Technical Appendix: Inference and Adaptation Details for STAR

Purpose. This appendix provides implementation-level details of the inference-stage procedures used in **STAR**, omitted from the main paper due to strict page limits. It focuses on (i) zero-shot cross-modal retrieval with open-world rejection and (ii) two standard few-shot adaptation strategies: linear probing and Tip-Adapter-style fusion. These materials are intended to facilitate reproducibility and practical deployment, and do not introduce additional claims beyond those in the main paper.

Figure references. We refer to the inference-stage designs illustrated in the main paper (e.g., Fig. 3(a–c)). For clarity, these designs may be shown as three separate figures: (i) zero-shot retrieval, (ii) linear-probe-based adaptation, and (iii) Tip-Adapter-style fusion.

1 Notation and Setup

STAR uses a dual-encoder architecture with a *logic encoder* and a *traffic encoder*, each followed by a projection head that outputs normalized embeddings. We denote:

- Traffic modality input: T (e.g., packet-level representation).
- Logic modality input: L (e.g., resource-level representation extracted by crawling).
- Traffic embedding: $\mathbf{z}^T \in \mathbb{R}^d$.
- Logic embedding: $\mathbf{z}^L \in \mathbb{R}^d$.

Let $\mathrm{TrafficEnc}(\cdot)$ and $\mathrm{LogicEnc}(\cdot)$ denote the two encoders, and $f_T(\cdot)$ and $f_L(\cdot)$ denote the projection heads. Embeddings are ℓ_2-normalized to enable cosine similarity as an inner product.

2 Zero-Shot Inference via Cross-Modal Retrieval

2.1 Overview

In the zero-shot setting, STAR identifies the website corresponding to a traffic trace without requiring any labeled traffic samples from target websites during training, as shown in Fig. 1. Inference is formulated as a cross-modal retrieval problem:

- **Query:** an encrypted traffic trace (traffic modality).
- **Gallery:** a set of crawl-time logic profiles (logic modality) for monitored websites.

2.2 Logic-Side Gallery Construction

For each monitored website class $c \in \{1, \dots, C\}$, STAR pre-computes a logic-side prototype embedding.

Step 1: Extract logic representation. Obtain the crawl-time logic representation L_c (e.g., from browser logs).

Step 2: Encode and normalize. Compute the logic embedding:

$$\mathbf{z}_c^L = \frac{f_L(\mathrm{LogicEnc}(L_c))}{\|f_L(\mathrm{LogicEnc}(L_c))\|_2}. \tag{1}$$

Step 3: Store as gallery. Store $\{\mathbf{z}_c^L\}_{c=1}^C$ as the **gallery** for retrieval. This gallery is constructed offline and reused for all inference queries.

2.3 Traffic Query Embedding

Given a test traffic trace T, STAR computes a traffic embedding:

$$\mathbf{z}^T = \frac{f_T(\mathrm{TrafficEnc}(T))}{\|f_T(\mathrm{TrafficEnc}(T))\|_2}. \tag{2}$$

The traffic encoder remains frozen during inference.

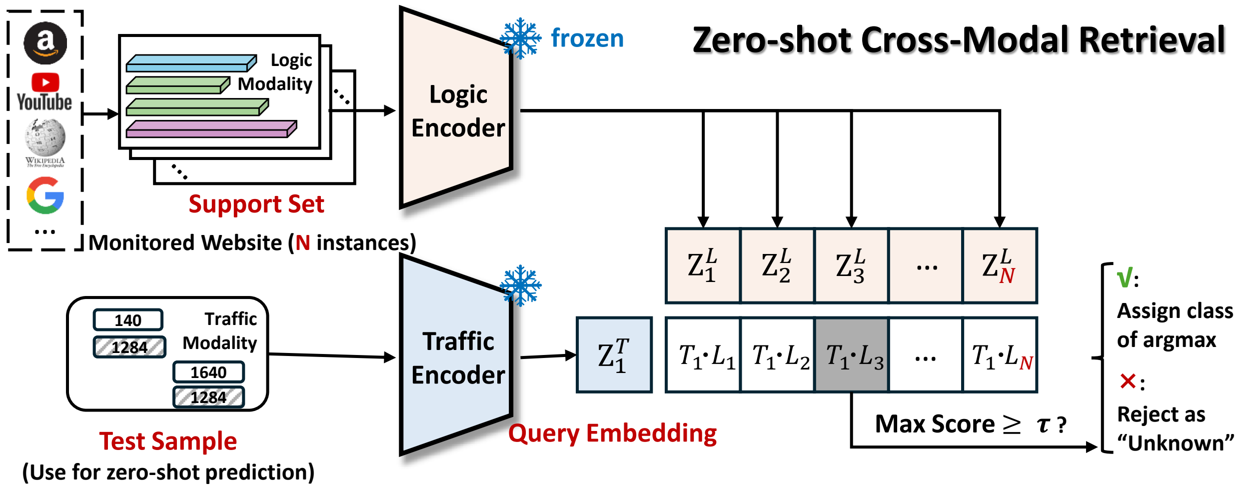

Figure 1: **Zero-shot** cross-modal retrieval with open-world rejection (corresponding to Fig. 3(a) in the main paper).

2.4 Cross-Modal Retrieval and Classification

STAR computes *cosine similarity* between the query embedding and each gallery entry:

$$s_c = \cos(\mathbf{z}^T, \mathbf{z}_c^L) = (\mathbf{z}^T)^\top \mathbf{z}_c^L. \tag{3}$$

The predicted class is obtained via nearest-neighbor retrieval:

$$\hat{c} = \arg\max_{c \in \{1,\dots,C\}} s_c. \tag{4}$$

2.5 Open-World Rejection via Thresholding

To support open-world recognition, STAR performs threshold-based rejection. Let $s^* = \max_c s_c$.

$$\text{Predict} = \begin{cases} \hat{c}, & \text{if } s^* \geq \tau, \\ \text{Unmonitored}, & \text{otherwise}, \end{cases} \tag{5}$$

where τ is a decision threshold selected on a validation set. Based on preliminary experiments that balance precision and recall across different operating points, we set $\tau = 0.6$ in all experiments unless otherwise specified.

Practical note. Because all embeddings are ℓ_2-normalized, cosine similarity reduces to a dot product, and the retrieval can be implemented efficiently via matrix multiplication between the query and a cached gallery matrix.

3 Few-Shot Adaptation via Linear Probe

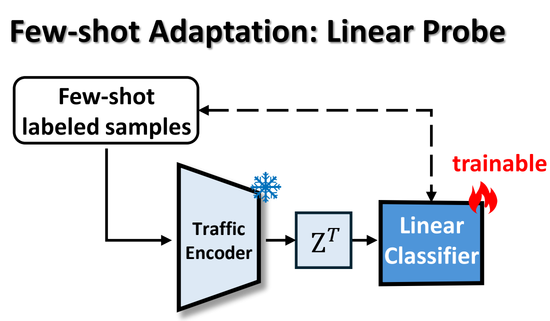

Figure 2: Few-shot adaptation via **linear probing** on frozen traffic embeddings (corresponding to Fig. 3(b) in the main paper).

3.1 Setting

In the few-shot setting, STAR is provided with a labeled support set:

$$\mathcal{S} = \{(T_i, y_i)\}_{i=1}^{N}, \qquad N = K \times C, \tag{6}$$

where K is the number of labeled samples per class, C is the number of monitored classes, and $y_i \in \{1, \dots, C\}$.

3.2 Training a Linear Classifier

Linear probing trains a lightweight classifier on top of *frozen* traffic embeddings (see Fig. 2).

Step 1: Extract embeddings. For each labeled support sample T_i, we extract a traffic-side embedding using the frozen traffic encoder and projection head:

$$\mathbf{z}_i^T = \mathbf{z}^T(T_i) = \frac{f_T(\mathrm{TrafficEnc}(T_i))}{\|f_T(\mathrm{TrafficEnc}(T_i))\|_2}. \quad (7)$$

These embeddings lie in a 256-dimensional space, corresponding to the output dimension of the projection head.

Step 2: Train linear head. We then train a linear classifier $g(\cdot)$ on top of the extracted traffic embeddings to predict the class label y_i:

$$\hat{y}_i = g(\mathbf{z}_i^T), \qquad \mathcal{L}_{\mathrm{CE}} = -\sum_{i=1}^{N} \log p(y_i \mid \mathbf{z}_i^T). \quad (8)$$

Here, $g(\cdot)$ is implemented as a fully-connected layer that maps the 256-dimensional traffic embedding to a C-dimensional class logit vector, where C is the number of monitored website classes. During training, only the parameters of the linear classifier are updated, while the traffic encoder, logic encoder, and all projection heads remain frozen.

3.3 Inference

At inference time, linear probing reduces STAR to a *single-modality* classifier operating solely on traffic embeddings. Given a test traffic trace T, we compute its embedding $\mathbf{z}^T$ using the frozen traffic encoder and predict the class label as:

$$\hat{y} = \arg\max_{c \in \{1,\ldots,C\}} \left[g(\mathbf{z}^T) \right]_c. \quad (9)$$

This design mirrors standard linear probing practices in multimodal representation learning (e.g., CLIP), where the pretrained encoder is treated as a fixed feature extractor and task adaptation is achieved through a lightweight linear classifier.

4 Few-Shot Adaptation via Tip-Adapter-Style Fusion

Tip-Adapter is a training-free or near-training-free adaptation approach that combines (i) a zero-shot prior from the aligned embedding space and (ii) a memory bank built from few-shot labeled examples (see Fig. 3). This design follows the Tip-Adapter framework proposed by Zhang *et al.* (2021), available at https://arxiv.org/abs/2111.03930.

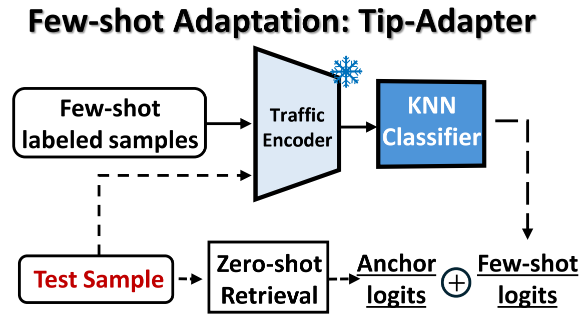

Figure 3: Few-shot adaptation via **Tip-Adapter-style** fusion between zero-shot anchors and kNN memory logits (corresponding to Fig. 3(c) in the main paper).

4.1 Memory Bank Construction

Given the labeled support set $\mathcal{S}$, we compute and store normalized traffic embeddings to form a memory bank:

$$\mathcal{M} = \{(\mathbf{z}_i^T, y_i)\}_{i=1}^{N}. \quad (10)$$

The memory bank is constructed once from the few-shot support set and remains fixed during inference.

4.2 kNN-Based Cache Aggregation

For a test embedding $\mathbf{z}^T$, we compute cosine similarity to each memory entry:

$$s_i = \cos(\mathbf{z}^T, \mathbf{z}_i^T) = (\mathbf{z}^T)^\top \mathbf{z}_i^T. \quad (11)$$

We adopt a memory-based kNN aggregation strategy, where support samples are treated as neighbors in the embedding space. Specifically, instead of selecting only the top-k nearest neighbors, we aggregate similarity scores from *all* support samples belonging to the same class:

$$\ell_c^{\mathrm{cache}} = \frac{1}{|\mathcal{S}_c|} \sum_{i: y_i = c} s_i, \quad (12)$$

where $\mathcal{S}_c$ denotes the set of support samples with label c. This formulation can be viewed as a class-wise kNN estimator with full neighborhood aggregation, which is empirically more stable in our setting.

4.3 Zero-Shot Anchor Logits

In parallel, we compute zero-shot anchor logits by measuring similarity between the traffic embedding and each logic-side prototype:

$$\ell_c^{\mathrm{ZS}} = \cos(\mathbf{z}^T, \mathbf{z}_c^L). \quad (13)$$

4.4 Logit Fusion and Prediction

We fuse the zero-shot anchor logits and the cache-based kNN scores via a linear combination:

$$\ell_c = \ell_c^{\text{ZS}} + \alpha \cdot \ell_c^{\text{cache}}, \tag{14}$$

where $\alpha \geq 0$ controls the contribution of the cache-based term. Based on empirical observations, we fix $\alpha = 5$ across all experiments. The final prediction is obtained as:

$$\hat{y} = \arg\max_{c \in \{1,\dots,C\}} \ell_c. \tag{15}$$

Practical notes.

- Tip-Adapter-style inference requires no encoder fine-tuning and introduces only a lightweight fusion hyperparameter α.
- The memory bank can be updated incrementally as new labeled samples become available, without retraining the encoders.
- If open-world rejection is required in the few-shot setting, a decision threshold can be applied either on the fused logits $\max_c \ell_c$ or on the underlying zero-shot similarity s^*, depending on the deployment preference.

5 Relationship Between the Three Inference Paradigms

The three inference modes form a unified spectrum:

- **Zero-shot retrieval:** inference relies solely on logic-side supervision and cross-modal alignment.
- **Linear probe:** supervised adaptation on frozen traffic representations via a trainable linear head.
- **Tip-Adapter fusion:** hybrid inference that combines retrieval-based alignment with a few-shot memory bank.

These correspond to the three inference-stage designs illustrated in the main paper (e.g., *Fig. 3(a–c)*).

6 Reproducibility Checklist (Short)

- All embeddings are ℓ_2-normalized, such that cosine similarity reduces to a dot product.
- The logic-side gallery $\{\mathbf{z}_c^L\}$ is pre-computed offline and cached for inference.
- Zero-shot inference is performed via nearest-neighbor retrieval with a fixed decision threshold τ for open-world rejection.
- Linear probe adaptation trains a single fully-connected classifier on frozen traffic embeddings.
- Tip-Adapter-style inference builds a memory bank from few-shot traffic samples and fuses zero-shot anchor logits with class-wise kNN similarity scores using a single fusion weight α.