Token-Sparse Diffusion Transformers

Matthew T. Jackson **Benjamin Ellis** **Shimon Whiteson** **Jakob N. Foerster**
University of Oxford
jackson@robots.ox.ac.uk

Abstract

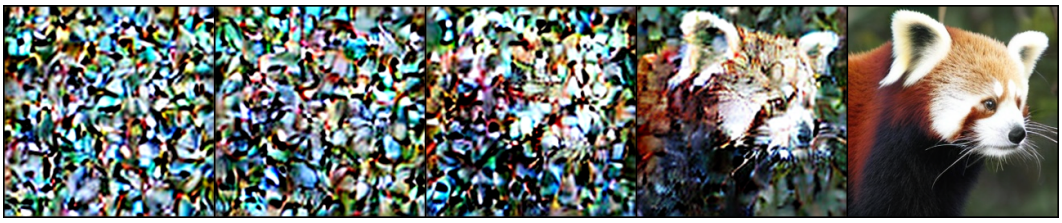
Real-time video generation, interactive world models, and on-device image synthesis are almost a reality. Diffusion models have achieved sufficient sample quality for these tasks, moving the bottleneck to the throughput of the models themselves. Typically, throughput constraints are addressed with model distillation or timestep subsampling, compromising the quality of the original model. Luckily, images are a sparse modality, where few patches contain a majority of global information. Despite this, predominant architectures such as diffusion transformers (DiTs) process patches uniformly, leading to suboptimal compute allocation and failing to exploit this sparsity. As a solution, we introduce SparseDiT, a simple DiT extension that dynamically subsamples image tokens within both self-attention and MLP operations. Specifically, SparseDiT employs a simple routing mechanism with constant per-sample token capacity for self-attention blocks and dynamic capacity for MLP blocks, enabling efficient parallel inference with dynamic compute allocation across tokens, images, and timesteps. Our experiments in image generation and video world modeling demonstrate that SparseDiT significantly reduces computational costs—processing as few as 12.5% of the tokens on video tasks—while simultaneously improving generation performance over unrouted baselines. Furthermore, we analyze learned compute allocation of SparseDiT models, demonstrating intuitive allocation to meaningful image patches and high-motion video patches.

1 Introduction

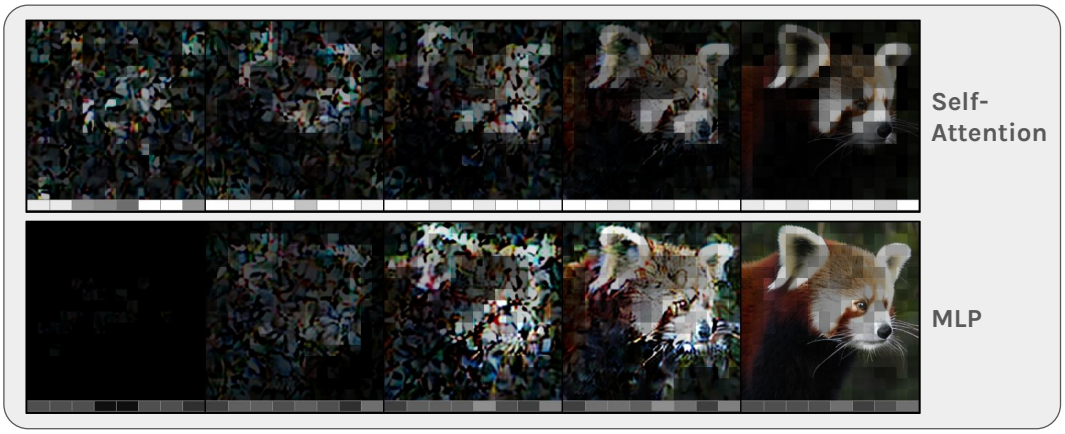
Recent advancements in real-time video generation, interactive world models, and on-device image synthesis promise to unlock an array of new applications for generative models and yield a new frontier of synthetic data. Diffusion models are at the forefront of this progress but their application has been limited by their iterative generation procedure, which requires multiple model evaluations to generate samples. The resulting latency is often reduced by subsampling diffusion timesteps to reduce model evaluations or distilling models into smaller variants to reduce the cost of each evaluation. However, both of these approaches sacrifice image quality in the name of speed. Improving the practical applicability of diffusion methods requires finding ways to reduce their latency while maintaining high image quality.

Fortunately, images and videos are a sparse modality, with relatively few patches containing a disproportionate amount of global information. For example, in many images the background is much simpler than the foreground and is therefore easier to denoise. This effect is magnified in videos, where frame patches containing static objects can be simply copied from the previous frame. Ideally, diffusion whould use less computation on simpler patches, reducing overall latency by only applying denoising when required. However, predominant diffusion architectures such as diffusion transformers (Peebles and Xie, 2023, DiT) fail to exploit this property, instead applying all operations to every token and thereby allocating compute uniformly over images and videos.

Preprint. Under review.



(a) Sample generation.



(b) Learned compute allocation.

Figure 1: In (a), a representative sample from SparseDiT-L/2 with 25% capacity, picked at random (seed 0). In (b), SparseDiT learns to allocate compute over samples, tokens (patches), and noise levels. Patch brightness corresponds to the proportion of blocks applied to the token at the given timestep. Compute allocation for register tokens is depicted below each sample, showing significant allocation by the self-attention operation at every timestep.

To solve this problem, we introduce **SparseDiT**, a simple extension to the DiT architecture that enables dynamic allocation of compute across images, patches, and noise levels. SparseDiT subsamples image tokens within the self-attention and MLP layers using a minimal routing mechanism (Section 3.1) and **requires no auxiliary loss or post-processing**. As shown in Figure 1, this enables SparseDiT to allocate more compute to the complex and meaningful elements of images, such as the red panda's ears and nose, with fewer FLOPs applied to the background. Furthermore, the model can allocate more computation at lower noise levels, when there is more signal with which to denoise the image. SparseDiT routes tokens with a dynamic per-sample token capacity for MLP blocks (Section 3.2), as well as a fixed per-sample token capacity for self-attention block (Section 3.3), thereby enabling efficient training and inference on parallel GPU hardware, without "ragged batches".

We evaluate SparseDiT on Imagenet (Deng et al., 2009) and the 1X World Modeling Challenge (1X, 2025), demonstrating its potential for image and video generation. On Imagenet (Section 4.1), SparseDiT improves performance across compute-normalized evaluation metrics against vanilla DiT models, considering a range of SparseDiT model capacities and diffusion sampling lengths. Using a SparseDiT model with only a 12.5% routing capacity, we similarly demonstrate improvements in video modeling PSNR on the 1X Challenge (Section 4.2), while requiring 33% fewer FLOPs than the DiT baseline. Furthermore, we analyze the routing behavior of SparseDiT on these tasks and demonstrate strong correlations between compute allocation and intuitive visual features. Namely, we observe increased compute allocation for image and patch classes with high token loss, as well as to spatio-temporal video patches with high optical flow.

In summary, our work presents and demonstrates an effective approach for efficient compute allocation in diffusion models, providing an intuitive analysis of the model's behavior. **SparseDiT improves generation quality and reduces FLOPs by dynamically allocating compute across samples, patches, and noise levels, requiring only a minimal routing operation.**

2 Background

2.1 Diffusion Models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Dhariwal and Nichol, 2021) generate images by iteratively removing noise from samples drawn from a known noisy distribution. Specifically, they consist of two complementary Markov processes: a forward noising process and a reverse denoising process. In the forward process, an initial clean image x_0 is progressively corrupted into a purely noisy image x_T across a discrete sequence of timesteps $t = 1, \ldots, T$, following the Gaussian conditional distribution $q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$, where $\beta_t \in (0, 1)$ defines the noise schedule at timestep t. Equivalently, this forward process can be expressed as $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$ and $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$. The reverse process, parameterized by a neural network, gradually reconstructs the original clean image from x_T to x_0 by approximating the conditional distributions $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. Model training involves minimizing the discrepancy between the predicted and actual noise at each timestep, enabling accurate modeling of the underlying data distribution.

2.2 Vision Transformers and Attention Sinks

Following their success in language modeling, transformers (Vaswani et al., 2017) have been successfully applied to vision tasks. Vision transformer (Dosovitskiy et al., 2020, ViT) achieves this by constructing a flattened token sequence from pixel patches. Similar to transformers, ViT models consist of alternating (multi-head) self-attention (Bahdanau et al., 2014) and multi-layer perceptron (MLP) blocks, with intermediate layer normalization (Ba et al., 2016). In these models, self-attention distributes information across the sample, in this case passing information between image patches, while the MLP processes tokens independently. Diffusion transformers (Peebles and Xie, 2023, DiT) adapt ViT models for image diffusion with adaptive layer normalization (adaLN), which conditions the normalization operation on the diffusion timestep and image class.

Outside of diffusion, various extensions to ViT have been proposed. Darcet et al. (2023) analyse the attention map of ViT models, finding the emergence of low-information image patches with anomalously high attention weights. These tokens, akin to *attention sinks* (Xiao et al., 2023), are used to pool global information from across the image, but require this information to be undone from the token. Therefore, the authors propose *register tokens* with learned embedding values that are appended to the image sequence and discarded at the end of the forward pass. Darcet et al. (2023) demonstrate that these tokens receive high attention weights and remove the attention sinks from the image, improving performance and the alignment of the saliency map with semantic features.

Finally, mixture-of-experts (Shazeer et al., 2017; Fedus et al., 2022, MoE) approaches improve transformer scaling by dividing transformer MLP blocks into multiple "expert" MLPs, with each token typically applied to only a single expert. Expert selection is determined by a routing module, in which a small MDP computes a scalar score for each token-expert pair before the highest expert scores for each token, or the highest token scores for each expert (Zhou et al., 2022), are selected.

3 Sparse Diffusion Transformers with Token Routing

Visual data is typically sparse so some image patches are significantly more challenging to denoise than others. Despite this, DiT models process all tokens uniformly, applying each self-attention and MLP operation to all tokens. To effectively process visual data with minimal operations, we therefore hypothesize that tokens corresponding to challenging denoising objectives should be applied to *more* blocks, thereby dynamically allocating compute over tokens.

To achieve this, we propose SparseDiT, a simple, routing-based extension to DiT, requiring no auxiliary losses. At the start of each self-attention and MLP block, SparseDiT uses a lightweight router (Section 3.1) to select a subset of tokens to be processed by that block. In self-attention blocks (Section 3.3), SparseDiT selects a fixed *capacity* of K tokens from each sample, and applies the self-attention block to those tokens, thereby enabling higher levels of compute to be allocated to important tokens. SparseDiT similarly selects the most important tokens to be routed to MLP blocks (Section 3.2); however, this is performed over the entire batch, allowing compute to be dynamically allocated across samples and noise levels, as well as tokens.

3.1 A Minimal Routing Mechanism for Diffusion Transformer Sparsity

At the core of our method is a simple routing mechanism, with similar design to an MoE routing module. Our router blocks consist of a tiny, two-layer MLP R_ϕ that is applied independently to each token t_i to compute a scalar score $R_\phi(t_i)$. During training, Gumbel noise $g_1, g_2 \sim \text{Gumbel}(0, 1)$ is applied to all router scores to encourage exploration in top-K selection (Jang et al., 2016), as well as inducing a bimodal score distribution with a threshold approaching 0.5 (Figure 3). For stability, we finally apply a sigmoid operation $\sigma(\cdot)$ with a fixed temperature (which we set to 5.0) to all scores, giving the token score

$$r_i = \begin{cases} \sigma(-\beta \cdot (R_\phi(t_i) + g_1 - g_2)), & \text{during training,} \\ \sigma(-\beta \cdot R_\phi(t_i)), & \text{during evaluation,} \end{cases} \tag{1}$$

where β is the inverse temperature. Following Raposo et al. (2024), we multiply the output of the block for each token by its routing score r_i, thereby ensuring that the router weights are updated by gradient-based optimizers. Intuitively, this also weights the magnitude of the update from the block by the "importance" of the token, ensuring that routed tokens with low scores have a reduced update from the block.

3.2 Dynamic-Capacity Routing for MLP Blocks

SparseDiT performs dynamic-capacity routing of tokens for MLP blocks, in which the number of routed tokens can vary across samples and noise levels. At training time, we compute routing scores for each token and perform a top-K operation over the entire training batch, with an overall capacity of $B \cdot K$ for a batch of size B. This ensures that, at training time, K tokens are routed *on average* per sample, but allows a variable number of tokens to be processed from each sample.

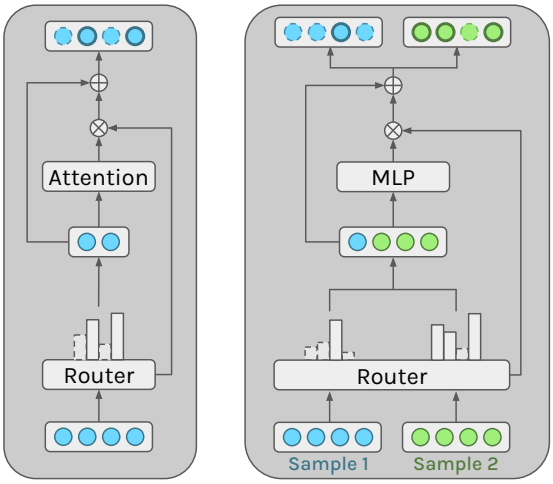


Figure 2: SparseDiT self-attention and MLP blocks—samples are processed independently in attention blocks to ensure fixed-size self-attention kernels, whilst the top-K tokens are computed across the entire batch (training) or with a static threshold (evaluation) for MLP blocks.

At test time, we aim to maintain dynamic allocation over samples and noise levels. However, naïvely applying the top-K approach can present issues at this stage. If samples are generated individually, top-K routing over tokens induces a constant capacity for each sample and diffusion timestep, losing the dynamic allocation. Similarly, if all samples in a batch are at the same noise level or come from the same class, a top-K operation induces a constant capacity at that level and class, sacrificing dynamic allocation. Instead, we require a proxy for the global top-K operation applied during training, when batches are uniformly sampled over samples and timesteps. This could be achieved by estimating the threshold for a token's router score to be in the top-K scores across all samples and timesteps, for instance by tracking this threshold over training. However, storing a separate threshold for each block introduces model complexity, as the thresholds most be estimated at the end of training and stored with the model.

As a simple and effective proxy, we propose generating samples using a constant MLP score threshold of 0.5. While this is an approximation of the true global threshold, we find that trained models naturally learn a threshold close to 0.5 when scores are computed across all samples and timesteps, regardless of their capacity (Figure 3, Appendix A). This is intuitive due to the Gumbel-sigmoid operation in our routing blocks, which introduces noise in the token scores during training. In order for a model to optimally select the most important tokens under this stochasticity, it is encouraged to produce a bimodal score distribution—with scores close to 1.0 for the top-K tokens and close to 0.0 for all other tokens—thereby minimizing the chance of suboptimal tokens being routed by the model due to Gumbel noise.

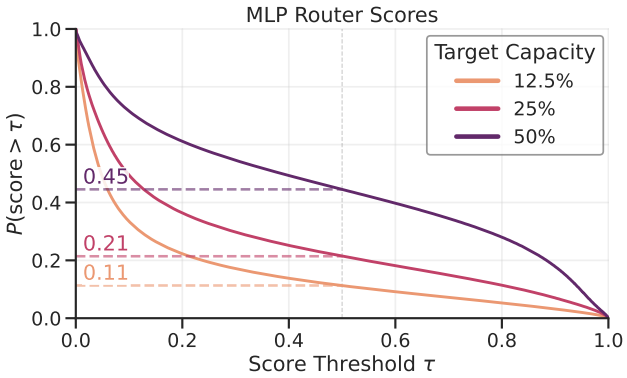


Figure 3: Router scores for various SparseDiT-L/2 router capacities—we observe that a static threshold of 0.5 closely approximates the target capacity, despite the model being trained only with top-K sampling and no auxiliary loss. Router scores for each block can be found in Appendix A.

3.3 Fixed-Capacity Routing for Self-Attention Blocks

Routing in self-attention blocks introduces additional computational constraints to MLP routing, requiring a distinct approach. To perform efficient batch generation on parallel GPU hardware, it is critical that the shapes of tensors remain constant for all operations in the batch. For dynamic application of MLP blocks this is not an issue, as tokens are processed independently by the MLP, allowing a variable number of tokens to be allocated from each sample. However, in the self-attention operation, all tokens from the *same* sample attend to each other, yielding an attention matrix with width equal to the number of per-sample tokens in the self-attention. In order for this attention matrix to have constant dimensionality across all samples, we therefore require attention blocks to have *constant capacity* for all samples. We implement this with a simple top-K operation over each sample at both training and test time, ensuring a fixed shape. While this prevents dynamic compute allocation across samples and timesteps, it enables allocation across tokens within a sample and avoids the issue of "ragged batches" with varying sequence lengths per sample.

4 Experiments

In this section, we evaluate the performance of SparseDiT on image generation (Section 4.1) and video world modeling tasks (Section 4.2). In each of these, we demonstrate improved sample quality against DiT models at the same number of model FLOPs, in addition to strong correlations between token routing proportion and semantic features, such as image and patch class, as well as motion in videos. These results demonstrate the potential of SparseDiT to achieve competitive generation performance with significantly reduced computation and provide insight into the behavior of learned compute allocation.

4.1 Image Generation

Experimental Setup We perform experiments on ImageNet (Deng et al., 2009), using the Face-blurred ILSVRC 2012–2017 variant (Yang et al., 2022) at 256×256 resolution. We train all models from scratch for 1M steps (requiring approximately 6 days on 4 L40S GPUs) using the DiT-L/2 configuration and all hyperparameters from Peebles and Xie (2023) without further tuning. This includes using the same pre-trained variational autoencoder (Kingma et al., 2013, VAE) from Stable Diffusion (Rombach et al., 2022), with an $8\times$ downsample factor, giving a latent embedding with dimensions $32 \times 32 \times 4$. To make SparseDiT comparable in FLOPs with DiT, we construct SparseDiT by replacing alternate DiT blocks with two SparseDiT blocks, approximately matching DiT FLOPs for a 50% capacity model with the same configuration. Finally, due to the effectiveness of register tokens in previous ViT models, we append every latent image with 8 register tokens, giving a total embedding length of 264.

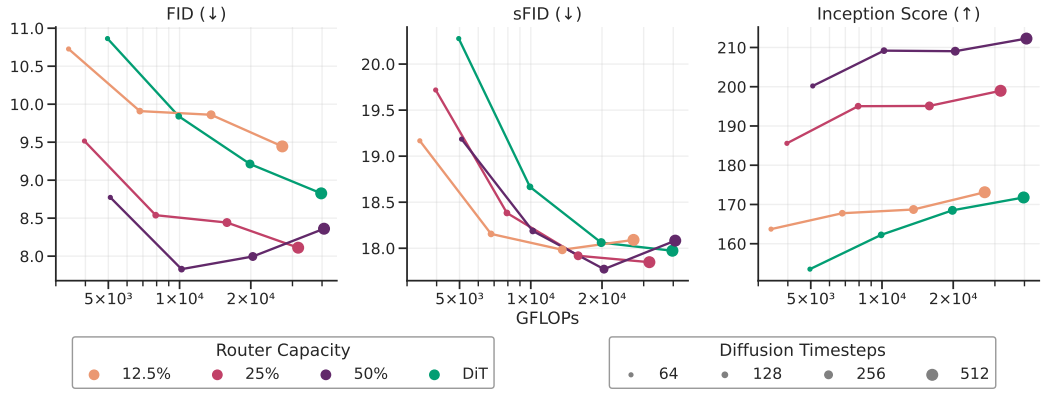


Figure 4: ImageNet performance—we report FID-10K, sFID-10K, and IS for samples generated with varying numbers of diffusion timesteps. Precision and recall evaluation can be found in Appendix B.

Evaluation Metrics We evaluate the performance of image generation using standard evaluation metrics of generative models: Fréchet Inception Distance (Heusel et al., 2017, FID), spatial FID (Nash et al., 2021, sFID), Inception Score (Salimans et al., 2016, IS), and Precision/Recall (Kynkäänniemi et al., 2019). Due to computational constraints, we report FID-10K and sFID-10K rather than the more extensive -50K variants, allowing us to report statistics at multiple diffusion lengths for each model. Furthermore, to analyze compute allocation in SparseDiT models we use the image classes from Imagenet and estimate patch classes using the semantic segmentation map from a pre-trained SegFormer model (Xie et al., 2021).

Results Figure 4 and Appendix B show the FLOPs-normalized performance of 50%, 25%, and 12.5% SparseDiT models, as well as a vanilla DiT model. At the same number of FLOPs, we observe that 50% and 25% capacity SparseDiT models outperform DiT at all numbers of FLOPs in FID, sFID, IS, and Precision (with the exception of sFID at the highest number of FLOPs). Furthermore, the 12.5% capacity model is competitive with DiT in these metrics, particularly at lower FLOPs levels when SparseDiT is able to perform significantly more diffusion timesteps than DiT, which using the same number of FLOPs for generation. On metrics other than sFID, we find that the 12.5% capacity model underperforms higher capacity SparseDiT models. However, a constant capacity is used for both MLP and self-attention blocks, which have different routing behavior (Figure 1), and therefore distinct constraints on capacity. While not explored further here, the investigation of these constraints provides a promising avenue for future work.

Figure 5: High-loss patch and image classes are allocated more compute—mean routing percentage for patch and image classes is shown (right), as well as generated samples from the image classes with highest and lowest compute allocations (left). Only the MLP routing percentage is presented for image classes since the self-attention capacity is constant per sample. Results are shown for a 25% capacity model—results for both 12.5% and 50% capacity models can be found in Appendix C.

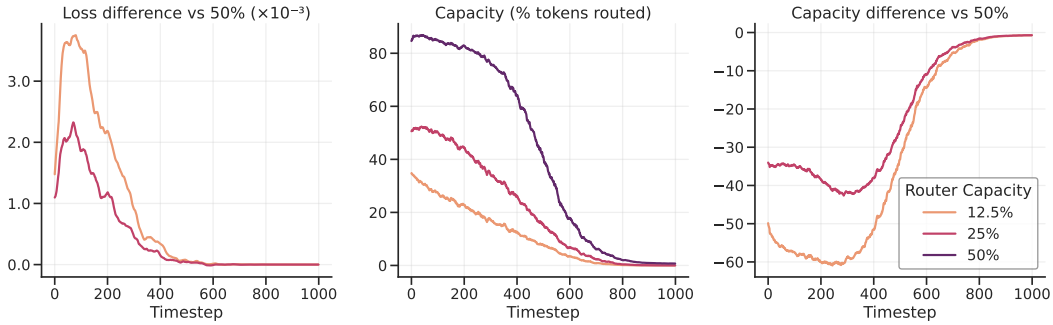


Figure 6: SparseDiT compute allocation increases at lower noise levels—we present the dynamic MLP router capacity across diffusion timesteps, as well as differences in capacity and loss.

Compute Allocation over Sample and Patches To understand how SparseDiT allocates compute between samples and image patches, we analyze the relationship between routing percentage and model loss on various sample and patch classes (Figure 5, Appendix C). Namely, we compute the mean loss and token routing proportion of SparseDiT models on all image and patch classes across uniformly sampled diffusion timesteps. Since self-attention capacity is constant on all samples, we show only the MLP routing proportion for image classes. We observe a strong positive correlation between image and patch loss with the proportion of routed tokens for that class, suggesting that SparseDiT correctly allocates more compute to harder classes. Furthermore, we observe that the "shower" patch class is an outlier for every model capacity, achieving the highest loss with relatively little compute allocation. We hypothesize that this is due to the highly entropic nature of such patches, meaning that increased compute provides limited reduction in loss.

Compute Allocation over Noise Levels In Figure 6, we present the compute allocation of samples over noise levels (diffusion timesteps), as well as the difference in diffusion loss of SparseDiT models with different capacity levels. Again, we show only the MLP routing proportion since the self-attention capacity is constant across samples. We observe increased compute allocation at lower noise levels (smaller timesteps) for all models, as well as a greater difference in allocated capacity between models, validating the qualitative example in Figure 1. This is understandable since the samples with less noise preserve more of the original signal, therefore allowing lower loss to be achieved with high compute allocation. Furthermore, we observe that the difference in loss between SparseDiT models is roughly proportional to the difference in compute allocation between them, suggesting that high-capacity models effectively leverage their capacity at low noise levels to decrease loss. At very low levels of noise, the differences in loss and allocated capacity sharply decrease. We hypothesize that this is due to the high level of precision required to estimate noise, which fails to leverage additional allocated compute.

4.2 Video World Modeling

Experimental Setup We use the raw video dataset from the 1X World Modeling Challenge (1X, 2025), containing approximately 100 hours of humanoid robot POV video at 30 frames-per-second and 512×512 resolution, as well as continuous controller actions for the robot at every frame. To embed these videos for latent diffusion, we use the pre-trained Cosmos Tokenizer (NVIDIA, 2025), selecting the continuous variant with an $8\times$ spatial and temporal downsample factor. This embeds 8 contiguous video frames with dimension $8 \times 512 \times 512 \times 3$ into a latent frame with dimension $1 \times 64 \times 64 \times 16$. During training, we condition models on two latent context frames, equivalent to 16 image frames, as well as the next 8 robot actions (corresponding to 1 latent frame). We train models to denoise the next latent frame conditioned on this context, and autoregressively sample latent frames during evaluation.

In order to support efficient training on video data, we also introduce spatio-temporal (ST) variants of DiT and SparseDiT, ST-DiT and ST-SparseDiT. These separate the self-attention operation into separate spatial and temporal attention operations, in which tokens attend to other tokens in the same frame, or other tokens in the same position in *previous* frames respectively. We train ST-DiT and ST-SparseDiT using the DiT-L/2 configuration for 200K steps, requiring 4 days on 8 H100 GPUs.

Results Following the challenge guidelines, we evaluate PSNR of the 60th generated video frame (2.0 seconds in the future) against the true frame on the validation subset. To increase clarity and evaluate the model closer to its training setup—that is, predicting the next 8 frames—we also evaluate generated frames at 15 and 30 frame intervals (0.5 and 1.0 seconds). Comparing DiT to SparseDiT with a capacity of 12.5%, we observe improvements in PSNR at 1.0s and 2.0s intervals, with the DiT model using 50% more FLOPs than SparseDiT (Table 1). This improvement in performance is likely attributed to the increase in parameters in the SparseDiT model. However, this is rarely a bottleneck for real-world deployment, with inference and training cost being much more significant factors.

Table 1: ST-SparseDiT has more parameters, fewer FLOPS, and higher performance on the 1X World Modeling dataset—performance of a 12.5% capacity SparseDiT model is shown.

Model	Parameters	GFLOPs	PSNR +0.5s	PSNR +1.0s	PSNR +2.0s
ST-DiT	652M	1238.0	19.7	18.1	17.2
ST-SparseDiT	943M	**823.5**	19.7	**18.6**	**17.4**

Compute Allocation over Optical Flow In video generation, models are conditioned on previous frames, making it trivial to denoise future tokens corresponding to static objects (which could be achieved by a simple copy operation). We therefore hypothesize that dynamic objects (those with high optical flow) should therefore benefit from increased compute allocation in SparseDiT models. To evaluate this, we visualize the mean routing percentage and the absolute pixel difference against the previous frame, as a simple approximation of optical flow (Figure 7). We observe a strong positive correlation between frame difference and routed tokens, validating our hypothesis and confirming that SparseDiT uses more FLOPs to generate frames with many dynamic objects.

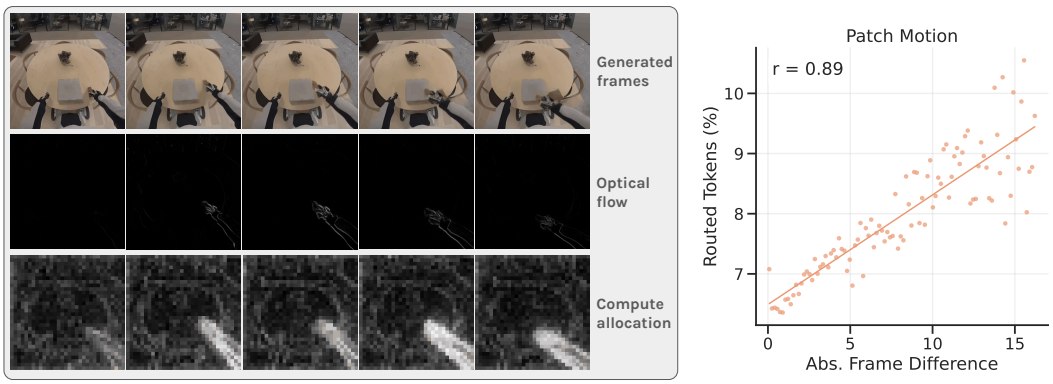


Figure 7: Compute allocation is strongly correlated with video motion—we show mean routing percentage for binned values of patch difference (clipped at the 99th percentile). Patch difference is computed as the magnitude of the difference between generated patches in adjacent frames and the model is trained on the 1X World Modeling Dataset, with a capacity of 12.5%.

5 Related Work

Transformers have become the predominant architecture for flow-based generative modeling, in both image generation with models such as DiT (Peebles and Xie, 2023), DALL-E 3 (Goh et al., 2023), and Latte (Ma et al., 2024), as well as video generation in GAIA-2 (Russell et al., 2025), Genie 2 (Parker-Holder et al., 2024), Mochi (GenmoAI, 2024), Sora (Brooks et al., 2024), and MovieGen (Polyak et al., 2024). This paradigm naturally invites adaptive compute allocation, since denoising difficulty varies across noise levels and spatio-temporal regions. Furthermore, many of the downstream applications of these models, such as large-scale image generation and real-time video generation, have strict constraints on latency that require fast inference. Despite this, adaptive computation techniques have not yet been adopted in these state-of-the-art models.

However, research interest in this area has recently increased. Wang et al. (2024) learn sparsity masks for pre-trained diffusion models without content-dependent selection, while Xi et al. (2025) investigate sparsity in spatial-temporal attention maps, applying an online profiling strategy to generate sparse attention patterns for video diffusion transformers. Zhao et al. (2025) introduce an approach for dynamic width reduction conditioned on diffusion timesteps, as well as patch sparsity in their MLP block, using an auxiliary loss to balance overall FLOPs. By contrast, our method dynamically selects tokens within both attention and MLP blocks, preserving tensor shapes for efficient batch inference with per-sample flexibility in the MLP block, and without any auxiliary loss. Furthermore, our work uncovers correlations between routing proportion and semantic attributes—such as optical flow magnitude and semantic classes—offering interpretability alongside performance gains.

Outside of diffusion modeling, adaptive computation has become increasingly prevalent in transformer models, in order to allocate resources efficiently based on input complexity. Xin et al. (2020) propose a language model architecture with adaptive depth via an early exiting mechanism. Inspiring this work, mixture-of-depths (Raposo et al., 2024) uses a routing mechanism to limit the capacity of transformer layers in autoregressive language models. While this method shows improvements in the FLOPs-normalized performance frontier of such models, we find diffusion to be an even more suitable paradigm for this architecture, as the models avoid causal structure interfering with the top-K operation. Mixture-of-experts mechanisms (Shazeer et al., 2017; Fedus et al., 2022, MoE) are indirectly related to dynamic compute allocation and have also been applied to diffusion modeling (Shi et al., 2025)[1], focusing on specialization of MLP heads rather than selective application. Finally, a range of methods orthogonal to model sparsity have been applied to accelerate the sampling process in diffusion. These commonly reduce the sampling timesteps, via more efficient integration schemes (Song et al., 2021; Lu et al., 2022) or progressive distillation (Salimans and Ho, 2022).

6 Conclusion

Summary SparseDiT represents a significant advancement in efficient generative modeling, effectively addressing a computational inefficiency inherent in existing diffusion models by dynamically allocating compute across tokens, samples, and noise levels. By employing a minimal yet powerful routing mechanism, SparseDiT demonstrates considerable performance improvements at the same level of compute, on both image and video generation tasks. Our evaluation also provides an intuitive understanding of the learned routing mechanism, highlighting SparseDiT's capability to align computational effort with semantic importance and temporal dynamics. Our work thus provides a step towards unlocking generative modeling in latency and hardware-constrained settings and offers insights into the distribution of computational resources within diffusion models.

Limitations and Future Work While our work provides an extensive evaluation, it is ultimately limited by our computational resources, providing the opportunity for further evaluation with alternate datasets, architectures, and generative algorithms. Namely, image generation datasets such as ArtBench (Liao et al., 2022) and Food (Bossard et al., 2014), as well as video generation datasets such as Taichi-HD (Siarohin et al., 2019) and Minecraft VPT (Baker et al., 2022), would strengthen confidence in the performance of SparseDiT. Regarding architectures, our routing mechanism could be similarly applied to other flow-based generative architectures such as Latte (Ma et al., 2024), as well as alternative algorithms such as flow matching (Lipman et al., 2022). In contrast to our evaluation, which trains SparseDiT models from scratch, it would be fruitful to investigate the finetuning of pre-trained, open-source generative models—such as DiT-XL (Peebles and Xie, 2023) and Mochi (GenmoAI, 2024)—with the addition of the SparseDiT routing mechanism. This would enable post-hoc addition of sparsity to large-scale models, with a limited computational budget.

Finally, SparseDiT introduces a capacity hyperparameter for the MLP and self-attention blocks, providing a range of promising avenues for further investigation. While our method avoids auxiliary losses, the addition of such a loss could enable automatic tuning of this capacity, with a regularization penalty against high compute budgets. Furthermore, we use a constant capacity for both MLP and self-attention blocks in this work. It is clear that these blocks have distinct allocation dynamics (Figure 1), meaning their optimal capacities likely differ and should be investigated.

[1] Shi et al. (2025) was released after March 1st 2025, and is thus considered contemporaneous work under NeurIPS 2025 guidelines, however, we cite it here for completeness.

References

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

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*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

1X. 1x world model challenge raw dataset, 2025. URL `https://huggingface.co/datasets/1x-technologies/world_model_raw_data`. CC BY-NC-SA 4.0 license.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

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

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

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. *arXiv preprint arXiv:1701.06538*, 2017.

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.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter C. Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Kaiyu Yang, Jacqueline H Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A study of face obfuscation in imagenet. In *International conference on machine learning*, pages 25313–25330. PMLR, 2022.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

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.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

NVIDIA. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Gabriel Goh, James Betker, Li Jing, Aditya Ramesh, Tim Brooks, Jianfeng Wang, Lindsey Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Prafulla Dhariwal, Casey Chu, and Joy Jiao. DALL-E 3. OpenAI Technical Report, 2023. `https://cdn.openai.com/papers/dall-e-3.pdf`.

Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024.

Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025.

Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model. Google DeepMind Blog, December 2024. `https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/`.

GenmoAI. Mochi: Asymmetric diffusion transformer for video generation. urlhttps://github.com/genmoai/mochi, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence W. Y. Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI technical report, 2024.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Kafeng Wang, Jianfei Chen, He Li, Zhenpeng Mi, and Jun Zhu. Sparsedm: Toward sparse efficient diffusion models. *arXiv preprint arXiv:2404.10445*, 2024.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. In *International Conference on Learning Representations (ICLR)*, 2025.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2246–2251, 2020.

Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, Pengfei Wan, Di Zhang, and Kun Gai. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks. *arXiv preprint arXiv:2206.11404*, 2022.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer, 2014.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made regarding model performance and analysis of model behavior are supported by our experiments in image and video generation, as detailed in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss our method's limitations and propose suggestions for future work based on these in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All details of the method architecture and hyperparameters are detailed in Section 4, for which we use the same model configuration as Peebles and Xie (2023) for all non-novel components.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not currently release our codebase, however, our work requires a simple extension to an existing, open-source codebase, which we cite in our work. This can be easily completed from the provided method description to reproduce all of our results, and we plan to release code upon publication of the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: These are detailed in the "Experimental Setup" paragraph of each evaluation subsection.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the emergent properties of large-scale generative models, all of our experiments use large models that are significantly computationally expensive to train. In the case of our video model experiments, these require multiple days over 8 H100 GPUs, costing thousands of dollars. This means we do not train multiple models to derive confidence intervals in performance. While we could have trained models at smaller scale to allow this, it is unclear whether the insights derived would have been useful, as these model would have had very poor performance. We note that our image generation experiments contain multiple models at varying capacity, giving an estimate of the variance of our evaluation metrics across the model class.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide an estimate of the computational requirements of our experiments in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the Code of Ethics and believe the work conforms with it entirely.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed, as it is an technical advancement to an existing generative modeling approach.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not release data or models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We appropriately cite both datasets used in our experiments, Imagenet (Deng et al., 2009) and the 1X World Modeling Challenge (1X, 2025).

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not use human subjects for any of our experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not use human subjects for any experiments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs for any important or original components of our research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Router Score Distribution

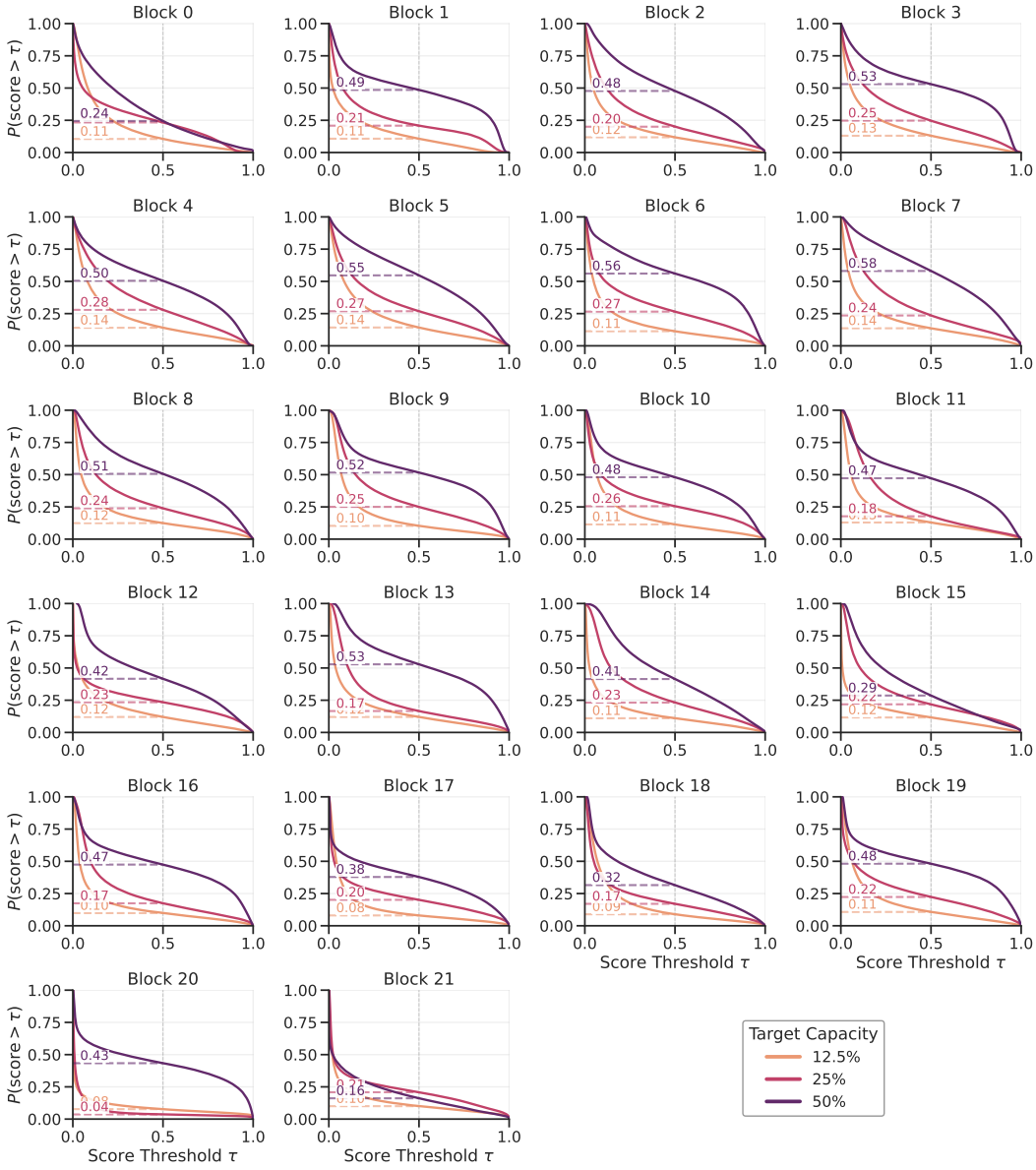


Figure 8: Per-block router scores for various SparseDiT-L/2 router capacities—score distribution varies in each block, but most blocks learn a target threshold close to 0.5, making it suitable as a default value.

B Further ImageNet Performance

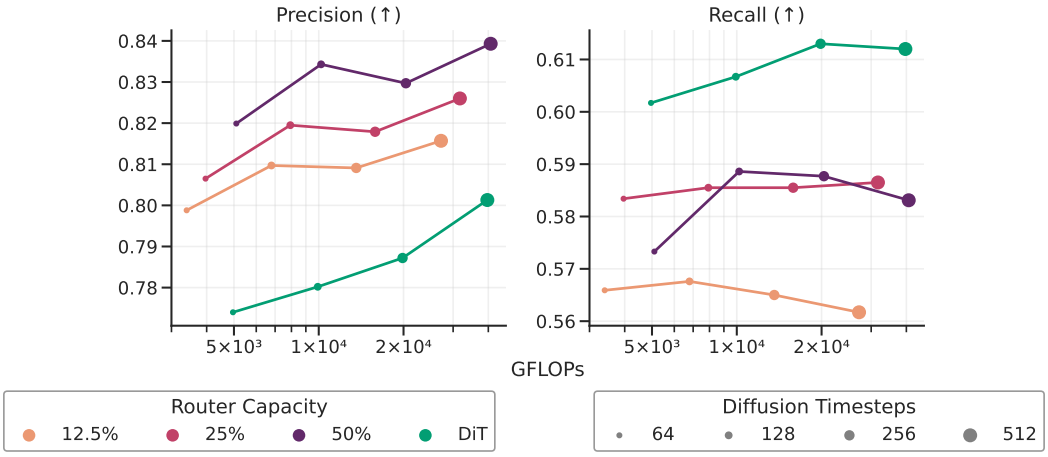


Figure 9: ImageNet performance.

C Router Capacity Correlations

(a) 12.5% capacity.

(b) 50% capacity.

Figure 10: High-loss patch and image classes are allocated more compute—mean routing percentage for patch and image classes is shown, with only the MLP routing percentage being shown for image classes due to the self-attention capacity being constant per sample. Notably, the outlier patch classes (maximum loss, plus maximum and minimum routing percentage) are the same for both 12.5% and 25% capacity models (Figure 5).