Which Layer Causes Distribution Deviation? Entropy-Guided Adaptive Pruning for Diffusion and Flow Models

Changlin Li[1]*, Jiawei Zhang[2], Zeyi Shi[3], Zhihui Li[4], and Xiaojun Chang[4]*

[1] Stanford University, Stanford, CA, USA
[2] NCEPU, Beijing, China
[3] University of Technology Sydney, Ultimo, NSW, Australia
[4] University of Science and Technology of China, Hefei, China

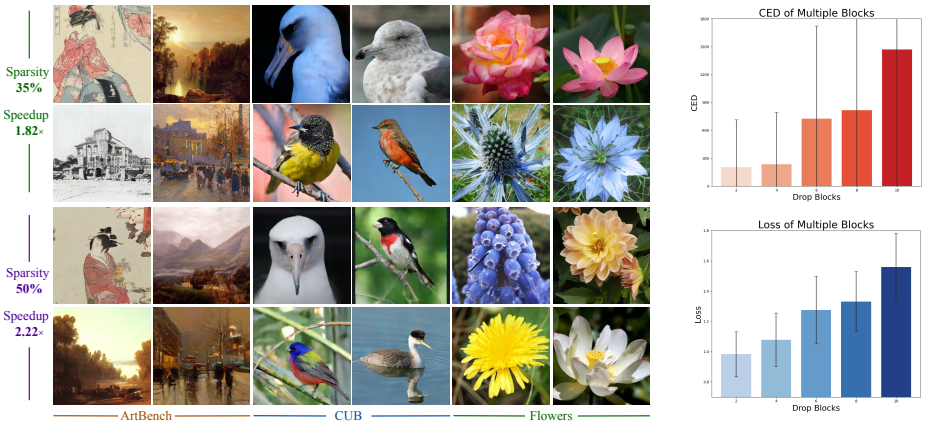
Abstract. Large-scale vision generative models, including diffusion and flow models, have demonstrated remarkable performance in visual generation tasks. However, transferring these pre-trained models to downstream tasks often results in significant parameter redundancy. In this paper, we propose EntPruner, an entropy-guided automatic progressive pruning framework for diffusion and flow models. First, we introduce entropy-guided pruning, a block-level importance assessment strategy specifically designed for generative models. Unlike discriminative models, generative models require preserving the diversity and condition-fidelity of the output distribution. As the importance of each module can vary significantly across downstream tasks, EntPruner prioritizes pruning of less important blocks using data-dependent Conditional Entropy Deviation (CED) metric. CED quantifies how much the distribution diverges from the learned conditional data distribution after removing a block. Second, we propose a zero-shot adaptive pruning framework to automatically determine when and how much to prune during training. This dynamic strategy avoids the pitfalls of one-shot pruning, mitigating mode collapse, and preserving model performance. Extensive experiments on DiT and SiT models demonstrate the effectiveness of EntPruner, achieving up to 2.22× inference speedup while maintaining competitive generation quality on ImageNet and three downstream datasets.

Keywords: Generative Models · Pruning · Efficient Inference

1 Introduction

A myriad of recent breakthroughs in diffusion and flow models have demonstrated their remarkable capabilities in image generation. The success has also been extended to audio, video, and language domains [14, 44, 55]. The Denoising Diffusion Probabilistic Model (DDPM) [12] highlighted the effectiveness of

* Equal advising. changlinli.ai@gmail.com, xjchang@ustc.edu.cn
Project page: https://github.com/changlin31/EntPruner



(a) Generated results of a series of flow models pruned by EntPruner.

(b) Effect of removing different number of blocks in SiT.

Fig. 1: (a) Generated results of a series of *flow models* pruned by EntPruner. Entpruner achieves up to 2.22× speedup while maintaining generation quality. Base model: SiT-XL/2 with ODE solver, CFG=4.0, 250 sampling steps. **(b) Effect of removing different number of blocks in SiT.** Strong correlation between *Conditional Entropy Deviation* (CED) and loss confirms CED's effectiveness in quantifying block importance in flow models.

the U-Net backbone. Recently, owing to the superior performance of Diffusion Transformers [38], studies have increasingly adopted transformer-based architectures. Lipman et al. [30] further introduced Flow Matching, a more direct and faster generative trajectory that offers an alternative perspective on training and inference for diffusion models. While diffusion models have evolved rapidly in recent years, achieving near-photorealistic quality, their practical deployment remains limited due to computational inefficiency. Recent trends in architectural design—particularly the adoption of transformer-based backbones like DiT [38] and SiT [34]—have significantly improved scalability and expressivity. However, these advancements come at the cost of increased parameter counts and memory usage. These models suffer from efficiency issues when deployed on edge devices or used in low-latency settings such as interactive applications. The high computational cost and slow inference speed of diffusion and flow models motivate us to explore more effective solutions.

To solve these problems, a majority of the arts have explored efficient pruning strategies for Stable Diffusion (SD) models [42]. BK-SDM [17] employs a heuristically handcrafted pruning scheme and leverages distillation to recover performance after pruning. However, its manual design limits transferability and requires substantial human efforts and computational costs. Diff-Pruning [9] removes filters by identifying unimportant weights through gradient analysis, but its threshold must be tuned for specific tasks, restricting practical applicability. LD-Pruner [3] introduces a novel metric to evaluate the importance of each operator and prunes unimportant convolutional and attention layers, but this

metric is task-independent. Critically, these approaches treat diffusion models as discriminative networks, failing to account for how pruning affects the generative output distribution. Their reliance on weight magnitudes or gradient-based metrics overlooks the distributional properties that define generative model quality.

These discrepancies raise a critical challenge: *how to adaptively compress diffusion models while preserving their* ***distributional expressiveness, diversity, and condition-fidelity****?* Current pruning methods fail to address this in a task-aware and scalable manner, especially when applied to transformer-based diffusion backbones. Furthermore, prior pruning work has also demonstrated a delicate trade-off between model performance and pruning ratio: aggressive pruning often leads to instability and catastrophic forgetting or distribution collapse, making the post-pruning training process inefficient. These challenges highlight the need for a efficient, and robust pruning framework that can adapt to various downstream settings while preserving the learned generative distribution.

Numerous studies have demonstrated that not all parameters in deep neural networks contribute equally to model performance [7, 10, 16, 22, 50, 52]. We observe that removing different blocks causes different types of distribution deviation in generative models. As shown in Fig. 2, the signed entropy change reveals ***two distinct patterns***: *positive values indicate* ***drift toward randomness or noise*** *(increased entropy)*, while *negative values indicate* ***mode collapse or oversimplified solutions*** *(decreased entropy)*. For instance, SiT-XL/2 exhibits critical positive deviations in mid-layers (block 6), while DiT-XL/2 shows substantial negative deviations in final layers (blocks 27-28), suggesting architectural differences in how blocks maintain distributional balance. This observation motivates us to measure the absolute change of entropy as a generative-specific importance metric that quantifies the magnitude of distribution deviation after removing each block.

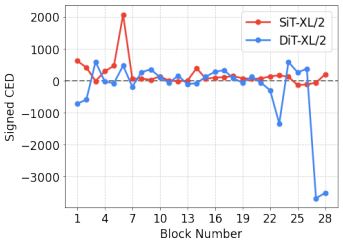


Fig. 2: Signed CED for each block in SiT-XL/2 and DiT-XL/2. The sign of CED reveals the type of distributional degradation: positive values indicate drift toward randomness/noise (increased entropy), while negative values indicate mode collapse or oversimplified solutions (decreased entropy). Critical blocks exhibit large absolute CED.

We propose a task-aware pruning framework for diffusion and flow models. The framework performs block-level structured pruning through a two-stage process: 1) Entropy-Guided Importance Estimation: We evaluate block importance in the pretrained generative model using Conditional Entropy Deviation (CED), which quantifies how much the distribution diverges from the learned conditional data distribution after removing each block. This enables us to rank blocks based on their contribution to maintaining distributional expressiveness, diversity, and condition-fidelity. 2) Zero-shot Adaptive Pruning: We propose a zero-shot adaptive pruning framework that automatically deter-

mines when and how much to prune during training. Unlike one-shot pruning that removes layers at initialization, our progressive approach adapts the pruning schedule dynamically based on training dynamics. To enable efficient pruning decisions without additional validation overhead, we integrate multiple zero-shot metrics, allowing us to reformulate pruning as a lightweight architecture search problem that can be solved with minimal computational cost.

Our framework accurately identifies the least important components for each downstream task, thereby minimizing distribution deviation from the pretrained model. It autonomously determines the optimal pruning schedule at different training stages, resulting in faster convergence while maintaining near-lossless performance. Through extensive experiments on multiple benchmark datasets and two widely-used diffusion architectures (DiT and SiT), we show that our entropy-guided pruning framework not only reduces model size and inference cost but also achieves superior or comparable generative performance compared to full fine-tuning and existing pruning baselines. As demonstrated in Fig. 1(a), EntPruner maintains high visual quality even at aggressive pruning ratios, validating that CED-guided adaptive pruning effectively preserves the learned distribution. This work bridges a gap in the compression of diffusion Transformers and offers a practical solution for their deployment in resource-constrained environments. The main contributions are summarized as follows:

- We introduce **Conditional Entropy Deviation (CED)**, a generative-specific metric that measures distribution deviation after removing each block, enabling importance ranking that preserves diversity and condition-fidelity during pruning.
- We propose **EntPruner**, a zero-shot adaptive pruning framework that automatically determines when and how much to prune during training, achieving stable convergence with minimal loss of distributional quality.
- Our method achieves an average FID drop of only 1.76 at 50% pruning rate with **2.22**× inference speedup, demonstrating effectiveness across different diffusion and flow models on multiple benchmark datasets.

2 Related Work

Diffusion Models [12, 43, 54] have achieved remarkable progress in image synthesis, with recent designs shifting from U-Nets to Transformer backbones for better scalability [4, 34, 38]. While prior works such as DiT, SiT, and PixArt-α have focused on improving generative performance in large-scale diffusion models, our work instead targets the compression and efficient deployment of these Transformer-based architectures through structured Entropy-guided pruning.

Automated Machine Learning. AutoML automates model design and optimization [8, 31, 45]. These techniques collectively reduce the need for manual intervention and enable the efficient discovery of high-performing models across diverse tasks. By simplifying the bi-level optimization problem inherent in AutoML, NAS approaches fall into multi-shot [40, 45], one-shot [11, 19–21, 39, 39,

46, 48], and zero-shot [13, 28, 51]categories. These proxies are often designed based on theoretical insights or structural analysis of neural networks, offering an efficient alternative to conventional NAS approaches [23]. Our method leverages zero-shot NAS metrics to select and prune subnetworks from pretrained diffusion Transformers, combining efficiency from AutoML with the stability of pretrained models.

Efficient Inference for Diffusion Models. The inference cost of diffusion models is primarily influenced by the number of inference steps and the computational cost. Existing researches can be broadly categorized into two directions: **1)** Reducing the number of sampling steps: DPM-Solver [33] advances the numerical approximation of diffusion ODEs. Additionally, [26, 29, 41] employed improved distillation strategy to compress the sampling steps with pre-trained models. [32, 35, 53] introduces a caching mechanism to reuse highly similar features between different sampling steps and reduce redundant computations. **2)** Compressing model architectures: works [3, 9, 17] applied structured pruning to reduce model size and works [25] adopted quantization techniques to lower both computation and memory requirements. In contrast, [5] proposes a one-shot compression approach that leverages automated pruning and erasure mechanisms to directly extract high-performing sub-networks from any given pre-trained model, without requiring additional fine-tuning. [10] proposes a Progressive Channel Pruning (PCP) framework that iteratively performs a try-select-prune strategy across multiple layers to gradually compress the network. Our work extends this line by reframing block-level pruning as a zero-shot NAS problem, achieving lightweight yet performant diffusion Transformers.

3 Entropy-Guided Progressive Pruning for Diffusion and Flow Models

3.1 Diffusion and Flow Models

Both diffusion and flow models corrupt data $\boldsymbol{x}_* \sim p(\boldsymbol{x})$ by progressively injecting noise $\epsilon \sim \mathcal{N}(0, I)$. A unified forward process can be written as $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_* + \sigma_t \epsilon$, where α_t and σ_t are time-dependent functions. In score-matching diffusion models, the process is usually defined on discrete time steps, and x_t converges to $\mathcal{N}(0, I)$ as $t \to \infty$. Flow-matching methods, in contrast, restrict the trajectory to $t \in [0, 1]$ and set $\alpha_0 = \sigma_1 = 1$ and $\alpha_1 = \sigma_0 = 0$, so that $\boldsymbol{x}_t$ converges to $\mathcal{N}(0, I)$ precisely at $t = 1$. In the reverse process, both flow matching and score-based diffusion models can be implemented by learning to invert the forward dynamics using either a stochastic differential equation (SDE) [2] or probability flow ordinary differential equation (ODE) [1].

3.2 Overview

Our goal is to enhance the efficiency of diffusion and flow models by removing blocks that cause minimal distribution deviation, thereby achieving model compression without sacrificing generative quality. To this end, we establish a

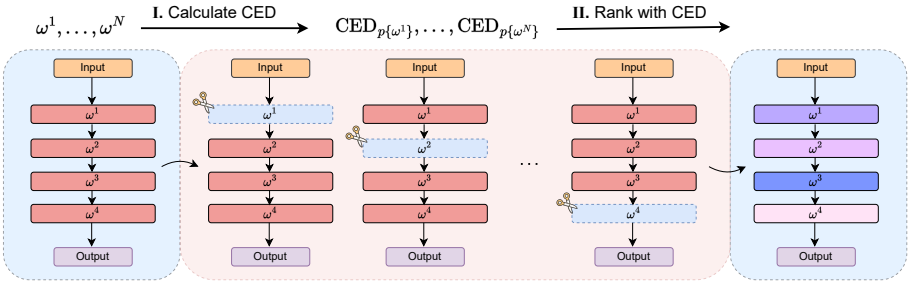


Fig. 3: Quantifying Block Importance with Conditional Entropy Deviation. Before training, we employ CED to evaluate and rank the expressiveness of individual blocks, where darker regions in the heatmap indicate stronger interactions with the overall network.

pruning priority using Conditional Entropy Deviation (CED): given a target pruning ratio r, blocks with low CED (minimal distribution deviation when removed) should be pruned first, while high-CED blocks (critical for maintaining the learned distribution) are preserved.

Prior studies have shown that aggressive one-shot pruning often disrupts the pretrained model's learned distribution, leading to irreversible quality degradation or mode collapse. Therefore, we propose a adaptive progressive pruning framework that gradually removes blocks over the training process. We define a pruning schedule Ψ that consists of a sequence of sub-networks with decreasing model sizes. Let $\mathcal{L}$ denote the task loss, Ω the model size, and $\boldsymbol{\omega}$ the model parameters. The objective of progressive pruning is:

$$\min_{\boldsymbol{\omega}, \Psi} \{\mathcal{L}(\boldsymbol{\omega}, \Psi),\ \Omega\}\,. \tag{1}$$

To reduce the complexity of optimizing over Ψ, we adopt a linear pruning schedule based on the target pruning ratio r. The pruning process is divided into k stages (default $k = 4$), with each stage consisting of $s = S/k$ training iterations. The schedule is defined as $\Psi = \{\boldsymbol{\psi}_i\}_{i=1}^{k}$, where $\boldsymbol{\psi}_k$ retains $(1 - r)\%$ of the original parameters.

We optimize the pruning schedule Ψ via zero-shot NAS [6], where each candidate subnetwork is evaluated without additional training using metrics such as NTK condition numbers and ZiCo scores [24, 48]. These metrics jointly capture convergence properties and gradient stability, enabling Ψ to automatically determine when and how much to prune while preserving model trainability and distributional quality.

3.3 Quantifying Block Importance with Conditional Entropy Deviation

Why Entropy for Generative Models? Unlike discriminative models where layer importance can be assessed through accuracy or gradient magnitudes, gen-

erative models require measuring how pruning affects the *output distribution*. For diffusion and flow models, the goal is to preserve the learned conditional distribution $p(\boldsymbol{x})$ that captures both sample quality and diversity. We leverage entropy as a natural measure of distributional uncertainty.

Entropy Quantification. Given a denoising network, let the distribution of output be denoted as $p(x)$, $x \in \mathbf{X}$, where $\mathbf{X}$ represents the output from the network. The entropy is expressed as:

$$\mathcal{H}(\mathbf{X}) = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x})\, d\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbf{X}. \tag{2}$$

For tractability, we assume that $p(\boldsymbol{x})$ follows a Gaussian distribution, *i.e.*, $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$. Eq. (2) can be written as:

$$\begin{aligned} \mathcal{H}(\mathbf{X}) &= -\mathbb{E}[\log \mathcal{N}(\mu, \sigma^2)] \\ &= -\mathbb{E}[\log[(2\pi\sigma^2)^{-1/2} \exp(-\frac{1}{2\sigma^2}(f-\mu)^2)]] \\ &= \log(\sigma) + \frac{1}{2}\log(2\pi) + \frac{1}{2} \end{aligned} \tag{3}$$

Measuring Distribution Deviation with CED. To quantify how much removing a block disrupts the learned distribution, we introduce *Conditional Entropy Deviation* (CED). As shown in Fig. 3, CED measures the absolute change in entropy after removing each block, capturing the magnitude of distribution deviation regardless of direction:

$$\begin{aligned} \mathrm{CED}_i &= |\mathcal{H}_{\text{original}}(\mathbf{X}) - \mathcal{H}_{\text{pruned}}(\mathbf{X}, i)| \\ &= |\mathcal{H}(\mathbf{X}) - \mathcal{H}(\mathbf{X} \mid \texttt{Drop}\{block_i\})|, \end{aligned} \tag{4}$$

where $\mathcal{H}(\mathbf{X})$ denotes the entropy of the original network output, and $\mathcal{H}(\mathbf{X} \mid \texttt{Drop}\{block_i\})$ represents the entropy after dropping the i-th block.

Interpreting CED. The sign of the entropy change reveals the type of distribution deviation: positive values ($\mathcal{H}_{\text{pruned}} > \mathcal{H}_{\text{original}}$) indicate drift toward randomness, while negative values ($\mathcal{H}_{\text{pruned}} < \mathcal{H}_{\text{original}}$) indicate mode collapse or oversimplified solutions. Both directions signal that the block is critical. For importance ranking, we use the absolute value CED_i: blocks with low CED exhibit minimal distribution deviation when removed (redundant), while high-CED blocks are structurally critical for maintaining the learned distribution.

3.4 Automatic Pruning via Zero-Shot Metrics.

After quantifying block importance, we gradually remove less important blocks over the training process following a adaptive pruning schedule Ψ. As shown in Fig. 4, we automatically decide the pruning schedule Ψ in a efficient way using zero-shot predictors. Assume that $\mathcal{K}$ is a zero-shot performance predictor used to estimate the loss of each candidate sub-network. The optimal sub-network at

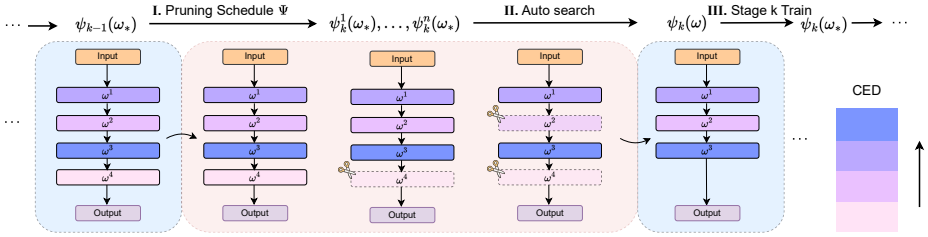


Fig. 4: Automatic Pruning via Zero-Shot Metrics. At each progressive pruning stage during training, the pruning schedule explores candidate sub-networks under different pruning ratios, selects the optimal one using zero-shot proxies, and inherits parameters from the previous stage.

stage k can be identified by solving:

$$\begin{aligned} \boldsymbol{\psi}_k^* &= \arg\min_{\boldsymbol{\psi}_k \in \Lambda_k} \{\mathcal{K}(\boldsymbol{\omega}(\boldsymbol{\psi}_k)),\ \Omega\}, \\ \text{where} \quad \Lambda_k &= \{\boldsymbol{\psi} \in \xi \mid |\boldsymbol{\omega}(\boldsymbol{\psi})| \leq |\boldsymbol{\omega}(\boldsymbol{\psi}_{k-1})|\}. \end{aligned} \tag{5}$$

where Λ_k denotes the set of candidate sub-networks whose parameter sizes are no larger than that of $\boldsymbol{\psi}_{k-1}$, the sub-network from the previous pruning stage.

Trainability via the NTK Condition Number in Flow Matching. The trainability of a neural network reflects how effectively it can be optimized via gradient descent. While larger models offer more expressivity, this does not guarantee practical trainability. The Neural Tangent Kernel (NTK) provides a useful tool for analyzing convergence in the infinite-width regime [15, 37].

In flow matching, a denoising network learns the velocity field $\mathbf{v}(t, bmx_t)$ that transports noisy samples $\boldsymbol{x}_t = \alpha_t \boldsymbol{x}_* + \sigma_t \epsilon$ toward data $\boldsymbol{x}_*$. The true velocity is $\mathbf{v}(t, \boldsymbol{x}_t) = \dot{\alpha}_t \boldsymbol{x}_* + \dot{\sigma}_t \epsilon$, and the model $\mathbf{v}_\theta(\boldsymbol{x}_t, t)$ is trained to approximate it. For a candidate sub-network with parameters $\boldsymbol{\omega}$, the update of predicted velocity satisfies

$$\Delta \mathbf{v}(\boldsymbol{x}_t) = -\eta\, \hat{\Theta}(\boldsymbol{x}_t, \boldsymbol{x}_t) \nabla \mathbf{v}(\boldsymbol{x}_t) \mathcal{L}, \tag{6}$$

where $\hat{\Theta}$ is the NTK of the velocity predictor.

In the infinite-width limit, the training dynamics are governed by the eigenvalues $\{\lambda_i\}$ of $\hat{\Theta}$. The maximum stable learning rate scales as $\eta \sim 2/\lambda_0$, while the convergence of the slowest mode depends on $1/\kappa$, where $\kappa = \lambda_0/\lambda_m$. A smaller κ indicates faster and more stable convergence. Thus, we adopt the NTK condition number as a zero-shot metric:

$$\mathcal{K}_\kappa(\boldsymbol{\psi}) = \frac{\lambda_0}{\lambda_m}. \tag{7}$$

where λ_0 and λ_m denote the largest and smallest eigenvalues of the NTK, respectively. More details are provided in the Supplementary.

Convergence Rate and Generalization Capacity via Gradient Analysis. Convergence and expressivity also directly influence the final performance

of a neural network. After a number of training steps, a network with a larger absolute mean of gradient and smaller standard deviation of gradient is generally associated with lower training loss and faster convergence. Interestingly, a smaller gradient variance often correlates with a lower maximum eigenvalue Θ of the NTK, which implies a smoother loss landscape and better generalization performance [18].

We adopt ZiCo as one of the zero-shot metrics, which jointly considers the absolute mean and standard deviation of gradient. The parameters of a candidate sub-network $\boldsymbol{\omega}$ are also inherited from the trained parameters $\boldsymbol{\omega}_*$ of the previous pruning stage. Since the ZiCo metric has been shown to correlate positively with network trainability, we introduce a negative sign to make it positively correlated with loss, allowing it to be minimized during optimization:

$$\mathcal{K}_{\text{ZiCo}}(\boldsymbol{\psi}) = -\sum_{l=1}^{N} \log\left(\sum_{\boldsymbol{\omega}\in\boldsymbol{\omega}_l} \frac{\mathbb{E}\left[|\nabla_{\boldsymbol{\omega}}\mathcal{L}^*|\right]}{\sigma\left(|\nabla_{\boldsymbol{\omega}}\mathcal{L}^*|\right)}\right). \tag{8}$$

where N denotes the number of layers in the candidate sub-network, $\boldsymbol{\omega}_l$ is the set of parameters in the l^{th} layer, $\mathcal{L}^*$ is the loss $\mathcal{L}(\boldsymbol{x}_{t,i}, \mathbf{v}_{t,i}; \boldsymbol{\omega})$, $i \in \{1, \ldots, D\}$ and D is the number of training batches, typically set to 2 to balance stability and efficiency.

3.5 Entropy-Guided Adaptive Pruning Framework

We summarize the overall algorithmic workflow of EntPruner as follows. First, we perform block-level importance ranking using CED to evaluate the contribution of each block in the pretrained model (Fig. 3). Second, we employ two zero-shot proxies to guide the pruning schedule at each training stage (Fig. 4). The optimization objective of the adaptive pruning can be reformulated from Eq. (5):

$$\boldsymbol{\psi}_k^\star = \arg\min_{\boldsymbol{\psi}_k\in\Lambda_k} \left\{\mathcal{K}_\kappa(\boldsymbol{\psi}_k),\ \mathcal{K}_{\text{ZiCo}}(\boldsymbol{\psi}_k),\ \Omega\right\}. \tag{9}$$

A key challenge lies in how to jointly optimize the two proxies. We assume both proxies are equally important for maintaining network performance and training efficiency. A common strategy is to apply a voting-based algorithm to select the optimal candidate sub-network. This approach helps mitigate differences in scale between the two metrics. Additionally, we incorporate model' parameters as a regularization term in the selection process to encourage model efficiency. The final optimization is defined as:

$$\begin{aligned} \boldsymbol{\psi}_k^\star &= \arg\min_{\boldsymbol{\psi}_k\in\Lambda_k} R(\boldsymbol{\psi}_k), \\ \text{s.t.} R(\boldsymbol{\psi}_k) &= R(\mathcal{K}_\kappa(\boldsymbol{\psi}_k)) + R(\mathcal{K}_{\text{ZiCo}}(\boldsymbol{\psi}_k)) + \gamma R(\Omega). \end{aligned} \tag{10}$$

where $R(\cdot)$ denotes the ranking score (*e.g.*, 1st, 2nd, ..., R-th) and γ is the regularization factor, set to 0.5. The candidate with the lowest rank in each individual metric receives the smallest score, and the sub-network $\boldsymbol{\psi}_k^\star$ with the lowest total rank is selected for the next training stage. Please refer to the Supplementary for the Algorithm of EntPruner.

Fig. 5: Qualitative comparison of *flow models* pruned by different methods. Base model is SiT-XL/2, with 35% pruning rate. Datasets are Flowers (column 1-2), CUB (column 3-4), and ArtBench (column 5-6).

4 Experiment

4.1 Implementation Details

Training. We evaluate the effectiveness of our proposed method on DiT-XL/2 and SiT-XL/2 models with an image resolution of 256×256. We conduct experiments on both diffusion-based DiT and flow-matching-based SiT. All experiments are conducted on a computing platform equipped with 8 NVIDIA A800 GPUs (80 GB), with DiT trained for 240K steps and SiT for 60K steps. We fine-tune SiT and DiT on downstream tasks following the configuration in [34, 49].
Evaluation. For the DiT model, we adopt the DDPM sampler, while for SiT, we employ an ODE solver and SDE solver. We report results based on 50 sampling steps. To assess generative quality, we utilize two widely adopted evaluation metrics: Fréchet Inception Distance (FID) and Inception Score (IS). We measure efficiency using inference latency and the number of parameters.
Datasets. We evaluate our pruning method on ImageNet and three fine-grained image datasets: CUB-200-2011 [47], Oxford Flowers [36], and ArtBench-10 [27]. Notably, ArtBench-10 exhibits a distribution that significantly differs from ImageNet, allowing us to comprehensively evaluate the generalization performance of EntPruner on out of distribution tasks.

4.2 Entropy-Guided Adaptive Pruning on Downstream Datasets

Class to Image Generation with Flow Models. To evaluate the effectiveness of our approach, we compare it with LD-Pruner [3] on the SiT architecture. As shown in Tab. 1, our method consistently outperforms LD-Pruner at both 35% and 50% pruning ratios, when pruning 35% of the parameters, we achieve 3.9%, 2.2%, and 6.9% FID improvements on the three benchmark datasets with an ODE sampler. Notably, on Flowers dataset, our method surpasses full fine-tuning when pruning 35% of the parameters, achieving the best overall performance. We observe slight variations across different samplers, with the SDE

Table 1: Comparison of *flow models* pruned by different methods. We use SiT as the base model and evaluate with both ODE and SDE samplers on three datasets. ↓ and ↑ indicate whether lower or higher values are better.

Method	Sparsity	CUB		Flowers		ArtBench		Params (M)	Speedup
		FID↓	IS↑	FID↓	IS↑	FID↓	IS↑		
w/ ODE sampler									
Fine-tuning	/	5.32	6.02	11.78	3.71	8.80	7.32	675.12	×1
LD-Pruner	35%	5.70	6.03	12.02	3.75	10.78	6.63	435.78	×1.82
Ours	35%	**5.48**	**6.07**	**11.75**	**3.82**	**10.03**	**6.88**	**435.78**	**×1.82**
LD-Pruner	50%	6.86	6.16	12.09	3.79	12.81	6.35	334.67	×2.22
Ours	50%	**6.68**	**6.18**	**11.86**	**3.82**	**12.65**	**6.41**	**334.67**	**×2.22**
w/ SDE sampler									
Fine-tuning	/	5.17	5.87	12.47	3.77	13.33	6.56	675.12	×1
LD-Pruner	35%	5.24	6.10	12.32	3.74	16.16	6.20	**435.78**	×1.49
Ours	35%	**5.22**	**6.11**	**12.10**	**3.74**	**15.25**	**6.31**	**435.78**	**×1.49**
LD-Pruner	50%	5.98	6.19	12.92	3.75	18.74	5.90	**334.67**	×1.85
Ours	50%	**5.83**	**6.15**	**12.77**	**3.77**	**18.69**	**5.90**	**334.67**	**×1.85**

Table 2: Comparison of *diffusion models* trained by different methods. We use DiT-XL/2 as the base model. Results are measured in FID.

Method	Sparsity	CUB	Flowers	ArtBench	Params (M)	Speedup
Full Fine-tuning	/	5.68	21.05	25.31	673.8	×1
Ours	30%	5.50	**11.99**	24.99	**471.66**	**×1.33**
Adapt-Parallel	-	7.73	21.24	38.43	678.08	-
Adapt-Sequential	-	7.00	21.36	35.04	678.08	-
BitFit	-	8.81	20.31	24.53	674.41	-
VPT-Deep	-	17.29	25.59	40.74	676.61	-
LoRA-R8	-	56.03	164.13	80.99	674.94	-
LoRA-R16	-	58.25	161.68	80.72	675.98	-
DiffFit	-	**5.48**	20.18	**20.87**	674.63	-

sampler exhibiting longer sampling times compared to the ODE. Nevertheless, our method consistently outperforms LD-Pruner regardless of the sampling strategy. While both pruning methods exhibit performance degradation on ArtBench, likely due to the substantial distribution gap between ArtBench and ImageNet, EntPruner still outperforms LD-Pruner, demonstrating strong robustness and superior generalization.

We qualitatively compare the image generation quality of EntPruner and LD-Pruner. We use the same random seed, set the classifier-free guidance scale to 4.0, and adopt 250 sampling steps. As shown in Fig. 5, our method consistently produces images with finer details and better quality. On the CUB and Oxford Flowers datasets, EntPruner tends to generate close-up views of the subject, enriching detail and enhancing aesthetic quality. On the ArtBench dataset, our approach captures more realistic textures in structures such as houses and trees, resulting in improved visual fidelity compared to LD-Pruner.

Fig. 6: Generated results of a series of *diffusion models* pruned by EntPruner. Base model is DiT-XL/2. The pruning rates are set to 30%. Inference is performed on the CUB (column 1-2), ArtBench (columns 3-4), and Flowers (columns 5-6) datasets, with the classifier-free guidance coefficient set to 4.0.

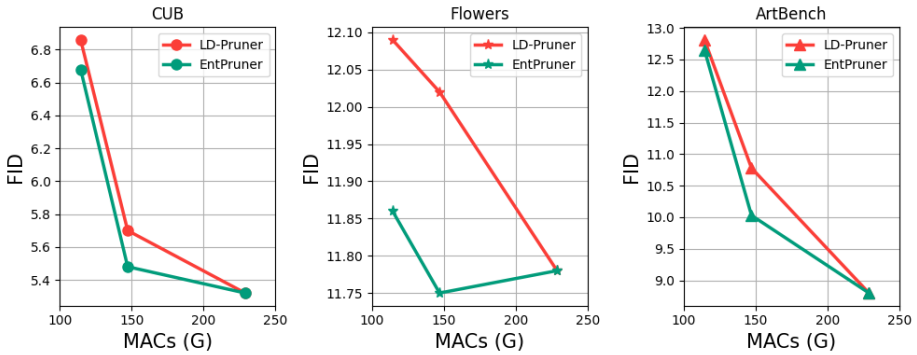


Fig. 7: Tradeoff between computational cost (MACs) and generative quality (FID). Rightmost point: full model.

Class to Image Generation with Diffusion Models. We apply EntPruner on DiT and compare with full fine-tuning. As shown in Tab. 2, our method outperforms full fine-tuning across all three datasets. Notably, on Flowers, EntPruner reduces the FID relatively by **43.04%**. Furthermore, EntPruner improves inference efficiency, achieving a **1.33**× speedup compared to full fine-tuning. In addition, we benchmark EntPruner against *state-of-the-art* fine-tuning strategies. Our method matches or surpasses the performance of efficient fine-tuning baselines, while offering **1.33**× faster inference. Example of generated images are shown in Fig. 6. On the CUB dataset, the generated results exhibit fine-grained feather textures and natural color transitions. On the ArtBench dataset, our method produces artwork with complete scene structure and stylistic consistency. On the Flowers dataset, the generated results display vibrant and well-separated color tones and clear petal boundaries.

Fig. 8: Qualitative comparison of *flow models* pruned by different methods on ImageNet 256×256. Base model is SiT with 30% pruning rate. Our EntPruner consistently produces more photorealistic images with *fewer generation artifacts*.

Table 3: Comparison of *flow models* pruned by different methods on ImageNet 256×256. Base model is SiT with 30% pruning rate. We also include full model performance (marked with gray) of different generative models as a reference.

Method	Params (M)	FID↓	sFID↓	IS↑	Precision↑	Recall↑	MACs	Speedup
BigGAN-deep	112	6.95	7.36	171.4	0.87	9.28	-	-
AR w/ VQGAN	227	26.52					-	-
MaskGIT	227	6.18	-	182.1	-	-	-	-
ADM	554	10.94	6.02	100.98	0.69	0.63	-	-
LDM-4-G	400	3.60	-	247.67	0.87	0.48	-	-
DiT-XL/2 (DDPM)	675.12	2.27	4.60	258.09	0.81	0.60	228.85	×0.43
SiT-XL/2 (ODE)	675.12	2.15	4.60	258.09	0.81	0.60	228.85	×1
BK-SDM (SiT,ODE)	471.66	3.48	4.94	209.93	0.78	**0.60**	163.48	×1.33
Diff-Pruning (SiT,ODE)	471.66	2.79	4.80	223.79	0.79	0.58	163.48	×1.33
LD-Pruner (SiT,ODE)	471.66	6.81	4.82	148.29	0.75	0.57	163.48	×1.33
Ours (SiT,ODE)	**471.66**	**2.69**	**4.75**	**225.05**	**0.80**	0.59	**163.48**	**×1.33**

Inference Efficiency Across Parameter Budgets. Fig. 7 compares the FID scores *vs.* computational complexity (measured by MACs) of EntPruner and LD-Pruner across three datasets: CUB, Flowers, and ArtBench. Notably, EntPruner consistently outperforms LD-Pruner across all configurations. At medium complexity, the gap between the two is most obvious. Tab. 1 and Fig. 7 jointly demonstrate that EntPruner maintains lower FID scores even under reduced MACs on CUB and Flowers datasets, demonstrating better robustness in resource-constrained settings. On ArtBench dataset, the performance gap between the two methods is most pronounced, further highlighting EntPruner's superior generalization ability and pruning efficiency for complex image generation tasks. See the Supplementary for more details.

4.3 Entropy-Guided Adaptive Pruning on ImageNet 256×256

To further demonstrate the effectiveness and robustness of our pruning method on pretrained models, we directly apply our pruning strategy to compress models

pretrained on ImageNet 256×256. The experimental setup follows the same configuration as Sec. 4.1, where SiT is trained using flow matching. During sampling, we employ an ODE solver and adhere to standard evaluation protocols [34, 38]. BK-SDM, Diff-Pruning, and LD-Pruning are used as baseline methods. BK-SDM is specifically designed for SD based on heuristic strategies. We treat the first 14 blocks of SiT as the encoder and the last 14 blocks as the decoder, pruning them in pairs with every two blocks forming one group. All three methods perform depth pruning at the block level.

As shown in Tab. 3, with a pruning ratio of 30% on ImageNet, our method achieves a final FID of **3.53**, only a degradation of 1.38 compared to the original SiT-XL/2. Notably, it surpasses the recent pruning method LD-Pruner by 60.5% and completely outperforms other pruning methods, further validating our method's ability to mitigate parameter collapse often caused by one-shot pruning. Moreover, in terms of inference speed, our method achieves a 1.33× speedup compared to SiT, and a **209.30%** speed improvement compared to DiT, highlighting its practical inference efficiency.

As shown in Fig. 8, we present qualitative results demonstrating the impact of pruning on image generation quality. It is evident that pruning with LD-Pruner leads to significant degradation in visual fidelity. For instance, generated images exhibit missing features such as the eyes of a dog, fine details of a raccoon, aesthetically pleasing text on hot air balloons, well-structured cakes, and the tusks of an elephant. In contrast, our method preserves more structural details and visual coherence. By pruning at more appropriate stages, our approach enables the final model to maintain robust generative performance while retaining essential semantic content.

4.4 Ablation Study

Entropy-Guided CED ranking. We perform an ablation study to evaluate the effectiveness of entropy-guided importance ranking. Random pruning leads to high variance and instability, so we prune blocks with the highest Conditional Entropy Deviation, which are identified as most critical by our metric. As illustrated in Tab. 4, pruning blocks with high entropy causes a substantial performance drop, which cannot be fully recovered even through continued fine-tuning. which proves that Conditional Entropy Deviation is a reliable indicator of parameter importance in pretrained networks.

Adaptive Pruning. We conduct an ablation study on automated pruning. The results are presented in Tab. 4, demonstrating that one-shot pruning or premature pruning significantly degrades model performance. In contrast, our method autonomously determines both the timing and extent of pruning, allowing the model to maintain competitive performance, or even achieving better results than full fine-tuning in some cases.

Table 4: Ablation studies on CED ranking and adaptive pruning. Results are reported with SiT on Flowers. Latency denotes the sampling time for an images executed on a single A800 GPU.

Method	FID ↓	IS ↑	Params (M)	MACs	latency (s)	Speedup
Full Fine-tuning	11.78	3.71	675.12	228.85	0.20	×1
w/o CED	12.06	3.81	435.78	147.13	0.09	×1.82
w/o Ada. Pruning	11.84	3.80	435.78	147.13	0.09	×1.82
Ours	**11.75**	**3.82**	**435.78**	**147.13**	**0.09**	**×1.82**

5 Conclusion

We present EntPruner, an entropy-guided adaptive pruning framework specifically designed for diffusion and flow models. By introducing Conditional Entropy Deviation (CED), we provide a generative-specific metric that measures distribution deviation after removing each block—capturing both drift toward randomness and mode collapse. Our zero-shot adaptive pruning framework automatically determines when and how much to prune during training, preserving distributional quality, diversity, and condition-fidelity while achieving up to 2.22× inference speedup. Extensive experiments demonstrate that our method maintains generation quality comparable to full fine-tuning across multiple benchmarks.
Societal Impacts. The ease of deployment of generative model may enable misuse in unregulated or adversarial scenarios. We plan to limit the code and model access for research only purposes to address potential societal impacts.

Acknowledgment

This work was partially supported by New Generation Artificial Intelligence-National Science and Technology Major Projection (2025ZD0123100) and by The National Natural Science Foundation of China (NSFC) under no. 62573399 and U25A20530.

References

1. Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E.: Stochastic interpolants: A unifying framework for flows and diffusions. arXiv preprint arXiv:2303.08797 (2023)
2. Albergo, M.S., Vanden-Eijnden, E.: Building normalizing flows with stochastic interpolants. arXiv preprint arXiv:2209.15571 (2022)
3. Castells, T., Song, H.K., Kim, B.K., Choi, S.: 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. pp. 821–830 (2024)
4. Chen, J., YU, J., GE, C., Yao, L., Xie, E., Wang, Z., Kwok, J., Luo, P., Lu, H., Li, Z.: Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In: The Twelfth International Conference on Learning Representations (2024)

5. Chen, T., Ding, T., Zhu, Z., Chen, Z., Wu, H., Zharkov, I., Liang, L.: Otov3: Automatic architecture-agnostic neural network training and compression from structured pruning to erasing operators. arXiv preprint arXiv:2312.09411 (2023)
6. Chen, T., Liang, L., Ding, T., Zhu, Z., Zharkov, I.: Otov2: Automatic, generic, user-friendly. arXiv preprint arXiv:2303.06862 (2023)
7. Cheng, H., Zhang, M., Shi, J.Q.: A survey on deep neural network pruning-taxonomy, comparison. Analysis, and Recommendations (2023)
8. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 113–123 (2019)
9. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16091–16101 (2023)
10. Guo, J., Zhang, W., Ouyang, W., Xu, D.: Model compression using progressive channel pruning. IEEE Transactions on Circuits and Systems for Video Technology **31**(3), 1114–1124 (2020)
11. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. In: ECCV (2020)
12. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
13. Huang, M., Huang, Z., Li, C., Chen, X., Xu, H., Li, Z., Liang, X.: Arch-graph: Acyclic architecture relation predictor for task-transferable neural architecture search. In: CVPR (2022)
14. Huang, R., Huang, J., Yang, D., Ren, Y., Liu, L., Li, M., Ye, Z., Liu, J., Yin, X., Zhao, Z.: Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In: International Conference on Machine Learning. pp. 13916–13932. PMLR (2023)
15. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. Advances in neural information processing systems **31** (2018)
16. Jiang, Z., Li, C., Chang, X., Chen, L., Zhu, J., Yang, Y.: Dynamic slimmable denoising network. IEEE Transactions on Image Processing (2023)
17. Kim, B.K., Song, H.K., Castells, T., Choi, S.: Bk-sdm: A lightweight, fast, and cheap version of stable diffusion. In: European Conference on Computer Vision. pp. 381–399. Springer (2024)
18. Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., Gur-Ari, G.: The large learning rate phase of deep learning: the catapult mechanism. arXiv preprint arXiv:2003.02218 (2020)
19. Li, C., Lin, S., Tang, T., Wang, G., Li, M., Liang, X., Chang, X.: Bossnas family: Block-wisely self-supervised neural architecture search. IEEE Transactions on Pattern Analysis and Machine Intelligence **47**(5), 3500–3514 (2025)
20. Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X.: Block-wisely supervised neural architecture search with knowledge distillation. In: CVPR (2020)
21. Li, C., Tang, T., Wang, G., Peng, J., Wang, B., Liang, X., Chang, X.: Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In: ICCV (2021)
22. Li, C., Zhang, J., Liu, S., Lin, S., Shi, Z., Li, Z., Chang, X.: Efficient training for human video generation with entropy-guided prioritized progressive learning. In: CVPR. pp. 5967–5977 (2026)

23. Li, G., Hoang, D., Bhardwaj, K., Lin, M., Wang, Z., Marculescu, R.: Zero-shot neural architecture search: Challenges, solutions, and opportunities. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
24. Li, G., Yang, Y., Bhardwaj, K., Marculescu, R.: Zico: Zero-shot nas via inverse coefficient of variation on gradients. arXiv preprint arXiv:2301.11300 (2023)
25. Li, X., Liu, Y., Lian, L., Yang, H., Dong, Z., Kang, D., Zhang, S., Keutzer, K.: Q-diffusion: Quantizing diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17535–17545 (2023)
26. Li, Y., Wang, H., Jin, Q., Hu, J., Chemerys, P., Fu, Y., Wang, Y., Tulyakov, S., Ren, J.: Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. Advances in Neural Information Processing Systems **36**, 20662–20678 (2023)
27. Liao, P., Li, X., Liu, X., Keutzer, K.: The artbench dataset: Benchmarking generative models with artworks. arXiv preprint arXiv:2206.11404 (2022)
28. Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., Jin, R.: Zen-nas: A zero-shot nas for high-performance image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 347–356 (2021)
29. Lin, S., Wang, A., Yang, X.: Sdxl-lightning: Progressive adversarial diffusion distillation. arXiv preprint arXiv:2402.13929 (2024)
30. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
31. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV). pp. 19–34 (2018)
32. Liu, X., Li, Z., Gu, Q.: Cachequant: Comprehensively accelerated diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 23269–23280 (2025)
33. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. Advances in Neural Information Processing Systems **35**, 5775–5787 (2022)
34. Ma, N., Goldstein, M., Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E., Xie, S.: Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In: European Conference on Computer Vision. pp. 23–40. Springer (2024)
35. Ma, X., Fang, G., Bi Mi, M., Wang, X.: Learning-to-cache: Accelerating diffusion transformer via layer caching. Advances in Neural Information Processing Systems **37**, 133282–133304 (2024)
36. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian conference on computer vision, graphics & image processing. pp. 722–729. IEEE (2008)
37. Novak, R., Sohl-Dickstein, J., Schoenholz, S.S.: Fast finite width neural tangent kernel. In: International Conference on Machine Learning. pp. 17018–17044. PMLR (2022)
38. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4195–4205 (2023)
39. Peng, J., Zhang, J., Li, C., Wang, G., Liang, X., Lin, L.: Pi-nas: Improving neural architecture search by reducing supernet training consistency shift. In: ICCV (2021)
40. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)

41. Ren, Y., Xia, X., Lu, Y., Zhang, J., Wu, J., Xie, P., Wang, X., Xiao, X.: Hyper-sd: Trajectory segmented consistency model for efficient image synthesis. arXiv preprint arXiv:2404.13686 (2024)
42. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
43. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: 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. pp. 22500–22510 (2023)
44. Sahoo, S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J., Rush, A., Kuleshov, V.: Simple and effective masked diffusion language models. Advances in Neural Information Processing Systems **37**, 130136–130184 (2024)
45. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: CVPR (2019)
46. Tang, T., Li, C., Wang, G., Yu, K., Chang, X., Liang, X.: Learning self-regularized adversarial views for self-supervised vision transformers. arXiv preprint arXiv:2210.08458 (2022)
47. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
48. Wang, G., Li, C., Yuan, L., Peng, J., Xian, X., Liang, X., Chang, X., Lin, L.: Dna family: Boosting weight-sharing nas with block-wise supervisions. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(5), 2722–2740 (2023)
49. Xie, E., Yao, L., Shi, H., Liu, Z., Zhou, D., Liu, Z., Li, J., Li, Z.: Difffit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4230–4239 (2023)
50. Yang, B., Deng, X., Shi, H., Li, C., Zhang, G., Xu, H., Zhao, S., Lin, L., Liang, X.: Continual object detection via prototypical task correlation guided gating mechanism. In: CVPR (2022)
51. Yang, J., Liu, Y.: Etas: Zero-shot transformer architecture search via network trainability and expressivity. In: Findings of the Association for Computational Linguistics ACL 2024. pp. 6780–6795 (2024)
52. Yang, M., Lin, S., Li, C., Chang, X.: Let llm tell what to prune and how much to prune. In: ICML (2025)
53. Zhang, H., Gao, T., Shao, J., Wu, Z.: Blockdance: Reuse structurally similar spatio-temporal features to accelerate diffusion transformers. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 12891–12900 (2025)
54. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3836–3847 (2023)
55. Zhu, S., Chen, J.L., Dai, Z., Dong, Z., Xu, Y., Cao, X., Yao, Y., Zhu, H., Zhu, S.: Champ: Controllable and consistent human image animation with 3d parametric guidance. In: European Conference on Computer Vision. pp. 145–162. Springer (2024)