# TLC-Calibrator: Latency-Efficient LoRA-Split Inference for Edge LLMs via Two-Tiered Communication Calibration

**Anonymous ACL submission**

## Abstract

Large language models (LLMs) achieve strong generalization across diverse tasks, but their size hinders personalized deployment on user devices. Low-Rank Adaptation (LoRA) enables user-specific fine-tuning with minimal additional parameters, and its structural separability naturally supports collaborative edge inference, where the frozen base model runs on an edge server and lightweight LoRA adapters reside on user devices for privacy and scalability. However, each LoRA layer induces two edge–device communication rounds to exchange hidden states and LoRA-updated projections, making communication latency dominate inference time. Our empirical analysis shows that the impact of LoRA is highly uneven across layers and tokens, and skipping LoRA in low-impact regions leads to negligible accuracy loss. Building on this observation, we propose `TLC-Calibrator`, a two-tiered communication calibration framework that adaptively decides when LoRA-related communication is necessary. A server-side calibrator determines whether to transmit intermediate activations to the device, while a device-side calibrator decides whether the resulting LoRA projections should be sent back. Experiments show that `TLC-Calibrator`achieves up to 2.4× speedup with less than 1.9% accuracy loss.

## 1 Introduction

Recent advances in large language models (LLMs) have enabled strong general-purpose reasoning and generation across diverse natural language tasks (Brown et al., 2020; Xu et al., 2025; Wei et al., 2022). To support downstream adaptation without retraining full models, *Low-Rank Adaptation* (LoRA) has emerged as a parameter-efficient fine-tuning method (Hu et al., 2022; Yin et al., 2023). By injecting a small number of trainable low-rank matrices, typically into the Q/K/V projections of each Transformer layer, LoRA achieves strong task-specific performance with minimal parameter and memory overhead.

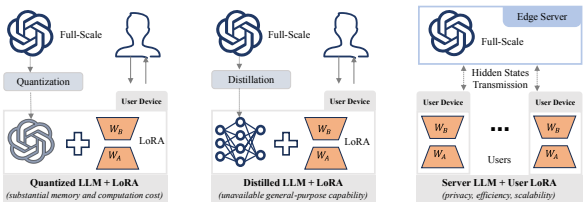



Figure 1: Comparison of three strategies for personalized LLM deployment on user devices.

With the growing demand for personalized and privacy-preserving LLM deployment in user environments (Qu et al., 2025; Zheng et al., 2025; Liu et al., 2024), extensive efforts have been made to reduce the resource demands of LoRA-based LLMs on resource-constrained user devices, such as model quantization (Xiao et al., 2023; Lee et al., 2025; Xia et al., 2024) in Fig. 1 (**Left**) and model distillation (Muralidharan et al., 2024; Ko et al., 2024) in Fig. 1 (**Middle**). However, quantization still requires substantial memory and computation for mainstream LLM inference, often beyond consumer devices, and typically incurs non-negligible performance degradation (Xu et al., 2024; Jin et al., 2024; Gong et al., 2024). Model distillation, in contrast, greatly reduces resource usage but compromises the general-purpose capability of LLMs (Yang et al., 2024; Zhu et al., 2024).

Inspired by split inference (Matsubara et al., 2022; Wang et al., 2024; Jung et al., 2025), a promising paradigm for mitigating the above dilemma is to exploit the *inherent structural separability of LoRA* and decouple the LLM into two parts: (1) the frozen base model, deployed on a capable edge server; and (2) user-specific LoRA adapters, deployed on resource-constrained user devices. As shown in Fig. 1 (**Right**), this architecture, termed *Lo*RA-*S*plit *Inference* (***LoS-Inference***), together with low-latency edge infrastructure providing microsecond-level edge–device communica-

Table 1: Impact of skipping LoRA in different layers, reported as EM score drop (%) on a QA task.

| Layer | EM Drop (%) | | |
|---|---|---|---|
| | LLaMA-3B | LLaMA-7B | OPT-13B |
| 1 | $< 0.1$ | $< 0.1$ | $< 0.1$ |
| 8 | $< 0.5$ | $< 0.5$ | $< 0.5$ |
| 16 | 0.7 | 1.0 | 1.2 |
| 24 | 1.2 | 1.7 | 2.0 |

tion (Mao et al., 2017; Luo et al., 2021; Hua et al., 2023), enables scalable multi-user inference: the heavy base model is shared across users, while lightweight LoRA adapters capture personal or task-specific preferences locally without exposing sensitive parameters.

However, unlike most split-inference frameworks, where only a single forward transmission crosses the partition (Wu et al., 2022; Lim et al., 2024), LoS-Inference for LLMs incurs substantial communication overhead. Each base model layer requires **two rounds** of edge–device interaction: **(1)** the edge sends intermediate hidden states to the device for LoRA projection, and **(2)** the device returns the adapted Q/K/V values, which are merged with the base projections at the edge to continue the layer computation. *This overhead becomes particularly pronounced in autoregressive generation*, where inference proceeds token by token across all layers, raising a natural question:

*Can we skip LoRA communication and computation without noticeably degrading performance?*

To answer this, we perform a preliminary study that measures the per-layer accuracy impact when selectively disabling LoRA across different layers under the same setting, as shown in Table 1. We observe: (1) for some layers, especially early ones handling generic or factual context, the output quality remains nearly unchanged when LoRA is skipped; (2) in contrast, skipping LoRA in other layers causes substantial drops in the final EM score. These results indicate that *the utility of LoRA varies across layers and tokens*, and that *skipping low-impact LoRA is both feasible and beneficial*, provided we can accurately identify when and where to do so.

Therefore, we first conduct a comprehensive empirical study to identify which layer-wise metrics best capture the impact of a layer's LoRA adapter on final performance. We then propose a Two-Tiered LoS-Inference Communication Calibrator (`TLC-Calibrator`), which selectively prunes unnecessary communication during split inference of LoRA-based LLMs. `TLC-Calibrator` comprises two lightweight decision modules: Calibrator-1 on the edge decides whether to send the current hidden state to the device for LoRA processing, and Calibrator-2 on the device decides whether the resulting LoRA projections are significant enough to be transmitted back. Both calibrators are trained to estimate the contextual importance of LoRA adaptation and are optimized to minimize communication while preserving accuracy. Notably, Calibrator-2 requires no access to base model projections and relies solely on local statistics, introducing no extra communication. Experiments show that `TLC-Calibrator` achieves speedup by up to 60% with less than 1.9% performance degradation.

The main contributions are summarized as:

• We present the first split-inference framework for LoRA-adapted LLMs in edge environments, enabling collaborative edge-device inference while preserving user personalization and privacy.

• We propose efficient and interpretable communication calibrators based on lightweight features, supporting selective low-latency transmission of hidden states and LoRA-adapted projections.

• We conduct extensive experiments on multiple LLMs and network settings, showing that our two-tiered calibration substantially reduces communication overhead with negligible impact on accuracy.

## 2 Background

### 2.1 Low-Rank Adaptation

LLMs consist of stacked Transformer layers, where each layer applies Q/K/V projections, multi-head attention (MHA), and a feedforward network (FFN) to the incoming hidden states. Fully fine-tuning these layers on user tasks is often impractical due to the large dimensionality and resource constraints. Low-Rank Adaptation provides a lightweight alternative by injecting trainable low-rank matrices into pretrained weights such as the Q/K/V projections.

Formally, the Q/K/V projection weight matrix $W \in \mathbb{R}^{d\times d}$ in each layer is reparameterized as

$$W' = W + \Delta W, \quad \Delta W = BA,$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times d}$, $r \ll d$, and only $A$ and $B$ are updated on user-specific data while $W$ remains frozen. Since the number of trainable parameters satisfies $2dr \ll d^2$, LoRA attains strong adaptation performance with minimal parameter and memory overhead.

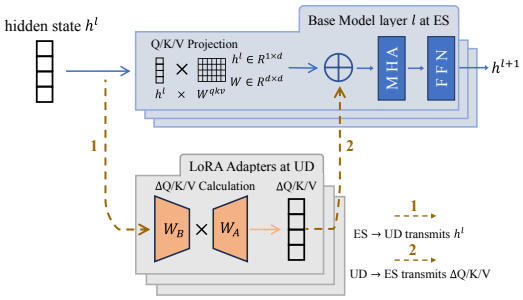



Figure 2: The architecture of LoS-Inference in Layer $l$, where Add&Norm is omitted for brevity.

### 2.2 LoRA Split Inference

By harnessing the powerful computational capabilities of edge servers and high-speed transmission capabilities of edge computing, this paper introduces a novel collaborative inference paradigm for LoRA-based LLM in edge environments, where the full LLM base model is hosted on edge, while personalized LoRA adapters are placed on device. This paradigm enables efficient reuse of the heavy base model across multiple users on the edge, as well as the personalized and privacy-preserving adaptation via LoRA modules on the device.

As shown in Fig. 2, the collaborative LoS-Inference in each Layer $l$ involves five steps:

1) Compute base projections, $Q_{\text{base}}$, $K_{\text{base}}$, $V_{\text{base}}$ at edge using the hidden state $\mathbf{h}^l$ of the base model;

2) Send the hidden state $\mathbf{h}^l$ from edge to device;

3) Compute LoRA projections, $Q_{\text{lora}}$, $K_{\text{lora}}$, $V_{\text{lora}}$ at device using the transmitted $\mathbf{h}^l$;

4) Send the LoRA projections back to edge;

5) Merge base and LoRA projections at edge, followed by MHA and FFN of Layer $l$ at edge.

### 2.3 Communication Bottleneck

Despite the low-latency communication supported by edge computing, the two rounds of edge–device communication in each layer during auto-regressive token generation (coupled with the high dimension of hidden states and LoRA adaptations) quickly become a bottleneck.

1) Each layer takes two-round transmission, i.e., the dotted green arrows 1 and 2 in Fig. 2.

2) The two-round transmission includes high-dimensional vectors $\mathbf{h}^l \in \mathbb{R}^d$ and three sets of LoRA Q/K/V projections.

3) For $L$-layer LLM, this yields $2L \times T$ rounds for the auto-regressive generation of $T$ tokens.

We profile the latency breakdown of LoS-Inference for several LLMs in Appendix A.1, where communication overheads are over 90%.

## 3 LoRA Impact Indicators

### 3.1 Layer-Wise and Output-Level Metrics

We assess three layer-wise metrics, which capture distinct aspects of the impact perturbations introduced by the LoRA adapter against the final output-level metric, to identify the most reliable proxy for communication calibration.

***1) Layer-Wise Metrics***

• *Attention Output Divergence (AOD)* (Tong et al., 2019), measures the magnitude of raw perturbations in the attention[1] outputs caused by LoRA adapter. Formally, for the $l$-th layer,

$$\text{AOD}^l = \|\text{Attn}^l_{\text{base}} - \text{Attn}^l_{\text{lora}}\|_2,$$

where $\text{Attn}^l_{\text{base}}$ denotes the attention output of the base model and $\text{Attn}^l_{\text{lora}}$ denotes the output after applying LoRA.

• *Representation Direction Alteration (RDA)* (Chang et al., 2023), captures the extent to which the semantic direction of the attention changes after LoRA integration. It is defined as:

$$\text{RDA}^l = 1 - \cos\left(\text{Attn}^l_{\text{base}}, \text{Attn}^l_{\text{lora}}\right).$$

Here, $\cos(\cdot,\cdot)$ is cosine similarity.

• *Variance Shift (VS)* (DeRose et al., 2020), quantifies how LoRA affects attention dispersions.

$$\text{VS}^l = \frac{\text{Var}(\text{Attn}^l_{\text{lora}})}{\text{Var}(\text{Attn}^l_{\text{base}})}.$$

When $\text{VS}^l > 1$, LoRA amplifies the variance of representation space; If $\text{VS}^l < 1$, it compresses it.

***2) Output-Level Metric***

• *Final Logit Shift (FLS)* (Hanna et al., 2023), measures the extent to which the final prediction logits[2] change by LoRA adaptation. Formally, let $\mathbf{y}_{\text{base}}$ denote the logits of base model and $\mathbf{y}_{\text{lora}}$ the corresponding logits after integrating LoRA, then $\text{FLS} = \|\mathbf{y}_{\text{base}} - \mathbf{y}_{\text{lora}}\|_2$.

### 3.2 Metric Correlation Analysis

The correlation analysis in Table 2 shows that **AOD provides the strongest alignment with output-level indicators** across all model–dataset pairs. Its

[1] In the $l$-th Transformer layer, $\text{Attn}^l = \text{Softmax}\left(\frac{Q^l {K^l}^\top}{\sqrt{d_k}}\right) V^l$, where $Q/K/V^l = H^{(l-1)} W^l_{Q/K/V}$ are the Q/K/V projections of the hidden state $H^{(l-1)}$ from previous layer, and $d_k$ is the key dimension.

[2] $\mathbf{y} \in \mathbb{R}^{|V|}$ denotes the vector of unnormalized scores (logits) over the vocabulary $V$ at the final layer, which are converted into token probabilities by $p(t) = \text{Softmax}(\mathbf{y})$.

Table 2: Pearson correlation between the layer-wise metric (AOD/RDA/VS) and output-level metric (FLS).

| Metric | OPT-13B | | LLaMA-7B | |
|---|---|---|---|---|
| | ELI5 | COQA | ELI5 | COQA |
| AOD | **0.91** | **0.88** | **0.91** | **0.91** |
| RDA | 0.75 | 0.72 | 0.81 | 0.85 |
| VS | 0.55 | 0.55 | 0.61 | 0.64 |

correlations with FLS reaches 0.91 on ELI5, consistently outperforming the other metrics. RDA offers moderate predictive strength (0.72–0.85), while VS shows the weakest association, typically below 0.64, respectively.

We also observe a generally positive association between layer-wise metrics and FLS across models and datasets, as shown in Fig. 10 of Appendix B.1. Among the metrics, AOD aligns most tightly with FLS, showing clear monotonic trends in both shallow and deep layers. RDA also tracks performance degradation but with more variance, while VS appears the least stable, often exhibiting dispersed and noisy correlations.

Taken together, these observations highlight AOD as the most reliable layer-wise proxy for predicting downstream degradation.

# 4 Two-Tiered Communication Calibration

We now present `TLC-Calibrator`, a runtime framework that adaptively determines whether to invoke LoRA communication at each layer and token. Building on the empirical observations in Section 3, `TLC-Calibrator` aims to preserve model performance while minimizing redundant transmissions. It consists of two lightweight calibrators deployed at edge and device, respectively, each responsible for filtering unnecessary communication based on local statistics. Together, they enable a fine-grained, per-layer and per-token control over LoRA invocation, as shown in Fig. 3.

- **Tier-1 Calibrator** $\pi_{\mathrm{T1}}$ (edge-side, on the top of Fig. 3): decides whether to send the current activation $\mathbf{h}^{(l)}$ to device.
- **Tier-2 Calibrator** $\pi_{\mathrm{T2}}$ (device-side, at the bottom of Fig. 3): decides whether to transmit the resulting LoRA Q/K/V projections $Q/K/V_{\mathrm{LoRA}}$ back to edge.

These calibrators are model-agnostic, require only lightweight statistics as input, and are trained to approximate the importance of LoRA adapters with minimal latency overhead.

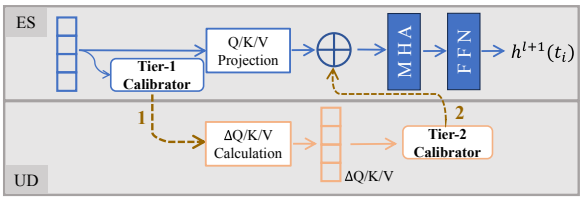



Figure 3: Workflow of `TLC-Calibrator` at each layer $l$.

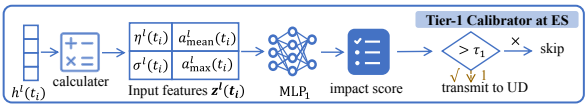



Figure 4: Tier-1 calibrator at ES, decides whether to transmit the hidden state $\mathbf{h}^l(t_i)$ to UD for LoRA adaptation.

## 4.1 Tier-1 Calibrator

The Tier-1 calibrator $\pi_{\mathrm{T1}}$ determines whether the intermediate activation $\mathbf{h}^l(t_i)$ at layer $l$ contains sufficient semantic or syntactic information to warrant transmission to device for LoRA adaptation.

**1) Input Features.** For each layer $l$ and token $t_i$, we extract the following features from $\mathbf{h}^l(t_i)$:

**Token-Wise $\ell_2$ Norm** (Wang et al., 2025) measures the importance of token $t_i$ by evaluating the magnitude of its activation at layer $l$, defined as:

$$\eta^l(t_i) = \|\mathbf{h}^l(t_i)\|_2 = \sqrt{\sum_{j=1}^{d} h_j^l(t_i)^2} \tag{1}$$

where $d$ is the activation dimension.

**Mean and Max Activations** of token $t_i$ at layer $l$ are defined as:

$$a^l_{\mathrm{mean}}(t_i) = \frac{1}{d}\sum_{j=1}^{d} h_j^l(t_i), \tag{2}$$

$$a^l_{\mathrm{max}}(t_i) = \max_j \left(h_j^l(t_i)\right) \tag{3}$$

where $h_j^l(t_i)$ is the $j$-th component of $\mathbf{h}^l(t_i)$.

**Positional Standard Deviation** (Chi et al., 2023) is computed as:

$$\sigma^l(t_i) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{h}_n^l(t_i) - \bar{\mathbf{h}}^l(t_i)\right)^2} \tag{4}$$

where $\mathbf{h}_n^l(t_i)$ is the activation of token $t_i$ at layer $l$ for input sequence $n$, $\bar{\mathbf{h}}^l(t_i)$ is the mean activation of token $t_i$ across all sequences, and $N$ is the total number of input sequences.

**2) Predictor Design.** As shown in Fig. 4, the Tier-1 calibrator $\pi_{\mathrm{T1}}$ utilizes a shallow Multi-Layer Perceptron (MLP) at layer $l$ of the base model in

edge to learn the relationship between the feature vector $z^l(t_i)$ and the utility of transmitting $\mathbf{h}^l(t_i)$ for LoRA adaptation. The MLP outputs an impact score $\hat{\Delta}^l_{\text{T1}}$, which indicates the contribution of LoRA at layer $l$:

$$\hat{\Delta}^{l,t_i}_{\text{T1}} = \pi_{\text{T1}}(z^l(t_i)).$$

The decision to transmit $\mathbf{h}^l(t_i)$ from edge to device is made by comparing $\hat{\Delta}^{l,t_i}_{\text{T1}}$ to a threshold $\tau_1$:

$$\hat{S}_{t,l} = \begin{cases} 1 & \text{if } \hat{\Delta}^{l,t_i}_{\text{T1}} > \tau_1 \\ 0 & \text{otherwise} \end{cases}$$

### 4.2 Tier-2 Calibrator

The Tier-2 calibrator at device decides whether LoRA adaptations should be sent back to edge.

**1) Input Features.** For each layer $l$ and token $t$, the following features are computed from the LoRA adaptation values $\Delta Q^l_t, \Delta K^l_t, \Delta V^l_t$ at device:

**Energy Ratio** (Ma et al., 2021) is defined as:

$$r^l_{Q/K/V}(t_i) = \frac{\|Q/K/V^{l,t_i}_{\text{LoRA}}\|_2}{\|Q/K/V^{l,t_i}_{\text{base}}\|_2}, \tag{5}$$

where $Q/K/V^{l,t_i}_{\text{LoRA}} = \Delta W^l_{Q/K/V}\mathbf{h}^l(t_i)$ and $Q/K/V^{l,t_i}_{\text{base}} = W^l_{Q/K/V}\mathbf{h}^l(t_i)$. The base model's $\ell_2$-norm is lightweight, and its computation finishes before LoRA projections, ensuring low overhead.

**Self-Similarity** of Q projection is computed as:

$$\psi^l_Q(t_i) = \cos(Q^{l,t_i}_{\text{LoRA}}, Q^{l-1,t_i}_{\text{LoRA}}). \tag{6}$$

Similar self-similarities, $\psi^l_K(t_i)$ and $\psi^l_V(t_i)$, can be computed for K/V projections.

**Token Entropy** (Lu et al., 2024) of Q projection is calculated as:

$$e^l_Q(t_i) = -\sum_{\alpha=1}^{d} p_\alpha \log(p_\alpha) \tag{7}$$

where $p_\alpha$ is the probability distribution over the token's LoRA projection, computed as:

$$p_\alpha = \frac{\exp(v_\alpha)}{\sum_{j=1}^{d}\exp(v_j)}, \tag{8}$$

with $\mathbf{v} = Q^{l,t_i}_{\text{LoRA}} \in \mathbb{R}^d$ being the Q projection.

**Projection Diversity** (Nguyen et al., 2022) of Q projection is calculated as:

$$d^l_Q(t_i) = \|Q^{l,t_i}_{\text{LoRA}} - \bar{Q}^l_{\text{LoRA}}\|^2_2 \tag{9}$$

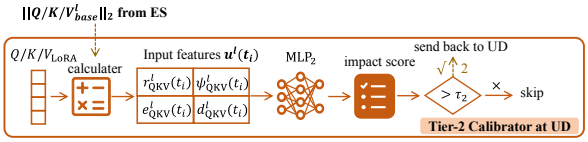



Figure 5: Tier-2 calibrator at the UD, decides whether to send back the LoRA adaptation $Q/K/V_{\text{LoRA}}$ to ES for merging.

where $\bar{Q}^l_{\text{LoRA}}$ is the mean projection across tokens.

**2) Predictor Design.** Similar to Tier-1, the Tier-2 calibrator $\pi_{\text{T2}}$ uses a shallow MLP (Fig. 5) to output an importance score $\hat{\Delta}^{l,t_i}_{\text{T2}}$:

$$\hat{\Delta}^{l,t_i}_{\text{T2}} = \pi_{\text{T2}}(u^l(t_i)).$$

If $\hat{\Delta}^{l,t_i}_{\text{T2}} > \tau_2$, the LoRA projections are transmitted back to edge; otherwise, they are discarded.

Inference algorithm is detailed in Appendix B.2.

## 5 Experiments

### 5.1 Experimental Setup

`Models`: 1) OPT-1.3B (FP32): a lightweight transformer suitable for fully on-device execution. 2) LLaMA-3B (FP16): a compact LLaMA-series model that offers stronger reasoning capability. 3) LLaMA-7B (FP16): a widely adopted mid-size LLM that serves as a standard reference point. 4) OPT-13B (INT8): a high-capacity model that stresses both communication and computation in edge-server pipelines.

`Datasets`: 1) **COQA** (Reddy et al., 2019), a multi-turn dialogue dataset where each question depends on prior user turns. 2) **ELI5** (Fan et al., 2019), a long-form, user-generated QA dataset from Reddit.

`Baselines`: 1) **Local** [Device-Only] (Hu et al., 2022), runs base model and LoRA adapters on device, ensuring full privacy. 2) **MLoRA** [Device-Only] (Li et al., 2025), is an on-device optimization that reduces KV-cache memory by dynamically pruning low-importance entries. 3) **LInfer** [Device-Edge], is a native edge-device pipeline where the base model runs on edge and LoRA adapters on device. 4) **DLoRA** [Device-Edge] (Gao and Zhang, 2024), skips LoRA activations of low-importance tokens. 5) **Server** [Edge-Only], places both base model and LoRA adapters entirely on edge, achieving the highest throughput regardless of privacy.

`Metrics`: 1) **Latency**, denotes the average per-token inference latency. 2) **F1 Score**, denotes the accuracy of the generated answer.

`Implementation`: Experiments run on NVIDIA A800 (edge) and 13 TFLOPS-class GPUs (device).

Table 3: Average latency/token (s) and F1 (%). OOM: Out-Of-Memory. D-O: Device-Only, D-E: Device-Edge, E-O: Edge-Only.

| Type | Method | COQA | | | | | | | | ELI5 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | OPT-1.3B | | LLaMA-3B | | LLaMA-7B | | OPT-13B | | OPT-1.3B | | LLaMA-3B | | LLaMA-7B | | OPT-13B | |
| | | Lat. | F1 | Lat. | F1 | Lat. | F1 | Lat. | F1 | Lat. | F1 | Lat. | F1 | Lat. | F1 | Lat. | F1 |
| D-O | Local | 0.236 | 62.1 | 0.263 | 67.4 | OOM | OOM | OOM | OOM | 0.223 | 40.2 | 0.254 | 43.6 | OOM | OOM | OOM | OOM |
| | MLoRA | 0.203 | 61.4 | 0.222 | 66.7 | OOM | OOM | OOM | OOM | 0.197 | 38.9 | 0.216 | 42.7 | OOM | OOM | OOM | OOM |
| D-E | LInfer | 0.107 | 62.1 | 0.150 | 67.4 | 0.206 | 75.3 | 0.262 | 77.6 | 0.094 | 40.2 | 0.142 | 43.6 | 0.204 | 46.9 | 0.263 | 52.1 |
| | DLoRA | 0.078 | 61.2 | 0.105 | 66.5 | 0.139 | 74.6 | 0.165 | 76.7 | 0.073 | 38.7 | 0.095 | 42.5 | 0.123 | 45.9 | 0.154 | 51.1 |
| | Ours | 0.057 | 60.8 | 0.074 | 66.2 | 0.094 | 74.5 | 0.107 | 76.3 | 0.054 | 38.3 | 0.072 | 42.1 | 0.098 | 45.6 | 0.102 | 50.7 |
| E-O | Server | **0.026** | 62.1 | **0.032** | 67.4 | **0.038** | 75.3 | **0.045** | 77.6 | **0.027** | 40.2 | **0.032** | 43.6 | **0.039** | 46.9 | **0.045** | 52.1 |

System and method parameters are set as follows, unless otherwise specified: LoRA rank is 16, bandwidth is 1Gbps, edge and device computing capacities are 150TFLOPS and 13TFLOPS. Tier-1 and Tier-2 calibration thresholds are 0.6 and 0.8, Tier-1 feature weights are 0.2, 0.3, 0.3, 0.2, Tier-2 feature weights are 0.3, 0.3, 0.2, 0.2.

## 5.2 Main Results

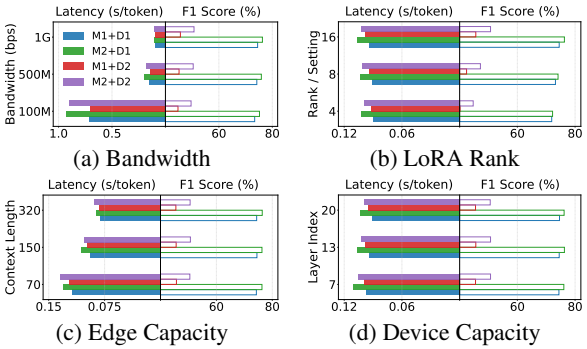
Table 3 reports end-to-end latency and F1 for four models on COQA and ELI5, where TLC-Calibrator delivers the best latency–accuracy tradeoff among all privacy-preserving methods (the first five lines except for Server, which provides the global lower-bound latency, e.g., 0.026–0.045 s, but sacrifices privacy completely).

`Latency Reduction:` TLC achieves consistent, large latency reductions relative to collaborative baselines (LInfer and DLoRA). On COQA, TLC reduces OPT-1.3B latency from 0.107 s (LInfer) and 0.078 s (DLoRA) to 0.057 s, corresponding to 1.9× and 1.37× improvements, respectively. For larger models (LLaMA-7B, OPT-13B), TLC further lowers latency to 0.094 s and 0.107 s, yielding 2.2–2.4× speedups over LInfer. On ELI5, where sequences are much longer, TLC's gains are even more pronounced, achieving 1.7–2.0× improvements across all models.

`Accuracy Preservation:` Despite aggressive communication pruning, TLC maintains accuracy nearly identical to full LoRA usage. On COQA, TLC matches or closely tracks LInfer and DLoRA, within 0.3%–1.3% F1 of the strongest baseline for each model. On ELI5, where long-form generation is more sensitive to pruning, TLC similarly preserves performance (e.g., 38.3% vs. 40.2% for OPT-1.3B and 45.6% vs. 46.9% for LLaMA-7B).

## 5.3 System Parameter Analysis

`Impact of Bandwidth:` Across 100, 500, and 1000 Mbps settings, TLC-Calibrator shows substantial latency scaling while maintaining stable accuracy. At 100 Mbps, communication dominates inference, yielding high delays (e.g., 0.712 s and 0.926 s on COQA; 0.703 s and 0.902 s on ELI5). Increasing bandwidth to 500 Mbps reduces latency by 4–5×, and moving to 1 Gbps provides another 1.5–2× improvement.



(a) Bandwidth (b) LoRA Rank (c) Edge Capacity (d) Device Capacity

Figure 6: System parameter analysis under varying bandwidths, LoRA ranks, and edge/device computing capacities, showing latency and F1 trends across models LLaMA-7B (M1), OPT-13B (M2) and datasets COQA (D1), ELI5 (D2).

`Impact of LoRA Rank:` Varying the LoRA rank $r = \{4, 8, 16\}$ produces only mild latency growth in TLC-Calibrator, with increases of merely 3–5% from $r = 4$ to $r = 16$ (e.g., 0.090 → 0.094 s on LLaMA-7B/COQA and 0.099→0.102 s on OPT-13B/ELI5). In contrast to traditional LoRA inference—where projection size scales linearly with rank—TLC absorbs most of the additional overhead by pruning low-impact layers.

`Impact of Edge Capacity:` Across all models and datasets, TLC latency improves modestly (e.g., 0.123→0.094 s on LLaMA-7B/COQA and 0.132→0.107 s on OPT-13B/COQA), while accuracy stays nearly unchanged. Since cross-device transfers constitute the main bottleneck, edge computing capacity primarily affects absolute runtime rather than TLC's pruning behavior.

`Impact of Device Capacity:` Increasing device capacity from 7→13→20 TFLOPS yields only minor latency gains for TLC (typically 3–6%, e.g.,

0.097→0.091 s on LLaMA-7B/COQA), since communication dominates overall cost.

### 5.4 Method Parameter Analysis

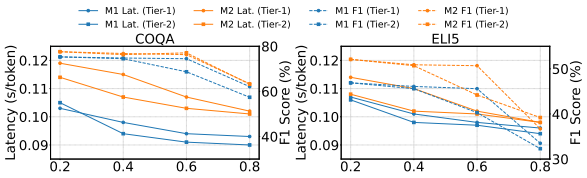
`Tier-1 Threshold:` Fig. 7(a) shows that lower thresholds (e.g., $\tau_1 = 0.2$) retain more layers and lead to the highest latency (e.g., 0.103 s on LLaMA-7B+COQA and 0.119 s on OPT-13B+COQA). Increasing $\tau_1$ gradually suppresses edge→device LoRA transfers on less-personal tokens, with $\tau_1 = 0.6$ achieving the best latency-accuracy tradeoff (0.094-0.107 s) while maintaining stable F1. However, at $\tau_1 = 0.8$ latency reduction saturates—due to transmission startup costs that persist regardless of transfer size—and accuracy drops, especially on ELI5 (12–13% F1), indicating that overly aggressive Tier-1 pruning is detrimental.

`Tier-2 Threshold:` In Fig. 7(a), raising $\tau_2$ from 0.2 to 0.4 steadily reduces device→edge returns and lowers latency (e.g., LLaMA-7B+COQA: 0.105→0.094s; OPT-13B+COQA: 0.114→0.107s). When $\tau_2 > 0.6$, latency gains plateau while accuracy degrades sharply. At $\tau_2 = 0.8$, accuracy collapses further (e.g., to 57.4% F1 on COQA and 32.3% F1 on ELI5), even though latency barely changes. This reflects that Tier-2 controls only whether to return device-side LoRA updates and excessive pruning removes useful adaptation needed for final prediction quality.

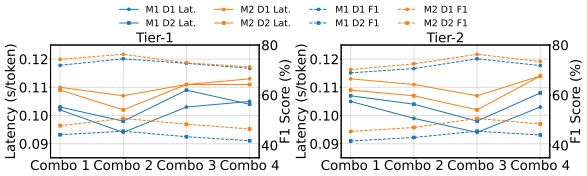
`Tier-1 and Tier-2 Feature Weights:` Fig. 7(b) assess the effect of feature-weights,
*W1:* 0.1, 0.2, 0.3, 0.4 *W2:* 0.2, 0.3, 0.3, 0.2
*W3:* 0.25, 0.25, 0.25, 0.25 *W4:* 0.4, 0.3, 0.2, 0.1
Since $\tau_1$ and $\tau_2$ are fixed, the effective pruning ratio remains nearly unchanged, producing modest latency variation (5–10%). In contrast, accuracy is more sensitive to weight imbalance: biased schemes such as W1 and W4 yield noticeably lower F1, whereas the uniform setting (W3) consistently achieves the best overall accuracy (e.g., 74.5% on COQA and 45.6% on ELI5 for LLaMA-7B).

### 5.5 Ablation Study

`Tier-1 Feature Ablation:` In Fig. 8(a), removing any of the four Tier-1 feature groups ($\eta, a_{\text{mean}}, a_{\max}, \sigma$) increases latency (from 0.094-0.102 s to 0.112-0.139 s), while accuracy remains largely stable within 30–60% F1. The largest slowdown appears when feature 1 (token-wise $\ell_2$ norm) is removed (e.g., 0.094→0.127 s on COQA; 0.098→0.125 s on ELI5), suggesting that incomplete feature sets cause Tier-1 to misidentify less-personal tokens as essential and preserve unnecessary communication.



(a) Tier-1 & Tier-2 threshold.



(b) Tier-1 & Tier-2 feature weights.

Figure 7: Method parameter analysis on calibration thresholds and feature weights, across models LLaMA-7B (M1), OPT-13B (M2) and datasets COQA (D1), ELI5 (D2).

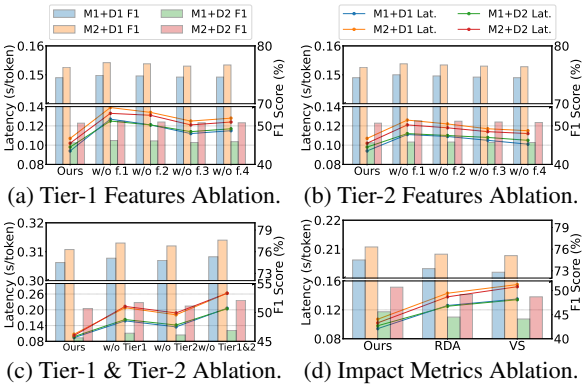
`Tier-2 Feature Ablation:` In Fig. 8(b), similar trends are observed, with modest magnitude compared to Tier-1 (Tier-2 Feature: $r, \psi, e, d$). Feature 1 contributes most: its removal raises latency from 0.094–0.102 s to 0.111–0.121 s across model-dataset pairs. Dropping features 2–4 produces smaller slowdowns (2–5%).

`Effect of Tier-1 and Tier-2 Calibrators:` Fig. 8(c) illustrates that disabling Tier 1 increases latency from 0.094–0.102 s to 0.158–0.213 s, due to the loss of layer-level pruning. Removing Tier 2 also increases latency (0.135–0.188 s), but to a lesser extent, as it mainly optimizes device→edge returns of LoRA adaptation results. When both tiers are removed, latency rises significantly (0.204–0.263 s), more than doubling the TLC latency. Accuracy remains stable with the removal of a single tier (within $\leq$80% F1), but removing both tiers allows full LoRA usage, slightly boosting F1.

`Impact Metric Ablation:` In Fig. 8(d), AOD (Ours) clearly provides the best latency–accuracy balance: across all model-dataset pairs, it achieves the lowest latency (0.094-0.107 s) and the highest F1, outperforming RDA and VS by 1.0%-1.5% F1 and 12-18% latency reduction. These trends mirror the correlation analysis in Sec. 3.2, reinforcing AOD as the most reliable layer-wise LoRA impact signal for communication calibration.

### 5.6 Visualization Analysis

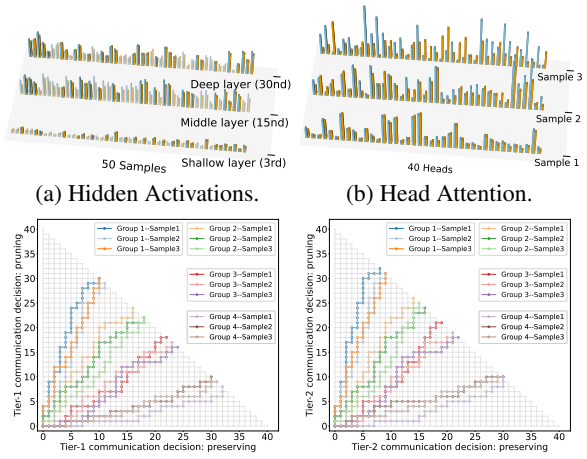
`Hidden Activation Consistency.` Fig. 9(a) visualizes the hidden activations of *shal-*




(a) Tier-1 Features Ablation. (b) Tier-2 Features Ablation.

(c) Tier-1 & Tier-2 Ablation. (d) Impact Metrics Ablation.

Figure 8: Ablation studies across models LLaMA-7B (M1), OPT-13B (M2) and datasets COQA (D1), ELI5 (D2).




(a) Hidden Activations. (b) Head Attention.

(c) Tier-1 Behavior. (d) Tier-2 Behavior.

Figure 9: Visualization Analysis. **(a)** Hidden activations under pruned (*orange-bar*) and preserved (*blue-bar*) LoRA decisions for shallow/middle/deep layers. **(b)** Multi-head attention under pruned (*orange-bar*) and preserved (*blue-bar*) LoRA decisions for three representative samples. **(c-d)** Tier-1 and Tier-2 communication decisions across sample groups.

*low*/*middle*/*deep* layers across 50 random samples, comparing pruned (*orange-bar*) and preserved (*blue-bar*) representations. (1) *shallow* layers show highly overlapping activation distributions, indicating early-layer LoRA has limited influence; (2) *middle* layers exhibit moderate divergence, where TLC selectively skips tokens whose activation (*orange-bar*) remain close to preserved (*blue-bar*) distributions; (3) *deep* layers present the largest representational sensitivity, yet skipped activations still cluster within the manifold of unskipped ones, demonstrating that TLC's decisions align with the AOD-identified stability regions.

`Multi-Head Attention Consistency`. Fig. 9(b) visualizes the 40 attention heads of a fixed layer for three representative query types: common Sample 1, medium Sample 2, and personal Sample 3. The *common* sample 1 shows almost indistinguishable attention maps, indicating low AOD sensitivity and allowing aggressive but safe pruning. The *medium* Sample 2 exhibits moderate head-level deviations while maintaining global alignment, prompting TLC to preserve only part of the layer. In contrast, for the *personal* Sample 3, attention maps show noticeable sparsity variation between pruned and unpruned runs, yet their aggregated structure remains coherent.

`Calibration Behavior Trend`. In Fig. 9(c-d), the horizontal axis denotes preserved edge→device communication, while the vertical axis indicates pruned device→edge transmission. Sample groups range from common factual queries (Group 1) to highly personalized ones (Query 4). Group 1 concentrates near the pruning axis, reflecting low semantic complexity and low AOD/RDA scores; Groups 2–3 shift rightward with increasing personalization, requiring more retained communication; Group 4 sits in the high-communication region, consistent with strong deep-layer semantic contributions. This shows that TLC can adjust pruning according to sample personalization, aggressively pruning less-personal queries while preserving deeper edge-device interaction for more personalized ones.

### 5.7 Case Studies & Overhead Analysis

We additionally give a detailed case study on two representative QA samples (common factual and highly personal) in Appendix C.1, showing the impact scores, calibration decisions, and time costs of each layer for each generated token, along with the entire QA text. These details reveal the running rules of TLC-Calibrator on different samples and their tokens, in each layer.

We also conduct an overhead analysis and find that the time and memory costs are much smaller than the efficiency gains of TLC, details of which could be seen in Appendix C.2.

## 6 Conclusion

This paper presented a two-tiered communication calibration framework that adaptively decides when LoRA-related communication is needed in split inference. By coordinating server- and device-side calibrators, TLC-Calibrator reduces communication overhead while maintaining accuracy, making LoRA-based personalized LLM deployment more practical in edge environments.

## Limitations

The proposed LoS-Inference framework has some limitations. It heavily relies on high-speed edge-device communication, which could become a bottleneck in real-world scenarios with limited network bandwidth. Furthermore, LoRA computation on user devices is not fully optimized for long sequences, and the overhead may increase significantly when processing longer queries, such as those involved in in-context learning or retrieval-augmented generation.

## References


Tom Brown, Benjamin Mann, Nick Ryder, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Tyler Chang, Kishaloy Halder, Neha Anna John, Yogarshi Vyas, Yassine Benajiba, Miguel Ballesteros, and Dan Roth. 2023. Characterizing and measuring linguistic dataset drift. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8953–8967.

Ta-Chung Chi, Ting-Han Fan, Li-Wei Chen, Alexander Rudnicky, and Peter Ramadge. 2023. Latent positional information is in the self-attention variance of transformer language models without positional embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1183–1193.

Joseph F DeRose, Jiayao Wang, and Matthew Berger. 2020. Attention flows: Analyzing and comparing attention mechanisms in language models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1160–1170.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567.

Chao Gao and Sai Qian Zhang. 2024. Dlora: Distributed parameter-efficient fine-tuning solution for large language model. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13703–13714.

Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. 2024. What makes quantization for large language model hard? an empirical study from the lens of perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18082–18089.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060.

Edward J Hu, Yelong Shen, Phillip Wallis, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. 2023. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9):1–35.

Renren Jin, Jiangcun Du, Wuwei Huang, Wei Liu, Jian Luan, Bin Wang, and Deyi Xiong. 2024. A comprehensive evaluation of quantization strategies for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 12186–12215.

Daeyoung Jung, Jaewook Lee, Hyeonjae Jeong, Dongju Cha, Heewon Kim, and Sangheon Pack. 2025. Split computing for mobile devices: Energy and latency perspective. *IEEE Transactions on Services Computing*.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Seyoung Yun. 2024. Distillm: Towards streamlined distillation for large language models. In *The Forty-first International Conference on Machine Learning*. ICML.

Geonho Lee, Janghwan Lee, Sukjin Hong, Minsoo Kim, Euijai Ahn, Du-Seong Chang, and Jungwook Choi. 2025. Rilq: Rank-insensitive lora-based quantization error compensation for boosting 2-bit large language model accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18091–18100.

Borui Li, Yitao Wang, Haoran Ma, Ligeng Chen, Jun Xiao, and Shuai Wang. 2025. Mobilora: Accelerating lora-based llm inference on mobile devices via context-aware kv cache optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23400–23410.

Jeong-A Lim, Joohyun Lee, Jeongho Kwak, and Yeongjin Kim. 2024. Cutting-edge inference: Dynamic dnn model partitioning and resource scaling for mobile ai. *IEEE Transactions on Services Computing*, 17(6):3300–3316.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, and 1 others. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking

detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735.

Quyuan Luo, Shihong Hu, Changle Li, Guanghui Li, and Weisong Shi. 2021. Resource scheduling in edge computing: A survey. *IEEE communications surveys & tutorials*, 23(4):2131–2165.

Weicheng Ma, Kai Zhang, Renze Lou, Lili Wang, and Soroush Vosoughi. 2021. Contributions of transformer attention heads in multi-and cross-lingual tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1956–1966.

Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358.

Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102.

Tan Nguyen, Tam Nguyen, Hai Do, Khai Nguyen, Vishwanath Saragadam, Minh Pham, Khuong Duy Nguyen, Nhat Ho, and Stanley Osher. 2022. Improving transformer with an admixture of attention heads. *Advances in neural information processing systems*, 35:27937–27952.

Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. 2025. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Xiao Tong, Li Yinqiao, Zhu Jingbo, Yu Zhengtao, and Liu Tongran. 2019. Sharing attention weights for fast transformer. In *Proceedings of the IJCAI*, pages 5292–5298.

Hui Wang, Shujie Liu, Lingwei Meng, Jinyu Li, Yifan Yang, Shiwan Zhao, Haiyang Sun, Yanqing Liu, Haoqin Sun, Jiaming Zhou, and 1 others. 2025. Felle: Autoregressive speech synthesis with token-wise coarse-to-fine flow matching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10229–10238.

Yingchao Wang, Chen Yang, Shulin Lan, Liehuang Zhu, and Yan Zhang. 2024. End-edge-cloud collaborative computing for deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jing Wu, Lin Wang, Qiangyu Pei, Xingqi Cui, Fangming Liu, and Tingting Yang. 2022. Hitdl: High-throughput deep learning inference at the hybrid mobile edge. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4499–4514.

Yifei Xia, Fangcheng Fu, Wentao Zhang, Jiawei Jiang, and Bin Cui. 2024. Efficient multi-task llm quantization and serving for multiple lora adapters. *Advances in Neural Information Processing Systems*, 37:63686–63714.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Daliang Xu, Wangsong Yin, Hao Zhang, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2024. Edgellm: Fast on-device llm inference with speculative decoding. *IEEE Transactions on Mobile Computing*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*.

Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024. Survey on knowledge distillation for large language models: methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*.

Dongshuo Yin, Yiran Yang, Zhechao Wang, Hongfeng Yu, Kaiwen Wei, and Xian Sun. 2023. 1% vs 100%: Parameter-efficient low rank adapter for dense predictions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20116–20126.

Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. 2025. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 57(8):1–35.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

# A Appendix for Motivation

## A.1 Communication Bottleneck of LoS-Inference

As shown in Table 4, communication dominates the per-layer latency under a 1 Gbps edge–device setup. For all four models, the combined E→D and D→E time is around 1–3 ms, accounting for over 80% of the total, while the base forward computation remains below 1 ms and the LoRA projection below 0.15 ms. The increase in model size from OPT-1.3B to OPT-13B leads to only modest growth in on-device and server-side computation, but barely changes the communication cost (%), indicating that network bandwidth rather than GPU throughput is the primary bottleneck. Moreover, E→D and D→E latencies are roughly symmetric, so both directions must be reduced to achieve meaningful speedups. These observations suggest that further kernel-level optimization on GPU computations offers limited gains, and that selectively pruning LoRA-related communication is essential for improving end-to-end latency in LoS-Inference.

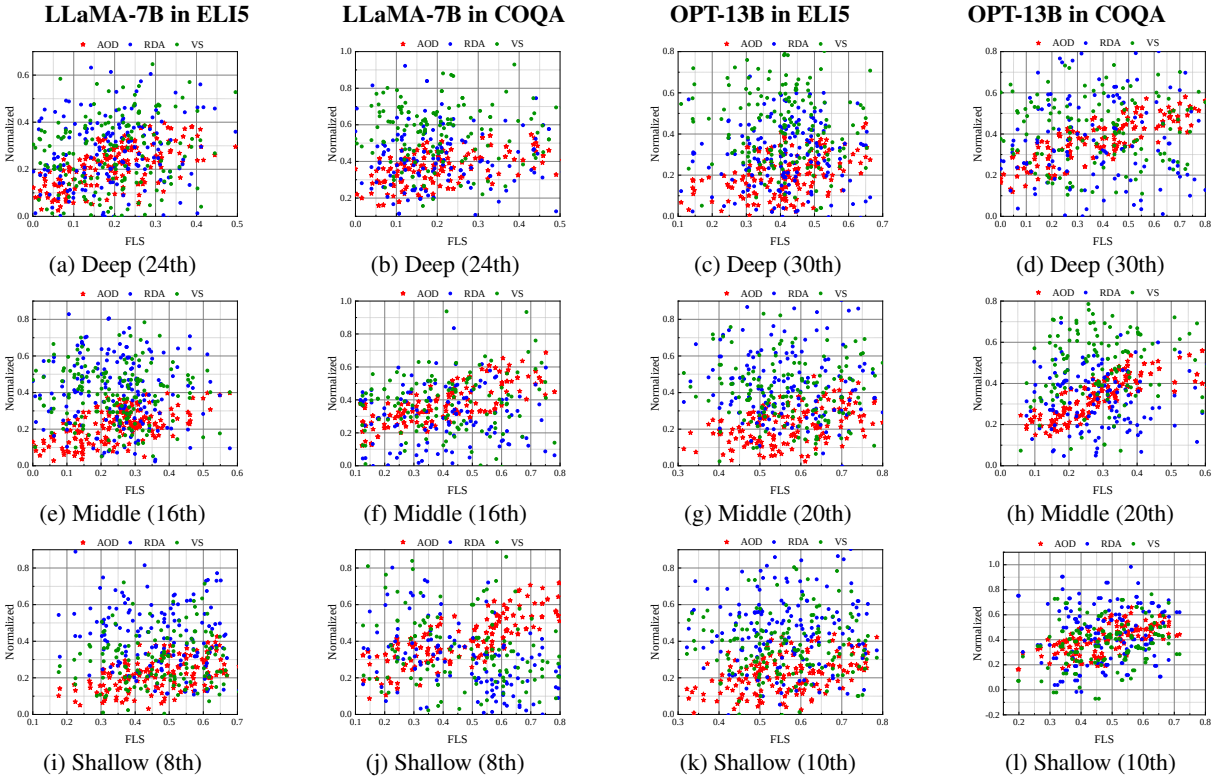
Table 4: Per-layer latency (*ms*) for different LLMs under a 1 Gbps edge-device setup. “Base" refers to forward computation (projection + attention + FFN) on the edge server. “LoRA" is the low-rank projection computation time on the user device. “Comm” groups the communication latencies from edge to device (E→D) and device to edge (D→E). “Comm%” is the communication time as a percentage of total per-layer inference time.

| Model | Base | LoRA | Comm | | Total | Comm% |
|---|---|---|---|---|---|---|
| | | | E→D | D→E | | |
| OPT-1.3B | 0.52 | 0.07 | 1.03 | 1.00 | 2.68 | 80.6 |
| LLaMA-3B | 0.63 | 0.10 | 2.15 | 2.06 | 4.21 | 85.1 |
| LLaMA-7B | 0.79 | 0.10 | 2.68 | 2.57 | 6.44 | 87.7 |
| OPT-13B | 0.86 | 0.13 | 2.76 | 2.66 | 6.55 | 86.8 |

# B Appendix for Method

## B.1 Metric Correlation Analysis

In Fig. 10, we observe a generally positive association between layer-wise metrics (AOD, RDA, VS) and FLS across models and datasets, using 200 query samples from COQA. Among the metrics, AOD align most tightly with FLS, showing clear monotonic trends in both shallow and deep layers, as shown by red star points. RDA also tracks performance degradation but with more variance, while VS appears the least stable, often exhibiting dispersed and noisy correlations. Model capacity influences separability: larger models (e.g., OPT-13B) reveal sharper distinctions between low- and high-impact layers as shown in the right column




(a) Deep (24th) (b) Deep (24th) (c) Deep (30th) (d) Deep (30th)

(e) Middle (16th) (f) Middle (16th) (g) Middle (20th) (h) Middle (20th)

(i) Shallow (8th) (j) Shallow (8th) (k) Shallow (10th) (l) Shallow (10th)

Figure 10: Layer-wise metric correlations. Each figure plots AOD/RDA/VS against FLS for the indicated layer.

of Fig. 10, indicating that LoRA-induced representational shifts translate more directly into output-level effects as scale increases. Dataset characteristics also modulate the patterns: ELI5 yields relatively consistent monotonicity, while CoQA introduces wider spreads, especially for VS and RDA.

### B.2 Inference Algorithm

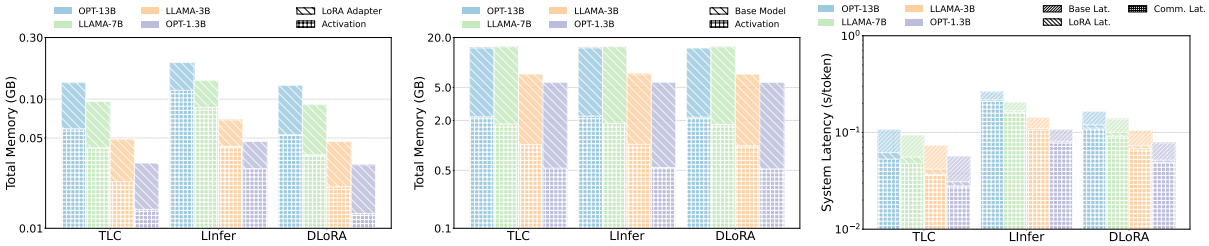
Algorithm 1 details one autoregressive step of LoS-Inference equipped with the two-tier TLC-Calibrator. The procedure starts on the user device, where the previous token $t_i$ is embedded into $\mathbf{h}^0(t_i)$, and this embedding is the only token-level representation that leaves the device at the beginning of the step. For each Transformer layer $l$, the edge server first computes the base forward pass and extracts lightweight Tier-1 features $z^l(t_i)$ from the intermediate activation $\mathbf{h}^l(t_i)$. The Tier-1 policy $\pi_{\text{T1}}$ predicts an importance score $\hat{\Delta}_{\text{T1}}^{(l,t_i)}$; only when this score exceeds a learned threshold $\tau_1$ does the server transmit $\mathbf{h}^l(t_i)$ to the device for LoRA processing, otherwise LoRA is skipped and the layer relies solely on the base projections. If $\mathbf{h}^l(t_i)$ is offloaded, the device computes the corresponding low-rank updates $\Delta Q_t^l, \Delta K_t^l, \Delta V_t^l$ and derives Tier-2 features $u^l(t_i)$, from which the Tier-2 policy $\pi_{\text{T2}}$ estimates $\hat{\Delta}_{\text{T2}}^{(l,t_i)}$. Only when this second score exceeds $\tau_2$ are the LoRA projections sent back; otherwise, transmission is suppressed and the edge continues with the base $Q/K/V$.

In this way, each token-layer pair can either use full LoRA, base-only computation, or partially pruned communication, so bandwidth is concentrated on high-impact locations while low-utility updates are skipped. Importantly, token embedding (Line 1) and final logit computation (Lines 23 and 24) are always performed on the device, so only intermediate activations and optional low-rank updates are exchanged. This design reduces communication overhead while preserving both model accuracy and user privacy.

## C Appendix for Experiments

### C.1 Case Study

We present a case study for two representative queries, i.e., common factual (Sample 1 in Table 5) and highly personal (Sample 3 in Table 6), showing the impact scores, calibration decisions of distinct layers for different generated tokens, along with the entire QA text.




(a) Overhead of device GPU usage (b) Overhead of edge GPU usage (c) Overhead of system time costs

Figure 11: Overhead analysis of TLC w.r.t. device GPU usage, edge GPU usage, and time costs.

**Algorithm 1:** LoS-Inference with TLC-Calibrator at Step $i+1$

**Input:** Token $t_i$ from the previous step $i$
**Output:** Token $t_{i+1}$ of the current step $i+1$

`/* Runs on device */`

$\mathbf{h}^0(t_i) \leftarrow \text{TokenEmbedding}(t_i)$;
Transmit $\mathbf{h}^0(t_i)$ to edge;
**for** $l = 1, \ldots, L$ **do**
  `/* Runs on edge */`
  Extract Tier-1 features $z^l(t_i)$ from $\mathbf{h}^l(t_i)$;
  $\hat{\Delta}_{\text{T1}}^{(l,t_i)} \leftarrow \pi_{\text{T1}}(z^l(t_i))$;
  **if** $\hat{\Delta}_{T1}^{(l,t_i)} > \tau_1 \quad \& \quad l > 1$ **then**
    Transmit $\mathbf{h}^l(t_i)$ to device;
  **else**
    Skip LoRA for token $t_i$ at layer $l$;
  `/* Runs on device */`
  **if** *receives* $\mathbf{h}^l(t_i)$ *from edge* **then**
    Compute LoRA projection: $\Delta Q_t^l, \Delta K_t^l, \Delta V_t^l \leftarrow \Delta W_Q \mathbf{h}^l(t_i), \Delta W_K \mathbf{h}^l(t_i), \Delta W_V \mathbf{h}^l(t_i)$;
    Extract Tier-2 features $u^l(t_i)$ from LoRA projections;
    $\hat{\Delta}_{\text{T2}}^{(l,t_i)} \leftarrow \pi_{\text{T2}}(u^l(t_i))$;
    **if** $\hat{\Delta}_{T2}^{(l,t_i)} > \tau_2$ **then**
      Transmit $\Delta Q/K/V$ back to edge;
    **else**
      Skip projection transmission for $t_i$ at layer $l$;
  **else**
    Skip LoRA for $t_i$ at layer $l$;
  `/* edge resumes MHA and FFN */`
  Merge received $\Delta Q/K/V$ if available, else use base projections $Q/K/V$;
  Compute MHA and FFN, producing $\mathbf{h}^{l+1}(t_i)$;
`/* Runs on edge */`
Transmits $\mathbf{h}^L(t_i)$ to device;
`/* Runs on device */`
Computes final logits $\hat{y} = \text{Decoder}(\mathbf{h}^L(t_i))$;
Sample token $t_{i+1}$ based on $\hat{y}$;
**return** $t_{i+1}$

## C.2 Overhead Analysis

As shown in Fig. 11, we analyze the overhead of TLC-Calibrator w.r.t. GPU memory usage of device and edge, and time costs.

**1) Device GPU Memory Usage**. On the device side, TLC-Calibrator obviously reduces memory usage compared to edge-device collaborative framework, e.g., LInfer and DLoRA. For example, in the case of OPT-13B, TLC-Calibrator uses 75.16MB for LoRA adapter, and 58.59MB for activation, whereas LInfer uses 75.16/117.37 MB. This reduction is crucial for resource-constrained devices, enabling more efficient personalization and lowering the risk of out-of-memory errors. Across all models, TLC-Calibrator consistently requires less memory than LInfer and DLoRA, emphasizing its advantage for resource-efficient deployment of personalized LLMs on user devices.

**2) Edge GPU Memory Usage**. The edge GPU memory usage of TLC-Calibrator is similar to that of LInfer and DLoRA across different models. For instance, for OPT-13B, TLC-Calibrator uses 12.75GB for base model, and 2.21GB for activation, which is almost identical to LInfer (12.75/2.25 GB) and DLoRA (12.75/2.19 GB). This shows that TLC-Calibrator does not significantly increase memory demands, allowing the system to handle large models with minimal overhead. The same pattern is seen in OPT-1.3B, LLAMA-3B and LLAMA-7B, where base model usage remain stable, with minor variations in activation. This efficient use of edge GPU memory is key for scalable LLM deployment in edge environments.

**3) Latency Costs**. TLC-Calibrator consistently offers lower system latency than LInfer and DLoRA, particularly in reducing communication overhead. For OPT-13B, TLC-Calibrator achieves 0.107 seconds total latency, including 0.045 seconds for base model processing, 0.009 seconds for LoRA adaptation, and 0.053 seconds for communication. In contrast, LInfer incurs 0.263 seconds

and DLoRA 0.165 seconds. Across models, TLC-Calibrator consistently achieves the lowest latency, enhancing split-inference efficiency. This reduction in communication time is especially valuable for edge devices, where minimizing latency is crucial, making TLC-Calibrator ideal for large-scale, personalized, and privacy-preserving LLM deployment.

Table 5: Common Sample 1: *"What habits help improve sleep?"*'s Impact Scores (IS) and Calibrator Decision (CD) across 8 layers. (× denotes that CD is pruning, ✓ denotes that CD is preserving.)

| Tokens | Layer3 | | | | Layer4 | | | | Layer5 | | | | ⇒ skip ⇒ | Layer20 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | | IS-1 | CD-1 | IS-2 | CD-2 |
| *Regular* | .001 | × | – | – | .001 | × | – | – | .002 | × | – | – | | .616 | ✓ | .294 | × |
| *sleep* | .001 | × | – | – | .001 | × | – | – | .003 | × | – | – | | .213 | × | – | – |
| *habits* | .003 | × | – | – | .001 | × | – | – | .048 | × | – | – | | .148 | × | – | – |
| *can* | .001 | × | – | – | .001 | × | – | – | .044 | × | – | – | | .768 | ✓ | .938 | ✓ |
| *significantly* | .002 | × | – | – | .001 | × | – | – | .012 | × | – | – | | .302 | × | – | – |
| *improve* | .002 | × | – | – | .002 | × | – | – | .003 | × | – | – | | .019 | × | – | – |
| *sleep* | .001 | × | – | – | .001 | × | – | – | .002 | × | – | – | | .720 | ✓ | .263 | × |
| *quality* | .003 | × | – | – | .001 | × | – | – | .018 | × | – | – | | .012 | × | – | – |
| . | .003 | × | – | – | .001 | × | – | – | .237 | × | – | – | | .221 | × | – | – |
| *Going* | .003 | × | – | – | .002 | × | – | – | .099 | × | – | – | | .379 | × | – | – |
| *to* | .004 | × | – | – | .001 | × | – | – | .143 | × | – | – | | .030 | × | – | – |
| *bed* | .003 | × | – | – | .003 | × | – | – | .096 | × | – | – | | .153 | × | – | – |
| *and* | .005 | × | – | – | .002 | × | – | – | .005 | × | – | – | | .873 | ✓ | .044 | × |
| *waking* | .004 | × | – | – | .001 | × | – | – | .436 | × | – | – | | .003 | × | – | – |
| *up* | .001 | × | – | – | .009 | × | – | – | .069 | × | – | – | | .242 | × | – | – |
| ... | | | | | | | | | | | | | | | | | |
| *Keeping* | .003 | × | – | – | .001 | × | – | – | .006 | × | – | – | | .951 | ✓ | .526 | × |
| *the* | .003 | × | – | – | .001 | × | – | – | .065 | × | – | – | | .024 | × | – | – |
| *bedroom* | .001 | × | – | – | .002 | × | – | – | .014 | × | – | – | | .881 | ✓ | .850 | ✓ |
| *dark* | .001 | × | – | – | .003 | × | – | – | .003 | × | – | – | | .794 | ✓ | .036 | × |
| , | .004 | × | – | – | .001 | × | – | – | .001 | × | – | – | | .085 | × | – | – |
| *quiet* | .003 | × | – | – | .002 | × | – | – | .056 | × | – | – | | .011 | × | – | – |
| , | .001 | × | – | – | .003 | × | – | – | .058 | × | – | – | | .097 | × | – | – |
| *and* | .005 | × | – | – | .001 | × | – | – | .004 | × | – | – | | .106 | × | – | – |
| *cool* | .003 | × | – | – | .003 | × | – | – | .019 | × | – | – | | .026 | × | – | – |
| *supports* | .003 | × | – | – | .004 | × | – | – | .057 | × | – | – | | .123 | × | – | – |
| *better* | .003 | × | – | – | .003 | × | – | – | .035 | × | – | – | | .686 | ✓ | .009 | × |
| *sleep* | .001 | × | – | – | .006 | × | – | – | .003 | × | – | – | | .192 | × | – | – |
| *quality* | .004 | × | – | – | .001 | × | – | – | .113 | × | – | – | | .347 | × | – | – |
| . | .002 | × | – | – | .001 | × | – | – | .334 | × | – | – | | .029 | × | – | – |

| Tokens | Layer21 | | | | Layer22 | | | | ⇒ skip ⇒ | Layer35 | | | | Layer36 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 |
| *Regular* | .986 | ✓ | .034 | × | .105 | × | – | – | | .899 | ✓ | .277 | × | .033 | × | – | – |
| *sleep* | .996 | ✓ | .884 | ✓ | .002 | × | – | – | | .264 | × | – | – | .171 | × | – | – |
| *habits* | .959 | ✓ | .673 | × | .004 | × | – | – | | .928 | ✓ | .827 | ✓ | .082 | × | – | – |
| *can* | .164 | × | – | – | .077 | × | – | – | | .985 | ✓ | .660 | × | .003 | × | – | – |
| *significantly* | .476 | × | – | – | .121 | × | – | – | | .919 | ✓ | .461 | × | .452 | × | – | – |
| *improve* | .732 | ✓ | .322 | × | .252 | × | – | – | | .836 | ✓ | .987 | ✓ | .128 | × | – | – |
| *sleep* | .758 | ✓ | .859 | ✓ | .028 | × | – | – | | .982 | ✓ | .794 | ✓ | .774 | ✓ | .029 | × |
| *quality* | .979 | ✓ | .394 | × | .006 | × | – | – | | .813 | ✓ | .319 | × | .005 | × | – | – |
| . | .637 | × | – | – | .009 | × | – | – | | .133 | × | – | – | .025 | × | – | – |
| *Going* | .776 | ✓ | .434 | × | .039 | × | – | – | | .882 | ✓ | .699 | × | .120 | × | – | – |
| *to* | .956 | ✓ | .984 | ✓ | .006 | × | – | – | | .539 | × | – | – | .064 | × | – | – |
| *bed* | .988 | ✓ | .822 | ✓ | .205 | × | – | – | | .049 | × | – | – | .071 | × | – | – |
| *and* | .384 | × | – | – | .027 | × | – | – | | .129 | × | – | – | .038 | × | – | – |
| *waking* | .997 | ✓ | .960 | ✓ | .078 | × | – | – | | .817 | ✓ | .909 | ✓ | .009 | × | – | – |
| *up* | .751 | ✓ | .964 | ✓ | .032 | × | – | – | | .493 | × | – | – | .772 | ✓ | .058 | × |
| ... | | | | | | | | | | | | | | | | | |
| *Keeping* | .516 | × | – | – | .029 | × | – | – | | .412 | × | – | – | .011 | × | – | – |
| *the* | .861 | ✓ | .979 | ✓ | .039 | × | – | – | | .848 | ✓ | .142 | × | .046 | × | – | – |
| *bedroom* | .358 | × | – | – | .012 | × | – | – | | .776 | ✓ | .409 | × | .001 | × | – | – |
| *dark* | .996 | ✓ | .857 | ✓ | .003 | × | – | – | | .934 | ✓ | .902 | ✓ | .555 | ✓ | .004 | × |
| , | .744 | ✓ | .652 | × | .014 | × | – | – | | .976 | ✓ | .098 | × | .003 | × | – | – |
| *quiet* | .371 | × | – | – | .011 | × | – | – | | .776 | ✓ | .977 | ✓ | .021 | × | – | – |
| , | .884 | ✓ | .848 | ✓ | .113 | × | – | – | | .892 | ✓ | .886 | ✓ | .008 | × | – | – |
| *and* | .856 | ✓ | .983 | ✓ | .619 | ✓ | .005 | × | | .399 | × | – | – | .014 | × | – | – |
| *cool* | .856 | ✓ | .934 | ✓ | .002 | × | – | – | | .229 | × | – | – | .878 | ✓ | .079 | × |
| *supports* | .887 | ✓ | .849 | ✓ | .025 | × | – | – | | .444 | × | – | – | .106 | × | – | – |
| *better* | .883 | ✓ | .503 | × | .004 | × | – | – | | .020 | × | – | – | .127 | × | – | – |
| *sleep* | .793 | ✓ | .949 | ✓ | .012 | × | – | – | | .919 | ✓ | .329 | × | .007 | × | – | – |
| *quality* | .947 | ✓ | .969 | ✓ | .105 | × | – | – | | .804 | ✓ | .878 | ✓ | .010 | × | – | – |
| . | .990 | ✓ | .985 | ✓ | .005 | × | – | – | | .971 | ✓ | .762 | ✓ | .0190 | × | – | – |

Table 6: Personal Sample 3: *"What can help me sleep better if I wake up at 3 a.m. because of work stress?"*'s Impact Scores (IS) and Calibrator Decision (CD) across 8 layers. (× denotes that CD is pruning, ✓ denotes that CD is preserving.)

| Tokens | Layer3 | | | | Layer4 | | | | Layer5 | | | | ⇒ skip ⇒ | Layer20 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | | IS-1 | CD-1 | IS-2 | CD-2 |
| *If* | .012 | × | – | – | .007 | × | – | – | .224 | × | – | – | | 688 | ✓ | .870 | ✓ |
| *you* | 027 | × | – | – | .128 | × | – | – | .136 | × | – | – | | .999 | ✓ | .526 | × |
| *wake* | .033 | × | – | – | .072 | × | – | – | .919 | ✓ | .005 | × | | .989 | ✓ | .788 | ✓ |
| *up* | .304 | × | – | – | .054 | × | – | – | .983 | ✓ | .002 | × | | .969 | ✓ | .206 | × |
| *at* | .001 | × | – | – | .506 | ✓ | .627 | ✓ | .149 | × | – | – | | .993 | ✓ | .985 | ✓ |
| *3* | .583 | ✓ | .877 | ✓ | .002 | × | – | – | .649 | ✓ | .946 | ✓ | | .993 | ✓ | .005 | ✓ |
| *a* | .317 | × | – | – | .044 | × | – | – | .179 | × | – | – | | .978 | ✓ | .356 | × |
| *.m* | .819 | ✓ | .883 | ✓ | .997 | ✓ | .383 | ✓ | .999 | ✓ | .962 | ✓ | | .959 | ✓ | .825 | ✓ |
| . | .032 | × | – | – | .003 | × | – | – | .008 | × | – | – | | .998 | ✓ | .243 | × |
| , | .036 | × | – | – | .001 | × | – | – | .003 | × | – | – | | .763 | ✓ | .812 | ✓ |
| *try* | .037 | × | – | – | .018 | × | – | – | .003 | × | – | – | | .459 | × | – | – |
| *claming* | .032 | × | – | – | .002 | × | – | – | .116 | × | – | – | | .339 | × | – | – |
| *your* | .03+ | × | – | – | .003 | × | – | – | .002 | × | – | – | | .844 | ✓ | .867 | ✓ |
| *body* | .033 | × | – | – | .001 | × | – | – | .002 | × | – | – | | .015 | × | – | – |
| . | .035 | × | – | – | .001 | × | – | – | .001 | × | – | – | | .668 | ✓ | .215 | × |
| ... | | | | | | | | | | | | | | | | | |
| *Keep* | .032 | × | – | – | .019 | × | – | – | .001 | × | – | – | | .809 | ✓ | .872 | ✓ |
| *lights* | .018 | × | – | – | .004 | × | – | – | .003 | × | – | – | | .021 | × | – | – |
| *dim* | .036 | × | – | – | .001 | × | – | – | .076 | × | – | – | | .063 | × | – | – |
| *and* | .029 | × | – | – | .001 | × | – | – | .001 | × | – | – | | .191 | × | – | – |
| *stay* | .031 | × | – | – | .001 | × | – | – | .002 | × | – | – | | .456 | × | – | – |
| *away* | .034 | × | – | – | .001 | × | – | – | .006 | × | – | – | | .442 | × | – | – |
| *from* | .039 | × | – | – | .006 | × | – | – | .002 | × | – | – | | .413 | × | – | – |
| *your* | .036 | × | – | – | .013 | × | – | – | .020 | × | – | – | | .068 | × | – | – |
| *phone* | .029 | × | – | – | .001 | × | – | – | .002 | × | – | – | | .948 | ✓ | .164 | × |
| *and* | .029 | × | – | – | .001 | × | – | – | .001 | × | – | – | | .443 | × | – | – |
| *practicing* | .031 | × | – | – | .002 | × | – | – | .001 | × | – | – | | .061 | × | – | – |
| *relaxation* | .039 | × | – | – | .001 | × | – | – | .118 | × | – | – | | .316 | × | – | – |
| *techniques* | .033 | × | – | – | .052 | × | – | – | .001 | × | – | – | | .532 | × | – | – |
| . | .041 | × | – | – | .005 | × | – | – | .018 | × | – | – | | .926 | ✓ | .858 | ✓ |

| Tokens | Layer21 | | | | Layer22 | | | | ⇒ skip ⇒ | Layer35 | | | | Layer36 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 | | IS-1 | CD-1 | IS-2 | CD-2 | IS-1 | CD-1 | IS-2 | CD-2 |
| *If* | .523 | × | – | – | .523 | × | – | – | | .819 | ✓ | .919 | ✓ | .954 | ✓ | .882 | ✓ |
| *you* | .429 | × | – | – | .998 | ✓ | .326 | × | | .888 | ✓ | .960 | ✓ | .975 | ✓ | .902 | ✓ |
| *wake* | .082 | × | – | – | .071 | × | – | – | | .981 | ✓ | .988 | ✓ | .998 | ✓ | .909 | ✓ |
| *up* | .773 | ✓ | .985 | ✓ | .987 | ✓ | .367 | × | | .897 | ✓ | .993 | ✓ | .864 | ✓ | .441 | × |
| *at* | .901 | ✓ | .946 | ✓ | .136 | × | – | – | | .812 | ✓ | .994 | ✓ | .954 | ✓ | .134 | × |
| *3* | .925 | ✓ | .990 | ✓ | .963 | ✓ | .111 | × | | .728 | ✓ | .997 | ✓ | .975 | ✓ | .709 | ✓ |
| *a* | .253 | × | – | – | .531 | × | – | – | | .944 | ✓ | .975 | ✓ | .857 | ✓ | .495 | × |
| *.m* | .870 | ✓ | .961 | ✓ | .954 | ✓ | .294 | × | | .897 | ✓ | .989 | ✓ | .926 | ✓ | .991 | ✓ |
| . | .003 | × | – | – | .993 | ✓ | .851 | ✓ | | .929 | ✓ | .991 | ✓ | .961 | ✓ | .992 | ✓ |
| , | .854 | ✓ | .888 | ✓ | .289 | × | – | – | | .643 | ✓ | .971 | ✓ | .975 | ✓ | .609 | × |
| *try* | .905 | ✓ | .689 | × | .944 | ✓ | .868 | ✓ | | .952 | ✓ | .614 | × | .995 | ✓ | .996 | ✓ |
| *claming* | .993 | ✓ | .997 | ✓ | .884 | ✓ | .548 | × | | .936 | ✓ | .982 | ✓ | .831 | ✓ | .295 | × |
| *your* | .999 | ✓ | .402 | × | .021 | × | – | – | | .997 | ✓ | .964 | ✓ | .736 | ✓ | .896 | ✓ |
| *body* | .963 | ✓ | .993 | ✓ | .542 | × | – | – | | .977 | ✓ | .973 | ✓ | .985 | ✓ | .567 | × |
| . | .947 | ✓ | .973 | ✓ | .128 | × | – | – | | .973 | ✓ | .816 | ✓ | .424 | × | – | – |
| ... | | | | | | | | | | | | | | | | | |
| *Keep* | .995 | ✓ | .943 | ✓ | .039 | × | – | – | | .948 | ✓ | .701 | ✓ | .791 | ✓ | .927 | ✓ |
| *lights* | .763 | ✓ | .988 | ✓ | .184 | × | – | – | | .996 | ✓ | .997 | ✓ | .950 | ✓ | .693 | × |
| *dim* | .998 | ✓ | .909 | ✓ | .153 | × | – | – | | .972 | ✓ | .979 | ✓ | .915 | ✓ | .955 | ✓ |
| *and* | .666 | ✓ | .537 | × | .868 | ✓ | .086 | × | | .895 | ✓ | .991 | ✓ | .872 | ✓ | .738 | ✓ |
| *stay* | .865 | ✓ | .718 | ✓ | .253 | × | – | – | | .990 | ✓ | .861 | ✓ | .997 | ✓ | .552 | × |
| *away* | .963 | ✓ | .599 | × | .965 | ✓ | .327 | × | | .952 | ✓ | .998 | ✓ | .955 | ✓ | .801 | ✓ |
| *from* | .983 | ✓ | .935 | ✓ | .124 | × | – | – | | .998 | ✓ | .971 | ✓ | .959 | ✓ | .971 | ✓ |
| *your* | .986 | ✓ | .649 | × | .392 | × | – | – | | .872 | ✓ | .778 | ✓ | .297 | × | – | – |
| *phone* | .988 | ✓ | .941 | ✓ | .568 | × | – | – | | .825 | ✓ | .948 | ✓ | .757 | ✓ | .269 | × |
| *and* | .955 | ✓ | .944 | ✓ | .795 | ✓ | .125 | × | | .342 | × | – | – | .991 | ✓ | .928 | ✓ |
| *practicing* | .853 | ✓ | .915 | ✓ | .947 | ✓ | .983 | ✓ | | .838 | ✓ | .708 | ✓ | .954 | ✓ | .644 | × |
| *relaxation* | .970 | ✓ | .948 | ✓ | .009 | × | – | – | | .859 | ✓ | .942 | ✓ | .384 | × | .486 | × |
| *techniques* | .965 | ✓ | .940 | ✓ | .564 | × | – | – | | .882 | ✓ | .993 | ✓ | .997 | ✓ | .945 | ✓ |
| . | .940 | ✓ | .809 | ✓ | .277 | × | – | – | | .989 | ✓ | .793 | ✓ | .915 | ✓ | .903 | ✓ |