Cease at the Ultimate Goodness: Towards Efficient Website Fingerprinting Defense via Iterative Mutual Information Minimization

Rong Wang[†], Zhen Ling[†*] , Guangchi Liu[†], Shaofeng Li[†], Junzhou Luo[†‡] and Xinwen Fu[§]
[†]Southeast University, Email: {junowang, zhenling, gc-liu, shaofengli, jluo}@seu.edu.cn
[‡]Fuyao University of Science and Technology
[§]University of Massachusetts Lowell, Email: xinwen_fu@uml.edu

***Abstract*—In response to growing online privacy threats, the Tor network offers essential protection against surveillance by routing traffic through a decentralized, encrypted infrastructure. However, Website Fingerprinting Attacks (WFA) present a formidable challenge to Tor's anonymity. This paper introduces FRUGAL, a traffic obfuscation method that leverages the mutual information (MI) reduction between website traffic and labels as an optimization goal, advancing a novel perspective for Website Fingerprinting Defense (WFD). By strategically injecting dummy packets at positions within website traffic that contribute most to cumulative MI reduction, FRUGAL achieves notable performance compared to state-of-the-art (SOTA) defense mechanisms. It effectively reduces attack success rates (ASR) across diverse attack models while maintaining minimal bandwidth overhead (BWO) and mitigating the impact of adversarial training. Extensive experiments validate the efficacy of FRUGAL across a comprehensive set of scenarios, including closed-world, open-world, and real-world simulation settings. For example, in the closed-world setting, FRUGAL reduces the ASR of the DF model to 2.68% with a 30% BWO, substantially outperforming previous SOTA defenses, such as Palette (11.54% with 87% BWO). When the BWO of FRUGAL is increased to a comparable level of 80%, the ASR further drops below 1%, demonstrating significant resilience by remaining low at 9.42% even after adversarial training, compared to 20.27% for Palette. This work not only introduces a fresh perspective on WFD research but also establishes FRUGAL as a robust and universal defense framework against WFA.**

I. INTRODUCTION

Tor is designed to protect the anonymity of user communications by routing their website traffic through globally distributed Tor nodes [10], [40], achieving decentralized and encrypted communication. This setup helps conceal users' online activities, making it challenging for others to track or monitor users' internet behavior. However, it is still vulnerable to local eavesdroppers through *Website Fingerprinting Attacks* (WFA) [37], [2], [1], [33]. By analyzing patterns in the size and direction of traffic packet traces—known as 'website fingerprints'—attackers can infer which specific website the user is visiting. With recent advances in deep learning, these attacks have posed a serious challenge to Tor's privacy protections.

To address this challenge, *Website Fingerprinting Defense* (WFD) techniques have been developed to counter WFA by disrupting an attacker's ability to identify websites through traffic obfuscation methods. Existing website fingerprinting defenses can be broadly classified into two categories. The first category comprises *feature-morphing-based* defenses [31], [14], [25], [24], which aim to alter a website's traffic profile to resemble a target website, thereby causing the classifier to misclassify the former as the latter. The second category includes *feature-suppression-based* defenses [36], [3], [4], [20], [13], which work by homogenizing the traffic features of all websites, rendering the classifier unable to differentiate between them. While both categories have made progress, challenges remain that limit their effectiveness and robustness in dynamic adversarial environments.

(C1) Attack Model Agnostic: Feature-morphing-based defense methods typically operate under the assumption that the target attack model remains static and accessible, allowing defensive strategies to adjust based on the attack model's outputs. However, this reliance poses significant limitations when the attack model is either inaccessible or evolves continuously as adversaries adapt to countermeasures. Moreover, these defense methods often exhibit poor generalization across different attack models, further restricting their effectiveness in dynamic and diverse adversarial settings.

(C2) Efficiency of Bandwidth Overhead: While feature-suppression-based methods offer universal protection by reducing the Attack Success Rate (ASR) of various attack models through homogenizing website features, they inevitably lead to excessive and uncontrollable Bandwidth Overhead (BWO). A defense mechanism capable of maximizing ASR reduction while adhering to predefined bandwidth limits remains an unrealized goal. Such a solution is especially critical in environments with varying bandwidth constraints, where efficiency and adaptability are paramount.

(C3) Adversarial Training Resilience: Prior work [22] shows that despite reduced attack accuracy, defended traffic often retains high Mutual Information (MI) with original labels, aiding website identification by attackers. This phenomenon, known as *information leakage* [22], highlights why many defenses struggle to remain effective against adversarially trained attack models. Adversarial training retrains attack models on defended traffic, exploiting residual patterns that defenses cannot fully hide, thus weakening WFD effectiveness in post-adversarial settings.

To address the aforementioned challenges, we propose FRUGAL, a defense framework that shifts the focus from deceiving

* Corresponding author: Prof. Zhen Ling of Southeast University, China.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240786
www.ndss-symposium.org

specific attack models to fundamentally eliminating a website's traffic fingerprint. Our approach centers on minimizing the MI between website traffic features and their corresponding labels, using MI reduction as the core optimization objective. From an information-theoretic perspective, reducing the MI between traffic features and labels is, by definition, equivalent to increasing the information entropy (uncertainty) of the labels conditioned on the traffic features. As a result, by maximizing MI reduction, our approach directly increases the label uncertainty with respect to the traffic features, thereby maximizing the attacker's potential classification error. We model this process as a Markov Decision Process (MDP) and employ reinforcement learning (RL) to solve it. In WFD research, MI is frequently used to quantify the amount of information shared between website traffic and its original labels, making it a key metric for evaluating the effectiveness of WFD strategies. Most existing approaches focus on reducing the ASR as the primary objective, with MI treated only as a performance indicator. In contrast, FRUGAL is novel in directly optimizing for MI reduction, setting our approach apart from prior work.

FRUGAL addresses **C1** by generating modified website traffic that minimizes MI with its original label, thereby hindering attack models from accurately inferring labels. For a given website traffic trace, FRUGAL leverages a reinforcement learning algorithm to determine an efficient policy for injecting dummy packets at key positions, maximizing MI reduction and achieving effective traffic obfuscation. Crucially, FRUGAL's emphasis on minimizing MI without relying on knowledge of specific attack models ensures robust and adaptable protection against evolving threats.

To tackle **C2**, FRUGAL performs dummy packet injection iteratively. In each iteration, a small set of dummy packets is injected to maximize cumulative MI reduction. By configuring the iteration count as a hyperparameter, FRUGAL enables fine-grained control over bandwidth overhead, ensuring efficiency across diverse network conditions.

Finally, to overcome **C3**, FRUGAL dynamically adjusts its dummy packet injection positions conditioned on the resulting traffic patterns from previous steps. This adaptive strategy ensures that the actor trained in FRUGAL consistently targets the most informative residual patterns within the traffic trace, effectively diminishing the ASR even for adversarially trained attack models.

We implemented FRUGAL and conducted extensive experiments using public WF datasets [37]. Our evaluation spans closed-world, open-world, one-page settings, and real-world simulations, including scenarios with adversarially trained attack models. We benchmark FRUGAL against state-of-the-art (SOTA) defenses, evaluating both its defensive effectiveness and bandwidth overhead (BWO). The results demonstrate that FRUGAL consistently outperforms existing SOTA methods by achieving stronger defense performance with significantly lower BWO, while also maintaining robustness against adversarial training. For instance, in the closed-world setting, FRUGAL reduces the ASR of DF [37] to 2.68% and RF [35]

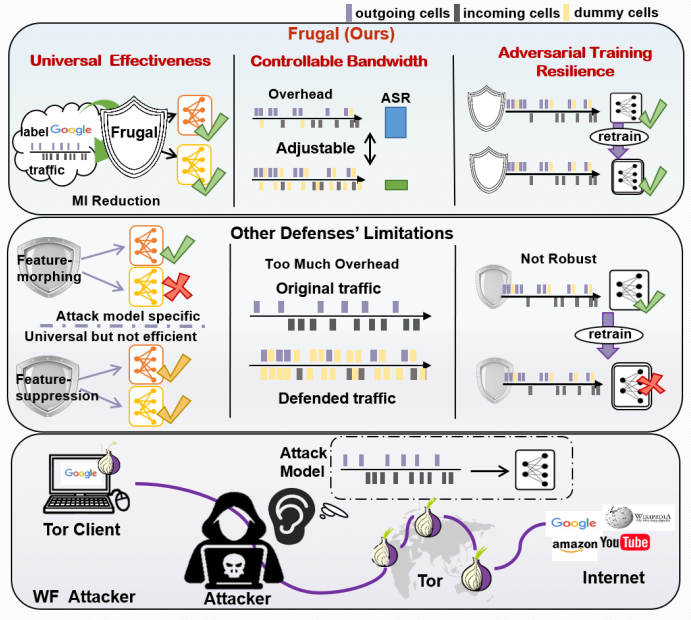


Fig. 1. **Threat Model of FRUGAL**

to 12.7% with just 30% BWO, substantially outperforming previous SOTA defenses, such as Palette[36], which yields 11.54% and 46.43% ASR for DF and RF, respectively, with 87.17% BWO. When the BWO of FRUGAL is increased to a comparable level of 80%, the ASR drops further (below 2% for DF and 8.12% for RF), and remains robustly low after adversarial training (9.42% for DF and 18.2% for RF), compared to 20.27% and 46.43% for Palette under the same conditions. In a more realistic real-world simulation, the effectiveness of FRUGAL is further confirmed, where its online implementation (FRUGAL-online) reduces the ASR of DF and RF to just 4.69% and 14.1%, respectively, with 30% BWO.

In summary, with the introduction of FRUGAL, we aim to address the following research questions: **RQ1:** What does efficiently defended traffic that achieves (1) attack model agnosticism, (2) bandwidth overhead efficiency, and (3) resilience against adversarial training look like? **RQ2:** How does FRUGAL ensure that these characteristics are effectively met? Our contributions are outlined as follows.

- **Novel WFD Research Perspective**: To our knowledge, FRUGAL is the first WFD framework to leverage the MI reduction between website traffic and corresponding labels as an optimization target, providing a new direction for advancing WFD research.
- **Precise Bandwidth Overhead Control**: FRUGAL is the first to introduce a method ensuring efficient WFD under precise bandwidth overhead limits, enabling flexible deployment across scenarios with diverse bandwidth constraints.
- **Mitigation of Adversarial Training**: We theoretically demonstrate and implement an effective mechanism to counter adversarial training, significantly enhancing the robustness of FRUGAL.

- **State-of-the-Art Performance**: Extensive experimental results demonstrate that traffic defended by FRUGAL effectively counters website fingerprinting attacks, achieving state-of-the-art performance and establishing a new benchmark for WFD research.

II. BACKGROUND

In this section, we provide the necessary background on website fingerprinting, deep reinforcement learning, and mutual information.

A. Website Fingerprinting

To achieve online anonymity, the Tor network [10] randomly selects three volunteer nodes, designated as the guard node, middle node, and exit node, to establish a circuit. In a circuit, each node can only view the preceding and following nodes. Through this circuit, user web traffic is encapsulated into fixed-size encrypted packets known as *Tor cells*, which are then transmitted across the network.

The rise of Website Fingerprinting attacks [37], [2], [1], [33] has posed a significant challenge to the Tor network's anonymity. Website fingerprinting is a traffic analysis technique that monitors data flow between users and their guard node. In WFA, Tor traffic is parsed as a sequence of +1' or -1' values [43], where +1' represents upstream cells from the user to the server, and -1' represents downstream cells from the server to the user. Based on this sequence, the attacker extracts distinct features such as the number of Tor cells, direction, timing and cumulative features, which form unique traffic fingerprints for different websites. These fingerprints can be utilized to identify specific websites.

Most current defenses [36], [18], [34] attempt to thwart WF attacks by inserting dummy packets and/or delaying data packets. The injection of numerous dummy packets results in considerable BWO, which in turn increases network load and potentially leads to congestion. On the other hand, delaying data packets effectively hides website packet timing information but prolongs page load times, harming the user experience. Striking a balance between overhead and performance remains an urgent challenge.

B. Deep Reinforcement Learning

In RL, the agent iteratively interacts with the environment to generate the 5-tuple $(s, a, r, s_{next}, s_{terminal})$. Here, s is the current state in the entire state set $\mathcal{S}$, a is the chosen action in the action set $\mathcal{A}$, r is the reward received after taking action a from the environment, and s_{next} is the next state after the action is executed, $s_{terminal}$ represents the terminal state indicating the termination of the iteration. The goal of agents is to maximize long-term cumulative rewards by learning an efficient policy. This is achieved by updating its Q-function, which represents the expected cumulative reward (also known as Q-value) for taking action a in state s. The Q-function guides the agent toward maximizing its cumulative reward. In recent cutting-edge deep reinforcement learning (DRL), *e.g.,* Deep Q Network (DQN) [27], Double Deep Q Network (DDQN) [39], and Soft Actor-Critic (SAC) [15], neural networks are used as policy networks to approximate the agent's strategy. These approaches are highly efficient for decision-making in complex environments.

C. Mutual Information

Mutual information (MI) between two random variables $\mathbf{x}$ and $\mathbf{y}$, denoted as $I(x; y)$, quantifies the amount of information shared between them, effectively measuring how much knowledge one variable reveals about the other. Specifically, $I(x; y)$ can be expressed as:

$$\begin{aligned} I(x; y) &= D_{\mathrm{KL}}(p(x, y) \| p(x)p(y)) \\ &= H(y) - H(y \mid x). \end{aligned} \tag{1}$$

In Equation (1), $I(x; y)$ is defined as the Kullback–Leibler (KL) divergence between the joint distribution $p(x, y)$ and the product of their marginal distributions $p(x)p(y)$. However, directly computing KL-divergence is infeasible, as it requires closed-form expressions for $p(x, y)$, $p(x)$, and $p(y)$. Alternatively, as shown in Equation (1), $I(x; y)$ can be reformulated as the change in the entropy of y when x is introduced, i.e., $H(y) - H(y \mid x)$. In this way, $I(x; y)$ can be solved as $H(y)$ and $H(y \mid x)$ can be estimated using variational techniques.

Furthermore, when considering the effect of a third variable z, the concept extends to Conditional Mutual Information (CMI), which is defined as:

$$\begin{aligned} I(x; y \mid z) &= D_{\mathrm{KL}}\left(p(x, y \mid z) \| p(x \mid z)p(y \mid z)\right) \\ &= H(y \mid z) - H(y \mid x, z). \end{aligned} \tag{2}$$

Here, $I(x; y \mid z)$ measures the mutual dependence between x and y, conditioned on z. Notably, in this paper, Equation (2) is further extended (details are presented in Appendix B) as

$$H\left(y \mid \boldsymbol{x} \cup \boldsymbol{x}_i\right) = H\left(y \mid \boldsymbol{x}\right) - I\left(\boldsymbol{x} \cup \boldsymbol{x}_i; y \mid \boldsymbol{x}\right), \tag{3}$$

where $\boldsymbol{x}$ denotes the website-traffic features and $\boldsymbol{x}_i$ denotes a dummy packet injected at the i-th position. Equation (3) indicates that the information entropy (uncertainty) of y conditioned on $\boldsymbol{x}$, i.e., $H(y \mid \boldsymbol{x})$, can be further increased by injecting $\boldsymbol{x}_i$ into $\boldsymbol{x}$, where the increase is quantified by $I\left(\boldsymbol{x}_i \cup \boldsymbol{x}; y \mid \boldsymbol{x}\right)$. Therefore, by strategically disrupting the patterns in the traffic features $\boldsymbol{x}$ through injecting an $\boldsymbol{x}_i$ that minimizes $I(\boldsymbol{x}_i \cup \boldsymbol{x};\, y \mid \boldsymbol{x})$, the resulting entropy $H(y \mid \boldsymbol{x} \cup \boldsymbol{x}_i)$ (uncertainty of the label y given $\boldsymbol{x} \cup \boldsymbol{x}_i$) is maximized, thereby making it more difficult for a classifier to accurately predict y from $\boldsymbol{x} \cup \boldsymbol{x}_i$.

III. WEBSITE FINGERPRINTING DEFENSE

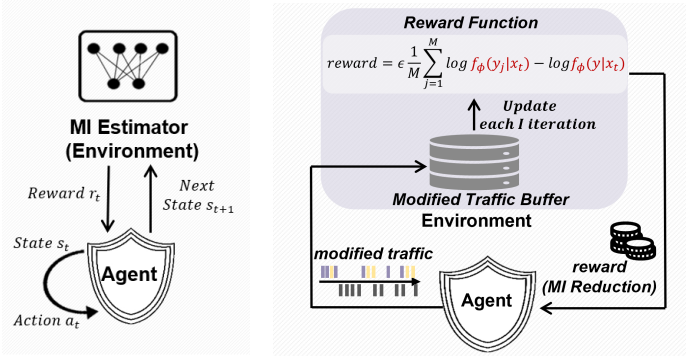
A. Basic Idea

The core idea of FRUGAL is to minimize the MI between website traffic and its labels. By injecting dummy packets into the original traffic to maximize MI reduction between the modified traffic and its corresponding labels, FRUGAL establishes a universal and robust defense mechanism that is independent of any specific attack model **(C1)**. FRUGAL performs dummy packet injection iteratively based on a learned injection policy.

The injection policy, which determines the effective positions for packet insertion, is learned using the SAC algorithm. By pre-defining the number of iterations, FRUGAL ensures precise control over BWO **(C2)**. Finally, FRUGAL counters attacks based on adversarial training by eliminating specific positions critical for distinguishing between different traffic patterns. As a result, FRUGAL demonstrates robustness against adversarial training **(C3)**.

Technical Challenges: **I.** The first challenge lies in the computational complexity of employing MI as an optimization target. Existing MI estimation methods [22], [35], which primarily rely on hand-crafted features, demand significant domain expertise and high computational costs. Consequently, they are unsuitable for direct use as optimization objectives. **II.** The high dimensionality of network traffic traces introduces a significant challenge commonly referred to as the "curse of dimensionality". More specifically, searching for the most effective positions to inject dummy packets through brute force in the entire action space is infeasible. In addition, such complexity substantially impedes the ability of reinforcement learning algorithms to learn an effective injection policy. **III.** The injection of dummy packets, while designed to obfuscate traffic patterns and reduce MI, inevitably causes a distribution shift from the original website traffic. This modification continuously perturbs the MI between the remaining traffic features and the corresponding labels, thus progressively degrading the accuracy of the neural network–based MI estimator, which is trained under the original traffic distribution.

Solution: I. To construct an efficient MI estimator for the optimization process conducted via reinforcement learning (as outlined in the first technical challenge), FRUGAL employs the Contrastive Log-ratio Upper Bound (CLUB) estimator [5], which uses a neural network to approximate the upper bound of MI. CLUB serves as the core component of FRUGAL's reward function, guiding the learning of the efficient dummy packet injection policy (Equation (14)). **II.** To mitigate the curse of dimensionality (the second technical challenge), a Convolutional Neural Network (CNN)-based encoder is employed to learn compact representations of the packet position information in the high-dimensional traffic data, facilitating coarse-grained position selection for dummy packet injection. **III.** To address the distribution drift challenge of MI introduced by dummy packet injection (the third challenge), FRUGAL adopts Conditional Mutual Information (CMI) [7] as the core optimization objective to determine efficient injection positions. Specifically, positions are selected based on their potential to maximize CMI reduction, where CMI represents the MI between the traffic and its labels, conditioned on the traffic modified in previous iterations. The theoretical analysis (in Section B) demonstrates that, with a dynamically updated CMI estimator, a greedy position selection strategy ultimately achieves the global maximum MI reduction across the entire injection process.



(a) **Interaction between Agent and Environment.** (b) **Process of Reward Function Updating.**

Fig. 2. **Overview of FRUGAL**

B. Threat Model

In this study, we adopt the standard assumption that an adversary is positioned between a user and their guard node, as shown in Figure 1. We assume that the adversary engages in passive monitoring, collecting and analyzing the traffic between the user and their guard node to construct a dataset for model training. The term 'passive' indicates that the adversary cannot alter the user's traffic. Moreover, it is assumed that the user accesses only one website at a time, ensuring that the traffic captured by the adversary represents a complete session for a single website. The adversary then uses this dataset to train a deep learning model for a WFA, with the goal of identifying the websites visited by the user.

Website fingerprinting scenarios are typically classified into two categories: closed-world and open-world. In the closed-world scenario [6], it is assumed that users access only a limited set of websites, which are known as monitored sites. The adversary trains a model on the traffic from these monitored sites to identify which one the user is visiting. In the open-world scenario, users may visit both monitored sites and any number of unmonitored sites. The adversary collects traffic from a limited subset of unmonitored sites and combines it with the monitored dataset to train a model. This model is then used to determine whether the user is visiting a monitored site and, if so, which specific one.

C. Training Framework

Leveraging the SAC-based RL technique [15], the training framework of FRUGAL comprises an *agent* and an *environment*, where the agent learns through iterative *interaction* with the environment.

Agent: The agent, named FRUGAL, contains a traffic encoder to transform traffic traces into compact *state* representations, and a policy network, which is responsible for identifying efficient positions in website traffic and injecting dummy packets to achieve MI reduction. FRUGAL takes a website traffic trace as input, performs an *action* by determining the efficient injection positions, and modifies the traffic accordingly before passing it to the environment.

Environment: The environment represents the external system with which the agent interacts and learns. It is implemented as an MI estimator, which functions as the reward mechanism. The estimator receives the modified traffic from the agent, evaluates the MI reduction, and provides a reward signal to the agent. It also monitors the packet injection process and terminates the interaction for a given traffic instance upon reaching the maximum iteration limit.

Interaction: During the t-th interaction between FRUGAL and the environment, FRUGAL receives an input traffic x_t and generates a state s_t, a compact representation of the traffic. Using the policy network, FRUGAL selects an action a_t based on s_t, identifying efficient positions for dummy packet injection and executing the injection to produce the modified traffic x_{t+1}. x_{t+1} is the input of the agent for the next iteration. Concurrently, the environment evaluates x_{t+1} using the MI estimator and computes the reward r_t, which is fed back to FRUGAL. FRUGAL utilizes the reward to guide the refinement of the policy network using the SAC algorithm. More details are depicted in Figure 2(a).

D. Online Defense

Although the agent (FRUGAL), trained via the framework described in Section III-C, can identify the globally effective dummy packet injection positions for a given website's traffic, it cannot be directly deployed in an online defense setting because it requires access to the complete traffic trace in advance, which is unavailable during a live browsing session. To enable practical online deployment, we derive an online defense solution from FRUGAL, described in Section IV-C and referred to as FRUGAL-online. Given a partially observed packet sequence and the corresponding website label, FRUGAL-online determines, in real time, whether to inject dummy packets and how many to insert immediately following the observed sequence.

IV. DESIGN DETAILS OF FRUGAL

In this section, we first detail the design of FRUGAL and its training framework, which is composed of two main components: the *agent* and the *environment*. We then introduce the design of FRUGAL-online as the online defense solution.

A. Agent

The agent, referred to as FRUGAL, is designed to iteratively inject dummy packets at effective positions to maximize MI reduction. As shown in Figure 3, the agent comprises two core components: (1) a Traffic Encoder and (2) a Policy Network. The Traffic Encoder is pre-trained using a supervised learning approach, while the Policy Network is trained using the RL algorithm, i.e., SAC. This section provides a detailed explanation of the architectures and respective training processes for both components.

1) Traffic Encoder: Given a website traffic trace as input, the agent (FRUGAL) identifies the effective positions to inject dummy packets based on a compact representation, rather than directly processing the original traffic. This compact

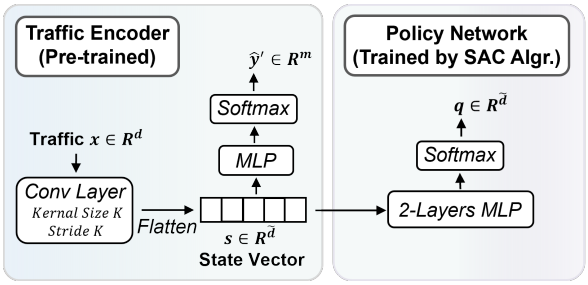


Fig. 3. **The Architecture of Agent.** Our agent comprises two core components: a Traffic Encoder and a Policy Network.

representation, generated by an encoder pre-trained on the input traffic, mitigates the "curse of dimensionality" by greatly shrinking the search space while preserving the information needed to determine injection positions.

As shown in the Figure 3, this traffic encoder, which is implemented as a single-layer convolutional neural network (CNN) following a softmax layer, can be expressed as

$$\begin{aligned} s =& \mathrm{CNN}(x), \\ \hat{y}' =& \mathrm{Softmax}(\mathrm{MLP}(s)), \end{aligned} \tag{4}$$

where $x \in \mathbb{R}^d$, $s \in \mathbb{R}^{\tilde{d}}$, $\hat{y}' \in \mathbb{R}^m$. As shown in Figure 3, the encoder applies a set of learnable filters that slide across x. By setting the convolution kernel size and stride to K, each element at the i-th index of s corresponds to a continuous segment of K elements in x, spanning indices $x_{(i-1)K}$ to x_{iK-1}. Notably, the dimension of s, i.e., $\tilde{d}$, can hence be expressed as

$$\tilde{d} = \frac{d-K}{K} + 1 = \frac{d}{K}, \tag{5}$$

where d is the dimension of traffic x.

By setting the dimension of the output normalized logits $\hat{y}'$ to match the number of classes m, the encoder can be trained in a supervised manner using the cross-entropy loss [8] between the prediction $\hat{y}'$ and its corresponding ground-truth label y. Once trained, the encoder processes trace x to generate state s, which is $1/K$ of the length of the original traffic x. The state representation s serves as input to the policy network responsible for determining the effective injection positions of dummy packets.

2) Policy Network: The policy network, also known as the actor network in the SAC algorithm, is designed to iteratively identify packet positions with the highest potential for information leakage. It then injects dummy packets into these positions to minimize the MI of the modified traffic. As shown in the bottom of Figure 3, the policy network π_θ consists of a 2-layer MLP followed by a softmax layer, which can be expressed as

$$q_t = \mathrm{Softmax}(\mathrm{MLP}_2(s_t)), \tag{6}$$

where $q_t, s_t \in \mathbb{R}^{\tilde{d}}$, MLP_2 denotes a 2-layer MLP. Equation (6) indicates that the policy network takes the *state* (denoted as s_t) generated by the traffic encoder as input and produces a logits vector (denoted as q_t), also known as the *Q-value vector* in the

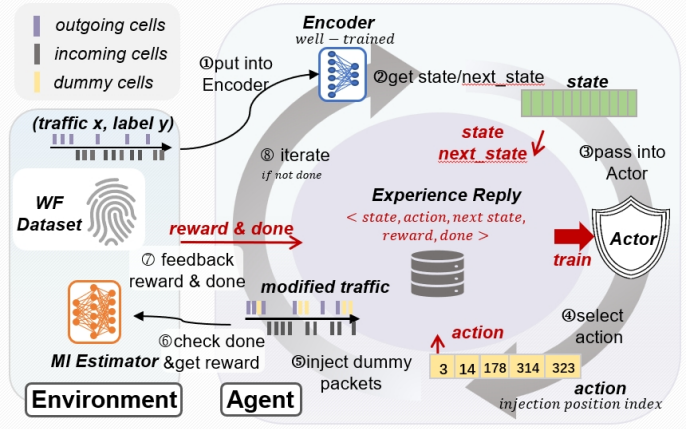


Fig. 4. **Training Process of FRUGAL.** The grey arrows illustrate a full iteration of experience collection, while the red arrows denote procedures associated with the Experience Replay Buffer.

context of RL. Each value in q_t represents the probability of selecting a corresponding position in the input traffic. FRUGAL takes an *action* (denoted as a_t) by injecting dummy packets into the positions corresponding to the top-n values in q_t. The number of dummy packets injected at each selected position is sampled from a Poisson distribution. This approach intentionally preserves policy stochasticity, a common technique in reinforcement learning aimed at enhancing robustness and improving generalization [30], [32]. The detailed process of the action selection is outlined in Algorithm 1. The modified traffic x_{t+1}, containing these injected packets, becomes the agent's output. This trace, x_{t+1}, is used to compute the *next state* (denoted as s_{t+1}) in the subsequent iteration. Simultaneously, the modified traffic is evaluated within the environment, producing a *reward* (denoted r_t) that guides the training of the agent. At the end of each step, the episode is checked for termination (*done*, denoted as d_t) based on whether the bandwidth overhead—defined as the ratio of injected packets to the original traffic size—has reached a predefined threshold. This configurable BWO threshold enables FRUGAL to adapt to various deployment scenarios, accommodating diverse BWO constraints effectively.

Unlike the traffic encoder, the policy network π_θ within the agent is trained using the SAC algorithm, an RL approach. In this setup, the policy network π_θ functions as the *Actor* module, where its parameters θ are updated by interacting with a *Critics* module, as illustrated in Figure 5. Specifically, a detailed training process is described in Algorithm 2. As shown in Algorithm 2, it begins by initializing an experience replay buffer to store experience tuples in the form $\langle s_t, a_t, s_{t+1}, r_t, d_t \rangle$ collected across all iterations. Training starts when the number of experience tuples in the replay buffer surpasses a predefined threshold. Once training begins, a batch of experience tuples B is sampled from the replay buffer at the end of each iteration. For each tuple $\langle s_t, a_t, s_{t+1}, r_t, d_t \rangle \in B$, the policy network π_θ (Actor) processes s_t to produce a logits vector q_t. Collectively, these logits form a set of Q-value vectors, denoted as Q, corresponding

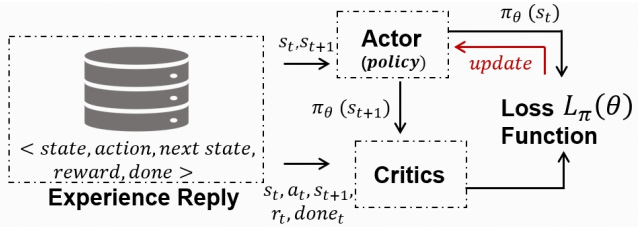


Fig. 5. **Data Flow during Training.** The red arrow represents the last step of the training process is to update the parameters of Actor.

to the batch B.The batch B and its associated Q-value set Q are passed to the Critics module, which computes a Critics-based Q-value vector $\hat{Q}$. A loss function $L_\pi(\theta)$ is then applied to measure the discrepancy between Q and $\hat{Q}$. The gradient of this loss function with respect to θ is used to update the parameters of the policy network π_θ. Further details about the SAC algorithm can be found in Appendix A.

Algorithm 1 Actor Network Function

Initial: Actor network π_θ;
Input: Website traces (x_t);
    Well-trained state encoder $F_e(.)$;
    Number of injected positions n
Output: Modified traffic x_{t+1};
    Action a_t

s_t = $F_e(x_t)$ //Get state
probs = $\pi_\theta(s_t)$
a_t = RandomSample(probs, n) //Sample n positions
l_t = get_traffic_length(x_t)
LOOP:
    IF all indices in a_t are less than l_t **THEN BREAK**
    ELSE
        Set probs[index] = *-infinity*
        Rechoose a_t with updated probs
    END IF
injection_counts = POISSON_SAMPLE(probs[a_t])
x_{t+1} = get_modified_traffic(x_t,a_t)
return x_{t+1}, a_t

B. Environment

1) Reward Function: In the FRUGAL training framework, the environment functions as an MI estimator, which we implement as the CLUB [5] estimator, denoted $\mathrm{I}_{CLUB}(x, y)$. This estimator takes a website traffic trace x and its associated label y as input to compute an upper bound on the MI between them. The full derivation of CLUB is detailed in Appendix C.

Building upon CLUB, the reward function that drives the learning of the injection policy in FRUGAL is defined in Equation (7). In this equation, x_t indicates the traffic evaluated at the t-th iteration, ϵ represents a weight coefficient, and M denotes the number of monitored websites in the dataset. The

Algorithm 2 Training Process of FRUGAL

Initial: Initialize the critic networks Q_{ω_1} and Q_{ω_2}, and the actor network π_θ using random network parameters ω_1, ω_2, and θ;
    Initialize Replay Buffer $\mathcal{B}$;
    Copy parameters $\omega_1^- \leftarrow \omega_1$ and $\omega_2^- \leftarrow \omega_2$ to initialize the target critic networks $Q_{\omega_1^-}$,$Q_{\omega_2^-}$
Input: Website traffic and labels $x \in X, y \in Y$;
    Well-trained state encoder $F_e(.)$;
    Environment env, Batch size N;
    Terminal timesteps T, Target update interval I;
Output: modified traces x_T

$t \leftarrow 0$
LOOP:
    IF $t \geq T$ **THEN BREAK END IF**
    $s_t = F_e(x_t)$ // Get current state
    $a_t = \pi_\theta(s_t)$ // Sample action from policy
    $x_{t+1} =$ get_modified_traffic(x_t, a_t)
    $r_t = env$.get_reward(x_{t+1})
    $done_t = env$.is_terminal(x_{t+1})
    $s_{t+1} = F_e(x_{t+1})$ // Get next state
    Store $\langle s_t, a_t, r_t, s_{t+1}, done_t \rangle$ in $\mathcal{B}$
    IF buffer size is larger than N **THEN**
        Sample a minibatch from $\mathcal{B}$
        Update critic networks $Q_{\omega_1}, Q_{\omega_2}$
        Update actor network π_θ
        Update entropy coefficient α
        IF $t \mod I == 0$ **THEN**
            Update target weights $\omega^- \leftarrow \omega$
        END IF
    END IF
    $t \leftarrow t + 1$
END LOOP
RETURN x_T

function f_ϕ is a neural network classifier pre-trained directly on raw website traffic and its corresponding labels.

$$R(x_t) = -\log f_\phi\left(y \mid x_t\right) + \epsilon \cdot \frac{1}{M} \sum_{j=1}^{M} \log f_\phi\left(y_j \mid x_t\right), (y_j \neq y). \tag{7}$$

Conceptually, Equation (7) consists of two components. The first component, $-\log f_\phi(y \mid x_t)$, minimizes the log-likelihood that the traffic x_t aligns with its original label y, effectively obfuscating the label. The second component increases the likelihood that x_t is associated with labels from other monitored websites, introducing ambiguity.

This reward function offers two key advantages. First, by leveraging the neural network f_ϕ, it bypasses the need for hand-crafted feature engineering and domain expertise, making it both computationally efficient and easy to update. Second, it employs the CLUB estimator's MI upper bound as a tractable objective, allowing us to directly optimize for MI minimization in a way that is fully aligned with our framework's goals.

2) Dynamic Feature Elimination: As illustrated in Equation (7), f_ϕ is pre-trained on the original website traffic, under the assumption that the traffic distribution, $p(x_t)$, remains static. However, this assumption is progressively violated as dummy packets are injected, inducing a significant distribution shift. As a consequence, the pre-trained MI estimator, $\mathrm{I}_{CLUB}(x, y)$, becomes increasingly inaccurate. This "estimator drift" undermines the injection policy's ability to effectively target the most informative residual patterns as the injection process continues. Ultimately, this vulnerability allows an attacker to exploit these residual patterns via adversarial training, thereby compromising the overall effectiveness of FRUGAL.

To counter the adversarial training issue, we integrate Dynamic Feature Elimination (DFE) into the training process of FRUGAL, drawing inspiration from advancements in Dynamic Feature Selection (DFS) [7]. DFE is implemented by periodically updating the classifier f_ϕ, as defined in Equation (7), every I iterations, where I is a hyperparameter controlling the update frequency. As depicted in Figure 2(b), during each update cycle, the modified traffic samples and their corresponding label from the most recent I iterations, i.e., $\{(x_i, y) \mid i \in [t - I, t]\}$, are collected to fine-tune f_ϕ. Specifically, we calculate the cross-entropy loss between these modified traces and their label y to update the classifier's parameters ϕ. This process effectively transforms the environment from a static MI estimator into a CMI estimator. This allows the environment to accurately estimate the MI of the current traffic, conditioned on all modifications (dummy packet injections) made in previous iterations.

Through interaction with the CMI estimator, the policy network within FRUGAL is guided to iteratively identify and inject packets into positions that maximize CMI reduction during each iteration. A detailed derivation of how the CMI estimator facilitates this process is provided in Appendix B. Additionally, by greedily injecting dummy packets at positions estimated to yield the greatest CMI reduction (as guided by Equation (7)), the process is guaranteed to maximize the cumulative MI reduction over the entire injection process (see proof in Theorem 1 of Appendix B). This method lets FRUGAL dynamically adapt its policy, identifying and eliminating residual patterns as more dummy packets are injected.

C. Online Defense

FRUGAL learns an efficient offline policy but cannot be directly deployed in the real world, as it requires the complete traffic trace in advance. To enable practical online defense, we distill this policy into a set of pre-computed, website-specific injection patterns, which we call FRUGAL-online. These patterns are indexed by website labels, facilitating rapid, on-the-fly lookup and deployment. An overview of FRUGAL-online's workflow is shown in Figure 6.

To develop FRUGAL-online, we first use FRUGAL to generate defended traffic by applying it to the original traces of each monitored website offline. For each original traffic

instance $\boldsymbol{x}$, we construct a corresponding injection pattern $\mathbf{x} \in \mathbb{R}^{1\times(d+1)}$, where d denotes the maximum traffic length among all traces.The vector $\mathbf{x}$ records the number of injected packets at each position in $\boldsymbol{x}$. Specifically, $\mathbf{x}[i]$ indicates the number of dummy packets inserted between the i-th and $(i+1)$-th packets; $\mathbf{x}[0]$ records the number injected before the first packet; and $\mathbf{x}[|x|]$ records the number injected after the last packet. For all i such that $|x| < i \leq d$, we set $\mathbf{x}[i] \equiv 0$. Since FRUGAL injects only "+1" packets at each position, each entry in $\mathbf{x}$ is a scalar value. A detailed justification of this injection strategy is provided in Section V-A3.

To build a lookup profile for each website, we aggregate all injection patterns into a matrix $\mathbf{X} \in \mathbb{R}^{M\times(d+1)}$, where M is the number of monitored websites. The k-th row of $\mathbf{X}$ corresponds to website k and stores the cumulative injection counts across all its defended traces, which can be expressed as $\mathbf{X}[k,:] = \sum_{\mathbf{x}\in C_k} \mathbf{x}$, where C_k is the set of all defended traces $\mathbf{x}$ belonging to website k. As shown in the heatmaps of $\mathbf{X}$ in Figure 12 (with a detailed analysis in Appendix D), the injection positions generated by FRUGAL for each website are highly sparse and concentrated. This observation motivates the design of FRUGAL-online, which leverages $\mathbf{X}$ to generate injection positions in an online manner.

Specifically, at runtime, given a website label k as a query, FRUGAL-online generates an injection pattern by sampling from a Dirichlet–Multinomial distribution:

$$\begin{aligned} \boldsymbol{p}_k &\sim \mathrm{Dir}(\boldsymbol{c_k}), \\ \hat{\mathbf{x}} &\sim \mathrm{Multi}(\boldsymbol{p_k}, m_k). \end{aligned} \tag{8}$$

Here, the Dirichlet parameter $\boldsymbol{c}_k = \mathbf{X}[k,:]$ is the pre-computed pattern vector for website k, retrieved from $\mathbf{X}$. The multinomial parameter $\boldsymbol{p}_k \in \mathbb{R}^{1\times(d+1)}$ governs the sampling probability for each position. The sample size $m_k = \lfloor \mathrm{BWO} \cdot \frac{1}{n_k} \sum i_k = 1^{n_k} |x i_k| \rfloor$ is the total packet budget, derived from the predefined BWO and the average trace length for website k. The resulting $\hat{\mathbf{x}} \in \mathbb{R}^{1\times(d+1)}$ specifies the number of dummy packets to inject at each position for the current trace. This Dirichlet–Multinomial sampling introduces stochasticity, creating diversity across different visits to the same website and thereby improving robustness.

With FRUGAL-online's lightweight implementation, $\hat{\mathbf{x}}$ can be generated in real time by querying with the website label k prior to packet transmission, allowing online defense.

V. EVALUATION

We present a comprehensive evaluation of FRUGAL, detailing the experimental setup and baselines. We assess performance across Closed-World, Open-World, and One-Page scenarios, including resilience to Adversarial Training. Finally, we validate practical viability through real-world simulation, sensitivity analysis, and a temporal generalization study.

A. Experimental Setup

1) Dataset: The experiments in this paper are based on the publicly available DF dataset collected by Sirinam *et al.* [37]. This dataset is specifically designed for evaluating

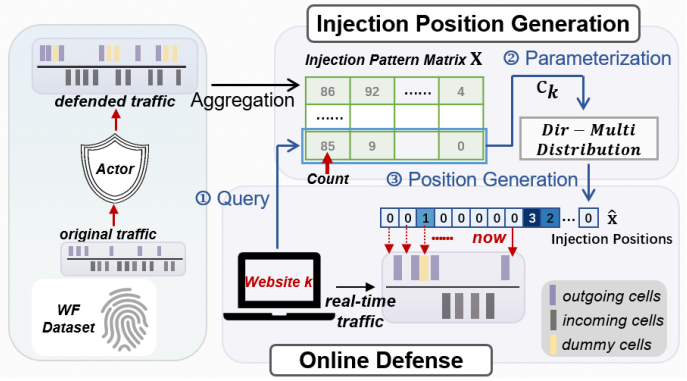


Fig. 6. **Overview of FRUGAL-online.** We use "website k" as an example to illustrate how FRUGAL-online works, where the blue arrows labeled (1) Query, (2) Parameterization, and (3) Position Generation indicate the runtime workflow.

TABLE I
DATASET IN THE FRUGAL EXPERIMENTS

		Websites	Traces
Train set	Monitored	95	20
	Unmonitored	20	1
Validation set	Monitored	95	100
	Unmonitored	10000	1
Testing set	Monitored	95	100
	Unmonitored	10000	1

WFD solutions in both closed-world and open-world scenarios. Its traffic traces were collected under realistic, dynamic conditions, inherently capturing real-world dynamics. It comprises traffic from monitored and unmonitored websites. The monitored websites consist of traffic from the top 95 Alexa websites, each represented by 1,000 sample traces. The unmonitored websites consist of traffic from 40,000 other websites, with each represented by a single trace. In the closed-world scenario, the adversary's access is restricted to user traffic from the monitored websites. In contrast, the open-world scenario allows the adversary to access traffic from both monitored and unmonitored websites.

To expedite the training process, we pre-select a high-confidence training set called the Goodsample set from the monitored set. Specifically, for each website, we select 20 traffic traces that are correctly classified by a pre-trained attack model with a confidence score of at least 90%. It is noteworthy that, while the training set is carefully selected to accelerate training, the test set used in our experiments is comprehensive and uncurated, confirming FRUGAL's generalization ability. To further address potential concerns regarding overfitting or bias introduced by this setup, we conducted a sensitivity analysis with respect to the training set, as discussed in Section V-G.

Throughout our experiments, we trained FRUGAL using the training set and evaluated its defensive effectiveness on the testing set. The number of traces is shown in Table I.

2) Metrics: The metrics used in this paper to evaluate the effectiveness and efficiency of WFD solutions are *Attack Success Rate* (ASR) and *Bandwidth Overhead* (BWO), respectively. Specifically, ASR is defined in CW as

TABLE II
PARAMETER SETTINGS IN THE FRUGAL EXPERIMENTS

Parameter	Default value
Discount Factor γ	0.9
Sample Batch N	32
Regularization Coefficient α	0.01
Weight Coefficient ϵ	0.01

TABLE III
CLASSIFIERS RESULTS ON THE DATASET IN CLOSED-WORLD AND OPEN-WORLD SCENARIOS

	ASR					
	DF	Var-CNN	NetCLR	TF	AWF	RF
CW	98.27%	97.47%	97.73%	97.81%	95.41%	98.8%
OW	97.80%	97.23%	97.23%	97.21%	93.82%	98.1%

$$\mathrm{ASR} = \frac{N_{\mathrm{cor}}}{N_{\mathrm{all}}}, \tag{9}$$

where N_{cor} is the number of correctly classified traces, and N_{all} is the total number of traffic traces evaluated by the attack model. In OW, ASR is defined as

$$\mathrm{ASR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{10}$$

where TP is the number of correctly classified traces in the monitored set, and FP is the number of wrongly classified traces in the unmonitored set. A lower ASR indicates a more effective WFD solution.

On the other hand, BWO is defined as:

$$\mathrm{BWO} = \frac{l_{\mathrm{def}} - l_{\mathrm{ori}}}{l_{\mathrm{ori}}}, \tag{11}$$

where l_{def} is the length of the defended traffic, and l_{ori} is the length of the original traffic. A lower BWO signifies fewer dummy packets injected in addition to the original traffic, highlighting greater efficiency.

Notably, the time overhead introduced by FRUGAL is negligible. This is because it injects dummy packets only into the client's outgoing traffic without delaying existing packets, and it does not alter the incoming web response. This approach differs from other defenses evaluated in [44], which add packets to traffic in both directions and consequently incur greater delays.

3) Implementation: FRUGAL is implemented using the PyTorch 2.0 framework[1]. Specifically, the traffic encoder, which processes traffic input x and outputs a state vector s to the policy network, is implemented as a one-layer CNN, where the kernel and stride sizes K are set to 5. The components of the actor network and critics module are both implemented as a two-layer MLP. The number of packet positions per injection is set to 5, with only '+1' used for dummy packets. Please refer to Appendix E for a comprehensive discussion of the hyperparameter configuration.

In FRUGAL, an arbitrary neural network is employed to construct the MI estimator. Since the classifier in the MI estimator is designed to efficiently classify traffic labels, this study adopts the architecture of the DF model. The classifier in MI estimator is initially trained using traffic from the DF dataset and is continuously updated with modified traffic, as detailed in Section IV-B. It is important to emphasize that while the classifier shares the same architecture as one of the attack models used during testing, it remains entirely distinct from the attack model, as their parameters are completely independent. Thus, the assumption that the defender has no access to the attack model remains valid in FRUGAL.

All neural networks involved in FRUGAL were trained on Nvidia RTX A6000 GPU. The remaining hyperparameters used in our experiments are determined empirically and are detailed in Table II. The complete implementation code will be made available shortly.

4) Baseline & Benchmark: The effectiveness of FRUGAL is evaluated against five SOTA attack models, including DF [37], Var-CNN [2], NetCLR [1], TF [38], AWF [33] and RF[35]. Initially, we assess the ASR of these models on the original traffic from the DF dataset, and the results are presented in Table III, which serve as the baseline. FRUGAL is applied to the traffic from the DF dataset, and the ASR of each attack model is re-evaluated on the defended traffic. To provide a comprehensive performance comparison, we also benchmark FRUGAL against five SOTA WFD methods from two different categories, evaluating their effectiveness in reducing the ASR across different attack models on the DF dataset. The selected defense methods include WTF-PAD [20], Surakav [14], Regulator [17], FRONT [13], Palette [36], Tamaraw[4] and RUDOLF [18]. And for all defense strategies, we opt for their top-performing setup.

B. Closed-World Performance

In this section, we evaluate the efficacy of FRUGAL in the standard closed-world scenario. This evaluation involves testing against six benchmark WFA models and comparing the performance with random injection and seven other WFD methods. The results are presented in Table IV and Figure 7.

Table IV highlights FRUGAL's high efficiency. At a 20% BWO, FRUGAL reduces the ASR of the DF model to just 6.87%. When FRUGAL's BWO is increased to 30%, its ASR against DF drops further to 2.68%, substantially outperforming competing defenses that impose much higher BWOs. This strong performance-to-cost ratio holds true for the other WFA models shown in the table, with FRUGAL consistently achieving a superior ASR for its BWO level. The only exception is Tamaraw, which achieves a lower ASR of 1.05%. However, Tamaraw's commendable performance comes at the cost of an impractically high BWO, rendering it unsuitable for deployment in performance-critical anonymous networks.

To comprehensively analyze its effectiveness, we evaluate FRUGAL under BWO constraints ranging from 10% to 100% in 10% intervals, a scope that encompasses the overheads of most prior WFD methods. The results are illustrated in Figure 7, which plots the ASR against BWO for all evaluated defenses. In this visualization, superior performance

[1]The code is available at https://github.com/Junowww/FRUGAL-ndss.

TABLE IV
PERFORMANCE IN THE CLOSED-WORLD SCENARIO

Defenses	BWO	ASR					
		DF	Var-CNN	NetCLR	TF	AWF	RF
Random Injection	20%	93.98%	91.98%	90.6%	94.3%	92.76%	96.58
	30%	76.59%	80.17%	76.59%	79.34%	75.19%	95.3
WTF-PAD	60.7%	80.92%	78.14%	86.92%	88.65%	59.96%	96.58%
Tamaraw[1]	121%	1.05%	0.98%	1.01%	1.12%	1.05%	2.09%
FRONT	79.6%	73.62%	60.25%	73.62%	76.46%	60.44%	93.34%
Surakav	81%	64%	54.6%	56.69%	60.95%	67.65%	79.94%
Palette	87.17%	11.54%	10.99%	11.2%	12.91%	11.54%	46.43%
RegulaTor	68.3%	20.41%	40.52%	32.31%	35.52%	45.6%	53.11%
RUDOLF	27.46%	18.59%	-	-	23.71%	-	28%
FRUGAL	**20%**	**6.87%**	**8.03%**	**12.73%**	**10.37%**	**10.12%**	**16.6%**
	30%	**2.68%**	**2.61%**	**6.68%**	**5.67%**	**5.73%**	**12.7%**

[1]. While Tamaraw achieves commendable performance on benchmark datasets, its excessive bandwidth overhead renders it impractical for deployment in anonymous network environments where performance and user experience are critical.

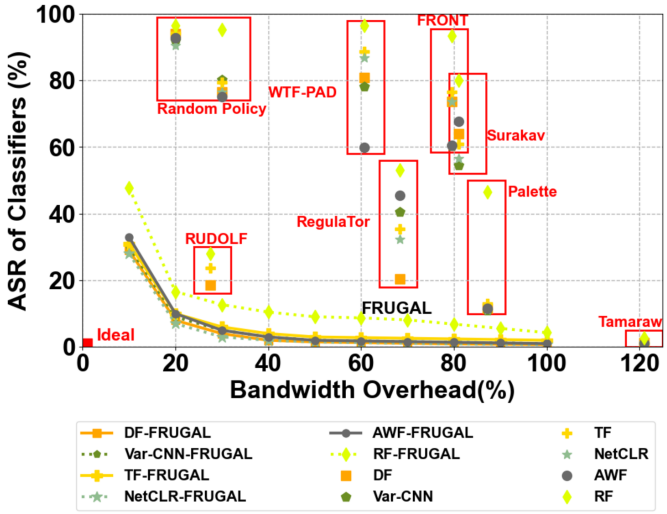


Fig. 7. **Performance of FRUGAL in the Closed-World Scenario.** The area within the red rectangle shows a performance comparison of different defense methods. Our method, FRUGAL, represented by the plotted lines, consistently occupies the bottom-left corner, which signifies the notable trade-off between security and overhead.

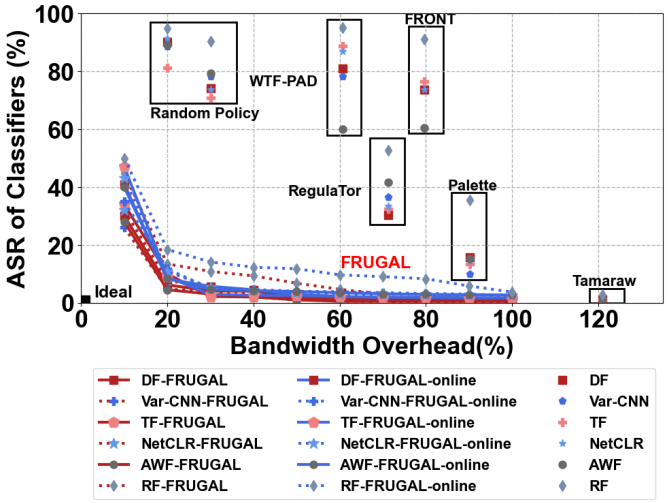


Fig. 8. **Performance of FRUGAL in the Open-World Scenario.**

is indicated by data points closer to the bottom-left corner, representing a low ASR achieved with minimal BWO. The trend is clear: across the entire spectrum of BWO constraints, FRUGAL establishes a new SOTA, consistently achieving a lower ASR than competing WFD methods at any given level of overhead.

C. Open-World Performance

In this section, we extend our evaluation to a more realistic open-world scenario. The baselines, metrics, and other evaluation configurations are held consistent with those used in the closed-world scenario for comparability.

We begin by evaluating FRUGAL under BWO constraints ranging from 10% to 100%, with the results presented in Figure 8. All WFD methods experience a slight degradation in defensive performance across all WFA models, reflecting the heightened challenge of defending against attacks in the open-world setting. As shown in Table V, at a 20% BWO level, FRUGAL lowers the ASR of DF to 6.2%, Var-CNN to 6.55%, NetCLR to 7.8%, TF to 5.7%, AWF to 4.5% and RF to 13.43%. When we increase the BWO to 30%, the ASRs drop further; for example, the ASR for DF decreases to 4.09% and for RF it drops to 10.85%. For a comprehensive comparison, in Table V we also present the performance of FRUGAL-online, evaluated on the same dataset but in a real-world simulation setting. A detailed discussion of FRUGAL-online is provided in Section V-F.

The plot in Figure 8 visualizes the trade-off between ASR and BWO. Specifically, FRUGAL's performance curve consistently outperforms competing defenses, remaining closer to the bottom-left "Ideal" corner. While Tamaraw achieves a lower ASR, its associated BWO is impractically high. The plot thus makes it clear that FRUGAL provides the best performance trade-off against all tested attack models.

D. One-Page Setting Performance

To further evaluate the robustness of FRUGAL, we conduct a more challenging setup known as the one-page setting[41]. Our evaluation focuses on the closed-world scenario because attackers' performance is generally stronger in the closed-

TABLE V
PERFORMANCE IN THE OPEN-WORLD SCENARIO

Defenses	BWO	ASR					
		DF	Var-CNN	NetCLR	TF	AWF	RF
Random Injection	20%	90.12%	88.6%	91.43%	81.3%	89.3%	94.8%
	30%	74.04%	78.25%	70.63%	73.75%	79.4%	90.3%
WTF-PAD	60.7%	80.92%	78.14%	86.92%	88.65%	59.96%	95.12%
Tamaraw	121%	1.02%	0.9%	1.0%	1.12%	0.95%	2.07%
FRONT	99%	57.23%	50.25%	54.62%	56.46%	57.44%	91.2%
RegulaTor	71.32%	30.41%	36.52%	33.5%	32.12%	41.6%	52.61%
Palette	90.2%	15.81%	9.89%	14.31%	13.41%	15.32%	35.42%
FRUGAL	**20%**	**6.2%**	**6.55%**	**7.8%**	**5.7%**	**4.5%**	**13.43%**
	30%	**4.09%**	**4.7%**	**3%**	**2.17%**	**2.58%**	**10.85%**
FRUGAL-online	**20%**	**8.4%**	**11.3%**	**10.1%**	**9.3%**	**8.8%**	**18.2%**
	30%	**4.69%**	**4.8%**	**5.33%**	**2.86%**	**4.6%**	**14.1%**

TABLE VI
EVALUATION OF FRUGAL IN THE ONE-PAGE SETTING COMPARED TO OTHER DEFENSE METHODS

	Defenses			
	FRUGAL	Palette	RUDOLF	RegulaTor
BWO	**19.63%**	109.17%	27.46%	48.3%
Average ASR	**6.54%**	36.85%	67.3%	55.71%

TABLE VII
ADVERSARIAL TRAINING PERFORMANCE OF FRUGAL

	Attack Models	Bandwidth Overhead Control (%)			
		20	30	60	80
CW	DF	56.85%	43.93%	18.68%	9.42%
	Var-CNN	47.66%	25.48%	15.22%	8.56%
	TF	61.21%	28.56%	15.6%	7.93%
	NetCLR	61.87%	40.23%	16.4%	9.41%
	AWF	35.35%	30.77%	6.73%	4.52%
	RF	60.35%	49%	29.3%	18.2%
OW	DF	53.5%	40.02%	16.13%	8.2%
	Var-CNN	45.1%	29.65%	17.22%	4.56%
	TF	43.2%	35.14%	11.6%	3.3%
	NetCLR	48.7%	38.23%	11.6%	6.54%
	AWF	33.3%	27.92%	5.8%	4.2%
	RF	60.2%	47%	27.3%	17.14%

world than in the open-world scenario [4], [37], which makes it a tougher challenge for our defense.

In the one-page setting, the attacker is only trying to find out if a user visited one single, specific website. We apply the whole DF dataset in this experiment. In each test, we pick one website to be the monitored site, and the other 94 websites become the unmonitored set. We repeat this 95 times so that every website gets a turn to be the monitored one. The DF model is employed as the attack model to evaluate our performance within this setting, utilizing the ASR of the DF model as the benchmark. We execute the experiment with a 20% BWO, and the results are in Table VI. The average ASR was 6.54%, with an average bandwidth overhead of 19.63%. This result is much better than other defenses like Palette (36.85% ASR with 109.17% BWO), RUDOLF (67.3% ASR with 27.46% BWO), and RegulaTor (55.71% ASR with 48.3% BWO).

E. Adversarial Training Performance

In this section, we evaluate FRUGAL's resilience to adversarial training, which is the most challenging evaluation toward the efficacy and robustness of a WFD method. Adversarial training involves the process where an attacker retrains the WFA model with the protected traffic, such that the WFA model is able to recapture the indicative patterns in the protected traffic, hence mitigating the defense effectiveness of the WFD method.

We perform adversarial training experiments utilizing defended traffic under both CW and OW scenarios and employ the ASR of attack models utilized in our experiments. Our experiments assess defensive performance across various BWO ranging from 10% to 80% with intervals of 10%, where the results are delineated in Table VII.

When compared with other defenses in Figure 9, both FRUGAL and FRUGAL-online can achieve SOTA adversarial training performance under similar BWO. For RUDOLF, we select the performance corresponding to the BWO reported in the CW scenario. In comparison to Palette [36], which reduces the ASR of DF to 20.27% with 80% BWO, our FRUGAL is able to reduce the ASR of DF to less than half of Palette, i.e., 9.42% under 80% BWO. On the other hand, with 60% BWO, FRUGAL is able to achieve a similar ASR reduction as the result achieved by Palette under 80% BWO. Recall that FRUGAL aims at minimizing the MI between the traffic and its associated label, as a result, the traffic protected by FRUGAL presents more resilience to adversarial training.

F. Online Defense in Real-World Simulation

To validate the practical effectiveness of FRUGAL, we evaluate FRUGAL-online in a real-world simulation and assess the performance of various attack models after adversarial training on our validation set.

We first examine FRUGAL-online's performance, with results shown in Figure 10. The figure reports the ASR of various models against our defense with BWO values of 20% and 30%. Although FRUGAL-online experiences a minor performance drop compared to FRUGAL on the same OW dataset, which is caused by the information loss introduced by the distillation from FRUGAL, it still achieves competitive results and significantly outperforms the other benchmarks.

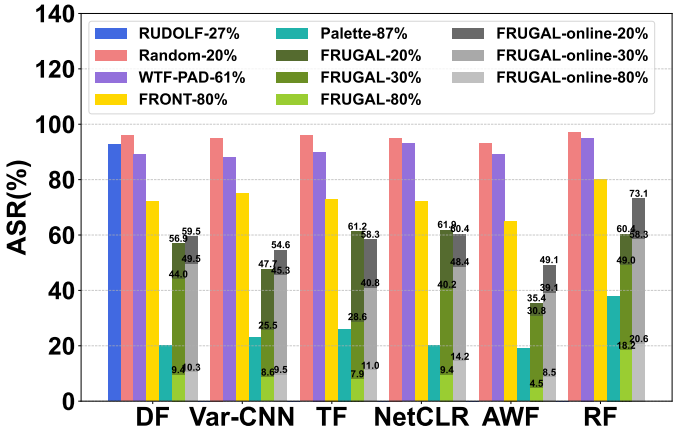


Fig. 9. **Adversarial Training Performance.** In the figure legend, we adopt the format 'defense-bwo' as labels for each defense strategy.

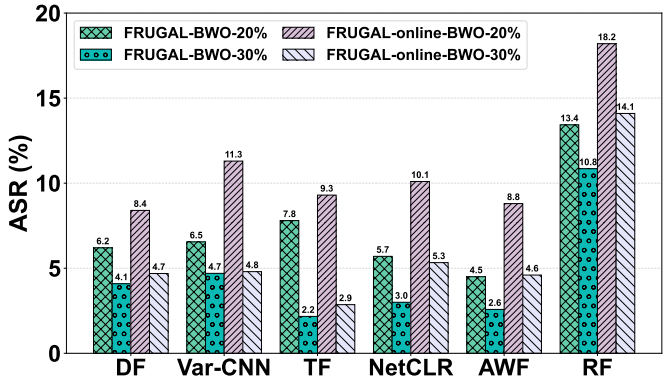


Fig. 10. **Performance in the Real-World Simulation.**

As shown in Figure 8, when BWO is 20%, FRUGAL-online reduces the ASR of DF to 8.4%, Var-CNN to 11.3%, NetCLR to 10.1%, TF to 9.3%, AWF to 8.8%, and RF to 18.2%. When the BWO is increased to 30%, the ASR reduction approaches that of FRUGAL; for example, DF's ASR drops to 4.69% and RF's to 14.1%.

To assess the adversarial robustness of FRUGAL in a real-world simulation, we collected traffic defended by FRUGAL-online and used it to retrain the attack models, following the procedure described in Section V-E. The results show that increasing the BWO substantially reduces the success rates of all retrained attack models. For instance, DF's accuracy decreases from 59.45% to 10.3% at 80% BWO. As illustrated in Figure 9, FRUGAL-online continues to outperform other benchmarks and remains close in performance to FRUGAL. This trend underscores the practical effectiveness and resilience of FRUGAL-online.

G. Sensitivity Analysis of Training Set

In our experiments, we used a high-confidence *Goodsample* subset (20 samples per site, with $\geq$ 90% confidence) to accelerate training. However, this choice may raise concerns about potential overfitting or bias, which could limit the model's generalization performance.

To validate our choice and assess FRUGAL-online's robustness with respect to the training set (TS), we conducted

TABLE VIII
ADVERSARIAL PERFORMANCE OF REAL-WORLD SIMULATION

	BWO			
	20%	30%	60%	80%
DF	59.45%	49.55%	22.71%	10.3%
Var-CNN	54.56%	45.34%	19.3%	9.5%
TF	58.32%	40.84%	15.6%	10.99%
NetCLR	60.39%	48.41%	26.8%	14.23%
AWF	49.14%	39.11%	25.7%	8.52%
RF	73.1%	58.3%	35.2%	20.62%

TABLE IX
SENSITIVITY RESULTS OF TRAINING SET SCALE.

TS*	TC*	ASR					
		DF	Var-CNN	NetCLR	TF	AWF	RF
GS	1.42h	2.8%	2.6%	5.4%	1.3%	5.8%	10.7%
FD	45.88h	2.8%	2.4%	5.5%	1.2%	5.7%	10.5%

* Training Set (TS), Time Consumption (TC), Goodsample(GS), Full Dataset(FD).

a sensitivity analysis. We compared FRUGAL-online trained under two settings: (1) **Baseline (Goodsample)** using the *Goodsample* subset, and (2) **Full Dataset** using the entire training set. Both models were evaluated on the complete test dataset under the CW scenario with a 30% BWO limit, aiming to match the ASR of different attack models. Training Time Consumption (TC) for both models were recorded. Table IX illustrates the trade-off: FRUGAL-online trained on the Full Dataset showed negligible improvement over the *Goodsample* baseline, indicating that *Goodsample* captures the essential features for training. This minimal performance gain is vastly outweighed by the substantial 32-fold increase in training time required by the Full Dataset.

This sensitivity analysis strongly supports our methodological choice of using the *Goodsample* subset, demonstrating that it achieves an excellent balance between training efficiency and model effectiveness.

H. Temporal Generalization Evaluation

To assess FRUGAL-online's generalizability against "concept drift", i.e., the natural evolution of website traffic patterns over time, we performed a temporal evaluation. For this experiment, we collected a new, time-shifted dataset comprising two distinct sets: *Base Dataset:* Collected in February 2025, this set contains 1,000 traffic traces for each of 90 monitored websites. *Drift Dataset:* Collected in October 2025, this set contains 150 new traffic traces for the same 90 websites, representing an 8-month temporal gap. We split both the *Base Dataset* and *Drift Dataset* into training and testing sets.

We evaluate ASR in three scenarios: Base-Base (train/test on *Base*), Base-Drift (train on *Base*, test on *Drift*), and Drift-Drift (train/test on *Drift*). As shown in Table X, classifiers are highly accurate on temporally-aligned data (e.g., DF: 98.6% in Base-Base, 95.2% in Drift-Drift). However, in the Base-Drift scenario, which measures generalization across the 8-month gap, performance plummets: DF's drops from 98.2% to 66.9%. This confirms that static classifiers fail to adapt to concept drift and generalize poorly over time.

TABLE X
CLASSIFIERS RESULTS ON BASE DATASET AND DRIFT DATASET

	ASR					
	DF	Var-CNN	NetCLR	TF	AWF	RF
Base-Base	98.2%	97.4%	97.1%	97.6%	93.1%	98.6%
Base-Drift	66.9%	63.6%	67.2%	65.7%	53.5%	76.4%
Drift-Drift	95.2%	91.4%	92.5%	93.5%	81.7%	95.6%

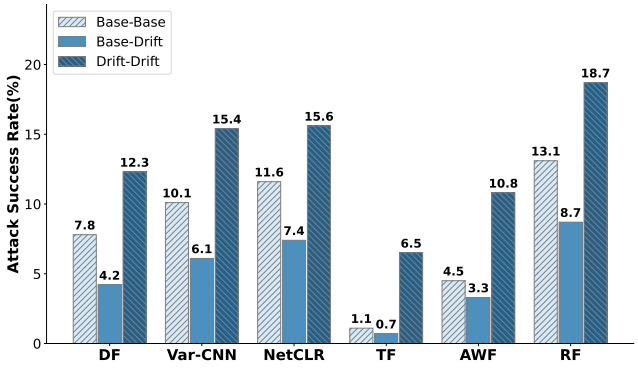


Fig. 11. **Sensitivity of Temproal Drift.**

We trained a single static FRUGAL-online policy on the *Base GoodSample* (30% BWO) and evaluated it on *Base* and *Drift* sets. As shown in Figure 11, the defense effectively suppresses *Base*-trained models (e.g., DF ASR 7.8%), improving to 4.2% on the *Drift* set. Against *Drift*-trained models, ASR increases marginally to 12.3% but remains low. Furthermore, our efficient 1.4-hour training time enables practical regular retraining, demonstrating robustness to "concept drift".

VI. RELATED WORK

A. The Evolution of Website Fingerprinting Attacks

Initial explorations into WFA have established its viability through the application of traditional machine learning models, which depended on handcrafted statistical features. Pioneers like [29], [28], [42], [16] established foundational methods using SVMs, Random Forests and k-NN respectively. While effective, these early approaches frequently demanded substantial manual effort and extensive domain-specific knowledge for the purpose of feature engineering.

The field entered a new era with the advent of deep learning. The pivotal work, DF by Sirinam et al. [37], marked a paradigm shift. By employing a Convolutional Neural Network, DF could directly process raw traffic data, eliminating the need for manual feature extraction and achieving an unprecedented ASR of over 98% in closed-world settings. This breakthrough spurred a wave of increasingly sophisticated DL-based attacks, each leveraging more advanced neural network architectures, from the ResNet in Var-CNN [2] and various deep neural networks in AWF [33] to the recent applications of Triplet Network [38] and contrastive learning in NetCLR [1]. Concurrently, a new focus on robust traffic representation emerged, exemplified by RF [35], which was explicitly designed to be resilient against defensive measures, further escalating the attacker's capabilities. Today, these powerful and efficient DL-based models represent a formidable threat to user privacy in anonymous communication systems, necessitating the development of equally advanced defenses [26].

B. The Development of Website Fingerprinting Defenses

In response to the growing threat of WFAs, a parallel field of WFD has emerged, largely following two distinct categories.

The first, feature suppression[12], [3], [4], [20], [45], [36], represents the most intuitive approach, which seeks to homogenize traffic traces to obscure identifying features. Foundational methods like Tamaraw [4] established this by padding traffic to a constant rate, but at the cost of prohibitive bandwidth and latency overhead. This fundamental trade-off persists even in contemporary, sophisticated suppression techniques like Palette [36], limiting their practical applicability.

The second and more recent methods, feature morphing[31], [34], [25], [14], [18], [24], [21], aim for a more targeted and efficient defense by actively altering traffic features to mislead and confuse WFA models. These defenses [31], [34], [21] often generate noise tailored to specific traffic characteristics. The most advanced in this category leverage adaptive learning. For instance, RUDOLF [18] uses Reinforcement Learning to receive feedback from specific classifiers for deterministic policy. The critical limitation of these advanced methods is model-dependency. Training on known attackers leads to poor generalization, leaving them vulnerable to unforeseen or adversarially retrained attacks.

Our work addresses the urgent need for a defense that effectively counters diverse, adaptive attacks, is efficient with low overhead, and remains robust in the evolving WFA landscape.

VII. DISCUSSION

Outgoing-packets-only Perturbation: We adopt an outgoing-packet-only injection strategy for three reasons. (1) Information-theoretic analyses (e.g., [22]) show the low-volume client stream is disproportionately feature-rich. (2) Our experiments prove this approach can effectively defeat state-of-the-art attacks that use bidirectional traffic. (3) This client-side design adds negligible latency and requires no response modifications.

Real-world Integration: FRUGAL-online can be integrated as a Pluggable Transport (PT), aligning with our threat model by obfuscating the client-to-entry-node path. This approach uses two components: (1) a client-side proxy first obtains the ground-truth website label from the URL [36], samples a pre-computed defense pattern using that label, and then injects the dummy cells; (2) a server-side proxy on the entry node that removes these cells before forwarding the original traffic. This PT-based design is practical, as it requires no core Tor modifications and encapsulates the defense as an optional transport to protect against website fingerprinting.

Stronger Adversary: While focused on local adversaries, FRUGAL extends to stronger threats like colluding nodes by integrating with Tor's circuit-level padding. An adversary controlling multiple relays could pose a significant risk by observing traffic at different points along a circuit. However, FRUGAL's defense can extend beyond first-hop adversaries

by integrating with Tor's circuit-level padding mechanisms to counteract stronger attacks. By utilizing dummy packets as *DROP* relay cells, FRUGAL can effectively obfuscate traffic, hindering potential WF attacks from malicious relays. Although a compromised exit node can access unencrypted data, it cannot trace the user's origin because of the design of Tor.

Multi-tab Scenario: Our evaluation, consistent with the prevalent assumption in WF studies, considers a single-tab browsing scenario. While this setting enables reproducible comparisons, we acknowledge that real-world browsing behavior is more complex and often involves multi-tab activity [19], [9]. FRUGAL is well-positioned to address this challenge. The FRUGAL framework can be expanded by distinguishing per-tab or per-domain traffic flows and applying the padding policy to each stream, making it compatible with protecting concurrent browsing sessions.

VIII. CONCLUSION

This paper addresses the critical challenge of designing an effective defense against WFA. The core problem is to generate defended traffic that is simultaneously attack-model-agnostic, bandwidth-efficient, and resilient to adversarially-trained models—three properties that existing defenses often struggle to balance.

To tackle this challenge, we introduce FRUGAL, a novel defense framework. To the best of our knowledge, FRUGAL is the first work to leverage the reduction of MI between website traffic and its identity label as the primary optimization objective. This allows FRUGAL to fundamentally minimize the information leakage that attackers can exploit, rather than merely overfitting to a known set of attack models. Extensive evaluations conducted in closed-world, open-world, and the more challenging one-page scenario demonstrate the effectiveness of the defended traffic generated by FRUGAL over SOTA defenses. It consistently achieves a significant reduction in the ASR of various attack models, including those that have undergone adversarial training, while incurring minimal overhead. The evaluation also includes a real-world simulation of FRUGAL-online to validate its robustness, as well as a sensitivity analysis on the size of our training dataset. FRUGAL not only establishes a new benchmark for WFD research but also paves the way for future exploration of effective, efficient, and robust defenses against evolving WFA threats.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their constructive comments and suggestions. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 62232004, 92467205, and 62502086, the Natural Science Foundation of Jiangsu Province under Grant No. BK20251295, the Start-up Research Fund of Southeast University under Grant No. RF1028624178, the Jiangsu Provincial Key Laboratory of Network and Information Security under Grant No. BM2003201, the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant No. 93K-9, and the Collaborative Innovation Center of Novel Software Technology and Industrialization. We also acknowledge the support of the Big Data Computing Center of Southeast University. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

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

[2] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-cnn: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2018.

[3] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, 2014.

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

[5] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *International conference on machine learning (ICML)*, 2020.

[6] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter. On web browsing privacy in anonymized netflows. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2007.

[7] Ian Connick Covert, Wei Qiu, Mingyu Lu, Na Yoon Kim, Nathan J White, and Su-In Lee. Learning to maximize mutual information for dynamic feature selection. In *International Conference on Machine Learning(ICML)*. PMLR, 2023.

[8] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 2005.

[9] Xinhao Deng, Qilei Yin, Zhuotao Liu, Xiyuan Zhao, Qi Li, Mingwei Xu, Ke Xu, and Jianping Wu. Robust multi-tab website fingerprinting attacks in the wild. In *2023 IEEE symposium on security and privacy (S&P)*, 2023.

[10] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. Tor: The second-generation onion router. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2004.

[11] Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. Datum-wise classification: a sequential approach to sparsity. In *Machine Learning and Knowledge Discovery in Databases: European Conference, (ECML) PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I 11*, 2011.

[12] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-Boo, I still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2012.

[13] Jiajun Gong and Tao Wang. Zero-delay lightweight defenses against website fingerprinting. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020.

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

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning(ICML)*, 2018.

[16] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2016.

[17] James K Holland and Nicholas Hopper. Regulator: A straightforward website fingerprinting defense. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2020.

[18] Meiyi Jiang, Baojiang Cui, Junsong Fu, Tao Wang, Lu Yao, and Bharat K Bhargava. Rudolf: An efficient and adaptive defense approach against website fingerprinting attacks based on soft actor-critic algorithm. *IEEE Transactions on Information Forensics and Security(TIFS)*, 2024.
[19] Zhaoxin Jin, Tianbo Lu, Shuang Luo, and Jiaze Shang. Transformer-based model for multi-tab website fingerprinting attack. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023.
[20] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an Efficient Website Fingerprinting Defense. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2016.
[21] Ding Li, Yuefei Zhu, Minghao Chen, and Jue Wang. Minipatch: Undermining dnn-based website fingerprinting with adversarial patches. *IEEE Transactions on Information Forensics and Security(TIFS)*, 2022.
[22] Shuai Li, Huajun Guo, and Nicholas Hopper. Measuring information leakage in website fingerprinting attacks and defenses. In *Proceedings of the ACM SIGSAC conference on computer and communications security (CCS)*, 2018.
[23] Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In *International conference on machine learning (ICML)*, 2021.
[24] Zhen Ling, Gui Xiao, Lan Luo, Rong Wang, Xiangyu Xu, and Guangchi Liu. Wfguard: an effective fuzzing-testing-based traffic morphing defense against website fingerprinting. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2024.
[25] Zhen Ling, Gui Xiao, Wenjia Wu, Xiaodan Gu, Ming Yang, and Xinwen Fu. Towards an efficient defense against deep learning based website fingerprinting. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022.
[26] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. Sok: A critical evaluation of efficient website fingerprinting defenses. In *2023 IEEE Symposium on Security and Privacy (S&P)*, 2023.
[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
[28] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *Proceedings of the Network Distributed System Security Symposium (NDSS)*, 2016.
[29] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the annual ACM workshop on Privacy in the Electronic Society (WPES)*, 2011.
[30] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems(ICAAMS)*, 2018.
[31] Mohammad Saidur Rahman, Mohsen Imani, Nate Mathews, and Matthew Wright. Mockingbird: Defending Against Deep-LearningBased Website Fingerprinting Attacks With Adversarial Traces. *IEEE Transactions on Information Forensics and Security (TIFS)*, 2020.
[32] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations (ICLR)*, 2017.
[33] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated Website Fingerprinting through Deep Learning. In *Proceedings of the Network Distributed System Security Symposium (NDSS)*, 2018.
[34] A M Sadeghzadeh, B Tajali, and R Jalili. Awa: Adversarial website adaptation. *IEEE Transactions on Information Forensics and Security (TIFS)*, 2021.
[35] 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 USENIX Security Symposium (USENIX Security)*, 2023.
[36] Meng Shen, Kexin Ji, Jinhe Wu, Qi Li, Xiangdong Kong, Ke Xu, and Liehuang Zhu. Real-time website fingerprinting defense via traffic cluster anonymization. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2024.
[37] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the ACM SIGSAC conference on computer and communications security (CCS)*, 2018.
[38] 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)*, 2019.
[39] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, 2016.
[40] Chunmian Wang, Junzhou Luo, Zhen Ling, Lan Luo, and Xinwen Fu. A comprehensive and long-term evaluation of tor v3 onion services. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications (INFOCOM)*, 2023.
[41] Tao Wang. The one-page setting: A higher standard for evaluating website fingerprinting defenses. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security(CCS)*, 2021.
[42] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2014.
[43] Tao Wang and Ian Goldberg. Improved Website Fingerprinting on Tor. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, 2013.
[44] Ethan Witwer, James K Holland, and Nicholas Hopper. Padding-only defenses add delay in tor. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society (WPES)*, 2022.
[45] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network Distributed System Security Symposium (NDSS)*, 2009.

APPENDIX A
SOFT ACTOR-CRITIC ALGORITHM

The Soft Actor-Critic (SAC) algorithm [15] in DRL improves learning stability and efficiency by maximizing policy entropy, encouraging the agent to explore a broader range of actions instead of focusing solely on those with immediate high rewards. This strategy enables the algorithm to develop an efficient policy network that effectively balances exploration with the objective of maximizing cumulative rewards. The SAC algorithm employs a policy network π_θ, also called actor network, alongside two critic networks, denoted by $\mathcal{Q}_{\omega_1}$ and $\mathcal{Q}_{\omega_2}$, which evaluate the quality of actions by estimating the expected cumulative reward (i.e., the $\mathcal{Q}$-value) for a given state-action pair. Each critic network has a corresponding target critic network, $\mathcal{Q}_{\omega_1^-}$ and $\mathcal{Q}_{\omega_2^-}$, which are used to stabilize training by providing more reliable target values. To mitigate the issue of overestimation of $\mathcal{Q}$-values observed in other RL algorithms, SAC uses the minimum of the two $\mathcal{Q}$-values produced by the critic networks during each computation of the $\mathcal{Q}$-value. The $\mathcal{Q}$ function facilitates the aggregation of expected rewards, as defined in Equation (12):

$$Q_\omega\left(s_t, a_t\right) = r\left(s_t, a_t\right) + \gamma \mathbb{E}_{s_{t+1}}\left[V_{\omega^-}\left(s_{t+1}\right)\right], \tag{12}$$

where $r\left(s_t, a_t\right)$ denotes the immediate reward received for taking action a_t in state s_t, γ is the discount factor, and $V_{\omega^-}\left(s_{t+1}\right)$ is the value function for the subsequent state s_{t+1}. The value function can be formulated as Equation (13):

$$\begin{aligned} V_{\omega^-}\left(s_t\right) &= \mathbb{E}_{a_t \sim \pi_\theta}\left[Q_\omega\left(s_t, a_t\right) - \alpha \log \pi_\theta\left(a_t \mid s_t\right)\right] \\ &= \mathbb{E}_{a_t \sim \pi_\theta}\left[Q_\omega\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi_\theta\left(\cdot \mid s_t\right)\right)\right], \end{aligned} \tag{13}$$

where $\mathcal{H}$ represents the entropy at state s_t, and α is a regularization coefficient that controls the importance of entropy. Hence, the aim of the SAC algorithm is articulated as the development of an efficient policy network π_θ^*, which seeks to both optimize the expected cumulative rewards and enhance the entropy of the policy, thereby promoting exploration. The objective can be formulated as:

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_t \left[r\left(s_t, a_t\right) + \alpha\mathcal{H}\left(\pi_\theta\left(\cdot \mid s_t\right)\right)\right]\right]. \tag{14}$$

In SAC, entropy regularization enhances the exploratory behavior of policy. The larger the α, the stronger the exploration, which can accelerate the learning of the policy and reduce the likelihood of the policy getting trapped in local optima. In SAC, α is updated automatically:

$$L(\alpha) = \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[-\alpha\log\pi_\theta\left(a_t \mid s_t\right) - \alpha\mathcal{H}_0\right], \tag{15}$$

where $\mathcal{H}_0$ is the target entropy. When the policy entropy is lower than the target entropy, $L(\alpha)$ causes the value of α to increase. The target entropy is usually set to the negative of the action space size.

APPENDIX B
DYNAMIC FEATURE ELIMINATION

In this section, we provide a detailed introduction to the dynamic feature elimination (DFE) method.

Let $\boldsymbol{x}$ denote the input traffic and y represent the corresponding label. The input $\boldsymbol{x}$ consists of d distinct features, expressed as $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, where $0 < i \leq d$ indicates a feature index. Bold symbols $\mathbf{x}, \mathbf{y}$ refer to random variables, while $\boldsymbol{x}, y$ represent their specific values. The data distribution is given by $p(\mathbf{x}, \mathbf{y})$. Union operation $\boldsymbol{x} \cup \boldsymbol{x}_i$ denotes the injection operation where a dummy packet is injected at index i of $\boldsymbol{x}$.

Inspired by dynamic feature selection methods [23], [11], this work proposes Dynamic Feature Elimination (DFE), a technique that iteratively removes the feature with the greatest contribution to the mutual information (MI) between $\boldsymbol{x}$ and its label y. The method is considered dynamic because these features shift as dummy packets are injected into $\boldsymbol{x}$ during each iteration.

To account for these evolving dynamics, conditional mutual information (CMI) is used in place of MI. Specifically, the CMI between the operation $\boldsymbol{x} \cup \boldsymbol{x}_i$ and y, conditioned on the traffic from the current iteration (denoted as $\boldsymbol{x}$), is expressed using KL divergence [7] as:

$$\begin{aligned}&\mathrm{I}\left(\boldsymbol{x} \cup \boldsymbol{x}_i; \boldsymbol{y} \mid \boldsymbol{x}\right)\\ &= D_{\mathrm{KL}}\left(p\left(\boldsymbol{x} \cup \boldsymbol{x}_i, y \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{x} \cup \boldsymbol{x}_i \mid \boldsymbol{x}\right) p\left(y \mid \boldsymbol{x}\right)\right)\\ &= H\left(\boldsymbol{y} \mid \boldsymbol{x}\right) - H\left(\boldsymbol{y} \mid \boldsymbol{x} \cup \boldsymbol{x}_i\right).\end{aligned} \tag{16}$$

Equation (16) indicates that an efficient policy network for FRUGAL can be expressed as $\pi^* = \arg\min_i \mathrm{I}\left(\boldsymbol{x} \cup \boldsymbol{x}_i; y \mid \boldsymbol{x}\right)$. π^* ensures that by injecting dummy packets at the index i conditioned upon $\boldsymbol{x}$ at each iteration, the increase of information entropy, i.e., $H\left(\boldsymbol{y} \mid \boldsymbol{x} \cup \boldsymbol{x}_i\right) - H\left(\boldsymbol{y} \mid \boldsymbol{x}\right)$, can be maximized.

Directly incorporating the term $\mathrm{I}\left(\boldsymbol{x} \cup \boldsymbol{x}_i; y \mid \boldsymbol{x}\right)$ into the end-to-end learning process of FRUGAL is not feasible, as it cannot be optimized directly in an end-to-end fashion. To overcome this limitation, and inspired by [7], we propose a variational approach for learning π^*, which is formulated as:

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{p(\boldsymbol{x}, y)}\left[\mathbb{E}_{i\sim\pi_\theta(\boldsymbol{x})}\left[-\ell_{CE}\left(f_\phi(y \mid \boldsymbol{x_i} \cup \boldsymbol{x}), y\right)\right]\right]. \tag{17}$$

In this formulation, $-\ell_{CE}$ is negative cross-entropy loss. π_θ and f_ϕ are neural networks with trainable parameters θ and ϕ, respectively. Specifically, π_θ serves as the policy network within FRUGAL, while f_ϕ functions as the classifier in the mutual information (MI) estimator, as detailed in Section IV-B. The notation $\boldsymbol{x} \cup \boldsymbol{x}_i$ denotes the dummy packet is injected at index i. Equation (17) demonstrates that minimizing $\mathcal{L}(\phi, \theta)$ can be achieved by alternately performing two operations: (1) updating π_θ and (2) maintaining f_ϕ as the Bayesian classifier for the label y. As described in Section IV-A2, operation (1) is carried out by training π_θ using the SAC algorithm within a reinforcement learning framework, while operation (2) involves updating f_ϕ every $\mathcal{T}_{update}$ iterations through the Dynamic Feature Elimination (DFE) mechanism. Consequently, the policy network π_θ^* is equipped to effectively execute the highly efficient injection operation for the given traffic $\boldsymbol{x}$. The validity of Equation (17) can be demonstrated through Theorem 1. Before introducing Theorem 1, we firstly present proposition 1.

Proposition 1. *For a discrete label y, and when employing the cross-entropy loss function l , we can derive $f^*(\boldsymbol{x_i} \cup \boldsymbol{x}) = p(y|\boldsymbol{x_i} \cup \boldsymbol{x})$.*

Proof. It is assumed that the classifier forecasts an output represented by $\hat{y}$, and the label set of monitored websites is indicated by $\mathcal{Y}$.

$$\begin{aligned}f^*\left(x_i \cup x\right) &= \arg\min_{\hat{y}} \mathbb{E}_{y|x_i\cup x}[\ell(\hat{y}, y)]\\ &= \arg\min_{\hat{y}} \sum_{j\in\mathcal{Y}} p\left(y = j \mid x_i \cup x\right)\log\hat{y}_j\\ &= \arg\min_{\hat{y}} \sum_{j\in\mathcal{Y}} p\left(y{=}j \mid x_i \cup x\right)\cdot\\ &\qquad \log\{\frac{\hat{y}_j}{p(y{=}j|x_i\cup x)}p(y{=}j|x_i\cup x)\}\\ &= \arg\min_{\hat{y}} \sum_{j\in\mathcal{Y}} p\left(y{=}j \mid x_i \cup x\right)\log\frac{\hat{y}_j}{p(y{=}j|x_i\cup x)}\\ &\quad -\sum_{j\in\mathcal{Y}} p\left(y{=}j \mid x_i \cup x\right)\log p(y{=}j|x_i\cup x)\\ &= \arg\min_{\hat{y}} D_{\mathrm{KL}}\left(p\left(y \mid x_i \cup x\right)\|\hat{y}\right) + H\left(y \mid x_i \cup x\right)\\ &= p\left(y \mid x_i \cup x\right)\end{aligned} \tag{18}$$

□

Theorem 1. *For a discrete label y, and when employing the cross-entropy loss function l, the global optimum of Equation* (17) *comprises a Bayesian classifier $f_\phi^*(y \mid \boldsymbol{x_i} \cup \boldsymbol{x})$ and*

a policy network $\pi^*_\theta(x)$. These two components collectively determine the position $i^* = \arg\min_i \mathrm{I}(\boldsymbol{x} \cup \boldsymbol{x}_i; y \mid \boldsymbol{x})$.

Proof. Using Equation (18), we can define the high-performance classifier as $f^*(x') = p(y|x')$, and the optimal global optimum can be defined as minimizing the expected loss of input $\boldsymbol{x}$ with the given high-performance classifier:

$$\begin{aligned}
&\mathbb{E}_{y,x_i|x}\left[-\ell_{CE}\left(f^*\left(x_i \cup x\right), y\right)\right] \\
&= \mathbb{E}_{y,x_i|x}\left[-\ell_{CE}\left(p\left(y \mid x_i \cup x\right), y\right)\right] \\
&= \mathbb{E}_{x_i|x}\left[\mathbb{E}_{y|x_i\cup x}\left[-\ell_{CE}\left(p\left(y \mid x_i \cup x\right), y\right)\right]\right]. \\
&= \mathbb{E}_{x_i|x}\left[H\left(y \mid x_i \cup x\right)\right] \\
&= H\left(y \mid x\right) - \mathrm{I}\left(x_i \cup x; y|x\right)
\end{aligned} \tag{19}$$

Given that $H(y \mid x)$ represents a constant independent of $x_i \cup x$, upon identifying the target position i and producing a modified flow $x_i \cup x$ that minimizes the expected loss, we obtain:

$$\arg\min_i \mathbb{E}_{y,x_i|x}\left[-\ell\left(f^*\left(x_i \cup x\right), y\right)\right] = \arg\max_i I\left(x_i \cup x; y|x\right). \tag{20}$$

Considering a defined set of target positions, executing the packet injection operation at these specific positions yields the traffic $x_{i^*} \cup x$ characterized by the minimal expected CMI:

$$\mathbb{E}_{x_{i^*}\cup x|x}\left[H\left(y \mid x_{i^*}, x\right)\right] < \mathbb{E}_{x_i|x}\left[H\left(y \mid x_i \cup x, x\right)\right] \quad \forall i \neq i^*. \tag{21}$$

□

The corresponding injection positions are identified by determining the positions that exhibit the highest MI with y. This is because:

$$I\left(x_{i^*} \cup x; y \mid x\right) = H\left(y \mid x\right) - \mathbb{E}_{x_{i^*}|x}\left[H\left(y \mid x_{i^*} \cup x, x\right)\right], \tag{22}$$

$H(y \mid x)$ remains constant throughout the analysis. Consequently, utilizing the results from the high-performance classifier, the policy network is capable of identifying positions that most significantly contribute to the CMI during each iteration of FRUGAL.

APPENDIX C
CONTRASTIVE LOG-RATIO UPPER BOUND

CLUB estimator is used to estimate the upper bound of MI. For the website traffic x, the label y can be obtained from the distribution $p(y|x)$, then the MI upper bound can be defined as:

$$\mathrm{I}_{CLUB}(x,y) = \mathbb{E}_{p(x,y)}[\log p(y \mid x)] - \mathbb{E}_{p(x)}\mathbb{E}_{p(y)}[\log p(y \mid x)]. \tag{23}$$

When we use our classifier $f_\phi(y|x)$ to derive $p(y|x)$, and M represents the number of monitored websites:

$$\begin{aligned}
\mathrm{I}_{\mathrm{vCLUB}} &= \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\left[\log f_\phi\left(y_i \mid x\right) - \log f_\phi\left(y_j \mid x\right)\right] \\
&= \frac{1}{M}\sum_{i=1}^{M}\left[\log f_\phi\left(y_i \mid x\right) - \frac{1}{M}\sum_{j=1,j\neq i}^{M}\log f_\phi\left(y_j \mid x\right)\right].
\end{aligned} \tag{24}$$

In the application of CLUB to achieve minimum MI, which entails minimizing the correlation between website traffic x and its corresponding label y, our RL framework necessitates the reduction of MI for each traffic sample throughout the iterative procedure. To this end, we compute the MI between the website traffic x and its label y_i, treating this as a positive sample. Conversely, the MI between the traffic and the remaining set of monitoring labels $y_j (j \neq i)$ is considered as the negative sample. The objective Equation (25) is to minimize the discrepancy between the positive and negative examples in order to attain minimal MI:

$$Loss = \log f_\phi\left(y_i \mid x\right) - \frac{1}{M}\sum_{j=1,j\neq i}^{M}\log f_\phi\left(y_j \mid x\right). \tag{25}$$

APPENDIX D
INJECTION POSITIONS ANALYSIS

Figure 12 presents heatmaps of packet injection positions for BWO limits of 10%, 20%, and 80%. In these visualizations, the x-axis represents the packet index within a traffic sequence, while the y-axis corresponds to distinct website labels. The intensity of the color at any point indicates the frequency of dummy packet injections.

A primary observation is that FRUGAL consistently "front-loads" dummy packets, concentrating them at the very beginning of the traffic sequence (approx. the first 700 packets). This strategy, which holds true across all sites and BWO limits, aligns with findings from [13] that the most uniquely identifying patterns reside in the traffic's "head."

Beyond this initial burst, subsequent injections are highly sparse and selective. The heatmaps reveal these positions are not unique but are often shared among many different websites. As BWO increases, the initial curtain becomes denser, and more of these shared, sparse positions are added.

FRUGAL not only neutralizes the most indicative features at the start of the communication but also carefully chooses later injection points to create ambiguity, forcing the traffic of different websites to conform to a common mold. This approach cleverly avoids introducing new, unique patterns that could inadvertently provide an adaptive attacker with fresh signals to fingerprint a website.

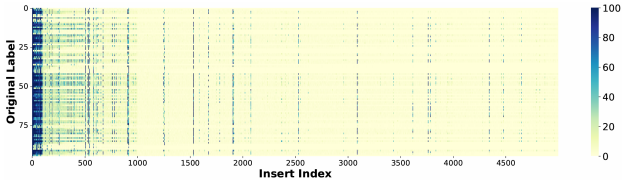
APPENDIX E
HYPERPARAMETERS TUNING

To determine if FRUGAL's performance is dependent on the architecture of the MI estimator, we conduct an evaluation using various attack models (e.g., DF, Var-CNN, NetCLR, TF, AWF, and RF) as the MI estimator. The experiments are performed under bandwidth overhead constraints of 20% and 30%. As shown in Table XI, the results demonstrate that FRUGAL maintains high performance irrespective of the estimator's architecture, a finding that holds for both closed- and open-world evaluations.

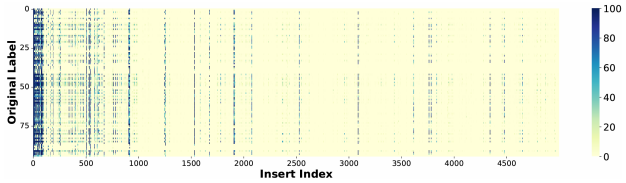
We investigate the impact of the kernel size and stride, denoted as K, on the system's performance by evaluating K values of 2, 5, 10 and 25. As presented in Table XII, a value of $K = 5$ yields the best results. This value is selected because

TABLE XI
PERFORMANCE OF DIFFERENT STRUCTURES OF MI ESTIMATOR

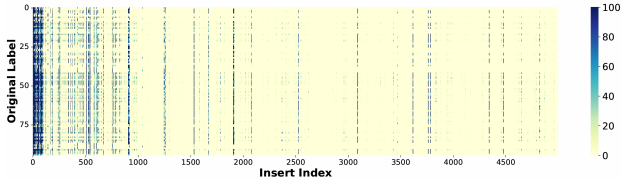
	Structures	Bandwidth Overhead	Attack Success Rate					
			DF	Var-CNN	NetCLR	TF	AWF	RF
CW	DF-based	20%	6.87%	8.03%	12.73%	10.37%	10.12%	16.6%
		30%	2.61%	2.63%	6.39%	5.67%	5.67%	12.7%
	Var-CNN-based	20%	28.76%	12.31%	25.82%	28.29%	21.34%	21.8%
		30%	21.53%	5.16%	18.6%	20.55%	12.32%	11.39%
	NetCLR-based	20%	8.03%	3.92%	15.34%	14.8%	10.8%	18.66%
		30%	5.57%	2.16%	9.84%	10.2%	6.36%	12.04%
	TF-based	20%	11.41%	10.98%	14.44%	15.34%	10.34%	15.85%
		30%	5.35%	4.05%	10%	10.08%	5.52%	13.41%
	AWF-based	20%	13.99%	16.68%	18.17%	18.78%	18.78%	22.17%
		30%	7.32%	6.84%	10.29%	11.77%	8.54%	18.6%
	RF-based	20%	10.52%	10.84%	14.84%	14.42%	8.83%	19.7%
		30%	5.24%	3.93%	6.62%	6.10%	3.43%	17.12%
OW	DF-based	20%	7.2%	6.55%	7.8%	5.7%	4.5%	13.43%
		30%	4.09%	4.7%	3%	2.17%	2.58%	10.85%
	Var-CNN-based	20%	12.67%	10.14%	7.97%	7.97%	3.93%	20.86%
		30%	5.59%	4.3%	3.16%	2.86%	2.75%	5.82%
	NetCLR-based	20%	12.51%	8.2%	6.82%	5.73%	4.69%	9.55%
		30%	6.99%	4.89%	3.82%	2.51%	2.16%	9.44%
	TF-based	20%	17.3%	9.14%	8.75%	9.53%	4.92%	11.67%
		30%	7.05%	3.68%	4.09%	2.52%	2.25%	4.5%
	AWF-based	20%	15.09%	11.26%	8.52%	7.96%	4.24%	8.18%
		30%	7.66%	3.67%	3.68%	2.84%	2.58%	3.68%
	RF-based	20%	9.8%	10.2%	11.3%	11.02%	6.2%	10.67%
		30%	4.24%	3.1%	5.79%	5.3%	3.03%	7.12%



(a) BWO of 20%



(b) BWO of 30%



(c) BWO of 80%

Fig. 12. **Injection Positions under BWO of 20%, 30%, 80%.**Under the various website and BWO limitations, the initial segment of FRUGAL's traffic accumulates a significant quantity of dummy packets.

it transforms the traffic sequence into a feature representation of appropriate dimensionality. This transformation helps mitigate the curse of dimensionality while enabling more precise dummy packet injection, thereby facilitating effective control over the final bandwidth overhead.

To determine the number of injection positions, we perform an experiment varying the parameter n over values of 1, 2, 5, and 10. These tests were conducted under a fixed 30% BWO constraint and with K fixed at 5. The empirical results, summarized in Table XIII, clearly identify $n = 5$ as the value that maximizes defensive efficacy under these conditions.

TABLE XII
DIFFERENT SELECTION OF K

	Attack Models	K			
		2	5	10	25
CW	DF	19.78%	2.68%	8.23%	22.4%
	Var-CNN	15.6%	2.61%	7.98%	30.17%
	TF	18.7%	5.67%	8.67%	28.32%
	NetCLR	10.04%	6.68%	14.32%	26.7%
	AWF	13.56%	5.73%	9.78%	28.6%
	RF	21.4%	12.7%	16.8%	45.1%
OW	DF	15.9%	4.09%	6.7%	19.42%
	Var-CNN	15.7%	4.7%	7.1%	26.6%
	TF	16.3%	2.17%	7.84%	25.2%
	NetCLR	11.2%	3%	14.24%	24.8%
	AWF	11.4%	2.58%	8.9%	26.71%
	RF	20.2%	4.85%	18.2%	43.28%

TABLE XIII
DIFFERENT SELECTION OF n

	Attack Models	n			
		1	2	5	10
CW	DF	23.36%	22.43%	2.68%	3.3%
	Var-CNN	13.23%	10.34%	2.61%	3.5%
	TF	28.4%	20.84%	5.67%	5.99%
	NetCLR	24.39%	18.41%	6.68%	8%
	AWF	13.04%	13.11%	5.72%	6.62%
	RF	13.04%	13.11%	5.72%	6.62%
OW	DF	25.9%	20.73%	4.09%	5.93%
	Var-CNN	17.1%	9.9%	4.7%	3.13%
	TF	27.28%	18.62%	2.17%	5.12%
	NetCLR	21.9%	17.67%	3%	7.8%
	AWF	14.3%	14.28%	2.58%	6.2%
	RF	19.2%	16.1%	4.85%	7.2%

APPENDIX F
ARTIFACT APPENDIX

This appendix is intended as a self-contained document presenting a roadmap for setting up and evaluating our artifact, FRUGAL.

A. Description & Requirements

This section lists all information necessary to recreate the experimental setup.

1) How to access: This artifact is publicly available and has been archived on Zenodo with the persistent identifier DOI: 10.5281/zenodo.17677723. The source code and latest updates are also accessible via GitHub at https://github.com/Junowww/FRUGAL-ndss.

2) Hardware dependencies:

- **GPU:** The training and evaluation presented in the paper were conducted on an Nvidia RTX A6000 GPU. A CUDA-enabled GPU is required to replicate the primary results (as specified by `--device cuda:0`).
- **CPU:** The artifact can also be executed in a CPU-only mode (by specifying `--device cpu`), though this will result in significantly slower training and evaluation times.
- **RAM / Disk:** A minimum of 32GB of RAM and 50GB of available disk space is recommended for storing the dataset and trained models.

3) Software dependencies:

- **OS:** Ubuntu 20.04 (or a compatible Linux distribution).
- **Framework:** PyTorch 2.0.
- **Python:** Python 3.9+
- **Dependencies:** All requisite Python dependencies are enumerated in the `mut info.yaml` environment file. This environment can be recreated using Conda.

4) Benchmarks:

- **Dataset:** This artifact requires the publicly available DF dataset collected by Sirinam et al..
- **Data Format:** The artifact expects this dataset to be pre-processed and organized according to the structure specified in the `Instruction.pdf`. The `dataset/` directory must contain the following files:
  - `train_data.pkl`
  - `train_labels.pkl`
  - `test_data.pkl`
  - `test_labels.pkl`

B. Artifact Installation & Configuration

The following steps outline the installation and configuration process required to prepare the evaluation environment.

1) **Clone the Repository:**
```
git clone git@github.com:Junowww/
FRUGAL-ndss.git
cd FRUGAL-ndss
```
2) **Create Conda Environment:** Use the provided `mut_info.yaml` file to create and activate the Conda environment. Alternatively, we provide a pre-built Docker image hosted on the Alibaba Cloud Registry; please refer to the README for usage instructions.
```
conda env create -f mut_info.yaml
conda activate mut_info
```
3) **Prepare Data:** Download and pre-process the DF dataset into the format described in section F-A4. Place the resulting `.pkl` files into the `dataset/` directory. The Goodsample dataset has been prearranged for utilization in training and test set has also been prepared for testing phases.
4) **Configure Paths:** Edit the `utility.py` file. Ensure the paths within the `LoadGoodSampleCW` and `LoadDataNoDefCW` functions correctly point to the `.pkl` files prepared in Step 3.

C. Experiment Workflow

The experimental workflow is divided into two primary stages:

1) **Training:** The `dqn_train_sac.py` script is used to train the FRUGAL policy network (Actor). This script loads a pre-trained attack model, which serves as the MI estimator, and the "Goodsample" training data. It trains the agent according to the specified bandwidth overhead (BWO) parameter. The trained Actor model is then saved to the `saved_trained_models/sac_models` directory.
2) **Evaluation:** The `cw_df_test_sac.py` script is used to evaluate the trained FRUGAL model. This script loads the test dataset, the saved Actor model from Stage 1, applies the defense to the test traffic, and then measures the Attack Success Rate (ASR) against the SOTA attack models (DF, Var-CNN, TF, AWF, NetCLR) evaluated in the paper.

D. Major Claims

The major claims for this artifact focus on its functionality. We claim that the artifact provides a functional, end-to-end workflow for training and evaluating the FRUGAL defense.

- **(C1): The artifact provides functional scripts to train a FRUGAL defense policy (Actor network) using the provided dataset and hyperparameters.**
  - **Evidence:** Executing the `dqn_train_sac.py` script will successfully load the data, run the training loop, and generate a trained policy model (e.g., a `.pth` file) as output.
- **(C2): The artifact provides functional scripts to evaluate a trained FRUGAL policy against a suite of Website Fingerprinting Attack (WFA) models.**
  - **Evidence:** Executing the `cw_df_test_sac.py` script will successfully load a trained policy, apply the defense to test traffic, and produce quantitative metrics (ASR and BWO) as output.
- **(C3): The artifact's training and evaluation workflow supports configurable BWO levels.**

– **Evidence:** By providing different values for the `--bwo_para` argument (e.g., `0.3` vs. `0.2`) to the scripts, the evaluator can observe that the workflow runs successfully under different configurations and produces different BWO metrics in the final output, demonstrating the control mechanism is functional.

E. Evaluation

This section provides the operational steps and experiments which must be performed to evaluate if the artifact is **functional** and validates the claims presented in Section D.

Experiment (E1):Verify Defensive Functionality at 30% BWO

- **[Estimated Time: 6+ hours(Nvidia A6000)]**
- **[Description]** This experiment trains a FRUGAL model and runs an evaluation to demonstrate its core functionality: applying defense to traffic, controlling bandwidth overhead, and measuring the resulting ASR.
- **[Preparation]** Execute the training script `dqn_train_sac.py` to train the model for 30% BWO.We assume `--bwo_para 0.3` corresponds to 30% BWO.

```
python dqn_train_sac.py \
        --device cuda:0 \
        --subdir frugal_cw_30bwo \
        --attack_model DF \
        --bwo_para 0.3 \
        --nb_classes 95
```

  Expected Output: The training process will print reward statistics. Upon completion, the trained model will be saved to `./saved_trained_models/`.
- **[Execution]** Execute the evaluation script `cw_df_test_sac.py`, loading the model trained in the previous step (`--subdir frugal_cw_30bwo`) and using the corresponding BWO parameter.

```
python cw_df_test_sac.py \
        --device cuda:0 \
        --subdir frugal_cw_30bwo \
        --attack_model DF \
        --bwo_para 0.3 \
        --nb_classes 95
```

- **[Results]**
- *Expected Output:* The script will print the final ASR for all evaluated attack models (DF, Var-CNN, TF, AWF, NetCLR) and the average BWO.
- *Functional Validation:* To confirm functionality, the evaluator should verify that:
  1) The script completes successfully without errors.
  2) It generates numerical ASR and BWO values for the different attack models.
  3) The resulting ASR values are significantly lower than the undefended baseline, and the BWO is reasonably close to the target parameter (0.3). This confirms the artifact successfully trained a policy and applied a functional defense.

Experiment (E2): Verify BWO Controllability at 20% BWO

- **[Estimated Time: 6+ hours (Nvidia A6000)]**
- **[Description]** This experiment demonstrates the BWO controllability aspect of claim (C3) by training and evaluating a model with a different BWO parameter (20%).
- **[Preparation]** Execute the training script, adjusting the `--bwo_para` to `0.2`.

```
python dqn_train_sac.py \
        --device cuda:0 \
        --subdir frugal_cw_20bwo \
        --attack_model DF \
        --bwo_para 0.2 \
        --nb_classes 95
```

- **[Execution]** Execute the evaluation script, loading the 20% BWO model and using the corresponding parameter.

```
python cw_df_test_sac.py \
        --device cuda:0 \
        --subdir frugal_cw_20bwo \
        --attack_model DF \
        --bwo_para 0.2 \
        --nb_classes 95
```

- **[Results]**
- *Expected Output:* The script will print ASR and BWO values.
- *Functional Validation:* The evaluator should verify that:
  1) The script completes successfully.
  2) It generates numerical ASR and BWO values.
  3) The reported `BWO` value is reasonably close to the target parameter (0.2). This confirms the functionality of the BWO control mechanism.