NIFT: Noise-Invariant Fine-Tuning for Privacy-Preserving LLMs

Anonymous ACL submission

Abstract

Privacy-preserving fine-tuning of large language models is appealing for many important applications, but remains challenging in practice. Adding differentially private (DP) noise to input representations can protect private data, but this noise is amplified by deep Transformer layers and sharply degrades performance. Prior work has primarily addressed this problem through noise-correction mechanisms that attempt to recover useful representations after amplification has already occurred, but these approaches still leave a substantial performance gap compared with the corresponding no-noise setting. In this paper, we propose a two-stage training framework that instead prepares the backbone to be less sensitive to the agreed DP noise distribution before downstream task fine-tuning. First, we train a LoRA adapter with a label-free pairwise consistency objective over multiple noisy embedding copies, encouraging the backbone's hidden states to remain stable under DP perturbations. Second, we fine-tune a lightweight client-side ladder side network that fuses local features with selected hidden states returned by the robust server-side backbone. This separation of noise-invariance preparation from task adaptation reduces the burden on the client model, improves robustness under privacy-preserving noise. Our experiments show that NIFT outperforms the state-of-the-art HiddenEcho across multiple datasets and privacy budgets.

1 Introduction

Privacy-preserving fine-tuning of large language models (LLMs) has become increasingly important for NLP systems that must adapt to user- or organization-specific data. The most straightforward way to achieve this goal is to fine-tune the model entirely on local hardware, so that sensitive user or organizational data never leaves the private environment. However, fully fine-tuning a multi-billion-parameter model locally is often impractical because private hardware may have limited memory, computation, and deployment capacity (Lin et al., 2022; Dettmers et al., 2023).

This resource constraint motivates split learning (Vepakomma et al., 2018; Thapa et al., 2022), which partitions the model at a chosen cut layer and distributes computation between local hardware and a cloud provider while keeping raw private data within the local environment (Ma et al., 2025; Li et al., 2025; Shi et al., 2022). This design reduces direct exposure of user data, but it does not eliminate privacy risk; instead, it shifts the attack surface from raw inputs to intermediate representations. A curious server may still recover sensitive inputs through embedding inversion or gradient leakage attacks (Zhu et al., 2019; Erdogan et al., 2022; Jin et al., 2021). A common defense is to add calibrated noise to the embeddings before they leave the private environment (Dwork and Roth, 2014; Qu et al., 2021; Shen et al., 2023). However, this noise does not remain benign as it propagates through the backbone: each Transformer block transforms and re-normalizes the perturbed embedding, causing deviations in deeper hidden states to grow substantially beyond the initial perturbation. As a result, the downstream classifier receives noisy, distorted representations, leading to sharp accuracy degradation under practical privacy budgets.

Prior work has addressed this accuracy degradation by treating error amplification as a downstream artifact to be repaired. Split-and-Denoise (Mai et al., 2024) introduced a client-side denoiser for noisy embeddings, while HiddenEcho (Li et al., 2026) extended this repair-based strategy to per-layer corrections and currently represents the strongest approach in this direction. Although these methods can partially recover accuracy, their corrective effect is inherently limited: once the server-side backbone has amplified a small privacy-preserving perturbation into a much larger repre-

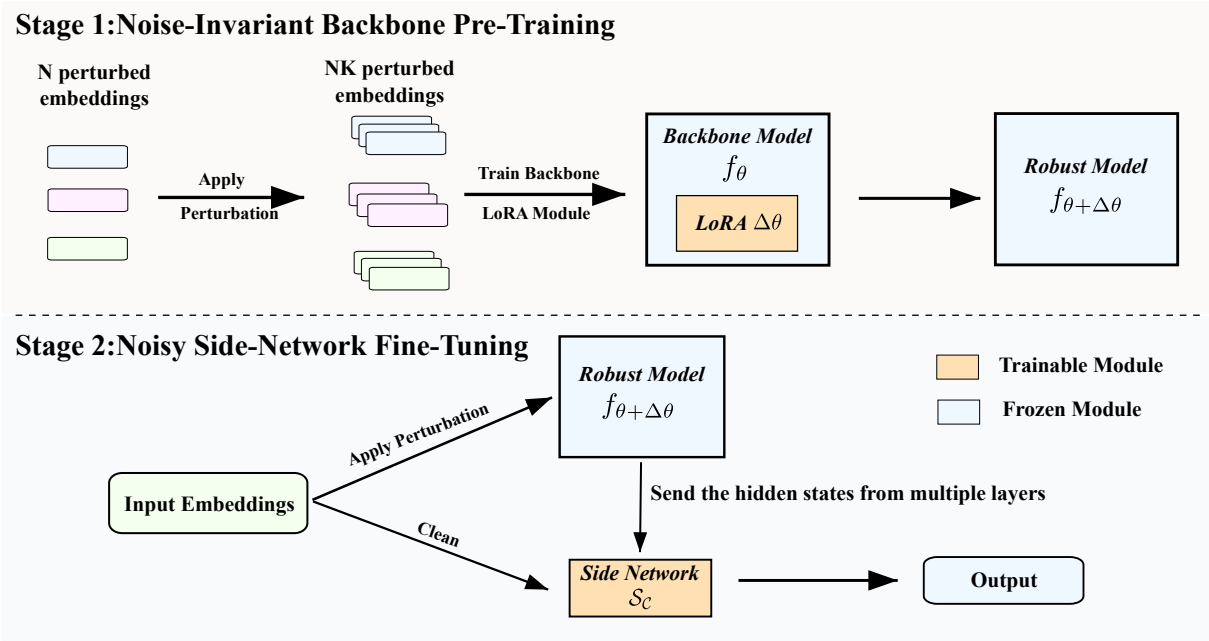

Figure 1: Overview of the proposed two-stage framework. *Stage 1 (top): noise-invariant backbone pre-training.* The client sends N DP-perturbed embeddings to the server; the server then perturbs each received embedding K times, yielding NK noisy copies in total. These copies are passed through the frozen backbone with a low-rank (LoRA) adapter and used to train the robust model $f_{\theta+\Delta\theta}$. *Stage 2 (bottom): noisy side-network fine-tuning.* The client embeds each input, sends a DP-perturbed copy through the frozen robust backbone on the server (which returns selected hidden states to the device), and trains a lightweight client-side side network that fuses these returned hidden states with a clean local embedding. Different clients can each train their own LoRA adapter on top of the same shared backbone.

sentation shift, the client may not be able to reconstruct the lost clean representation, and the resulting model often remains substantially less accurate than the same model evaluated on unperturbed inputs.

To address this limitation, we take the opposite view: **although the backbone remains frozen during each user's task-specific fine-tuning, it can be prepared in advance to make its hidden states less sensitive to the DP noise injected by the client.** We propose NIFT (Noise-Invariant Fine-Tuning), a two-stage framework for privacy-preserving split fine-tuning of LLMs in local–server settings. In the first stage, we fit a low-rank adapter (LoRA) (Hu et al., 2022) to the backbone using a label-free pairwise consistency objective. For each input, the client sends a single DP-perturbed embedding to the server; the server then creates K noisy copies internally, passes them through the backbone, and minimizes the mean pairwise ℓ_2 distance between their hidden states at selected layers. This objective encourages the LoRA parameters to absorb noise-induced variability, so that the resulting backbone-adapter model produces hidden states that are approximately invariant to the agreed-upon noise distribution. Because the consistency objective does not require labels, each client can train its own LoRA adapter on top of the same shared frozen backbone, matched to its local input distribution and privacy budget. In the second stage, we fine-tune the model on the user's downstream task: a lightweight ladder side network (Sung et al., 2022) on the device fuses the returned backbone hidden states with its own local representation to predict the label. Figure 1 summarizes the two stages and the data flow between the client and server.

This two-stage design offers three practical advantages. First, it separates noise-invariance preparation from task-specific fine-tuning, addressing DP-noise amplification before any private labels are used. Second, by suppressing noise-induced variation in the backbone, it lets the ladder side network fuse stable hidden states with local features rather than reconstruct distorted representations, improving accuracy under the same privacy budget. Finally, the resulting stability enables more aggressive activation compression, reducing communication without sacrificing performance.

Contributions. In summary, our contributions are as follows:

- We propose NIFT, a novel two-stage local–server framework for privacy-preserving fine-tuning of LLMs. The first stage prepares a noise-invariant LoRA-augmented backbone, and the second stage performs task-specific split fine-tuning with a lightweight client-side ladder network.
- We develop a server-side noise-consistency objective that improves robustness without increasing client exposure: the client sends only one DP-perturbed embedding, while the server constructs K noisy copies internally to train stable hidden representations.
- We show that noise-invariant representations make privacy-preserving split fine-tuning more effective and communication-efficient, enabling aggressive activation compression while preserving accuracy.
- Experiments on five text classification datasets show that NIFT consistently outperforms HiddenEcho (Li et al., 2026), with especially large gains under stringent privacy budgets.

2 Related Work

Split learning and side-tuning for LLMs. Split learning was originally proposed as a privacy-preserving distributed training paradigm: the model is partitioned at a cut layer so that raw data remains on the client, while only intermediate activations are exchanged with the server (Vepakomma et al., 2018; Thapa et al., 2022). Recent LLM fine-tuning systems adapt this partitioned execution model to make large models usable when local devices cannot host or update the full backbone: SplitFrozen (Ma et al., 2025) freezes device-side layers and trains a server-side LoRA module, MobiLLM (Li et al., 2025) offloads back-propagation through quantized activation transfer, and Shi et al. (2022) reduces communication via low-rank decomposition of the cut-layer weights. Related systems address similar access and deployment constraints from different angles: Offsite-tuning (Xiao et al., 2023) lets users tune a compact emulator and adapter without accessing the full foundation model, while Petals (Borzunov et al., 2023) distributes Transformer blocks across peers for collaborative inference and parameter-efficient fine-tuning. Although these systems improve model accessibility, compute efficiency, and bandwidth usage, they generally do not address the privacy risks that remain in the transmitted activations and gradients.

Privacy in distributed fine-tuning. Cross-device traffic can leak sensitive information: gradients may reveal training inputs (Zhu et al., 2019; Geiping et al., 2020; Jin et al., 2021), and an honest-but-curious server may reconstruct private data from intermediate activations (Erdogan et al., 2022; Shu et al., 2025). A standard mitigation is to apply local differential privacy to embeddings or gradients (Dwork and Roth, 2014; Qu et al., 2021; Shen et al., 2023; Yu et al., 2022; Li et al., 2022), but the injected perturbation is repeatedly transformed by subsequent Transformer blocks, causing noise amplification and substantial utility loss. The closest prior work addresses this degradation through post-hoc repair: HiddenEcho (Li et al., 2026) returns per-layer hidden states for layer-wise client-side correction, extending the earlier Split-and-Denoise approach (Mai et al., 2024). These methods mitigate the effect of amplified noise after it appears, whereas our goal is to make the server-side representations less sensitive to the injected noise in the first place.

Consistency-based representation learning. The first stage of NIFT is related to consistency-based regularization, which encourages a model to produce similar representations or predictions under input perturbations (Miyato et al., 2019; Xie et al., 2020). Randomized smoothing makes a similar connection between Gaussian noise injection and robustness, including in differentially private settings (Cohen et al., 2019; Lecuyer et al., 2019). We adapt this principle to privacy-preserving split fine-tuning: instead of using perturbations as data augmentation for downstream classification, we prepare the frozen backbone to absorb client-injected DP noise before the resulting representations are passed to the side network.

3 Problem Setup and Threat Model

Parties and assets. A client device $\mathcal{C}$ owns a private labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ and seeks to fine-tune a large pre-trained model for its local task. A server $\mathcal{S}$ hosts a backbone model f_θ that maps an input embedding sequence to a sequence of hidden states in each layer. The client fine-tunes a lightweight local model. For each private data

sample (x_i, y_i), the client maps the input tokens x_i to clean embeddings, injects calibrated noise, and transmits only the perturbed embeddings to the server. The server processes the perturbed embeddings and returns a subset of hidden states, which the client uses to make predictions.

Noise mechanism. We adopt the local-DP embedding mechanism used by recent split-LLM systems (Qu et al., 2021; Shen et al., 2023; Mai et al., 2024; Li et al., 2026). Formally, the DP-perturbed embedding is constructed by adding an unbounded noise term ξ_t to the clipped embedding:

$$\tilde{e}_t = \Pi_{\|\cdot\|_2 \le c}(e_t) + \xi_t, \tag{1}$$

where $\Pi_{\|\cdot\|_2 \le c}$ clips each token embedding to the ℓ_2 ball of radius c: embeddings with norm at most c are left unchanged, while embeddings with larger norm are rescaled to have norm c. The noise term ξ_t is drawn independently according to the ℓ_2-Laplace mechanism, with density

$$p(\xi_t) = \frac{\eta^d}{C_d \Gamma(d) c^d} \exp\left(-\frac{\eta \|\xi_t\|_2}{c}\right), \tag{2}$$

where d is the embedding dimension, $C_d = 2\pi^{d/2}/\Gamma(d/2)$ is the surface area of a unit sphere in $\mathbb{R}^d$, and $\Gamma(\cdot)$ denotes the Gamma function.

Threat model. The server is honest-but-curious: it follows the protocol but may inspect and store any message it receives. The attack surface consists of two channels: (i) the embeddings e sent to the server in the forward pass; (ii) the gradients sent back from the client during back propagation. We aim to defend against both attacks.

4 Method

Our solution consists of two stages of training that share the same noise specification $(\mathcal{N}_\eta, c)$.

4.1 Stage 1: Noise-Invariant Backbone Pre-Training

On the server, at first, we train an adaptation $\Delta\theta$ on top of the frozen backbone f_θ so that $f_{\theta+\Delta\theta}$ produces hidden states that are invariant to the noise distribution $\mathcal{N}_\eta$. Specifically, $\Delta\theta = \{B_\ell A_\ell\}$ is a rank-r LoRA adapter applied to the query, key, value, and output projections of every Transformer block, following common practice (Hu et al., 2022). The backbone parameters θ remain fixed, and only $\Delta\theta$ is updated.

Server-side multi-sample perturbations. For each input x in a batch, the client computes the clean embedding $e = Ex$, applies the DP embedding mechanism in Eq. (1), and sends the resulting perturbed embedding $\tilde{e}_0$ to the server. To obtain multiple stochastic views, the server generates K noisy variants $\tilde{e}_1, \ldots, \tilde{e}_K$ by adding independently sampled perturbations to $\tilde{e}_0$ following the same ℓ_2-Laplace noise distribution.

Pairwise consistency objective. Let $h_k^{(\ell)} = f_\ell(\tilde{e}_k; \theta + \Delta\theta)$ denote the hidden state at backbone layer ℓ for the k-th noisy copy. We penalize the average pairwise ℓ_2 distance among the hidden states of the K noisy copies:

$$\mathcal{L}_{\mathrm{P}}^{(\ell)} = \frac{2}{K(K-1)} \sum_{1 \le i < j \le K} \left\| h_i^{(\ell)} - h_j^{(\ell)} \right\|_2. \tag{3}$$

For a set of selected layers $\mathcal{S}$, we define the overall consistency objective as the average loss across these layers:

$$\mathcal{L}_{\mathrm{P}} = \frac{1}{|\mathcal{S}|} \sum_{\ell \in \mathcal{S}} \mathcal{L}_{\mathrm{P}}^{(\ell)}. \tag{4}$$

Minimizing Eq. (3) encourages hidden states of all K noisy copies of the same input to coincide, thereby making the backbone less sensitive to input noise within the batch.

Anti-collapse regularization. Although Eq. (3) encourages invariance across noisy copies, it admits a degenerate solution in which the backbone maps all inputs to a constant representation. To avoid this failure, we monitor two diversity diagnostics during training: ℓ_2^{sample}, the mean pairwise distance between hidden states $h_k^{(\ell)}$ for different inputs under the same noisy sample, and ℓ_2^{token}, the mean pairwise distance between hidden states at different valid token positions of the same input. Formally, these diagnostics are defined as

$$\ell_2^{\text{sample}} = \frac{2}{|\mathcal{B}|(|\mathcal{B}|-1)} \sum_{i<j} \|h_i^{(\ell)} - h_j^{(\ell)}\|_2, \tag{5}$$

$$\ell_2^{\text{token}} = \frac{2}{T(T-1)} \sum_{t<t'} \|h_{k,t}^{(\ell)} - h_{k,t'}^{(\ell)}\|_2, \tag{6}$$

where $\mathcal{B}$ is the batch of inputs and T is the number of valid tokens. Representation collapse is indicated when either diagnostic approaches zero. To discourage collapse without relying on manual

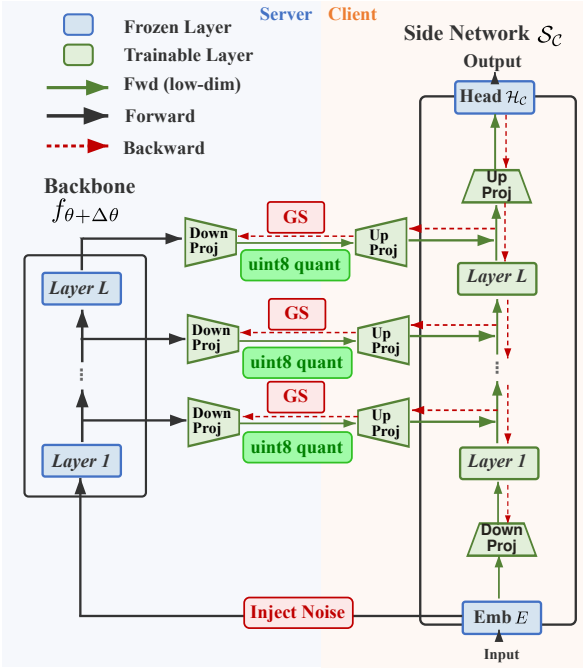

Figure 2: Stage 2 architecture in detail. The server (left) hosts the frozen, robust backbone $f_{\theta+\Delta\theta}$. At each selected layer, a trainable split bottleneck adapter $\mathcal{A}_{\mathcal{S}}$ places its server-side down-projection and client-side up-projection on opposite sides of the cut; the forward wire format is an 8-bit PACT-quantized tensor (`uint8 quant`), and the returned gradients are compressed by Top-k sparsification with error feedback (denoted `GS`). The client side (right) holds the frozen embedding E and classification head $\mathcal{H}_C$, together with the trainable side network $\mathcal{S}_C$, which fuses the bottleneck adapter outputs with its own clean embedding. The bottom-left red box marks the DP perturbation ξ added to the embeddings before they cross the cut.

early stopping, we record per-layer step-0 baselines $b_s^{(\ell)}$ and $b_t^{(\ell)}$ from a single no-gradient forward pass before training, normalize the diagnostics as $r_s^{(\ell)} = \ell_2^{\text{sample},(\ell)}/b_s^{(\ell)}$ and $r_t^{(\ell)} = \ell_2^{\text{token},(\ell)}/b_t^{(\ell)}$, and add a squared-hinge penalty that activates only when either ratio falls below a margin $C \in (0, 1]$:

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathcal{S}|} \sum_{\ell \in \mathcal{S}} \left[\phi(r_s^{(\ell)}) + \phi(r_t^{(\ell)})\right] \tag{7}$$

where $\phi(r) = \max(0, C - r)^2$.

The overall Stage 1 objective is $\mathcal{L}_{\text{P}} + \alpha\,\mathcal{L}_{\text{reg}}$, where $\alpha \geq 0$ controls the strength of the collapse penalty.

4.2 Stage 2: Noisy Side-Network Fine-Tuning

Stage 2 training involves three main components, illustrated in Figure 2:

- $f_{\theta+\Delta\theta}$: the frozen robust backbone on the server;
- $\mathcal{S}_C$: the trainable client-side side network, implemented as the ladder side network;
- $\mathcal{A}_{\mathcal{S}}$: the trainable per-layer bottleneck adapter, whose down-projection module is placed on the server side and whose up-projection module is placed on the client side.

Forward pass. The server first evaluates the frozen backbone on the perturbed embedding $\tilde{e}_0$ received from the client in Stage 1:

$$\left(h^{(1)}, h^{(2)}, \ldots, h^{(L)}\right) = f_{\theta+\Delta\theta}(\tilde{e}_0). \tag{8}$$

For each cut layer ℓ in a chosen subset $\mathcal{R} \subseteq \{1, \ldots, L\}$, the server applies the split bottleneck adapter described below and sends a compressed subset of hidden states back to the client. The client's ladder side network (Sung et al., 2022) reduces the embedding to a side-stream tensor and, layer by layer, fuses it with the returned bottleneck features through additive layer normalization followed by a reduced Transformer decoder block. The final pooled representation is fed to the classification head.

Split bottleneck adapter. The cut-layer adapter serves a dual purpose: it compresses the hidden states crossing the device-server boundary to reduce bandwidth, and provides a learnable interface that lets the client adapt to the server's representations. Our adapter design follows the bottleneck side-tuning approach of Quantized Side Tuning (QST) (Zhang et al., 2024), extended with PACT-style activation quantization (Choi et al., 2018) at the cut to further compress the on-the-wire payload from BF16 to uint8. For each cut layer ℓ, the adapter's two projections live on opposite sides of the cut:

$$\text{(server)}\quad u^{(\ell)} = \text{LN}\big(W_{\text{down}}^{(\ell)} h^{(\ell)} + b_{\text{down}}^{(\ell)}\big), \tag{9}$$

$$\text{(wire)}\quad \hat{u}^{(\ell)} = \mathcal{Q}_{\alpha^{(\ell)}}\big(u^{(\ell)}\big), \tag{10}$$

$$\text{(client)}\quad v^{(\ell)} = W_{\text{up}}^{(\ell)}\, \mathcal{Q}_{\alpha^{(\ell)}}^{-1}\big(\hat{u}^{(\ell)}\big), \tag{11}$$

where r is the reduction factor, $u^{(\ell)}, \hat{u}^{(\ell)} \in \mathbb{R}^{T \times d/r}$, $v^{(\ell)} \in \mathbb{R}^{T \times d_s}$ with $d_s = d/r$ the side-network dimension, and $\mathcal{Q}_\alpha(\cdot)$ is a PACT-style activation quantizer with a per-layer learnable clipping bound $\alpha^{(\ell)}$. Concretely, $\mathcal{Q}_\alpha$ clips its input to $[0, \alpha]$, divides by $\alpha/255$, and rounds to the nearest integer.

The client then rescales by multiplying with $\alpha/255$ to recover an approximation of the original activation. Compared with sending the raw $T \times d$ hidden state in BF16, the transmission overhead is reduced by a factor of $r \times 2$ per layer.

Backward pass and gradient sparsification. During back-propagation, the client-side loss sends gradients through the device-side up-projection, after which the corresponding gradient signal must cross the cut to update the server-side down-projection. We reduce this cross-device traffic with Top-k sparsification and error feedback (Lin et al., 2018; Stich et al., 2018), transmitting only a small fraction $\rho \in (0, 1]$ of the elements of the gradient tensor while accumulating the omitted elements for later updates. Beyond bandwidth savings, this compression also limits the gradient information available to a potential adversary. Gradient reconstruction attacks rely on access to the elements of the gradient tensor (Zhu et al., 2019); prior evidence shows that pruning more than 80% of them can already make recovered inputs unrecognizable, whereas our practical settings keep $\rho = 0.05$ of them.

Loss function. Let $\hat{y} = \mathcal{H}_{\mathcal{C}}(\mathcal{S}_{\mathcal{C}}(\{v^{(\ell)}\}_{\ell \in \mathcal{R}}))$ denote the prediction on the client. Stage 2 minimizes the supervised task loss over the parameters of $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{A}_{\mathcal{S}}$:

$$\mathcal{L}_{\text{ft}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\mathcal{L}_{\text{task}}(\hat{y}, y)\big] \tag{12}$$

Since this work targets downstream classification tasks, we instantiate $\mathcal{L}_{\text{task}}$ as the cross-entropy loss.

4.3 Privacy Interpretation

The end-to-end protection provided by our framework stems from three complementary sources. First, the client-side DP mechanism provides the standard local-DP guarantee for information transmitted from the device during the forward pass. Second, stage 1 does not weaken this guarantee; rather, it reshapes the backbone so that downstream representations become less sensitive to the injected noise. Third, the back-propagated gradients are Top-k sparsified, which not only reduces communication overhead but also places the gradients in a regime where gradient-inversion attacks are empirically unable to recover the input (Zhu et al., 2019).

4.4 Communication Overhead

We assume BF16 (2-byte) precision for all transmitted tensors, except for PACT-quantized cut activations, which are represented as uint8 (1-byte). Let N denote the number of training samples, T_{ft} the number of fine-tuning epochs, S the sequence length, L the number of transmitted layers, b the PACT bottleneck dimension used by NIFT, d the reduced hidden dimension used by HiddenEcho, and ρ the Top-k gradient keep ratio. The data transfer costs for hidden states and gradients between server and client, measured in bytes, are

$$\mathcal{C}_{\text{NIFT}} = bSLNT_{\text{ft}}(1 + 2\rho) \tag{13}$$

for NIFT, and

$$\mathcal{C}_{\text{HE}} = 4dSLNT_{\text{ft}} \tag{14}$$

for HiddenEcho. In our experiments, we set $b = 8$ and $\rho = 0.05$ for our method, while HiddenEcho uses $d = 96$. Under these settings, NIFT incurs only approximately 2.3% of HiddenEcho's data communication cost while achieving better accuracy.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate on five text classification datasets that span sentence-pair, single-sentence, multi-class, and domain-specific tasks: MRPC, RTE, and WNLI from the GLUE benchmark (Wang et al., 2018), the five-class BBC News corpus from SetFit, and the three-class Financial Phrasebank. The first three datasets (MRPC, BBC News, Financial Phrasebank) are those evaluated in the original HiddenEcho paper (Li et al., 2026). The other two (RTE, WNLI) extend the evaluation to smaller and harder sentence-pair tasks where DP noise amplification is most pronounced.

Baselines. Our main comparison is with HiddenEcho (Li et al., 2026), the state-of-the-art in the repair-based defense class. We do not include Split-and-Denoise (Mai et al., 2024) as a separate baseline because HiddenEcho already outperforms it across all reported settings.

Backbone and noise mechanism. All methods use a frozen Qwen2-1.5B (Yang et al., 2024) backbone on the server. The DP noise mechanism follows the DP formulation used by HiddenEcho: we apply the DP embedding mechanism (Eq. (1)) with the same privacy budgets

Dataset	Method	Privacy budget η			
		100	1000	5000	6000
MRPC	HiddenEcho	0.660	0.655	0.666	0.668
	Ours	**0.683**	**0.688**	**0.704**	**0.675**
BBC	HiddenEcho	0.732	0.747	0.805	0.951
	Ours	**0.993**	**0.992**	**0.995**	**0.991**
Financial	HiddenEcho	0.857	0.855	0.860	0.866
	Ours	**0.886**	**0.885**	**0.881**	**0.886**
RTE	HiddenEcho	**0.607**	0.562	0.579	0.598
	Ours	0.604	**0.596**	**0.617**	**0.636**
WNLI	HiddenEcho	0.675	0.781	0.647	0.663
	Ours	**0.765**	**0.799**	**0.766**	**0.737**

Table 1: Validation AUC: our framework vs HiddenEcho. Larger η = less noise.

Dataset	Method	Privacy budget η			
		100	1000	5000	6000
MRPC	Stage 2 only	0.657	0.681	0.684	0.658
	Stage 1 + 2	**0.683**	**0.688**	**0.704**	**0.675**
RTE	Stage 2 only	0.602	0.576	0.570	0.563
	Stage 1 + 2	**0.604**	**0.596**	**0.617**	**0.636**
WNLI	Stage 2 only	0.693	0.667	0.748	**0.760**
	Stage 1 + 2	**0.765**	**0.799**	**0.766**	0.737
BBC News	Stage 2 only	**0.993**	0.991	0.993	**0.993**
	Stage 1 + 2	**0.993**	**0.992**	**0.995**	0.991
Financial	Stage 2 only	**0.890**	**0.901**	**0.892**	**0.896**
	Stage 1 + 2	0.886	0.885	0.881	0.886

Table 2: Stage 1 ablation across all five datasets. *Stage 2 only* replaces the LoRA-augmented backbone with the vanilla Qwen2-1.5B backbone; all other components are identical to *Stage 1 + 2*.

$\eta \in \{100, 1000, 5000, 6000\}$ as HiddenEcho (Li et al., 2026) (larger η corresponds to less noise). The lowest value $\eta = 100$ corresponds to the high-noise regime in which backbone amplification is most severe.

Stage 1 and Stage 2 hyperparameters. For Stage 1, we apply a rank-16 LoRA adapter to the query, key, value, and output projections of every Transformer block, draw $K = 16$ noisy copies per input, and fix the collapse weight to $\alpha = 1.0$. For Stage 2, we set the side-network reduction factor to $r = 16$, the gradient keep ratio to $\rho = 0.05$, the batch size to 64, and the learning rate to 4×10^{-4}. More evaluation details and hyperparameter selection procedures are described in Appendix A.

5.2 Comparison with HiddenEcho

Table 1 reports validation AUC for NIFT against HiddenEcho across all five datasets and four privacy budgets.

NIFT outperforms HiddenEcho on 19 of the 20 (dataset, η) test cases. Most gains fall in the $+0.02$ to $+0.04$ AUC range, with the largest advantages under high noise: $+0.26$ AUC on BBC News at $\eta = 100$, where HiddenEcho's AUC collapses to 0.73, and $+0.119$ AUC on WNLI at $\eta = 5000$. The single exception is RTE at $\eta = 100$, where HiddenEcho leads by a negligible 0.003 AUC.

5.3 Effect of Noise-Invariance Pre-Training

To isolate the contribution of noise-invariance pre-training, we compare our full two-stage framework against a Stage 2–only variant, in which the side network is fine-tuned directly with the original Qwen2-1.5B backbone, without the LoRA adapter trained in Stage 1.

Table 2 reports this Stage 1 ablation across all five datasets. On MRPC, Stage 1 improves all four settings, with gains ranging from $+0.007$ to $+0.026$ AUC. On RTE, Stage 1 also improves all four, with the largest gain ($+0.073$ AUC) at $\eta = 6000$. On WNLI, Stage 1 improves three of the four settings, with the strongest gain ($+0.132$ AUC) at $\eta = 1000$.

On BBC News, both variants reach AUC ≈ 0.99 across all η, leaving no headroom for Stage 1 to contribute since the task is already saturated near the upper bound of 1.0. On Financial Phrasebank, Stage 1 introduces a small regression of about 0.01 AUC. Because Financial Phrasebank is a domain-specific corpus whose sentence-level judgments depend not only on syntax and lexical meaning but also on financial-domain knowledge encoded in the backbone, we tentatively attribute this regression to Stage 1's pairwise consistency objective partially eroding such knowledge.

Notably, even on these two datasets, the Stage-2-only variant alone already substantially outperforms HiddenEcho: it achieves ≥ 0.99 AUC on BBC News versus HiddenEcho's 0.73–0.95, and 0.88–0.90 on Financial Phrasebank versus HiddenEcho's 0.86. This indicates that NIFT's split architecture (side network, bottleneck adapter, and PACT quantization) is itself sufficient on these datasets, while Stage 1 provides the decisive advantage on the MRPC, RTE, and WNLI tasks.

η	$C = 0.55$	$C = 0.65$	$C = 0.75$
100	0.695	0.696	**0.696**
1000	0.667	0.688	**0.697**
5000	0.674	**0.704**	0.660
6000	0.697	**0.703**	0.676
mean	0.683	**0.698**	0.682

Table 3: NIFT validation AUC on MRPC at $C \in \{0.55, 0.65, 0.75\}$. $C = 0.65$ achieves the highest mean AUC across η.

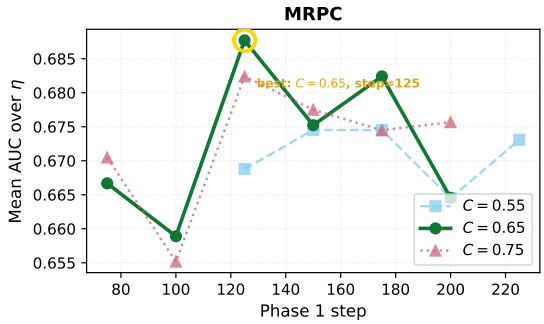

Figure 3: Mean AUC over η versus Stage 1 step on MRPC, for each value of C. The gold circle marks the global optimum at $C = 0.65$, step $= 125$.

Tuning the collapse margin. To understand how the collapse margin C in the anti-collapse regularizer (Eq. (7)) affects NIFT's performance, we sweep $C \in \{0.55,\ 0.65,\ 0.75\}$ on MRPC across all four privacy budgets. The theoretical reference $C = 0.75$ was derived from clean-baseline diagnostics on MRPC, but our empirical sweep finds $C = 0.65$ to be the strongest choice on average.

Table 3 reports the best AUC at each (η, C) combination, and Figure 3 plots the mean AUC over η against Stage 1 step. $C = 0.65$ achieves the highest mean AUC (0.698), outperforming the theoretical $C = 0.75$ by $+0.016$ AUC and $C = 0.55$ by $+0.015$ AUC. The corresponding sweeps on BBC News and Financial Phrasebank show all three C values within ± 0.002 AUC (Appendix B). We therefore adopt $C = 0.65$ as the default for the main experiments.

5.4 No-Privacy Reference

The no-privacy reference is the side network trained on clean (unperturbed) embeddings with the backbone path disabled, i.e., only the local stream feeds the classifier. Because no noise is added and no backbone signal is involved, the value is independent of the privacy budget η. Table 4 reports the AUC for each of the five datasets we evaluate. This

Dataset	AUC
MRPC	0.651
BBC News	0.990
Financial Phrasebank	0.898
RTE	0.560
WNLI	0.525

Table 4: No-privacy reference: side network trained on clean embeddings with the backbone path disabled. Values are η-independent.

baseline serves as an upper bound on what the side-network architecture alone can achieve. NIFT exceeds this baseline on 4 of the 5 datasets, with Financial Phrasebank the sole exception (consistent with the Stage 1 domain-knowledge erosion discussed in Section 5.3).

5.5 Data Communication Overhead

To demonstrate the communication advantage of NIFT over HiddenEcho, we measure the actual client-server traffic for hidden states and gradients. For 10 epochs of fine-tuning on MRPC (3700 samples), NIFT incurs 119 MB of data communication, while HiddenEcho requires 5.08 GB—a 43× reduction. This ratio is dataset-independent. Absolute costs scale linearly with the number of training samples. In practical deployment over a typical mobile uplink (∼10 Mbps), HiddenEcho's 5.08 GB transfer per MRPC fine-tuning run would require approximately 70 minutes of network time; NIFT's 119 MB requires ∼95 seconds. This gap becomes more critical as fine-tuning scales to larger datasets or more frequent updates.

6 Conclusion

We presented NIFT, a two-stage local–server framework for privacy-persevering fine-tuning of LLMs. Its key novelty is the separation of noise-invariance preparation from task adaptation: a label-free pairwise consistency objective stabilizes backbone hidden states under multiple noisy embedding views, while a lightweight client-side ladder side network adapts the robust backbone to downstream tasks. Our experiments show that NIFT improves both accuracy and communication efficiency, yielding a stronger privacy–accuracy–communication trade-off for split LLM fine-tuning.

Limitations

NIFT exhibits two main limitations, each pointing to a direction for future work. First, Stage 1's pairwise consistency objective may erode domain-specific knowledge encoded in the backbone, as observed on Financial Phrasebank (Section 5.3). A systematic study of when and why this erosion occurs—and how to preserve task-relevant knowledge during pre-training—is left to future work. Second, NIFT requires per-dataset hyperparameter tuning, primarily the collapse margin C and the Stage 1 training step, which is a deployment overhead that one-size-fits-all baselines do not incur. Automating this selection is another direction we leave to future work.

References

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Maksim Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. 2023. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*.

Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, pages 1310–1320.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407.

Ege Erdogan, Alptekin Kupcu, and A. Ercument Cicek. 2022. UnSplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. *Proceedings on Privacy Enhancing Technologies*.

Jonas Geiping, Hartmut Bauermeister, Hannah Droge, and Michael Moeller. 2020. Inverting gradients – how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Xiao Jin, Pin-Yu Chen, Chia-Yi Hsu, Chia-Mu Yu, and Tianyi Chen. 2021. CAFE: Catastrophic data leakage in vertical federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (SP)*, pages 656–672.

Liang Li, Xingke Yang, Wen Wu, Hao Wang, Tomoaki Ohtsuki, Xin Fu, Miao Pan, and Xuemin Shen. 2025. MobiLLM: Enabling LLM fine-tuning on the mobile device via server assisted side tuning. *arXiv preprint arXiv:2502.20421*.

Wenhao Li, Kunhao Li, and Lei Yang. 2026. HiddenEcho: Mitigating noise amplification in differentially private LLMs with hidden-state correction. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2022. Large language models can be strong differentially private learners. In *International Conference on Learning Representations (ICLR)*.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. 2022. On-device training under 256KB memory. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations (ICLR)*.

Jian Ma, Xinchen Lyu, Jun Jiang, Qimei Cui, Haipeng Yao, and Xiaofeng Tao. 2025. SplitFrozen: Split learning with device-side model frozen for fine-tuning LLM on heterogeneous resource-constrained devices. *arXiv preprint arXiv:2503.18986*.

Peihua Mai, Ran Yan, Zhe Huang, Youjia Yang, and Yan Pang. 2024. Split-and-Denoise: Protect large language model inference with local differential privacy. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993.

Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural language understanding with privacy-preserving BERT. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1488–1497.

Xicong Shen, Yang Liu, Huiqi Liu, Jue Hong, Bing Duan, Zirui Huang, Yunlong Mao, Ye Wu, and Di Wu. 2023. A split-and-privatize framework for large language model fine-tuning. *arXiv preprint arXiv:2312.15603*.

Shaohuai Shi, Qing Yang, Yang Xiang, Shuhan Qi, and Xuan Wang. 2022. An efficient split fine-tuning framework for edge and cloud collaborative learning. *arXiv preprint arXiv:2211.16703*.

Yunmeng Shu, Shaofeng Li, Tian Dong, Yan Meng, and Haojin Zhu. 2025. Model inversion in split learning for personalized LLMs: New insights from information bottleneck theory. *arXiv preprint arXiv:2501.05965*. Proposes the RevertLM two-stage inversion attack against split-LLM deployments.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. 2018. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. SplitFed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8485–8493.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems (NeurIPS)*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2022. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR)*.

Zhengxin Zhang, Dan Zhao, Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Qing Li, Yong Jiang, and Zhihao Jia. 2024. Quantized side tuning: Fast and memory-efficient tuning of quantized large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–17.

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Evaluation Setups

The collapse margin C is selected by a sweep over $\{0.55, 0.65, 0.75\}$ on validation AUC: $C = 0.65$ maximizes mean AUC on MRPC and is tied with the best alternative on BBC News and Financial Phrasebank (within ± 0.002 AUC; see Appendix B). We therefore adopt $C = 0.65$ uniformly across these three datasets. For RTE and WNLI we use $C = 0.55$, the value that emerged from initial exploratory runs on these smaller GLUE tasks (a full C sweep is left to future work; see the limitations discussion). For MRPC, BBC News, and Financial Phrasebank, where the step sweep covers a broader range, we select a single Stage 1 step per dataset that maximizes mean validation AUC across the four privacy budgets, yielding 125 for MRPC and 200 for BBC News and Financial Phrasebank. For RTE and WNLI, where the step sweep was restricted to a narrower range, we report the best step per $(\text{dataset}, \eta)$ cell from the available sweep, consistent with standard per-task hyperparameter selection on GLUE (Wang et al., 2018); sensitivity to step choice is reported in Appendix C.

We use the Hugging Face Trainer for Stage 2 fine-tuning and report the best validation AUC across epochs, matching HiddenEcho's reporting convention (Li et al., 2026). For HiddenEcho on MRPC, BBC News, and Financial Phrasebank, we use the numbers published in Table 1 of Li et al. (2026) directly; for RTE and WNLI, which are not evaluated in the original paper, we re-ran HiddenEcho with the authors' released code on our hardware.

B Selecting the Collapse Margin C

The collapse margin C controls when the anti-collapse hinge (Eq. 7) fires: smaller C is more permissive, larger C is stricter. We initially derived a theoretical value $C = 0.75$ from the ratio of clean to noisy-step-0 pairwise diversity on MRPC; however, our empirical sweep across $C \in \{0.55, 0.65, 0.75\}$ on the three main datasets finds $C = 0.65$ to be the best or tied-best choice everywhere.

Figure 4 reports mean validation AUC across the four privacy budgets at each Stage 1 step, with separate lines for each C value. On MRPC, the mean AUC of $C = 0.65$ leads $C = 0.55$ by $+0.015$ AUC (and $C = 0.75$ by $+0.016$). On BBC News, all three C values are within ± 0.002 AUC because the task is already saturated near the AUC ceiling of 1.0. On Financial Phrasebank, the mean AUC across the three C values varies by less than ± 0.002. We therefore adopt $C = 0.65$ uniformly across the three datasets.

C Sensitivity to Stage 1 Step

Figure 5 reports validation AUC against Stage 1 step for each $(\text{dataset}, \eta)$ cell at the selected $C = 0.65$. Sensitivity varies by dataset. On BBC News, AUC is essentially invariant to step (range < 0.005 across the swept values), consistent with the saturation observed above. On MRPC and Financial Phrasebank, AUC varies by up to 0.04 across step values, motivating the per-dataset step selection used in the main results.

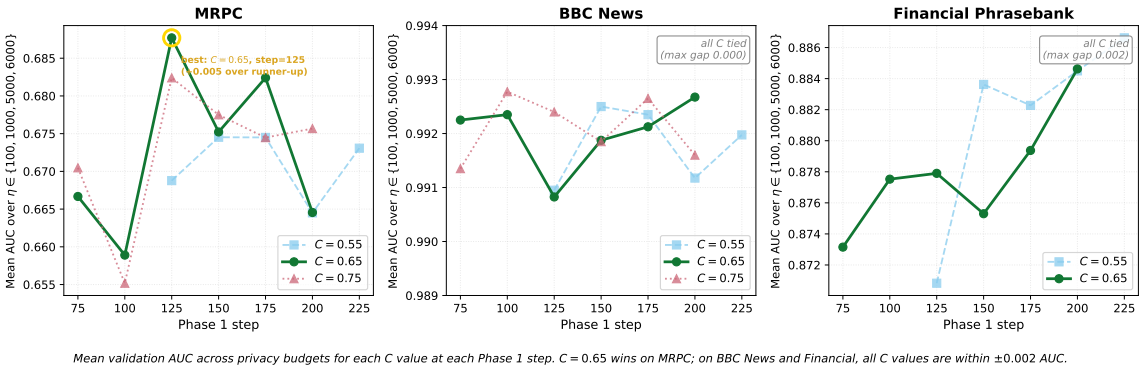

Figure 4: Mean validation AUC across $\eta \in \{100, 1000, 5000, 6000\}$ for each C value at each Stage 1 step. Gold circle marks a C choice that leads the runner-up by more than 0.005 AUC.

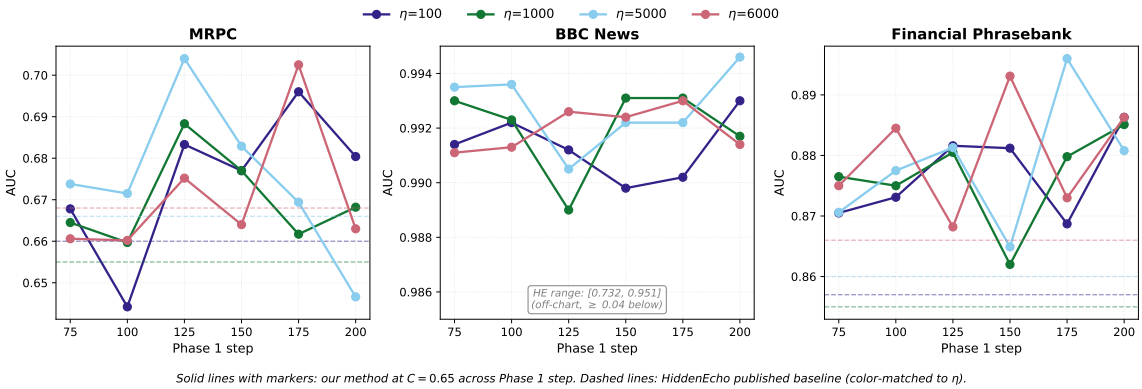

Figure 5: AUC versus Stage 1 step at $C = 0.65$ for each $(\text{dataset}, \eta)$ cell. Solid lines with markers: our method. Dashed lines: HiddenEcho published baseline at matched η. BBC News y-axis is tightened to its near-ceiling range; the HiddenEcho values fall off-chart below.