REACT: RESPONSE-BASED EXPERT ASSEMBLY FROM CONTEXTUAL TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense feed-forward networks are unaffected by how their neurons are grouped: every neuron is evaluated. Sparse execution makes this grouping consequential, because each input can access only a few experts. An arbitrary partition can therefore retain pretrained weights while separating computations that an input needs together. We argue that upcycling should organize neurons according to their **responses to similar inputs**. To establish this correspondence, we introduce a **context-guided upcycling approach REACT**, **R**esponse-based **E**xpert **A**ssembly from **C**ontextual **T**rajectories, that uses frozen denoising trajectories to connect input contexts with pretrained neuron responses. We first form contexts from timestep-bucket-centered residuals, then use their associated responses to allocate intact neurons to routed experts. The same context index initializes the corresponding router entry, **coupling response-based expert construction with router initialization**, while global activation statistics select an always-active shared expert. Layer-wise reconstruction and end-to-end adaptation refine this initialization for sparse execution. On ImageNet-1K, REACT-M32 reports an FID point estimate of 2.21; the published dense DiT-XL/2 reference is 2.27. It stores 1.61B parameters while activating an FFN parameter count equal to 37.5% of the dense FFN—489M active non-router parameters per token, 27.6% below the dense model's 675M under the reported token-wise accounting.

1 INTRODUCTION

In a dense feed-forward network (FFN), how hidden units are grouped has no functional effect because every input evaluates every unit. Sparse execution removes this invariance: once a token reaches only its top-K experts, the grouping of pretrained units determines which computations remain accessible (Figure 1). Dense-to-sparse upcycling is therefore a correspondence problem: sparse routing must preserve access between input contexts and the pretrained unit responses they require, rather than merely placing dense weights into a sparse parameter layout.

This correspondence is particularly consequential for Diffusion Transformers (DiTs) (Peebles & Xie, 2023), whose dense FFNs process every spatial token at every denoising step, whereas a sparse mixture-of-experts (MoE) layer activates only a few experts per token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). Sparse upcycling reuses a dense checkpoint (Komatsuzaki et al., 2022), but weight reuse alone does not specify which pretrained functions each routing decision can access. Existing conversions replicate the dense FFN, partially reinitialize it, partition its neurons, or initialize the router from intermediate representations (Komatsuzaki et al., 2022; Nakamura et al., 2025; Zhang et al., 2022; Zhu et al., 2024; Jiang et al., 2025); recent LLM and CLIP methods couple expert construction with router initialization (Pei et al., 2026; Chu et al., 2026), and Dense2MoE converts DiTs through importance-based initialization and distillation (Zheng et al., 2025). We ask whether denoising trajectories can provide a single context index for two coupled tasks: allocating intact source FFN units to routed experts and initializing the corresponding router rows from the same trajectory contexts before sparse recovery begins.

Diffusion makes this nontrivial because the same FFN serves every noise level, so activations pooled over a trajectory mix between-timestep mean shifts with the within-timestep variation that distinguishes local contexts. We introduce REACT (Figure 2) to address this. Frozen denoising trajectories provide FFN inputs and unit activations; subtracting the corresponding original-timestep-bucket

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

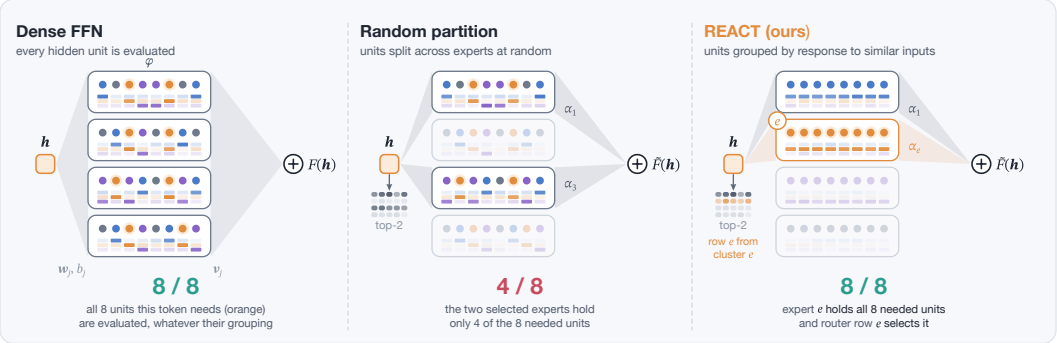

Figure 1: Schematic of unit access under sparse execution. Orange marks eight of 32 pretrained units needed by one token. Dense execution evaluates all 32 and reaches these eight, but a random partition leaves only four accessible through top-2 routing. REACT groups units by response and initializes each router row from the corresponding context, restoring access to all eight in this example under the same top-2 execution budget.

means and normalizing the residual directions yields a layer-local context partition. A trajectory-wide co-activation score selects units for an always-active shared expert, and cluster-conditioned response scores allocate units to routed experts, each copied intact as one input–nonlinearity–output tuple; padded units fill any remaining capacity. The normalized context centroids initialize the same-index rows of a learnable top-K router, so context e names both routed expert e and router row e. Layer-wise reconstruction against frozen dense teachers and end-to-end diffusion training then recover the sparse model.

Controlled recovery experiments on class-conditional ImageNet-1K at 256×256 support this construction. REACT reports lower FID than bundled random initialization at every evaluated checkpoint from 100K to 500K for both DiT-XL and DiT-M sources (2.54 versus 2.78 and 8.60 versus 10.26 at 500K), and a matched M-source contrast that holds expert assembly fixed favors centroid-based over random router initialization (8.60 versus 8.90). A separately trained DiT-XL/2 conversion, REACT-M32, reports 2.21 FID; the published DiT-XL/2 reference is 2.27 (Peebles & Xie, 2023). It stores 1.61B total parameters, activates 37.5% of the dense FFN parameters, and uses 489M active non-router parameters per token, 27.6% fewer than the dense model's 675M. Under the same profiling setup, it achieves a $1.47\times$ sampling speedup.

1. Layer-local token contexts constructed from timestep-bucket-centered denoising-trajectory residuals for dense-to-sparse DiT conversion.
2. A matched sparse initialization that allocates intact source units by cluster-conditioned response and initializes the corresponding router row from the same context index, coupling expert contents to router access.
3. Recovery curves at two source scales, partial initialization controls, routing diagnostics, and sensitivity studies that evaluate the complete pipeline, together with quality, activation, and latency operating points.

2 Related Work

Sparse experts for diffusion. Sparse MoEs activate a small expert subset per token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022), while shared-expert variants reserve always-active capacity for common computation (Dai et al., 2024). Switch-DiT and DiT-MoE introduce shared or token-choice experts along the denoising trajectory, and ProMoE augments routing with conditional and prototypical guidance (Park et al., 2024; Fei et al., 2024; Wei et al., 2026). These methods design routing and expert layouts for diffusion models. REACT instead studies how to initialize a shared-plus-Top-2 MoE from a dense checkpoint.

Dense-to-MoE upcycling and coupled initialization. Sparse Upcycling replicates a dense FFN, Drop-Upcycling partially reinitializes the copies, and MoEfication and LLaMA-MoE partition pretrained units (Komatsuzaki et al., 2022; Nakamura et al., 2025; Zhang et al., 2022; Zhu et al., 2024). Representation-based upcycling initializes routers from language-model states, activation-pattern restructuring constructs shared and routed LLM experts with an analytical router, and Cluster-Aware Upcycling couples activation clusters, centroid routing, and low-rank CLIP experts (Jiang et al., 2025; Pei et al., 2026; Chu et al., 2026). For DiTs, REACT derives clusters from timestep-centered trajectories, uses them to allocate intact units, and initializes the matching router rows. The router remains trainable throughout both recovery stages after its trajectory-based initialization.

Dense DiT conversion. Dense2MoE assigns pretrained units by a global Taylor-importance criterion and recovers a converted DiT through distillation (Zheng et al., 2025). REACT instead uses a layer-local trajectory partition for both unit allocation and router initialization. Appendix C covers methods that retain the dense FFN while adapting computation across timesteps or tokens, as well as diffusion MoEs trained from scratch (Zhao et al., 2025; Chang et al., 2026; Sehwag et al., 2025).

3 METHOD

Overview. REACT converts the FFNs of a pretrained Diffusion Transformer (DiT) into sparse mixture-of-experts (MoE) layers. Its key design is to derive expert contents and router initialization from one layer-specific partition of token contexts. For each layer, cluster e supplies the responses used to select units for routed expert e, and its normalized centroid initializes router row e. The router remains trainable during recovery.

Conversion is performed offline with the dense DiT frozen. Denoising trajectories provide FFN inputs and post-GELU activations. Pooled activation statistics select an always-active shared expert; cluster-conditioned responses select routed units. The sparse model is recovered by layer-wise teacher matching followed by end-to-end diffusion training (Figure 2).

3.1 SPARSE TARGET AND SOURCE-UNIT TRANSFER

Consider block $\ell \in \{1, \dots, L\}$ of a dense DiT with model width d and FFN width H, and let ϕ denote its tanh-approximated GELU. We represent hidden unit j by its input row $\boldsymbol{w}_{\ell,j}^\top$, input bias $b_{\ell,j}$, nonlinearity, and paired output column $\boldsymbol{v}_{\ell,j}$. An intact copy preserves the unit's computation. Summing these contributions over all H source units yields the dense FFN

$$F_\ell(\boldsymbol{h}) = \sum_{j=1}^{H} \boldsymbol{v}_{\ell,j}\, \phi\big(\boldsymbol{w}_{\ell,j}^\top \boldsymbol{h} + b_{\ell,j}\big) + \boldsymbol{b}_\ell^{(2)}. \tag{1}$$

The converted layer has one shared expert of width S and E routed experts of width R. Each expert retains the terms in Equation 1 indexed by a subset of source units: the shared expert uses $\mathcal{I}_\ell^{\mathrm{sh}}$ and keeps the dense output bias $\boldsymbol{b}_\ell^{(2)}$, and routed expert e uses $\mathcal{I}_{\ell,e}$ with a zero output bias. Routed slots that receive no source unit hold trainable padding units whose output columns are zero, so padding contributes nothing at initialization; all inherited and padded units are trained during recovery. The shared expert is always active, and a trainable router selects K routed experts per token. For the selected set $\mathcal{T}_{\ell,K}(\boldsymbol{h}, t)$ (Section 3.3), the converted layer computes

$$\widetilde{F}_\ell(\boldsymbol{h}, t) = F_\ell^{\mathrm{sh}}(\boldsymbol{h}) + \sum_{e \in \mathcal{T}_{\ell,K}(\boldsymbol{h},t)} \alpha_{\ell,e}(\boldsymbol{h}, t)\, F_\ell^{(e)}(\boldsymbol{h}), \tag{2}$$

where $\alpha_{\ell,e}$ is the router probability renormalized over the selected experts. The layer stores $S + ER$ expert-unit slots and activates $S + KR$ per token. In the configurations evaluated below, more than half of the routed slots are padding (Appendix A.2).

3.2 TRAJECTORY-CONDITIONED CONTEXTS

Token contexts are formed from the within-timestep variation of the FFN inputs after subtracting the mean of the corresponding original-timestep bucket. We run frozen class-conditional reverse

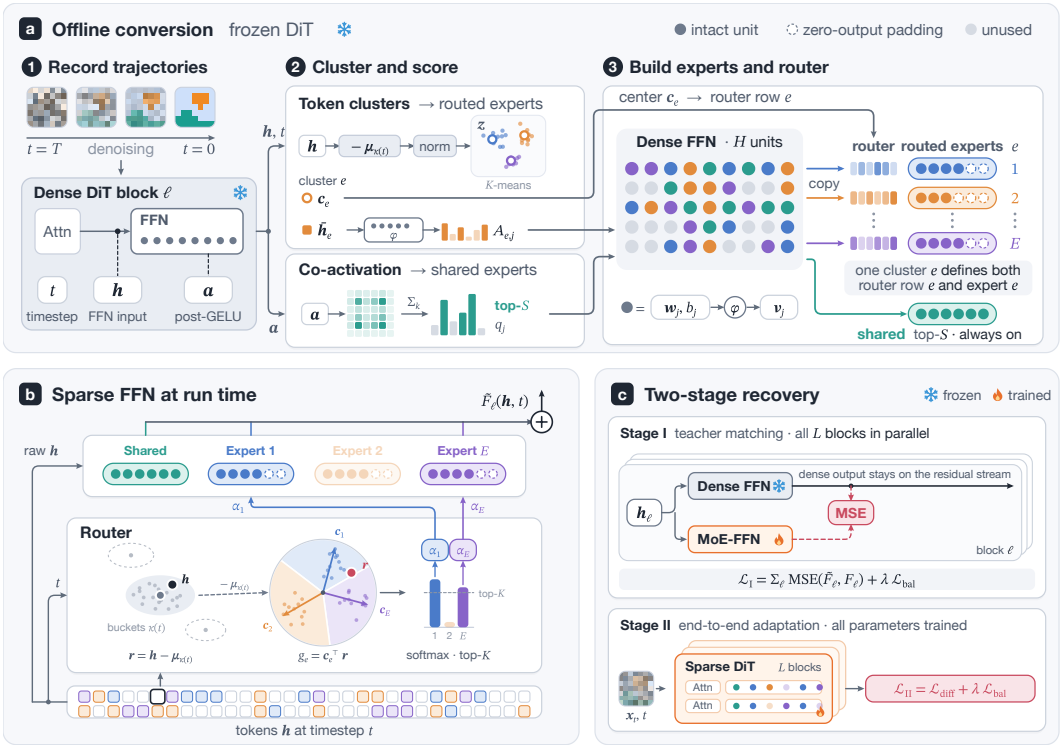

Figure 2: REACT pipeline. Frozen trajectories provide timestep-centered contexts for routed-unit allocation and matching router rows, while pooled co-activation selects shared units (a). At inference, centering is applied only to the router input; experts process the original state, and router rows initialized from $\boldsymbol{c}_e$ remain trainable (b). Stage I matches sparse FFNs to dense teachers before Stage II adapts the full denoiser (c).

trajectories and record, for each block, the AdaLN-modulated FFN input $\boldsymbol{h}_{\ell,i} \in \mathbb{R}^d$, its post-GELU activation $\boldsymbol{a}_{\ell,i} \in \mathbb{R}^H$, and the original diffusion timestep t_i, where i ranges over spatial tokens from both classifier-free-guidance branches. All recorded tokens contribute to the means and activation statistics. We retain one token per sequence and sampling call for the clustering set $\mathcal{S}_\ell$ (Appendix A.4).

Timesteps are grouped into B buckets by $\kappa(t) = \lfloor tB/T_{\text{diff}} \rfloor$, where T_{diff} is the number of original diffusion steps, and $\boldsymbol{\mu}_{\ell,b}$ is the mean FFN input over the recorded tokens of bucket b. Each sampled input is centered by its bucket mean and scaled to unit length,

$$\boldsymbol{z}_{\ell,i} = \operatorname{norm}\big(\boldsymbol{h}_{\ell,i} - \boldsymbol{\mu}_{\ell,\kappa(t_i)}\big), \qquad i \in \mathcal{S}_\ell, \tag{3}$$

where $\operatorname{norm}(\cdot)$ divides a vector by its Euclidean norm (floored at 10^{-12}). Euclidean K-means with E clusters partitions $\{\boldsymbol{z}_{\ell,i}\}$ into index sets $\Omega_{\ell,1}, \ldots, \Omega_{\ell,E}$, and the context direction $\boldsymbol{c}_{\ell,e}$ is the mean of the members of $\Omega_{\ell,e}$ scaled to unit length. Buckets are indexed by the original diffusion-process timestep, not by the respaced sampler-call index. The bucket means are non-trainable buffers that are updated as running averages during both recovery stages and held fixed at evaluation. Appendix A.3 specifies the centering state, and Section 4.5 varies B.

3.3 Cluster-Indexed Expert and Router Initialization

Shared units. Because the shared expert is active at every noise level, its units are selected from pooled, uncentered activation statistics. Unit j receives the score $q_{\ell,j} = \frac{1}{N_{\text{act}}} \sum_i a_{\ell,i,j} \sum_k a_{\ell,i,k}$, the average product of its activation with the total activation of the same token over all N_{act} recorded activation vectors. It is also the j-th column sum of the trajectory-wide activation second moment shown in Figure 2a. The S largest scores define $\mathcal{I}_\ell^{\text{sh}}$.

Routed units. For each cluster, we score a unit by its activation on the cluster's mean uncentered input. Let $\bar{\boldsymbol{h}}_{\ell,e}$ be the mean of the inputs $\{\boldsymbol{h}_{\ell,i} : i \in \Omega_{\ell,e}\}$. The prototype response of unit j to context e is its activation at that mean,

$$A_{\ell,e,j} = \phi\big(\boldsymbol{w}_{\ell,j}^{\top} \bar{\boldsymbol{h}}_{\ell,e} + b_{\ell,j}\big)\,. \tag{4}$$

After excluding shared units, expert-unit pairs are visited in descending order of $A_{\ell,e,j}$. A unit is added to expert e unless the expert is full or the unit has reached its replication limit. A second pass places each still-unassigned unit in its highest-scoring expert with remaining capacity; the unit stays unassigned if no expert has room. The resulting $\mathcal{I}_{\ell,e}$ may share a unit up to the replication limit and never overlap with $\mathcal{I}_{\ell}^{\mathrm{sh}}$; the capacity bounds and traversal order are given in Appendix A.1. Each selected unit is copied intact, and all non-FFN parameters come from the same dense checkpoint.

Same-index router initialization. Router row e is initialized from the same context that built expert e, and the router scores the centered but unnormalized residual of a token:

$$\boldsymbol{w}_{\ell,e}^{\mathrm{rt}}\big|_{\mathrm{init}} = \boldsymbol{c}_{\ell,e}, \qquad g_{\ell,e}(\boldsymbol{h}, t) = (\boldsymbol{w}_{\ell,e}^{\mathrm{rt}})^{\top}\big(\boldsymbol{h} - \boldsymbol{\mu}_{\ell,\kappa(t)}\big). \tag{5}$$

A softmax over the E logits gives routing probabilities, the Top-K experts form $\mathcal{T}_{\ell,K}(\boldsymbol{h}, t)$, and the selected probabilities are renormalized to give $\alpha_{\ell,e}$. Only the router receives the centered residual; every expert processes $\boldsymbol{h}$. The bias-free router remains trainable after initialization. Because the rows have unit norm, the residual magnitude scales all logits of a token equally: it sets the softmax sharpness but not the Top-K order. Initial routing can still differ from the K-means assignment, which uses Euclidean distance to unnormalized centroids, and training may move the router beyond the initial partition. Prior work has partitioned pretrained units into experts and initialized routers from input centroids (Zhang et al., 2022; Zhu et al., 2024; Chu et al., 2026); REACT combines timestep-centered contexts, response-based intact-unit allocation under capacity and replication limits, and a shared expert selected from pooled activation statistics.

3.4 Two-Stage Sparse Recovery

Sparse selection omits some dense units, replicates others, and combines only K routed experts under router weights, so the converted layer does not reproduce F_ℓ exactly. Recovery has two stages: layer-wise matching to the frozen dense FFNs and end-to-end adaptation of the sparse denoiser. Both objectives include the token-choice load-balancing loss $\mathcal{L}_{\mathrm{bal}}$ (Fedus et al., 2022) (Appendix A.4):

$$\begin{aligned} \mathcal{L}_{\mathrm{I}} &= \sum_{\ell=1}^{L} \mathrm{MSE}\Big(\widetilde{F}_\ell(\boldsymbol{h}_\ell, t), F_\ell(\boldsymbol{h}_\ell)\Big) + \lambda_{\mathrm{bal}}^{\mathrm{I}} \mathcal{L}_{\mathrm{bal}}, \\ \mathcal{L}_{\mathrm{II}} &= \mathcal{L}_{\mathrm{diff}} + \lambda_{\mathrm{bal}}^{\mathrm{II}} \mathcal{L}_{\mathrm{bal}}, \end{aligned} \tag{6}$$

where $\boldsymbol{h}_\ell$ is the AdaLN-modulated input to FFN ℓ, $\mathcal{L}_{\mathrm{diff}}$ is the source DiT diffusion objective, and the λ_{bal} coefficients weight the auxiliary loss in each stage.

Stage I: parallel layer-wise teacher matching. The original frozen FFN is attached to every converted block, all non-MoE parameters are frozen, and the router, shared expert, and routed experts are trained with $\mathcal{L}_{\mathrm{I}}$. Dense outputs stay on the residual stream, so every sparse FFN receives the teacher's hidden states and all layers are matched in parallel.

Stage II: end-to-end adaptation. The teachers are removed, sparse outputs are placed on the residual stream, and every normally trainable parameter is optimized with $\mathcal{L}_{\mathrm{II}}$. Appendices A.1–A.4 report the calibration, allocation, centering, optimizer, EMA, and loss settings used in both stages.

4 Experiments

We first compare generation quality, active parameter counts, and measured latency. We then examine recovery under controlled initialization changes, inspect routing after recovery, and vary centering granularity and Stage-I duration.

4.1 Evaluation Protocol and Comparison Design

Task and metrics. We evaluate class-conditional ImageNet-1K generation at 256×256 resolution (Deng et al., 2009). For each recovery checkpoint, 50,000 samples are used to compute FID and spatial FID (sFID), where lower is better, together with Inception Score (IS), precision, and recall, where higher is better (Heusel et al., 2017; Kynkäänniemi et al., 2019). All primary recovery comparisons use the same five Stage-II checkpoints from 100K to 500K end-to-end adaptation steps. Stage-I reconstruction is completed before the first checkpoint. Except in the explicitly labeled Stage-I duration study, the controlled comparisons match both Stage-I duration and Stage-II checkpoint across conditions.

Initialization conditions. XL and M denote conversions initialized from dense DiT-XL and DiT-M checkpoints; these source-scale labels differ from the expert-width suffixes used below. The full initialization combines shared and routed experts with their matched centroid router. In the M-source ablations, *Experts + random router* changes only the router, while *Centroid router + random experts* replaces the trajectory-based expert construction. *Bundled random* also removes timestep centering and randomizes shared-unit selection and routed-unit allocation. The M-source study includes all four conditions; the XL-source study compares REACT with bundled random under the same checkpoint protocol.

4.2 Quality, Activation, and Measured Latency

Table 1 reports parameter counts and generation metrics for three Top-2 variants converted from DiT-XL/2. REACT-M32 has $4.125\times$ the dense model's total FFN parameters but activates only $0.375\times$ its FFN parameters per token. Its model-level counts are 1.61B total and 489M active non-router parameters, compared with 675M total and active parameters for DiT-XL/2. REACT-M48 raises total FFN capacity to $6.125\times$ with the same token-wise activation. REACT-L32 uses $5.5\times$ total and $0.5\times$ active FFN parameters, or 2.02B total and 526M active non-router model parameters. The three variants report FIDs of 2.21, 2.22, and 2.18, respectively; the published DiT-XL/2 reference is 2.27. Because each nonshared unit is copied at most twice, at least 56.25% of the routed slots in REACT-M32 are zero-output padding at initialization and are trained during recovery (Appendix A.2).

Table 1: Parameter counts and ImageNet-256 generation metrics for DiT-XL/2 and its REACT conversions. FFN Size is the dense hidden width or the common shared and routed expert width. FFN multipliers are relative to dense DiT-XL/2; Total Param includes router weights, while Active Param counts token-wise active non-router weights. The dense quality result follows Peebles & Xie (2023) with classifier-free guidance 1.5; sparse results use 50K samples.

(a) Parameter capacity and token-wise activation

				FFN Params vs. Dense		Model Params	
Model	E	K	FFN Size	Total Param	Active Param	Total Param	Active Param
DiT-XL/2	–	–	4608	1.000×	1.000×	675M	675M
REACT-M32	32	2	576	4.125×	0.375×	1606M	489M
REACT-M48	48	2	576	6.125×	0.375×	2202M	489M
REACT-L32	32	2	768	5.500×	0.500×	2015M	526M

(b) ImageNet-256 generation quality

Model	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
DiT-XL/2	2.27	**4.60**	278.24	**0.83**	0.57
REACT-M32	2.21	4.75	276.53	0.82	0.59
REACT-M48	2.22	4.70	279.15	0.82	**0.60**
REACT-L32	**2.18**	4.73	**286.85**	0.82	0.59

In the shared profiling setup, both sparse variants reduce latency at every measured scope (Table 2). REACT-M32 provides $2.32\times$ FFN and $1.45\times$ full-forward speedups; end-to-end sampling falls from 26.81 to 18.19 seconds, a $1.47\times$ speedup. REACT-L32 achieves speedups of $1.74\times$, $1.33\times$, and $1.34\times$. The smaller end-to-end gains reflect non-FFN computation and routing overhead.

Table 2: Latency in the shared profiling setup; parentheses report speedup over Dense.

Model	FFN (ms)↓	Forward (ms)↓	Sampling (s)↓
Dense	2.117 (1.00×)	58.79 (1.00×)	26.81 (1.00×)
REACT-M32	**0.912 (2.32×)**	**40.41 (1.45×)**	**18.19 (1.47×)**
REACT-L32	1.217 (1.74×)	44.16 (1.33×)	20.04 (1.34×)

4.3 Controlled Recovery and Initialization Contrasts

At every observed checkpoint, REACT has lower FID than bundled random in both source-scale studies (Figure 3). At 500K, the M-source values are 8.60 and 10.26, and the XL-source values are 2.54 and 2.78. The two partial M-source controls compare expert-side and router initialization bundles over the same Stage-II interval.

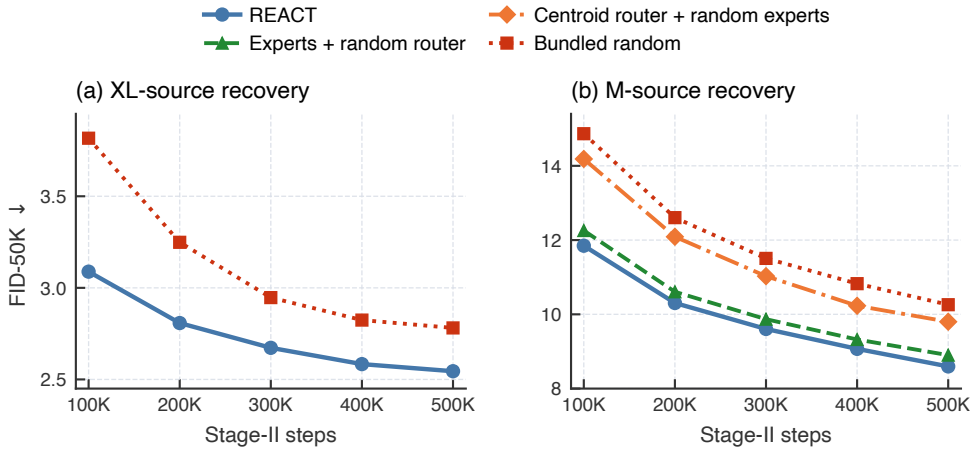

Figure 3: Single-run FID-50K trajectories during Stage-II adaptation. The M-source panel includes both partial controls; panels use independent vertical scales.

Table 3: FID-50K at every Stage-II checkpoint for the XL- and M-source initialization studies.

	XL source		M source			
Stage-II step	REACT	Random	REACT	Experts	Router	Random
100K	3.09	3.82	11.85	12.26	14.18	14.87
200K	2.81	3.25	10.31	10.61	12.09	12.60
300K	2.67	2.95	9.61	9.87	11.03	11.51
400K	2.58	2.82	9.07	9.32	10.23	10.83
500K	2.54	2.78	8.60	8.90	9.80	10.26

Table 4 summarizes all metrics at 500K. REACT has the lowest FID and sFID and the highest IS and precision at both source scales. Recall differs by at most 0.01 among the M-source conditions; in the XL study, it is 0.58 for REACT and 0.57 for bundled random.

Control contrasts. REACT and experts + random router share expert contents and timestep centering, so this pair compares centroid-based and random router initialization with the expert assembly fixed. The centroid router yields lower FID point estimates at all five checkpoints, with gaps of 0.25–0.41; at 500K, the endpoints are 8.60 versus 8.90 FID and 132.43 versus 130.31 IS.

The other controls change multiple components, but their ordering is stable at every checkpoint: REACT, experts + random router, centroid router + random experts, and bundled random. Relative to bundled random, experts + random router is 1.36–2.61 FID lower and the centroid-router control is 0.46–0.69 lower; the latter remains 1.16–2.33 above REACT (Table 3).

Table 4: Generation metrics at the 500K Stage-II checkpoint.

Source	Condition	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
M	REACT	8.60	6.01	132.43	0.74	0.55
M	Experts + random router	8.90	6.02	130.31	0.73	0.55
M	Centroid router + random experts	9.80	6.31	124.83	0.72	0.56
M	Bundled random	10.26	6.43	122.57	0.72	0.55
XL	REACT	2.54	4.82	247.14	0.82	0.58
XL	Bundled random	2.78	4.98	235.53	0.81	0.57

4.4 Routing Coherence and Expert Utilization

Recovered routing remains coherent without collapsing onto a small expert subset. For a stored REACT-M32 (XL/2-source) checkpoint, S_{ij} is the mean dot product between normalized router-input representations assigned to experts i and j, excluding identical self-pairs. Among the eight analyzed experts, every diagonal entry is largest in its row; the diagonal mean is 0.571 versus 0.460 off diagonal. All 32 experts receive traffic, with shares of 1.477%–4.333% and a coefficient of variation of 0.161, so the observed coherence does not arise from routing collapse (Figure 4).

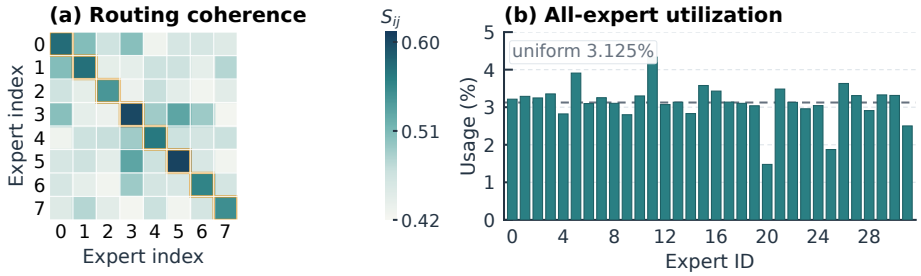

Figure 4: Routing diagnostics for recovered REACT-M32: (a) S_{ij} for the eight most frequently activated experts among all 32, locally indexed 0–7 (scale 0.42–0.61); (b) traffic across all 32 experts, with the uniform 3.125% share as reference.

4.5 Sensitivity to Centering and Stage-I Duration

Table 5 follows FID across matched Stage-II checkpoints as centering granularity and Stage-I duration change. Table 6 reports all five metrics at the common 400K checkpoint.

Table 5: M-source sensitivity (FID-50K↓) at matched Stage-II checkpoints. The reference uses 250 centering buckets and 200K Stage-I steps; each other row changes one factor.

		FID-50K↓ at Stage-II checkpoint			
Factor	Setting	100K	200K	300K	400K
Reference	250 buckets; Stage I = 200K	11.85	10.31	9.61	9.07
Centering	50 buckets	11.88	10.32	9.54	9.05
Stage-I duration	100K	12.70	10.88	10.06	9.43
	300K	11.56	10.11	9.41	8.88

The bucket-count variant crosses the reference trajectory rather than remaining consistently better or worse: its FID differences are $+0.03$, $+0.01$, -0.07, and -0.02 from 100K to 400K. Stage-I duration produces a stable ordering. The 300K run remains 0.19–0.29 FID below the 200K reference, whereas the 100K run stays 0.36–0.85 above it at every checkpoint.

Table 6: Generation metrics at the common 400K M-source checkpoint.

Configuration	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
REACT (250 buckets; Stage I = 200K)	9.07	6.05	128.93	0.73	0.55
Experts + random router	9.32	6.11	127.62	0.73	0.55
Centroid router + random experts	10.23	6.40	121.41	0.72	0.55
Bundled random	10.83	6.55	118.47	0.71	0.55
TC-50 (50 centering buckets)	9.05	6.06	129.39	0.73	0.55
Stage I = 100K	9.43	6.12	125.74	0.73	0.55
Stage I = 300K	8.88	6.05	129.91	0.74	0.54

At 400K, randomizing only the router raises FID by 0.25 and lowers IS by 1.31. Randomizing the experts widens the FID and IS gaps to 1.16 and 7.52, while bundled random widens them to 1.76 and 10.46. Precision varies by at most 0.02 and recall remains 0.55, so the clearest separation appears in FID, sFID, and IS. These single-run point estimates follow the trajectory ordering in Table 3.

5 Conclusion

REACT converts a dense DiT into sparse experts by using timestep-bucket-centered trajectory contexts to assemble intact source units and to initialize the corresponding router rows. Across the DiT-M and DiT-XL source checkpoints, it has lower FID than bundled random initialization at every evaluated Stage-II checkpoint, and the matched M-source control favors centroid-based over random router initialization. In the separate DiT-XL/2 experiment, REACT-M32 reports 2.21 FID with 489M active non-router parameters per token; the published dense reference is 2.27 FID with 675M parameters. It achieves a $1.47\times$ sampling speedup in the shared profiling setup. In the analyzed router, all 32 experts receive traffic, and the eight analyzed experts show higher within-expert than cross-expert coherence. Across the tested settings, the full initialization improves recovery over the bundled-random baseline at the evaluated Stage-II checkpoints.

AI use statement

We used Claude (Anthropic), accessed through Claude Code, as a writing and presentation assistant. Its assistance covered three areas: (i) the design of the method overview (Figure 2) and the motivation figure (Figure 1), including the panel layout, visual encoding, and vector rendering from author-specified content; (ii) the typesetting of the manuscript, including figure and table placement, moving supporting tables between the appendix and the main text, and balancing section lengths; and (iii) language polishing of the full text, including sentence-level editing of the introduction, related work, method, and captions, and tightening the presentation of the equations. All methods, experiments, measurements, and claims were designed, executed, and verified by the authors; the assistant generated no results, data, or references. The authors reviewed every AI-assisted edit and take full responsibility for the content of this paper.

Ethics statement

This work evaluates a conversion method for class-conditional image generation using ImageNet-1K and pretrained DiT checkpoints. It does not involve human participants or collect new data. The converted models may inherit bias from their source models and data or be misused to create misleading content. Sparse inference reduces active computation, but conversion and recovery training add compute, and the models have greater total capacity. Deployment should document the source model and data, evaluate bias and content safety for the intended setting, and apply appropriate safeguards.

Reproducibility statement

The main text reports the evaluation convention and quantitative comparisons. Appendix A gives conversion, accounting, implementation, and sensitivity details; Appendix D provides additional

samples; Appendix B discusses scope and future directions; and Appendix C extends the comparison across sparse diffusion and dense-to-MoE upcycling.

REFERENCES

Shuning Chang, Pichao Wang, Jiasheng Tang, Fan Wang, and Yi Yang. SparseDiT: Token sparsification for efficient diffusion transformer. *Advances in Neural Information Processing Systems*, 38:91456–91483, 2026.

Kun Cheng, Xiao He, Lei Yu, Zhijun Tu, Mingrui Zhu, Nannan Wang, Xinbo Gao, and Jie Hu. Diff-MoE: Diffusion transformer with time-aware and space-adaptive experts. In *Forty-second International Conference on Machine Learning*, 2025.

Sanghyeok Chu, Pyunghwan Ahn, Gwangmo Song, Seung Hwan Kim, Honglak Lee, and Bohyung Han. Enhancing mixture-of-experts specialization via cluster-aware upcycling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11283–11292, 2026.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

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

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Remix-DiT: Mixing diffusion transformers for multi-expert denoising. *Advances in Neural Information Processing Systems*, 37:107494–107512, 2024.

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.

Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*, 2024.

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.

Wangyi Jiang, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. Improved sparse upcycling for instruction tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 9485–9498, 2025.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

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.

Yunsung Lee, JinYoung Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13427–13436, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization. In *International Conference on Learning Representations*, volume 2025, pp. 84370–84397, 2025.

Byeongjun Park, Hyojun Go, Jin-Young Kim, Sangmin Woo, Seokil Ham, and Changick Kim. Switch diffusion transformer: Synergizing denoising tasks with sparse mixture-of-experts. In *European Conference on Computer Vision*, pp. 461–477. Springer, 2024.

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

Zehua Pei, Hui-Ling Zhen, Lancheng Zou, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. Analytical FFN-to-MoE restructuring via activation pattern analysis. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4777–4789, 2026.

Vikash Sehwag, Xianghao Kong, Jingtao Li, Michael Spranger, and Lingjuan Lyu. Stretching each dollar: Diffusion training from scratch on a micro-budget. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28596–28608. IEEE, 2025.

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.

Haotian Sun, Tao Lei, Bowen Zhang, Yanghao Li, Haoshuo Huang, Ruoming Pang, Bo Dai, and Nan Du. EC-DiT: Scaling diffusion transformers with adaptive expert-choice routing. In *International Conference on Learning Representations*, volume 2025, pp. 72383–72401, 2025.

Yujie Wei, Shiwei Zhang, Hangjie Yuan, Yujin Han, Zhekai Chen, Jiayu Wang, Difan Zou, Xihui Liu, Yu Liu, Hongming Shan, et al. Routing matters in MoE: Scaling diffusion transformers with explicit routing guidance. In *International Conference on Learning Representations*, volume 2026, pp. 23381–23405, 2026.

Yike Yuan, Ziyu Wang, Zihao Huang, Defa Zhu, Xun Zhou, Jingyi Yu, and Qiyang Min. Expert race: A flexible routing strategy for scaling diffusion transformer with mixture of experts. *arXiv preprint arXiv:2503.16057*, 2025.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, 2022.

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*, volume 2025, pp. 65520–65552, 2025.

Chenyu Zheng, Xinyu Zhang, Rongzhen Wang, Wei Huang, Zhi Tian, Weilin Huang, Jun Zhu, and Chongxuan Li. Scaling diffusion transformers efficiently via μP. *Advances in Neural Information Processing Systems*, 38:19372–19413, 2026.

Youwei Zheng, Yuxi Ren, Xin Xia, Xuefeng Xiao, and Xiaohua Xie. Dense2MoE: Restructuring diffusion transformer to MoE for efficient text-to-image generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18661–18670. IEEE, 2025.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 15913–15923, 2024.

A Conversion and Reproducibility Details

A.1 Offline Conversion Procedure

Conversion is applied independently to each dense DiT block:

1. Run the frozen DiT on class-conditional reverse trajectories. Record post-GELU co-activations and FFN-input sums and counts over all spatial tokens. Retain sampled token inputs and their original diffusion timesteps; these timesteps select the centering bucket during offline conversion, both recovery stages, and inference.

2. Map each original timestep to $\kappa(t) = \lfloor tB/T_{\text{diff}} \rfloor$. Reject the calibration if any bucket is unobserved. Store sample timesteps, bucket identifiers, mean counts, and mapping metadata in version-2 artifacts, ensuring a common centering convention in all downstream stages.

3. Subtract the matching bucket mean from every sampled token, normalize the result, and run Euclidean layer-wise K-means. Retain the cluster assignments and raw centers; normalize each center only when it is copied into the router.

4. Sum each column of the co-activation matrix and select the top S dense neurons to populate the always-active shared expert with intact source units.

5. Evaluate the remaining dense neurons on the raw mean input of every token cluster. With $u = (H - S)/E$, the first allocation pass targets $m_{\min} = \lceil k_{\min} u \rceil$ neurons per expert, caps each expert at $m_{\max} = \lfloor k_{\max} u \rfloor$, and limits each neuron to $r_{\max}$ copies. It scans all expert-neuron pairs in descending response order and stops when every expert reaches $m_{\min}$ or the list is exhausted. A second pass visits never-assigned nonshared neurons in source index order and places each in its highest-response expert with room under the hard cap $\min\{\lfloor (k_{\max} + 1)u \rfloor, R\}$. A neuron remains unassigned if no expert has room. Exact ties follow the tensor library's returned order. All reported configurations satisfy $m_{\max} \leq R$.

6. Slice the dense FFN input rows, input biases, and output columns using the selected indices. Copy the dense output bias only to the shared expert; zero routed output biases and the output columns of padded units so padding contributes nothing at initialization.

7. Copy the normalized cluster centers to same-index router rows. This aligns cluster, expert, and router-row identities; runtime assignments then follow the bias-free router's dot-product decision rule before the router is updated during sparse recovery.

Conversion is offline and requires no gradient-based model optimization. Stage I reconstructs the MoE FFNs against frozen dense teachers; Stage II removes the teachers and adapts the converted denoiser end to end under the diffusion objective.

A.2 Parameter and Activation Accounting

Ignoring biases for exposition, a dense FFN contains $2dH$ parameters, all of which are active for each token. The converted FFN contains $2d(S + ER)$ expert parameters plus a dE router, while a token activates $2d(S + KR)$ expert parameters. Whole-model totals also include biases, attention and conditioning modules, embeddings, and the output head. Expert parallelism changes parameter placement but not the global parameter count.

Total Param in Table 1 includes the router, whereas Active Param follows the token-wise accounting convention and omits the router. Including active router weights would add approximately 1.0M parameters to REACT-L32 and REACT-M32 and 1.5M to REACT-M48.

All three configurations use $d = 1152$, $H = 4608$, $L = 28$, and $K = 2$. REACT-M32 sets $E = 32$ and $S = R = 576$, giving $4.125\times$ total and $0.375\times$ active FFN parameters relative to dense DiT-XL/2. The complete model has 1.61B total and 489M active parameters per token. Increasing the expert width to $S = R = 768$ gives REACT-L32 $5.5\times$ total and $0.5\times$ active FFN parameters, or 2.02B total and 526M active model parameters. REACT-M48 keeps $S = R = 576$ and raises E to 48, for $6.125\times$ total and $0.375\times$ active FFN parameters, approximately 2.20B total parameters, and 489M active parameters. These counts describe structural sparsity, not wall-clock efficiency, which also depends on kernels, dispatch, communication, batching, and hardware.

If a routed expert inherits fewer than R neurons, its remaining slots are trainable but have zero output at initialization. The limit of $r_{\max}$ copies per nonshared neuron gives the following lower bound on the routed-slot padding fraction at initialization:

$$\rho_{\text{pad}} \geq \max\left\{0, 1 - \frac{r_{\max}(H - S)}{ER}\right\}. \tag{7}$$

For the REACT-M32 values $H = 4608$, $S = R = 576$, $E = 32$, and $r_{\max} = 2$, at least 56.25% of routed slots are padded and therefore count as newly added trainable capacity.

A.3 Centering-State Consistency

The centering state for each layer contains the mean matrix, observation counts, format version, number of original diffusion steps, and number of buckets. Sample files additionally contain one original timestep and bucket identifier per sampled token. Profiling, K-means, PyTorch inference, expert-parallel inference, model checkpoints, and EMA checkpoints validate the same metadata. Checkpoints that enable centering in only the model or only the EMA weights are rejected.

Version-1 artifacts indexed means by reverse sampling call order. For a 250-step reverse process, this can associate the first stored row with a high original timestep while runtime indexing expects a low timestep. Because online training updates can mix correctly and incorrectly addressed rows, a post-hoc row reversal is not generally valid. We therefore exclude version-1 results and use only calibration, clustering, mapping, and trained checkpoints produced under the version-2 convention.

A.4 Reference Implementation Configuration

The calibration procedure uses one reverse trajectory per ImageNet class, 250 sampling calls mapped to a 1000-step original diffusion process, 250 centering buckets, classifier-free guidance scale 1.5, and profiling seed 0. Both the conditional and unconditional branches enter the activation statistics. All spatial tokens contribute to the co-activation matrix, while one token per sequence and denoising call is retained for clustering. Layer-wise K-means uses scikit-learn's Euclidean implementation with E clusters and seed 0; initialization count, stopping tolerance, iteration limit, and solver use the installed library defaults. Routed-neuron allocation uses $k_{\min} = 0.6$, $k_{\max} = 2.0$, and $r_{\max} = 2$; padded first-layer parameters have standard deviation 10^{-2}.

The reference optimizer is AdamW with constant learning rate 10^{-4}, zero weight decay, and the PyTorch default $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 10^{-8})$; there is no learning-rate warmup or scheduler. The reference trainer defaults to global batch size 256 and seed 0. It uses the `sd-vae-ft-ema` encoder, center cropping, random horizontal flipping, uniform sampling over the 1000 training timesteps, FP32 tensors with CUDA TF32 matrix operations enabled, and no explicit autocast. Load-balancing coefficients are 10^{-2} in Stage I and 10^{-4} in Stage II, and EMA decay is 0.9999. Token-choice routing has no capacity factor or token dropping in the reference PyTorch implementation.

The auxiliary loss in both stages is computed from local-worker statistics; data-parallel training averages the resulting gradients. With G workers, it can be written as

$$\mathcal{L}_{\text{bal}} = \frac{1}{G} \sum_{g=1}^{G} \sum_{\ell=1}^{L} E \sum_{e=1}^{E} f_{\ell,e}^{(g)} \, \bar{p}_{\ell,e}^{(g)}, \tag{8}$$

where $f_{\ell,e}^{(g)}$ is the local-minibatch fraction of tokens whose top-K set contains expert e, and $\bar{p}_{\ell,e}^{(g)}$ is its mean pre-selection routing probability over all local tokens.

For DiT-XL/2, the 28 full 4608×4608 FP32 co-activation matrices contain about 595 million entries and occupy 2.38 GB before temporary buffers. Conversion also incurs collection, distributed aggregation, host transfer, clustering, and allocation before either optimization stage.

B Scope and Future Directions

Our experiments cover class-conditional ImageNet DiTs. Applying the conversion to multimodal text conditioning, video diffusion, and non-diffusion transformers remains open. The method trades

a lower token-wise active FFN count for greater total capacity, so deployment gains depend on batching, kernels, dispatch, communication, and hardware.

Offline conversion adds calibration and clustering costs, including an $H \times H$ co-activation matrix. The current design couples shared-neuron selection, routed-neuron allocation, replication, padding, and timestep centering. Factorized controls and cross-sampler tests are needed to separate these effects and assess transfer across settings and model families.

C Additional Related Work

Broader diffusion-expert designs. MEME and Remix-DiT form timestep specialists. EC-DiT, Diff-MoE, and Race-DiT study expert-choice, time- and space-adaptive, and competitive routing (Lee et al., 2024; Fang et al., 2024; Sun et al., 2025; Cheng et al., 2025; Yuan et al., 2025). MicroDiT combines MoE layers with deferred token masking for low-budget training from scratch (Sehwag et al., 2025). Unlike REACT, these methods do not initialize expert contents by recovering intact FFN units from a dense checkpoint during initialization.

Efficient DiT computation and scaling. DyDiT varies width and token computation across timesteps and spatial regions. SparseDiT changes token density by layer and denoising stage, while μP transfers optimization hyperparameters across DiT scales (Zhao et al., 2025; Chang et al., 2026; Zheng et al., 2026). These methods modify execution or scaling without converting a dense FFN into experts. REACT instead reuses a dense checkpoint by constructing experts from trajectory contexts and initializing router rows from the same partition.

D ADDITIONAL QUALITATIVE SAMPLES

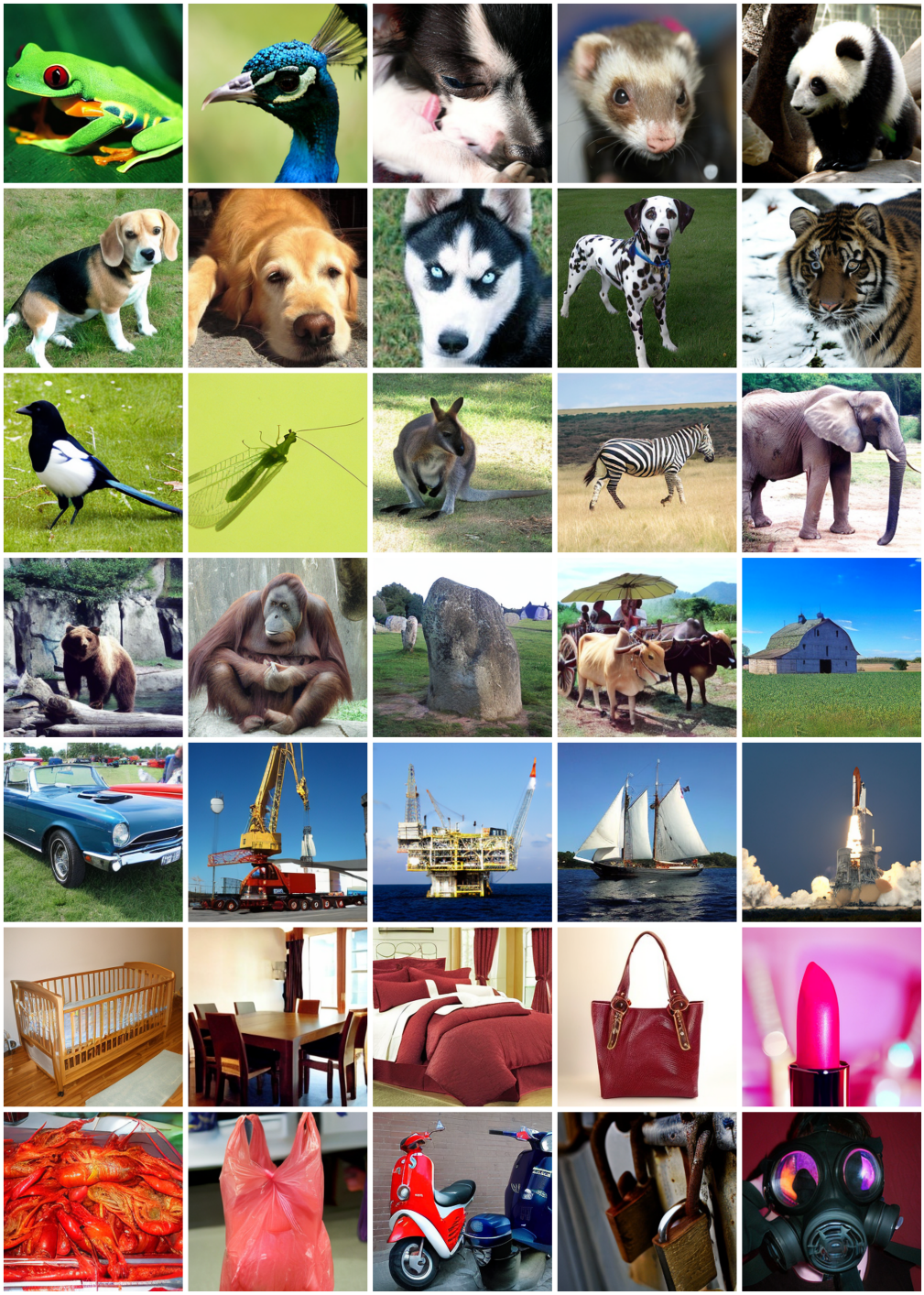

Figure 5: Additional REACT samples grouped by semantic content: animals, natural subjects, outdoor scenes, vehicles, and manufactured objects.