FastFLUX: Pruning FLUX with Block-wise Replacement and Sandwich Training

Fuhan Cai[1*], Yong Guo[2*], Jie Li[3*], Wenbo Li[4], Jian Chen[3], Xiangzhong Fang[1†]

[1]Shanghai Jiao Tong University
[2]Max Planck Institute for Informatics
[3]South China University of Technology
[4]Chinese University of Hong Kong
freyacaifuhan@sjtu.edu.cn, guoyongcs@gmail.com, selijie@mail.scut.edu.cn,
fenglinglwb@gmail.com, ellachen@scut.edu.cn, xzfang@sjtu.edu.cn

Abstract

Recent advancements in text-to-image (T2I) generation have led to the emergence of highly expressive models such as diffusion transformers (DiTs), exemplified by FLUX. However, their massive parameter sizes lead to slow inference, high memory usage, and poor deployability. Existing acceleration methods (e.g., single-step distillation and attention pruning) often suffer from significant performance degradation and incur substantial training costs. To address these limitations, we propose **FastFLUX**, an architecture-level pruning framework designed to enhance the inference efficiency of FLUX. At its core is the **Block-wise Replacement with Linear Layers (BRLL)** method, which replaces structurally complex residual branches in ResBlocks with lightweight linear layers while preserving the original shortcut connections for stability. Furthermore, we introduce **Sandwich Training (ST)**, a localized fine-tuning strategy that leverages LoRA to supervise neighboring blocks, mitigating performance drops caused by structural replacement. Experiments show that our FastFLUX maintains high image quality under both qualitative and quantitative evaluations, while significantly improving inference speed, even with 20% of the hierarchy pruned.

Code — https://github.com/freya9933/FastFLUX

Introduction

Recent advances in diffusion-based generative models (Zhang et al. 2024; Esser et al. 2024; Podell et al. 2024; Sauer et al. 2023; Chen et al. 2023; Zhang, Rao, and Agrawala 2023) have substantially improved the fidelity and controllability of text-to-image synthesis. Transformer-based architectures such as DiT (Peebles and Xie 2023) and FLUX (Black Forest Labs 2024) further enhance generation quality by stacking deep hierarchical blocks that increase model expressiveness and scalability. However, these designs also introduce significant computational overhead, including large memory consumption and high inference latency, which limits their suitability for real-time generation and deployment on resource-constrained hardware.

Although prior studies have investigated model compression through pruning (Castells et al. 2024; Fang, Ma, and Wang 2023), knowledge distillation (Chen et al. 2025; Meng et al. 2023; Salimans and Ho 2022), and quantization (Li et al. 2024; So et al. 2023; He et al. 2023a), these approaches primarily focus on parameter-level optimization, largely overlooking the hierarchical structural organization of deep generative transformers. Specifically, hierarchical redundancy in diffusion-based generative models remains largely underexplored, despite being a significant source of inefficiency in modern deep transformer architectures. Furthermore, many existing techniques (Shirkavand et al. 2025; Fang, Ma, and Wang 2023) require full-model fine-tuning after pruning or incur substantial performance degradation, which limits their practical applicability.

In this work, we revisit the problem of hierarchical redundancy in large diffusion transformers and propose **FastFLUX**, a general and efficient framework for architecture-level compression. FastFLUX operates by selectively replacing the computationally heavy residual branches of FLUX blocks with lightweight linear alternatives, while preserving generation quality through a localized fine-tuning strategy. Specifically, we introduce **Block-wise Replacement with Linear Layers (BRLL)**, which targets the high-cost residual pathways in ResBlocks. In FLUX, each ResBlock contains a heavy residual branch and a lightweight shortcut connection; the residual branch dominates both FLOPs and latency during inference. BRLL substitutes this branch with a simple linear layer while preserving the shortcut path, ensuring stable information flow. These linear layers can be initialized via least-squares estimation, providing a training-free initialization with controllable accuracy–efficiency trade-offs. To mitigate performance degradation after replacement, we propose a **Sandwich Training (ST)** strategy. Instead of full-model fine-tuning, ST performs localized adaptation by applying LoRA (Hu et al. 2022) modules to the neighboring unpruned layers of each re-

*These authors contributed equally.
†Corresponding author.

placed block, forming a small "sandwich" around the linear module. This local supervision enables the replaced branch to adapt effectively within its context, achieving stable and high-quality generation with minimal training overhead. Using these approaches, we prune 20% of the FLUX model while retaining high visual fidelity and significantly improving inference efficiency.

To sum up, our **contributions** are as follows:

- We propose **FastFLUX**, a novel architecture-level pruning framework built upon the 12B FLUX model. FastFLUX is lightweight, efficient, and incurs low training cost. Even under a 20% pruning ratio, it maintains high visual generation quality.
- We introduce **Block-wise Replacement with Linear Layers (BRLL)**, which replaces complex residual branches with linear layers. These replacements support training-free initialization via least-squares regression, with controllable performance degradation.
- We propose **Sandwich Training (ST)**, a localized fine-tuning strategy that applies LoRA to the adjacent unpruned layers of each replaced block to compensate for performance loss after pruning, enabling targeted adaptation with minimal training overhead.

Related Work

Diffusion-based Generative Models. Diffusion models have achieved remarkable progress across diverse domains (Cao et al. 2024), establishing foundations through discrete frameworks like DDPM (Ho, Jain, and Abbeel 2020) and latent space formulations such as Latent Diffusion (Rombach et al. 2022a). Transformer-based architectures like DiT (Peebles and Xie 2023) demonstrate strong scalability, with recent models such as FLUX (Black Forest Labs 2024) further increasing depth to enhance generation quality. However, deeper hierarchical stacks introduce substantial computational and memory overhead, motivating the need for efficient compression strategies.

Model Quantization and Pruning. To reduce computational costs, diffusion models employ quantization and pruning techniques (Shen et al. 2025). Post-training quantization methods like PTQ4DM (Shang et al. 2023), Q-Diffusion (Li et al. 2023), and PTQD (He et al. 2023b) address timestep-specific calibration challenges, while quantization-aware training approaches such as TDQ (So et al. 2023) and QALoRA (He et al. 2023a) learn optimized representations. Complementary pruning methods remove redundant weights via Taylor expansion (Fang, Ma, and Wang 2023), latent space guidance (Castells et al. 2024), or dynamic programming (Kim et al. 2024), achieving substantial size reduction with minimal quality loss.

Transformer Compression. Transformer compression tackles computational challenges through multiple strategies (Tang et al. 2024). Knowledge distillation transfers capabilities from large teachers to compact students (Sanh et al. 2019; Touvron et al. 2021), while quantization reduces precision via methods like FQ-ViT (Lin et al. 2021) and PTQ-ViT (Liu et al. 2021). Pruning techniques eliminate redundancy through head removal (Yu and Xiang 2023) or context selection (Anagnostidis et al. 2023). Architectural innovations target attention mechanisms (Ainslie et al. 2023; Hatamizadeh et al. 2023), feed-forward networks (Du et al. 2022; Touvron et al. 2023), or propose alternative designs like Mamba (Gu and Dao 2024; Zhu et al. 2024). While these methods address parameter and structural redundancy, explicit compression of depth-wise redundancy in diffusion transformers remains underexplored.

FastFLUX: Block-wise Efficient Pruning with Sandwich Training

FastFLUX is a lightweight and effective framework for pruning large-scale diffusion transformers, improving inference efficiency while preserving high-quality generation. We begin by introducing **Block-wise Replacement with Linear Layers (BRLL)**, which substitutes the computationally heavy residual branches in ResBlocks with linear approximations while keeping the original shortcut pathways intact to maintain stable information flow. We then describe **Sandwich Training (ST)**, a localized fine-tuning strategy that applies LoRA-based supervision to the neighboring unpruned layers of each replaced block, providing targeted adaptation with minimal training overhead. An overview of the full pipeline, together with implementation details, is provided in Figure 1 and Algorithm 1.

Block-wise Replacement with Linear Layers

To effectively compress ultra-large diffusion transformers such as FLUX, we propose a structural pruning strategy termed BRLL, which replaces the computationally expensive residual pathways of each block with lightweight linear modules. We observe that FLUX employs deep stacks of transformer-style ResBlocks, each of which consists of an expressive residual branch and a shortcut connection. Our profiling reveals that the residual branches dominate both parameter count and FLOPs, making them the primary bottleneck for inference. Motivated by this observation, BRLL substitutes only the residual branch with a lightweight linear approximation while preserving the original shortcut pathway. Unlike replacing the entire block, which results in severe degradation due to the loss of the identity pathway, selective replacement preserves stable information flow and significantly reduces computational cost. This distinguishes our approach from prior pruning methods that often disregard the functional asymmetry within ResBlocks. An overview of BRLL is provided in Figure 2.

Formally, let X denote the input to a ResBlock, $f(X)$ the output of its residual branch, and $Y = f(X)+X$ the original output of the ResBlock. To reduce computation while preserving the residual structure, we replace the residual branch with a lightweight linear layer $g(X) = WX+b$, resulting in the modified output $\hat{Y} = g(X) + X$. To preserve the representational behavior of the original block, we minimize the approximation error between the original residual output and its linear approximation. This leads to the following objective: $\min_{W,b} ||f(X)-(WX+b)||^2$. The optimal parameters W^* can be obtained using the least squares method based on the collected input-output pairs. By defining the augmented

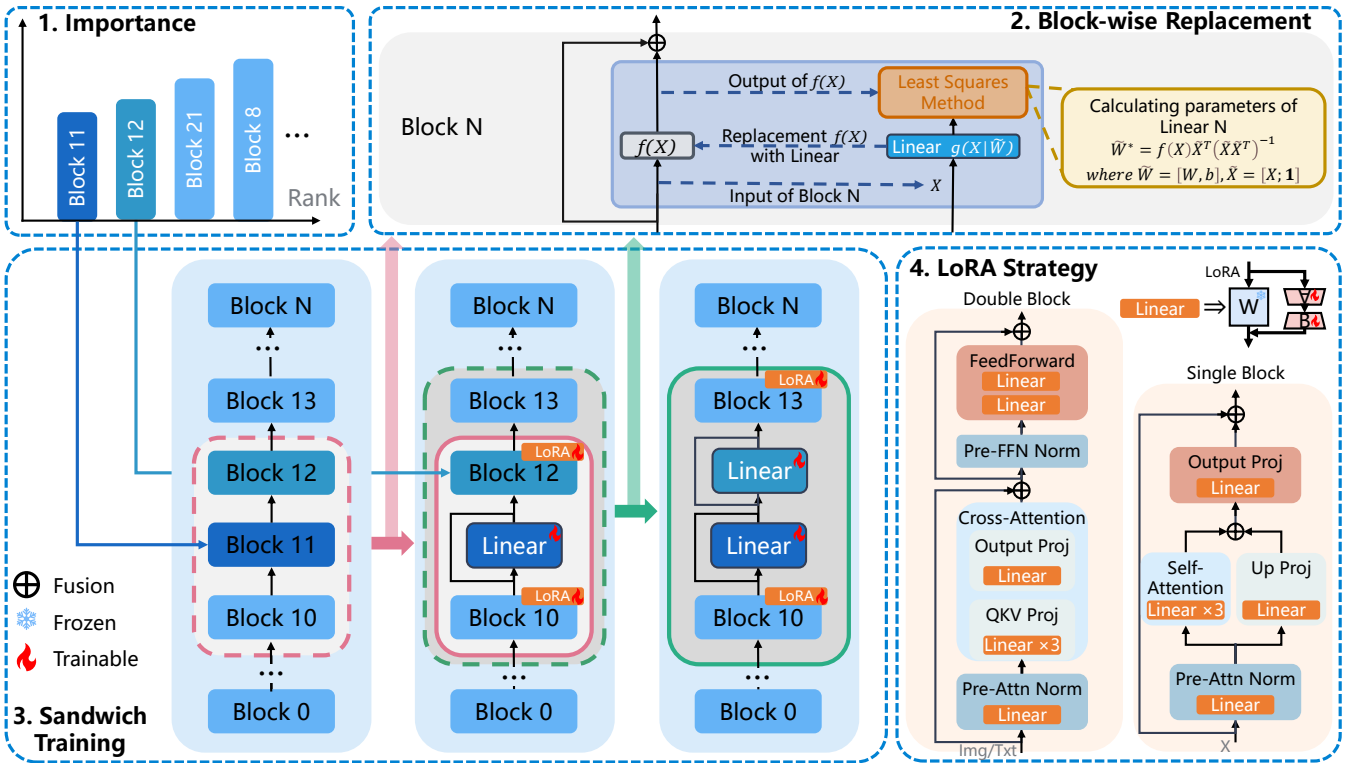

Figure 1: Overview of the FastFLUX Framework. **(1) Importance Estimation**: Each block is ranked by estimated importance to determine the pruning order. **(2) Block-wise Replacement with Linear Layers (BRLL)**: Selected blocks have their residual branches replaced with lightweight linear layers, whose parameters are initialized using a closed-form least-squares fit to the original block outputs. **(3) Sandwich Training (ST)**: For non-overlapping cases, we train the replaced block along with its immediate neighbors enhanced by LoRA. For overlapping cases, we search upward or downward to locate the nearest unpruned blocks, apply LoRA to those, and jointly train all intermediate replaced blocks. Input-output data for each training step are re-collected by loading all previously trained sandwich structures. **(4) LoRA Strategy**: LoRA modules are selectively inserted into either single or double blocks during ST to provide local supervision while keeping the rest of the model frozen. This framework enables efficient and structured pruning with minimal performance degradation.

input $\tilde{X} = [X; \mathbf{1}]$ and the combined parameters $\tilde{W} = [W, b]$, the closed-form solution is:

$$\tilde{W}^* = f(X)\tilde{X}^T(\tilde{X}\tilde{X}^T)^{-1}. \quad (1)$$

This procedure yields an efficient closed-form initialization of the linear layer's parameters, which can be further optimized during the subsequent Sandwich Training phase.

Sandwich Training

To mitigate performance degradation after block replacement, we introduce **Sandwich Training (ST)**, a localized fine-tuning strategy that adapts each replaced block using contextual supervision from its nearest unpruned neighbors. We observe that different blocks contribute unequally to the model's performance, and prioritizing the pruning of less important blocks leads to more stable and effective compression. Therefore, we begin this section by introducing a method for **Estimating Block Importance**. Based on the computed importance scores, block pruning may lead to structural overlap between selected regions. To address this, our ST strategy includes dedicated mechanisms for **Training Sandwich Blocks without Overlap** and **with Overlap**, respectively. Finally, in **Parameter Reuse across Pruning Ratios**, we present an empirical finding that transferring trained parameters from high-pruned models to lower-pruned configurations can further improve performance without additional training.

Block Importance Estimation. We observe that different blocks contribute unequally to the model's generative performance (see supplementary materials for more results). Based on this observation, we adopt an importance-aware pruning strategy that prioritizes the removal of less critical blocks. To quantitatively assess the importance of each block, we combine two commonly used metrics in text-to-image evaluation: FID (Heusel et al. 2017) and CLIP Score (Radford et al. 2021). Specifically, for each ResBlock, we temporarily remove it from the model and generate a set of images. We then compute the FID and CLIP scores for each generated set and normalize both scores independently across all blocks. Since a lower FID indicates better quality, we transform it by taking $1 - FID_{norm}$ so that higher values are consistently better across both metrics. The importance

Algorithm 1: Block-wise Replacement with Linear Layer and Sandwich Training.

Input: Pretrained FLUX model, replacement ratio $r \in [0, 1]$, the number of blocks in the model N, the weight of FID in computing the importance score α, the weight of CLIP score in computing the importance score β.
Output: Pruned model with replaced linear blocks.

for each block B_i **do**
  Temporarily remove B_i and conduct forward propagation on benchmark datasets;
  Compute FID and CLIP score;
  Compute the importance score $s_i = \alpha\text{FID} + \beta\text{CLIP}$;
end for
Sort the importance scores $\{s_i\}_{i=1}^{N}$ in the ascending order;
Select the top-r least important blocks $\{B_k\}_{k=1}^{rN}$ according to their importance scores;
for each $B_k \in \{B_k\}_{k=1}^{rN}$ **do**
  // Block-wise Replacement with Linear Layer
  Collect the input and output of the residual branch in B_k;
  Compute the weight of the linear layer that is used to replace the residual branch:
  $\tilde{W}^* = f(X)\tilde{X}^T(\tilde{X}\tilde{X}^T)^{-1}, \tilde{X} = [X; \mathbf{1}], \tilde{W} = [W, b]$
  // Sandwich Training
  Take B_k as the center and combine the ones before and after it to construct a **sandwich block**;
  Collect the input and output of the sandwich block for finetuning;
  Replace the residual branch of B_k with the linear layer using the weights $\tilde{W}^*$;
  Add LoRA into the first and last blocks inside the sandwich block;
  Finetune the sandwich by minimizing the difference between the original and pruned one.
end for

score s_i for the i-th block is defined as:

$$s_i = -(\alpha(1 - \text{FID}^{(\text{i})}_{\text{norm}}) + \beta\text{CLIP}^{(\text{i})}_{\text{norm}}), \tag{2}$$

where α and β are hyperparameters that balance the contribution of each metric. In our experiments, we set $\alpha = \beta = 0.5$. A lower score s_i implies that removing the block has a smaller impact on the overall generation quality, indicating that the block is less important and should be pruned earlier in the compression process. We therefore sort all blocks in ascending order of their importance scores and perform block replacement accordingly. Although Avrahami et al. also employs layer-wise importance estimation, it relies solely on similarity-based metrics. In contrast, we incorporate both similarity and image quality assessments, enabling a more comprehensive evaluation of each block's contribution to generation performance.

Training Sandwich Blocks without Overlap. When a ResBlock's residual branch is replaced with a single linear layer, the number of trainable parameters becomes significantly limited. To better align the pruned model with the original network and mitigate accuracy degradation caused by pruning, we introduce the ST strategy. The key idea is to leverage the neighboring blocks to enhance the training of the locally replaced block. Specifically, when the block to be replaced is $\mathbf{B}_i$, we first collect training data by recording the input to $\mathbf{B}_{i-1}$ and the output from $\mathbf{B}_{i+1}$ during a forward pass. Then, we isolate three blocks: $\mathbf{B}_{i-1}$, $\mathbf{B}_i$, $\mathbf{B}_{i+1}$. We replace the residual branch of $\mathbf{B}_i$ with a linear layer, apply LoRA to $\mathbf{B}_{i-1}$ and $\mathbf{B}_{i+1}$, and allow the full parameters of the linear layer in $\mathbf{B}_i$ to be trainable. The structure of the LoRA injection is illustrated in supplementary materials. The collected input-output pair forms a localized training dataset. This sandwich-like configuration, where the replaced linear layer is flanked by two LoRA-enhanced blocks, is shown in the ST part in Figure 1. Each linear layer is initialized using the solution described in Section BRLL . Before pruning a new block, we reload all previously trained sandwich structures to collect updated input-output pairs from the current model. This ensures that each new sandwich block is trained with data that reflects the latest model state, promoting consistency and stability during progressive pruning. This formulation above applies to the non-overlapping case, where both adjacent blocks have not yet been replaced, i.e., $\mathbf{B}_{i-1} \notin \mathcal{R}$ and $\mathbf{B}_{i+1} \notin \mathcal{R}$, with $\mathcal{R}$ denoting the set of previously replaced blocks. This is shown in the Sandwich Training part of Figure 1.

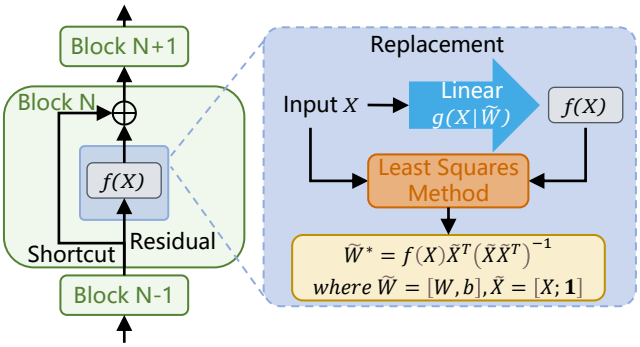

Figure 2: Illustration of the Block-wise Replacement with Linear Layers (BRLL) method. Each ResBlock consists of a residual branch and a shortcut branch. The residual branch is replaced with a linear layer, whose parameters are computed using input-output features collected during the forward pass. The shortcut connection is preserved to maintain stable information flow.

Training Sandwich Blocks with Overlap. When at least one adjacent block of the current block $\mathbf{B}_i$ has already been replaced with a linear layer, that is, $\mathbf{B}_{i-1} \in \mathcal{R}$ or $\mathbf{B}_{i+1} \in \mathcal{R}$, the resulting sandwich structure overlaps with previously replaced blocks. In such cases, we construct an extended sandwich structure by identifying the nearest unpruned blocks above and below $\mathbf{B}_i$ and applying LoRA. We define:

$$\begin{aligned} u &= \max\{j < i \mid \mathbf{B}_j \notin \mathcal{R}\}, \\ d &= \min\{j > i \mid \mathbf{B}_j \notin \mathcal{R}\}, \end{aligned} \tag{3}$$

where u and d represent the indices of the nearest unpruned blocks before and after $\mathbf{B}_i$, respectively. $\mathbf{B}_j \in \mathcal{R}$ refers to a previously replaced block. LoRA modules are applied to $\mathbf{B}_u$ and $\mathbf{B}_d$, while the linear layer in $\mathbf{B}_i$, together with all previously replaced linear layers between $\mathbf{B}_u$ and $\mathbf{B}_d$, are jointly trained using input-output data collected from the model with all prior sandwich structures loaded. The overlap-aware configuration is illustrated in the right column of the Sandwich Training section in Figure 1.

Parameter Reuse across Pruning Ratios. Our training strategy operates in a block-wise manner, where each replacement is optimized locally rather than through end-to-end fine-tuning, significantly reducing computational cost and memory consumption. Before replacing a new block, we load all previously trained sandwich structures, including linear layers and LoRA modules, and perform a forward pass to collect input–output pairs for the current block. This ensures that each stage of training reflects the latest model state, maintaining pruning stability and minimizing cumulative degradation. At inference time, we assemble the final pruned model by incorporating all trained sandwich structures. A phenomenon emerges when components trained under a higher pruning ratio model are reused in a lower pruning ratio configuration. Specifically, after training a model with a pruning ratio of $a\%$, we continue training it to a higher pruning ratio $b\%$, with $b > a$. After the higher pruning stage is completed, we extract the corresponding trained components, including linear layers and LoRA modules, and insert them into the $a\%$ model without any additional fine-tuning. This reuse consistently leads to improved inference performance compared to using only the components trained at the $a\%$ stage. We hypothesize that this improvement arises from block-level overlap across pruning stages. In higher pruning configurations, overlapping blocks participate in multiple ST setups and are therefore updated more frequently with diverse input distributions. This repeated exposure enhances their generalization ability and alignment with the global structure of the model. When reused in the lower pruning configuration, these components offer improved representational capacity and lead to better generation quality without additional training.

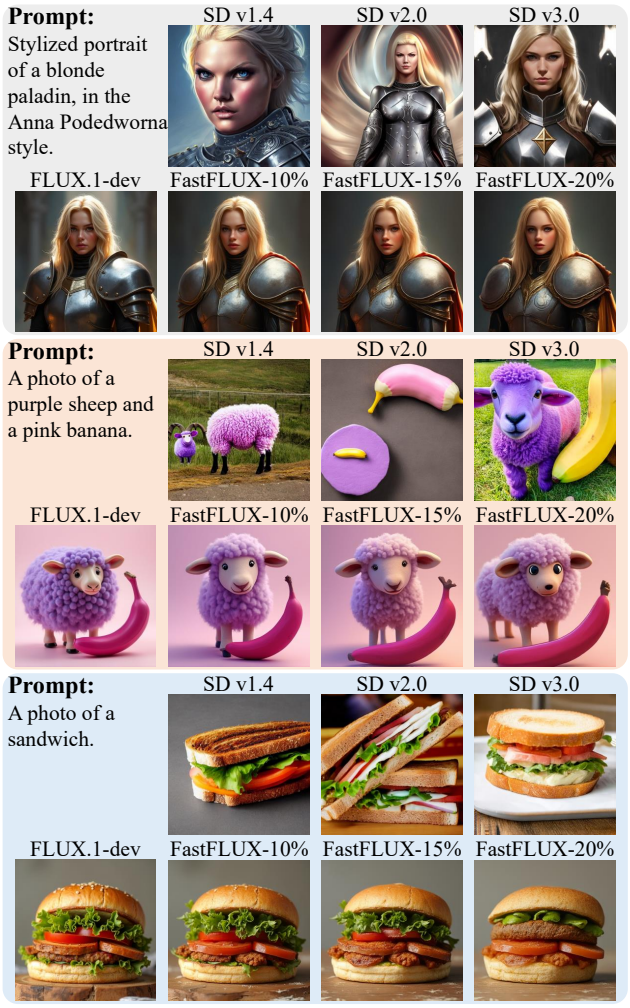

Figure 3: Qualitative comparison between FastFLUX and other mainstream T2I models under the same prompts. FastFLUX consistently generates images of higher visual fidelity. Even at different pruning ratios, the outputs of FastFLUX remain similar to those of the original FLUX.1-dev, while outperforming non-FLUX baselines in overall quality.

Experiments

Implementation Details

Our base model is FLUX.1-dev, and all experiments are conducted using the fluxdev-controlnet-16k dataset (Nar 2024), which consists of approximately 16.1k images. We sample 512 images to compute the parameters of the linear layers and another 512 images for training. Since we only need to fit the parameters of a single linear layer and a few LoRA modules at a time, only a small number of samples are required for training. We set the rank to 16 and the alpha to 8 of LoRA. To provide a comprehensive evaluation of our method, we employ a suite of text-to-image benchmarks, including HPSv2 (Wu et al. 2023) for assessing alignment with complex human preferences and GENEVAL (Ghosh, Hajishirzi, and Schmidt 2023) for short-prompt generation evaluation. All experiments are implemented in PyTorch and executed on two NVIDIA RTX 3090 GPUs using a block-swap mechanism for memory-efficient training.

Performance Comparison

Qualitative Results on Generated Samples. For qualitative evaluation, we present a visual comparison of image generation quality across different models. As shown in Figure 3, FastFLUX consistently produces images of higher quality compared to other mainstream text-to-image models. Among the FastFLUX variants, even under varying pruning ratios, the generated results remain visually similar to those from the original FLUX.1-dev model. Although higher pruning ratios may lead to the loss of fine details, the visual quality of FastFLUX remains superior to that of non-FLUX baselines. In Figure 4, we compare the visual outputs of FastFLUX and the models pruned via L1-norm.

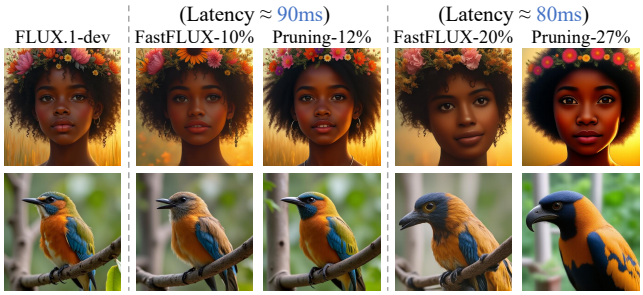

Figure 4: Qualitative comparison between FastFLUX and L1-norm-pruned variants under comparable latency budgets. Within each latency group, FastFLUX achieves better alignment with the original FLUX.1-dev output, preserving more texture and visual detail. In contrast, L1-norm-pruned models tend to lose fine-grained information, especially under higher pruning ratios. **Top row:** Oil portrait of a young black woman wearing a wildflower crown in golden light. **Bottom row:** A photo of a bird.

When inference latency is almost the same, FastFLUX is able to retain more texture and visual details, showing better alignment with the unpruned model. In contrast, the L1-norm-pruned models tend to lose more fine-grained information, especially at higher pruning ratios. These results demonstrate that FastFLUX effectively balances inference efficiency with high-fidelity image generation.

Quantitative Results on Multiple Benchmarks. We compare FastFLUX with the original FLUX.1-dev, the L1-norm-pruned variants of the FLUX model, and several state-of-the-art T2I models, including DALL·E mini (Ramesh et al. 2021), SD v1.4 (Rombach et al. 2022b), SD v2.0 (Rombach et al. 2022b), and SD v3.0 (Esser et al. 2024). Since both FastFLUX and the L1-norm method are model compression techniques, we group models by similar inference latency for fair comparison. The results are shown in Table 1 and Table 2. From the results, we observe that FastFLUX, when pruned by 10 percent, not only preserves generation quality but also outperforms the original FLUX.1-dev on the HPSv2 benchmark. Specifically, it achieves an improvement of 0.26 in score and ranks first among all evaluated T2I models. In addition, FastFLUX reduces inference latency by 7.48ms compared to the unpruned baseline. Within the same latency group, the L1-norm-pruned model removes 12 percent of the parameters but performs 0.38 lower than FastFLUX on the HPSv2 metric. Moreover, L1-based pruning requires full-model fine-tuning to adapt to the irregular parameter structure, resulting in higher training costs. In contrast, FastFLUX requires only minimal localized adaptation. These observations are consistent across other latency groups and evaluation settings. Overall, FastFLUX significantly improves inference efficiency while maintaining high-quality image generation.

Efficiency Analysis. In Table 1 and Table 2, we group models with similar inference latency together for fair comparison. The quantitative results show that although latency is comparable, the L1-norm-based pruning approaches are consistently outperformed by our proposed FastFLUX in terms of both image generation quality and pruning effectiveness. Moreover, the L1-norm method requires full model fine-tuning after pruning, which leads to significantly higher training cost. In contrast, FastFLUX achieves a better trade-off between efficiency and quality, making it a more practical and scalable solution for model acceleration.

Model	Anim.	Concept	Paint.	Photo	Avg.	Lat. (ms)↓
DALL·E mini	26.10	25.56	25.56	26.12	25.83	–
SD v1.4	27.26	26.61	26.66	27.27	26.95	–
SD v2.0	27.48	26.89	26.86	27.46	27.17	–
SD v3.0	28.58	27.87	28.15	27.72	28.08	–
FLUX.1-dev	28.94	27.90	28.16	28.28	28.32	98.26
FLUX-Pruning-12%	28.83	27.75	28.03	28.21	28.20	89.98
FastFLUX-10%	**29.04**	**28.24**	**28.49**	**28.57**	**28.58**	90.78
FLUX-Pruning-21%	28.41	27.39	27.67	27.91	27.84	85.66
FastFLUX-15%	**28.58**	**27.87**	**28.13**	**28.19**	**28.19**	85.39
FLUX-Pruning-27%	27.66	26.67	27.03	27.31	27.17	81.98
FastFLUX-20%	**28.06**	**27.48**	**27.72**	**27.77**	**27.76**	80.83

Table 1: Quantitative comparison on the HPSv2 benchmark. We group models by comparable inference latency budgets to fairly compare the effectiveness of FastFLUX and FLUX-Pruning at different compression ratios. Results show that FastFLUX achieves better or comparable performance across all categories, and even outperforms the original uncompressed FLUX.1-dev model when pruned by 10 percent. Underlined: best overall; **bold**: best within each latency group. Abbreviations: Anim., Concept, Paint., and Avg. denote Animation, Concept-art, Painting, and Averaged scores, respectively. Lat. denotes Latency.

Model	Single	Two	Count	Color	Pos.	Attr.	Overall	Lat. (ms)↓
DALL·E mini	0.73	0.11	0.12	0.37	0.02	0.01	0.23	–
SD v1.4	0.98	0.36	0.35	0.73	0.01	0.07	0.42	–
SD v2.0	0.98	0.50	0.48	0.86	0.06	0.15	0.51	–
SD v3.0	0.98	0.75	0.51	0.84	0.22	0.52	0.63	–
FLUX.1-dev	0.99	0.78	0.70	0.78	0.19	0.46	0.64	98.28
FLUX-Pruning-12%	0.98	**0.72**	0.59	**0.76**	0.15	0.40	0.60	90.00
FastFLUX-10%	**0.99**	0.69	**0.63**	0.74	**0.15**	**0.42**	**0.60**	90.85
FLUX-Pruning-21%	0.98	0.61	0.54	**0.76**	0.13	0.29	0.55	85.22
FastFLUX-15%	**0.98**	**0.65**	**0.56**	0.73	**0.13**	**0.35**	**0.57**	85.67
FLUX-Pruning-27%	0.97	0.43	0.44	**0.71**	0.09	0.19	0.47	81.59
FastFLUX-20%	**0.98**	**0.59**	**0.48**	0.68	**0.12**	**0.31**	**0.53**	80.78

Table 2: Quantitative results on the GENEVAL benchmark. We compare FastFLUX with FLUX-Pruning under different pruning ratios. Abbreviations: Single, Two, Count, Color, Pos., and Attr. denote the Single object, Two objects, Counting, Colors, Position, and Attribute binding, respectively.

Ablation Studies

Impact of Sandwich Training. As shown in Table 3, removing ST causes a notable drop in HPSv2 score from 28.58 to 27.99, resulting in a 0.59 degradation. This demonstrates that solely relying on the initial linear approximation is insufficient to fully preserve generative quality. By allowing neighboring blocks with linear replacements to participate in the training process through LoRA modules, the ST strategy introduces localized supervision that helps compensate

Method	HPSv2	CLIP Score	Latency (ms)↓
FLUX.1-dev	28.32	**31.35**	98.26
FastFLUX-10%	**28.58**	30.98	**90.78**
w/o ST	27.99	30.91	90.83

Table 3: Ablation study on the effectiveness of the Sandwich Training (ST) strategy. Without ST, the model applies BRLL to replace residual branches with linear layers using calculated parameters, but without further adaptation. As shown, omitting ST leads to a notable drop in HPSv2 performance.

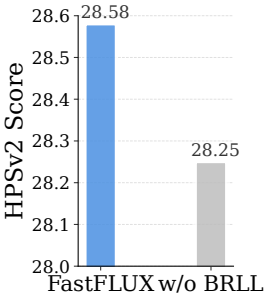

Figure 5: Comparison of HPSv2 scores with and without linear layers, under the same training strategy. The result shows that incorporating BRLL significantly improves performance.

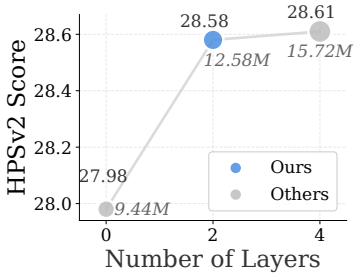

Figure 6: HPSv2 scores under different numbers of layers involved in ST. The x-axis indicates the total count of LoRA-enhanced layers. Performance improves substantially from 0 to 2 LoRA layers and saturates at 4, indicating a 2-layer structure provides a trade-off between performance and efficiency.

for performance loss caused by structural pruning. These results confirm that ST is both effective and essential for maintaining model quality under architectural modifications.

Impact of Block-wise Replacement with Linear Layer. To evaluate the necessity of block-wise replacement, we perform a variant where selected low-importance ResBlocks are directly removed without replacement. Specifically, we remove the residual branches of these blocks while preserving the shortcut connections, and apply the same ST strategy as in our full method. As shown in Figure 5, direct removal leads to a significant drop in performance compared to using linear replacements. The HPSv2 score decreases by 0.33, from 28.58 to 28.25, confirming that the linear replacement step is crucial for preserving model capability after pruning.

Number of layers in Sandwich Block. We investigate how the number of surrounding layers involved in ST affects model performance. Specifically, a setting of 0 layers corresponds to applying no LoRA modules. As shown in Figure 6, moving from 0 to 2 layers yields a substantial improvement in HPSv2 score by 0.60, demonstrating the benefit of local supervision from adjacent blocks. However, increasing from 2 to 4 layers results in only a marginal gain 0.03, while introducing a 25% increase in trainable parameters. This diminishing return, coupled with increased computational cost, makes the 2-layer configuration a favorable trade-off between performance and efficiency.

Replacement Strategy	HPSv2
Start-oriented	24.63
End-oriented	28.15
Importance-guided (FastFLUX)	**28.58**

Table 4: Comparison of different block replacement orders. Our importance-guided strategy achieves the best performance, while the model performs worst when pruning starts from the first block.

Ratio	Drop #Block	FLOPs (T)	Inference Time(ms)	HPSv2
0 (baseline)	0	38.19	98.26	28.32
5%	3	36.79 (**96.3%**)	93.24	28.23
10%	5	35.87 (**93.9%**)	90.78	**28.58**
15%	8	33.39 (**87.4%**)	85.39	28.19
20%	11	32.46 (**85.0%**)	80.83	27.76
25%	14	31.07 (**81.3%**)	76.85	27.10
30%	16	29.37 (**76.9%**)	72.30	26.70

Table 5: Performance under different block replacement ratios. As the ratio increased, the reduction in computational complexity and inference time became more significant, while the decrease in HPSv2 was relatively moderate.

Impact of the Order of Blocks to be Pruned. A key design in FastFLUX is to prioritize pruning blocks with lower estimated importance, thereby minimizing performance degradation. To assess the impact of pruning order, we compare our importance-guided strategy with two sequential baselines: pruning from the first block (*Start-oriented*) and from the last block (*End-oriented*). As shown in Table 4, the importance-based strategy achieves the highest HPSv2 score of 28.58, outperforming the other two baselines by 0.43 and 3.95 points, respectively. The results show that the choice of pruning order significantly influences model performance.

Performance with Different Pruning Ratios. We apply FastFLUX to prune the FLUX model under various pruning ratios. As shown in Table 5, while inference time and FLOPs decrease significantly with higher pruning ratios, the HPSv2 score remains relatively stable up to a moderate pruning level (e.g., 10–15%). This highlights the efficiency and robustness of the FastFLUX pruning strategy, which enables substantial acceleration with minimal quality loss.

Conclusion

In this work, we propose FastFLUX, a novel architecture-level pruning framework for efficient compression of large-scale diffusion models. Through Block-wise Replacement with lightweight Linear Layers and Sandwich Training for localized supervision, FastFLUX reduces inference cost and training overhead while maintaining high image quality. Even at a 20% pruning ratio, the model remains remarkably robust in visual fidelity and structural consistency. Extensive experiments validate the effectiveness of FastFLUX, offering a streamlined and flexible paradigm for efficient compression and acceleration of T2I diffusion models.

References

Ainslie, J.; Lee-Thorp, J.; De Jong, M.; Zemlyanskiy, Y.; Lebrón, F.; and Sanghai, S. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. arXiv:2305.13245.

Anagnostidis, S.; Pavllo, D.; Biggio, L.; Noci, L.; Lucchi, A.; and Hofmann, T. 2023. Dynamic Context Pruning for Efficient and Interpretable Autoregressive Transformers. *Advances in Neural Information Processing Systems*, 36: 65202–65223.

Avrahami, O.; Patashnik, O.; Fried, O.; Nemchinov, E.; Aberman, K.; Lischinski, D.; and Cohen-Or, D. 2025. Stable Flow: Vital Layers for Training-Free Image Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7877–7888.

Black Forest Labs. 2024. FLUX. https://github.com/black-forest-labs/flux. Official inference repo for FLUX.1 models.

Cao, H.; Tan, C.; Gao, Z.; Xu, Y.; Chen, G.; Heng, P.-A.; and Li, S. Z. 2024. A Survey on Generative Diffusion Models. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 2814–2830.

Castells, T.; Song, H.-K.; Kim, B.-K.; and Choi, S. 2024. Ld-pruner: Efficient pruning of latent diffusion models using task-agnostic insights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 821–830.

Chen, J.; Xue, S.; Zhao, Y.; Yu, J.; Paul, S.; Chen, J.; Cai, H.; Han, S.; and Xie, E. 2025. Sana-Sprint: One-Step Diffusion with Continuous-Time Consistency Distillation. arXiv:2503.09641.

Chen, J.; Yu, J.; Ge, C.; Yao, L.; Xie, E.; Wu, Y.; Wang, Z.; Kwok, J.; Luo, P.; Lu, H.; and Li, Z. 2023. PixArt-α: Fast Training of Diffusion Transformer for Photorealistic Text-to-Image Synthesis. arXiv:2310.00426.

Du, N.; Huang, Y.; Dai, A. M.; Tong, S.; Lepikhin, D.; Xu, Y.; Krikun, M.; Zhou, Y.; Yu, A. W.; Firat, O.; et al. 2022. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In *International Conference on Machine Learning*, 5547–5569.

Esser, P.; Kulal, S.; Blattmann, A.; Entezari, R.; Müller, J.; Saini, H.; Levi, Y.; Lorenz, D.; Sauer, A.; Boesel, F.; Podell, D.; Dockhorn, T.; English, Z.; Lacey, K.; Goodwin, A.; Marek, Y.; and Rombach, R. 2024. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *Proceedings of the International Conference on Machine Learning*.

Fang, G.; Ma, X.; and Wang, X. 2023. Structural Pruning for Diffusion Models. In *Advances in Neural Information Processing Systems*.

Ghosh, D.; Hajishirzi, H.; and Schmidt, L. 2023. GenEval: An Object-Focused Framework for Evaluating Text-to-Image Alignment. In *Advances in Neural Information Processing Systems*, volume 36, 52132–52152.

Gu, A.; and Dao, T. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *First Conference on Language Modeling*.

Hatamizadeh, A.; Heinrich, G.; Yin, H.; Tao, A.; Alvarez, J. M.; Kautz, J.; and Molchanov, P. 2023. FasterViT: Fast Vision Transformers with Hierarchical Attention. arXiv:2306.06189.

He, Y.; Liu, J.; Wu, W.; Zhou, H.; and Zhuang, B. 2023a. EfficientDM: Efficient Quantization-Aware Fine-Tuning of Low-Bit Diffusion Models. arXiv:2310.03270.

He, Y.; Liu, L.; Liu, J.; Wu, W.; Zhou, H.; and Zhuang, B. 2023b. Ptqd: Accurate Post-training Quantization for Diffusion Models. arXiv:2305.10657.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, 6840–6851.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the International Conference on Learning Representations*.

Kim, J.; Halabi, M. E.; Ji, M.; and Song, H. O. 2024. Layermerge: Neural Network Depth Compression through Layer Pruning and Merging. arXiv:2406.12837.

Li, M.; Lin, Y.; Zhang, Z.; Cai, T.; Li, X.; Guo, J.; Xie, E.; Meng, C.; Zhu, J.-Y.; and Han, S. 2024. SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models. arXiv:2411.05007.

Li, X.; Liu, Y.; Lian, L.; Yang, H.; Dong, Z.; Kang, D.; Zhang, S.; and Keutzer, K. 2023. Q-diffusion: Quantizing Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 17535–17545.

Lin, Y.; Zhang, T.; Sun, P.; Li, Z.; and Zhou, S. 2021. FQ-ViT: Post-Training Quantization for Fully Quantized Vision Transformer. arXiv:2111.13824.

Liu, Z.; Wang, Y.; Han, K.; Zhang, W.; Ma, S.; and Gao, W. 2021. Post-Training Quantization for Vision Transformer. *Advances in Neural Information Processing Systems*, 34: 28092–28103.

Meng, C.; Rombach, R.; Gao, R.; Kingma, D.; Ermon, S.; Ho, J.; and Salimans, T. 2023. On Distillation of Guided Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14297–14306.

Nar, K. 2024. FluxDev ControlNet 16k Dataset. Hugging Face Datasets.

Peebles, W.; and Xie, S. 2023. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4195–4205.

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2024. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. In *Proceedings of the International Conference on Learning Representations*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning*, 8748–8763.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning*, 8821–8831.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022a. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022b. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Salimans, T.; and Ho, J. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. arXiv:2202.00512.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. arXiv:1910.01108.

Sauer, A.; Lorenz, D.; Blattmann, A.; and Rombach, R. 2023. Adversarial Diffusion Distillation. arXiv:2311.17042.

Shang, Y.; Yuan, Z.; Xie, B.; Wu, B.; and Yan, Y. 2023. Post-training Quantization on Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1972–1981.

Shen, H.; Zhang, J.; Xiong, B.; Hu, R.; Chen, S.; Wan, Z.; Wang, X.; Zhang, Y.; Gong, Z.; Bao, G.; et al. 2025. Efficient Diffusion Models: A Survey. arXiv:2502.06805.

Shirkavand, R.; Yu, P.; Gao, S.; Somepalli, G.; Goldstein, T.; and Huang, H. 2025. Efficient Fine-Tuning and Concept Suppression for Pruned Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18619–18629.

So, J.; Lee, J.; Ahn, D.; Kim, H.; and Park, E. 2023. Temporal Dynamic Quantization for Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 36, 48686–48698.

Tang, Y.; Wang, Y.; Guo, J.; Tu, Z.; Han, K.; Hu, H.; and Tao, D. 2024. A Survey on Transformer Compression. arXiv:2402.05964.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training Data-Efficient Image Transformers & Distillation through Attention. In *International Conference on Machine Learning*, 10347–10357.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Wu, X.; Hao, Y.; Sun, K.; Chen, Y.; Zhu, F.; Zhao, R.; and Li, H. 2023. Human Preference Score v2: A Solid Benchmark for Evaluating Human Preferences of Text-to-Image Synthesis. arXiv:2306.09341.

Yu, L.; and Xiang, W. 2023. X-Pruner: eXplainable Pruning for Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24355–24363.

Zhang, L.; Rao, A.; and Agrawala, M. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3836–3847.

Zhang, Y.; Yang, M.; Zhou, Q.; and Wang, Z. 2024. Attention Calibration for Disentangled Text-to-Image Personalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4764–4774.

Zhu, L.; Liao, B.; Zhang, Q.; Wang, X.; Liu, W.; and Wang, X. 2024. Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model. arXiv:2401.09417.