# VIDEOEDITOR-MOE: SCALING COMPLEX VIDEO EDITING WITH MIXTURE OF EXPERTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Complex video editing requires understanding complex instructions while jointly fulfilling multiple editing requests. Meeting these demands through dense model scaling substantially increases computational cost. We introduce VideoEditor-MoE, the first open-source mixture-of-experts framework for video editing. Beyond sparse MLP-MoE layers, MoE-Attention expands attention capacity through routed key–value experts while preserving global interactions among video tokens. Initialized from a 5B video editing model, VideoEditor-MoE scales to 14B and 56B parameters, with approximately 8B parameters activated per token in both variants. Comparisons with existing methods and two complementary baselines, Explicit-MoE and MLP-MoE, assess its benefits over explicit expert guidance and MLP-only expert architectures. Further analyses demonstrate that all experts remain active and reveal that learned routing responds more strongly to sample content and diffusion noise levels than to human-defined editing categories. Spatial analysis further indicates sensitivity to object-level semantic features. We also identify decoupled weight decay as a strategy for stabilizing expert scaling.

## 1 INTRODUCTION

A complex video editing request succeeds only when every requested change is realized on the targeted objects. Recent models support diverse editing functionality, including object addition, removal, and appearance changes, style changes and so on (Lin et al., 2026; Wei et al., 2026; Liu et al., 2026; Jiang et al., 2025). Yet, a real requests often combine several operations across objects or apply multiple changes to the same object. *"Turning a car blue while replacing the sky with a sunset"*, for example, requires both edits to persist as the scene evolves. A visually plausible result can still leave the request incomplete if one change is omitted or only weakly expressed. Complex editing therefore demands both reliable instruction coverage and complete visual modification, together with preservation of unrelated content and temporal coherence.

Increasing model capacity is a natural direction for supporting these complex instructions. Yet, scaling a dense video model increases computation throughout the network. Mixture-of-experts (MoE) architectures offer a way to expand model capacity while activating only a small subset of experts for each token (Jiang et al., 2024; Dai et al., 2024). To this end, scaling video editing models through MoE provides a practical approach to balancing computational cost and model capacity. However, conventional MLP-MoE expands token-wise transformations while leaving the attention architecture unchanged. Complex editing also requires the exchange of semantic information across regions and frames to coordinate multiple modifications. This motivates extending expert capacity to attention, allowing MoE semantic representations to participate in global token interactions and support coherent editing across space and time.

We introduce VideoEditor-MoE, the first open-source MoE framework for video editing, supporting complex instructions with multiple editing requirements. VideoEditor-MoE incorporates both MoE-MLP and MoE-Attention to expand model capacity beyond conventional dense architectures. MoE-Attention combines shared queries with routed, paired key–value experts. Each token contributes multiple key–value representations, and every query attends to the routed key–value pairs from all tokens. This design expands attention capacity while preserving global token interactions. Together with sparse MLP layers, we develop two model variants, 14B-A8B and 56B-A8B, both with approximately 8B activated DiT parameters per token.



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.

In addition, two baselines are developed to provide complementary comparisons. Explicitly Guided MoE selects experts according to predefined editing categories that reflect an intuitive human understanding of editing tasks, testing whether such task divisions align with effective expert allocation. MLP-MoE expands capacity through MLP experts alone, serving as an ablation baseline for assessing the contribution of MoE-Attention. Beyond editing performance, we conduct detailed analyses of expert utilization, routing behavior, and scaling stability. All experts remain active during inference, and learned expert allocation responds more strongly to sample content and diffusion noise states than to human-defined editing categories. To further understand the content dependence, spatial analysis examines which visual features attract expert routing responses and suggests sensitivity to object-level semantic features. Finally, scaling analysis links expert-weight shrinkage under coupled regularization to suppressed branch outputs and identifies decoupled weight decay as a practical strategy for stabilizing expert scaling.

Our contributions are:

- We highlight complex instruction-based video editing as a practical setting that requires jointly satisfying multiple modifications to a single object or coordinating edits across multiple objects, motivating greater model capacity for compositional editing.
- We introduce VideoEditor-MoE, extending conventional MoE-MLP with MoE-Attention to expand sparse expert capacity into attention. This design expands model capacity and enables a more comprehensive MoE architecture for video generation.
- We demonstrate that all experts remain active and reveal that learned routing responds more strongly to sample content and diffusion noise levels than to human-defined editing categories. Spatial analysis further indicates sensitivity to object-level semantic features.
- We will publicly release both VideoEditor-MoE variants, 14B-A8B and 56B-A8B, together with two complementary baselines, to facilitate further research on MoE architectures for video generation and editing.

## 2 Related Work

### 2.1 Instruction based Video Editing

Instruction-based video editing aims to modify a source video according to natural-language instructions. Recent methods (MiniMax, 2026; Liu et al., 2026; Wei et al., 2026; Lin et al., 2026; Jiang et al., 2025) have demonstrated promising editing results, using either text instructions alone or additional guidance such as masks and reference inputs. These advances enable a range of editing operations, but handling complex instructions that combine multiple editing requirements remains challenging.

In real-world scenarios, users may request edits to multiple objects, multiple modifications to a single object, or a combination of both. Such instructions require models to associate each modification with the correct target and coordinate multiple edits while maintaining temporal consistency and preserving unrelated content. Existing methods may omit some requested edits or only partially realize the intended changes. Meeting these demands places greater requirements on model capacity, motivating the exploration of larger video editing models. However, scaling dense models entails substantial increases in computational cost. This motivates us to explore sparse MoE architectures as a means of expanding model capacity while limiting the growth in computation cost.

### 2.2 Mixture-of-Experts

#### 2.2.1 Preliminary

Mixture-of-experts (MoE) architectures enable model capacity to scale through sparse expert activation. Two representative designs in large language models are Mixtral (Jiang et al., 2024) and DeepSeekMoE (Dai et al., 2024). Mixtral replaces dense feed-forward networks (FFNs) with multiple expert networks and a token-level router. Given an input token representation $x$, the MoE module

computes

$$F_{\text{Mixtral}}(x) = \sum_{i=0}^{n-1} G(x)_i \, E_i(x), \tag{1}$$

where $n$ is the number of experts, $E_i$ denotes the $i$-th expert, and $G(x)_i$ is its routing weight. The router selects the top-$k$ experts for each token and assigns zero weights to the remaining experts.

DeepSeekMoE introduces fine-grained expert segmentation and shared expert isolation to improve parameter efficiency. Specifically, it reduces each expert's FFN intermediate dimension by a factor of $m$, yielding $mn$ smaller experts, and designates $k_s$ of them as shared experts that process every token. Using the same notation for expert outputs and sparse routing weights, its MoE module can be expressed as

$$F_{\text{DeepSeek}}(x) = \sum_{i=0}^{k_s-1} E_i(x) + \sum_{i=k_s}^{mn-1} G(x)_i \, E_i(x), \tag{2}$$

where the router selects $mk - k_s$ experts from the $mn - k_s$ routed experts. Together with the $k_s$ shared experts, this configuration activates $mk$ fine-grained experts per token, preserving approximately the same expert computation budget as activating $k$ original-sized experts. Fine-grained segmentation enables more flexible expert combinations, while shared experts aim to capture common knowledge and reduce redundancy among routed experts. Here, $G(x)_i$ denotes the routing weight under each architecture's respective gating rule, and both formulations omit residual connections.

#### 2.2.2 Related Methods

MoE architectures have been explored in visual generation to expand model capacity and adapt computation to different inputs and denoising stages. DiT-MoE (Fei et al., 2024) introduces shared and routed MLP experts into diffusion transformers, increasing total capacity while limiting the computation activated for each token. Switch-DiT (Park et al., 2024) uses timestep-conditioned routing to mitigate interference between denoising tasks at different noise levels. Diff-MoE (Shi et al., 2025) combines expert-specific timestep conditioning with spatially adaptive expert allocation, allowing expert computation to adapt to both token content and the generative stage.

MoE has also been explored in video generation field. Wan2.2 (Wan et al., 2025) assigns high- and low-noise denoising stages to two specialized dense experts, separating coarse content formation from detail refinement. This design allocates computation at the model level according to the noise stage. Mamoda2.5 (Shi et al., 2026) instead incorporates fine-grained routed experts and a shared expert into a unified multimodal framework, expanding capacity for understanding, generation, and editing with sparse activation. LingBot-Video (Ma et al., 2026) explores sparse MoE scaling for embodied video pretraining, targeting physical dynamics and action-conditioned generation.

These advances motivate an open-source MoE framework tailored to the demands of complex video editing, where multiple editing requirements call for greater model capacity. We introduce the first open-source MoE-based video editing model, with pretrained video editing models providing a practical starting point for scaling. A Mixtral-style MoE design enables expert initialization from existing models, while MoE-Attention further expands attention capacity to support interactions among video tokens. Together, these designs offer a practical path to scaling complex video editing capabilities while retaining the knowledge acquired by pretrained models.

## 3 Method

In this section, we first introduce two additional baselines: (1) Explicitly Guided MoE and (2) MLP-MoE, to examine MoE routing preferences and demonstrate the benefits of our MoE-Attention design, respectively. We then present the design of VideoEditor-MoE. All the baselines and our proposed VideoEditor-MoE are built on the same 5B 30-layer base model.

### 3.1 Baselines

**Explicitly Guided MoE.** This baseline guides expert selection using five categories that reflect an intuitive human understanding of editing tasks: object addition, object removal, object replacement,

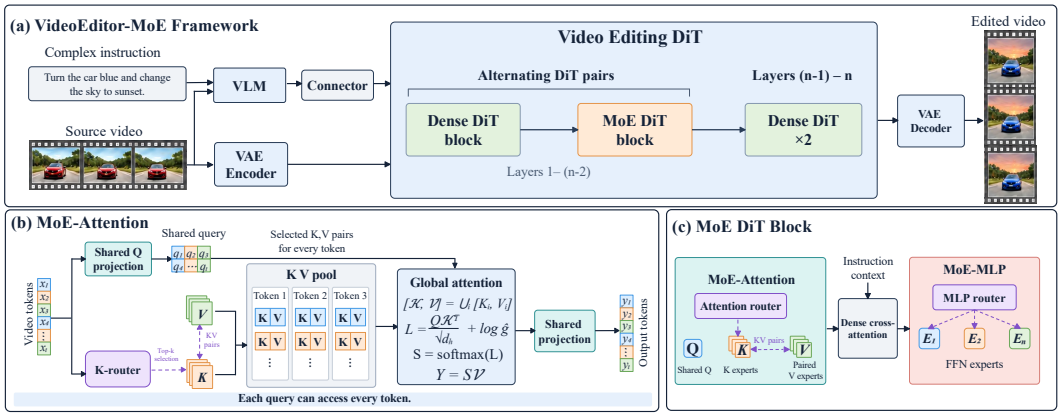



Figure 1: **VideoEditor-MoE.** (a) Dense and MoE DiT blocks alternate through 28 layers, followed by two dense blocks. (b) Shared queries attend to routed key–value pairs from all video tokens. (c) Each MoE block combines MoE-Attention, dense instruction cross-attention, and MoE-MLP.

style transfer, and background replacement. As shown in left side of Appendix Figure 6, a main editing branch integrates features from task-specific experts, each fine-tuned on one category with low-rank adaptation (LoRA) (Hu et al., 2021).

A VLM decomposes the complex instruction into individual edits in advance. The main branch receives the complete instruction, while each expert receives the instruction for its assigned edit. A router, pretrained on an editing-category classification task, predicts each individual edit's category from VLM features. Fusion layers are inserted into the first half of the main branch. These layers perform semi-attention: main-branch queries attend to main-branch tokens and expert key–value features, with spatial offsets applied to the RoPE (Su et al., 2024) embeddings of the expert features.

**MLP-MoE Baseline.** This baseline processes the complete instruction without editing-category supervision. It expands MLP capacity while keeping attention dense, serving as an ablation baseline for MoE-Attention. For the DiT side, MLPs at odd blocks $\mathcal{M} = \{2, 4, \ldots, 28\}$ are replaced with MoE layers while other blocks remain dense. Each MoE layer contains $N$ full-width MLP experts and a token-level router, with no shared expert. For MoE blocks, a video-token $x$ is represented by:

$$p(x) = \mathrm{softmax}(xW_g), \qquad G(x)_i = \frac{p(x)_i \, \mathbb{I}[i \in \mathcal{I}_k(x)]}{\sum_{j \in \mathcal{I}_k(x)} p(x)_j}, \qquad F_{\mathrm{MoE}}(x) = \sum_{i=1}^{N} G(x)_i E_i(x), \quad (3)$$

where $W_g$ is the router matrix, $\mathcal{I}_k(x)$ contains the top-$k$ expert indices, and $\mathbb{I}$ is the indicator function. The selected weights $G(x)_i$ sum to one, while unselected experts receive zero weight. The final output combines the selected MLP expert outputs $E_i(x)$. The overall framework is shown in Appendix Figure 6 right side. Here, we set $N = 12$ and choose top-3 routing, yielding a **19B-A8B** MoE-MLP baseline.

### 3.2 VideoEditor-MoE with MoE-Attention

MLP-MoE transforms individual tokens, while attention exchanges semantic information across regions and frames. VideoEditor-MoE extends the MoE-MLP baseline with paired key–value experts in self-attention mechanism within MoE blocks. As shown in Figure 1(b), query and output projections remain shared. Attention and MLP experts use separate routers, and instruction cross-attention remains dense.

For $T$ video-token representations $X \in \mathbb{R}^{T \times d}$, the attention router applies the top-$k$ rule in Equation 3. Each selected index $i$ chooses both a key projection $W_K^{(i)}$ and its paired value projection $W_V^{(i)}$. Let $X_i$ contain the tokens that select expert $i$; a token appears in $k$ such groups. For one attention head of dimension $d_h$,

$$Q = XW_Q, \qquad K_i = X_i W_K^{(i)}, \qquad V_i = X_i W_V^{(i)}, \quad (4)$$

where $W_Q, W_K^{(i)}, W_V^{(i)} \in \mathbb{R}^{d \times d_h}$. Biases, query/key normalization, and RoPE are omitted from the equations for clarity. They follow the base model, and each key representation retains its source token's position.

Following Figure 1(b), all selected key–value pairs are written as

$$[\mathcal{K}, \mathcal{V}] = \bigcup_i [K_i, V_i], \qquad \mathcal{K}, \mathcal{V} \in \mathbb{R}^{kT \times d_h}. \tag{5}$$

Here, $\bigcup$ collects the selected key–value pairs from all tokens, keeping each key together with its corresponding value. Each token contributes $k$ pairs, giving $kT$ pairs in total. The pair produced by expert $i$ for token $x_t$ carries the normalized routing weight $G^{\text{attn}}(x_t)_i$, computed using Equation 3. These weights form the vector $\hat{g} \in \mathbb{R}^{1 \times kT}$, ordered to match the corresponding pairs in $\mathcal{K}, \mathcal{V}$. Every query attends to all $kT$ routed pairs through a single attention operation:

$$L = \frac{Q\mathcal{K}^\top}{\sqrt{d_h}} + \log \hat{g}, \qquad S = \text{softmax}(L), \qquad Y = S\mathcal{V}. \tag{6}$$

The term $\log \hat{g}$ adds the logarithm of each pair's routing weight to every query's attention scores. It favors pairs preferred by the router, while $Q\mathcal{K}^\top$ measures their relevance to each query. Softmax normalizes each row over all $kT$ pairs, and $Y$ combines their values using these attention weights. The head outputs pass through the shared output projection.

MoE-Attention produces multiple expert-specific key–value representations for each token, enabling richer semantic modeling. Every query attends to the routed key–value pairs from all tokens, preserving global interactions across regions and frames.

**Training objective.** Training combines flow-matching reconstruction, temporal-difference supervision, and per-sample expert load balancing as in Appendices C and A.

## 4 Experiments and Analysis

We first compare edit execution, fulfillment, and output quality. We then examine (1) **expert utilization** and (2) **routing sensitivity** to diffusion noise levels and sample content. Building on this content sensitivity, we further investigate (3) whether routing responds to global color distributions or object-level semantic features, with spatial analysis providing evidence of **sensitivity to semantic features**. Finally, we analyze (4) the **instability** introduced by coupled $L_2$ regularization during model scaling and how decoupled weight decay stabilizes training.

### 4.1 Quantitative Results

We evaluate edited videos along three dimensions. **Edit Execution Rate** measures the fraction of requested edits successfully performed; **Edit Fulfillment** measures how completely and accurately each change is realized, including unsuccessful edits; and **Video Quality** assesses visual quality, temporal consistency, preservation of unedited content, and integration of the edits. The evaluation protocol and prompt templates are provided in Appendix D and Figures 7, 8, and 9.

Table 1: Quantitative comparison. VideoEditor-MoE-56B-A8B ranks first and 14B-A8B ranks second across all three metrics. Evaluation scope and protocol appear in Appendix D.

| Metrics\Model | Kiwi-Edit | MiniMax-H3 | CoinVE-Edit | Explicit-MoE | MLP-MoE (19B-A8B) | Ours (14B-A8B) | Ours (56B-A8B) |
|---|---|---|---|---|---|---|---|
| Edit Execution Rate ↑ | 73.60 | 76.82 | 87.99 | 81.67 | 88.45 | 90.15 | **94.70** |
| Edit Fulfillment ↑ | 73.11 | 76.06 | 88.23 | 81.69 | 88.78 | 90.72 | **94.70** |
| Video Quality ↑ | 61.36 | 61.36 | 69.89 | 69.89 | 68.75 | 71.59 | **77.27** |

As shown in Table 1, VideoEditor-MoE-56B-A8B achieves the highest scores across all three metrics, followed by the 14B-A8B variant. Both variants outperform the five baselines in edit execution, edit fulfillment, and video quality. As an ablation baseline, MLP-MoE baseline ranks third on both editing metrics, while its gap to VideoEditor-MoE highlights the importance of MoE-Attention.

### 4.2 Qualitative Results

As shown in Figure 2, we compare both VideoEditor-MoE variants with five baselines on complex instructions. For compact presentation, only the middle frame of each video is shown, and the displayed instructions are manually decomposed and simplified for readability.

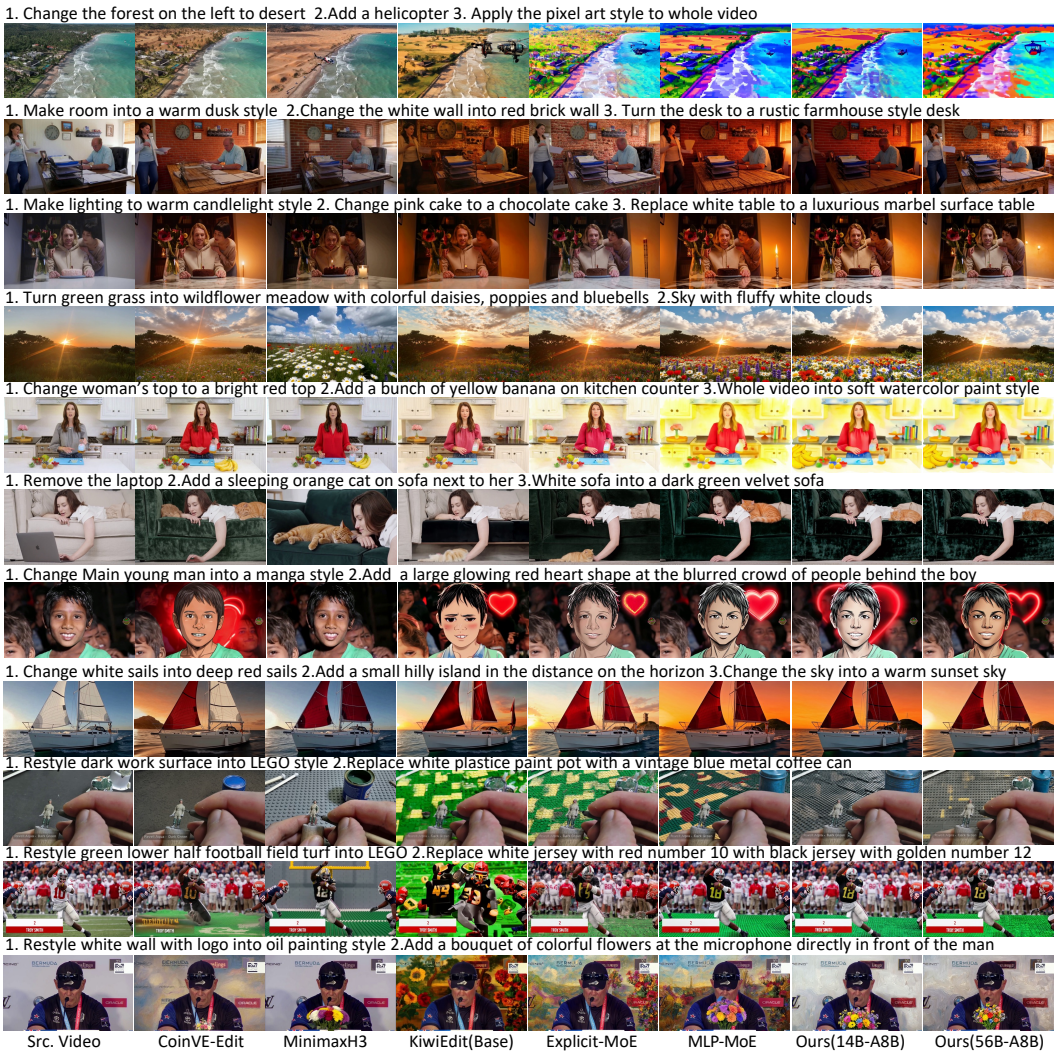



Figure 2: **Qualitative comparisons under complex instructions.** Each row shows the source video and outputs from five baselines and both VideoEditor-MoE variants. Middle frames are shown, with editing instructions summarized above each row.

VideoEditor-MoE jointly realizes multiple requested changes within the same scene. In the third row of Figure 2, our variants capture the requested candlelit atmosphere through warm scene lighting, while some baselines emphasize inserting visible candles. In the fifth row, both variants combine soft watercolor stylization with the addition of bananas. In the sixth row, VideoEditor-MoE and MLP-MoE place the cat beside the woman while preserving the scene structure. Most of baselines output omit requested changes or express them less clearly, illustrating the challenge of jointly fulfilling complex instructions.

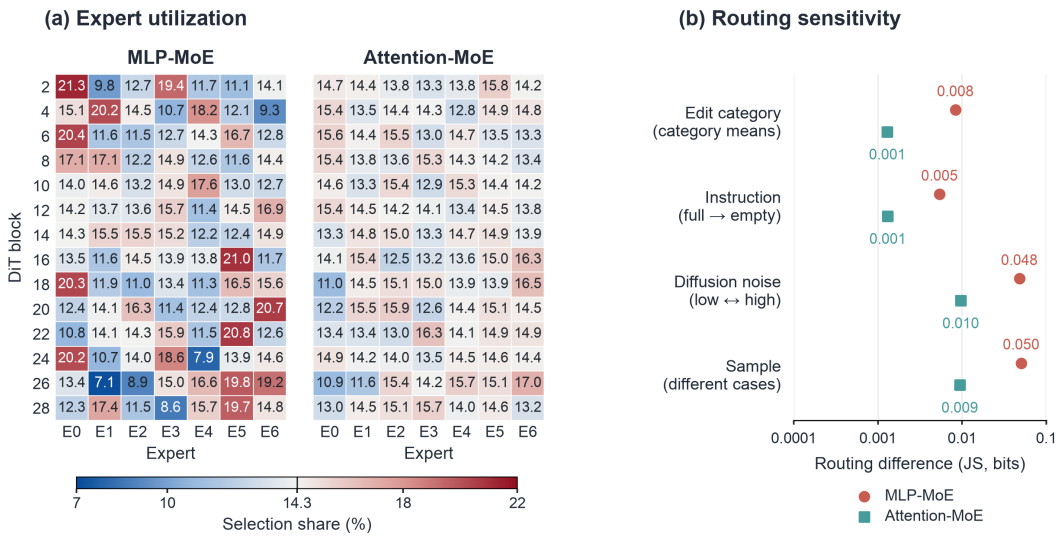



Figure 3: **Expert utilization and input-dependent allocation.** (a) Mean selection shares over 20 samples and three noise levels. Each row sums to $100\%$, with uniform allocation at approximately $14.3\%$. (b) Mean Jensen–Shannon (JS) divergence under the comparisons defined in the text. Larger values indicate greater allocation changes.

### 4.3 Expert Analysis

We examine whether experts remain active and adapt to editing inputs. We conduct experiments on the 14B-A8B variant using 20 training samples at noise levels $\sigma \in 0.355, 0.833, 0.978$. Additional instruction and noise controls bring total to 110 forward probes, as in Appendix E.2.

**All experts remain active.** Each of $T$ video tokens selects $k$ experts, yielding $kT$ selections. Expert $i$'s selection share is $s_i = \frac{\sum_{t=1}^{T} \mathbb{I}[i \in \mathcal{I}_k(x_t)]}{kT}$, the fraction of selections assigned to it, where $\mathcal{I}_k(x_t)$ contains the experts selected for token $x_t$. Larger shares indicate more frequent expert use. As shown in Figure 3(a), even the least-used experts receive mean selection shares of $7.1\%$ in MLP-MoE and $10.9\%$ in MoE-Attention. Moreover, **all 196 layer–module–expert units are selected in every probe**, confirming that no expert remains inactive.

**Routing is more sensitive to content and noise level.** We exam the router sensitivity to human predefined editing category, instruction, diffusion noise level and sample content as shown in Figure 3. For two samples or conditions $a$ and $b$, we form $\mathbf{p} = [s_1^{(a)}, \ldots, s_N^{(a)}]$ and $\mathbf{q} = [s_1^{(b)}, \ldots, s_N^{(b)}]$ for the same layer and router, and measure their Jensen–Shannon (JS) divergence:

$$\mathrm{JS}(\mathbf{p}, \mathbf{q}) = \tfrac{1}{2} \sum_i p_i \log_2 \frac{p_i}{m_i} + \tfrac{1}{2} \sum_i q_i \log_2 \frac{q_i}{m_i}, \qquad \mathbf{m} = (\mathbf{p} + \mathbf{q})/2. \tag{7}$$

JS is zero for identical distributions and increases with differences in expert selection shares, in bits. Sample comparisons use all $\binom{20}{2} = 190$ unordered pairs, without a fixed reference sample; category comparisons use all ten pairs of category-mean distributions. Matched controls compare full versus empty instructions or low versus high noise for each sample. Each point in Figure 3(b) reports the average JS divergence across the corresponding comparisons and 14 MoE layers, separately for MLP and attention. Details are provided in Appendix E.2.

As shown in Figure 3(b), MoE-MLP allocation changes more across samples ($0.0503$) and noise states ($0.0479$) than between editing-category means ($0.0083$) or after instruction removal ($0.0054$). MoE-Attention follows the same trend with smaller changes. Expert selection shares are therefore more sensitive to sample content and denoising state, and comparatively less sensitive to broad editing categories and instruction. The weak category-level routing sensitivity suggests that learned MoE routing is not primarily governed by the five predefined editing categories used in Explicitly

Guided MoE. We next investigate whether its sensitivity to sample content reflects global color distributions or object-level semantic features.

**Content-sensitive routing follows object motion.** As shown in Figure 4, a representative content-sensitive MLP expert's routing responses follow the moving person. Regions receiving higher routing weights shift with the object, while responses at its previously occupied location become sparser. This spatial correspondence complements the sample-level differences in Figure 3(b), suggesting that expert allocation adapts to object-associated content rather than global color distributions.

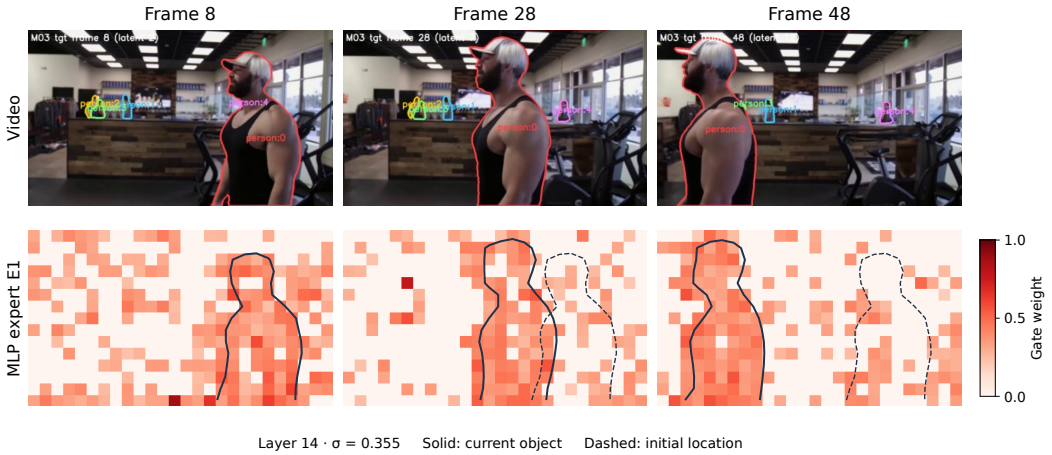



Figure 4: **Object-semantic expert routing.** Top: original video. Bottom: routing weights of a representative MLP expert sensitive to this sample's content. The high routing weight lies in the person region, which demonstrate its sensitive to semantic object.

### 4.4 Scaling Stability

**Decoupled weight decay stabilizes expert scaling.** When scaling VideoEditor-MoE to 56B-A8B, we observe optimization instability with Adam (Kingma & Ba, 2014) and coupled $L_2$ regularization: expert weights shrink disproportionately and their outputs become suppressed. As shown in Figure 5(a), MoE branches produce much smaller outputs than adjacent dense blocks, especially for MLP.

Switching to AdamW (Loshchilov & Hutter, 2017) yields comparable dense and MoE output magnitudes in Figure 5(b). The median MoE-to-dense output ratio increases from $0.005$ to $0.97$ for MLP and from $0.12$ to $0.93$ for attention, supporting more stable expert contributions at the larger scale.

Figure 5 measures relative branch output as $c_\ell = \mathrm{RMS}(\Delta h_\ell)/\,\mathrm{RMS}(h_\ell)$, where $h_\ell^+ = h_\ell + \Delta h_\ell$ and $\mathrm{RMS}(x) = \sqrt{n^{-1}\sum_{j=1}^{n} x_j^2}$ is the root mean square of $n$ elements. Here, $h_\ell$ is the branch input and $\Delta h_\ell$ is the final output added to it, including expert fusion for MoE branches and DiT output modulation. Each cell is the median over five samples; each MoE block is compared with the preceding dense block in the same model. Varying input noise level $\sigma$ from low to high tests whether the disparity persists across denoising conditions, rather than at only one noise level.

The key distinction is how regularization enters the update. Adam adds the $L_2$ term $\lambda W$ to the task gradient before computing its adaptive updates, where $W$ denotes the weights and $\lambda$ the regularization coefficient. When expert gradients are small, this term can **dominate the optimizer's moment estimates**. Adaptive normalization can then produce shrinkage steps that do not decrease proportionally with weight magnitude, eroding pretrained expert weights. Scaling can amplify this risk: enlarging the expert pool at fixed top-$k$ can spread task gradients across more experts, while regularization continues to act on their weights.

AdamW instead computes adaptive updates from task gradients alone and applies weight decay separately as $-\eta\lambda W$, where $\eta$ is the learning rate. The decay step therefore becomes smaller as the

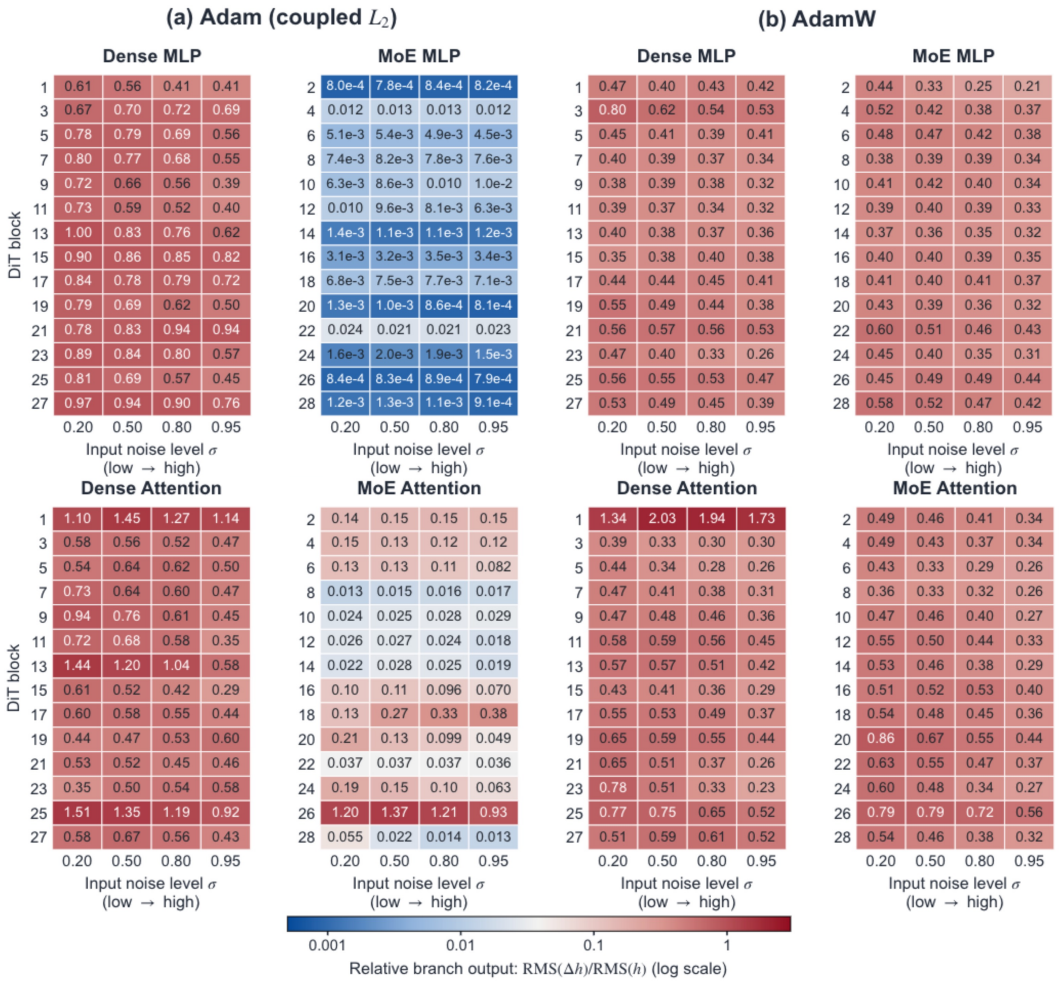



Figure 5: **Optimization stability at 56B-A8B.** MLP (top) and attention (bottom) outputs under (a) Adam and (b) AdamW. Cells report median relative branch output $c_\ell$ over five samples; columns increase input noise level $\sigma$. The high output of MoE-MLP and MoE-Attention demonstrate optimized under AdamW exihbits similar computation behavior compared with dense layers.

weights shrink, without being adaptively normalized. This separation helps preserve expert scales during expansion, consistent with the stronger expert outputs in Figure 5(b). Diagnostic settings, weight statistics, and training histories are provided in Appendix E.1.

## 5 Conclusions

We present VideoEditor-MoE, the first open-source MoE framework for video editing under complex instructions. By combining MoE-MLP with MoE-Attention, the framework scales to 14B-A8B and 56B-A8B variants while preserving global token interactions. Our analyses demonstrate active utilization of all experts and reveal that routing responds more strongly to sample content and diffusion noise levels than to human-defined editing categories, with spatial evidence of sensitivity to object-level semantic features. We further show how coupled $L_2$ regularization can undermine optimization stability during model scaling. Together, these contributions provide an open-source foundation for further research on MoE architectures for video generation and editing.

## 6 AI USE STATEMENT

We use AI tools to assist with language revision and the development of infrastructure for large-scale model training.

## REFERENCES


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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. URL https://arxiv.org/abs/2401.04088.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17191–17202. IEEE, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yiqi Lin, Guoqiang Liang, Ziyun Zeng, Zechen Bai, Yanzhe Chen, and Mike Zheng Shou. Kiwi-edit: Versatile video editing via instruction and reference guidance. *arXiv preprint arXiv:2603.02175*, 2026.

Jialun Liu, Tian Li, Xiao Cao, Yukuo Ma, Gonghu Shang, Haibin Huang, Chi Zhang, Xiangzhen Chang, Zhiyong Huang, Jiakui Hu, et al. Tele-omni: a unified multimodal framework for video generation and editing. *arXiv preprint arXiv:2602.09609*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Shuailei Ma, Jiaqi Liao, Xinyang Wang, Jingjing Wang, Chaoran Feng, Zijing Hu, Chong Bao, Zichen Xi, Yuqi Gan, Weisen Wang, Yanhong Zeng, Qin Zhao, Zifan Shi, Wei Wu, Hao Ouyang, Qiuyu Wang, Shangzhan Zhang, Jiahao Shao, Yipengjing Sun, Liangxiao Hu, Lunke Pan, Nan Xue, Kecheng Zheng, Yinghao Xu, Xing Zhu, Yujun Shen, and Ka Leong Cheng. Scaling mixture-of-experts video pretraining for embodied intelligence. *arXiv preprint arXiv:2607.07675*, 2026.

MiniMax. MiniMax H3: An open model breaking the boundaries between tasks and modalities. https://www.minimax.io/blog/minimax-h3, 2026.

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.

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

Yangming Shi, Shixiang Zhu, Tao Shen, Zhimiao Yu, Dengsheng Chen, Taicai Chen, Yunfei Yang, Juan Zhou, Chen Cheng, Liang Ma, et al. Mamoda2. 5: Enhancing unified multimodal model with dit-moe. *arXiv preprint arXiv:2605.02641*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Cong Wei, Quande Liu, Zixuan Ye, Qiulin Wang, Xintao Wang, Pengfei Wan, Kun Gai, and Wenhu Chen. Univideo: Unified understanding, generation, and editing for videos. In *International Conference on Learning Representations*, volume 2026, pp. 113905–113933, 2026.

# A Implementation Details

**Model and training.** VideoEditor-MoE is initialized from the pretrained 5B video editing DiT. The backbone contains 30 blocks with hidden dimension 3072, MLP intermediate dimension 14336, and 24 attention heads of dimension 128. Training clips contain 49 frames at $480 \times 832$ resolution. VAE encoding followed by $(1, 2, 2)$ patchification produces $13 \times 15 \times 26 = 5{,}070$ video tokens per clip. The full DiT is fine-tuned, while the VLM and VAE remain frozen. The 14B-A8B model is trained for 18,630 updates on 24 NVIDIA H100 GPUs with 80 GB memory each, using DeepSpeed ZeRO-3, bfloat16 computation, and activation checkpointing. Router scores and reconstruction losses are computed in float32. Training timesteps are sampled uniformly from a 1,000-step schedule with shift 5, without additional noise-level loss weighting. Instructions are replaced with empty text with probability 0.1 while source-video conditioning is retained.

**Optimization and inference.** AdamW uses separate learning rates for backbone, expert, and router parameters (Table 2). Expert and router learning rates decrease linearly during the first 2,000 updates and remain constant thereafter. The corrected 56B-A8B training run offloads optimizer states to CPU, uses AdamW updates, and enables ZeRO++ hpZ with partition size 8; the 14B-A8B run uses partition size 1. The corrected 56B run uses five clips per GPU and a global batch size of 120. Benchmark inference for 14B-A8B uses 50 Euler sampling steps with schedule shift 5 and a text-guidance scale of 3 in the selected decoding configuration. Outputs contain 49 frames at $480 \times 832$ resolution and are saved at 16 frames per second. Source videos are processed by taking their first 49 native frames, resizing to cover the target resolution, and center-cropping. The VLM's visual positional-encoding buffer was uninitialized during the 14B training run; evaluation uses corrected initialization of this buffer after training, without changing the trained DiT checkpoint.

Table 2: Training configuration for VideoEditor-MoE-14B-A8B. Learning-rate transitions span the first 2,000 updates.

| Setting | Value |
|---|---|
| Training updates | 18,630 |
| Clips per GPU / global batch size | $8$ / $192$ |
| Backbone learning rate | $10^{-5}$ |
| Expert learning rate | $10^{-4} \rightarrow 10^{-5}$ |
| Router learning rate | $10^{-3} \rightarrow 10^{-4}$ |
| Optimizer / weight decay | AdamW / 0.01 |
| Gradient clipping norm | 1.0 |
| Training schedule steps / shift | $1{,}000$ / $5$ |
| Instruction dropout | 0.1 |
| Motion weight / balancing coefficient | $0.1$ / $10^{-4}$ |
| Residual threshold | 50 |

**Attention implementation and parameter accounting.** Keys and values use the same selected expert indices and routing weights. Query and key normalization follow the base model; each selected key copy receives its original token's RoPE position. The $kT$ key–value candidates remain separate throughout attention. The logarithm in Equation 6 is evaluated as $\log \max(\hat{g}, 10^{-6})$ elementwise. If all selected expert copies produce identical keys and values and the clamp is inactive, their normalized routing weights sum to one, so Equation 6 reduces exactly to the corresponding dense attention. This is a property of the formulation, not an assumption that the actual pretrained initialization sources are identical. The 14B and 56B counts refer to the DiT backbones; including the other components gives approximately 18.5B and 60.5B parameters. Activated parameters are counted per token and do not imply equal total memory use, latency, or attention FLOPs.

## B BASELINE ARCHITECTURES

Figure 6 contrasts explicit task-guided expert composition with token-level MLP routing. Only the main branch produces the final video in Explicitly Guided MoE; expert branches supply features through one-way attention. MLP-MoE uses a single editing backbone with dense attention and sparse MLP layers.

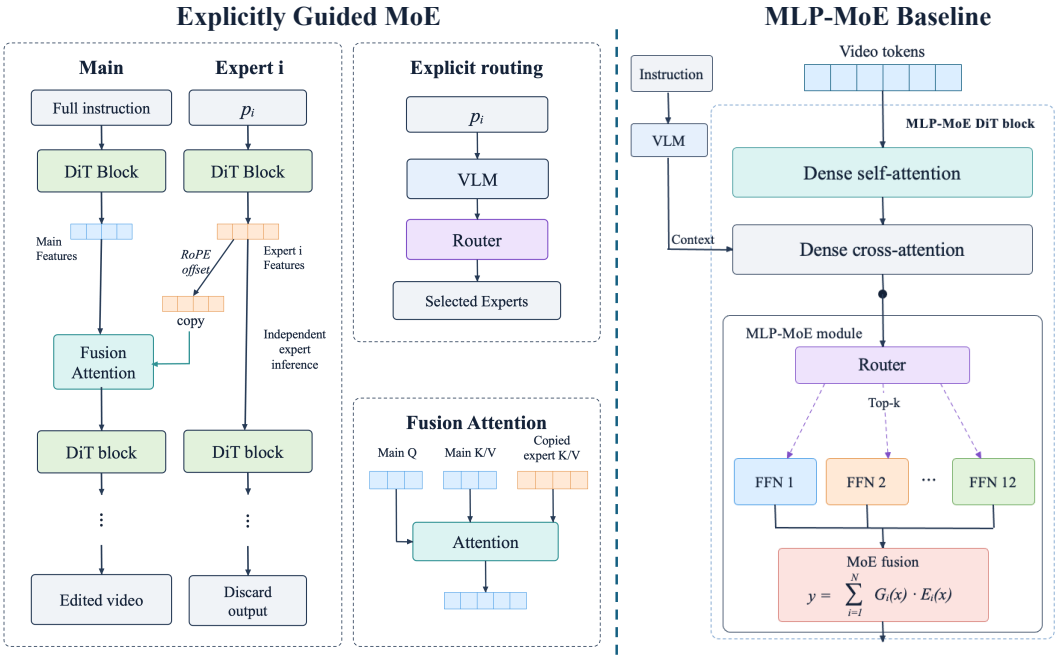



Figure 6: **Two MoE baselines.** Left: Explicitly Guided MoE sends the complete instruction to the main branch and decomposed instructions to selected experts. Fusion layers are injected into the first half of the main branch incorporates expert features. Right: MLP-MoE routes contextualized video tokens to MLP experts while retaining dense attention.

## C TRAINING OBJECTIVE

VideoEditor-MoE combines the base model's flow-matching reconstruction and motion objectives with an expert load-balancing loss. These terms supervise the edited video, encourage accurate temporal changes, and discourage expert underutilization, respectively.

**Reconstruction loss.** Given the target video latent $x_0$ and noise $\epsilon \sim \mathcal{N}(0, I)$, the flow-matching input and target velocity are

$$x_\sigma = (1-\sigma)x_0 + \sigma\epsilon, \qquad v^\star = \epsilon - x_0, \tag{8}$$

where $\sigma \in [0,1]$ follows the training noise schedule. The model predicts $\hat{v} = v_\theta(x_\sigma, c, \sigma)$, conditioned on the source video and complete instruction through $c$. Reconstruction supervision matches the predicted velocity to its target:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}[\rho_\tau(\hat{v} - v^\star)], \qquad \rho_\tau(r) = \frac{1}{D_r}\sum_{a=1}^{D_r} r_a^2\,\mathbf{1}[|r_a| \le \tau], \tag{9}$$

where $D_r$ is the number of elements in residual tensor $r$, and $\mathbf{1}[\cdot]$ is the indicator function. The expectation is over training examples, noise, and noise levels. The elementwise mask excludes residuals larger than $\tau = 50$ to limit the influence of extreme errors; normalization still uses all $D_r$ elements. This objective learns the conditional velocity field used to generate the edited video.

**Motion loss.** To explicitly supervise changes between consecutive latent frames, the motion objective matches temporal differences of the predicted and target velocities:

$$\mathcal{L}_{\text{motion}} = \mathbb{E}[\rho_\tau(\Delta_f\hat{v} - \Delta_f v^\star)], \qquad (\Delta_f u)_f = u_{f+1} - u_f. \tag{10}$$

Here, $f$ indexes latent frames, and the differences are computed between consecutive frames. Penalizing differences in the velocity prediction error encourages the model to reproduce the target's temporal changes.

**Expert load balancing.** Sparse routing can concentrate computation on a small subset of experts. Following DeepSeekMoE (Dai et al., 2024), a balancing objective is computed separately for each sample and each attention or MLP router. For a sample with $T$ tokens, define

$$f_i^{\ell,r} = \frac{N}{kT}\sum_{t=1}^{T}\mathbf{1}\left[i \in \mathcal{I}_k^{\ell,r}(t)\right], \qquad P_i^{\ell,r} = \frac{1}{T}\sum_{t=1}^{T} p_{t,i}^{\ell,r}, \tag{11}$$

where $\ell \in \mathcal{M}$ indexes MoE blocks and $r \in \{\text{attn}, \text{mlp}\}$ identifies the router. The quantity $f_i^{\ell,r}$ measures expert $i$'s selection frequency relative to uniform allocation, while $P_i^{\ell,r}$ is its average router probability. Here, $p_{t,i}^{\ell,r}$ is the probability over all $N$ experts before top-$k$ selection, as defined in Equation 3. The loss sums contributions from both routers across MoE blocks:

$$\mathcal{L}_{\text{bal}} = \sum_{\ell\in\mathcal{M}}\sum_{r\in\{\text{attn},\text{mlp}\}} \mathbb{E}_{\text{sample}}\left[\sum_{i=1}^{N} f_i^{\ell,r} P_i^{\ell,r}\right]. \tag{12}$$

Coupling observed expert loads with differentiable routing probabilities discourages persistent preference for heavily used experts, without prescribing editing-category assignments.

The overall training objective is

$$\mathcal{L} = (1-\lambda_m)\mathcal{L}_{\text{rec}} + \lambda_m\mathcal{L}_{\text{motion}} + \alpha\mathcal{L}_{\text{bal}}, \tag{13}$$

where $\lambda_m$ controls the contribution of temporal supervision and $\alpha$ controls the strength of expert load balancing.

The implementation uses $\lambda_m = 0.1$, $\alpha = 10^{-4}$, and $\tau = 50$.

## D EVALUATION PROTOCOL

**Evaluation scope.** All methods are compared on the same cases using their original fixed inference configurations. This outcome-dependent selection favors VideoEditor-MoE-56B-A8B and characterizes favorable examples rather than overall benchmark performance.

Table 1 reports case-level means multiplied by 100 for all seven methods, including both VideoEditor-MoE variants. Edit Fulfillment includes every requested edit, including unsuccessful edits. Video Quality uses the overall quality score directly, without multiplying by the execution rate; it is a holistic score rather than an arithmetic average of its constituent considerations.

**Scoring.** Evaluation uses Qwen3.6-27B with the instruction and aligned source and edited frames at the beginning, middle, and end (frames 1, 25, and 49). Method identities are not supplied. Each instruction is decomposed into atomic editing requirements. An edit is executed if the intended change is visible on the correct target in at least two of the three frames. Edit Fulfillment scores every requested edit between 0 and 1 according to its completeness and accuracy, using anchors at 0, 0.25, 0.5, 0.75, and 1; unsuccessful edits are limited to at most 0.25. Unverifiable edits receive zero. Execution and fulfillment are judged together, and a separate prompt assesses overall video quality. An unchanged output with no executed edit is capped at 0.25 for quality. The three-frame protocol assesses consistency at sampled times and does not measure all intermediate-frame dynamics.

**Inference configuration.** Evaluated videos have 49 frames at $832 \times 480$ resolution and 16 frames per second. Kiwi-Edit and Explicit-MoE use text guidance scale 1 and scheduler shift 5; MLP-MoE and VideoEditor-MoE-14B-A8B use the archived segmented-guidance configuration with text guidance scale 3 and shift 5. The documented VideoEditor-MoE sampler uses 50 flow-matching Euler steps. CoinVE-Edit and MiniMax-H3 retain their archived default and returned outputs, respectively. The 56B-A8B results also use the archived segmented-guidance variant with shift 5. These are the fixed method-specific configurations used to generate the evaluated videos.

**Prompt templates.** Figures 7, 8, and 9 present the criteria for execution, fulfillment, and quality. The first two panels describe the two scores obtained from the joint edit-evaluation prompt.

# E Additional Analysis Details

## E.1 Scaling Diagnostics

The analysis compares existing 56B training runs using Adam with coupled $L_2$ regularization and AdamW. Both use a regularization coefficient of $0.01$, but differ in batch and restart history; the comparison is therefore a diagnostic study rather than a controlled single-variable ablation. Weight and optimizer-state statistics use checkpoints at trainer step 1,500. The Adam run has an optimizer step count of 917 following a weights-only restart, whereas the AdamW run has 1,500 optimizer steps. Moment-based statistics use the corresponding optimizer counters.

For a tensor $x$ with $n$ elements, $\mathrm{RMS}(x) = \sqrt{n^{-1}\sum_{j=1}^{n} x_j^2}$. Weight scales are measured as $\mathrm{RMS}(W)/\mathrm{RMS}(W_{\mathrm{source}})$ relative to the source checkpoints used to initialize each matrix, rather than a saved step-zero checkpoint. Reported category-level values are medians of per-matrix RMS ratios. In the Adam run, these medians are $1.62\%$ for MLP experts and $52.4\%$ for dense MLP matrices; the median cosine similarity between MLP expert weights and their first-moment estimates is $0.99996$. Dense MLP statistics include only dense MLP weight matrices, excluding biases, normalization parameters, and other backbone modules.

**Noise-level probes.** Activation probes compare the Adam checkpoint at trainer step 1,500 with the AdamW checkpoint at step 1,000. Both are evaluated on the same five cached video samples at $\sigma \in \{0.2015, 0.5009, 0.8, 0.95\}$; the first two values are displayed as $0.20$ and $0.50$ in Figure 5. For each sample, the source and target latents, instruction context, and one Gaussian noise realization $\epsilon$ are fixed across noise levels and runs. The input $z_\sigma = (1-\sigma)z_0 + \sigma\epsilon$ mixes the clean target latent $z_0$ with this noise, and the model receives the corresponding timestep. Varying $\sigma$ changes noise strength, whereas changing the random seed would change the noise realization. These no-gradient BF16 forward passes on two GPUs probe controlled denoising conditions at fixed weights, without following a full generation trajectory. Testing several noise levels assesses whether branch-output suppression persists across the sampled conditions; it does not establish behavior at every possible noise level.

**Branch-output measurement.** For each attention or MLP branch, $h_\ell$ denotes the hidden state immediately before that branch's additive update, and $h_\ell^+$ denotes the state immediately after it. The measured output is

$$h_\ell^+ = h_\ell + \Delta h_\ell, \qquad c_\ell = \frac{\mathrm{RMS}(\Delta h_\ell)}{\mathrm{RMS}(h_\ell)}. \tag{14}$$

Here, $\Delta h_\ell$ names the final output added through the residual connection; it is neither an additional network module nor an optimizer update to the parameters. For MoE-MLP, the module first computes

$$y_{\ell,t} = \sum_{i \in \mathcal{I}_k(x_{\ell,t})} G_i(x_{\ell,t}) E_i(x_{\ell,t}), \tag{15}$$

where $x_{\ell,t}$ is the token representation supplied to the MLP after input normalization and modulation. The selected experts can differ for every token. Let $Y_\ell \in \mathbb{R}^{T\times d}$ stack the token outputs as rows, with hidden dimension $d$. $\Delta h_\ell$ is $Y_\ell$ after the DiT's output modulation; omitting that modulation gives $\Delta h_\ell = Y_\ell$. There is no parallel shared MLP expert. Attention outputs likewise include their output projection and modulation. RMS is taken over the full valid output tensor after these operations, not averaged over individual expert RMS values. Each branch uses its own immediate input: the MLP input already includes preceding self-attention and cross-attention updates.

**Within-model comparison and aggregation.** Dense references are the blocks retained within each MoE model, rather than blocks from a separate dense model. Using one-based block indices, the pairs are $(1,2), (3,4), \ldots, (27,28)$, with the dense block first; the final two dense blocks are excluded. Both members process their own hidden states at adjacent depths. For sample $a$, pair $j$, and noise level $\sigma$, we compute $r_{a,j,\sigma} = c_{a,2j,\sigma}/c_{a,2j-1,\sigma}$ separately for attention and MLP. Reported paired ratios are medians over samples, pairs, and noise levels, computed before any heatmap aggregation. Each heatmap cell in Figure 5 reports the median $c_\ell$ over five samples at one block and noise level. Cell numbers give $c_\ell$ itself, while the shared logarithmic color scale runs from blue for

smaller outputs through white to red for larger outputs. White or dark numerals are chosen solely for readability against the cell background; their text color does not indicate significance, a separate category, or a different measurement. These measurements describe output amplitude relative to input amplitude, not an information fraction or editing-quality score. Layer and noise-level observations are correlated and are not treated as independent video samples.

### E.2 Expert Allocation Comparisons

**Distributions and coverage.** For each probe, layer, and router, $\mathbf{s} = [s_1, \ldots, s_N]$ contains the $N$ expert selection shares defined in Section 4.3, with $\sum_i s_i = 1$. Shares count top-$k$ selections, rather than gate weights or output magnitudes; attention shares count selected key–value expert pairs. The reported experiment uses 14B-A8B with $N = 7$, $k = 3$, and $L = 14$ MoE layers. Inputs are constructed from noised target-video latents, as in training, rather than collected along a generated sampling trajectory. Its 60 full-instruction probes cover 20 samples at three noise levels; another 20 empty-instruction, ten first-subinstruction, and 20 seed-control probes bring the total to 110. Figure 3(a) averages the 60 full-instruction probes with equal weight. Nonzero selection counts are also checked separately for all $14 \times 2 \times 7 = 196$ layer–router–expert units in each of the 110 probes.

**All-pairs sample comparison.** At $\sigma = 0.833$, let $\mathbf{s}_\ell^{(a)}$ be the full-instruction distribution of sample $a$ at layer $\ell$ for one router type. Every unordered pair of distinct samples is compared once, with no fixed reference sample. With $M = 20$, the reported value is

$$D_{\text{sample}} = \frac{1}{L\binom{M}{2}} \sum_{\ell=1}^{L} \sum_{1 \le a < b \le M} \mathrm{JS}\left(\mathbf{s}_\ell^{(a)}, \mathbf{s}_\ell^{(b)}\right). \tag{16}$$

JS is computed for each of the 190 sample pairs within each layer before averaging. Attention and MLP routers are summarized separately. The statistic measures changes in expert selection proportions; identical proportions need not imply identical token-to-expert assignments.

**Category and matched-condition comparisons.** Category comparisons use the same full-instruction, middle-noise probes. For each layer and router, selection shares are first averaged within each of five editing categories, then JS is computed for all ten category pairs and averaged over pairs and layers. Multi-edit samples belong to every applicable category. These overlapping groups and within-category averaging can reduce divergence, so this descriptive comparison does not isolate a causal effect of editing category. Instruction comparisons use 20 matched full–empty pairs at the middle noise level. Noise comparisons use 25 matched low–high pairs: one per sample and an additional seed pair for five samples. Within each pair, source/target latents and noise realization are fixed; instruction comparisons also fix $\sigma$, and noise comparisons retain the full instruction. Matched-control JS values are first averaged within each sample, then equally over samples and layers. This prevents samples with additional seed pairs from receiving extra weight. All JS values use base-two logarithms and are measured in bits, with zero indicating identical distributions.

# F Evaluation Prompt Templates

## Prompt for Edit Execution Rate

**Role:** You are a strict, impartial evaluator of video editing. Judge only visible evidence and avoid speculation.

**Task:** Determine whether each requested edit has been executed. Compare the edited frames with the corresponding source frames and evaluate every requirement in the checklist.

**Input Data:**
1. Source video: three frames at the first, middle, and last positions (frames 1, 25, and 49).
2. Edited video: the three corresponding, time-aligned frames.
3. Editing instruction: {instruction}
4. Atomic requirement checklist: {checklist}

**Evaluation Criteria:**
1. Correct target: the change must appear on the object or region named in the requirement.
2. Visible execution: the requested change must be recognizable in at least two of the three edited frames.
3. Complete coverage: evaluate every requirement separately. Return one result for each checklist ID, in the same order, without adding or omitting IDs.
4. Observable evidence: an unverifiable requirement receives executed = 0 and score = 0.00. State what you looked for and could not find.
5. Content comparison: ignore small differences in framing, field of view, or aspect ratio introduced by clip preparation. Never infer the producing model.

**Scoring:**
For each edit requirement, return executed = 1 when the requested change is visible on the correct target in at least two frames. Otherwise, return executed = 0. Use integers only.
Preservation requirements are evaluated separately and excluded from the edit execution rate.

**Metric Aggregation:**
For each video, divide the number of executed edit requirements by the total number of requested edits. Average these per-video rates and multiply by 100 for the reported result.

**Output Format:**
Return valid JSON only. The shared response contains:
{"requirements": [{"id": "e1", "executed": 0, "score": 0.00, "evidence": "..."}]}
Evidence: at most 25 words identifying the visible result and frame index.

Figure 7: Evaluation criteria and output format for Edit Execution Rate.

## Prompt for Edit Fulfillment

**Role:** You are an impartial video-editing evaluator. Score only visible evidence in the provided frames.

**Task:** Score how fully and accurately each edit changes the intended target and persists across frames.

**Input Data:**
1. Three time-aligned source and edited frame pairs (frames 1, 25, and 49).
2. Editing instruction: {instruction}
3. Atomic requirement checklist: {checklist}

**Scoring Anchors:**
1.00 Fully realized: correct target and attribute, strong and unambiguous, and maintained in all three frames.
0.75 Clearly and correctly performed, with a visible flaw such as a slightly incorrect color, shape, or material, mild temporal drift, or minor spill onto neighboring regions.
0.50 Weakened: the change is faint, partial, covers only part of the target, or appears in only some frames. A viewer could miss it.
0.25 Only a trace of the requested change, or a change applied to the wrong object or region.
0.00 No trace of the requested change.

**Evaluation Rules:**
1. Use exactly one of the five scores for every edit requirement. Do not reward intent or verbal similarity to the instruction.
2. If an edit receives executed = 0 in the shared evaluation, its score must be at most 0.25. Unverifiable requirements receive 0.00.
3. Do not let overall visual appeal raise a requirement score. Artifacts elsewhere are assessed separately.
4. Return every checklist ID in its original order. Support each judgment with visible evidence of at most 25 words.

**Metric Aggregation:**
Average scores over all requested edits within each video, including unsuccessful edits. Exclude preservation requirements. Then average over videos and multiply by 100.

**Output Format:**
Return valid JSON only. The shared response contains:
{"requirements": [{"id": "e1", "executed": 0, "score": 0.00, "evidence": "..."}]}
The score field determines Edit Fulfillment.

Figure 8: Evaluation criteria and output format for Edit Fulfillment.

## Prompt for Video Quality

**Role:** You are an impartial video-editing evaluator. Judge only visible evidence in the provided frames.

**Task:** Assess the edited video's overall production quality and integration of the requested changes.

**Input Data:**
1. Three time-aligned source and edited frame pairs (frames 1, 25, and 49).
2. Editing instruction: {instruction}

**Evaluation Criteria:**
1. Artifact freedom: inspect warping, smearing, melted textures, duplicated or missing limbs, garbled faces or text, and color banding.
2. Temporal consistency: check whether identity, layout, style, and the applied edits remain consistent across the edited frames.
3. Source fidelity: assess preservation of content the instruction did not ask to change, including subject identity, background, composition, and scene geometry.
4. Edit integration: judge lighting, shadows, perspective, scale, edge quality, and color matching between the edited content and its surroundings.
5. Overall quality: make a holistic judgment of the video as a deliverable. This is not an average of the preceding dimensions. A severe artifact may cap the score.

**Scoring and Rules:**
Use only 1.00, 0.75, 0.50, 0.25, or 0.00 for each quality score.
Set edit_present to 1 if any requested change is recognizable, and to 0 otherwise. If no edit is present, set edit_integration = 0.00 and overall_quality at most 0.25.
Do not penalize explicitly requested changes. Ignore small framing differences introduced by clip preparation.
Do not infer the producing model.
List up to four concrete visible defects in artifacts, or return an empty list.

**Metric Aggregation:**
Video Quality is the overall_quality score, averaged over videos and multiplied by 100. It is not multiplied by the edit execution rate.

**Output Format:**
Return valid JSON only:
{"artifact_free": 0.00, "temporal_consistency": 0.00, "source_fidelity": 0.00,
"edit_integration": 0.00, "edit_present": 0, "overall_quality": 0.00, "artifacts": []}

Figure 9: Evaluation criteria and output format for Video Quality.