ITC-MoE: Importance-guided Token-aware Compression for MoE Diffusion Language Models

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) Diffusion Language Models (DLMs) offer flexible parallel decoding and increased model capacity, but their large number of expert parameters incurs substantial computation and storage costs. Existing low-rank MoE compression methods largely rely on static factorization and fixed rank allocation, which overlook the distinctive properties of MoE DLMs. Specifically, we identify two properties: *cross-mode non-uniform redundancy*, where parameter redundancy and sensitivity to rank truncation vary across the input, output, and expert modes, and *token-wise utilization variation*, where hot and cold tokens exhibit distinct spectral characteristics and expert activation patterns. To address these challenges, we propose **ITC-MoE**, an Importance-guided Token-aware Compression framework for MoE DLMs. ITC-MoE consists of two complementary components. First, **Importance-guided Adaptive Tucker Compression (IATC)** incorporates activation and gradient importance into expert weight transformation, jointly factorizes expert weights across multiple modes, and adaptively allocates ranks under a fixed parameter budget. Second, **Token-aware Compensation and Routing (TCR)** applies lightweight low-rank compensation to compression-sensitive hot tokens and restricts the candidate expert set for cold tokens with concentrated routing patterns. By jointly adapting compression capacity and inference execution to both parameter redundancy and token-wise variation, ITC-MoE substantially reduces the computation and storage costs of MoE DLMs while preserving their generation quality. For example, on SDAR-30B-A3B-Chat-b32, ITC-MoE maintains an accuracy of 96.33% on MultiArith under a 30% compression budget, while achieving up to a $7.22\times$ end-to-end speedup.

1 Introduction

Mixture-of-Experts (MoE)-based Diffusion Language Models (DLMs) have recently emerged as a promising alternative to autoregressive language models (Li et al., 2022; Sahoo et al., 2024). By generating sequences through multiple denoising iterations, MoE DLMs increase model capacity with multiple experts and selectively activate a subset of experts for each token (Zhu et al., 2026). Unlike autoregressive models, DLMs can predict tokens at multiple positions in parallel within each iteration, enabling flexible parallel decoding and strong performance across various tasks (Nie et al., 2025; Wang et al., 2026).

However, the large number of expert parameters in MoE models introduces substantial computation and storage overhead, becoming a critical bottleneck for practical DLM inference (Wei et al., 2026a). Low-rank compression provides a natural approach to reducing the computation and storage costs of MoE models. Existing methods (Li et al., 2025; Gu et al., 2025; Chen et al., 2026; Mi et al., 2026) have demonstrated the effectiveness of low-rank factorization for conventional autoregressive MoE models. However, their compression strategies are not well suited to the distinctive characteristics of MoE DLMs. As shown in Fig. 2, we identify two properties of MoE DLMs: *cross-mode non-uniform redundancy* and *token-wise utilization variation*.

First, as shown in Fig. 2(a)-(e), the redundancy of MoE parameters is inherently non-uniform across modes. The input and output feature dimensions exhibit strong low-rank structures, while the ex-

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.

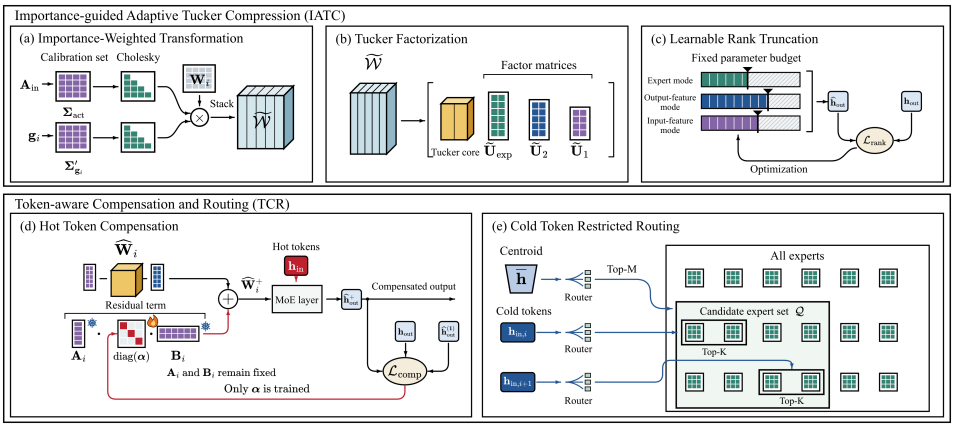


Figure 1: Overview of the ITC-MoE framework.

pert dimension also contains substantial cross-expert redundancy, and different modes show distinct sensitivities to rank truncation. Second, during diffusion decoding, MoE DLMs exhibit pronounced token-wise utilization variation. Hot and cold tokens differ in their feature spectral distributions and expert utilization patterns, with hot tokens being more sensitive to rank truncation and cold tokens exhibiting more concentrated expert activation. However, existing low-rank MoE compression methods (Li et al., 2025; Gu et al., 2025; Chen et al., 2026; Mi et al., 2026; Wei et al., 2026a) largely rely on static compression schemes, such as independent expert factorization or pre-determined rank and routing configurations. Such approaches neither explicitly exploit the non-uniform redundancy across the input, output, and expert dimensions nor adapt the compressed model to token-wise variation during diffusion decoding (Huang et al., 2026; Jazbec et al., 2026), leaving opportunities for further reducing computation and storage of MoE DLMs while preserving model capacity.

In this paper, we propose **ITC-MoE**, an Importance-guided Token-aware Compression framework for MoE DLMs, as shown in Fig. 1. ITC-MoE jointly exploits non-uniform parameter redundancy and token-wise utilization variation. At first, ITC-MoE introduces Importance-guided Adaptive Tucker Compression (IATC) to exploit cross-mode non-uniform redundancy. IATC incorporates activation and gradient importance into expert weight transformation, jointly factorizes the transformed expert weights with Tucker decomposition, and adaptively allocates ranks across different modes under a fixed parameter budget. ITC-MoE then introduces Token-aware Compensation and Routing (TCR) to adapt the compressed experts to token-wise variation. TCR applies lightweight low-rank compensation to hot tokens, which are more sensitive to compression errors, while restricting the candidate expert set for cold tokens, whose routing decisions are more concentrated.

By jointly integrating IATC and TCR, ITC-MoE enables more efficient MoE DLM inference while maintaining model fidelity. For example, on SDAR-30B-A3B-Chat-b32 Cheng et al. (2026), under a 30% compression budget, ITC-MoE still achieves an accuracy of 96.33 on MultiArith while delivering up to a 7.22× end-to-end inference speedup.

2 RELATED WORK

Mixture-of-Experts for Diffusion Models. The iterative nature of diffusion models naturally aligns with MoE routing, as the computational demands vary across timesteps and token positions, enabling dynamic expert assignment. Switch-DiT (Park et al., 2024) first introduces sparse MoE into Diffusion Transformers, learning timestep-dependent expert paths. EC-DiT (Sun et al., 2025) adopts expert-choice routing to adaptively allocate computation according to heterogeneous text and image-patch complexity. Diff-MoE (Cheng et al., 2025) introduces time-aware and space-adaptive experts, conditioning expert assignment jointly on timestep and spatial token features. Race-DiT (Yuan et al., 2025) formulates routing as a competition between tokens and experts, dynamically assigning additional expert capacity to important tokens. Complementary to these routing-centric approaches, Dense2MoE (Zheng et al., 2025) converts pretrained dense diffusion transformers into sparse MoE architectures with block-level conditional execution. ProMoE (Wei et al., 2026b) argues that conventional language-model routing does not directly transfer to visual diffusion models, and proposes

explicitly guided conditional and prototype-based routing to promote expert specialization. Beyond architectural innovations, MoE has also been incorporated into compute-efficient training pipelines to scale model capacity under constrained budgets (Sehwag et al., 2025).

Efficient Mixture-of-Experts. MoE models dynamically route each token to a small subset of experts, which incurs considerable computational overhead. Many efficient research methods have been developed to improve routing strategies and expert utilization Dai et al. (2024); Lewis et al. (2021); Zhou et al. (2022); Komatsuzaki et al. (2023). Soft MoE (Puigcerver et al., 2024) replaces discrete routing with differentiable soft assignment, improving training stability and expert utilization. Another line of work reduces the memory and computational cost of experts by employing quantization Xu et al. (2026); Chen et al. (2025); Duanmu et al. (2025). Expert-level pruning methods identify and remove redundant experts (Jie et al., 2025; Huang et al., 2025b; Bai et al., 2025). FloE (Zhou et al., 2025b) combines compression with on-demand parameter loading for memory-constrained inference, and Oracle-MoE (Zhou et al., 2025a) improves routing locality to reduce expert swapping. Recent methods explicitly target low-rank redundancy within experts. MoBE (Chen et al., 2026) represents experts using shared basis matrices. MoNE (Zhang et al., 2026) replaces redundant experts with lightweight novice modules. TD-MoE (XU et al., 2026) performs Tucker decomposition to capture cross-expert structure. MiLo (Huang et al., 2025a) combines aggressive expert quantization with low-rank compensators. MoE-SVD (Li et al., 2025) shares decomposed components across experts to reduce both parameters and latency. However, these low-rank methods target autoregressive MoE LLMs and have not been adapted for MoE DLMs, which use bidirectional attention. As a recent effort, TEAM (Wei et al., 2026a) addresses this gap by exploiting spatiotemporal consistency in MoE DLM decoding to adaptively activate experts according to token states, thereby reducing redundant computation. However, TEAM only reduces inference overhead, without fully exploiting the structural redundancy in experts to compress the model footprint.

3 METHOD

3.1 PRELIMINARIES

Mixture-of-Experts. A mixture-of-experts (MoE) layer replaces a dense feed-forward network with E experts and a router (Shazeer et al., 2017). For a input $\mathbf{h}_{\text{in}} \in \mathbb{R}^{d_{\text{in}}}$, the router computes $\boldsymbol{\pi}(\mathbf{h}_{\text{in}}) = \text{softmax}(\mathbf{W}^{\boldsymbol{\pi}}\mathbf{h}_{\text{in}})$, where $\mathbf{W}^{\boldsymbol{\pi}} \in \mathbb{R}^{E\times d_{\text{in}}}$ is the router weight and $\pi_i(\mathbf{h}_{\text{in}})$ is the routing probability for expert $\mathbf{E}_i$. The selected expert set is $\mathcal{E} = \text{Top}_K(\boldsymbol{\pi}(\mathbf{h}_{\text{in}}))$, where $\text{Top}_K(\cdot)$ returns the indices of the K highest-scoring experts. For a gated feed-forward expert, the output of expert $\mathbf{E}_i$ for the input $\mathbf{h}_{\text{in}}$ is:

$$\mathbf{o}_i = \mathbf{W}_i^{\text{down}}\left[\phi(\mathbf{W}_i^{\text{gate}}\mathbf{h}_{\text{in}}) \odot (\mathbf{W}_i^{\text{up}}\mathbf{h}_{\text{in}})\right], \tag{1}$$

where $\mathbf{W}_i^{\text{down}}, \mathbf{W}_i^{\text{gate}}, \mathbf{W}_i^{\text{up}}$ are the weights of $\mathbf{E}_i$, $\phi(\cdot)$ is an element-wise activation, and $\odot$ denotes element-wise multiplication. The MoE output is:

$$\mathbf{h}_{\text{out}} = \sum_{i\in\mathcal{E}} \widehat{\pi}_i\, \mathbf{o}_i, \qquad \widehat{\pi}_i = \frac{\pi_i(\mathbf{h}_{\text{in}})}{\sum_{j\in\mathcal{E}} \pi_j(\mathbf{h}_{\text{in}})}, \tag{2}$$

where $\widehat{\pi}_i$ is the routing weight of the ith expert after renormalization over $\mathcal{E}$ and $\mathbf{h}_{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$ is the aggregated output.

Hot and Cold Tokens in DLMs. During diffusion decoding, currently masked positions are classified as hot state or cold state according to their likelihood of being accepted in subsequent decoding steps (Wei et al., 2026a). At diffusion step t, let $\mathcal{M}_t$ and $\mathcal{D}_t$ denote the sets of masked and accepted token positions, respectively. The hot token set is:

$$\mathcal{H}_t = \{i \in \mathcal{M}_t \mid c_{i,t} > \tau_h \text{ or } \exists q \in \mathcal{D}_t : |i-q| \le L_h\}, \tag{3}$$

where i and q index masked and accepted token positions, respectively, $c_{i,t}$ is the confidence at position i, τ_h is the confidence threshold, and L_h is the neighborhood radius. The confidence is:

$$c_{i,t} = \max_{v\in\mathcal{V}} \Pr_{\theta}(y_i = v \mid \mathbf{x}, \mathbf{Y}_t). \tag{4}$$

The vocabulary is denoted by $\mathcal{V}$, the candidate token by v, the model parameters by θ, and the token at position i by y_i. The conditioning input is denoted by $\mathbf{x}$, and the partially masked target sequence at step t is denoted by $\mathbf{Y}_t$. In particular, $\mathbf{x}$ is the conditioning input, and $\mathbf{Y}_t$ is the partially masked sequence. The remaining masked positions form the cold token set $\mathcal{C}_t = \mathcal{M}_t \setminus \mathcal{H}_t$.

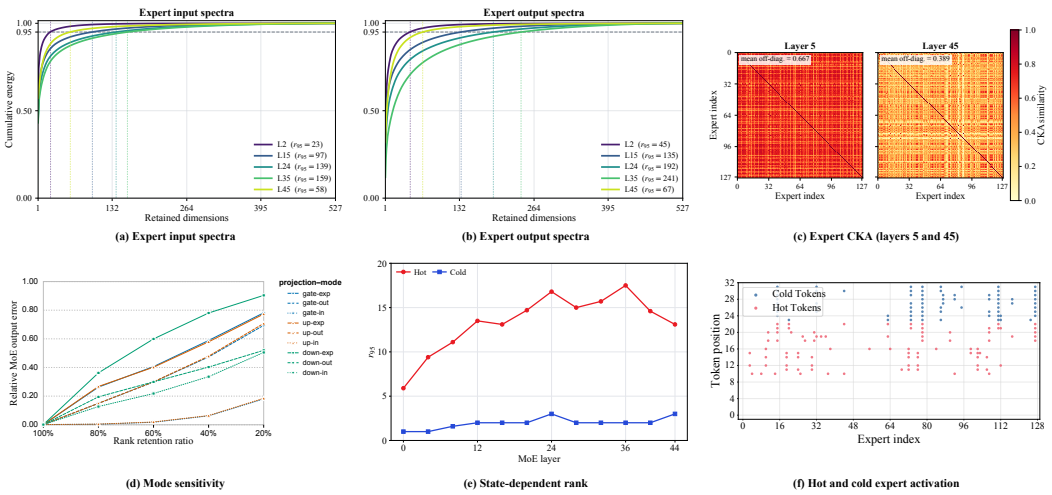


Figure 2: **Observations.** Results on SDAR-30B-A3B-Chat-b32 (Cheng et al., 2026). Panels (a)-(d) use the MBPP benchmark, while (e) and (f) use the GSM8K benchmark. **(a,b) Expert weight spectra.** Cumulative spectral energy of the layerwise expert-weight tensor along input- and output-feature modes, respectively. **(c) Expert representation similarity.** Pairwise linear CKA between expert output representations within a layer, where higher values indicate higher similarity. "mean off-diag." is the mean off-diagonal CKA. **(d) Mode sensitivity.** Relative MoE output reconstruction error *vs.* dense reference when each Tucker mode is truncated independently with the other two at full rank. **(e) Token-dependent rank.** Ranks retaining 95% energy for hot and cold tokens across MoE layers. **(f) Token-dependent expert patterns.** Distribution of experts activated by cold and hot tokens under Top-8 routing.

3.2 Motivation

As shown in Fig. 2, we identify two key properties of MoE DLMs: *cross-mode non-uniform redundancy* and *token-wise utilization variation*. Specifically, Fig. 2(a) and (b) show that the spectra of both input and output modes concentrate most of their energy in a few dominant directions, indicating a strong low-rank structure. Fig. 2(c) further presents the linear CKA values (Kornblith et al., 2019) between paired experts, which quantify the similarity of their output feature geometries and thus reflect structural redundancy across experts (see Appendix A.1 for details). The observed repeated patterns across experts confirm the presence of cross-expert redundancy. Fig. 2(d) further shows that different modes exhibit varying sensitivity to rank truncation. Together, these results reveal substantial cross-mode non-uniform redundancy in MoE DLMs, suggesting that effective compression should adaptively exploit redundancy across all modes. Moreover, as shown in Fig. 2(e), hot-token features exhibit broader energy distributions, whereas cold tokens concentrate their energy in fewer ranks, indicating that hot tokens are more sensitive to rank truncation. Fig. 2(f) also shows that, compared with hot tokens, cold tokens generally concentrate their activated experts on a smaller set. These results indicate substantial token-wise utilization variation within MoE DLMs, suggesting that effective compression should be tailored to different token states.

However, existing low-rank MoE compression methods (Li et al., 2025; Gu et al., 2025; Chen et al., 2026; Mi et al., 2026; Wei et al., 2026a) fail to jointly accommodate these two properties. They either factorize each expert independently without explicitly exploiting non-uniform redundancy across all of the input, output, and expert modes, or pre-determine rank budgets and expert routing statically and uniformly. Such static schemes cannot adapt to the token-wise utilization variation that emerges dynamically during the denoising process of MoE DLMs (Huang et al., 2026; Jazbec et al., 2026), limiting their effectiveness and efficiency.

3.3 Importance-guided Token-aware Compression for MoE DLMs

As shown in Fig. 1, the proposed Importance-guided Token-aware Compression (ITC-MoE) contains two components, including importance-guided Adaptive Tucker Compression (IATC) and token-aware Compensation and Routing (TCR).

3.3.1 IMPORTANCE-GUIDED ADAPTIVE TUCKER COMPRESSION

To adapt cross-mode non-uniform redundancy, as shown in Fig. 1, ITC-MoE introduces Importance-guided Adaptive Tucker Compression (IATC), which consists of three sequential steps, including Importance-Weighted Transformation, Tucker Factorization, and Learnable Rank Truncation.

Importance-Weighted Transformation. For the i-th expert $\mathbf{E}_i$, we consider a weight matrix $\mathbf{W}_i \in \mathbb{R}^{d_2 \times d_1}$ that may be $\mathbf{W}_i^{\text{down}}$, $\mathbf{W}_i^{\text{gate}}$, or $\mathbf{W}_i^{\text{up}}$. Its input and output activations are denoted by $\mathbf{A}_{\text{in}} \in \mathbb{R}^{d_1}$ and $\mathbf{A}_{\text{out}} \in \mathbb{R}^{d_2}$, respectively, and the model loss over the calibration set is denoted by $\mathcal{L}_{\text{cal}}$. We define the output gradient, activation second-moment matrix, and gradient second-moment matrix as:

$$\mathbf{g} = \partial \mathcal{L}_{\text{cal}} / \partial \mathbf{A}_{\text{out}}, \qquad \mathbf{\Sigma}_{\text{act}} = \mathbb{E}_{\text{cal}}\left[\mathbf{A}_{\text{in}} \mathbf{A}_{\text{in}}^{\top}\right], \qquad \mathbf{\Sigma}_{\mathbf{g}} = \mathbb{E}_{\text{cal}}\left[\mathbf{g}\mathbf{g}^{\top}\right], \tag{5}$$

where $\mathbb{E}_{\text{cal}}[\cdot]$ aggregates statistics over calibration tokens for the corresponding projection. Here, $\mathbf{\Sigma}_{\text{act}}$ captures frequently activated input directions, while $\mathbf{\Sigma}_{\mathbf{g}}$ captures loss-sensitive output directions. To avoid ill-conditioning of the gradient statistics, we use the shrinkage estimator (Ollila, 2024):

$$\mathbf{\Sigma}'_{\mathbf{g}} = (1-\eta)\mathbf{I}_{d_2} + \eta \frac{\mathbf{\Sigma}_{\mathbf{g}}}{\text{tr}(\mathbf{\Sigma}_{\mathbf{g}})/d_2}, \tag{6}$$

where $\mathbf{I}_{d_2}$ is the $d_2 \times d_2$ identity matrix and $\eta \in [0,1]$ controls the shrinkage strength. Let $\mathbf{L}_{\text{act}}$ and $\mathbf{L}_{\mathbf{g}}$ be the Cholesky inverses factors (Lu, 2021) satisfying $\mathbf{\Sigma}_{\text{act}} = \mathbf{L}_{\text{act}}\mathbf{L}_{\text{act}}^{\top}$ and $\mathbf{\Sigma}'_{\mathbf{g}} = \mathbf{L}_{\mathbf{g}}\mathbf{L}_{\mathbf{g}}^{\top}$. We then transform $\mathbf{W}_i$ into an importance-weighted representation $\widetilde{\mathbf{W}}_i = \mathbf{L}_{\mathbf{g}}^{\top}\mathbf{W}_i\mathbf{L}_{\text{act}}$. This emphasizes reconstruction along frequently activated input and loss-sensitive output directions, yielding a suitable representation for low-rank factorization (Martens & Grosse, 2015; Weng et al., 2024). See Appendix A.2 for theoretical analysis.

Tucker Factorization. Then, for each MoE layer, we stack the importance-weighted expert matrices along the expert mode to form $\widetilde{\mathcal{W}} = \text{stack}_{i=1}^{E}(\widetilde{\mathbf{W}}_i) \in \mathbb{R}^{E \times d_2 \times d_1}$, where E is the number of experts. The resulting tensor is factorized by Tucker factorization (Yamamoto et al., 2022; Lei et al., 2026):

$$\widetilde{\mathcal{W}} \approx \widetilde{\mathcal{G}} \times_1 \widetilde{\mathbf{U}}_{\text{exp}} \times_2 \widetilde{\mathbf{U}}_2 \times_3 \widetilde{\mathbf{U}}_1, \tag{7}$$

where $\times_n$ denotes the mode-n tensor-matrix product and the Tucker core $\widetilde{\mathcal{G}} \in \mathbb{R}^{r_{\text{exp}} \times r_2 \times r_1}$ captures interactions among modes. The factor matrices $\widetilde{\mathbf{U}}_{\text{exp}} \in \mathbb{R}^{E \times r_{\text{exp}}}$, $\widetilde{\mathbf{U}}_2 \in \mathbb{R}^{d_2 \times r_2}$, $\widetilde{\mathbf{U}}_1 \in \mathbb{R}^{d_1 \times r_1}$ correspond to the expert, output-feature, and input-feature modes, with retained ranks r_{exp}, r_2, and r_1. Setting $r_{\text{exp}} < E$, $r_2 < d_2$, and $r_1 < d_1$ reduces both the parameter footprint and computational overhead. The reconstructed weight in the original space is:

$$\widehat{\mathbf{W}}_i = \mathbf{L}_{\mathbf{g}}^{-\top}\widetilde{\mathbf{U}}_2 \left(\sum\nolimits_{q=1}^{r_{\text{exp}}} [\widetilde{\mathbf{U}}_{\text{exp}}]_{i,q} \widetilde{\mathcal{G}}^{(q)}\right) \widetilde{\mathbf{U}}_1^{\top}\mathbf{L}_{\text{act}}^{-1}, \tag{8}$$

where $[\widetilde{\mathbf{U}}_{\text{exp}}]_{i,q}$ is the coefficient of the q-th expert-mode component for expert $\mathbf{E}_i$, and $\widetilde{\mathcal{G}}^{(q)}$ is the q-th matrix slice of $\widetilde{\mathcal{G}}$ along the expert mode. Since $\widetilde{\mathbf{U}}_1$, $\widetilde{\mathbf{U}}_2$, and the Cholesky inverses are fixed during inference, they can be fused offline and stored in advance, avoiding additional online computation.

Learnable Rank Truncation. As shown in Fig. 2(d), different modes exhibit varying sensitivity to rank truncation, making uniform rank allocation suboptimal for balancing compression efficiency and fidelity. Motivated by this, we therefore propose Learnable Rank Truncation, which adaptively allocates the rank budget across modes by minimizing output reconstruction error. Consider a weight tensor $\widetilde{\mathcal{W}} \in \mathbb{R}^{E \times d_2 \times d_1}$ with Tucker ranks r_a, $a \in \{\text{exp}, 2, 1\}$. Under a fixed parameter budget $N_{\text{param}} = r_{\text{exp}}r_2r_1 + Er_{\text{exp}} + d_2r_2 + d_1r_1$, the discrete ranks cannot be directly optimized due to non-differentiable truncation. We therefore relax each rank into a continuous truncation position:

$$\rho_a = r_a^{\text{lower}} + \left(r_a^{\text{upper}} - r_a^{\text{lower}}\right)\sigma(\vartheta_a + \xi), \tag{9}$$

where r_a^{lower} and r_a^{upper} are the minimum and maximum ranks of mode a, respectively. The learnable parameter ϑ_a determines the relative rank allocation across modes, while the shared offset ξ is determined via binary search to satisfy the fixed budget N_{param}. A differentiable soft mask for the k-th component of mode a is given by $m_{a,k} = \sigma\left(\frac{\rho_a - k}{\tau}\right)$, $k = 1, \ldots, D_a$, where D_a is the

dimension size of mode a and τ is the temperature. The mask preserves components before ρ_a and suppresses those beyond it, providing a differentiable approximation to hard truncation. The temperature τ is gradually annealed during optimization, causing the soft masks to approach hard truncation. The rank allocation is optimized with the reconstruction loss:

$$\mathcal{L}_{\text{rank}} = (\mathbf{h}_{\text{out}} - \widehat{\mathbf{h}}_{\text{out}})^\top \mathbf{H} (\mathbf{h}_{\text{out}} - \widehat{\mathbf{h}}_{\text{out}}), \tag{10}$$

where $\mathbf{h}_{\text{out}}$ and $\widehat{\mathbf{h}}_{\text{out}}$ are the dense and softly truncated outputs, respectively, and $\mathbf{H}$ is the trace-normalized second-moment matrix of output gradients aggregated over all experts. Minimizing $\mathcal{L}_{\text{rank}}$ thus allocates more rank capacity to modes that are more sensitive to truncation.

Shared-factor Execution Strategy. This factorization enables a shared-factor execution strategy that accelerates runtime. For up and gate projections, the input factors are shared across experts since all activated experts receive the same input. Thus, the shared input projections are computed once and reused. For the down projection, both input and output factors are shared. Although the intermediate activations differ, the per-expert weighted contributions can be aggregated in the low-rank core space, followed by a single shared output projection. This lowers the computational cost of both input and output projections while preserving the compressed MoE formulation. In our experiments, we found that this strategy achieves up to a $2.18\times$ speedup. Full details are in Appendix A.3.

3.3.2 Token-aware Compensation and Routing

Then, to adapt the token-wise utilization variation, as shown in Fig. 1, ITC-MOE introduces Token-aware Compensation and Routing (TCR), including Hot Token Compensation and Cold Token Restricted Routing.

Hot Token Compensation. To protect hot tokens likely retained in later decoding, we introduce trainable compensation. For hot tokens, we augment each truncated Tucker reconstructed weight $\widehat{\mathbf{W}}_i$ with a residual term $\widehat{\mathbf{W}}_i^+ = \widehat{\mathbf{W}}_i + \mathbf{A}_i \operatorname{diag}(\boldsymbol{\alpha}) \mathbf{B}_i$, where $\mathbf{A}_i \in \mathbb{R}^{d_2 \times r_{\text{res}}}$ and $\mathbf{B}_i \in \mathbb{R}^{r_{\text{res}} \times d_1}$ span the compensation subspace, r_{res} is the residual rank, and $\boldsymbol{\alpha} \in \mathbb{R}^{r_{\text{res}}}$ are learnable parameters. As the initialization, we set $\boldsymbol{\alpha} = \mathbf{1}$ and solve $\boldsymbol{\alpha}, \mathbf{A}_i, \mathbf{B}_i$ by:

$$(\mathbf{A}_i^*, \mathbf{B}_i^*) = \arg\min_{\mathbf{A}_i, \mathbf{B}_i} \left\| \left(\mathbf{W}_i - \widehat{\mathbf{W}}_i - \mathbf{A}_i \mathbf{B}_i \right) \mathbf{L}_{\text{act}} \right\|_F^2, \tag{11}$$

where $\mathbf{L}_{\text{act}}$ is the Cholesky factor of the input second-moment matrix estimated from hot tokens, emphasizing input directions frequently activated by hot tokens. After initialization, $\mathbf{A}_i$ and $\mathbf{B}_i$ remain fixed, and only $\boldsymbol{\alpha}$ is trained with the following objective:

$$\mathcal{L}_{\text{comp}} = \left\| \widehat{\mathbf{h}}_{\text{out}}^+ - \mathbf{h}_{\text{out}} \right\|_2^2 + \lambda \left\| \widehat{\mathbf{h}}_{\text{out}}^+ - \widehat{\mathbf{h}}_{\text{out}}^{(\mathbf{1})} \right\|_2^2, \tag{12}$$

where $\mathbf{h}_{\text{out}}$ is the original uncompressed MoE-layer output, $\widehat{\mathbf{h}}_{\text{out}}^+$ is the compensated output, and $\widehat{\mathbf{h}}_{\text{out}}^{(\mathbf{1})}$ is the fixed reference output at $\boldsymbol{\alpha} = \mathbf{1}$. The first term minimizes reconstruction error relative to the original MoE-layer output, recovering truncation-induced information loss. The second anchors the compensated output to its initialization, preventing overfitting and stabilizing optimization. The coefficient λ controls regularization strength.

Table 1: Comparison of Top-8 recall for different candidate set sizes on cold and hot tokens. Top-8 recall is defined as the fraction of tokens whose ground-truth Top-8 expert set is fully contained in the candidate set. Evaluation uses the SDAR-30B-A3B-Chat-b32 model with held-out GSM8K and MBPP calibration prompts, with 128 experts and Top-8 routing per token.

M	Cold recall	Hot recall
16	0.8975	0.6930
32	0.9627	0.8186
48	0.9795	0.8834
64	0.9890	0.9257

Cold Token Restricted Routing. During diffusion decoding, numerous masked tokens repeatedly undergo expert routing, activating many distinct experts within a single forward pass and increasing expert weight memory access and execution overhead (Wei et al., 2026a; Fang et al., 2025). Fortunately, cold tokens generally concentrate their activated experts on a small set, indicating that full-expert routing is unnecessary. As shown in Fig. 2(f), with Top-8 expert selection, cold tokens collectively activate only about 17 experts, while hot tokens span 42. We attribute this disparity to higher routing similarity among cold tokens (Appendix A.4).

Motivated by this, we propose candidate-restricted routing that applies filtering exclusively to cold tokens, preserving routing fidelity while reducing computation. Specifically, for a cold token set $\mathcal{C}_t$ at step t, we compute the centroid $\bar{\mathbf{h}} = \frac{1}{|\mathcal{C}_t|}\sum_{i\in\mathcal{C}_t}\mathbf{h}_{\text{in},i}$, and obtain its routing probabilities $\boldsymbol{\pi}(\bar{\mathbf{h}}) \in \mathbb{R}^E$. The candidate expert set is then defined as $\mathcal{Q} = \text{Top}_M\big(\boldsymbol{\pi}(\bar{\mathbf{h}})\big)$, where $K < M < E$. Each cold token i performs Top-K selection strictly within $\mathcal{Q}$:

$$\mathcal{E}_i = \text{Top}_K\Big(\{\pi_j(\mathbf{h}_{\text{in},i}) : j \in \mathcal{Q}\}\Big), \tag{13}$$

where $\pi_j(\mathbf{h}_{\text{in},i})$ denotes the routing probability of token i for expert j. As reported in Table 1, with $M = 48$, the candidate set achieves a Top-8 recall of 97.95% for cold tokens, compared to 88.34% for hot tokens, confirming that cold tokens are more amenable to candidate-restricted routing. By limiting cold token expert selection to a set of size M, the routing computation cost is reduced from $\mathcal{O}(E)$ to $\mathcal{O}(M)$ per cold token, alleviating the expert-selection overhead. In our experiments, we found that this candidate-restricted routing achieves up to a 13.0% speedup.

4 Experiments

4.1 Experimental Setup

Baselines and configurations. We compare ITC-MoE with D^2-MoE (Gu et al., 2025) and TD-MoE (XU et al., 2026), with the uncompressed *Original* model serving as the performance baseline. Following the settings of the compared methods, ITC-MoE compresses only the Gate, Up, and Down projections of the MoE layers. We further compare with TEAM (Wei et al., 2026a), which focuses on model acceleration without model compression. For a fair comparison, all experiments are conducted on a single NVIDIA A800 GPU, with the same decoding configuration, prompts, and evaluation protocol. The detailed hyperparameters are provided in Appendix B. Following prior work (XU et al., 2026), we report the compression ratio over the MoE layers only.

Models, datasets, and metrics. We conduct zero-shot evaluation on SDAR-30B-A3B-Chat-b32 (Cheng et al., 2026) and LLaDA2.0-mini (Bie et al., 2025), covering MBPP (Austin et al., 2021), GSM8K (Cobbe et al., 2021), MultiArith, SingleOp, and SingleQ (Koncel-Kedziorski et al., 2016). Metrics include task accuracy and end-to-end speedup.

4.2 Main Results

Comparison under Different Compression Ratios. Table 2 compares ITC-MoE with D^2-MoE and TD-MoE under different compression ratios on five downstream benchmarks. ITC-MoE consistently achieves substantially higher speedups across the evaluated tasks while maintaining competitive task performance. In particular, since D^2-MoE and TD-MoE are not designed for MoE DLMs, they generally yield limited or even no speedup. For example, at a 30% compression ratio, D^2-MoE only achieves a 0.80× speedup on MBPP. Moreover, D^2-MoE and TD-MoE generally show unstable performance across benchmarks. For example, on MBPP at a 30% compression ratio, they only achieve scores of 34.24 and 12.45, respectively. In contrast, the proposed ITC-MoE present a score of 47.47 and also maintains stable and competitive performance across different tasks.

Comparison with Non-Compression Baseline. Table 3 compares ITC-MoE with TEAM on SDAR and LLaDA2.0-mini. Note that TEAM improves inference efficiency without compressing model parameters. Therefore, we set the compression ratio of ITC-MoE to a small value of 3% for a fair comparison. Across five benchmarks, ITC-MoE achieves substantially higher speedups than TEAM on both models while maintaining comparable task performance. For example, on SDAR, ITC-MoE achieves $3.46\times$-$4.61\times$ speedups versus $1.74\times$-$2.08\times$ for TEAM, while obtaining the highest score on MultiArith and matching the Original model on SingleQ. On LLaDA2.0-mini, it achieves $2.66\times$-$4.53\times$ speedups versus $1.26\times$-$1.72\times$ for TEAM, while achieving the highest score on MBPP and matching the Original model on SingleOp and SingleQ. These results demonstrate that ITC-MoE achieves a more favorable efficiency-performance trade-off than TEAM.

4.3 Ablation Studies

We use SDAR-30B-A3B-Chat-b32 as the default model for all ablation studies.

Table 2: Comparison of D^2-MoE (Gu et al., 2025), TD-MoE (XU et al., 2026), and our ITC-MoE on SDAR-30B-A3B-Chat-b32 Cheng et al. (2026) under different compression ratios. The orange-signed numbers indicate that the corresponding speedups are below $1.00\times$.

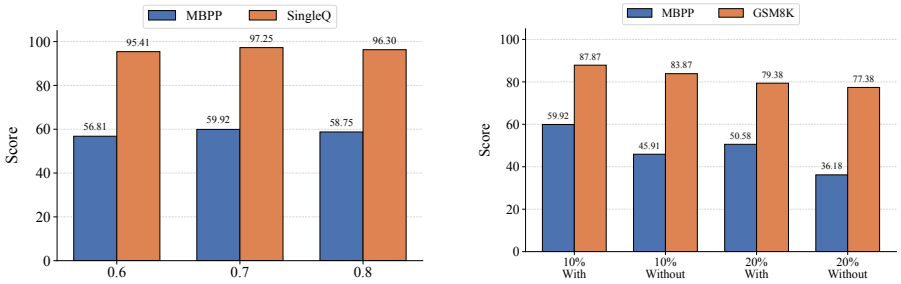
Benchmark	Method	10% Ratio		20% Ratio		30% Ratio	
		Score ↑	Speedup ↑	Score ↑	Speedup ↑	Score ↑	Speedup ↑
MBPP	Original	65.76	1.00×	65.76	1.00×	65.76	1.00×
	D^2-MoE	56.42	1.26×	**52.53**	1.14×	34.24	0.80×
	TD-MoE	58.75	1.34×	51.75	1.31×	12.45	1.35×
	ITC-MoE	**59.92**	**5.68×**	50.58	**5.68×**	**47.47**	**6.61×**
GSM8K	Original	90.60	1.00×	90.60	1.00×	90.60	1.00×
	D^2-MoE	83.47	0.86×	77.10	0.89×	**66.57**	0.82×
	TD-MoE	**88.48**	1.10×	72.18	0.96×	39.65	0.93×
	ITC-MoE	87.87	**4.60×**	**79.38**	**4.37×**	63.46	**4.71×**
MultiArith	Original	98.67	1.00×	98.67	1.00×	98.67	1.00×
	D^2-MoE	97.67	1.06×	96.83	0.87×	**98.00**	0.88×
	TD-MoE	98.17	1.22×	96.67	1.20×	73.83	1.09×
	ITC-MoE	**99.17**	**5.27×**	**99.33**	**5.86×**	96.33	**7.22×**
SingleOp	Original	97.15	1.00×	97.15	1.00×	97.15	1.00×
	D^2-MoE	**95.55**	0.95×	**96.62**	0.82×	**90.70**	0.80×
	TD-MoE	95.02	1.13×	89.15	1.05×	74.02	0.71×
	ITC-MoE	95.37	**5.34×**	95.02	**5.26×**	88.08	**5.56×**
SingleQ	Original	98.17	1.00×	98.17	1.00×	98.17	1.00×
	D^2-MoE	94.50	0.87×	**96.33**	0.84×	**88.83**	0.90×
	TD-MoE	96.33	1.20×	86.24	1.18×	61.47	1.00×
	ITC-MoE	**97.25**	**5.36×**	95.41	**5.69×**	84.40	**6.16×**

Table 3: Comparison of ITC-MoE and TEAM across different backbone models, with model compression applied to ITC-MoE.

Benchmark	Method	SDAR		LLaDA2.0-mini	
		Score ↑	Speedup ↑	Score ↑	Speedup ↑
MBPP	Original	65.76	1.00×	80.54	1.00×
	TEAM	**65.76**	2.08×	80.16	1.56×
	ITC-MoE	65.57	**4.61×**	**80.93**	**4.53×**
GSM8K	Original	90.60	1.00×	91.89	1.00×
	TEAM	**90.30**	1.83×	**92.80**	1.72×
	ITC-MoE	90.08	**3.46×**	92.56	**2.94×**
MultiArith	Original	98.67	1.00×	98.33	1.00×
	TEAM	98.83	1.77×	**98.17**	1.49×
	ITC-MoE	**99.00**	**4.09×**	**98.17**	**2.69×**
SingleOp	Original	97.15	1.00×	97.86	1.00×
	TEAM	95.20	1.89×	97.51	1.26×
	ITC-MoE	**95.55**	**4.31×**	**97.86**	**2.66×**
SingleQ	Original	98.17	1.00×	99.08	1.00×
	TEAM	95.41	1.74×	98.17	1.52×
	ITC-MoE	**98.17**	**3.89×**	**99.08**	**2.95×**

Sensitivity to the Hot-Token Confidence Threshold. Fig. 3a examines ITC-MoE's sensitivity to the hot-token confidence threshold τ_h with $L_h = 3$. Varying τ_h from 0.6 to 0.8 causes only small changes on MBPP and SingleQ, where MBPP ranges from 56.81 to 59.92, and SingleQ remains within 95.41–97.25. The best performance is achieved at $\tau_h = 0.7$.

Effect of Importance-Weighted Transformation. As shown in Figure 3, the importance-weighted transformation consistently improves performance. At a 10% compression rate, MBPP increases from 45.91 to 59.92 and GSM8K from 83.87 to 87.87. At 20%, MBPP increases from 36.18 to 50.58 and GSM8K from 77.38 to 79.38. These results indicate that the importance-weighted transformation effectively mitigates performance degradation caused by compression.



(a) Sensitivity analysis of the hot-token confidence threshold τ_h.

(b) Importance-Weighted Transformation ablation at 10% and 20% compression.

Figure 3: Hot-token threshold sensitivity and effect of Importance-Weighted Transformation.

Table 4: Effect of Hot-token compensation under the 10% compression ratio.

Setting	MBPP Score ↑	SingleQ Score ↑
w.	**59.92**	**97.25**
w/o.	51.75	94.49

Table 5: Comparison of learned rank allocation and fixed rank allocation across the three Tucker modes.

Compression Ratio	Setting	MBPP Score ↑	GSM8K Score ↑
10%	Learned ranks	**59.92**	**87.87**
10%	Fixed	53.09	84.00
20%	Learned ranks	**50.58**	**79.38**
20%	Fixed	46.19	68.00

Effectiveness of Hot Token Compensation. To evaluate the contribution of Hot-token compensation, we reduce the rank of the compensation residual term from 11 to 0 while keeping all other settings unchanged. As shown in Table 4, enabling Hot-token compensation improves the MBPP score from 51.75 to 59.92 and the SingleQ score from 94.49 to 97.25, demonstrating its effectiveness in recovering the representation loss introduced by compression.

Effectiveness of Learned Rank Allocation. As shown in Table 5, learned rank truncation consistently outperforms fixed-proportion allocation on both MBPP and GSM8K under the 10% and 20% compression ratios. More ablation studies are provided in Appendix C.

5 Limitations and Future Work

Despite achieving a superior balance between speedup and model capacity, ITC-MoE still exhibits a performance gap relative to the full model, which can be alleviated by stronger compensation mechanisms. In addition, although candidate-restricted routing reduces the expert-selection complexity for cold tokens from $O(E)$ to $O(M)$, further hardware-aware implementations that fully exploit this optimization are still required to translate this algorithmic advantage more fully into end-to-end inference speedup.

6 Conclusion

In this paper, we identify two key properties of Mixture-of-Experts (MoE) Diffusion Language Models (DLMs): cross-mode non-uniform redundancy and token-wise utilization variation. The former stems from heterogeneous redundancy and truncation sensitivity across expert, input-feature, and output-feature modes, while the latter is reflected in the distinct spectral characteristics and expert activation patterns of hot and cold tokens. Motivated by these observations, we propose Importance-guided Token-aware Compression (ITC-MoE), comprising importance-guided Adaptive Tucker Compression (IATC) and token-aware Compensation and Routing (TCR). IATC performs importance-guided Tucker factorization and adaptively allocates ranks across modes under a fixed parameter budget. TCR mitigates information loss via lightweight low-rank compensation for compression-sensitive hot tokens, while applying candidate-restricted routing to cold tokens with concentrated expert activation patterns. Experiments across diverse DLMs and datasets confirm that, compared with existing methods, ITC-MoE achieves superior performance and efficiency.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing, code development, and debugging. Generative AI tools were not used to generate experimental results or serve as a source of unverified research conclusions or literature evidence. All AI-assisted content has been manually checked and verified by the authors. The authors take full responsibility for the final content of this paper, including all text, code, claims, results, and other related research materials.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we have included source code and necessary experimental configurations in the supplementary material to facilitate reproduction of the main results reported in this work. The reproducibility of our paper is also supported by providing complete descriptions of the proposed method, implementation details, and experimental settings in the experimental section and appendix. Additional ablation studies and supplementary results are also provided in the appendix.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021.

Sikai Bai, Haoxi Li, Jie ZHANG, Zicong Hong, and Song Guo. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025.

Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. MoBE: Mixture-of-basis-experts for compressing MoE-based LLMs. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026.

Zhixuan Chen, Xing Hu, Dawei Yang, Zukang Xu, Xu Chen, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. MoEQuant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-AutoRegression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Haojie Duanmu, Xiuhong Li, Zhihang Yuan, Size Zheng, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. MxMoE: Mixed-precision quantization for MoE with accuracy and performance co-design. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Zhiyuan Fang, Yuegui Huang, Zicong Hong, Yufeng Lyu, Wuhui Chen, Yue Yu, Fan Yu, and Zibin Zheng. Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2025.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Beichen Huang, Yueming Yuan, ZELEI SHAO, and Minjia Zhang. Milo: Efficient quantized moe inference with mixture of low-rank compensators. In *Proceedings of Machine Learning and Systems(MLSys)*, 2025a.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Fei Chao, Rongrong Ji, and Liujuan Cao. Discovering important experts for mixture-of-experts models pruning through a theoretical perspective. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, Joao Monteiro, Victor Guilherme Turrisi da Costa, Jason Ramapuram, and marco cuturi. Learning unmasking policies for diffusion language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Shibo Jie, Yehui Tang, Kai Han, Yitong Li, Duyu Tang, Zhi-Hong Deng, and Yunhe Wang. Mixture of lookup experts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 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. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2016.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

Qifeng Lei, Zhiyong Yang, Qianqian Xu, Cong Hua, Peisong Wen, and Qingming Huang. Tucka: hierarchical compact tensor experts for efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Moe-svd: Structured mixture-of-experts LLMs compression via singular value decomposition. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jun Lu. Numerical matrix decomposition, 2021.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Zhendong Mi, Yixiao Chen, Pu Zhao, Xiaodong Yu, Hao Wang, Yanzhi Wang, and Shaoyi Huang. Effective moe-based LLM compression by exploiting heterogeneous inter-group experts routing frequency and information density. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan LI. Large language diffusion models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Esa Ollila. Linear shrinkage of sample covariance matrix or matrices under elliptical distributions: A review. In *Elliptically Symmetric Distributions in Signal Processing and Machine Learning*. Springer, 2024.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2024.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Vikash Sehwag, Xianghao Kong, Jingtao Li, Michael Spranger, and Lingjuan Lyu. Stretching each dollar: Diffusion training from scratch on a micro-budget. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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, 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 *Proceedings of International Conference on Learning Representations (ICLR)*, 2025.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026.

Linye Wei, Zixiang Luo, Pingzhi Tang, and Meng Li. TEAM: Temporal–spatial consistency guided expert activation for moe diffusion language model acceleration. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026a.

Yujie Wei, Shiwei Zhang, Hangjie Yuan, Yujin Han, Zhekai Chen, Jiayu Wang, Difan Zou, Xihui Liu, yingya zhang, Yu Liu, and Hongming Shan. Routing matters in moe: Scaling diffusion transformers with explicit routing guidance. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026b.

Xi Weng, Yunhao Ni, Tengwei Song, Jie Luo, Rao Anwer, Salman Khan, Fahad Khan, and Lei Huang. Modulate your spectrum in self-supervised learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2024.

Yuebin XU, YANHONG WANG, Xuemei Peng, Hui Zang, Minghao Chen, Pengfei Xia, and Zeyi Wen. Td-moe: Tensor decomposition for moe models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. Kbvq-moe: Klt-guided svd with bias-corrected vector quantization for moe large language models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2026.

Ryuki Yamamoto, Hidekata Hontani, Akira Imakura, and Tatsuya Yokota. Fast algorithm for low-rank tensor completion in delay-embedded space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Geng Zhang, Han Yuxuan, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. Mone: Replacing redundant experts with lightweight novices for structured pruning of moe. In *Proceedings of International Conference on Learning Representations (ICLR)*, 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Jixian Zhou, Fang Dong, Ruijun Huang, Hengjie Cao, Mengyi Chen, Yifeng Yang, Anrui Chen, Mingzhi Dong, Yujiang Wang, Dongsheng Li, David A. Clifton, Qin Lv, Rui Zhu, Chun Zhang, Fan Yang, Tun Lu, Ning Gu, and Li Shang. Oracle-MoE: Locality-preserving routing in the oracle space for memory-constrained large language model inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025a.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yuxin Zhou, Zheng Li, Jun Zhang, Jue Wang, Yiping Wang, Zhongle Xie, Ke Chen, and Lidan Shou. FloE: On-the-fly MoE inference on memory-constrained GPU. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025b.

Fengqi Zhu, Shaoxuan Xu, Jingyang Ou, Zebin You, Yipeng Xing, Huabin Liu, Xiaolu Zhang, Jun Zhou, Zhenzhong Lan, Yankai Lin, Wayne Xin Zhao, Jianguo Li, Chongxuan Li, and Ji-Rong Wen. Llada moe v2: Scaling mixture-of-experts diffusion language models, 2026.

A Appendix

A.1 Centered Kernel Alignment

To measure the similarity of experts within a MoE layer, we employ Centered Kernel Alignment (CKA) (Kornblith et al., 2019), which captures the geometric structure of output features. For any two experts $\mathbf{E}_i$ and $\mathbf{E}_j$ located in the same MoE layer, we first collect their output feature matrices $\mathbf{X}_i, \mathbf{X}_j \in \mathbb{R}^{B\times d}$ using the same set of B calibration samples, where d denotes the output dimension. We then center these features by subtracting the column-wise means:

$$\widetilde{\mathbf{X}}_i = \mathbf{J}_B\mathbf{X}_i, \qquad \widetilde{\mathbf{X}}_j = \mathbf{J}_B\mathbf{X}_j, \qquad \mathbf{J}_B = \mathbf{I}_B - \frac{1}{B}\mathbf{1}_B\mathbf{1}_B^\top, \tag{14}$$

where $\mathbf{J}_B$ is the centering matrix, $\mathbf{I}_B$ is the B-dimensional identity matrix, and $\mathbf{1}_B$ is the B-dimensional all-ones vector. Finally, the linear CKA between the two experts is defined as:

$$\mathrm{CKA}(\mathbf{X}_i, \mathbf{X}_j) = \frac{\left\|\widetilde{\mathbf{X}}_i^\top \widetilde{\mathbf{X}}_j\right\|_F^2}{\left\|\widetilde{\mathbf{X}}_i^\top \widetilde{\mathbf{X}}_i\right\|_F \left\|\widetilde{\mathbf{X}}_j^\top \widetilde{\mathbf{X}}_j\right\|_F}, \tag{15}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This score is invariant to isotropic scaling and lies in $[0, 1]$, with higher values indicating stronger alignment between the geometric patterns induced by the two experts.

A.2 Rationale for Importance-Weighted Transformation

Standard Tucker decomposition typically identifies low-rank structure by minimizing the Frobenius reconstruction error. Let $\mathbf{W}_i$ and $\widehat{\mathbf{W}}_i$ denote the original and compressed weights of expert i, respectively, and let $\mathbf{\Delta}_i = \mathbf{W}_i - \widehat{\mathbf{W}}_i$ denote the reconstruction error. The standard objective is:

$$\min_{\widehat{\mathcal{W}}} \left\|\mathcal{W} - \widehat{\mathcal{W}}\right\|_F^2 = \min_{\widehat{\mathcal{W}}} \sum_{i=1}^{E} \|\mathbf{\Delta}_i\|_F^2. \tag{16}$$

Here, $\mathcal{W}$ and $\widehat{\mathcal{W}}$ are tensors obtained by stacking the original and compressed weights along the expert mode, and E is the number of experts. This objective penalizes reconstruction errors equally across weight directions. However, input directions differ in activation energy, and output directions differ in their sensitivity to the task loss. Consequently, weight errors of the same magnitude can have different effects on task performance.

The input second-moment matrix $\mathbf{\Sigma}_{\mathrm{act}}$ characterizes the energy of calibration inputs along different directions. Let $\mathbf{q}_j^{\mathrm{act}}$ be a unit eigenvector of $\mathbf{\Sigma}_{\mathrm{act}}$, with corresponding eigenvalue λ_j^{act}. Then:

$$\mathbf{\Sigma}_{\mathrm{act}}\mathbf{q}_j^{\mathrm{act}} = \lambda_j^{\mathrm{act}}\mathbf{q}_j^{\mathrm{act}}, \mathbb{E}_{\mathrm{cal}}\left[\left((\mathbf{q}_j^{\mathrm{act}})^\top \mathbf{A}\right)^2\right] \quad = \lambda_j^{\mathrm{act}}. \tag{17}$$

Here, j indexes input directions, $\mathbf{A}$ is the projection input, and $\mathbb{E}_{\mathrm{cal}}$ denotes expectation over calibration samples. A large λ_j^{act} indicates that calibration inputs have substantial energy along $\mathbf{q}_j^{\mathrm{act}}$, so weight errors along this direction can produce larger output deviations. Conversely, when λ_j^{act} is close to zero, calibration inputs have small components along this direction, reducing its contribution to output reconstruction error. Specifically:

$$\mathbb{E}_{\mathrm{cal}}\left[\|\mathbf{\Delta}_i\mathbf{A}\|_2^2\right] = \mathrm{tr}\left(\mathbf{\Delta}_i^\top \mathbf{\Delta}_i \mathbf{\Sigma}_{\mathrm{act}}\right) \quad = \sum_{j=1}^{d_{\mathrm{in}}} \lambda_j^{\mathrm{act}} \left\|\mathbf{\Delta}_i \mathbf{q}_j^{\mathrm{act}}\right\|_2^2. \tag{18}$$

Here, d_{in} is the input dimension and $\mathrm{tr}(\cdot)$ denotes the matrix trace. This identity shows that the input second moment distinguishes reconstruction errors according to the energy of the corresponding input directions.

The importance of an output error further depends on its influence on the task loss. The output gradient $\mathbf{g}_i = \partial\mathcal{L}_{\mathrm{cal}}/\partial\mathbf{o}_i$ and its second-moment matrix $\mathbf{\Sigma}_{\mathbf{g}}$, defined in the main text, characterize this

sensitivity. Each output gradient $\mathbf{g}_i$ first forms its outer product $\mathbf{g}_i\mathbf{g}_i^\top$; these outer products are then accumulated over all routed token-expert records in the layer into the shared second-moment matrix $\mathbf{\Sigma}_\mathbf{g}$, which after normalization and shrinkage is Cholesky-factorized as $\widetilde{\mathbf{\Sigma}}_\mathbf{g} = \mathbf{L}_\mathbf{g}\mathbf{L}_\mathbf{g}^\top$, yielding the shared factor $\mathbf{L}_\mathbf{g}$. Let $\mathbf{q}_{i,k}^g$ be a unit eigenvector of $\mathbf{\Sigma}_\mathbf{g}$, with corresponding eigenvalue $\lambda_{i,k}^g$. Then:

$$\mathbf{\Sigma}_\mathbf{g}\mathbf{q}_{i,k}^g = \lambda_{i,k}^g \mathbf{q}_{i,k}^g, \mathbb{E}_{\text{cal}}\left[\left((\mathbf{q}_{i,k}^g)^\top \mathbf{g}\right)^2\right] \quad = \lambda_{i,k}^g. \tag{19}$$

Here, k indexes output directions, $\mathcal{L}_{\text{cal}}$ is the calibration loss, and $\mathbf{o}_i$ is the projection output. Under a local first-order approximation of the loss, the directional derivative along $\mathbf{q}_{i,k}^g$ is $(\mathbf{q}_{i,k}^g)^\top \mathbf{g}$. Thus, a larger $\lambda_{i,k}^g$ indicates greater mean squared sensitivity along this output direction, motivating a stronger penalty on reconstruction errors in that direction.

Based on these observations, the transformation $\widetilde{\mathbf{W}}_i = \mathbf{L}_\mathbf{g}^\top \mathbf{W}_i \mathbf{L}_{\text{act}}$ incorporates both input energy and output sensitivity into the reconstruction metric. Define $\widehat{\widetilde{\mathbf{W}}}_i = \mathbf{L}_\mathbf{g}^\top \widehat{\mathbf{W}}_i \mathbf{L}_{\text{act}}$. Using the Cholesky factorizations $\mathbf{\Sigma}_{\text{act}} = \mathbf{L}_{\text{act}}\mathbf{L}_{\text{act}}^\top$ and $\widetilde{\mathbf{\Sigma}}_\mathbf{g} = \mathbf{L}_\mathbf{g}\mathbf{L}_\mathbf{g}^\top$ from the main text, we obtain:

$$\begin{aligned}\left\|\widetilde{\mathbf{W}}_i - \widehat{\widetilde{\mathbf{W}}}_i\right\|_F^2 &= \left\|\mathbf{L}_\mathbf{g}^\top \mathbf{\Delta}_i \mathbf{L}_{\text{act}}\right\|_F^2 \\ &= \text{tr}\big(\mathbf{L}_\mathbf{g}^\top \mathbf{\Delta}_i \mathbf{L}_{\text{act}} \mathbf{L}_{\text{act}}^\top \mathbf{\Delta}_i^\top \mathbf{L}_\mathbf{g}\big) \\ &= \text{tr}\Big(\mathbf{\Delta}_i^\top \widetilde{\mathbf{\Sigma}}_\mathbf{g} \mathbf{\Delta}_i \mathbf{\Sigma}_{\text{act}}\Big).\end{aligned} \tag{20}$$

Here, $\widetilde{\mathbf{\Sigma}}_\mathbf{g}$ is the normalized and shrunk gradient second-moment matrix used in the main text. This transformation therefore changes the reconstruction metric in weight space: the ordinary Frobenius error in the transformed space is exactly equivalent to a weighted error in the original space that accounts for both input energy and task sensitivity. We define this task-aware reconstruction error as:

$$\mathcal{E}_i\left(\mathbf{W}_i, \widehat{\mathbf{W}}_i\right) = \text{tr}\Big[(\mathbf{W}_i - \widehat{\mathbf{W}}_i)^\top \widetilde{\mathbf{\Sigma}}_\mathbf{g} (\mathbf{W}_i - \widehat{\mathbf{W}}_i) \mathbf{\Sigma}_{\text{act}}\Big]. \tag{21}$$

Here, $\mathcal{E}_i$ measures the weighted reconstruction error of expert i. It places stronger penalties on errors along input directions with higher activation energy and output directions with greater loss sensitivity. This quantity serves as a task-aware compression surrogate rather than an exact expression for the change in task loss.

Finally, second-moment weighting and Tucker factorization can be expressed as a unified optimization problem:

$$\begin{aligned}\min_{\substack{\widetilde{\mathcal{G}}, \widetilde{\mathbf{U}}_{\text{exp}}, \\ \widetilde{\mathbf{U}}_{\text{out}}, \widetilde{\mathbf{U}}_{\text{in}}}} \quad & \sum_{i=1}^{E} \mathcal{E}_i\left(\mathbf{W}_i, \widehat{\mathbf{W}}_i\right) \\ \text{s.t.} \quad & \widehat{\widetilde{\mathcal{W}}} = \widetilde{\mathcal{G}} \times_1 \widetilde{\mathbf{U}}_{\text{exp}} \times_2 \widetilde{\mathbf{U}}_{\text{out}} \times_3 \widetilde{\mathbf{U}}_{\text{in}}, \quad \widehat{\mathbf{W}}_i = \mathbf{L}_\mathbf{g}^{-\top} \widehat{\widetilde{\mathbf{W}}}_i \mathbf{L}_{\text{act}}^{-1}.\end{aligned} \tag{22}$$

Here, $\widehat{\widetilde{\mathcal{W}}}$ stacks the reconstructed transformed weights $\widehat{\widetilde{\mathbf{W}}}_i$ along the expert mode. The core tensor $\widetilde{\mathcal{G}}$ and the factor matrices $\widetilde{\mathbf{U}}_{\text{exp}}$, $\widetilde{\mathbf{U}}_{\text{out}}$, and $\widetilde{\mathbf{U}}_{\text{in}}$ follow the dimensions and compression ranks defined in the main text, and $\times_n$ denotes the mode-n product. The Tucker constraint is imposed in the weighted space. Thus, second-moment weighting first determines which reconstruction errors matter more, and Tucker factorization then identifies shared low-rank structure across the expert, output-feature, and input-feature modes under this metric.

A.3 Shared-factor Execution Strategy.

For a given MoE layer, let $\mathcal{E}$ be the set of K activated experts, and let $\widehat{\pi}_i$ be the normalized routing weight of expert $\mathbf{E}_i$. The MoE output is:

$$\mathbf{A}_{\text{out}} = \sum_{i \in \mathcal{E}} \widehat{\pi}_i \, \mathbf{W}_i^{\text{down}} \Big[\phi\big(\mathbf{W}_i^{\text{gate}} \mathbf{A}_{\text{in}}\big) \odot \big(\mathbf{W}_i^{\text{up}} \mathbf{A}_{\text{in}}\big)\Big], \tag{23}$$

where $\mathbf{A}_{\text{in}}$ is the input hidden state, ϕ is the element-wise activation, and $\odot$ denotes element-wise multiplication. Using Eq. 8, the importance-weighted transformation can be absorbed into the factors as follows:

$$\hat{\mathbf{U}}_2 = \mathbf{L}_g^{-\top}\widetilde{\mathbf{U}}_2, \qquad \hat{\mathbf{U}}_1^{\top} = \widetilde{\mathbf{U}}_1^{\top}\mathbf{L}_{\text{act}}^{-1}, \qquad \widetilde{\mathbf{G}}_i = \sum_{q=1}^{r_{exp}} [\widetilde{\mathbf{U}}_{\text{exp}}]_{i,q}\widetilde{\mathbf{G}}^{(q)}. \tag{24}$$

Therefore, given $\mathbf{W}_i \in \mathbb{R}^{d_2 \times d_1}$ corresponding to one of $\mathbf{W}_i^{\text{down}}$, $\mathbf{W}_i^{\text{gate}}$, or $\mathbf{W}_i^{\text{up}}$ for E_i, $\mathbf{W}_i$ is decomposed as follows:

$$\mathbf{W}_i = \hat{\mathbf{U}}_2\widetilde{\mathbf{G}}_i\hat{\mathbf{U}}_1^{\top}, \tag{25}$$

where $\hat{\mathbf{U}}_1 \in \mathbb{R}^{d_1 \times r_1}$ is the input factor, $\hat{\mathbf{U}}_2 \in \mathbb{R}^{d_2 \times r_2}$ is the output factor, and $\widetilde{\mathbf{G}}_i \in \mathbb{R}^{r_2 \times r_1}$ is the Tucker core.

(1) Up projection and Gate projection. For up projection $\mathbf{W}_i^{\text{up}}$, all activated experts receive the same $\mathbf{A}_{\text{in}}$. Thus, by using Eq. 25, the up projection is:

$$\mathbf{W}_i^{\text{up}}\mathbf{A}_{\text{in}} = \hat{\mathbf{U}}_2^{\text{up}}\widetilde{\mathbf{G}}_i^{\text{up}}\left[\left(\hat{\mathbf{U}}_1^{\text{up}}\right)^{\top}\mathbf{A}_{\text{in}}\right]. \tag{26}$$

Since $\left(\hat{\mathbf{U}}_1^{\text{up}}\right)^{\top}\mathbf{A}_{\text{in}}$ is identical across experts, it is computed once and reused, reducing the number of input projections from $\mathcal{O}(K)$ to one.

The gate projection $\mathbf{W}_i^{\text{gate}}$ also operates on the same $\mathbf{A}_{\text{in}}$:

$$\mathbf{W}_i^{\text{gate}}\mathbf{A}_{\text{in}} = \hat{\mathbf{U}}_2^{\text{gate}}\widetilde{\mathbf{G}}_i^{\text{gate}}\left[\left(\hat{\mathbf{U}}_1^{\text{gate}}\right)^{\top}\mathbf{A}_{\text{in}}\right]. \tag{27}$$

Similarly, $(\hat{\mathbf{U}}_1^{\text{gate}})^{\top}\mathbf{A}_{\text{in}}$ is computed once instead of K times. Thus, each expert then combines its gate and up outputs:

$$\mathbf{a}_i = \phi\big(\mathbf{W}_i^{\text{gate}}\mathbf{A}_{\text{in}}\big) \odot \big(\mathbf{W}_i^{\text{up}}\mathbf{A}_{\text{in}}\big). \tag{28}$$

(2) Down projection. The down projection $\mathbf{W}_i^{\text{down}}$ operates on the expert-specific activation $\mathbf{a}_i$:

$$\mathbf{W}_i^{\text{down}}\mathbf{a}_i = \hat{\mathbf{U}}_2^{\text{down}}\,\widetilde{\mathbf{G}}_i^{\text{down}}\left[\big(\hat{\mathbf{U}}_1^{\text{down}}\big)^{\top}\mathbf{a}_i\right], \tag{29}$$

where both $\hat{\mathbf{U}}_1^{\text{down}}$ and $\hat{\mathbf{U}}_2^{\text{down}}$ are shared. Although the input projection $\big(\hat{\mathbf{U}}_1^{\text{down}}\big)^{\top}\mathbf{a}_i$ must be computed per expert, the shared output factor allows the final output to be evaluated as:

$$\mathbf{A}_{\text{out}} = \hat{\mathbf{U}}_2^{\text{down}}\Big[\sum_{i\in\mathcal{E}}\widehat{\pi}_i\,\widetilde{\mathbf{G}}_i^{\text{down}}\,\big(\hat{\mathbf{U}}_1^{\text{down}}\big)^{\top}\mathbf{a}_i\Big], \tag{30}$$

thereby reducing K output projections to a single one.

A.4 Rationale for Candidate-restricted Routing

In the SDAR-30B-A3B-Chat-b32 evaluation on GSM8K and MBPP calibration prompts, the average pairwise hidden-state cosine similarity of cold tokens is 0.4904, higher than the 0.2986 observed for hot tokens. This indicates that cold tokens are more directionally concentrated in hidden-state space, which is consistent with their more stable expert routing preferences. We formalize this property as follows.

Let $\mathbf{h}_i$ be the hidden state of the i-th token and define its L_2-normalized representation as:

$$\mathbf{x}_i = \frac{\mathbf{h}_i}{\|\mathbf{h}_i\|_2}, \qquad \|\mathbf{x}_i\|_2 = 1. \tag{31}$$

For a token set of size N, its average pairwise cosine similarity is:

$$S = \frac{1}{N(N-1)}\sum_{i\neq j}\mathbf{x}_i^{\top}\mathbf{x}_j. \tag{32}$$

Let $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$. The average squared distance to the mean satisfies:

$$\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \bar{\mathbf{x}}\|_2^2 = \frac{N-1}{N}(1-S). \tag{33}$$

Equivalently, defining the normalized directional dispersion as:

$$D = \frac{N}{N-1}\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \bar{\mathbf{x}}\|_2^2\,, \tag{34}$$

we obtain$D = 1 - S$. Therefore, the higher pairwise cosine similarity observed for cold tokens directly implies lower directional dispersion,

$$D_{\text{cold}} < D_{\text{hot}}. \tag{35}$$

This directional concentration is consistent with the routing behavior observed in our experiments: cold tokens exhibit more similar routing probability distributions and larger Top-K expert overlap than hot tokens. Consequently, the centroid of cold-token hidden states provides a more representative basis for constructing the candidate expert set, motivating candidate-restricted routing for cold tokens.

Table 6: Architectural configurations of the evaluated MoE-based diffusion language models.

Model	Layers	MoE Layers	Routed Experts	Experts/Token
SDAR-30B-A3B-Chat-b32	48	48	128	8
LLaDA2.0-mini	20	19	256	8

Table 7: Generation and diffusion decoding configurations used in our experiments.

Configuration	SDAR-30B-A3B-Chat-b32	LLaDA2.0-mini
Generation length	4096	4096
Block length	32	32
Denoising steps	32	32
Confidence threshold	0.95	0.95
Temperature	0.0	0.0
Batch size	1	1

Table 8: Hyperparameter settings of ITC-MoE.

λ	M	Compensation Rank	τ_h	L_h
0.1	48	11	0.7	3

B Backbone Model Details

The architectural configurations of the evaluated MoE-based diffusion language models and the corresponding generation and diffusion decoding settings are summarized in Tables 6 and 7, respectively. The hyperparameter settings of ITC-MoE are summarized in Table 8. Our method is implemented on top of the TEAM framework, upon which the proposed Importance-guided Adaptive Tucker Compression and Token-aware Compensation and Routing modules are further developed. For the Compensation Rank, a larger value generally yields better performance and we set it to 11 to balance performance and efficiency. L_h follows the default setting in TEAM.

Table 9: Effect of λ.

λ	MultiArith ↑	SingleQ ↑
0.0	98.33	93.58
0.1	99.17	97.25
0.2	98.83	95.41
0.3	98.67	94.50

Table 10: Sensitivity to M.

M	SingleOp (%)	GSM8K (%)
16	93.24	85.52
32	94.48	86.96
48	95.37	87.87
64	95.37	88.02

C Additional ablations

Effect of the Compensation Regularization Coefficient. We investigate the sensitivity of the token-aware compensation module to the regularization coefficient λ in Eq. 12. As shown in Table 9, under a 10% compression ratio on SDAR, $\lambda = 0.1$ consistently achieves the best performance on both MultiArith and SingleQ. A smaller λ provides insufficient regularization and may lead to over-compensation, whereas a larger λ overly constrains the compensation module and limits its ability to correct compression-induced errors. Therefore, we set $\lambda = 0.1$ as the default value in all experiments.

Effect of Candidate Set Size M. We further investigate the effect of the candidate set size M on model performance. As shown in Table 10, the performance on SingleOp and GSM8K generally improves as M increases. When M reaches 48, the performance gains begin to saturate. Therefore, we set $M = 48$ as the default configuration to balance model performance and routing efficiency.