DYNAMIC MIXTURE-OF-EXPERTS FOR VISUAL AUTOREGRESSIVE MODEL*

Jort Vincenti[1], **Metod Jazbec**[1], **Guoxuan Xia**[2]
[1]University of Amsterdam, UvA-Bosch Delta Lab [2]Imperial College London

ABSTRACT

Visual Autoregressive Models (VAR) offer efficient and high-quality image generation but suffer from computational redundancy due to repeated Transformer calls at increasing resolutions. We introduce a dynamic Mixture-of-Experts router integrated into VAR. The new architecture allows to trade compute for quality through scale-aware thresholding. This thresholding strategy balances expert selection based on token complexity and resolution, without requiring additional training. As a result, we achieve $\sim 20\%$ fewer FLOPs, $\sim 11\%$ faster inference and match the image quality achieved by the dense baseline.

1 INTRODUCTION

Autoregressive Modelling has not matched diffusion models in image generation quality despite demonstrating strong performance in text Ma et al. (2025). The Visual Autoregressive Model (VAR) Tian et al. (2024a) bridges this gap by shifting from next-token to next-scale prediction, iteratively refining images by invoking an autoregressive transformer to predict each subsequent resolution. VAR is the first model to reach diffusion models in image quality while also improving runtime.

A key goal in autoregressive modeling is improving computational efficiency without sacrificing performance. Dynamic methods address this by allocating computation based on token complexity Graves (2016). Currently, VAR still makes multiple static transformer calls per scale, ignoring varying complexity across tokens and scales. Exploiting these variations could enhance efficiency by focusing computation on complex regions and progressively reducing resources at finer scales Rao et al. (2021). Dynamic allocation at the scale level remains unexplored, presenting an opportunity to improve VAR's efficiency. As a result, we adapt a dynamic Mixture-of-Experts (MoE) framework that routes experts based on token complexity, specifically for VAR's coarse-to-fine structure Szatkowski et al. (2024). We make the following contributions:

1. A variant of the VAR model where each feed-forward block is replaced by a Mixture-of-Experts layer with an adaptive router, enabling the model to allocate resources across tokens and scales based on their complexity.
2. An inference-tunable and scale-dependent threshold that activates fewer experts at higher resolutions, matching the observation that fine-scale tokens need less computation.
3. An empirical validation on ImageNet Deng et al. (2009) showing 19% fewer FLOPs and 11% faster wall-clock time while staying within 1% Fréchet Inception Distance score Heusel et al. (2018) of the VAR depth 16 baseline.

2 PRELIMINARIES

2.1 VISUAL AUTOREGRESSIVE MODELLING (VAR)

Rather than predicting tokens sequentially, VAR Tian et al. (2024b) redefines image generation as a coarse-to-fine process across multiple resolutions. An image is represented as a hierarchy of discrete token maps, ranging from a low to high resolution. The model then learns to autoregressively predict each finer scale conditioned on the coarser ones, progressively refining structure and texture until the full image is reconstructed. While this hierarchical design improves efficiency compared to

*Code is available at https://github.com/JortVincenti/DMoE-VAR.

raster-scan autoregression, related works for accelerating inference in VAR is discussed in Appendix A.

Stage 1: Multi-scale VQ-VAE encoder. VAR employs a multi-scale VQ-VAE to discretize images into a sequence of token maps at progressively coarser resolutions. The process starts from the full-resolution image, which is encoded into a dense feature map. This feature map is then quantized by assigning each vector to its nearest entry in a shared codebook, producing one map of tokens. This procedure is repeated multiple times, until the entire image is summarized by a single token. The result is a hierarchy of token maps which provide the discrete multi-scale representation used in Stage 2.

Stage 2: Next-scale prediction. After obtaining the token hierarchy $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_K)$, a Transformer is trained autoregressively to predict each scale conditioned on previous scales. During training, the Transformer takes the sequence $([\mathtt{s}], \mathbf{r}_1, \ldots, \mathbf{r}_{K-1})$ as input and employs a block-wise causal attention mask, ensuring that tokens at scale k can only attend to earlier scales. At inference, tokens are generated recursively from coarsest ($\mathbf{r}_1$) to finest ($\mathbf{r}_K$), after which a VQ-VAE decoder reconstructs the final image. While generation at each scale occurs in parallel, the computational cost increases significantly with higher resolutions due to the tokens growing quadratically from just one token at k=1 to 256 at k=K.

2.2 Dense-to-Dynamic-k Mixture-of-Experts (D2DMoE)

From dense FFN to experts. In each Transformer block the two-layer FFN computes the hidden vector $\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$ and outputs $F(\mathbf{x}) = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$. To make the activations $\mathbf{h}$ even sparser than with σ set to ReLU, D2DMoE adds a Hoyer penalty Hoyer (2004) to the cross entropy loss:

$$L(x) = L_{\text{CE}}(\hat{y}, y) + \alpha\, L_s(x), \qquad L_s(x) \quad = \frac{1}{L}\sum_{\ell=1}^{L} \frac{(\sum_i |h_i^{(\ell)}|)^2}{\sum_i (h_i^{(\ell)})^2} \tag{1}$$

where $\mathbf{h}^{(\ell)}$ is the post-activation vector of FFN block ℓ, L is the number of blocks, and α controls sparsity strength. After sparsity-aware training, columns of $\mathbf{W}_1$ respond to subsets of tokens. By grouping these columns via k-means clustering into balanced clusters Zhang et al. (2022), D2DMoE rearranges the original FFN parameters into multiple distinct experts creating a Mixture-of-Experts layer.

Regression-based routing. D2DMoE introduces a routing mechanism framed as a regression task rather than the traditional top-k selection. A lightweight router R predicts the ℓ_2-norm of each expert's output for an input token $\mathbf{x}$, minimising the difference between predicted and true norms via:

$$L_r(x) = \frac{1}{n}\sum_{i=1}^{n} \big(R(x)_i - \|E_i(x)\|\big)^2 \tag{2}$$

At inference, experts are dynamically selected using a relative thresholding mechanism. An expert would be selected only if $R(x)_i \geq \tau \cdot \max(R(x))$, where the adjustable scalar $\tau \in [0, 1]$ provides flexible control over computational cost and performance without needing to retrain.

3 Dynamic Mixture-of-Experts for Visual Autoregressive Model

Expert Construction. As detailed in section 2.2, at every Transformer layer, we replace the standard VAR's FFN with a MoE block. Experts are built offline by clustering the sparse FFN weights from the ReLUfied model Mirzadeh et al. (2023) finetuned with the Hoyer-norm regularisation (1). We then apply the same equation as in (2) to train a router that regresses the ℓ_2-norm of each expert.

τ-Based Expert Selection. At inference, we route tokens through a stack of S progressively finer scales, with each scale $s = 1, \ldots, S$ indexed by increasingly thresholds $\tau_1 < \tau_2 < \cdots < \tau_S$. At each scale s, the router outputs a vector of norm predictions $R^s(x) \in \mathbb{R}^E$ across the E experts. Expert selection is then determined by applying the following rule to each expert i:

$$G^s(x)_i = \begin{cases} 1 & \text{if } R^s(x)_i \geq \tau_s \max_j R^s(x)_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

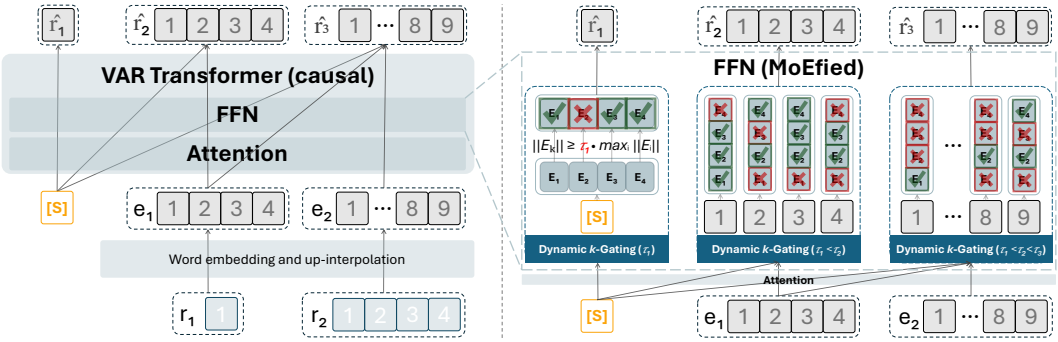


Figure 1: **Generation pipeline.** *Left:* The coarse-to-fine decoder of VAR performing next-scale prediction: it takes $([\mathrm{s}], r_1, r_2, \ldots, r_{K-1})$ as input to predict $(\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_K)$. *Right:* The FFN block is replaced with a dynamic-k gating MoE module. It executes expert E_j only when $\|E_j\| \geq \tau_s \max_i \|E_i\|$, filtering the experts by ℓ_2-norm. Because the thresholds grow with resolution $\tau_1 < \cdots < \tau_S$, many experts are used at coarse scales while fewer are consulted at fine scales.

where $G^s(x)_i$ indicates whether expert i receives the token x at scale s. This τ-based selection ensures that compute allocation is both token- and scale-aware: lower thresholds at coarse scales admit more experts, while higher thresholds at finer scales enforce greater sparsity with fewer selected experts (as illustrated in Fig. 1).

Overall, our method unifies the coarse-to-fine paradigm of VAR with the adaptive computational load of the MoE framework. By merging routing granularity (token-level) with resolution granularity (scale-level), we achieve:

1. **Scale-aware budget scheduling** via τ_k, whereby each successive scale requires less compute.
2. **Token-aware compute allocation**, which is crucial for handling the highly skewed sparsity patterns within larger scales.

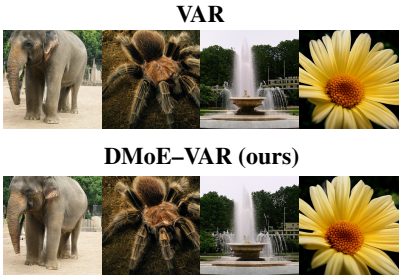
4 Results

Implementation Details. All experiments use the pre-trained VAR-d16 baseline, fine-tuned for two epochs with the sparse cross-entropy loss in Eq. (1). The full set of hyperparameters is provided in Appendix B. All experiments are conducted on NVIDIA A100 GPUs, measuring floating-point operations (FLOPs) using `fvcore`[1] and Fréchet Inception Distance (FID) using `torch-fidelity`[2], with evaluation on ImageNet using 10,000 samples Deng et al. (2009).

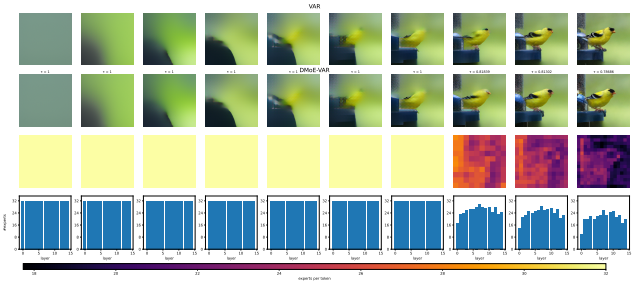
Design Exploration. Our design exploration in Appendix C provides several key insights. First, we find that Hoyer regularization mainly induces sparsity in early scales, contributing little to computational savings, while ReLUfication yields more substantial sparsity overall (Appendix C.1). Second, practical speed-ups require configuring fewer, larger experts and limiting MoEfication to later scales (Appendix C.2). Finally, routers with higher sparsity regularisation ($\alpha = 0.1$) achieve better FID, likely due to simpler routing from zero-valued expert norms (Appendix C.3). These findings motivate our final design which fine-tunes the ReLUfied model with Hoyer sparsity ($\alpha = 0.1$), then applies MoE layers using 32 experts of size 128, using the model only on the last three scales.

Main Results. Our method reduces FLOPs by 19% and inference time by 11%, while keeping FID within 1% of the dense VAR baseline. These gains result from applying MoE layers only at the last three scales, allowing lower τ thresholds without losing image quality. We first qualitatively compare our new architecture to the VAR baseline in Figure 2a. Images from both methods appear identical, highlighting our method's strength in preserving semantics with identical seeds. Second, in Figure 2b we examine generation throughout the scales. The VAR baseline is matched by keeping scales 1–7 dense. At scale 8, the router activates many experts per token to refine uncertain regions. By the last scale, it ignores settled background and triggers only a few experts around fine edges and textures. This progressive narrowing substantially reduces both FLOPs and time while consistently maintaining high overall image quality (additional comparisons and router maps are in Appendix D).

[1] https://github.com/facebookresearch/fvcore
[2] https://github.com/toshas/torch-fidelity



(a) Image comparison



(b) Distribution of experts.

Figure 2: **Quality–efficiency trade-off and expert routing behaviour. (a)** Qualitative comparison of DMoE-VAR and VAR samples. **(b)** Expert routing and generation patterns, shown across scales. *First and Second row:* generated images from the VAR baseline and DMoE-VAR. *Third row:* heat-maps of total experts allocated per token across layers. *Fourth row:* bar plots of the average number of experts used at each scale.

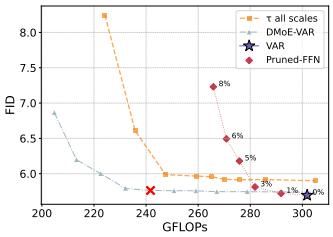


Figure 3: FID (↓) vs GFLOP (↓). Different optimization methods utilized with VAR.

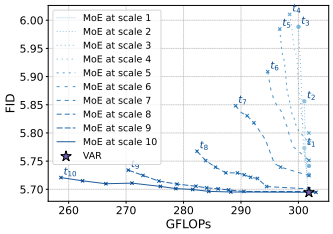


Figure 4: FID (↓) vs GFLOP (↓). MoEfication applied to a single scale.

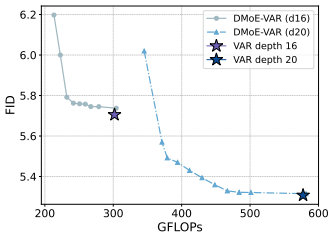


Figure 5: FID (↓) vs GFLOP (↓). DMoE-VAR integrated into different model depths.

τ-selection In Figure 3, "DMoE-VAR" employs a fixed τ for the last three scales. Varying τ produces a FID to FLOP trade-off curve, with the red cross denoting the configuration used in Figure 2. This thresholding is applied post-training, enabling flexible compute allocation at inference. Applying a uniform threshold to all scales ("τ all scales") performs worse in image quality and efficiency as errors from early scales propagate and low token counts fail to amortize the expert-loading overhead. As a result, the main gains come from sparsifying the later scales. To validate that token-wise routing is still required, we introduce "Pruned-FFN", which prunes low-activation weights in the last three scales. The sharp performance drop confirms the necessity of our dynamic thresholding.

Ablation Study. In Figure 4, we analyse the sensitivity of different scales to sparsification by selectively applying our methodology at individual scales and varying the threshold τ. Scales 8–10 display the best FID to FLOP trade-off validating our targeted sparsification strategy at the last three stages. Finally, Figure 5 illustrates that our method benefits from increased model depth. Comparing the fixed-τ strategy across architectures, the FLOP/FID curve of the depth 20 model outperforms the depth 16 one. We attribute this to deeper models containing more redundant computation, which our dynamic approach can exploit to achieve greater FLOP savings.

5 Conclusions

We introduce a dynamic Mixture-of-Experts mechanism into Visual Autoregressive Models (VAR) to address the computational inefficiency of dense scale-wise decoding. Our scale-aware sparsification adaptively selects experts per token and resolution, exploiting redundancy in both domains to cut FLOPs while preserving image quality. Fine-scale representations show high spatial redundancy, enabling more aggressive sparsification without loss. The proposed adaptive thresholding strategy enables this selective computation, adjusting dynamically to the content and complexity of each image during generation. Future work includes fine-tuning only the last three scales for targeted sparsity and developing routing that adapts better to semantic classes.

REFERENCES

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps, 2025. URL https://arxiv.org/abs/2501.09732.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction, 2024a. URL https://arxiv.org/abs/2404.02905.

Alex Graves. Adaptive computation time for recurrent neural networks. *ArXiv*, 2016.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021.

Filip Szatkowski, Bartosz Wójcik, Mikołaj Piórczyński, and Simone Scardapane. Exploiting activation sparsity with dense to dynamic-k mixture-of-experts conversion, 2024. URL https://arxiv.org/abs/2310.04361.

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. Ieee, 2009.

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, 2018. URL https://arxiv.org/abs/1706.08500.

Keyu Tian, Li Wang, and Hao Zhao. Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction. In *Neural Information Processing Systems (NeurIPS)*, 2024b.

Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints, 2004. URL https://arxiv.org/abs/cs/0408058.

Yikun Zhang, Zhi He, Shuming Wang, and Ning Jia. Moefication: Transformer feed-forward layers are mixtures of experts. *Findings of ACL 2022*, 2022.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models, 2023. URL https://arxiv.org/abs/2310.04564.

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware kv cache compression. *arXiv preprint arXiv:2505.19602*, 2025. URL https://arxiv.org/abs/2505.19602.

Zigeng Chen, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Collaborative decoding makes visual auto-regressive modeling efficient. *arXiv preprint arXiv:2411.17787*, 2024.

Rui Xie, Tianchen Zhao, Zhihang Yuan, Rui Wan, Wenxi Gao, Zhenhua Zhu, Xuefei Ning, and Yu Wang. Litevar: Compressing visual autoregressive modeling with efficient attention and quantization. *arXiv preprint arXiv:2411.17178*, 2024. URL https://arxiv.org/abs/2411.17178.

Chengyue Gong, Yekun Ke, Xiaoyu Li, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. On computational limits of flowar models: Expressivity and efficiency, 2025. URL https://arxiv.org/abs/2502.16490.

A EFFICIENT INFERENCE IN VAR

To improve VAR's efficiency, ScaleKV and Collaborative Decoding (CoDe) split the scale architecture into a "drafter" and a "refiner" stage. The drafter is responsible for generating a coarse, low-resolution version of the output sequence, while the refiner incrementally improves this draft to achieve high-quality results. ScaleKV reduces memory overhead by mapping transformer layers to these roles and compressing the key-value caches at higher resolutions Li et al. (2025). CoDe similarly adopts a two-stage decoding structure, using a large-capacity drafter for coarse representations and a lightweight refiner for fine adjustments, enabling more targeted and efficient computation Chen et al. (2024). These architectures show that dividing decoding into specialized components can yield substantial efficiency gains without compromising output quality Xie et al. (2024); Gong et al. (2025).

B HYPER-PARAMETER TABLE

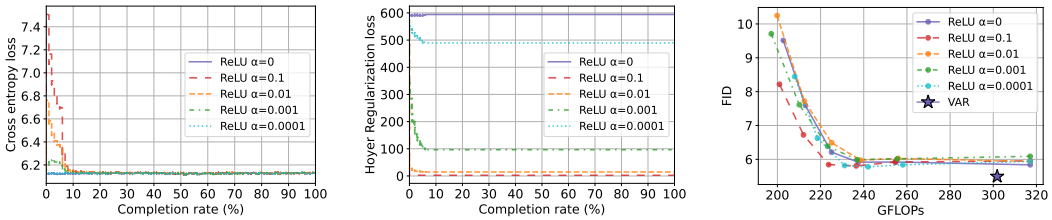
Hyper-parameter	**Value**	**Description**
	Sparsification	
Batch Size	512	Global batch size
Epochs	2	Training epochs
Loss Type	`ce`	Cross-entropy loss
Optimizer	`adam`	Adam Optimizer
Learning Rate	2×10^{-5}	Learning rate
Weight Decay	0.05	Regularization
Scheduler	`linear`	Learning Rate Scheduler
Warmup Steps	0.2 epochs	Initial learning rate warm-up
Gradient Norm Clip	1.0	Gradient clipping norm
	Expert Split	
Expert Size	128	Hidden dimension of each expert
Experts Class	`execute_all`	Expert selection mode
Activation	GeLU	Activation function between Experts
	Router Training	
Epochs	2	Training epochs
Batch Size	256	Batch size
Learning Rate	1×10^{-3}	Optimizer learning rate
Router Loss Type	`mse`	Mean Squared Error
Router Depth	2	Hidden layers in router
Router Width	128	Width of router hidden layers
Activation	`gelu`	Router internal activation
Output Activation	`abs`	Router output activation
Labels Norm	2	Label normalization
Number of Experts	128	Experts width per MoE layer
	Evaluation (FID/IS)	
Batch Size	128	Batch size during sampling
Tau	[0.81839, 0.81302, 0.78686]	Tau per scale
Tau as List	True	Interpret tau as list
Expert Index Switch	7	Scale to start replacing with MoE
CFG	1.5	Classifier-free guidance
Top-P	0.96	Nucleus sampling probability
Top-K	900	Sampling truncation parameter
Forward Mode	`dynk_max`	Expert selection mode
Samples per Class	10	Number of images per class
Random Seed	0	Deterministic sampling
TF32 Enabled	True	TensorFloat32 usage

Table 1: Hyperparameter Settings for all Experiments

C DESIGN EXPLORATION

To mitigate errors from the router, we employ an "oracle" router strategy in Sections C.1 and C.2. Specifically, tokens are passed through the expert modules twice: first, to compute the activation norms from (2) and, then, to apply dynamic thresholding based on determined τ values (Eq. (3)). We exclude the first pass from the fvcore measurements to obtain realistic FLOP estimates.

C.1 SPARSITY



(a) Cross-entropy loss (↓) (b) Hoyer regularization loss (c) FID (↓) vs GFLOPs (↓)

Figure 6: Training curves for our model fine-tuned under the combined loss $L = L_{\text{CE}} + \alpha\, L_s$ (see Eq. 1), where L_s is the Hoyer sparsity penalty, plotted for five different values of α over two epochs. For FID evaluation, we created 512 experts of size 8 using the oracle routing strategy.

We fine-tune models using the joint loss $L(x) = L_{\text{CE}}(\hat{y}, y) + \alpha L_s(x)$ (eq. (1)). As shown in Figures 6a and 6b, convergence is reached within roughly one-fifth of an epoch, requiring minimal additional fine-tuning steps. While the resulting models achieve comparable cross-entropy and FID scores, they differ in sparsity loss. We believe this happens because the sparsity regulariser mostly affects the early, complex scales, which have little impact on total FLOPs, while the later scales make up for the errors from earlier layers as they contain far more tokens. As a result, variations in sparsity translate into minimal FLOP savings. This points to ReLUfication as the dominant source of sparsity, a claim supported by our ablation without ReLUfication (Figure 7), where performance degrades quickly, showing Hoyer regularisation alone is insufficient for robust sparsity.

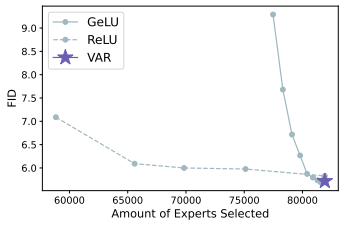
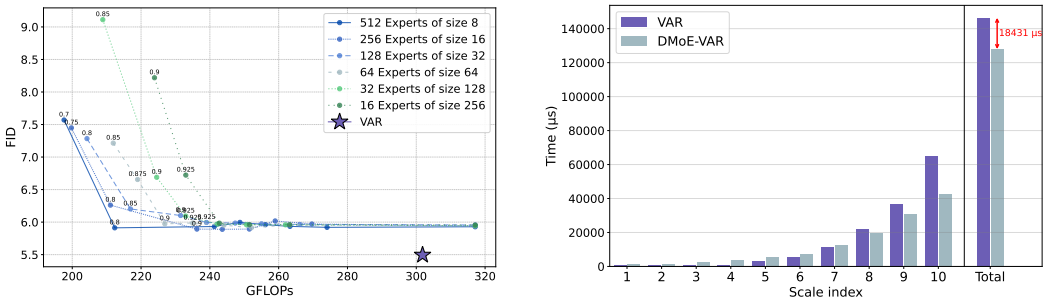


Figure 7: Comparison between the GeLU, ReLU and VAR models

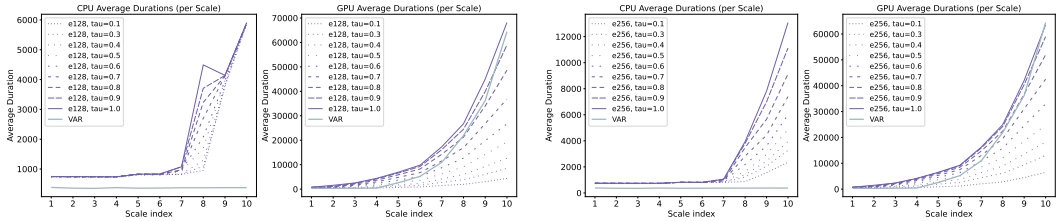
C.2 MOEFICATION AND EFFICIENCY



(a) Different Experts Sizes (b) Efficiency of VAR vs DMoE-VAR

Figure 8: **Expert sizes vs. compute and speed.** For the sparsified model fine-tuned under the combined loss $L = L_{\text{CE}} + \alpha\, L_s$ (Eqs. 1, $\alpha = 0.1$) using the oracle routing strategy. **(a)** FID (↓) vs. GFLOPs (↓) for various expert sizes and τ values (annotated). **(b)** Wall-clock time per scale comparing VAR and DMoE-VAR (32×128 experts, $\tau = 0.7$), showing combined CPU + GPU time for batch size 128.

To leverage sparse MoE architectures effectively, we evaluate the trade-off between expert granularity and performance (Figure 8a). The model achieves finer-grained representations as the number of experts increases, which is reflected in lower FID scores. However, due to GPU threading overhead for each expert, expert sizes of at least 128 are required to obtain runtime benefits. As a result, while a configuration of 512 experts of size 8 yields the best FID, practical deployment forces us to select the 32 experts of size 128, striking a balance between FID and speed-up.



(a) CPU and GPU time for 128 experts width.

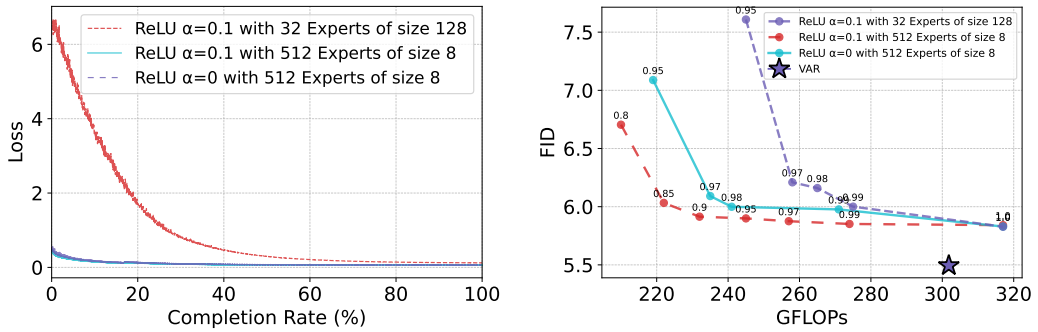
(b) CPU and GPU time for 256 experts width.

Figure 9: Running time for a single batch under the τ threshold rule.

It is important to note that expert size is not the only factor driving efficiency optimisations. Other parameters, such as the number of active experts (regulated by τ), CPU/GPU utilization, batch size and hardware specifications, all influence runtime. Figure 9 reports the wall clock time on the CPU and GPU for a single batch using the τ selection in (3). For a fixed batch size the GPU becomes faster when expert width grows, because each activation touches fewer experts. The CPU trend is reversed due to sorting of the experts for each token. Throughout the experiments we push the batch size until the model almost reaches an out of memory (OOM) event. On our hardware, a batch size of 128 is the largest configuration that fits. With this batch size, any smaller experts would result in an OOM, and increasing the batch size to 256 fails to run for any expert size. It is to note that this set-up is optimal for our machine, other systems will require different values.

In our most realistic setup, we compare the throughput of the VAR and DMoE-VAR model using 128-sized experts with a threshold of $\tau = 0.7$ (Figure 8b). Since GPU speed-ups scales with the amount of tokens, only the later scales (8–10) display meaningful acceleration, resulting in an overall runtime improvement of approximately 12%. These results underscore two practical guidelines: (i) fewer, larger experts are preferable for end-to-end efficiency, and (ii) MoEfication should only be applied on deeper scales where enough tokens are present to amortize the threading overhead.

C.3 Router Selection



(a) Router Loss

(b) FID for different routers

Figure 10: **Router analysis and hyperparameter tuning.** We optimize the router loss $L_r(z) = \frac{1}{n}\sum_{i=1}^{n}\left(R(z)_i - \|E_i(z)\|\right)^2$ using the previously sparsified model $L = L_{\text{CE}} + \alpha\, L_s$ (Eqs. 1) with $\alpha \in \{0, 0.1\}$ and two expert configurations (32 experts of size 128 and 512 experts of size 8). **(a)** Router loss (↓) vs. completion rate over two epochs. **(b)** FID (↓) vs GFLOPs (↓) for the different configurations.

Incorporating a learned router introduces additional parameters and potential prediction errors. Figures 10a and 10b illustrate the learned router's loss and FID. Notably, for equivalent expert sizes, the router achieves improved FID scores when trained with sparsity regularization $\alpha = 0.1$, compared to $\alpha = 0$. We hypothesize that this occurs because higher sparsity regularization results in more zero-valued expert norms, simplifying the router's prediction task and thus enhancing overall FID.

D VISUAL ANALYSIS

Figure 11: More qualitative comparisons of DMoE-VAR and VAR samples. Generated images were produced using classifier-free guidance and Gaussian smoothing.

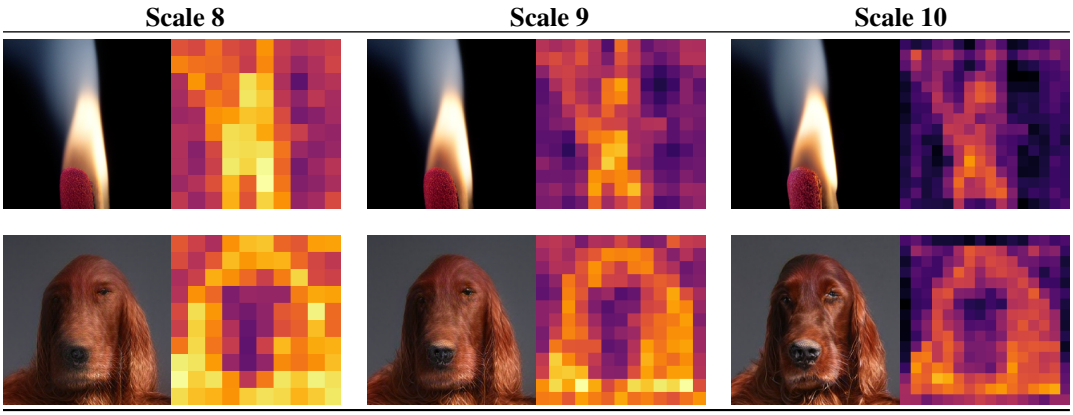


Figure 12: At each scale (8, 9, and 10), the generated image (left) is paired with its corresponding expert allocation map (right), obtained by summing the activated experts per token across both conditioned and unconditioned samples.

Figure 12 shows how DMoE-VAR allocates experts at scales 8 to 10 as τ decreases. At scale 8, the sparsity map reflects higher uncertainty and spreads compute across the whole image. As resolution grows finer, the experts are localised to the most important regions, separating the pattern into foreground and background. However, the dog's nose still triggers only a few experts, even if it lies at the centre of the image. We hypothesise that, with large expert width, just a small set of experts dominate the norms in such specialised regions, suppressing the others. While this focused allocation preserves global fidelity, it can miss very fine details compared to the dense VAR baseline. For example, the spider's eyes in Figure 11 are missing with our model.