we think: (1) WF attack papers should include properly implemented classical baselines with CUMUL+LL features alongside deep learning; (2) defense evaluations should test against both paradigms; (3) hybrid approaches that feed engineered features into neural networks are worth exploring; and (4) data efficiency should be reported as a standard metric alongside peak accuracy.

References

Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-cnn: A data-efficient website fingerprinting attack based on deep learning. In *Proceedings on Privacy Enhancing Technologies*, volume 2019, pages 292–310, 2019.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. Realistic website fingerprinting by augmenting network traces. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 295–309. ACM, 2023.

Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Trafficsliver: Fighting website fingerprinting attacks with

traffic splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2073–2089. ACM, 2020.

Wladimir De la Cadena, Asya Mitseva, Jan Pennekamp, Jens Hiller, Fabian Lanze, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Regulator: A straightforward website fingerprinting defense. In *Proceedings on Privacy Enhancing Technologies*, volume 2022, pages 344–362, 2022.

Xinhao Deng, Qi Li, and Ke Xu. A comprehensive survey of website fingerprinting attacks and defenses in Tor. *arXiv preprint arXiv:2510.11804*, 2025.

Jiajun Gong and Tao Wang. Zero-delay lightweight defenses against website fingerprinting. In *Proceedings of the 29th USENIX Security Symposium*, pages 717–734. USENIX Association, 2020.

Jiajun Gong, Wuqi Zhang, Charles Zhang, and Tao Wang. Surakav: Generating realistic traces for a strong website fingerprinting defense. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (S&P)*, pages 1558–1573. IEEE, 2022.

Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *Proceedings of the 25th USENIX Security Symposium*, pages 1187–1203. USENIX Association, 2016.

Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-Bayes classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pages 31–42. ACM, 2009.

Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. Toward an efficient website fingerprinting defense. In *Proceedings of the 21st European Symposium on Research in Computer Security (ESORICS)*, pages 27–46. Springer, 2016.

Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 255–263. ACM, 2006.

Mohammad Lotfollahi et al. Enhancing encrypted internet traffic classification through advanced data augmentation techniques. *arXiv preprint arXiv:2407.16539*, 2024.

Se Eun Oh, Saikrishna Sunkam, and Nicholas Hopper. Gandalf: Gan for data-limited fingerprinting. In *Proceedings on Privacy Enhancing Technologies*, volume 2021, pages 305–322, 2021.

Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. Website fingerprinting at internet scale. In *Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2016. doi: 10.14722/ndss.2016.23477.

Mohammad Saidur Rahman, Payap Sirinam, Nate Matthews, Kantha Girish Gangadhara, and Matthew Wright. Tik-tok: The utility of packet timing in website fingerprinting attacks. In *Proceedings on Privacy Enhancing Technologies*, volume 2020, pages 5–24, 2020.

Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.

Meng Shen, Kexin Ji, Zhenbo Gao, Qi Li, Liehuang Zhu, and Ke Xu. Subverting website fingerprinting defenses with robust traffic representation. In *Proceedings of the 32nd USENIX Security Symposium*. USENIX Association, 2023.

Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1928–1943. ACM, 2018. doi: 10.1145/3243734.3243768.

Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 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 (CCS)*, pages 1131–1148. ACM, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3319–3328, 2017.

Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *Proceedings of the 26th USENIX Security Symposium*, pages 1375–1390. USENIX Association, 2017.

Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium*, pages 143–157. USENIX Association, 2014.

Xinjie Wang et al. Deep learning and pre-training technology for encrypted traffic classification: A comprehensive survey. *Neurocomputing*, 2024.

A Hyperparameter Details

Table 12: Hyperparameter settings for deep learning models. DF is evaluated under two configurations: Sirinam's original published settings and our tuned settings, to ensure fair comparison.

Parameter	**DF-Sirinam**	**DF-Tuned**	**1D-CNN**	**Transformer**
Input length	5000	5000	5000	500
Batch size	128	128	64	64
Learning rate	0.002	0.002	0.001	0.0005
Epochs (max)	200	30	30	20
Early stopping	30	5	5	5
Optimizer	Adamax	Adam	Adam	Adam
Weight decay	0	10^{-4}	10^{-4}	10^{-4}
Gradient clipping	None	1.0	1.0	1.0
LR scheduler	None	ReduceLROnPlateau	ReduceLROnPlateau	ReduceLROnPlateau
Conv blocks	4	4	3	—
Attention heads	—	—	—	4
d_{model}	—	—	—	128
Encoder layers	—	—	—	3
Dropout	0.1–0.3	0.1–0.3	0.5	0.1
Total parameters	1,465,458		5,330,482	420,658

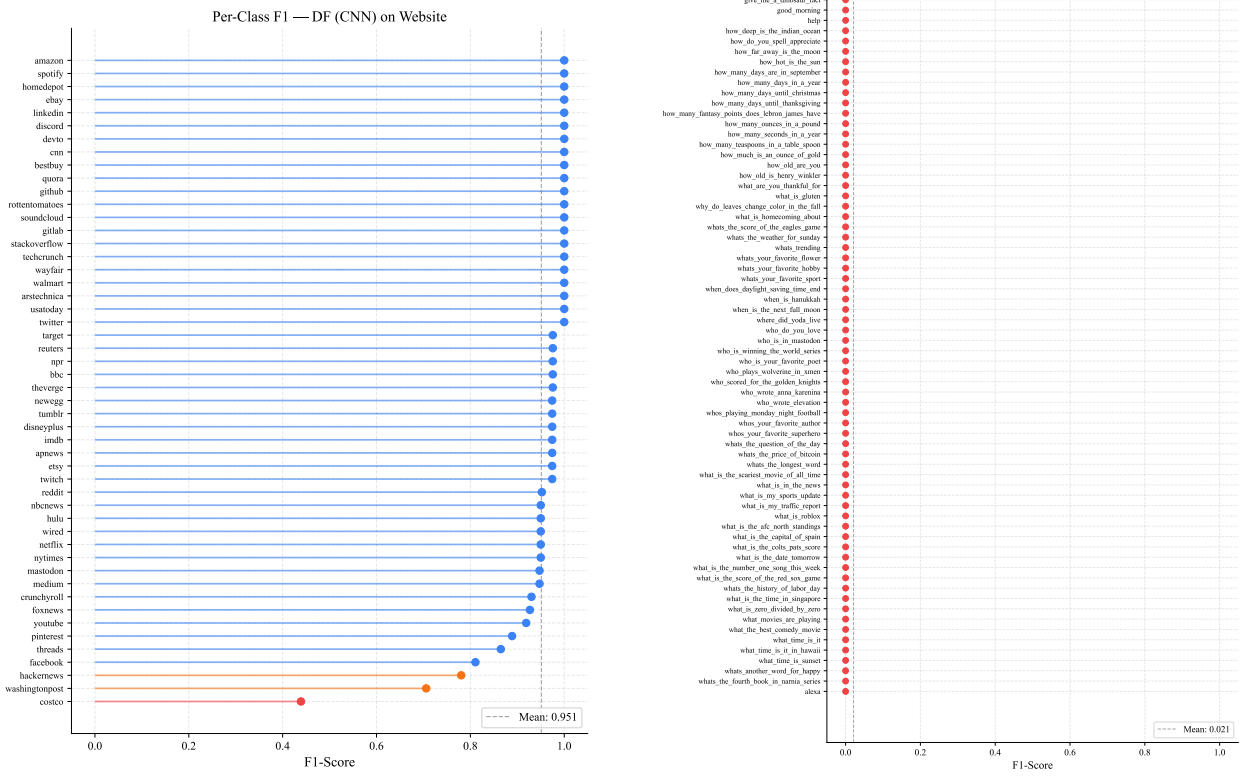
B Classical Feature Engineering Details

Table 13: Feature extraction configurations for classical baselines.

Feature Set	Dimensions	Key Components
CUMUL	139	Cumulative sums (104), packet stats (10), percentiles (13), bursts (5), concentration (3), aggregate stats (4)
LL	774	Size histograms (3×256 bins), stats (6)
Combined	913	CUMUL + LL concatenated

C Per-Class F1 Scores

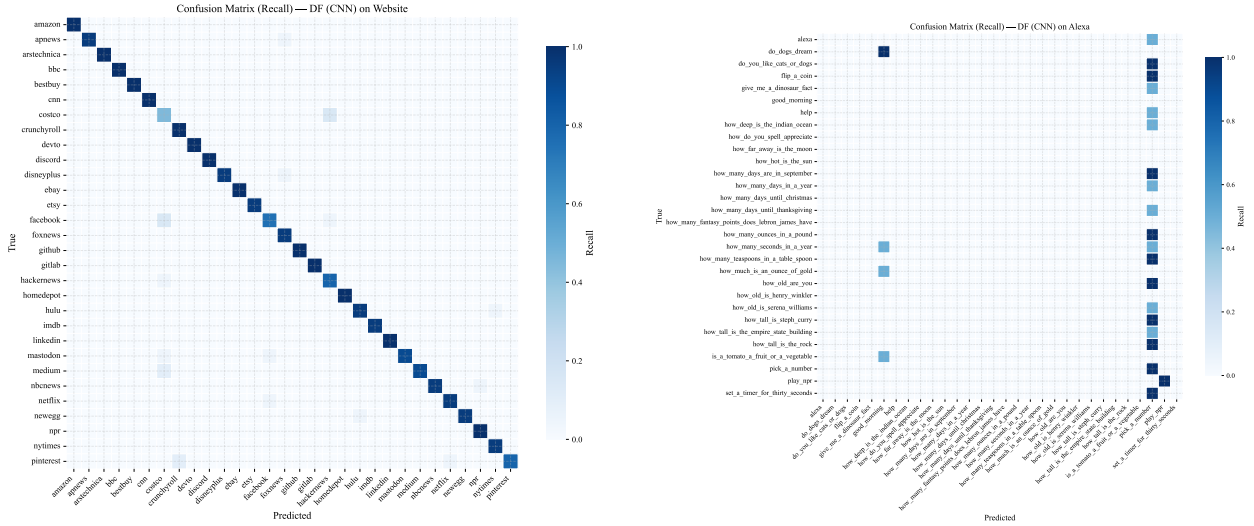
D Confusion Matrices



(a) DF on Website

(b) DF on Alexa

Figure 8: Per-class F1 scores for the DF model on both datasets.



(a) DF on Website

(b) DF on Alexa

Figure 9: Confusion matrices for the DF model. The Website matrix shows a strong diagonal; the Alexa matrix shows near-uniform confusion.