Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.084041

ARTICLE

ET-BERT with Adapter Fusion: Operating-Regime Analysis of Modular Continual Adaptation for Encrypted Traffic Classification[#]

Minsu Kim[1], Daeho Choi[1], Younghyo Cho[1], Yeog Kim[2], Jun Lee[3], Changhoon Lee[1] and Kiwook Sohn[1,*]

[1]Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, Republic of Korea
[2]Research Center of Electrical and Information Technology, Seoul, Republic of Korea
[3]Korea Institute of Science and Technology Information (KISTI), Republic of Korea

*Corresponding Author: Kiwook Sohn. Email: kiwook@seoultech.ac.kr

[#]A preliminary version of this work was presented at MobiSec 2025. This article substantially extends the conference version with additional experiments and analyses

Received: 15 April 2026; Accepted: 15 June 2026; Published: 27 July 2026

ABSTRACT: Future mobile Internet and convergence applications increasingly rely on encrypted protocols, making security monitoring difficult because payload inspection is unavailable while traffic classes and threats evolve continuously. Encrypted traffic classification models must therefore adapt to newly emerging traffic classes without repeatedly overwriting or fully retraining large Transformer backbones. This study presents and extends an ET-BERT Adapter Fusion framework for AI/ML-driven encrypted-traffic security monitoring in future mobile Internet and convergence applications. The framework keeps the ET-BERT backbone frozen, trains a Base Adapter on USTC-TFC2016 classes 0–9, trains an Incremental Adapter for class 10, and composes them through Adapter Fusion. Across three seeds, Adapter Fusion in the core USTC-TFC2016 10 + 1 setting reached 0.9947 Macro-F1, close to Full Fine-Tuning (0.9978) and Consolidated Adapter training on classes 0–10 (0.9967), while preserving a modular update structure. A controlled Fusion Boundary study over 2–11 fused adapters showed a limited decrease in Macro-F1 from 0.9947 to 0.9681 rather than an abrupt collapse. Consolidated Adapter capacity experiments showed strong USTC-TFC2016 performance up to 20 classes and a clearer scale-dependent decline on CSTNET-TLS 1.3 from 0.9238 Macro-F1 at 10 classes to 0.7944 at 120 classes. Latency measurements further show that Adapter Fusion has higher inference overhead than Full Fine-Tuning and single-adapter alternatives. The conclusion is therefore not that Adapter Fusion is universally faster or more accurate, but that it offers a modular continual-adaptation regime for AI/ML-driven security in future mobile Internet and convergence applications, with benefits and limitations that depend on adapter count, data availability, and deployment constraints.

KEYWORDS: Encrypted traffic classification; adapter fusion; parameter-efficient transfer learning; continual learning; ET-BERT

1 Introduction

Today, a significant portion of network traffic is transmitted based on encrypted protocols, which have become core technologies for strengthening security and privacy. However, such encryption also acts as a factor that increases the difficulty of security threat detection. Traditional packet analysis techniques or signature-based detection methods face limitations in effectively classifying normal and malicious traffic in encrypted environments, as they cannot access meaningful payload information. Consequently, recent

research has evolved toward developing machine learning and deep learning classifiers that leverage the metadata and sequence patterns of encrypted traffic itself [1].

In particular, the success of Transformer-based language models has opened new possibilities in the security domain. Transformer-based encrypted traffic classification models such as ET-BERT process traffic sequences as natural language tokens, achieving high classification accuracy [2]. However, since existing models have statically trained structures, when new types of attack traffic emerge, the entire model must be retrained, leading to inefficiency. This approach entails high training costs and storage resource consumption, and poses the risk of catastrophic forgetting that degrades existing performance [3]. Especially in situations where new traffic types appear in real time, retraining the entire model is impractical in terms of training speed, making rapid response difficult.

Incremental learning has been gaining attention as a method to address this issue. Incremental learning is designed to adapt to new tasks or classes while retaining previously learned models [4]. However, when applied to the security domain, the key challenge is to minimize forgetting of prior knowledge while quickly and efficiently learning new attacks [5].

In this study, we propose and analyze an adapter-based incremental learning framework built on the ET-BERT model. The adapter technique inserts small modules into each Transformer layer and improves parameter efficiency by training only the modules instead of the entire parameter set [6]. Adapter Fusion then composes multiple frozen adapters to integrate prior and new knowledge without destructively overwriting the source modules [7]. Rather than assuming that this modular design is universally superior, this study treats the central research question as an operating-regime problem: when is modular composition preferable to full fine-tuning or a single Consolidated Adapter trained on the expanded class set?

The main contributions of this study can be summarized as follows:

1. We formulate an ET-BERT-based modular continual-adaptation framework that separates a Base Adapter for base classes, an Incremental Adapter for newly emerging classes, and an Adapter Fusion stage for knowledge composition.
2. We provide a controlled comparison among Full Fine-Tuning, Base Adapter source-stage evaluation, Incremental Adapter diagnostics, Adapter Fusion, Consolidated Adapter training, and method-specific rehearsal-free continual-learning references under clearly separated regimes.
3. We characterize the operating range of Adapter Fusion using multi-seed Fusion Boundary experiments, Consolidated Adapter capacity experiments on USTC-TFC2016 and CSTNET-TLS 1.3, and deployment-oriented runtime/latency measurements.

Building on the preliminary conference version, this article provides a broader analysis of scalability, adapter capacity, and practical operating characteristics under extended continual-learning scenarios [8].

2 Related Work

Encrypted traffic classification and incremental learning have each been actively studied in their own right, but attempts to integrate the two research streams remain at an early stage. Prior work has primarily focused on designing static models to achieve high accuracy, relying on retraining the entire model whenever new attacks or service types emerge [1]. In contrast, in the field of continual learning, various algorithmic and architectural techniques have been proposed across domains to mitigate catastrophic forgetting; however, there are few cases that directly apply these techniques to the special environment of security traffic (high-dimensionality, sequence-based features, etc.) [3,4].

2.1 Encrypted Traffic Classification with Transformers

Encrypted traffic classification must compensate for the limitations of traditional Deep Packet Inspection (signature) methods due to restricted payload visibility. Recently, approaches that regard traffic patterns as sequences and leverage large-scale pre-trained representations have gained traction, with ET-BERT being a representative example. Lin et al. [2] demonstrated that ET-BERT, pre-trained on large-scale unlabeled traffic, achieved substantial performance gains over existing methods on various downstream tasks (ISCX-Tor [9], ISCX-VPN-Service [10], etc.) (e.g., ISCX-Tor F1 99.2%). This study follows that line of work, but explores how to simultaneously achieve knowledge retention and rapid adaptation of ET-BERT in an incremental learning setting.

Meanwhile, publicly available datasets such as USTC-TFC2016 [11] are widely used as benchmarks for encrypted (or mixed) traffic classification and continue to serve as baselines for subsequent studies. For example, BSTFNet proposed in 2024 reported accuracy and F1 of about 99.4% on USTC-TFC2016, improving the separability of malicious traffic by integrating global semantic features with spatiotemporal features [12].

2.2 Catastrophic Forgetting & Class-Incremental Learning

In incremental/continual learning, when additional training is performed using only new-class data, the central challenge is the problem of catastrophic forgetting, where past classes are rapidly forgotten. A 2023 comprehensive survey summarized class-incremental learning (CIL) methods and emphasized the need for aligning memory budgets and memory-agnostic metrics for fair comparison [4]. This perspective is especially important in situations where rehearsal is difficult due to sensitive data, as in the security domain. Moreover, a 2023 study pointed out that the use of rehearsal memory can entail security/privacy risks and raised the need for methods that reduce forgetting without data reuse [5]. The reason this study examines Adapter Fusion also stems from a practical requirement to combine knowledge at the module level without re-storing past data. Representative continual-learning baselines include regularization-based approaches such as elastic weight consolidation and Learning without Forgetting, as well as class-incremental methods such as iCaRL [13–15].

In security domains, the long-term storage or reuse of sensitive traffic data can be strictly limited by privacy regulations (e.g., data minimization under GDPR). Motivated by such constraints, this study focuses on rehearsal-free, parameter-efficient approaches (adapters and fusion) that do not require re-storing past data, and excludes rehearsal-based continual learning (CL) methods from direct comparison.

2.3 Parameter-Efficient Transfer Learning (Adapters)

When transferring large-scale pre-trained models to downstream tasks, fine-tuning all parameters is inefficient because a full replica is required for each task. Houlsby et al. (2019) inserted small bottleneck modules (adapters) between transformer layers and showed that learning only a very small number of parameters per task can achieve performance comparable to full fine-tuning [6]. MAD-X extended adapter-based transfer to multi-task cross-lingual settings, showing that modular adapter composition is useful beyond single-task adaptation, but it is not a general PEFT taxonomy survey [16]. These adapter advances offer a practical alternative for environments like security traffic classification, where tasks/classes are frequently added, by reducing storage and deployment burdens.

Furthermore, Hu et al. proposed LoRA (Low-Rank Adaptation), which implements efficient parameter learning using low-rank matrices rather than adapter bottleneck modules [17]. LoRA is a PEFT adaptation method, not by itself a forgetting-aware continual-learning algorithm. When LoRA is combined with a teacher-distillation mechanism, we therefore label it explicitly as LoRA+LwF rather than pure LoRA. More broadly, the PEFT literature has expanded through practical adapter ecosystems and prompt-based

tuning strategies, including AdapterHub, Prefix-Tuning, Prompt Tuning, and PEFT survey/systematization work [18–21].

2.4 Knowledge Composition via Adapter Fusion

Adapter Fusion was proposed as an approach to non-destructively combine adapters trained on multiple tasks. Pfeiffer et al. empirically demonstrated over 16 natural language understanding (NLU) tasks that by decoupling knowledge extraction (training adapters per task) from knowledge composition (training fusion layers), one can mitigate the forgetting and balance issues of sequential fine-tuning/multi-task learning [7]. This study extends that idea to class-incremental security classification by integrating a base-task adapter and a new-class adapter via Fusion, thereby exploring a lightweight solution that preserves existing knowledge while rapidly injecting new knowledge.

In addition, Karimi Mahabadi et al. (2021) proposed Compacter, a new adapter architecture that leverages parameter sharing and high-order tensor factorization techniques to remain even more compact than Adapter Fusion while maintaining high performance [22]. Compacter experimentally showed that, when combined with adapter fusion techniques in multi-task environments, it can maximize storage efficiency as well as mitigate forgetting. This suggests high applicability even in environments such as security traffic, where new tasks continue to be added [23].

In this study, we combine the research axes of encrypted traffic classification and incremental learning and focus on applying incremental learning techniques to Transformer-based encrypted traffic classification models—particularly parameter-efficient transfer learning based on adapters and knowledge integration through Adapter Fusion.

3 Proposed Method

In this study, to address the problem of failing to respond to new traffic types during encrypted traffic classification, we apply parameter-efficient transfer learning (PEFT) based on a Transformer pre-trained model (ET-BERT) in an incremental learning environment and examine Adapter Fusion as a modular knowledge-composition technique.

Throughout this paper, we use the following terminology. *Base Adapter* refers only to the adapter trained on the base USTC-TFC2016 classes 0–9. *Incremental Adapter* refers to the class-10 source adapter in the core USTC-TFC2016 10 + 1 setting, or to the corresponding class-specific source adapters used in the Fusion Boundary analysis. *Adapter Fusion* refers to a composition model that loads frozen source adapters and trains fusion parameters for the expanded evaluation set. *Consolidated Adapter* refers to a single adapter trained directly on an expanded class set such as USTC-TFC2016 0–10, USTC-TFC2016 0–14, USTC-TFC2016 0–19, or CSTNET-TLS 1.3 subsets. This distinction is important because the Base Adapter is a source-stage component of Adapter Fusion, whereas the Consolidated Adapter is the direct single-adapter comparison for expanded class sets.

The resulting framework is summarized in Fig. 1. Traffic byte sequences are encoded by the frozen ET-BERT backbone. The Base Adapter and Incremental Adapter are trained as separate source modules, then loaded into the Adapter Fusion stage, where only fusion parameters are trained before the final classifier predicts the expanded class set.

3.1 Problem Definition

The encrypted traffic classification problem is defined as a multi-class classification problem that maps input data x to one of the pre-defined class sets C. In this study, the input data are encrypted traffic at the

packet sequence and flow level, so the data are represented as integer sequences in bytes. Each sample x is normalized into a sequence of length L ($x = (x_1, x_2, \ldots, x_L)$), where insufficient length is padded with tokens 0. This input value is fed into the Transformer encoder, and the model outputs a class probability distribution $\hat{y} \in \mathbb{R}^C (\hat{y} = f_\Theta(x), \sum_{c=1}^{C} \hat{y}_c = 1)$. In the incremental learning environment, the class set expands over time step t (See Eq. (1)).

$$C_t = C_{t-1} \cup C_{\text{new}}. \tag{1}$$

That is, initially only C_{base} classes exist, and whenever a new traffic type occurs, C_{new} is added. Therefore, the goal of this study is to enable adaptation to these new classes while maintaining classification performance on existing classes.

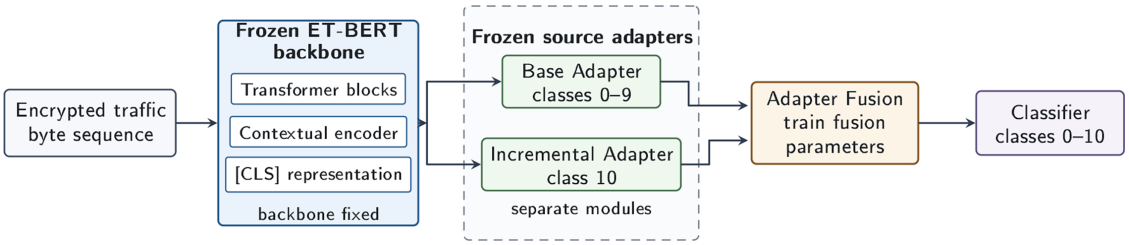

Figure 1: ET-BERT with adapter fusion pipeline for the core USTC-TFC2016 10 + 1 setting. Encrypted traffic byte sequences are encoded by a frozen ET-BERT backbone. A base adapter trained on classes 0–9 and an incremental adapter trained on class 10 are kept as separate modules, and adapter fusion learns trainable fusion parameters over their representations before the final classifier for classes 0–10.

3.2 Baseline: Full Fine-Tuning with Transformers

The starting point of this study is the ET-BERT [2] model based on a Transformer encoder architecture. Input sequences are converted into high-dimensional representations through an embedding layer, and then into contextualized hidden vectors through multi-layer self-attention blocks. During this process, padding tokens are ignored via an attention mask, and only actual data positions are included in meaningful computation.

Finally, the extracted [CLS] token is regarded as the vector representing the entire sequence, which is then used to construct the classifier. The classifier consists of a linear transformation and a softmax layer to predict the input sequence as one of the C classes. During training, cross-entropy loss is generally used, which is the most widely used metric for multi-class classification problems (See Eq. (2)).

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c}. \tag{2}$$

where N is the batch size, C is the number of classes, and $y_{i,c}$ is the one-hot label. This approach has the advantage of high classification performance but suffers from the problem that all parameters must be re-trained whenever new classes are added. In other words, as the number of tasks increases, model storage cost and training time increase linearly, which is a fundamental limitation.

3.3 Parameter-Efficient Incremental Learning with Adapters

Instead of training all parameters, in this study we secure parameter efficiency by inserting adapter modules into each Transformer block. An adapter has a bottleneck structure that temporarily projects the

input vector into a low-dimensional space and then expands it back to the original dimension. For a hidden vector $h \in \mathbb{R}^d$, the adapter transformation is defined as follows (See Eq. (3)).

$$h' = h + W_{\text{up}}\, \sigma(W_{\text{down}}\, h)\,. \tag{3}$$

where $W_{\text{down}} \in \mathbb{R}^{r\times d}$, $W_{\text{up}} \in \mathbb{R}^{d\times r}$, $r \ll d$, and σ is a nonlinear activation function.

This structure provides three advantages. First, since only a small fraction of the entire model parameters are trained, update-time parameter costs and storage requirements are reduced. Second, independent adapters can be stored per task or class group, making model management easier when new tasks are added. Third, the existing Transformer parameters are frozen, making it possible to preserve source modules without destructive overwriting. Therefore, adapters can serve as useful modules in incremental learning environments.

When new classes emerge in the incremental learning environment, the existing model weights are left unchanged, and only new adapters are trained. That is, the pre-trained ET-BERT backbone is kept fixed, and an incremental adapter specialized for the new classes is trained in isolation. This incremental adapter is trained on new-class samples and is evaluated as a source module or diagnostic expert, not as a complete classifier over all old and new classes.

This approach does not use rehearsal techniques, so it reduces the security and privacy concerns associated with data reuse [5]. It also has the advantage of adapting to new classes with only a small number of trainable parameters. However, such an independent training process does not itself preserve predictions for existing classes [3,4]. In other words, while the new class may be classified accurately, performance on existing classes is sacrificed unless an explicit retention or composition mechanism is used. Therefore, additional knowledge integration methods are necessary.

3.4 Adapter Fusion and Optimization

To compensate for the limitations of the incremental adapter method, this study introduces Adapter Fusion [7,22]. Adapter Fusion can be viewed as composing the representations produced by multiple adapters through a learned attention-like mixture, thereby enabling prior knowledge and newly acquired knowledge to be integrated without destructive overwriting.

For exposition, the normalized composition view of K adapters can be written as follows (See Eq. (4)).

$$h^{(\ell)}_{\text{fusion}}(x) = \sum_{k=1}^{K} \alpha^{(\ell)}_k(x) h^{(\ell)}_k(x), \qquad \sum_{k=1}^{K} \alpha^{(\ell)}_k(x) = 1. \tag{4}$$

where $h^{(\ell)}_k(x)$ denotes the representation produced by the k-th adapter at layer ℓ, and $\alpha^{(\ell)}_k(x)$ denotes the corresponding normalized mixture weight in this conceptual view.

Eq. (4) is a conceptual normalized composition equation, not a statement that the implementation manually enforces one fixed scalar coefficient per adapter. In the implementation, we use the Adapter Fusion mechanism provided by the adapters framework: source adapters are loaded and frozen, while the framework trains fusion/attention parameters that compute input- and layer-dependent adapter combinations. Thus, α in Eq. (4) should be interpreted as an abstraction of learned fusion attention rather than as a manually tuned scalar hyperparameter.

The core idea is that the base adapter preserves previously learned representations for existing classes, whereas the incremental adapter contributes information specialized for newly emerging classes. By activating these adapters jointly, the framework can balance stability and adaptability through a learned

combination of their outputs. In this way, Adapter Fusion serves not merely as a simple merger of modules, but as a knowledge-composition mechanism that reduces destructive overwriting while preserving rapid responsiveness to new classes.

Training is performed with cross-entropy loss and optimizer-level weight decay. In the baseline phase, the model is optimized with AdamW and a warmup-based learning-rate schedule. In the adapter-based phases, training is carried out through the Hugging Face/adapters training pipeline with warmup and weight decay. In the fusion stage, the pre-trained source adapters are loaded and kept fixed, while the fusion parameters are optimized through the Adapter Fusion training mechanism provided by the adapters framework. This design keeps the source adapters reusable while allowing the model to learn how to compose existing and new knowledge. Because fusion activates multiple adapters at inference time, however, the method can introduce inference latency overhead; this is measured explicitly in Section 4.2.

4 Experiments

In this section, we experimentally verify the classification performance and operating limits of the proposed adapter-based incremental learning framework. The objectives of the experiments are fourfold. First, we compare the core USTC-TFC2016 10 + 1 setting against Full Fine-Tuning, Base Adapter source-stage evaluation, Incremental Adapter diagnostics, Adapter Fusion, Consolidated Adapter training, and method-specific rehearsal-free continual-learning references. Second, we analyze how Adapter Fusion behaves as more source adapters are composed. Third, we compare this behavior with Consolidated Adapter capacity on USTC-TFC2016 and CSTNET-TLS 1.3. Fourth, we report runtime, parameter, and latency evidence so that efficiency claims are separated into update cost and inference cost.

4.1 Experimental Setup

In this study, we used the publicly available USTC-TFC2016 [11] dataset for experiments. The dataset consists of 10 normal traffic classes (e.g., Web, Email, Chat, etc.) and 10 malicious traffic classes (e.g., Zeus, Virut, Neris, etc.), each containing thousands of network flows. Each flow is composed of multiple packets, and in this study, payload byte sequences were normalized to a fixed length of 512 for input into the Transformer model.

The core USTC-TFC2016 10 + 1 split was defined as follows (See Table 1).

Table 1: Dataset split per class (Train/Valid/Test).

Class ID	Class Type	Train (80%)	Valid (10%)	Test (10%)
0–9	Base	4000 samples	500 samples	500 samples
10	Incremental	4000 samples	500 samples	500 samples

Classes 0–9 are the base classes used in the initial training stage, and class 10 is the class added during the incremental learning stage. For each class, the data was split into an 80:10:10 ratio for training, validation, and testing. The split (Base 0–9, Incremental 10) serves as a proxy scenario for sudden zero-day or variant outbreaks in a security operations center, rather than a mere random partition. Using the standard USTC-TFC2016 benchmark in this way allows us to assess relative responsiveness to newly emerging classes, while the additional CSTNET-TLS 1.3 experiments provide a larger and more modern encrypted-traffic capacity check. Following the ET-BERT benchmark context, CSTNET-TLS 1.3 is an anonymized real-world TLS 1.3 dataset collected on the China Science and Technology Network from March to July 2021 and released through the ET-BERT dataset source [2,24].

All training and latency runs used one physical NVIDIA RTX 3090 (24 GB) GPU per run. Results are reported over seeds 7, 21, and 42 when applicable. All reported values were computed from documented runs with recorded commands, seeds, dataset splits, summary artifacts, and hardware notes. Previously validated reference runs are used only where their summaries and class-set definitions were confirmed.

For performance evaluation, two metrics were used.

- **Accuracy:** Ratio of correctly classified samples among all samples (See Eq. (5)).

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i) \tag{5}$$

- **Macro-F1:** Arithmetic mean of F1 Scores per class, used to assess robustness against class imbalance (See Eq. (6)).

$$\text{Macro-F1} = \frac{1}{C}\sum_{c=1}^{C} \frac{2P_cR_c}{P_c + R_c}, \quad P_c = \frac{TP_c}{TP_c + FP_c}, \quad R_c = \frac{TP_c}{TP_c + FN_c} \tag{6}$$

Beyond the core USTC-TFC2016 10 + 1 class-incremental setting, we conducted extended analyses to examine the practical scalability of the framework. Specifically, we investigated (i) how fusion performance changes as the number of fused source adapters increases from 2 to 11, (ii) how many classes a single Consolidated Adapter can absorb on USTC-TFC2016, and (iii) whether this capacity trend remains consistent in the larger CSTNET-TLS 1.3 dataset. The CSTNET 60-class subset uses canonical contiguous class identifiers 0–59, matching the same label-mapping convention as the 10- and 120-class anchors. Forward Transfer (FWT) is omitted because the single-step 10 + 1 protocol does not define a meaningful pre-update future-class evaluation before class 10 is introduced.

4.2 Results and Analysis

4.2.1 Core USTC-TFC2016 10 + 1 Continual-Learning Setting

The core USTC-TFC2016 10 + 1 setting evaluates the modular adaptation scenario: a Base Adapter is trained on classes 0–9, an Incremental Adapter is trained for class 10, and Adapter Fusion composes the frozen source adapters for evaluation on classes 0–10. Table 2 separates this modular row from Full Fine-Tuning and from the direct single-adapter competitor, namely the Consolidated Adapter trained on all 11 classes. The Base Adapter row is included only as a source-stage reference on classes 0–9; it is not a direct 0–10 competitor.

Table 2: USTC-TFC2016 core 10 + 1 comparison. Values are mean ± std over seeds 7, 21, and 42.

Method	Role	Train/Eval Classes	Acc	Macro-F1	Time (s)	Params (Trainable/ Total)
Full Fine-Tuning	Joint full-model reference	0–10/0–10	0.9978 ± 0.0008	0.9978 ± 0.0008	2049.9 ± 15.8	132.13M/132.13M
Base Adapter	Source-stage reference only	0–9/0–9	0.9973 ± 0.0005	0.9973 ± 0.0005	1544.9 ± 7.4	1.49M/133.62M
Incremental Adapter	New-class expert diagnostic	10/0–10	0.0909 ± 0.0000	0.0152 ± 0.0000	154.7 ± 0.4	1.49M/133.62M

(Continued)

Table 2 (continued)

Method	Role	Train/Eval Classes	Acc	Macro-F1	Time (s)	Params (Trainable/ Total)
Adapter Fusion	Proposed modular composition	source adapters/0–10	0.9947 ± 0.0006	0.9947 ± 0.0006	2397.9 ± 14.2	14.77M/148.69M
Consolidated Adapter	Direct single-adapter comparison	0–10/0–10	0.9967 ± 0.0002	0.9967 ± 0.0002	1687.4 ± 6.1	1.49M/133.62M
EWC strict CL	New-task-only CL reference	10/0–10	0.0909 ± 0.0000	0.0152 ± 0.0000	234.8 ± 1.5	132.13M/132.13M
LwF strict CL	New-task-only CL reference	10/0–10	0.5836 ± 0.1798	0.5466 ± 0.1785	331.1 ± 4.7	132.13M/132.13M
LoRA+LwF	PEFT retention variant	10/0–10	0.8330 ± 0.0300	0.8099 ± 0.0417	320.4 ± 2.7	0.45M/132.59M

Note: The parameter column reports trainable/total parameters. Full Fine-Tuning, EWC, and LwF update the full model in their reported regimes, whereas adapter and LoRA-based rows train only parameter-efficient modules, fusion parameters, or the task head.

The results show that Full Fine-Tuning and Consolidated Adapter training are strong when joint or consolidated access to the expanded class set is allowed. Adapter Fusion is slightly lower in overall Macro-F1, but it preserves the modular separation between the Base Adapter and the Incremental Adapter. The standalone Incremental Adapter and EWC strict CL rows reach the new class but collapse on old classes in this protocol, illustrating why a retention or composition mechanism is required. LoRA+LwF is reported as a distillation-based PEFT retention variant, not as pure LoRA. To further separate old-class retention from new-class adaptation, Table 3 reports regime-separated old/new group Macro-F1 for the USTC-TFC2016 10 + 1 setting.

Table 3: Regime-separated old/new group metrics in the USTC-TFC2016 10 + 1 setting. Values are mean ± std over seeds 7, 21, and 42.

Regime	Method	Strict New-Task-Only CL?	Old Macro-F1	New Macro-F1	Overall Macro-F1
Joint/consolidation reference	Full Fine-Tuning	No	0.9984 ± 0.0007	0.9970 ± 0.0010	0.9978 ± 0.0008
Joint/consolidation reference	Consolidated Adapter	No	0.9975 ± 0.0003	0.9950 ± 0.0000	0.9967 ± 0.0002
Modular composition	Adapter Fusion	Not a strict CL baseline	0.9967 ± 0.0006	0.9892 ± 0.0006	0.9947 ± 0.0006
Strict new-task-only CL	EWC strict CL	Yes	0.0000 ± 0.0000	1.0000 ± 0.0000	0.0152 ± 0.0000
Strict new-task-only CL	LwF strict CL	Yes	0.5692 ± 0.1881	1.0000 ± 0.0000	0.5466 ± 0.1785

(Continued)

Table 3 (continued)

Regime	Method	Strict New-Task-Only CL?	Old Macro-F1	New Macro-F1	Overall Macro-F1
Strict new-task-only CL/PEFT retention	LoRA+LwF	Yes	0.8385 ± 0.0417	0.9990 ± 0.0010	0.8099 ± 0.0417

Note: Forward Transfer (FWT) is omitted because the single-step 10 + 1 protocol does not define a meaningful pre-update future-class evaluation before class 10 is introduced.

4.2.2 Fusion Boundary under Multiple Source Adapters

To examine how Adapter Fusion behaves beyond the single-increment case, we progressively fused source adapters over USTC-TFC2016 class ranges 0–10 through 0–19. The three-seed aggregate is shown in Fig. 2 and Table 4. All reported Fusion Boundary runtimes were measured with one physical GPU per run.

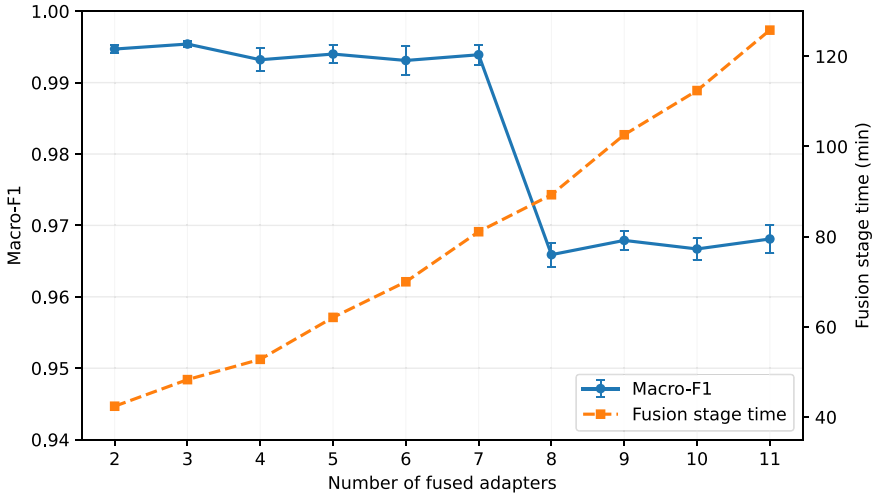

Figure 2: Fusion boundary trend on USTC-TFC2016. Macro-F1 remains high from 2–7 adapters, drops at the 7–8 transition, and then stabilizes through 11 adapters; fusion-stage time increases with the number of fused adapters. Macro-F1 error bars indicate standard deviation over seeds 7, 21, and 42.

Table 4: Adapter fusion boundary results on USTC-TFC2016. Values are mean ± std over seeds 7, 21, and 42. The time column measures the fusion-stage run only after the source adapters already exist, not end-to-end source-adapter training plus fusion.

# Adapters	Class Range	Accuracy	Macro-F1	Old Macro-F1	New Macro-F1	Fusion Stage Time (min)
2	0–10	0.9947 ± 0.0006	0.9947 ± 0.0006	0.9967 ± 0.0006	0.9892 ± 0.0006	42.4 ± 3.8
3	0–11	0.9954 ± 0.0004	0.9954 ± 0.0004	0.9971 ± 0.0004	0.9948 ± 0.0013	48.3 ± 0.2
4	0–12	0.9937 ± 0.0013	0.9932 ± 0.0016	0.9964 ± 0.0012	0.9934 ± 0.0006	52.8 ± 0.5
5	0–13	0.9946 ± 0.0010	0.9940 ± 0.0013	0.9971 ± 0.0008	0.9946 ± 0.0003	62.1 ± 0.3
6	0–14	0.9936 ± 0.0016	0.9931 ± 0.0020	0.9967 ± 0.0011	0.9937 ± 0.0009	70.0 ± 0.2
7	0–15	0.9945 ± 0.0012	0.9939 ± 0.0014	0.9967 ± 0.0009	0.9954 ± 0.0008	81.1 ± 0.4
8	0–16	0.9656 ± 0.0010	0.9659 ± 0.0017	0.9833 ± 0.0009	0.9745 ± 0.0024	89.3 ± 1.4
9	0–17	0.9676 ± 0.0008	0.9679 ± 0.0013	0.9830 ± 0.0006	0.9783 ± 0.0015	102.6 ± 0.2
10	0–18	0.9665 ± 0.0010	0.9667 ± 0.0015	0.9839 ± 0.0012	0.9746 ± 0.0020	112.4 ± 0.8
11	0–19	0.9681 ± 0.0018	0.9681 ± 0.0020	0.9834 ± 0.0028	0.9771 ± 0.0048	125.8 ± 0.5

The Fusion Boundary result refines the operating-regime interpretation. Under the controlled multi-seed protocol, Adapter Fusion does not abruptly collapse by 11 adapters. Instead, Macro-F1 decreases by 2.65 percentage points from count 2 to count 11, with the main transition occurring between counts 7 and 8. This suggests a gradual operating-regime boundary: fusion remains highly reliable in the low-to-mid adapter range, but the cost of composing many independently trained source adapters appears in both accuracy and training time.

We further inspected prediction-distribution diagnostics around this transition. At adapter counts 7, 8, and 11, the number of unique predicted labels matched the expected class counts (16, 17, and 20), the maximum predicted-label fraction remained low (0.0655, 0.0622, and 0.0548), and normalized prediction-label entropy remained close to 1.0 (0.9954, 0.9958, and 0.9966). Thus, the count 7–8 drop is not explained by a degenerate one-label or few-label prediction collapse. These diagnostics rule out a degenerate prediction-concentration failure mode, but they do not by themselves identify the underlying cause of the 7–8 transition; deeper fusion-weight allocation and leave-one-adapter-out analyses remain future work.

4.2.3 Consolidated Adapter Capacity on USTC-TFC2016 and CSTNET-TLS 1.3

To separate class capacity from adapter-composition difficulty, we trained or verified Consolidated Adapter rows that learn the included classes within one adapter. Fig. 3 and Table 5 summarize the USTC-TFC2016 and CSTNET-TLS 1.3 capacity results.

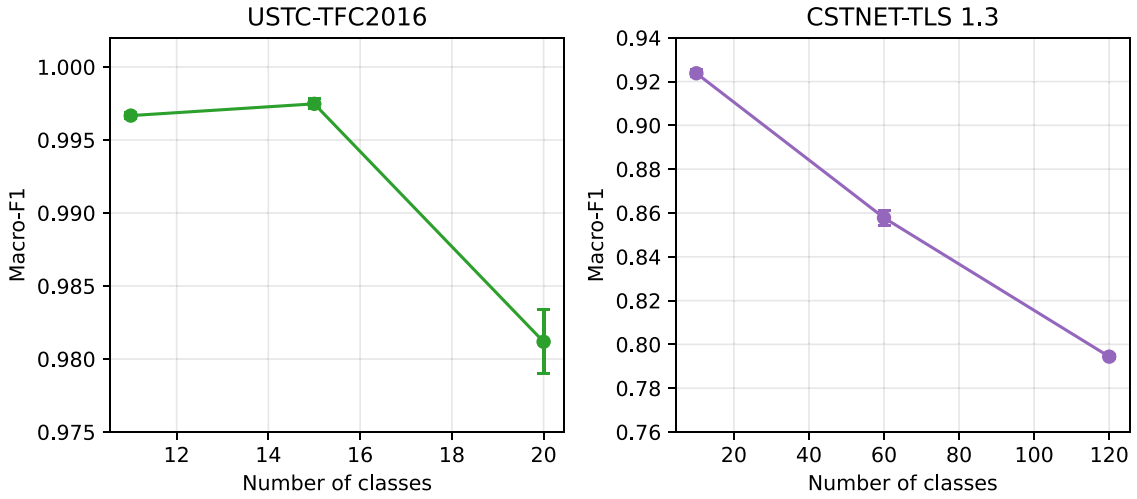

Figure 3: Consolidated adapter capacity on USTC-TFC2016 and CSTNET-TLS 1.3. USTC-TFC2016 remains near saturation up to 15 classes and remains high at 20 classes, whereas CSTNET-TLS 1.3 shows a clearer scale-dependent decline. Error bars indicate standard deviation over seeds 7, 21, and 42.

These results show that USTC-TFC2016 is comparatively easy for a Consolidated Adapter: even the 20-class row remains above 0.98 Macro-F1. In CSTNET-TLS 1.3, however, Macro-F1 decreases from 0.9238 at 10 classes to 0.8578 at 60 classes and 0.7944 at 120 classes. These results indicate that capacity conclusions are dataset-dependent, with CSTNET-TLS 1.3 providing a more demanding check than USTC-TFC2016.

Table 5: Consolidated adapter capacity on USTC-TFC2016 and CSTNET-TLS 1.3. Values are mean ± std over seeds 7, 21, and 42.

Dataset	Class ids	Classes	Accuracy	Macro-F1	Time (min)
USTC-TFC2016	0–10	11	0.9967 ± 0.0002	0.9967 ± 0.0002	28.1 ± 0.1
USTC-TFC2016	0–14	15	0.9977 ± 0.0003	0.9975 ± 0.0003	49.4 ± 6.1
USTC-TFC2016	0–19	20	0.9808 ± 0.0022	0.9812 ± 0.0022	29.8 ± 0.1
CSTNET-TLS 1.3	0–9	10	0.9275 ± 0.0018	0.9238 ± 0.0016	9.4 ± 0.1
CSTNET-TLS 1.3	0–59	60	0.8563 ± 0.0037	0.8578 ± 0.0035	118.4 ± 23.3
CSTNET-TLS 1.3	0–119	120	0.7925 ± 0.0012	0.7944 ± 0.0010	120.5 ± 1.0

Note: All rows follow the same training/evaluation protocol, class-mapping convention, seed policy, and metric definitions. Some rows reuse previously completed runs after protocol validation, while others were newly executed to complete the class-count grid.

4.2.4 Runtime and Inference Latency

The update-time and inference-time conclusions are different. Adapter Fusion preserves modular source adapters, but its inference path activates multiple adapters and therefore carries latency overhead. Fig. 4 and Table 6 report the core latency comparison. The core latency measurements in Table 6 were collected using the same CUDA-synchronized timing protocol as the adapter-count latency measurements in Table 7.

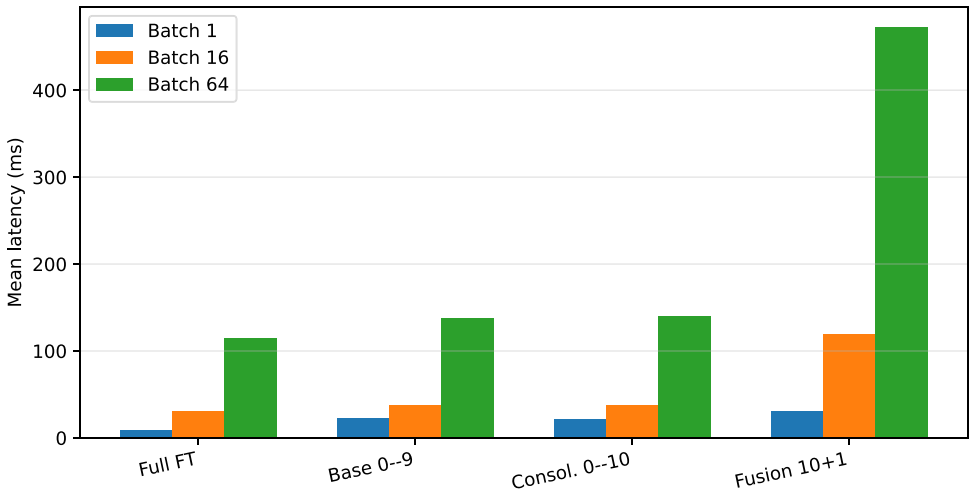

Figure 4: Inference latency comparison on USTC-TFC2016 10 + 1 checkpoints. Adapter fusion has higher inference latency than full fine-tuning and single-adapter alternatives across all measured batch sizes.

Table 6: Inference latency in milliseconds. Values are mean ± std over seeds 7, 21, and 42 with 200 measured iterations per seed.

Method	Batch 1	Batch 16	Batch 64
Full Fine-Tuning 0–10	9.026 ± 0.038	30.857 ± 0.334	114.940 ± 1.402
Base Adapter 0–9	22.719 ± 0.297	37.351 ± 0.262	137.018 ± 0.972
Consolidated Adapter 0–10	21.262 ± 0.044	37.962 ± 0.064	140.160 ± 0.173
Adapter Fusion 10 + 1	30.108 ± 0.040	119.273 ± 0.235	472.439 ± 1.724

Table 7: Adapter Fusion inference-latency scaling by fused adapter count. Values are mean ± std in milliseconds over seeds 7, 21, and 42.

# Adapters	Batch 1	Batch 16	Batch 64
2	30.207 ± 0.091	117.805 ± 1.808	474.223 ± 6.267
7	43.837 ± 0.333	137.079 ± 1.479	539.530 ± 1.145
8	46.645 ± 0.173	142.972 ± 3.572	565.882 ± 6.216
11	55.757 ± 0.765	154.998 ± 2.810	607.360 ± 1.853

To check whether this overhead scales with the number of fused adapters, we also measured Adapter Fusion latency for representative adapter counts 2, 7, 8, and 11 using the same CUDA-synchronized protocol. Table 7 shows a monotonic latency increase across all measured batch sizes.

Accordingly, Adapter Fusion should be interpreted as a modular update and source-adapter preservation strategy rather than an inference-latency optimization. When joint data are available and a single artifact is acceptable, Consolidated Adapter training can be more attractive in both accuracy and inference latency.

4.3 Discussion

The experimental results provide the following implications.

1. **Operating-regime framing:** Adapter Fusion is best understood as a modular continual-adaptation strategy, not as a universally superior replacement for Full Fine-Tuning or Consolidated Adapter training.
2. **Base vs. consolidated adapter:** The Base Adapter is a source module trained on classes 0–9. The direct single-adapter competitor for the 0–10 task is the Consolidated Adapter, which performs strongly in this setting.
3. **Fusion boundary:** Under the controlled multi-seed protocol, Fusion Boundary performance shows limited degradation through 11 adapters rather than abrupt collapse. The main transition appears between 7 and 8 fused adapters.
4. **Mechanism diagnostic:** The count 7–8 transition is not a one-label or few-label prediction-concentration collapse; prediction-label diversity remains high, while deeper fusion-weight and leave-one-adapter-out analyses require additional diagnostic tooling.
5. **Modern-dataset capacity:** CSTNET-TLS 1.3 shows a clearer class-scale decline than USTC-TFC2016, indicating that USTC-TFC2016-only evidence is insufficient for broad capacity claims.
6. **Latency limitation:** Adapter Fusion has higher inference latency than Full Fine-Tuning and single-adapter alternatives in the measured 10 + 1 setting, and the overhead increases with the number of fused adapters. Deployment use should therefore weigh modular update benefits against inference overhead.
7. **Metric limitation:** Forward Transfer is not reported because the single-step USTC-TFC2016 10 + 1 scenario lacks a meaningful pre-update future-task test. Reporting an FWT value under this protocol would be misleading.

5 Conclusion

This study presented the ET-BERT Adapter Fusion framework as an operating-regime analysis of modular continual adaptation for encrypted traffic classification. The core method keeps the ET-BERT backbone frozen, trains source adapters for base and newly emerging classes, and composes them through Adapter Fusion. The experiments show that this modular design can retain high USTC-TFC2016 performance in

the core USTC-TFC2016 10 + 1 setting, but they also show that Full Fine-Tuning and Consolidated Adapter training remain strong references when joint or consolidated access to the expanded class set is possible.

The Fusion Boundary analysis further shows that Adapter Fusion did not exhibit abrupt collapse through 11 adapters. Instead, it exhibits limited degradation, with the main transition around the 7–8 adapter range and increasing training time as more adapters are composed. Consolidated Adapter capacity experiments indicate that USTC-TFC2016 remains comparatively easy for a single adapter, whereas CSTNET-TLS 1.3 exposes clearer class-scale difficulty. The latency measurements also show that Adapter Fusion is not inference-time faster than Full Fine-Tuning or single-adapter alternatives in the measured 10 + 1 setting.

Accordingly, the practical contribution of the framework is modularity rather than unconditional superiority: Adapter Fusion is useful when previously trained adapters must be preserved, isolated, and composed without full retraining, while Consolidated Adapter training may be preferable when old and new data can be jointly used and a single low-latency artifact is desired. Future work will investigate selective or hierarchical fusion, stronger mechanism diagnostics, and deployment policies that decide when to keep composing adapters and when to consolidate them.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2023-00235509, Development of Security Monitoring Technology Based on Network Behavior against Encrypted Cyber Threats in ICT Convergence Environment).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Minsu Kim, Daeho Choi and Younghyo Cho; methodology, Minsu Kim; software, Minsu Kim; validation, Minsu Kim, Daeho Choi and Younghyo Cho; formal analysis, Minsu Kim; investigation, Minsu Kim; resources, Jun Lee, Changhoon Lee and Kiwook Sohn; data curation, Minsu Kim; writing—original draft preparation, Minsu Kim; writing—review and editing, Daeho Choi, Younghyo Cho and Kiwook Sohn; visualization, Minsu Kim; supervision, Yeog Kim, Jun Lee, Changhoon Lee and Kiwook Sohn; project administration, Yeog Kim, Jun Lee, Changhoon Lee and Kiwook Sohn; funding acquisition, Jun Lee, Changhoon Lee and Kiwook Sohn. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: USTC-TFC2016 is publicly available from its cited dataset source [11]. CSTNET-TLS 1.3 is released as anonymized data through the ET-BERT dataset source and is described in the ET-BERT benchmark context [2, 24]. Derived preprocessing artifacts, split manifests, and run summaries used in this study are available from the corresponding author upon reasonable request, subject to institutional and security constraints.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN). Piscataway, NJ, USA: IEEE; 2017. p. 712–7.
2. Lin X, Xiong G, Gou G, Li Z, Shi J, Yu J. ET-BERT: a contextualized datagram representation with pre-training transformers for encrypted traffic classification. In: Proceedings of the ACM Web Conference 2022. New York, NY, USA: ACM; 2022. p. 1975–89.
3. De Lange M, Aljundi R, Masana M, Parisot S, Jia X, Leonardis A, et al. A continual learning survey: defying forgetting in classification tasks. IEEE Trans Pattern Anal Mach Intell. 2022;44(7):3366–85.

4. Masana M, Liu X, Twardowski B, Menta M, Bagdanov AD, Van de Weijer J. Class-incremental learning: survey and performance evaluation on image classification. IEEE Trans Pattern Anal Mach Intell. 2023;45(5):5513–33.
5. Verma T, Jin L, Zhou J, Huang J, Tan M, Choong BCM, et al. Privacy-preserving continual learning methods for medical image classification: a comparative analysis. Front Med. 2023;10:1227515. doi:10.3389/fmed.2023.1227515.
6. Houlsby N, Giurgiu A, Jastrzebski S, Morrone B, de Laroussilhe Q, Gesmundo A, et al. Parameter-efficient transfer learning for NLP. In: Proceedings of the 36th International Conference on Machine Learning. Cambridge, MA, USA: PMLR; 2019. p. 2790–9.
7. Pfeiffer J, Kamath A, Rücklé A, Cho K, Gurevych I. AdapterFusion: non-destructive task composition for transfer learning. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. Stroudsburg, PA, USA: ACL; 2021. p. 487–503.
8. Kim M, Choi D, Cho Y, Kim Y, Song J, Lee C, et al. ET-BERT with adapter fusion: time-efficient continual learning framework for encrypted traffic classification. In: The 9th International Conference on Mobile Internet Security (MobiSec 2025). Sapporo, Japan; 2025 Dec 16–18.
9. Lashkari AH, Draper-Gil G, Mamun MSI, Ghorbani AA. Characterization of tor traffic using time-based features. In: Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP). Setúbal, Portugal: SciTePress; 2017. p. 253–62.
10. Draper-Gil G, Lashkari AH, Mamun MSI, Ghorbani AA. Characterization of encrypted and VPN traffic using time-related features. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP). Setúbal, Portugal: SciTePress; 2016. p. 407–14.
11. University of Science and Technology of China. USTC-TFC2016 dataset [Dataset]. [cited 2026 Apr 13]. Available from: https://www.scidb.cn/en/detail?dataSetId=58b9d3c2.
12. Huang H, Zhang X, Lu Y, Li Z, Zhou S. BSTFNet: an encrypted malicious traffic classification method integrating global semantic and spatiotemporal features. Comput Mater Contin. 2024;78(3):3929–51. doi:10.32604/cmc.2024.047918.
13. Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, et al. Overcoming catastrophic forgetting in neural networks. Proc Natl Acad Sci USA. 2017;114(13):3521–6. doi:10.1073/pnas.1611835114.
14. Li Z, Hoiem D. Learning without forgetting. IEEE Trans Pattern Anal Mach Intell. 2018;40(12):2935–47. doi:10.1109/tpami.2017.2773081.
15. Rebuffi SA, Kolesnikov A, Sperl G, Lampert CH. iCaRL: incremental classifier and representation learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE; 2017. p. 2001–10.
16. Pfeiffer J, Vulić I, Gurevych I, Ruder S. MAD-X: an adapter-based framework for multi-task cross-lingual transfer. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Stroudsburg, PA, USA: ACL; 2020. p. 7654–73.
17. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al. LoRA: low-rank adaptation of large language models. arXiv:2106.09685. 2021.
18. Pfeiffer J, Rücklé A, Poth C, Kamath A, Vulić I, Ruder S, et al. AdapterHub: a framework for adapting transformers. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Stroudsburg, PA, USA: ACL; 2020. p. 46–54.
19. Li XL, Liang P. Prefix-tuning: optimizing continuous prompts for generation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. Stroudsburg, PA, USA: ACL; 2021. p. 4582–97.
20. Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA, USA: ACL; 2021. p. 3045–59.
21. Ding N, Qin Y, Yang G, Wei F, Yang Z, Su Y, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. Nat Mach Intell. 2023;5(3):220–35. doi:10.1038/s42256-023-00626-4.

22. Karimi Mahabadi R, Henderson J, Ruder S. Compacter: efficient low-rank hypercomplex adapter layers. In: Advances in neural information processing systems. Red Hook, NY, USA: Curran Associates, Inc.; 2021. Vol. 34, p. 1022–35.
23. Rücklé A, Geigle G, Glockner M, Beck T, Pfeiffer J, Reimers N, et al. AdapterDrop: on the efficiency of adapters in transformers. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA, USA: ACL; 2021. p. 7930–46.
24. Linwhitehat ET-BERT Project. CSTNET-TLS 1.3 dataset [Dataset]. 2022 [cited 2026 May 21]. Available from: https://github.com/linwhitehat/ET-BERT/tree/main/datasets/CSTNET-TLS%201.3.