GROW, DON'T PRUNE: COMPUTE-EFFICIENT TRAINING OF DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion Transformers (DiTs) achieve strong generation quality but are expensive to train and evaluate. Existing dynamic-width methods reduce inference cost by pruning a pretrained dense model, leaving its pretraining cost unchanged. We propose GrowDiT, a dynamic-growth training strategy that starts from a narrow network and progressively activates attention heads and MLP channel groups across diffusion noise levels. A gradient-based need score selects which units to activate under an increasing compute budget. By reducing the FLOPs of early training updates, GrowDiT reallocates the saved computation to additional optimization steps. On ImageNet-256 and FFHQ-256, under a fixed training-FLOP budget and approximately matched inference FLOPs, GrowDiT consistently outperforms fixed-width, random-growth, and dense-training baselines across DiT-S/2, DiT-B/2, and DiT-L/2. On the 130M-parameter DiT-B/2, it improves FID by 7% over fixed-width training while using the same nominal training and inference FLOPs. Matching budgets by wall-clock rather than FLOPs gives a smaller 5.3% on DiT-B/2 and 1.1% on DiT-S/2. GrowDiT also outperforms depth-growth configurations and a DyDiT-inspired train-then-prune baseline. These results suggest that allocating width during training is a promising approach to improving the training efficiency of diffusion transformers.

1 INTRODUCTION

Diffusion models have become a leading approach to generative modeling (Sohl-Dickstein et al., 2015; Ho et al., 2020; Nichol & Dhariwal, 2021; Song et al., 2021b; Dhariwal & Nichol, 2021; Rombach et al., 2022), with diffusion transformers (DiTs) widely adopted as backbones for high-quality generation (Peebles & Xie, 2023; Bao et al., 2023; Ma et al., 2024a). As denoising proceeds, the task shifts from recovering coarse structure at high noise levels to refining fine details at low noise levels (Balaji et al., 2022; Ganjdanesh et al., 2024). A standard DiT, however, uses the same amount of computation at every timestep. DyDiT (Zhao et al., 2025) identifies substantial computational redundancy in standard DiTs and reduces it by learning a timestep-dependent pruning configuration for attention heads and MLP channel groups, while allowing some tokens to bypass MLP computation. Fine-tuning a pre-trained DiT in this way lowers inference FLOPs while maintaining competitive generation quality.

DyDiT reduces inference cost, but still requires the full model to be pretrained. We propose dynamic-growth-based DiT (GrowDiT): start with fewer active attention heads and MLP channel groups, then activate units as needed across diffusion noise levels. A gradient-based score guides these additions under the compute constraint. The lower FLOP cost of early steps allows more optimization steps within the same training budget.

We evaluate GrowDiT, on ImageNet and FFHQ at up to three model scales, approximately matching inference compute, and training compute except for *train-then-prune* at $1.1\times$. GrowDiT achieves the lowest FID among the evaluated baselines. On DiT-B, GrowDiT reduces FID by 7% relative to a compute-matched baseline with uniform capacity across noise levels and by 43% relative to a train-then-prune pipeline using DyDiT's routers. In addition, it also reduces FID by 9–11% relative to depth growth on DiT-S. The gains increase with model scale.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Diffusion transformers and efficient generation. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) with transformer backbones (Peebles & Xie, 2023; Bao et al., 2023; Ma et al., 2024a; Chen et al., 2024) achieve strong generation performance but are computationally expensive. Efficient generation methods include samplers requiring fewer model evaluations (Song et al., 2021a; Lu et al., 2022), distillation and consistency-based methods for few-step sampling (Salimans & Ho, 2022; Song & Dhariwal, 2024), and caching or token merging to reduce redundant computation (Ma et al., 2024b; Fang et al., 2025). These methods target sampling cost, whereas we allocate capacity during training under a fixed compute budget, so few-step generation is orthogonal to our contribution rather than an alternative to it.

Efficient training of diffusion transformers. Other work reaches a given quality with less compute: MaskDiT masks most patches and TREAD routes a subset of tokens past intermediate layers, both during training (Zheng et al., 2024; Krause et al., 2025), REPA aligns hidden states with features from a pretrained encoder (Yu et al., 2025), Min-SNR reweights the loss across timesteps (Hang et al., 2023), and scaling laws and μP transfer guide how a run is sized and parameterized (Liang et al., 2026; Zheng et al., 2025). These approaches improve training efficiency without reducing inference cost. Our method reduces both training and inference compute and can be used alongside these training-efficiency methods.

Dynamic architectures. Compression and pruning remove redundant parameters or components (Han et al., 2015b;a; Frankle & Carbin, 2019; Fang et al., 2023), while dynamic networks adapt computation to inputs or intermediate features (Han et al., 2021; Graves, 2016). In diffusion models, DyDiT (Zhao et al., 2025) fine-tunes a pre-trained DiT with timestep-dependent pruning and token-wise MLP bypassing. Sparse mixture-of-experts models scale capacity by routing inputs through selected experts (Shazeer et al., 2017; Fedus et al., 2022; Ruiz et al., 2021); diffusion variants further adapt expert use to denoising inputs (Shi et al., 2025; Cheng et al., 2025). Other methods specialize denoisers, subnetworks, or LoRA experts by noise range (Ganjdanesh et al., 2024; Balaji et al., 2022; Zhuang et al., 2025), dynamically skip blocks and adjust MLP width (Wang et al., 2026), or vary mediator tokens to reduce attention cost (Pu et al., 2024). These methods use dynamic computation for pruning, sparse scaling, stage specialization, or efficient inference. Our method instead allocates active width as training proceeds, reducing early training computation while retaining the learned allocation at inference.

Growing networks. Network growth ranges from function-preserving transformations to progressive training (Chen et al., 2015; Karras et al., 2018). For transformers and language models, one approach initializes larger models from smaller ones (Gong et al., 2019; Chen et al., 2022; Wang et al., 2023; Du et al., 2024). Scaling laws have been used to decide when to expand (Shen et al., 2022), while gradients can guide which units to add or how to initialize them (Evci et al., 2022; Wu et al., 2020), and attention entropy can guide where to add depth (Chen et al., 2026). Growth can also be scheduled under a fixed training budget (zhuofan xia et al., 2023), or applied to input tokens rather than capacity (Huang et al., 2024). MorphNet sets layer widths under a FLOP constraint by alternating sparsification and uniform re-expansion (Gordon et al., 2018). In continual learning, expansion helps models accommodate new tasks (Rusu et al., 2016; Yoon et al., 2018; Wu et al., 2020; Douillard et al., 2022). For diffusion models, SANA 1.5 expands a pre-trained model in depth to reduce the cost of training a larger one (Xie et al., 2025). Our method instead grows model width across different noise levels under a fixed training compute budget.

3 Dynamic-Growth Training for DiT

3.1 Setup and notation

A DiT (Peebles & Xie, 2023) consists of L transformer blocks with hidden dimension d. Each block contains H attention heads of dimension $d_h = d/H$ and an MLP with hidden dimension $4d$. Following (Zhao et al., 2025), we divide the MLP hidden channels into G equal groups. We refer to attention heads and MLP channel groups as *units* and control their activation across noise levels. Specifically, we divide the diffusion timesteps $\{0, \ldots, 999\}$ into B equal *bins*. Bin 0 contains the lowest-noise timesteps, and bin $B-1$ contains the highest. Fig. 1 provides an overview of the method.

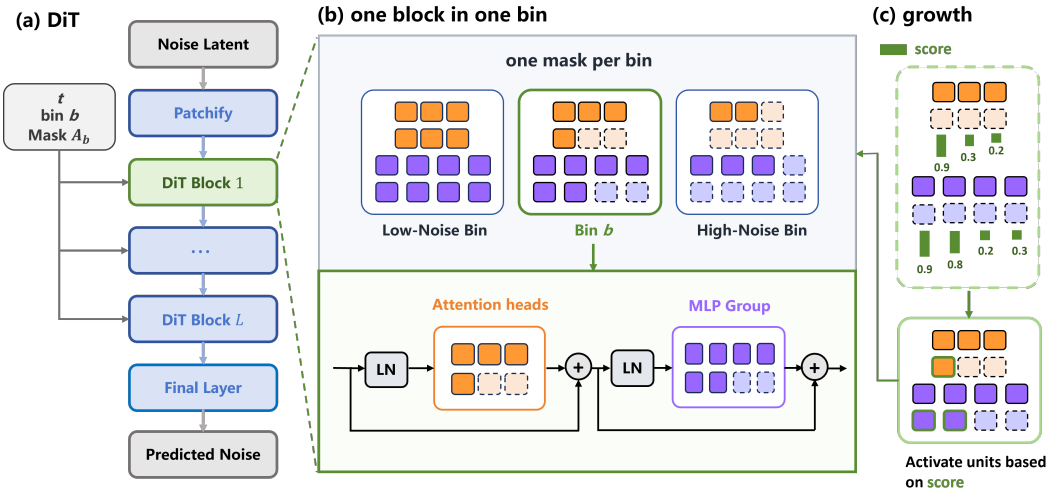


Figure 1: Overview of dynamic-growth training for DiT. (a) timestep t selects bin b and its mask for each DiT block. (b) the mask determines which attention heads (orange) and MLP groups (purple) run. (c): every K budget units, inactive units are ranked by need score per unit of added cost. The selected units are activated with gates initialized to $\alpha = 0$.

3.2 Per-bin masks

Each unit can be activated or deactivated independently in each bin: a binary mask $A \in \{0,1\}^{L\times B\times(H+G)}$ indicates whether unit u of block l is active in bin b, and a gate $\alpha_{l,b,u}$ scales its output. For a timestep in bin b, block l computes attention only over its active heads $\mathcal{H}_l^b = \{h : A_{l,b,h}{=}1\}$,

$$\mathrm{Attn}_l^b(x) \;=\; \sum_{h\in\mathcal{H}_l^b} \alpha_{l,b,h}\,\mathrm{head}_h(x)\,W_O^{(h)}, \tag{1}$$

where $\mathrm{head}_h(x) \in \mathbb{R}^{N\times d_h}$ is a standard self-attention head and $W_O^{(h)} \in \mathbb{R}^{d_h\times d}$ is the h-th block of the output projection. The MLP similarly runs only its active groups, each scaled by its gate. All bins share the same weights but use different activation configurations. The trainable parameters are the block weights and the gates α, one scalar per unit and bin, all updated by backpropagation; the mask A is not trained but set by the growth procedure (Sec. 3.4).

Cost. We define *cost* as the computation required for one forward pass, measured in FLOPs and normalized so that activating all units costs 1. We use *width* to describe the proportion of active units in a bin: *full width* activates all units at a cost of 1, while *half width* activates half of the attention heads and MLP groups at a cost of 0.5. Let $|\mathcal{H}^b|$ and $|\mathcal{G}^b|$ denote the total numbers of active heads and MLP groups across all blocks in bin b. The cost of bin b and the model's average cost are given by

$$c(b) = \tfrac{1}{3}\tfrac{|\mathcal{H}^b|}{LH} + \tfrac{2}{3}\tfrac{|\mathcal{G}^b|}{LG}, \qquad \bar{c} = \tfrac{1}{B}\textstyle\sum_b c(b), \tag{2}$$

where $1/3$ and $2/3$ approximate the relative costs of attention and MLP computation based on their linear layers. Per token, the four attention projections (W_Q, W_K, W_V, W_O, each $d\times d$) cost $\approx 4d^2$, while the two MLP layers ($d \to 4d \to d$, i.e. $d\times 4d$ followed by $4d\times d$) cost $\approx 8d^2$. The token-interaction term ($QK^\top$ and $\mathrm{softmax}(QK^\top)V$) adds approximately $2Nd$ per token for N tokens. We omit it from the normalization because its effect on our results is small, as quantified in Sec. 6.

3.3 Training on one bin at a time

For each training batch, we sample a single bin b and draw all timesteps from that bin, so every sample uses the same active units. Inactive units are excluded from computation: the corresponding Q/K/V rows, output-projection columns, and MLP channels are omitted from the matrix multiplications

rather than merely masked. A step in bin b therefore has cost $c(b)$. We express the training budget $\tau_{\max}$ in full-width steps, with each full-width step consuming one budget unit and each narrower step consuming $c(b)$.

3.4 THE NEED SCORE

Every K units of training budget, we estimate the benefit of activating each inactive unit; K is the *growth interval*. For each bin, we run a batch with all units computed but keep the inactive units' gates at α=0. Their outputs therefore leave the predictions and loss unchanged, while backpropagation gives the gradient with respect to each gate. Let $\mathcal{L}$ be the loss on one noised sample of bin b. We define the *need score* of unit u in block l and bin b as:

$$s(l,b,u) \;=\; \left|\,\mathbb{E}\left[\,\partial\mathcal{L}/\partial\alpha_{l,b,u}\,\right]\right|_{\alpha_{l,b,u}=0}. \tag{3}$$

where $\mathbb{E}$ denotes expectation over noised samples with timesteps drawn from bin b, estimated using M samples. Because gates can take either sign, we use the gradient magnitude as the score. The following proposition shows how it relates to the first-order loss reduction from activating a unit.

Proposition 1 (First-order loss reduction)**.** *For an inactive unit u in block l and bin b, moving its gate a small distance δ from zero in the loss-reducing direction decreases the loss by $\delta s(l,b,u) + O(\delta^2)$. When several gates change by small amounts, their first-order loss reductions add.*

Appendix E provides the proof of the proposition. We train the gates with AdamW (Loshchilov & Hutter, 2019), which moves a gate by about the same amount whatever its gradient, so the loss decrease is proportional to s, as the proposition assumes. This estimate is local, so we recompute the scores at every growth event.

3.5 GROWING WITHIN THE BUDGET

The allowed average cost grows linearly from c_0 to the target $\bar{c}^*$:

$$c_{\text{allow}}(\tau) = c_0 + (\bar{c}^* - c_0)\,\min\!\big(1,\ \tau/(\rho\,\tau_{\max})\big), \tag{4}$$

where τ is the training budget spent so far and the allowed cost reaches $\bar{c}^*$ at $\tau = \rho\tau_{\max}$. Let $\Delta c(u)$ denote the increase in average cost from activating unit u in one bin. From Eq. (2), the increments for an attention head and an MLP group are, respectively,

$$\Delta c_{\text{attn}} = \frac{1}{3}\cdot\frac{1}{LHB}, \qquad \Delta c_{\text{mlp}} = \frac{2}{3}\cdot\frac{1}{LGB}. \tag{5}$$

Proposition 1 shows that the predicted loss reductions from small gate changes add to first order. At each growth event, we sort the inactive units in descending order of $s(l,b,u)/\Delta c(u)$ which measures the need score per unit of added cost, and activate them from highest to lowest until the next activation would exceed $c_{\text{allow}}(\tau)$. Each newly activated unit starts with $\alpha_{l,b,u} = 0$, leaving the model's output unchanged. Its gate then becomes trainable, allowing the unit to contribute gradually.

One scoring event runs a full-width forward and backward pass for each bin. Let c_{score} denote the cost of one scoring event. We add this cost to the budget spent whenever scores are computed. Algorithm 1 summarizes the procedure, and Table 1 lists the notation.

4 EXPERIMENTAL SETUP

Task and data. We evaluate class-conditional generation on ImageNet (Deng et al., 2009) and unconditional generation on FFHQ (Karras et al., 2019), both at 256×256 resolution. Following latent diffusion (Rombach et al., 2022), we encode each training image into a $32 \times 32 \times 4$ latent using the frozen SD-VAE.

Training and sampling. We train the models with the standard DDPM objective (Ho et al., 2020), predicting the added noise ϵ at uniformly sampled timesteps. We generate samples using 50-step DDIM sampling (Song et al., 2021a), with classifier-free guidance (Ho & Salimans, 2021) at a scale of 1.5 on ImageNet and no guidance on FFHQ. We report FID (Heusel et al., 2017) using 5k samples at intermediate checkpoints and 50k samples for the final results.

Backbones and budget.

Algorithm 1 Dynamic growth for DiT.

Initialize A so that each bin has cost c_0; set $\alpha \leftarrow A$ and $\bar{c} \leftarrow c_0$
$\tau \leftarrow 0$; $\tau_{\text{next}} \leftarrow K$
while $\tau < \tau_{\max}$ **do**
  Sample bin b
  if $\tau + c(b) > \tau_{\max}$ **then**
    break
  end if
  Train for one step using only the active units in bin b
  $\tau \leftarrow \tau + c(b)$
  if $\tau \geq \tau_{\text{next}}$ **then**
    $\tau_{\text{next}} \leftarrow \tau_{\text{next}} + K$
    if $\bar{c} < c_{\text{allow}}(\tau)$ **and** $\tau + c_{\text{score}} \leq \tau_{\max}$ **then**
      Compute $s(l, b, u)$ using Eq. (3)
      $\tau \leftarrow \tau + c_{\text{score}}$
      Sort inactive entries by $s(l, b, u)/\Delta c(u)$, descending
      for each (l, b, u) in this order **do**
        if $\bar{c} + \Delta c(u) > c_{\text{allow}}(\tau)$ **then**
          break
        end if
        $A_{l,b,u} \leftarrow 1$; $\alpha_{l,b,u} \leftarrow 0$ and becomes trainable
        $\bar{c} \leftarrow \bar{c} + \Delta c(u)$
      end for
    end if
  end if
end while

Symbol	Meaning
L	Number of transformer blocks
H, G	Attention heads, MLP groups per block
B	Number of timestep bins
A, α	Activation mask; output gates
$s(l, b, u)$	Need score of unit u in block l, bin b
$c(b)$, $\bar{c}$	Cost of bin b; average across bins
$\Delta c(u)$	Rise in $\bar{c}$ from activating u in one bin
τ, $\tau_{\max}$	Budget spent so far; total budget
τ_{next}	Next growth threshold
c_0, $\bar{c}^*$	Initial, target average cost
$c_{\text{allow}}(\tau)$	Allowed average cost at budget τ
ρ	Budget fraction where the limit is reached
K	Growth interval in budget units
c_{score}	Cost of one scoring event

Table 1: Notation.

- *Backbones.* We use DiT-S/2 (d=384, 12 layers, H=6, 33M parameters), DiT-B/2 (d=768, 12 layers, H=12, 130M parameters), and DiT-L/2 (d=1024, 24 layers, H=16, 458M parameters). Each model has G=8 MLP groups and uses B=8 timestep bins.
- *Growth.* The allowed cost increases linearly from c_0=0.5 and reaches its target after ρ=0.6 of the training budget. Growth events occur every K=5 000 budget units. Each need score is averaged over M=128 training samples per bin.
- *Inference cost.* We target approximately 75% of full-width compute. With 5 of 6 attention heads and 6 of 8 MLP groups active, DiT-S/2 has cost $\bar{c}^* = \frac{1}{3} \cdot \frac{5}{6} + \frac{2}{3} \cdot \frac{6}{8} = 0.778$ by Eq. (2). We scale the head count proportionally for DiT-B/2, which reaches the same 0.778 with 10 of 12 heads. DiT-L/2 has 16 heads, so $5/6$ is not an integer: it uses 13 of 16 and runs at 0.771. Unless otherwise stated, all methods are evaluated at the average cost of their own scale.
- *Seeds.* Every number we report is the mean over two training seeds.
- *Training budget.* The 1× budget is 150k full-width-step units × 0.778 ≈ 117k units, i.e. 117k full-width steps at batch size 256 (about 30 million samples for the dense model, more for narrower ones). Larger budgets (2×, 5×) continue the same runs.

Compared methods. We compare six ways of spending the same training budget, except *train-then-prune*.

GrowDiT (ours), the DiT trained with dynamic growth, starts at half width and grows to an average cost of 0.778, as described in Sec. 3.

Depth growth increases the number L' of blocks rather than model width Xie et al. (2025). It runs the first L' of L full-width blocks, at a cost of L'/L, and increases L' on the same growth schedule. Each

newly activated block starts as an identity mapping with its residual gates set to zero. For $L{=}12$, the costs bracketing our target of 0.778 are 0.750 ($L'{=}9$) and 0.833 ($L'{=}10$).

Random growth uses the same initial width, cost schedule, and number of activations per event as dynamic growth, but selects units at random rather than by need score. This comparison isolates the effect of the selection criterion.

Uniform keeps a fixed cost of 0.778 in every bin throughout training. It uses the same DiT backbone and matches our total training budget and inference cost. It therefore isolates the combined effect of progressive growth and noise-level-dependent capacity allocation.

Dense uses the full DiT at cost 1.0. Under the same training budget, it completes fewer steps and costs 29% more at inference than the methods running at 0.778.

Train-then-prune first trains a dense DiT for the full budget. It then adds DyDiT's timestep-dependent routers (Zhao et al., 2025), which select attention heads and MLP groups in each layer, and fine-tunes the model and routers for an additional 10% of the budget, so the total training cost is $1.1\times$. Each router maps the timestep embedding to one score per unit using a single linear layer. Together, the routers increase the parameter count by 0.13–0.20%. Units whose scores exceed a threshold are activated. We set it so the average inference cost is 0.778 at every scale.

5 Results

5.1 Where capacity is needed, and whether growth finds it

To examine how capacity should be allocated across noise levels, we use two toy datasets, each a two-dimensional Gaussian mixture: (1) eight small clusters arranged around a circle and (2) four clusters, each divided into four subclusters (insets of Fig. 2). The backbone is a small residual MLP with 16 groups of hidden units per layer. We mask, gate, and grow these groups as described in Sec. 3, using the same noise schedule, $B{=}8$ timestep bins, and ϵ-prediction loss as in our DiT experiments (Sec. 4).

To estimate where capacity is needed, we compare two reference denoisers that can be computed in closed form for each mixture. The Bayes-optimal denoiser gives the lowest possible prediction error, while the best affine denoiser gives the lowest error achievable by an affine map. The gap between their errors captures the nonlinear part of the task: a larger gap suggests a greater need for capacity. Computing this gap for every timestep bin tells us where capacity is needed across the diffusion process.

Fig. 2 shows that the need for capacity is greatest at low noise. At high noise, the clusters overlap and the noisy distribution approaches a single Gaussian, making affine denoising nearly optimal. Dynamic growth succeeded to follow the measured pattern: its final capacity per bin correlates with the error gap at 0.94 and 0.98 on the two datasets. The lowest-noise bin reaches 0.97–0.99 of full width, while the high-noise bins remain near their starting width. *Random growth* follows the same schedule without the need score but ends with a nearly flat allocation. Appendix D reports generation quality on these toy datasets.

5.2 Main results

Results across datasets and scales. At the reported training budgets and inference costs, *GrowDiT* achieves the lowest FID at all three ImageNet model scales (Table 2) and both FFHQ scales (Table 3). It also leads at every budget shown in Fig. 3. Its FID advantage over *uniform* increases with model scale, from 3% on DiT-S to 7% on DiT-B and 10% on DiT-L. At every scale, *GrowDiT* also has the highest Inception Score and Recall, with comparable Precision (Table 2). Qualitative samples appear in Appendix F (Fig. 8, Fig. 10).

Longer training does not close the gap. Extending the runs to larger budgets widens the FID advantage over *uniform* (Fig. 3). On DiT-B, it grows from 7% at $1\times$ to 9% at $2\times$ (50k-sample FID 27.4 vs 30.0). On DiT-S, it grows from 3% at $1\times$ to 5% at $2\times$ (47.7 vs 50.3; Table 6).

Less training compute for the same quality. The curves in Fig. 3 also show how much compute each method needs to reach a given FID. *GrowDiT* reaches the FID of *uniform* at $1\times$ using $0.94\times$

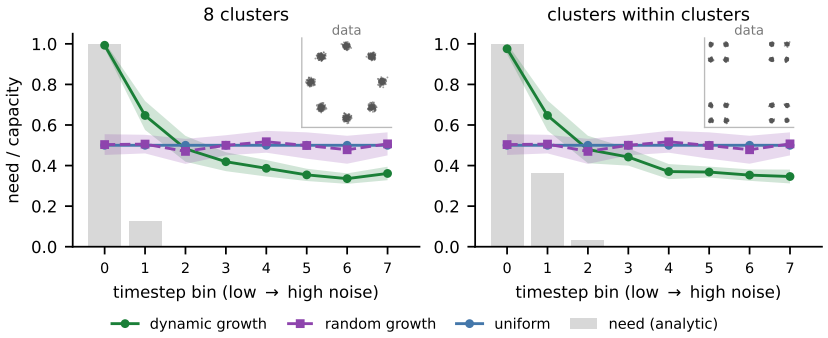


Figure 2: Toy experiment: capacity allocation across noise levels. Insets show 1,500 samples from each mixture: eight clusters on a circle (left) and four clusters containing four subclusters each (right). Gray bars show the normalized theoretical capacity requirement for each timestep bin. Lines show the final active capacity per bin, averaged over 18 seeds; shaded regions indicate one standard deviation.

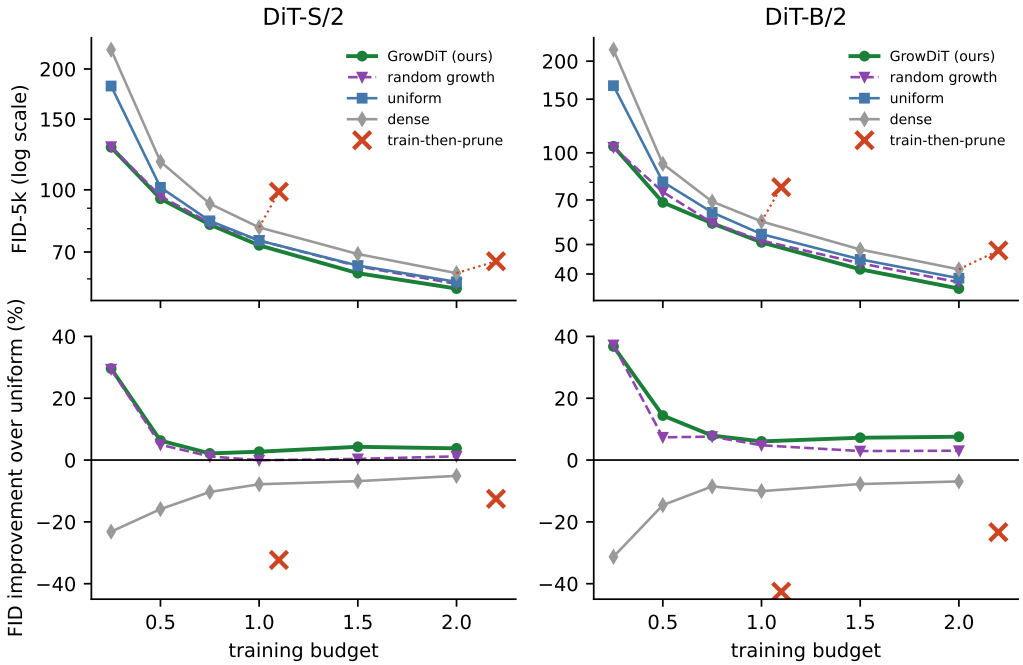


Figure 3: Training curves on ImageNet-256 at matched inference cost. Top: FID on a logarithmic scale. Bottom: percentage improvement in FID over uniform width at the same training budget. The $1\times$ budget is defined in Sec. 4; For *train-then-prune*, dense models trained to $1\times$ or $2\times$ are then pruned and fine-tuned. Dotted lines connect them to the dense checkpoints from which fine-tuning begins.

of the budget on DiT-S, $0.89\times$ on DiT-B, and $0.90\times$ on DiT-L, a saving of 6 to 11%. Conversely, *uniform* needs $1.10\times$ the budget on DiT-S and $1.16\times$ on DiT-B to match *GrowDiT* at $1\times$. These estimates use log-linear interpolation between the 5k-sample FID milestones.

Comparison with train-then-prune. *Train-then-prune* spends part of its budget on capacity that is later removed and performs worse than every other method, including the *dense* model from which it is initialized. *GrowDiT* performs 43% better on DiT-B (FID 42.2 vs. 74.4) and 31% better on DiT-S (63.8 vs. 92.4). Pruning an undertrained model removes capacity that has not yet become redundant, and a recovery stage of 11.7k steps does not restore the lost performance. Growth avoids this: a unit is activated with its gate at zero, so nothing is destroyed and nothing has to be repaired. Longer dense pretraining narrows the gap: at $1\times$, *train-then-prune* is 30% worse than *dense* on DiT-S and 48%

ImageNet, 1× budget	FID↓	IS↑	Prec↑	Rec↑
DiT-S/2 (33M)				
GrowDiT (ours)	**63.8**	**21.8**	**.637**	**.092**
random growth	65.2	21.5	.630	.087
uniform	65.7	21.0	.636	.083
dense	71.2	18.7	.630	.067
train-then-prune	92.4	12.3	.532	.024
DiT-B/2 (130M)				
GrowDiT (ours)	**42.2**	**33.4**	.673	**.180**
random growth	43.2	32.8	.677	.168
uniform	45.4	30.3	.679	.155
dense	50.2	25.5	**.689**	.124
train-then-prune	74.4	16.1	.588	.040
DiT-L/2 (458M)				
GrowDiT (ours)	**22.9**	**57.4**	.711	**.322**
random growth	23.6	54.9	**.720**	.306
uniform	25.4	53.2	**.712**	.302
dense	32.4	40.8	.706	.239
train-then-prune	36.2	36.7	.644	.200

Table 2: ImageNet-256 results across three model scales: 50k-sample FID, Inception Score, Precision, and Recall. Inference cost is 0.778 on DiT-S and DiT-B and 0.771 on DiT-L, except for dense (1.0). Training cost is 1.1× for train-then-prune and 1.0× for the others.

FFHQ-256	DiT-S	DiT-B
GrowDiT (ours)	**18.2**	**10.1**
random growth	19.5	10.6
uniform	19.2	11.3
dense	20.0	11.2
train-then-prune	50.7	58.5

Table 3: FFHQ-256 (unconditional) 50k-sample FID at the 1× budget.

DiT-S	**GrowDiT**	depth, L'=9	L'=10
cost	0.778	0.750	0.833
FID↓	**63.8**	69.9	71.6
IS↑	**21.8**	20.1	19.5
Prec↑	**.637**	.614	.607
Rec↑	**.092**	.075	.071

Table 4: Depth growth versus width growth (ours) on ImageNet-256 at the 1× training budget. Depth growth runs the first L' of 12 blocks at full width, giving an inference cost of $L'/12$. Because no integer value of L' matches our target cost of 0.778, we report the two closest settings.

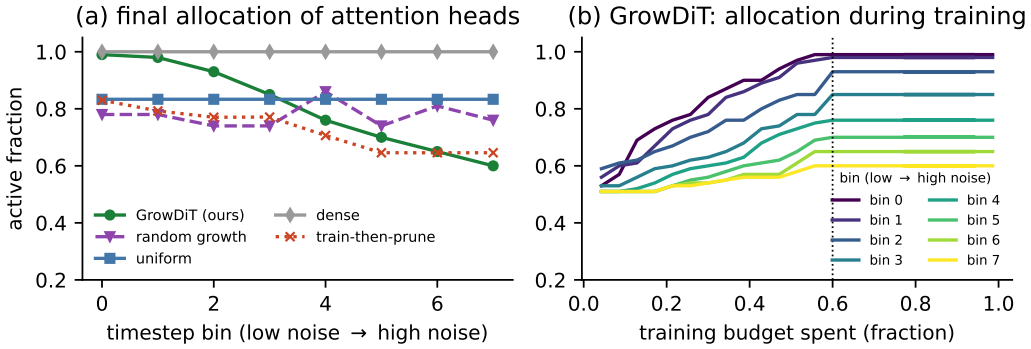


Figure 4: Capacity allocation for DiT-B on ImageNet. (a) Final fraction of active attention heads in each timestep bin for each method; MLP groups show a similar pattern. (b) Fraction of active heads in each bin during dynamic growth training. Each curve represents one bin, and the dotted line marks the end of growth at 60% of the training budget.

worse on DiT-B; at 2×, these gaps fall to 8% and 19%, respectively. Train-then-prune therefore suits a model that is already well pretrained, and growth suits training from scratch.

The learned allocation. Fig. 4(a) shows that *GrowDiT* activates more attention heads at low noise and fewer at high noise. The *train-then-prune* routers produce the same overall pattern. Dynamic growth reaches a similar allocation without first training capacity that is later removed, and it does so progressively, with low-noise bins growing earlier and reaching greater widths (Fig. 4(b)). Fig. 5 shows the detailed attention heads and MLP groups active in each bin and layer. High-noise bins use fewer units, and the selected units differ across layers.

Which axis to grow. Depth growth performs worse at both configurations that bracket our inference cost (Table 4). With L'=9 of 12 blocks, it uses less compute than *GrowDiT* but has an FID 6.1 higher; with L'=10, it uses more compute and has an FID 7.8 higher. Depth growth activates whole blocks at every timestep, each adding 1/12 to the cost of a step, whereas activating a head or MLP group in one bin adds less than 0.1% to the average cost.

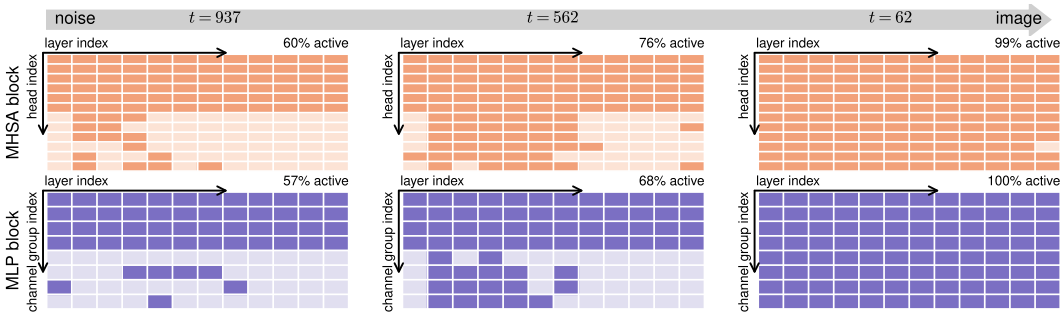


Figure 5: Learned architecture of *GrowDiT* on DiT-B on ImageNet. Each panel shows a layer-by-unit grid for one noise level, with attention heads on top and MLP channel groups below. Filled cells indicate active units.

6 Limitations

(1) Our FLOPs estimates omit attention's token-interaction terms (Sec. 3). Including them raises attention's share of block compute from 0.33 to 0.40 for DiT-S and 0.36 for DiT-L, but changes estimated training and inference costs by at most 0.25%. (2) FLOP savings translate only partly into wall-clock savings (Table 5), because normalization and modulation of the full residual stream take the same time whatever the active width. That width-independent part is 29% on DiT-S, 26% on DiT-B and 24% on DiT-L. The share of the width-independent part decreases as the model grows, so the gap between FLOPs and wall-clock narrows at larger scale. The numbers are hardware- and implementation-dependent.

	DiT-S		DiT-B		DiT-L	
Budgets matched by	FLOPs	clock	FLOPs	clock	FLOPs	clock
GrowDiT	2.9	1.1	7.0	5.3	9.8	7.6
random growth	0.8	−1.0	4.8	2.6	7.1	4.8
dense	−8.4	−3.7	−10.6	−5.6	−27.6	−19.8

Table 5: FID improvement over *uniform* (%) at the $1\times$ budget, with budgets matched by FLOPs and by wall-clock on one A100 with the model compiled.

7 Conclusion

DyDiT shows that a trained diffusion transformer contains substantial redundant capacity, raising the question of whether that capacity needs to be trained at all. The proposed GrowDiT begins with fewer active attention heads and MLP groups and adds units where they are most useful across noise levels. Under a fixed training budget, cheaper early updates leave more compute for additional optimization steps.

At the same training budget apart from *train-then-prune*, and at approximately the same inference cost apart from *dense*, GrowDiT achieves lower FID than fixed-width training and *train-then-prune* on ImageNet and FFHQ, and its advantage grows with the training budget. Under the same growth schedule, selecting units by need score outperforms random selection at every scale, while width growth outperforms depth growth by 6–8 FID points. The advantage is smaller when budgets are matched by wall-clock time rather than FLOPs, and that gap narrows as the model grows. These results suggest that gradients can guide capacity allocation across noise levels during training, without first pruning a fully trained model. We grow width only, but the need score is defined for any gate, so depth and tokens could be allocated the same way.

AI use statement

In this work, we used generative AI tools to improve the readability of the manuscript and to assist with literature retrieval and discovery. We did not use these tools to develop the methodology,

formulate mathematical claims, design or implement experiments, analyze data, or interpret the results. All AI-assisted text was reviewed and revised by the authors, and every reference identified with AI assistance was independently read and verified against the original publication. We take full responsibility for the final content of this work, including text produced with the aid of generative AI.

Reproducibility statement

Section 4 gives the model sizes, training budgets and inference costs of every run, and Algorithm 1 states the full training procedure. Appendix A lists the optimizer, numerical precision, noise schedule, mask initialisation and evaluation protocol.

References

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22669–22679. IEEE, 2023.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2bert: Towards reusable pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2134–2148, 2022.

Junsong Chen, Jincheng YU, Chongjian GE, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=eAKmQPe3m1.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Yao Chen, Yilong Chen, Yinqi Yang, Junyuan Shang, Zhenyu Zhang, Zefeng Zhang, Shuaiyi Nie, Shuohuan Wang, Yu Sun, Hua Wu, et al. Sparse growing transformer: Training-time sparse depth allocation via progressive attention looping. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6168–6193, 2026.

Kun Cheng, Xiao He, Lei Yu, Zhijun Tu, Mingrui Zhu, Nannan Wang, Xinbo Gao, and Jie Hu. Diff-moe: Diffusion transformer with time-aware and space-adaptive experts. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=JCUsWrwkKw.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 9275–9285. IEEE, 2022.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. *Advances in Neural Information Processing Systems*, 37:10491–10540, 2024.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Fabian Pedregosa, and Max Vladymyrov. Gradmax: Growing neural networks using gradient information. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qjN4h_wwUO.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101. IEEE, 2023.

Haipeng Fang, Sheng Tang, Juan Cao, Enshuo Zhang, Fan Tang, and Tong-Yee Lee. Attend to not attended: Structure-then-detail token merging for post-training dit acceleration. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18083–18092. IEEE, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Alireza Ganjdanesh, Yan Kang, Yuchen Liu, Richard Zhang, Zhe Lin, and Heng Huang. Mixture of efficient diffusion experts through automatic interval and sub-network selection. In *European Conference on Computer Vision*, pp. 54–71. Springer, 2024.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pp. 2337–2346. PMLR, 2019.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595. IEEE, 2018.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(11): 7436–7456, 2021.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7407–7417. IEEE, 2023.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Wenxuan Huang, Yunhang Shen, Jiao Xie, Baochang Zhang, Gaoqi He, Ke Li, Xing Sun, and Shaohui Lin. A general and efficient training for transformer via token expansion. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15783–15792. IEEE, 2024.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 4396–4405. IEEE, 2019.

Felix Krause, Timy Phan, Ming Gui, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. Tread: Token routing for efficient architecture-agnostic diffusion training. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15703–15713. IEEE, 2025.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Zhengyang Liang, Hao He, Ceyuan Yang, and Bo Dai. Scaling laws for diffusion transformers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T985gm4sDA.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=2uAaGwlP_V.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024a.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=ZupoMzMNrO.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11256–11264. IEEE, 2019.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://openreview.net/forum?id=-NEXDKk8gZ.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Yifan Pu, Zhuofan Xia, Jiayi Guo, Dongchen Han, Qixiu Li, Duo Li, Yuhui Yuan, Ji Li, Yizeng Han, Shiji Song, et al. Efficient diffusion transformer with step-wise dynamic attention mediators. In *European Conference on Computer Vision*, pp. 424–441. Springer, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Carlos Riquelme Ruiz, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=NGPmH3vbAA_.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *International Conference on Machine Learning*, pp. 19893–19908. PMLR, 2022.

Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, et al. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pp. 2256–2265, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WNzy9bRDvG.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Jiangshan Wang, Zeqiang Lai, Jiarui Chen, Jiayi Guo, Hang Guo, Xiu Li, Xiangyu Yue, and Chunchao Guo. Elastic diffusion transformer. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=493YXJPdeq.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cDYRS5iZ16f.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: a general approach for growing neural networks. *Advances in neural information processing systems*, 33: 22373–22383, 2020.

Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Chengyue Wu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk7KsfW0-.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DJSZGGZYVi.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=taHwqSrbrb.

Chenyu Zheng, Xinyu Zhang, Rongzhen Wang, Wei Huang, Zhi Tian, Weilin Huang, Jun Zhu, and Chongxuan Li. Scaling diffusion transformers efficiently via μp. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VfIOdGiBAv.

Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=vTBjBtGioE.

Shaobin Zhuang, Yiwei Guo, Yanbo Ding, Kunchang Li, Xinyuan Chen, Yaohui Wang, Fangyikang Wang, Ying Zhang, Chen Li, and Yali Wang. Timestep master: Asymmetrical mixture of timestep loRA experts for versatile and efficient diffusion models in vision. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9mPhtzB9XP.

zhuofan xia, Xuran Pan, Xuan Jin, Yuan He, Hui Xue', Shiji Song, and Gao Huang. Budgeted training for vision transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sVzBN-DlJRi.

A Implementation details

Data and diffusion. Images are encoded by the frozen SD-VAE into $32 \times 32 \times 4$ latents scaled by 0.18215. ImageNet-256 is class-conditional over 1000 classes plus a null class used for guidance, and FFHQ-256 is unconditional. We use the linear beta schedule of ADM, with β from 10^{-4} to 0.02 over 1000 steps, an ϵ-prediction target and a squared error loss.

Optimization. Every method uses AdamW with learning rate 10^{-4}, weight decay 0, moments $(0.9, 0.999)$, and no warmup or decay. The batch size is 256 and gradient norms are clipped to 1.0. Forward and backward passes run in bfloat16 while parameters and optimizer states stay in float32, with TF32 matrix multiplication enabled. We keep an exponential moving average of every floating-point tensor with decay 0.9999 and report all numbers from it. *Train-then-prune* additionally uses learning rate 10^{-2} for its routers and weight 2.0 on the FLOPs penalty.

Masks and gates. At initialization each bin activates a fraction c_0 of the heads and of the MLP groups in every block, the same units in each block, with their gates at 1. Growth activates a unit with its gate at 0, which leaves the function unchanged, and the gate is trained from then on. *Uniform* instead keeps five sixths of the attention heads and three quarters of the MLP groups active for the whole run, and *dense* keeps every unit active.

Evaluation and hardware. We report FID against 50k reference images from the training set, using 5k samples at intermediate checkpoints and 50k for every number in a table, which also give Inception Score, Precision and Recall. Each run uses one NVIDIA A100 with 64 GB of memory.

B Doubling the training budget

Table 6 repeats the main comparison at the $2\times$ budget on ImageNet. *GrowDiT* still has the lowest FID at both scales, and its lead over *uniform* is wider than at $1\times$ (Table 2); *random growth* again sits between *GrowDiT* and *uniform*. The lead does not close with more training.

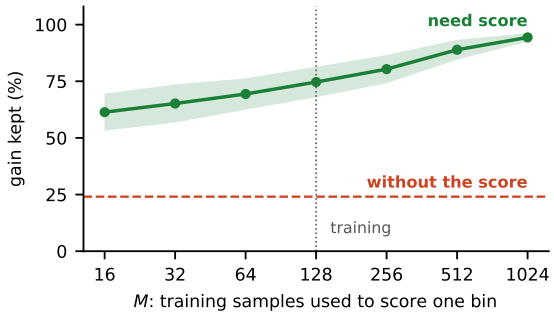


Figure 6: Sensitivity of *gain kept* to M, the number of training samples used to estimate each need score. The dotted line marks $M = 128$, the setting used in training; the dashed line shows random selection of 28 units.

starting width c_0	0.25	**0.5**	0.75
FID-50k	64.3	**64.6**	64.7
growth end ρ	0.4	**0.6**	0.8
FID-50k	64.7	**64.6**	62.9
growth interval K	2 500	**5 000**	10 000
FID-50k	64.6	**64.6**	64.4
timestep bins B	4	**8**	16
FID-50k	63.6	**64.6**	66.4

Table 7: Sensitivity of *GrowDiT* to the growth hyperparameters. For each hyperparameter, the first row lists the tested values and the second reports the corresponding FID scores (lower is better). Default values are in bold.

FID-50k, 2× budget	DiT-S	DiT-B
GrowDiT (ours)	**47.7**	**27.4**
random growth	49.5	29.3
uniform	50.3	30.0
dense	52.9	32.9
train-then-prune	56.9	39.1

Table 6: Same methods and protocol as Table 2, but each run trained for the 2× budget (ImageNet-256). Train-then-prune is at 2.2× (dense training to 2× plus fine-tuning).

C THE NEED SCORE AND GROWTH HYPERPARAMETERS

We examine the sensitivity of dynamic growth to four growth hyperparameters and to M, the number of training samples used to estimate each need score. Both analyses use DiT-S on ImageNet-256 at the 1× budget, with results averaged over two random seeds. Table 7 varies one growth hyperparameter at a time and reports 50k-sample FID. Fig. 6 examines how M affects unit selection. For each M, we select the 28 highest-scoring units, as in one growth event. We define *gain kept* as their summed need score, evaluated using 4096 samples, divided by the sum for the 28 units selected using those same scores.

Growth hyperparameters. Every tested setting is within two FID points of the default. The default is not the best in every case: ρ=0.8 lowers FID by 1.7 points, and B=4 lowers it by 1.0.

Score exponent. Ranking inactive units by $s^2/\Delta c$ instead of $s/\Delta c$ gives FID-50k 64.1 on DiT-S, against 63.8 for the default one.

Samples for the need score. At the training setting M=128, *gain kept* is 75%, compared with 25% for random selection. Increasing M to 1024 raises it to 94% but requires eight times as many samples. Thus, more samples improve the need-score estimates and unit selection, while requiring more computation.

D TOY STUDY DETAILS

Setup. The two mixtures live in two dimensions: eight clusters of standard deviation 0.05 on the unit circle, and four clusters of four subclusters of standard deviation 0.03. The noise schedule, the $B{=}8$ bins and the ϵ-prediction loss match the DiT experiments (Sec. 4). The denoiser is a linear input map to 32 channels, three residual blocks of 64 hidden units in $G{=}16$ groups, and a linear output map, with a mask and a gate per group and bin. Each step draws one bin and a batch of 512, using AdamW at learning rate 10^{-3}, $\tau_{\max}{=}20\,000$ full-width steps, $c_0{=}0.25$, $\bar{c}^*{=}0.5$, $\rho{=}0.6$, $K{=}1000$ and $M{=}512$. *Train-then-prune* runs at full width for 80% of the budget, then in each bin switches off the groups whose removal changes the loss least to first order until the cost reaches $\bar{c}^*$, and fine-tunes for the rest. Quality is the excess loss over the Bayes-optimal denoiser, $\mathbb{E}\|\hat{\epsilon} - \epsilon^*\|^2$, which is zero when the two agree; we average it over bins on 20k fresh samples per bin.

Quality. Fig. 7 reports the mean and standard error over 18 seeds at the same budget. Running at full width leaves fewer steps here as it does on the DiTs (Sec. 5.2): *dynamic growth* has a lower mean excess loss than *uniform*, *dense* and *train-then-prune* on both mixtures, and beats *dense* in 14 of 18 seeds on each. Where the capacity goes matters less than on the DiTs, whose latents have 4096 dimensions rather than two: *dynamic growth* beats *random growth* in only 8 and 10 of 18 seeds. We therefore read the allocation from this study (Fig. 2) and leave the quality comparison to the DiT experiments.

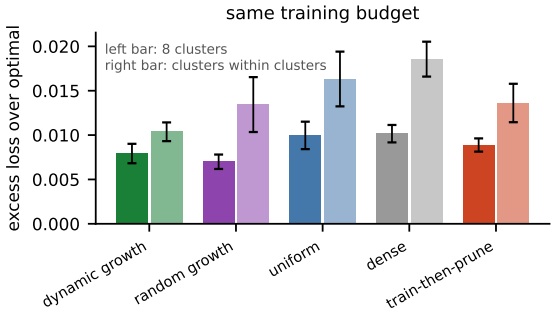


Figure 7: Excess loss over the Bayes-optimal denoiser at the same training budget (lower is better). Bars show the mean and standard error over 18 seeds. In each pair, the left bar represents eight clusters and the right bar represents clusters within clusters.

E PROOF OF PROPOSITION 1

The proof applies a first-order Taylor expansion to gates that scale unit outputs. This is related to sensitivity-based pruning (LeCun et al., 1989; Hassibi & Stork, 1992; Molchanov et al., 2019) and gradient-guided growth (Evci et al., 2022; Wu et al., 2020).

Proof. Let $\mathcal{L}$ be the loss on one noised sample and $\bar{\mathcal{L}}_b = \mathbb{E}[\mathcal{L}]$ the expected loss for bin b. We hold the model weights and existing active gates fixed. Assume that the expected loss is twice continuously differentiable in the gates and that differentiation can be interchanged with expectation.

Changing an inactive unit's gate from zero to α changes the block's residual stream from r to $r + \alpha z_{l,u}(x)$. Here $z_{l,u}(x)$ is the unit's contribution after its output projection: $\mathrm{head}_h(x)W_O^{(h)}$ for an attention head, or the group's hidden activations multiplied by the corresponding rows of the second MLP weight matrix. The gate affects the loss through this contribution. The chain rule then gives

$$\bar{\mathcal{L}}'_b(0) = \mathbb{E}\langle \nabla_r \mathcal{L}, z_{l,u}(x) \rangle, \qquad s(l,b,u) = |\bar{\mathcal{L}}'_b(0)|.$$

Taylor expansion around $\alpha = 0$ gives

$$\bar{\mathcal{L}}_b(\alpha) - \bar{\mathcal{L}}_b(0) = \alpha \bar{\mathcal{L}}'_b(0) + O(\alpha^2).$$

Choosing $\alpha = -\delta \,\mathrm{sign}(\bar{\mathcal{L}}'_b(0))$ therefore yields

$$\bar{\mathcal{L}}_b(\alpha) - \bar{\mathcal{L}}_b(0) = -\delta s(l,b,u) + O(\delta^2).$$

Now consider a set $\mathcal{U}_b$ of inactive units in bin b. Move each gate by the same small distance δ in its loss-reducing direction. The multivariate Taylor expansion gives

$$\bar{\mathcal{L}}_b(\boldsymbol{\alpha}_{\mathcal{U}_b}) - \bar{\mathcal{L}}_b(\mathbf{0}) = -\delta \sum_{(l,u)\in\mathcal{U}_b} s(l,b,u) + O(\delta^2).$$

Here, $\mathbf{0}$ denotes the selected gates before activation. The first-order terms add, while interactions between gates enter the second-order remainder. □

Selection under a cost budget. With uniformly sampled bins, the training objective is $\bar{\mathcal{L}} = B^{-1}\sum_b \bar{\mathcal{L}}_b$. For a set $\mathcal{U}$ of selected entries (l,b,u), the predicted first-order loss reduction is therefore

$$\frac{\delta}{B} \sum_{(l,b,u)\in\mathcal{U}} s(l,b,u).$$

Since δ/B is common to all candidates, maximizing this reduction under the remaining allowance $c_{\text{allow}}(\tau) - \bar{c}$ gives the knapsack problem

$$\max_{\mathcal{U}} \sum_{(l,b,u)\in\mathcal{U}} s(l,b,u) \quad \text{subject to} \quad \sum_{(l,b,u)\in\mathcal{U}} \Delta c(u) \le c_{\text{allow}}(\tau) - \bar{c}.$$

Sorting units in descending order of $s(l,b,u)/\Delta c(u)$ gives the optimum of the fractional relaxation, in which the final unit may be selected partially. Our discrete rule follows the same ordering but stops before the first unit that exceeds the budget. The fractional solution can include at most a fraction of this unit, so its total score exceeds ours by at most s_{next}. Because the fractional optimum upper-bounds the best selection of whole units, our total score is within s_{next} of that optimum. The corresponding gap in predicted first-order loss reduction is at most $\delta s_{\text{next}}/B$.

This bound applies to the local first-order objective and assumes exact scores. In practice, we estimate the scores from samples and recompute them at every growth event.

F Qualitative samples

Figures 8, 9 and 10 show samples from the $1\times$ budget comparison: DiT-B and DiT-L on ImageNet-256, and DiT-B on FFHQ-256. Each column fixes the initial noise and, on ImageNet, the class label, so the rows within a column differ only in how the training budget was spent. In all three figures the *train-then-prune* row loses fine texture; on FFHQ its faces are noticeably smoothed.

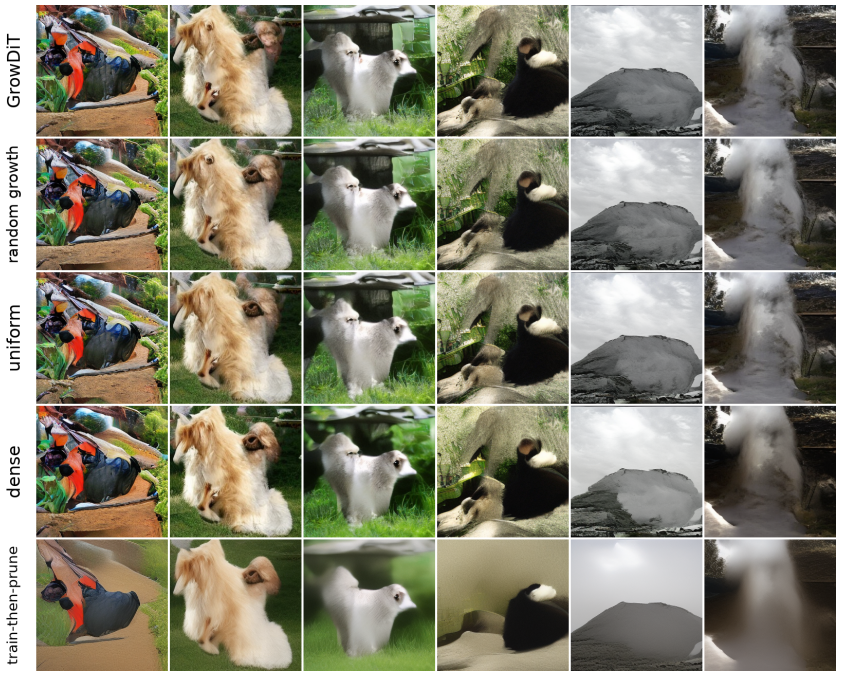


Figure 8: Samples from DiT-B models in the ImageNet-256 $1\times$ budget comparison. Within each column, all rows share the same initial noise and class label. Samples use 50 DDIM steps and a guidance scale of 1.5. Rows, from top to bottom: *GrowDiT*, *random growth*, *uniform*, *dense*, and *train-then-prune*. Inference cost is 0.78 for all methods except *dense* (1.0).

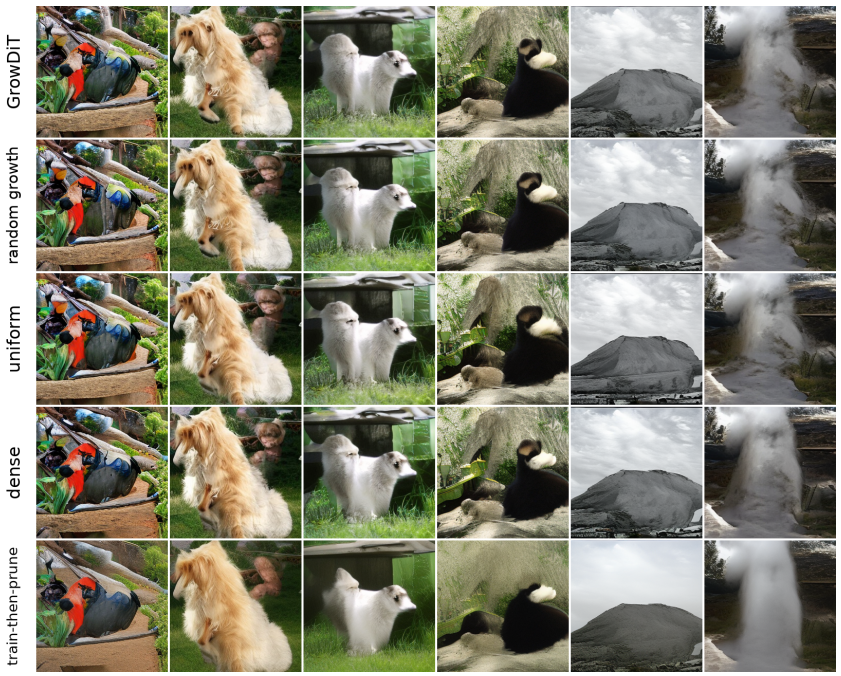


Figure 9: ImageNet-256 samples from DiT-L models in the $1\times$ budget comparison, using the protocol of Fig. 8. Inference cost is 0.771 for *GrowDiT*, *random growth*, and *uniform*; 0.778 for *train-then-prune*; and 1.0 for *dense*.



Figure 10: Samples from DiT-B models in the FFHQ-256 $1\times$ budget comparison, following the protocol of Fig. 8 except that sampling is unconditional and uses no guidance. Within each column, all rows share the same initial noise. Rows, from top to bottom: *GrowDiT*, *random growth*, *uniform*, *dense*, and *train-then-prune*.