

Latent Task Embedding: Geometric Expansion for Single-Path Multi-Task PEFT

Junhyun Lee*, Seoha Song*, Dhananjaya Gowda

Samsung Research

{junhyun8.lee, seoha.song, d.gowda}@samsung.com

Abstract

Parameter-efficient fine-tuning (PEFT) typically relies on additive weight perturbation. However, in dynamic deployment environments like multi-tenant cloud serving or mobile edge devices, additive methods force a dilemma: merge weights and lose task-switching flexibility, or remain unmerged and suffer severe parallel-branching latency. To break this bottleneck, we propose **Latent Task Embedding (LatTE)**, which shifts the PEFT design axis to *geometric expansion*. Instead of perturbing frozen parameters, LatTE concatenates a task-specific auxiliary embedding to the input and expands the corresponding weight matrices. This structural fusion ensures the forward pass remains a single, optimized matrix multiplication, bypassing the routing overhead of unmerged adapters. Furthermore, LatTE natively supports router-free, intra-batch multi-tasking by isolating tasks into distinct subspaces controlled via binary masking. Extensive evaluations across 1B to 14B parameter models demonstrate that LatTE matches highly optimized baselines like LoRA, while raising throughput by $1.7\times$ over routed multi-adapter LoRA under heterogeneous-task batching.

1 Introduction

Parameter-efficient fine-tuning (PEFT) has established itself as the standard for adapting large language models (LLMs) to downstream tasks. By updating only a fraction of the total parameters, methods such as LoRA (Hu et al., 2022) have made large-scale adaptation computationally tractable. However, nearly all existing weight-based PEFT methods share a fundamental geometric perspective: they view adaptation as an *additive perturbation* to frozen base weights (e.g., adding a low-rank delta ΔW). While empirically successful, this additive paradigm imposes severe structural constraints during real-world deployment.

When serving multiple tasks, additive methods face an unavoidable deployment dilemma. Adapter weights can be mathematically merged into the base parameters to achieve zero inference overhead, but this locks the model into a static, single-task state, precluding the rapid task switching required in modern applications. Conversely, keeping the weights unmerged preserves dynamic flexibility but introduces significant computational overhead, as the model must compute and route parallel adaptation branches.

In highly parallelized multi-tenant serving environments, this routing overhead destroys batching efficiency. Furthermore, on memory-constrained edge devices, such as mobile NPUs and CPUs where memory bandwidth and power consumption are strict bottlenecks, the latency penalty of parallel branch computation is often prohibitive.

In this work, we ask: *Is additive perturbation the only viable geometric operation for PEFT?* We propose **Latent Task Embedding (LatTE)**, which explores a fundamentally different design axis: *geometric expansion* of the hidden representation space. Instead of perturbing the parameter space, LatTE concatenates a small, task-specific auxiliary embedding to the input and extends the corresponding weight matrices to process this expanded representation.

Critically, because the adaptation is fused into the expanded dimension, the forward pass remains a single, optimized matrix multiplication. LatTE preserves the inference structure of the base model *without requiring weight merging*. This architectural shift offers distinct structural properties absent from additive PEFT. Because adaptation occurs in auxiliary dimensions, multiple tasks occupy disjoint subspaces of the expanded representation. This enables seamless intra-batch multi-tasking: heterogeneous requests can be processed in a single forward pass via simple masking, removing the routing logic and reducing the latency penalty that

* Equal contribution

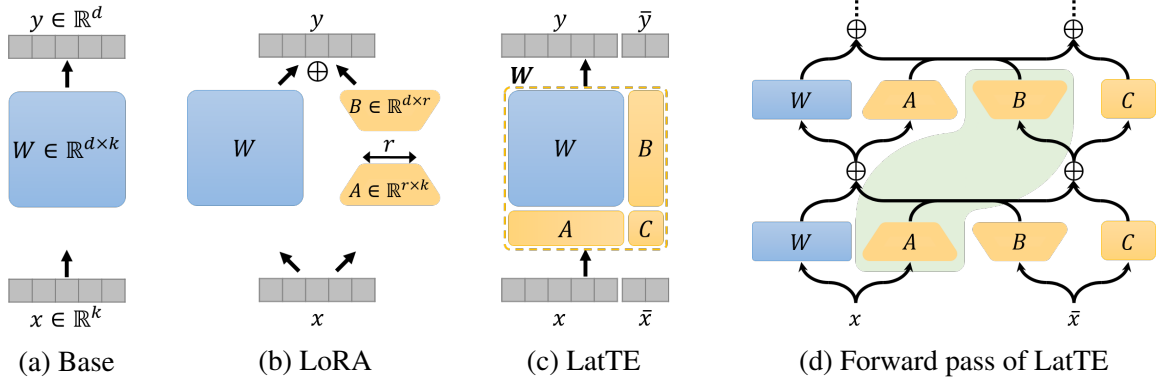


Figure 1: (a-c) The comparison of the forward pass through a single weight for the base, LoRA, and LatTE model. While the A and B matrices in LoRA are expressed conventionally as trapezoids to emphasize their low ranks, they are identical in the matrix dimensions with the A and B in LatTE. (d) The forward pass a MLP model with LatTE. We have omitted the non-linearity between the layers for simplicity. The shaded region can be conceptually regarded as a single LoRA unit.

unmerged additive methods incur.

We position LatTE not as an attempt to artificially inflate accuracy beyond LoRA, which already approaches full fine-tuning performance, but as a structurally distinct alternative designed to break the deployment bottleneck. LatTE offers a new point on the accuracy-efficiency Pareto frontier, preserving state-of-the-art accuracy while fundamentally optimizing the computational graph for dynamic deployment.

Our contributions are threefold:

1. We identify geometric expansion as a complementary design axis to additive perturbation in PEFT, formalizing the structural and hardware execution differences between these approaches.
2. We propose LatTE, a novel PEFT method that expands latent embeddings to create auxiliary adaptation subspaces, achieving *single-path* inference – one matrix multiplication per layer – without weight merging.
3. We demonstrate through extensive evaluations on scales ranging from 1B to 14B parameters that LatTE matches the accuracy of strong PEFT baselines. Furthermore, our hardware profiling confirms that LatTE avoids the $\sim 25\%$ latency penalty of unmerged LoRA, and that under heterogeneous-task batching it reaches $1.7\times$ the throughput of routed multi-adapter LoRA.

2 Related Works

Parameter-Efficient Fine-Tuning (PEFT).

PEFT methods update only a small fraction of the model’s parameters during fine-tuning while keeping the rest frozen. This enables adaptation to new tasks with minimal computational overhead. We broadly categorize existing PEFT methods into the following classes:

- **Module-based methods** introduce additional trainable modules (adapters) to the model architecture. The adapters mostly consist of a down- and up-projection with a nonlinearity in between. There are serial adapter (Houlsby et al., 2019; Pfeiffer et al., 2021) and parallel adapter (He et al., 2022) based on how the adapter is attached to the base module. CIAT (Zhu et al., 2021) and CoDA (Lei et al., 2023) are also variants of the parallel adapter.
- **Weight-based methods** aim to efficiently update the weights of the model, often acting as a direct approximation of full fine-tuning (FFT). The updates can be additive or multiplicative. LoRA (Hu et al., 2022) and its variants (Zhang et al., 2023b; Liu et al., 2024a; Kopiczko et al., 2024; Li et al., 2024) are additive methods. Multiplicative-update method includes OFT (Qiu et al., 2023), BOFT (Liu et al., 2024b), and HRA (Yuan et al., 2024). Selective fine-tuning is another group of method which updates a subset of weight elements, often by masking (Guo et al., 2021; Sung et al., 2021; Lawton et al., 2023).

- **Representation-based methods** use representations as a tool for fine-tuning. Soft-prompt methods, where learnable embeddings are prefixed to the prompt, include prefix-tuning (Li and Liang, 2021), prompt-tuning (Lester et al., 2021), and p-tuning (Liu et al., 2022b). Another group modifies or edits intermediate representations, such as (IA)³ (Liu et al., 2022a), SSF (Lian et al., 2022), and ReFT (Wu et al., 2024c).

Network Expansion vs. Adaptation. While modifying a network’s architecture is a well-known technique, prior expansion methods like Net2Net (Chen et al., 2015) and LLaMA-Pro (Wu et al., 2024a) expand model depth or width to increase overall capacity for continued pre-training or full-parameter tuning. In contrast, LatTE keeps the base model entirely frozen, utilizing controlled dimension expansion specifically as a mechanism for parameter-efficient, multi-task adaptation.

Inference-Efficient PEFT. Recent works have recognized the need for reducing inference-time overhead in PEFT. SPLoRA (Hedegaard et al., 2024) and CA-LoRA (Zhao et al., 2024) combine LoRA with pruning or quantization for faster inference. However, these are often engineering solutions that sacrifice precision or require specialized kernels. Zero-latency PEFT methods like PaFi and HiWi (Liao et al., 2023) partially update the original model as in FFT. zFLoRA (Gowda et al., 2025) fuses low-rank adapter branches into the base computation, and GAWS (Alsuradi et al., 2026) traces unmerged-adapter latency to segmented kernel launches and narrows it with a grouped weight-sharing design.

Multi-Adapter and Multi-Task Adaptation. Scenarios involving multiple downstream tasks often demand flexible and composable fine-tuning strategies. MAD-X (Pfeiffer et al., 2020) and Adapter Fusion (Pfeiffer et al., 2021) allow for combining multiple adapters, yet require additional merging logic. Mixture of Expert (MoE) style combinations of multiple task LoRAs (Wu et al., 2024b; Xu et al., 2025) typically introduce routers or gating networks, adding both parameter count and runtime logic. LatTE supports multi-tasking intrinsically: different tasks are assigned specific slots in the auxiliary embedding space, controlled via simple masking. This allows for deterministic, router-free multi-adapter inference that scales efficiently with

the number of tasks.

3 Our Method

We propose LatTE, a fine-tuning method in which a task-specific auxiliary embedding is concatenated to the original token embeddings. This increases the embedding dimensionality, which correspondingly expands the associated weight matrices. The fine-tuned knowledge is thus stored in the additional weight parameters introduced by this expansion, and the interaction with the original forward pass is mediated by the auxiliary embedding. Architecturally, LatTE resembles LoRA and other weight-based PEFT methods, as it modifies individual weights without altering the model’s overall structure (see Figure 1). However, a unique feature of LatTE is that the forward pass through one weight is calculated by a single matrix multiplication while keeping the original weight frozen. This enables inference-time performance comparable to that of the original model, making LatTE both efficient and scalable.

The comparison between the base model, LoRA, and LatTE is illustrated in Figure 1 (a-c), which shows the forward pass through a single weight matrix $W \in \mathbb{R}^{d \times k}$. Given an input embedding $x \in \mathbb{R}^k$, the base model computes the output $y \in \mathbb{R}^d$ as $y = Wx$. In LoRA, task-specific information is introduced via a low-rank adapter, yielding: $y = Wx + BAx$, where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$, and r is the low-rank dimension. In LatTE, the input embedding is concatenated with an auxiliary embedding $\bar{x} \in \mathbb{R}^r$ to form $[x; \bar{x}] \in \mathbb{R}^{k+r}$. The forward pass is computed in the expanded embedding space:

$$\begin{bmatrix} y \\ \bar{y} \end{bmatrix} = \begin{bmatrix} W & B \\ A & C \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix}, \quad (1)$$

where A , B , and $C \in \mathbb{R}^{r \times r}$ are trainable matrices. For convenience, we denote the expanded vectors and matrix as $\mathbf{x} = \begin{bmatrix} x \\ \bar{x} \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y \\ \bar{y} \end{bmatrix}$, and

$\mathbf{W} = \begin{bmatrix} W & B \\ A & C \end{bmatrix}$, so that Eq. (1) simplifies to the familiar linear form $\mathbf{y} = \mathbf{W}\mathbf{x}$. In Figure 1 (c), the block matrix $\mathbf{W}$ is highlighted with a yellow dashed outline. We deliberately set the auxiliary embedding dimension r to match the low-rank dimension used in LoRA, allowing for a fair parameter comparison. LoRA introduces $(d + k)r$ additional parameters per weight matrix, while LatTE

adds $(d + k + r)r$: exactly r^2 more per adapted matrix, contributed by C . This term is small under the common assumption $r \ll \min(d, k)$ (at $r = 16$, $d = k = 3584$ it is 0.2% of the adapter budget). LatTE’s advantage lies in the computation graph – a single matrix multiplication per layer – with a marginal parameter cost, and Section 4.3 shows that removing C , which matches LoRA’s count exactly, loses no accuracy.

This block matrix formulation fundamentally alters the physical computation graph compared to additive adapters. While LoRA applies a localized low-rank update at each individual layer, LatTE effectively distributes a single low-rank transformation across two consecutive layers “off-phase.” The auxiliary representation $\bar{x}$ acts as a continuous working memory: instead of computing a delta and discarding the state, the first expanded weight matrix creates the auxiliary state, which is then carried forward, compressed, and interacted with in the subsequent layer’s expanded weight matrix. This structural distribution across network depth is what enables the single-path computation graph. The recombination into a single low-rank update is exact only without an intervening nonlinearity, so we treat the off-phase reading as design intuition rather than an identity.

We define f_{in} and f_{out} applied once per forward pass:

$$f_{\text{in}}(x) = [x; c], \quad f_{\text{out}}([y; \bar{y}]) = y + B\bar{y},$$

where c is a learnable constant vector. The matrix B in f_{out} is a separate trainable parameter. We initialize A with Kaiming-uniform initialization and B, C with zeros, ensuring the initial forward pass exactly matches the base model. This specific initialization strategy was chosen to directly mirror the default configuration of LoRA in the Huggingface PEFT library, ensuring a strictly fair empirical comparison without requiring extensive hyperparameter searches to account for initialization sensitivity.

3.1 Application to the multi-layer perceptron

With the LatTE building block in place, we now apply it to a multilayer perceptron (MLP) network. Consider an L -layer MLP with weight matrices W_i for $1 \leq i \leq L$ and an activation function σ . Given an input embedding x , the MLP computes the output as:

$$y = W_L \cdot \sigma(W_{L-1} \cdot \sigma(\cdots \sigma(W_1 x))).$$

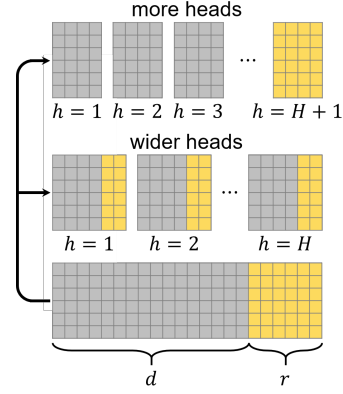


Figure 2: The two strategies on allocating the additional dimension of Wx (yellow) to attention heads. Here, the vertical dimension represents the context length.

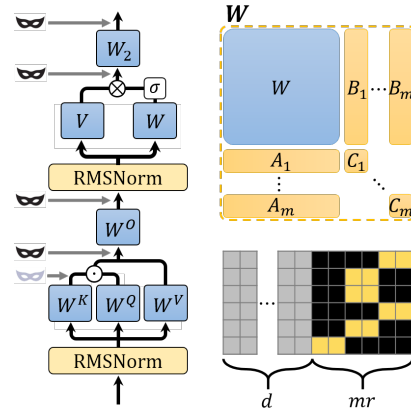


Figure 3: Masking (black squares) in intra-batch multi-adapter scenario. The mask symbol indicates where masking should be applied for Transformers.

To incorporate LatTE, we replace each W_i with its expanded counterpart $\mathbf{W}_i$ and insert the expansion and compression functions f_{in} and f_{out} at the input and output, respectively. The input embedding is first expanded from a d -dimensional vector to a $(d + r)$ -dimensional one by f_{in} , then propagated through the network using the sequence of $\mathbf{W}_i$ matrices. The final embedding is compressed back to a d -dimensional output by f_{out} . The resulting output becomes:

$$y = f_{\text{out}}(\mathbf{W}_L \cdot \sigma(\mathbf{W}_{L-1} \cdot \sigma(\cdots \sigma(\mathbf{W}_1 f_{\text{in}}(x)))). \quad (2)$$

The feed-forward network (FFN) block in a Transformer architecture (Vaswani et al., 2017) is typically a two-layer MLP of the form:

$$\text{FFN}_{\sigma}(x, W_1, W_2) = W_2 \cdot \sigma(W_1 x).$$

Modern LLMs, including those used in our experiments (Grattafiori et al., 2024; Yang et al.,

2024), often use the SwiGLU activation function (Shazeer, 2020), a variant of the Gated Linear Unit (GLU) (Dauphin et al., 2017) based on Swish (Ramachandran et al., 2017). A single FFN using SwiGLU can be expressed as:

$$\begin{aligned} \text{FFN}_{\text{SwiGLU}}(x, W, V, W_2) \\ = W_2 \cdot (\text{Swish}_1(Wx) \otimes Vx), \end{aligned}$$

where $\text{Swish}_\beta(x) = x \cdot \text{sigmoid}(\beta x)$ and $\otimes$ denotes the element-wise (Hadamard) product. Applying LatTE to this FFN block involves replacing each weight matrix with its expanded form. The resulting forward pass becomes:

$$\begin{aligned} \text{FFN}_{\text{SwiGLU}}(\mathbf{x}, \mathbf{W}, \mathbf{V}, \mathbf{W}_2) \\ = \mathbf{W}_2 \cdot (\text{Swish}_1(\mathbf{W}\mathbf{x}) \otimes \mathbf{V}\mathbf{x}). \quad (3) \end{aligned}$$

3.2 Application to the attention module

Unlike MLP layers – where applying LatTE is straightforward and leaves little room for design choices beyond f_{in} and f_{out} – the attention module presents several implementation options. The core strategy remains the same: expand the embedding dimension and replace the weight matrices with their extended counterparts. However, the key challenge lies in how to allocate the auxiliary embedding dimensions across the attention heads.

Assume a standard multi-head attention (MHA) mechanism with H heads, where the per-head dimension is $d_H = d/H$. The usual output of an MHA layer can be written as:

$$\sum_{h=1}^H (W_h^O)^\top W_h^V x \cdot \text{softmax} \left((W_h^K x)^\top W_h^Q x \right),$$

where W_h denotes the h -th row-wise split sub-matrix of a full weight matrix W , i.e., $W_h = W[(h-1)d_H : hd_H, :]$. When the embedding dimension increases from d to $d+r$, we must adjust the MHA configuration so that $d_{H'} = (d+r)/H'$ holds. Two simple approaches can achieve this:

1. **More heads:** We introduce additional heads to accommodate the extra dimensions. Specifically, we add $\lceil r/d_H \rceil$ new heads solely for the auxiliary representation. This isolates the task embedding from the original heads.

2. **Wider heads:** We distribute the auxiliary dimension across the existing H heads by increasing the head dimension d_H . To ensure compatibility with rotary positional embeddings (RoPE) (Su et al., 2024), which rotate pairs of elements, the

expansion must be even. Thus, we increase each head’s dimension by $2\lceil r/(2H) \rceil$.

If r is not divisible by d_H (the more-heads case) or by $2H$ (the wider-heads case), the unmatched dimensions are zero-padded. Figure 2 provides a schematic of these two integration strategies.

With the application of LatTE to both FFN and attention layers explained, we are now ready to implement it in Transformer-based LLMs. In diffusion models, PEFT is typically applied to the text encoder and the cross-attention modules within the U-Net architecture (Ruiz et al., 2023; Zhang et al., 2023a; Mou et al., 2024). This means that the same recipe used for LLMs can be applied to diffusion models as well.

3.3 Comparison with LoRA

We provide theoretical context for LatTE by drawing comparisons with LoRA. The intuition behind LoRA stems from the observation that the intrinsic dimension of many NLP tasks is significantly lower than the dimension of large pretrained models (Aghajanyan et al., 2021). Building on this, LoRA hypothesizes that the required weight updates for task adaptation also lie in a low-rank subspace (Hu et al., 2022). However, the argument in Aghajanyan et al. (2021) supports a low intrinsic dimension as a *necessary*, but not *sufficient*, condition for low-rank adaptation to be effective.

Interestingly, LatTE can also be interpreted as a composition of low-rank updates. As illustrated in Figure 1 (d), when examining the green-shaded information flow over two layers (omitting activation for simplicity), the input embedding x is transformed by a low-rank operation BA and reintegrated with the base embedding. Thus, while LoRA applies a single low-rank update per layer, LatTE effectively performs two low-rank transformations across two layers. This structural similarity suggests that the same theoretical motivation underlying LoRA – namely, that task-specific transformations can be captured in low-rank subspaces – also supports the design of LatTE.

Beyond the qualitative arguments presented above, we provide two formal results establishing cases where the expressive power of LatTE is equivalent to that of LoRA. Both are stated for *simplified* settings, and therefore *lower-bound* the expressivity of the deployed configuration.

Theorem 1 For linear models, the minimum low-rank dimension required for adapter models to exactly recover the FFT target is identical for both

Model	Fintuning	Params (%)	Accuracy								
			OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.
Qwen2.5-3B	Prompt	0.0021	77.04	80.46	80.60	73.54	69.95	62.63	93.82	82.98	77.63
	Serial	0.7546	86.70	86.86	85.81	79.20	91.99	79.99	93.39	83.53	85.93
	Parallel	0.7546	85.90	87.08	85.77	79.77	92.26	82.75	94.03	83.32	86.36
	OFT	0.7025	87.50	87.58	85.69	78.97	92.36	82.44	93.98	83.49	86.50
	BOFT	0.7025	87.90	87.74	86.29	79.86	92.01	82.06	93.81	83.21	86.61
	LoRA	0.7546	87.15	87.50	86.28	79.38	92.46	83.11	93.61	83.29	86.60
	LatTE-m	0.7573	86.04	87.60	85.58	79.02	91.98	84.35	93.60	83.12	86.41
	LatTE-w	0.7573	86.80	87.97	85.73	78.67	92.20	83.94	94.14	83.55	86.63
Qwen2.5-7B	Prompt	0.0015	86.50	85.85	86.04	76.67	81.59	68.01	96.20	89.57	83.81
	Serial	0.5273	92.84	89.45	89.40	81.88	94.03	85.83	96.19	89.18	89.85
	Parallel	0.5273	92.04	89.30	89.15	81.26	94.01	87.83	96.50	89.76	89.98
	OFT	0.4943	91.84	89.04	90.08	80.99	93.84	85.65	96.68	89.80	89.74
	BOFT	0.4943	92.35	89.46	89.67	79.94	94.14	86.15	96.39	89.69	89.72
	LoRA	0.5303	92.55	89.59	90.22	81.94	93.84	86.76	96.57	90.29	90.22
	LatTE-m	0.5317	91.84	89.81	90.08	81.28	94.52	87.49	96.74	90.76	90.32
	LatTE-w	0.5317	92.60	89.49	90.59	80.73	94.67	87.86	97.00	90.25	90.40
Llama-3.2-3B	Prompt	0.0031	71.64	79.30	77.20	67.41	62.12	60.10	85.47	72.54	71.97
	Serial	0.7550	82.10	86.73	84.22	77.19	89.61	80.84	88.90	77.52	83.39
	Parallel	0.7550	82.10	87.50	83.42	77.79	90.66	82.57	89.27	78.65	83.99
	OFT	0.7666	84.84	88.44	85.01	79.17	90.91	82.89	91.25	79.82	85.29
	BOFT	0.7666	86.30	88.76	85.55	78.51	91.32	81.77	90.88	78.82	85.24
	LoRA	0.7568	84.44	88.56	85.61	77.94	91.41	83.78	89.82	78.07	84.95
	LatTE-m	0.7599	85.70	88.35	85.36	78.39	91.42	84.89	89.98	79.24	85.42
	LatTE-w	0.7599	86.40	88.74	85.23	79.21	91.15	84.47	90.49	79.76	85.68
Llama-3.1-8B	Prompt	0.0016	82.35	86.12	84.48	77.38	83.29	72.26	92.56	80.87	82.41
	Serial	0.4570	89.70	88.73	87.81	80.45	93.61	84.35	94.00	83.92	87.82
	Parallel	0.4570	89.75	88.87	88.15	81.35	94.10	86.03	93.80	84.91	88.37
	OFT	0.4492	89.00	89.30	88.45	81.54	93.79	84.02	94.43	84.68	88.15
	BOFT	0.4492	88.90	89.72	88.75	80.81	93.58	84.85	94.32	83.94	88.11
	LoRA	0.4570	88.90	89.09	88.85	80.92	94.31	87.23	93.81	84.85	88.50
	LatTE-m	0.4585	90.15	89.63	88.80	80.80	94.39	88.38	94.14	84.45	88.84
	LatTE-w	0.4585	90.60	89.33	88.87	80.28	94.25	86.68	94.73	84.47	88.65

Table 1: Accuracy results on the commonsense QA benchmark, with eight diverse reasoning tasks. Adapters are applied to all layers.

LoRA and LatTE.

Theorem 2. Any attention matrix produced by LoRA can be represented as an attention matrix from LatTE, given that $\bar{x}$ is a linear transform of x .

They remain simplified expressivity evidence, where the evidence for the full nonlinear architecture is empirical (Tables 1 and 2). Proofs are provided in Appendix A.

3.4 Intra-Batch Multi-Task Masking

Beyond single-task inference speed, LatTE natively supports efficient intra-batch multi-tasking without routing logic. In dynamic serving environments, it is highly challenging to process multiple adapters within a single batch (e.g., Task A for User 1, Task B for User 2) using standard unmerged methods.

Because LatTE’s structure expands the representation space rather than perturbing the base weights, distinct tasks can be assigned distinct, disjoint slots in the auxiliary embedding space (e.g., allocating $m \times r$ auxiliary dimensions for m distinct tasks). A batch containing heterogeneous requests can then

be processed in a single forward pass by applying a binary mask to the auxiliary attention heads and MLP blocks. This task mask ensures that input tokens only interact with their designated auxiliary weights. This architectural property allows deterministic, router-free multi-tasking, avoiding the gating networks and per-task routing computation required by standard multi-adapter architectures. Because the mask zeroes every interaction outside a request’s own auxiliary slot, a masked heterogeneous batch is numerically identical to running each request in a single-task batch, up to floating-point error; we verified this empirically, so the accuracies in Tables 1 and 2 transfer unchanged to the multi-task setting.

4 Experiments

We evaluate LatTE across three distinct domains to demonstrate its versatility: general commonsense reasoning, complex chain-of-thought (CoT) reasoning, and generative capabilities in text-to-image

Model	Fintuning	Params (%)	Accuracy					
			AOuA	GPQA	MATH	GSM8K	SVAMP	ave.
Qwen2.5-1.5B	Prompt	0.0016	37.80	24.24	37.80	54.80	72.33	45.39
	Serial	0.2982	42.13	25.25	42.40	65.00	71.67	49.29
	Parallel	0.2982	50.79	29.29	45.00	63.00	70.67	51.75
	OFT	0.2847	40.09	30.81	39.40	52.40	64.00	45.34
	BOFT	0.2847	41.34	28.79	39.60	54.00	68.00	46.35
	LoRA	0.2991	49.61	31.82	42.40	61.80	74.00	51.92
	LatTE-m	0.3003	50.79	31.31	40.80	62.60	73.67	51.83
	LatTE-w	0.3003	49.61	30.30	43.20	63.80	73.00	51.98
Qwen2.5-3B	Prompt	0.0011	46.85	25.76	49.20	67.00	70.67	51.89
	Serial	0.1877	49.61	26.26	52.80	76.40	78.67	56.75
	Parallel	0.1877	48.43	28.79	49.60	73.00	83.33	56.63
	OFT	0.1821	43.31	22.73	49.00	65.40	72.67	50.62
	BOFT	0.1821	45.67	28.79	48.20	65.60	71.00	51.85
	LoRA	0.1886	45.28	28.28	51.40	73.60	81.33	55.98
	LatTE-m	0.1894	49.61	30.30	53.20	71.00	80.33	56.89
	LatTE-w	0.1894	48.03	26.26	52.40	73.80	78.33	55.77
Llama-3.2-1B	Prompt	0.0027	21.46	25.25	-	36.10	53.50	34.08
	Serial	0.3991	18.11	25.00	-	37.10	55.33	33.89
	Parallel	0.3991	31.50	36.77	-	36.70	50.67	36.41
	OFT	0.3979	25.20	24.49	-	34.10	52.67	34.11
	BOFT	0.3979	25.98	20.96	-	34.40	52.17	33.38
	LoRA	0.3991	22.24	24.24	-	37.70	56.00	35.05
	LatTE-m	0.4009	24.80	22.22	-	38.10	57.17	35.57
	LatTE-w	0.4009	24.41	22.22	-	38.50	56.00	35.28
Llama-3.2-3B	Prompt	0.0015	54.92	23.74	40.60	71.20	78.83	53.86
	Serial	0.1874	51.57	30.56	37.10	69.10	78.83	53.43
	Parallel	0.1874	51.38	27.78	39.10	67.70	81.33	53.46
	OFT	0.1977	48.62	28.54	39.60	69.90	81.33	53.60
	BOFT	0.1977	50.59	28.54	37.20	68.00	78.67	52.60
	LoRA	0.1892	50.59	27.78	39.30	68.10	78.17	52.79
	LatTE-m	0.1902	45.67	26.26	39.10	67.70	79.67	51.68
	LatTE-w	0.1902	45.47	30.56	39.70	69.30	81.50	53.31

Table 2: Accuracy on multiple-choice and arithmetic reasoning benchmarks using various PEFT methods. MATH results for Llama-1B is omitted as it replied in forms which cannot be parsed in all methods.

models.

For natural language processing (NLP), we conduct experiments using the Llama 3 (Grattafiori et al., 2024) and Qwen2.5 (Yang et al., 2024) language models, covering a range of model sizes from 1B to 14B parameters. For text-to-image generation (T2I), we fine-tune Stable Diffusion v1.5 (Rombach et al., 2022) as the base model. All fine-tuning is performed on 4 NVIDIA H100 GPUs, and inference is conducted using a single H100.

We compare both variants of LatTE – more heads (LatTE-m) and wider heads (LatTE-w) – against several strong PEFT baselines. The baseline methods include prompt tuning (Lester et al., 2021), serial (Houlsby et al., 2019) and parallel (He et al., 2022) adapters, OFT (Qiu et al., 2023), BOFT (Liu et al., 2024b), and LoRA (Hu et al., 2022). We follow the training setups from the respective references and use official implemen-

tations from the Huggingface PEFT library (Man-grulkar et al., 2022) for prompt tuning, LoRA, OFT, and BOFT.

Unless otherwise stated, we set the rank $r = 16$ for both LatTE and LoRA. For a fair comparison, the adapter hidden size and the block sizes for OFT and BOFT are chosen such that the number of trainable parameters closely matches that of LatTE and LoRA; we use 2 blocks for BOFT by default. All models are trained for up to 20 epochs for commonsense QA and 4 epochs for reasoning, and we report results using the best checkpoint selected via measuring with 4 different seeds.

4.1 Commonsense QA

We first assess general knowledge adaptation using the commonsense QA, a multiple-choice question answering task. Models are fine-tuned on the Commonsense 170K dataset (Hu et al., 2023), which comprises eight distinct QA benchmarks: OBQA (Mihaylov et al., 2018), BoolQ (Clark et al.,

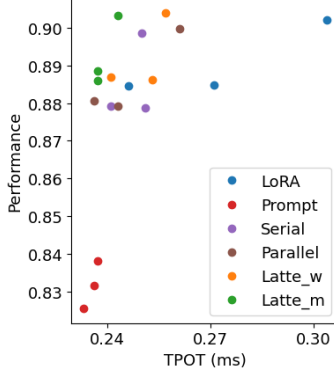


Figure 4: QA task score and inference speed on Qwen2.5-7B.

2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy, and ARC-challenge (Clark et al., 2018). These datasets focus on direct-answer selection without requiring chain-of-thought (CoT) reasoning (Wei et al., 2022).

Table 1 presents the performance of Llama-3.2-3B, Llama-3.1-8B and Qwen2.5-3B/7B models fine-tuned with LatTE and baseline methods. Across all model sizes and configurations, LatTE consistently matches the performance of established PEFT baselines. Notably, LatTE-w achieves the highest average performance on three of the four models and ranks second on the remaining model.

Beyond final accuracy, we observe that LatTE converges rapidly. As detailed in Appendix B, LatTE frequently reaches peak performance within a single epoch, significantly reducing the computational budget required for adaptation compared to other baselines.

4.2 Multiple-Choice and Arithmetic Reasoning

To test the method’s capacity for complex reasoning, we evaluate on math and logic benchmarks using Chain-of-Thought (CoT) prompting. We construct the training set by filtering examples from the Llama-Nemotron-Post-Training-Dataset (Bercovich et al., 2025), selecting those whose answers contain numeric characters and do not involve blank spaces. This filtering yields approximately 0.3M examples per epoch for training.

We evaluate on five benchmarks: AQUA (Ling et al., 2017), GPQA (Rein et al., 2024), MATH-500 (Lightman et al., 2023), GSM8K (Cobbe et al.,

2021), and SVAMP (Patel et al., 2021). Following standard practice, evaluation is based on the accuracy of the final answer, independent of the CoT content. Table 2 presents results for Llama-3.2-1B/3B and Qwen2.5-1.5B/3B. The results demonstrate that LatTE remains effective in reasoning-heavy tasks.

4.3 Equal-Parameter and Cross-Layer Ablations

Two confounds could explain LatTE’s parity with LoRA without the expansion mechanism itself doing any work: the extra r^2 parameters contributed by C , and over-parameterization masking the cross-layer role of the auxiliary stream. We isolate both on Qwen2.5-3B with the commonsense QA suite, re-running LoRA and LatTE-w in a single fixed environment and seed; per-benchmark results are given in Table 6 in Appendix B. First, setting $C = 0$ makes LatTE’s trainable parameter count exactly equal to LoRA’s $(d + k)r$ per adapted matrix, yet accuracy does not degrade (86.98 avg. against 86.89 for full LatTE-w and 86.59 for LoRA): parity is not attributable to the additional parameters. Second, to test cross-layer persistence, we apply f_{in} and f_{out} at every layer boundary rather than once at the network input and output, which resets the auxiliary state to the learned constant c at each layer – a local, input-dependent correction with cross-layer information flow removed. Both variants fall below full LatTE-w and below LoRA (85.85 and 85.94), indicating that the persistence of the auxiliary state across layers is a functioning component of the method; characterizing what it transports is left to future work.

4.4 Efficiency and Scalability

Inference Latency. A key advantage of LatTE is that it preserves the base model’s computation graph structure – a single matrix multiplication per layer – without the need for separate adapter branches. We quantify this by measuring the Time-Per-Output-Token (TPOT) on a Qwen2.5-7B model.

Figure 4 plots TPOT with the QA score with 7, 14, and 28 layers of adapters applied. We measured the inference time of 100 token generation with 10k context length, and averaged over 10 trials. OFT and BOFT results are not included as they were an order of magnitude slower than the others. One observes that LatTE achieves the optimal Pareto frontier: matching the high performance of

Method	Split	Base (=merged LoRA)	unmerged LoRA	LatTE-m	LatTE-w
Single example	All	0.236	0.294	0.243	0.264
	Attention	0.126	0.168	0.133	0.154
	FFN	0.051	0.066	0.053	0.053
	Embedding	0.059	0.060	0.057	0.057
8 Batch (1 task)	All	0.229	0.285	0.238	0.254
	Attention	0.138	0.174	0.145	0.160
	FFN	0.047	0.062	0.047	0.047
	Embedding	0.044	0.049	0.046	0.047
8 Batch (4 tasks)	All	-	-	0.247	0.251
	Attention	-	-	0.153	0.155
	FFN	-	-	0.049	0.050
	Embedding	-	-	0.045	0.046

Table 3: Inference latency breakdown by component: time per output token (in milliseconds) on Qwen2.5-7B with an H100 GPU, context length 10k, averaged over 10 runs. In the 4-task rows, merged and unmerged LoRA have no single-batch equivalent.

Method	TTFT / TPOT (s)	Tok/s	VRAM (GiB)
LatTE (masked)	0.1533 / 0.0081	988	7.16
LoRA (adapter routing)	0.2270 / 0.0137	584	7.30
LoRA (separate batching)	0.4503 / 0.0322	248	6.23

Table 4: Heterogeneous-task serving on Qwen2.5-3B (single H100, context length 1k, total batch size 8, four concurrent tasks with two requests each). TTFT: time to first token; TPOT: time per output token.

heavy adapters while retaining the low latency of lightweight ones. A comprehensive per-component breakdown, including heterogeneous-task batches, is shown in Table 3: per-token latency stays essentially flat as adapters are co-hosted and tasks are mixed within a batch, whereas unmerged LoRA pays a per-branch cost for every active adapter.

Multi-Task Serving under Heterogeneous Batching. We further benchmark LatTE against the two standard ways of serving multiple LoRA adapters when a batch mixes tasks: *adapter routing* (a standard $d \times 4$ routing matrix gates each request to its branch) and *separate batching* (each task’s requests run separately), using Qwen2.5-3B on a single H100 with context length 1k and batch size 8 over four concurrent tasks – a smaller model and shorter context than the profiling above. As Table 4 shows, LatTE reaches 988 tok/s: $1.7\times$ routed LoRA and roughly $4\times$ separate batching, with per-token latency about 40% below the routed baseline. The gap exceeds the $\sim 25\%$ single-adapter branch overhead in Table 3, consistent with routing overhead compounding as distinct tasks share a batch, whereas the masked single-path design pays no per-task routing cost. Peak memory is comparable to routed LoRA (7.16 vs. 7.30 GiB); separate batching’s lower footprint reflects running only two

sequences at a time.

Energy Profiling and Execution Efficiency.

The latency reduction achieved by LatTE directly translates to improved execution efficiency on power-constrained hardware. By fusing task-specific computation into a slightly larger matrix multiplication, LatTE avoids the memory-bound additions and separate kernel launches required by parallel adapter branches. To accurately capture the extent of this hardware advantage during high-frequency measurements, we avoid relying on baked-in hardware averaging logic. Instead, measuring instantaneous voltage and current readings indicates that the single-path computation graph of LatTE utilizes memory bandwidth far more efficiently, drastically reducing the active power spikes typically associated with routing and unmerged branch computations.

5 Conclusion

We proposed LatTE, a PEFT method that explores geometric expansion as a structural alternative to additive weight perturbation. By placing adaptation in auxiliary subspaces rather than perturbing frozen weights, LatTE preserves single-matrix-multiplication inference without weight merging, avoiding the parallel-branching latency of unmerged adapters and enabling router-free intra-batch multi-tasking. Experiments across 1B to 14B models confirm that LatTE matches state-of-the-art accuracy on diverse tasks and remains robust under quantization. While merged LoRA is optimal for static single-task serving, LatTE reduces the inference bottlenecks of dynamic, multi-tenant serving and power-constrained edge deployment.

Limitations

LatTE is not a drop-in adapter. Unlike LoRA, which attaches side branches to an unmodified network, LatTE changes the hidden dimension and, in the attention module, the head allocation (Section 3.2); serving stacks must load expanded weight matrices rather than attach adapters to a fixed checkpoint. The required operations remain standard dense linear algebra: no custom kernels are needed, the expanded matrices shard under conventional tensor parallelism, and accuracy is robust to INT8 quantization (below 0.3% degradation; Table 8). We judge the integration effort comparable to that of prefix tuning.

Scaling with the number of hosted tasks. Hosting N tasks expands each adapted weight matrix by Nr auxiliary dimensions (Section 3.4), so parameter count, compute, and – under the dense allocation used in all of our measurements – KV-cache footprint grow linearly with N . For a single task at $r = 16$ the per-token KV-cache overhead is below 0.5% for Qwen2.5-7B ($d = 3584$) and below 0.8% for Qwen2.5-3B ($d = 2048$), and our serving benchmark (Table 4) covers four concurrent tasks; we have not validated behavior at substantially larger task counts. Because masking zeroes a request’s auxiliary state outside its own r -dimensional slot, a mask-aware allocator could in principle hold the per-request KV cache at $d + r$ dimensions independent of N . This is an architectural implication of the design, not a property of our implementation: we have neither built nor benchmarked such an allocator.

Ethics and Broader Impact

This paper presents work whose goal is to advance the field of machine learning, specifically by improving the inference efficiency of fine-tuned models. By reducing the computational overhead required for task-specific adaptation, our method facilitates the deployment of Large Language Models and Diffusion Models on resource-constrained edge devices. This contributes to the democratization of advanced AI capabilities and potentially reduces the energy consumption and carbon footprint associated with serving large-scale models. While our method increases the accessibility of powerful generative models, which carries inherent dual-use risks, it does not introduce new safety concerns beyond those present in the base models.

References

- Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328.
- Eman Alsuradi, Junhyun Lee, Kyenghun Lee, Hyeonmok Ko, and Fahed Jubair. 2026. Grouped adaptive weight sharing (gaws): An inference-efficient adaptation method for large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 31790–31806.
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shafaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. *Llama-nemotron: Efficient reasoning models*. Preprint, arXiv:2505.00949.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.
- Dhananjaya Gowda, Seoha Song, Harshith Goka, and Junhyun Lee. 2025. zFLoRA: Zero-latency fused

- low-rank adapters. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21412–21429.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Demi Guo, Alexander M Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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 4884–4896.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.
- Lukas Hedegaard, Aman Alok, Juby Jose, and Alexandros Iosifidis. 2024. Structured pruning adapters. *Pattern Recognition*, 156:110724.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.
- Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276.
- Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.
- Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. 2023. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8506–8515.
- Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, and 1 others. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.
- Yang Li, Shaobo Han, and Shihao Ji. 2024. Vb-lora: Extreme parameter efficient fine-tuning with vector banks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:109–123.
- Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4260.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *ACL*.
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohata, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.
- Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.
- Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, and 1 others. 2024b. Parameter-efficient orthogonal finetuning via butterfly factorization. In *The Twelfth International Conference on Learning Representations*.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

- Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.
- Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. 2024. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 4296–4304.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.
- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503.
- Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673.
- Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. 2023. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362.
- Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. **GPQA: A graduate-level google-proof q&a benchmark**. In *First Conference on Language Modeling*.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.
- Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.
- Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- 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.
- Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. 2024a. Llama pro: Progressive llama with block expansion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6518–6537.
- Xun Wu, Shaohan Huang, and Furu Wei. 2024b. Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.
- Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024c. Left: Representation

- finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962.
- Jingwei Xu, Junyu Lai, and Yunpeng Huang. 2025. Me-teoRA: Multiple-tasks embedded loRA for large language models. In *The Thirteenth International Conference on Learning Representations*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.
- Shen Yuan, Haotian Liu, and Hongteng Xu. 2024. Bridging the gap between low-rank and orthogonal adaptation via householder reflection adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.
- Yuchen Zeng and Kangwook Lee. 2024. The expressive power of low-rank adaptation. In *The Twelfth International Conference on Learning Representations*.
- Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023a. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.
- Weilin Zhao, Yuxiang Huang, Xu Han, Zhiyuan Liu, Zhengyan Zhang, Kuai Li, Chen Chen, TAO YANG, and Maosong Sun. 2024. Ca-lora: Adapting existing lora for compressed llms to enable efficient multi-tasking on personal devices. In *First Conference on Language Modeling*.
- Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. 2021. Counter-interference adapter for multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2812–2823.

A Proof of Theorems

Here we provide the proofs of the two theorems in Section 3.3.

Theorem 1 For linear models, the minimum low-rank dimension required for adapter models to exactly recover the FFT target is identical for both LoRA and LatTE.

Proof.

The proof closely follows the derivation of the expressive power of LoRA in linear models in (Zeng and Lee, 2024). Let us define the linear target model f and the linear frozen model f_0 with L layers.

- $\bar{f}(x) = \bar{W}x$
- $f_0(x) = \left(\prod_{l=1}^L W_l\right) x$

Now, we define the LatTE adapted linear model as f . We assume a low-rank of R .

$$\bullet f(x) = \begin{bmatrix} 1 & 0 \end{bmatrix} \prod_{l=1}^L \begin{bmatrix} W_l & B_l \\ A_l & C_l \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} x$$

Note that in the LatTE model, we used $f_{\text{in}}(x) = [x; 0]$ and $f_{\text{out}}([y; \bar{y}]) = y$ as the expansion and compression function for simplicity.

We consider the following quantity:

$$\left\| \begin{bmatrix} 1 & 0 \end{bmatrix} \prod_{l=1}^L \begin{bmatrix} W_l & B_l \\ A_l & C_l \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \bar{W} \right\|. \quad (4)$$

With the subtraction, all terms inside the norm can be rearranged as the following $L - 1$ terms:

$$\begin{aligned} & \begin{bmatrix} 1 & 0 \end{bmatrix} \prod_{l=2}^L \begin{bmatrix} W_l & B_l \\ A_l & C_l \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} A_1 \\ & + \begin{bmatrix} 1 & 0 \end{bmatrix} \prod_{l=3}^L \begin{bmatrix} W_l & B_l \\ A_l & C_l \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} A_2 W_1 \\ & \quad \vdots \\ & + \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} W_L & B_L \\ A_L & C_L \end{bmatrix} \begin{bmatrix} W_{L-1} & B_{L-1} \\ A_{L-1} & C_{L-1} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} A_{L-2} \prod_{i=1}^{L-3} W_i \\ & + \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} W_L & B_L \\ A_L & C_L \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} A_{L-1} \prod_{i=1}^{L-2} W_i. \end{aligned}$$

Now let us define the error matrix E as $E \equiv \bar{W} - \prod_{l=1}^L W_l$. Using SVD:

$$\begin{aligned} E &= UDV^\top = UD\mathbb{I}_{1:R(L-1)}V^\top + \Delta E \\ &= \sum_{l=1}^{L-1} UD\mathbb{I}_{R(l-1)+1:Rl}V^\top + \Delta E \\ &\equiv \sum_{l=1}^{L-1} E'_l + \Delta E, \end{aligned} \quad (5)$$

where $\mathbb{I}$ is the identity matrix. Note that all E'_l 's have a rank of R .

We assign each E'_l 's to the $L - 1$ terms in the expansion of Eq. (4). Then the last equation becomes $B_L A_{L-1} \prod_{i=1}^{L-2} W_i = E_{L-1}$, and thus, $B_L A_{L-1} = E_{L-1} \left(\prod_{i=1}^{L-2} W_i \right)^{-1}$. Assuming that the inverse

exists, since the right-hand side of the equation is of rank R , it can be decomposed to $B_L \in \mathbb{R}^{d \times R}$ and $A_{L-1} \in \mathbb{R}^{R \times d}$. One example of the decomposition can be achieved by SVD.

For simplicity, let us assume all $C = 0$. This does not affect the claim of the theorem, because setting $C = 0$ can only decrease the expressive power of LatTE. With this simplification, we can write all other pairwise product of A and B matrices in a closed form: $B_{L-1}A_{L-2} = W_L^{-1}E_{L-2} \left(\prod_{l=1}^{L-3} W_l \right)^{-1}$, $B_{L-2}A_{L-3} = (W_L W_{L-1} + B_L A_{L-1})^{-1} E_{L-3} \left(\prod_{l=1}^{L-4} W_l \right)^{-1}, \dots$. The existence of the left-hand side decomposition is guaranteed by SVD.

This proves that

$$\min \left\| [1 \ 0] \prod_{l=1}^L \begin{bmatrix} W_l & B_l \\ A_l & C_l \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \bar{W} \right\| = \sigma_{RL-R+1}(E), \quad (6)$$

where $\sigma_i(E)$ indicates the i -th largest singular value of E .

Zeng and Lee (2024) shows the same equation as Eq. (6) for LoRA, which proves that LoRA and LatTE has identical expressivity for linear models.

Theorem 2. Any attention matrix produced by LoRA can be represented as an attention matrix from LatTE, given that $\bar{x}$ is a linear transform of x .

Proof.

i) Attention matrix of LoRA:

$$\begin{aligned} & x^\top \left((W_k + \Delta W_k)^\top (W_Q + \Delta W_Q) \right) x \\ &= x^\top \left((W_k + b_k^\top a_k^\top)^\top (W_Q + a_Q b_Q) \right) x \end{aligned} \quad (7)$$

ii) Attention matrix of LatTE:

$$\begin{aligned} & [x^\top \ \bar{x}^\top] \begin{bmatrix} W_K^\top & A_K^\top \\ B_K^\top & C_K^\top \end{bmatrix} \begin{bmatrix} W_Q & B_Q \\ A_Q & C_Q \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix} \\ &= x^\top (W_K^\top W_Q + B_K^\top B_Q) x + x^\top (W_K^\top A_Q + B_K^\top C_Q) \bar{x} \\ &+ \bar{x}^\top (A_K^\top W_Q + C_K^\top B_Q) x + \bar{x}^\top (A_K^\top A_Q + C_K^\top C_Q) \bar{x} \end{aligned} \quad (8)$$

We claim that any LoRA attention can be expressed by LatTE attention. Let $A_Q = a_Q$, $A_K = a_K$, $B_Q = a_K^\top W_Q$, $C_Q = a_K^\top a_Q$, B_K , C_K satisfies $B_K + C_K b_Q = b_K - b_Q$, and $\bar{x} = b_Q x$. Note that the solution for $B_K + C_K b_Q = b_K - b_Q$ always exists since $B_K = b_K$ and $C_K = -1_r$ (identity matrix) satisfies the condition.

Then the LatTE attention becomes:

$$\begin{aligned} & x^\top (W_K^\top W_Q + B_K^\top B_Q + W_K^\top a_Q b_Q + B_K^\top C_Q b_Q \\ &+ b_Q^\top a_K^\top W_Q + b_Q^\top C_K^\top B_Q + b_Q^\top a_K^\top a_Q b_Q + b_Q^\top C_K^\top C_Q b_Q) x \\ &= x^\top (W_K^\top + b_K^\top a_K^\top) (W_Q + a_Q b_Q) x \\ &+ x^\top ((b_Q - b_K)^\top a_K^\top W_Q + (b_Q - b_K)^\top a_K^\top a_Q b_Q \\ &+ (B_K + C_K b_Q)^\top B_Q + (B_K + C_K b_Q)^\top C_Q b_Q) x \end{aligned} \quad (9)$$

The second term identically vanishes with $C_Q = a_K^\top a_Q$, $B_Q = a_K^\top W_Q$ and $B_K + C_K b_Q = b_K - b_Q$. This proves any LoRA attention matrix can be expressed as LatTE attention matrix assuming $\bar{x}$ is a linear transform of x .

B More Results on Language Tasks

Here we provide additional experimental results on the LLMs.

B.1 Ablation in embedding size (r)

We present an ablation on the embedding size, from $r = 8$ to $r = 32$, in Table 5. These are Qwen2.5-7B results on the commonsense QA task, and the adapters are applied to half of the layers.

Rank	Fintuning	Accuracy								
		OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.
8	Prompt	85.60	85.55	76.18	76.01	78.65	67.68	95.64	88.44	81.72
	Serial	89.95	88.43	88.07	79.18	91.89	80.82	95.62	89.40	87.92
	Parallel	90.44	88.70	88.66	78.36	92.27	82.04	95.35	88.72	88.07
	OFT	90.30	88.65	87.46	78.21	90.40	80.51	96.40	90.38	87.79
	BOFT	90.00	88.43	87.60	78.25	91.13	80.62	96.35	90.02	87.80
	LoRA	90.90	89.27	88.97	78.45	92.42	82.87	95.81	88.91	88.45
	LatTE-m	92.10	89.21	88.52	78.80	92.67	82.20	95.56	89.80	88.61
	LatTE-w	91.20	89.44	88.74	79.72	92.56	82.46	95.95	89.44	88.69
16	Prompt	87.95	85.24	86.60	76.67	76.44	66.75	96.39	89.23	83.16
	Serial	89.64	87.98	87.91	79.30	91.66	81.14	95.79	89.57	87.87
	Parallel	90.90	88.32	87.85	79.11	91.94	80.27	95.63	89.29	87.92
	OFT	90.35	88.49	88.40	79.49	91.89	82.20	96.32	89.78	88.36
	BOFT	90.00	88.43	87.60	78.25	91.13	80.62	96.35	90.02	87.80
	LoRA	91.10	89.05	88.29	78.68	92.67	83.09	95.57	89.38	88.48
	LatTE-m	91.70	98.34	88.79	79.45	92.20	83.80	95.91	89.69	88.86
	LatTE-w	91.20	89.26	88.55	78.90	91.94	83.19	95.76	90.17	88.62
32	Prompt	85.80	83.75	86.28	76.26	70.95	62.49	96.03	88.82	81.30
	Serial	89.50	88.29	87.99	79.35	91.64	80.80	95.80	89.68	87.88
	Parallel	90.70	88.52	88.15	79.55	92.00	80.60	95.75	89.31	88.07
	OFT	90.55	89.20	88.48	79.26	92.20	83.28	96.26	89.93	88.65
	BOFT	91.50	89.32	88.23	79.08	92.37	83.11	96.37	89.68	88.71
	LoRA	91.64	89.14	89.01	78.43	92.64	83.52	95.40	88.78	88.57
	LatTE-m	92.04	89.33	88.69	80.12	92.21	82.32	96.30	90.27	88.91
	LatTE-w	92.44	89.07	88.69	79.93	92.29	82.64	96.02	90.34	88.93

Table 5: Accuracy results on the commonsense QA benchmark, different auxiliary embedding size. Adapters are applied to half of the layers.

B.2 Equal-parameter and cross-layer ablations

Table 6 reports the per-benchmark accuracies for the equal-parameter and cross-layer ablations discussed in Section 4.3.

Method	OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.
LoRA	86.79	86.63	86.82	79.21	92.98	82.58	94.10	83.61	86.59
LatTE-w	87.22	87.84	86.34	80.14	92.68	83.00	94.19	83.70	86.89
LatTE-w w/o C	87.25	87.46	86.63	80.46	92.79	82.76	94.50	83.98	86.98
LatTE-w w/o cross-layer (1)	84.87	85.85	86.58	78.33	92.45	81.65	93.92	83.15	85.85
LatTE-w w/o cross-layer (2)	84.87	86.01	87.05	79.26	92.37	81.40	93.78	82.79	85.94

Table 6: Equal-parameter and cross-layer ablations on commonsense QA with Qwen2.5-3B. All rows were re-run in a single fixed environment and seed, so the LoRA and LatTE-w rows differ slightly from Table 1. “w/o C ” sets $C = 0$, making the trainable parameter count identical to LoRA’s. The cross-layer variants apply $f_{\text{in}}/f_{\text{out}}$ at every layer boundary, removing cross-layer transport of the auxiliary state; variant (1) uses $f_{\text{out}}([y; \bar{y}]) = y$ and variant (2) uses $f_{\text{out}}([y; \bar{y}]) = y + B\bar{y}$, both with $f_{\text{in}}(x) = [x; c]$.

B.3 Effectiveness in larger models

To ensure LatTE is effective across different model sizes, we add an experiment on Qwen2.5-14B in Table 7. These are results on the commonsense QA task, and the adapters are applied to half of the layers. The results show that LatTE is effective in Qwen2.5-14B, and suggests that its effectiveness compared to LoRA will not scale over parameter size.

Model	Fintuning	Accuracy								
		OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.
Qwen2.5-14B	LoRA	94.20	90.30	91.07	81.41	93.85	87.04	97.61	93.28	91.09
	LatTE-m	94.10	90.31	91.01	80.90	94.07	87.12	97.63	93.58	91.09
	LatTE-w	94.10	90.47	91.57	81.12	94.09	87.25	97.54	93.75	91.24

Table 7: Accuracy results on the commonsense QA benchmark for Qwen2.5-14B and rank $r = 16$. Adapters are applied to half of the layers.

B.4 Quantization

We have verified that LatTE works when quantizing the model. For the new results on Qwen2.5-14B, with adapters in half of the layers, and $r = 16$, quantizing to INT8 results are shown in Table 8

Precision	Fintuning	Accuracy									Degradation
		OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.	
BF16	LoRA	94.20	90.30	91.07	81.41	93.85	87.04	97.61	93.28	91.09	-
	LatTE-m	94.10	90.31	91.01	80.90	94.07	87.12	97.63	93.58	91.09	-
	LatTE-w	94.10	90.47	91.57	81.12	94.09	87.25	97.54	93.75	91.24	-
INT8	LoRA	93.76	90.22	90.92	80.67	93.51	86.86	97.50	93.26	90.84	0.27%
	LatTE-m	94.24	90.41	91.02	81.01	93.93	86.54	97.47	93.47	91.01	0.09%
	LatTE-w	94.04	90.24	91.51	80.67	93.86	87.39	97.48	93.36	91.07	0.19%

Table 8: Quantization results on the commonsense QA benchmark for Qwen2.5-14B and rank $r = 16$. Adapters are applied to half of the layers.

B.5 Design Choices

While LatTE does not introduce numerical hyperparameters beyond the rank r , it does involve several architectural design choices. As discussed in Section 3, these include: (1) the implementation of multi-head attention, (2) the initial expansion function $f_{\text{in}}(x)$, and (3) the final compression function $f_{\text{out}}([y, \bar{y}])$.

In addition to our default settings for f_{in} and f_{out} , we consider several plausible alternatives. For the expansion function f_{in} , we experiment with $f_{\text{in}}(x) = [x; \mathbf{0}_r]$, where the auxiliary embedding is an r -dimensional zero vector $\mathbf{0}_r$. For the compression function f_{out} , we test discarding the auxiliary output, $f_{\text{out}}([y, \bar{y}]) = y$. $f_{\text{in}}(x) = [x; \text{pool}(x)]$, where the auxiliary embedding is derived via pooling, and $f_{\text{out}}([y, \bar{y}]) = y + \text{repeat}(\bar{y})$ was also considered but showed suboptimal performance. Table 9 reports the impact of these alternatives on commonsense QA performance.

Design	Accuracy (Qwen2.5-7B)									
	OBQA	BoolQ	PIQA	SIQA	Hella.	Wino.	ARC-e	ARC-c	ave.	
Baseline (LatTE-w)	92.60	89.49	90.59	80.73	94.67	87.86	97.00	90.25	90.40	
LatTE-m	91.84	89.81	90.08	81.28	94.52	87.49	96.74	90.76	90.32	
$f_{\text{in}}(x) = [x; \mathbf{0}_r]$	90.53	96.56	89.50	92.00	89.61	80.73	94.40	86.50	89.98	
$f_{\text{out}}([y, \bar{y}]) = y$	90.19	96.77	89.86	91.56	89.89	81.12	93.83	85.78	89.88	
$f_{\text{in}}(x) = [x; \mathbf{0}_r]; f_{\text{out}}([y, \bar{y}]) = y$	90.44	96.72	89.77	92.40	89.69	80.30	94.27	86.44	90.00	

Table 9: Accuracy results on the commonsense QA benchmark compared across different design choices, including f_{in} , f_{out} , and the more/wider heads.

B.6 Epoch-wise performance

For the Commonsense QA task, we measured the performance as a function of epochs (Figure 5). The LatTE variants show fast convergence to peak performance compared to other methods.

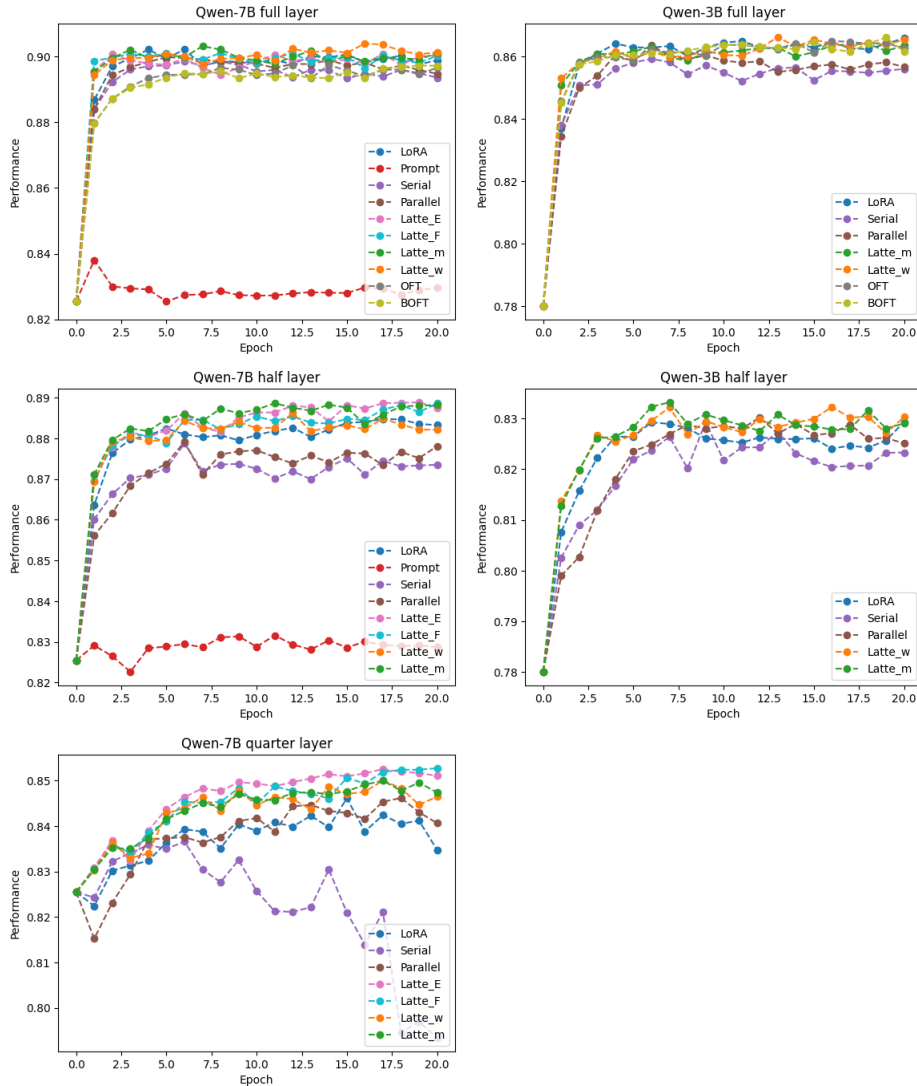


Figure 5: Epoch-wise performance for QA task.

C Text-to-Image Generation

Finally, we verify that LatTE extends beyond language models by applying it to Stable Diffusion v1.5 for subject-driven generation. Specifically, we use DreamBooth (Ruiz et al., 2023) dataset on Stable Diffusion v1.5 and test LoRA and LatTE’s adaptation to subject-driven generation. We follow the default settings of the Huggingface Diffusers library for LoRA finetuning and applied LatTE to the identical positions, which are the attention blocks of the U-net (Ronneberger et al., 2015).

The qualitative results are shown in Figure 6. LatTE successfully captures subject details (e.g., specific dog breeds or objects) with high fidelity, comparable to LoRA. Crucially, this confirms that the “auxiliary representation” paradigm is model-agnostic and can be effective in continuous, spatial signal processing contexts as well as discrete token processing.

Note that unlike transformers, the convolution layers in the Stable Diffusion v1.5 (and other similar latent diffusion models) changes its latent dimension frequently. To effectively apply LatTE in these models, one needs to skip-connect the auxiliary dimension across convolution layers to avoid multiple expansion (f_{in}) and compression (f_{out}).



Figure 6: Subject-driven generation of LoRA and LatTE. All examples share the same seed for the two methods.