S²DiT: Sandwich Diffusion Transformer for Mobile Streaming Video Generation

Supplementary Material

A. Search algorithm

We provide the detailed search algorithm as follows Algorithm 1. Moreover, we adopt design improvements that dramatically reduce computation and yield higher training efficiency.

Efficient Search Space Formulation. We reduce the search space in three ways: (i) We use group-wise rather than block-wise masking, yielding an exponential reduction in the search space. With two blocks per group, the space reduces from $2^{(2n)}$ to 2^n. (ii) We set $m_1 = m_M = 1$ to make the spatial scale matches the VAE compression ratio, further reducing the space to $2^{(n-2)}$. (iii) Base on the dynamic programming, we fix the counts of groups assigned to LCHA (k) and SSA ($n-k-2$) in the search space, reducing the space from $2^{(n-2)}$ to C_{n-2}^{k}. Overall, these choices reduce the space from $2^{(2n)}$ to C_{n-2}^{k}, yielding a substantially more efficient and well-structured search algorithm.

Algorithm 1 Sandwich Architecture Search Algorithm

Input: Input feature y, groups M, group modules $\{(f_L^n, f_S^n)\}_{n=1}^{M}$, upsampler $\mathrm{up}_n(\cdot)$, downsampler $\mathrm{down}_n(\cdot)$

Learnable: Binary routers $\{m_n\}_{n=1}^{M}$ (trained via Gumbel-Softmax + STE)

Output: Output feature $\hat{y}$

Initialize $y_L^0 \leftarrow y,\ y_S^0 \leftarrow \mathrm{down}_1(y)$

Initialize $S^0 \leftarrow 0$

Initialize $m_0 \leftarrow 1$

for $n = 1$ **to** M **do**

`// Branch-Switch Triggers`

$u_n \leftarrow \max\{m_n - m_{n-1},\ 0\}$

$d_n \leftarrow \max\{m_{n-1} - m_n,\ 0\}$

`// Prepare Inputs for Group` n

$x_L^n \leftarrow (1-u_n)\, y_L^{n-1} \ + \ u_n \cdot \left(\mathrm{up}_n(y_S^{n-1}) + S^{n-1}\right)$

$x_S^n \leftarrow (1-d_n)\, y_S^{n-1} \ + \ d_n \cdot \mathrm{down}_n(y_L^{n-1})$

`// Per-Group Forward`

$\tilde{y}_L^n \leftarrow f_L^n(x_L^n)$

$\tilde{y}_S^n \leftarrow f_S^n(x_S^n)$

`// Update Long-Skip Buffer`

$S^n \leftarrow d_n \cdot y_L^{n-1} \ + \ (1-d_n) \cdot S^{n-1}$

`// Differentiable Routing via Gumbel-Softmax + STE`

$y_L^n \leftarrow m_n \cdot \tilde{y}_L^n \ + \ (1-m_n) \cdot y_L^{n-1}$

$y_S^n \leftarrow (1-m_n) \cdot \tilde{y}_S^n \ + \ m_n \cdot y_S^{n-1}$

end for

`// Final Merge`

$\hat{y} \leftarrow m_M \cdot y_L^{(M)} \ + \ (1-m_M) \cdot \mathrm{up}_M\big(y_S^{(M)}\big)$

return $\hat{y}$

Efficient training design. We replace costly pre-training with self-distillation to achieve efficient training, as expressed as:

$$\mathcal{L}_{\mathrm{sd}} = \mathbb{E}\Big[\big\|\hat{y} - T_\theta(y_L^1)\big\|_2^2\Big], \tag{10}$$

where $\mathcal{L}_{\mathrm{sd}}$ is the self-distillation loss, $T_\theta(.)$ denotes the pre-trained teacher model with all self-attention blocks. For the student model, we initialize the LCHA and SSA by inheriting parameters from $T_\theta(.)$ wherever compatible. Since both LCHA and SSA are variants of self-attention, most parameters are transferable, yielding a more efficient training setup.

B. Model Design

RoPE-3D. We follow RoPE [35] to incorporate rotary position embeddings into the linear attention formulation, as shown in Eq. (11). As discussed in [35], RoPE injects position while keep norm unchanged. Therefore, the RoPE transformation is applied only to the outputs of non-linear functions. Meanwhile, the denominator remains unchanged to avoid potential division-by-zero issues.

$$\mathrm{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V})_m = \frac{\sum_{n=1}^{N} \left(\mathbf{R}_{\Theta,m}^{d}\phi(\mathbf{q}_m)\right)^{\top} \left(\mathbf{R}_{\Theta,n}^{d}\varphi(\mathbf{k}_n)\right)\mathbf{v}_n}{\sum_{n=1}^{N}\phi(\mathbf{q}_m)^{\top}\varphi(\mathbf{k}_n)}. \tag{11}$$

Normalization. We employ Layer Normalization in transformer blocks and also as QK norm. In mobile deployment we identify that RMS norm yields higher numeric error, as a result, we use LayerNorm with affine transformation instead. We use AdaLN as timestep encoding which follows common practices [10, 40].

C. More details for 2-in-1 Distillation

C.1. Details of Offline Cached Knowledge Distillation.

After evaluating visual fidelity across several candidates, we adopt Wan2.2-14B [40] as the teacher. A key challenge is Wan2.2's Mixture-of-Experts design, with separate high-noise and low-noise experts, which makes on-the-fly distillation computationally prohibitive. We provide the details of our distillation procedure here.

We propose *Offline Cached Knowledge Distillation*, an offline two-stage protocol: (i) *Cache stage*: for the teacher model, we precompute and cache text embedding e_t, the diffusion tuple of timestep, noise, and velocity of high noise expert (t_h, n_h, v_h) and low noise expert (t_l, n_l, v_l). (ii) *Distillation stage*: during distillation, the training only uses

# Steps	DD	OC	AQ	Quality	Semantic	Total
1	55.09	85.42	52.85	82.80	70.65	80.37
2	57.50	91.25	64.04	85.08	73.15	82.69
4	59.17	89.49	65.64	85.26	73.79	83.26

Table A1. **Analysis of the number of inference steps.** We measure VBench [17] score with different numbers of inference steps. In the results, "DD", "OC", and "AQ" denote the dynamic degree, object class, and aesthetic quality scores, respectively.

Decoder	Params.	Latency	PSNR	SSIM	FVD
Wan 2.1	60M	OOM	32.16	0.8856	23.9
Ours	14M	80ms	31.71	0.8788	27.1

Table A2. Mobile efficient decoder for real-time decoding.

cached tuples and skips teacher forward passes, which substantially reduces both FLOPs and peak memory. The process can be formally defined by:

$$\mathcal{L}_{\text{KD}} = \mathbb{E}\Big[w_l \left\|v_l - V_\theta(t_l, n_l, e_t)\right\|_2^2 + w_h \left\|v_h - V_\theta(t_h, n_h, e_h)\right\|_2^2\Big], \quad (12)$$

where $V_\theta(.)$ indicates the predicted velocity of our model, w_l and w_h are the hyper-parameter to adjust the weight between the two experts.

We set $w_l = w_h = 0.5$ in our experiments.

C.2. Details of Self-Forcing Distillation.

Self-Forcing [16] fine-tuning pipeline has shown promising results for 4-step auto-regressive generation. However, its performance tends to degrade significantly when applied to fewer steps (*e.g.*, 1-step or 2-step generation). A common approach to address this issue is adversarial fine-tuning, which aims to enhance quality in low-step generation. Nonetheless, adversarial fine-tuning often suffers from training instability due to the substantial gap between "real" and "fake" samples, since previous works [46, 52, 54] typically use real-world data as the "real" samples. To mitigate the misalignment, a more intuitive strategy is to adopt progressive adversarial fine-tuning, where samples generated with more sampling steps are treated as the "real" samples instead of real-world data. Following backward simulation [52], we denote $x_{0;T}$ and $x_{0;T'}$ as the x_0 predictions obtained with T and T' sampling steps, respectively, where $T < T'$. In this setting, $x_{0;T}$ serves as the "fake" sample and $x_{0;T'}$ as the "real" sample. Inspired by R3GAN, we employ the RpGAN+R_1 + R_2 formulation as the adversarial fine-tuning objective:

$$\begin{aligned} R_1(\psi) &= \frac{\gamma}{2}\frac{1}{\epsilon}\mathbb{E}\left[\mathcal{D}(x_{t;T'} + \epsilon) - \mathcal{D}(x_{t;T'})\right], \\ R_2(\psi) &= \frac{\gamma}{2}\frac{1}{\epsilon}\mathbb{E}\left[\mathcal{D}(x_{t;T} + \epsilon) - \mathcal{D}(x_{t;T})\right], \end{aligned} \quad (13)$$

$$\begin{aligned} \mathcal{L}_{\text{adv}}^{\mathcal{D}} &= \mathbb{E}\left[\text{softplus}\left(-(\mathcal{D}(x_{t;T'}) - \mathcal{D}(x_{t;T}))\right)\right], \\ \mathcal{L}_{\text{adv}}^{\mathcal{G}} &= \mathbb{E}\left[\text{softplus}\left(\mathcal{D}(x_{t;T'}) - \mathcal{D}(x_{t;T})\right)\right], \end{aligned} \quad (14)$$

where the $x_{t;T'}$ and $x_{t;T}$ are noisy latent that backward simulated with T and T' sampling step added by t noise-level.

Training details. We adopt the AdamW optimizer for the generator and discriminator, using a learning rate of $1.0e{-}5$ for the generator and $2.0e-6$ for the discriminator with betas set to [0, 0.999]. An exponential moving average (EMA) with a decay rate of 0.99 is also applied to the generator for improved training stability.

Evaluation on inference-step. We provide the performance of different inference step in Tab. A1. A single step already delivers surprisingly strong results with 80.37 total score, indicating that most of the global structure and appearance can be formed in one-step. Besides, the two-step setting provides the best speed–quality trade-off and the most stable behavior across metrics. Finally, increasing to four steps further improves accuracy, reaching the highest score (83.26) with acceptable latency.

D. Details of Mobile Deployment

D.1. Efficient Decoder

We freeze the VAE encoder of Wan2.1 and train an efficient decoder that decodes in real-time on mobile, as shown in (Tab. A2). We build the efficient decoder with narrow 3D convolutions, group normalizations, and use Hardswish as the activation function. We encode video data with Wan2.1 encoder and obtain the latents, and the efficient decoder is trained to reconstruct the video with L1 loss, Perceptual loss and GAN loss. Our efficient decoder is $4\times$ smaller in size and can decode beyond real time on mobile, while still deliver on-par reconstruction quality.

Model	Latency (ms)	Steps
Text Encoder	4	1
S^2DiT	260	4
Decoder	80	1
Total	1124	FPS:10.7

Table A3. Latency breakdown for generating a single chunk with 3 latent frames, which will be decoded to 12 pixel frames.

D.2. Deployment Details and Speed Benchmark

We provide a latency breakdown in Tab. A3. In the case of mobile deployment, we solely use CLIP-ViT-L as the text encoder. We generate 3 latent frames for each chunk, and use a window size of 2 for KV cache. We deploy the model with Apple Coremltools [1], with the VAE decoder running on GPU and S^2DiT running on neural engine to maximize

resource utilization. We follow official practice [1] to deploy S^{2}DiT with 8-bit activation quantization and mixed-precision quantization for weights. Specifically, we keep sensitive layers (e.g., project in and out layers, text embedding layers) in 8-bit, but put most other layers in 4-bit. With deploy-time calibration, we observe minimal quality drop compared to the server BF16 model.

E. User Study

We perform a user study of text-to-video generation on 250 randomly sampled MovieGen VideoBench [32] prompts. We compare the S^{2}DiT-pretrained model, knowledge distilled model (KD), and autoregressive model (AR) with two baseline models, i.e., Wan2.1 1.3B [40] and LTX-0.9.5 [10]. Human labelers are asked to pick the best from three anonymous and randomly shuffled videos. We instruct human labelers to focus on two metrics, (i) Text alignment, which evaluates whether the generated video follows the provided input prompt. (ii) Overall Quality, whether the generated video is visually pleasing, *i.e.*, has a completely generated object, meaningful and smooth motion, less flickering and artifacts, etc. Each metric is evaluated by at least 5 labelers, and we take the average win rate. We find that with the instructions, the variance of the picked results from different human labelers is generally low ($< 3\%$ difference in win rate).

As shown in Tab. A4, our base model achieves on-par performance with the server-SOTA Wan2.1 1.3B model [40] and outperforms the server-efficient LTX [10] by a large margin, demonstrating the superior performance of the proposed Sandwich Diffusion Transformer. With the subsequent 2-in-1 distillation, our KD full-step model and streaming generation model achieve an even higher win rate. We are the first to demonstrate high-quality streaming video generation under a mobile budget.

Model:	S^{2}DiT-Pre.	Wan2.1-1.3B	LTX-0.9.5
Text Alignment	49.80%	47.39%	2.81%
Overall Quality	46.99%	43.37%	9.64%
Model:	S^{2}DiT-KD	Wan2.1-1.3B	LTX-0.9.5
Text Alignment	61.45%	36.66%	1.89%
Overall Quality	54.30%	36.29%	9.41%
Model:	S^{2}DiT-AR	Wan2.1-1.3B	LTX-0.9.5
Text Alignment	60.08%	37.48%	2.44%
Overall Quality	57.39%	35.62%	6.99%

Table A4. User study on 250 Randomly Selected Prompts from MovieGen VideoBench [32]. We show the win rate of each model.

F. Comparison with Existing Linear Attention Mechanisms

SANA Video [3] employs ReLU on linear attention and places decoupled temporal and (expanded) spatial Convs in FFN. In contrast, the local path in LCHA directly captures 3D dependency without channel expansion. We show that our proposed LCHA delivers better efficiency and quality metrics in Tab. A5.

Method	Latency↓	$\mathcal{L}_{\text{eval}}$ ↓	CLIP↑	FID↓	FVD↓
Full Attention	OOM	0.252	0.281	32.035	317.130
SANA-Video	930	0.265	0.268	35.732	347.010
LCHA	900	0.259	0.270	33.185	339.380

Table A5. Latency and quality comparisons with existing methods.

G. Longer video generation

We benchmark 30-second video generation on VBench-Long as in Tab. A6. For a fair comparison, we use the same training and inference settings for Wan 1.3B and S^{2}DiT, including a block size of 3, a window size of 2, and the same teacher model Wan2.2 14B. The results show that S^{2}DiT achieves quality very close to Wan 1.3B, while being substantially more efficient for mobile deployment.

AR Model	Mobile	Subject Consistency	Background Consistency	Total
Wan-1.3B	✗	95.05	94.89	82.9
S^{2}DiT	11 FPS	95.03	94.62	82.5

Table A6. VBench-Long benchmark on 30-second videos

H. Generalization across different devices.

We test more platforms for streaming setting to verify the generalization of S^{2}DiT. We get 373ms latency on the older iPhone 14PM, which is 43% slower than iPhone 16PM. We further test on Qualcomm SnapDragon platform, and get 422ms latency on Galaxy S25.

I. Performance without teacher model.

We further include results for S^{2}DiT-AR with self-guidance in Tab. A7, demonstrating that the performance gains of S^{2}DiT are primarily attributed to its architectural design, rather than relying solely on knowledge distillation.

Model	Total Score	Quality Score	Semantic Score
S^{2}DiT-AR	83.26	85.63	73.79
S^{2}DiT-AR $w.o.$ teacher	83.04	84.39	77.64

Table A7. The performance for S^{2}DiT-AR without teacher model.

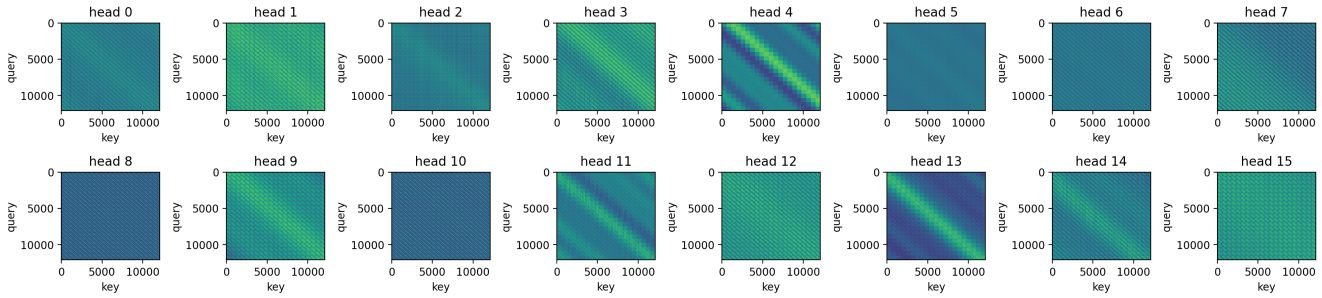

Figure A1. **Visualization of attention maps from our proposed linear attention.**

J. Analysis of Linear Attention

We visualize the attention maps produced by our linear-attention module. As shown in Fig. A1, the heads exhibit clear specialization: some emphasize temporal dynamics (e.g., heads 12 and 15), while others focus on spatial structure (e.g., heads 4 and 13), consistent with prior observations for self-attention [47].

Besides, several heads capture global context (e.g., heads 2 and 5). In conjunction with the local path, this combination enables our model to learn global context and local detail simultaneously. Consistent with this, our full model consistently outperforms the Local path-only variant across all evaluation metrics as illustrated in Tab. 3, confirming the importance of combining global and local.

K. More Visualizations

Beyond the horizontal results reported in the main paper, we include the comparisons of vertical videos that demonstrate the effectiveness of S^2DiT. As shown in Figs. A2 to A4, S^2DiT variants achieves stronger text–video alignment and richer visual details. We include more videos and comparisons in the media file.

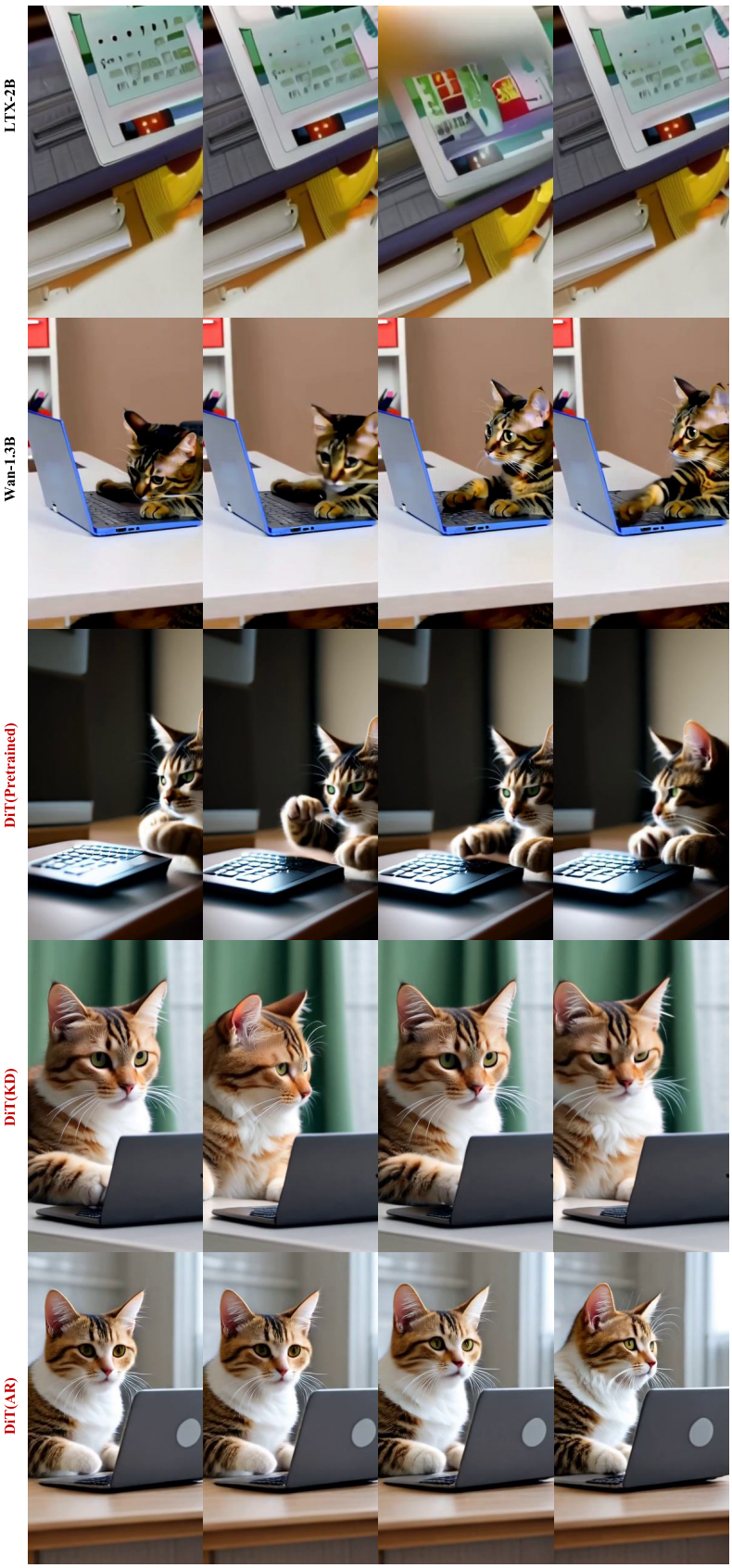

Figure A2. **Qualitative comparison on vertical videos.**

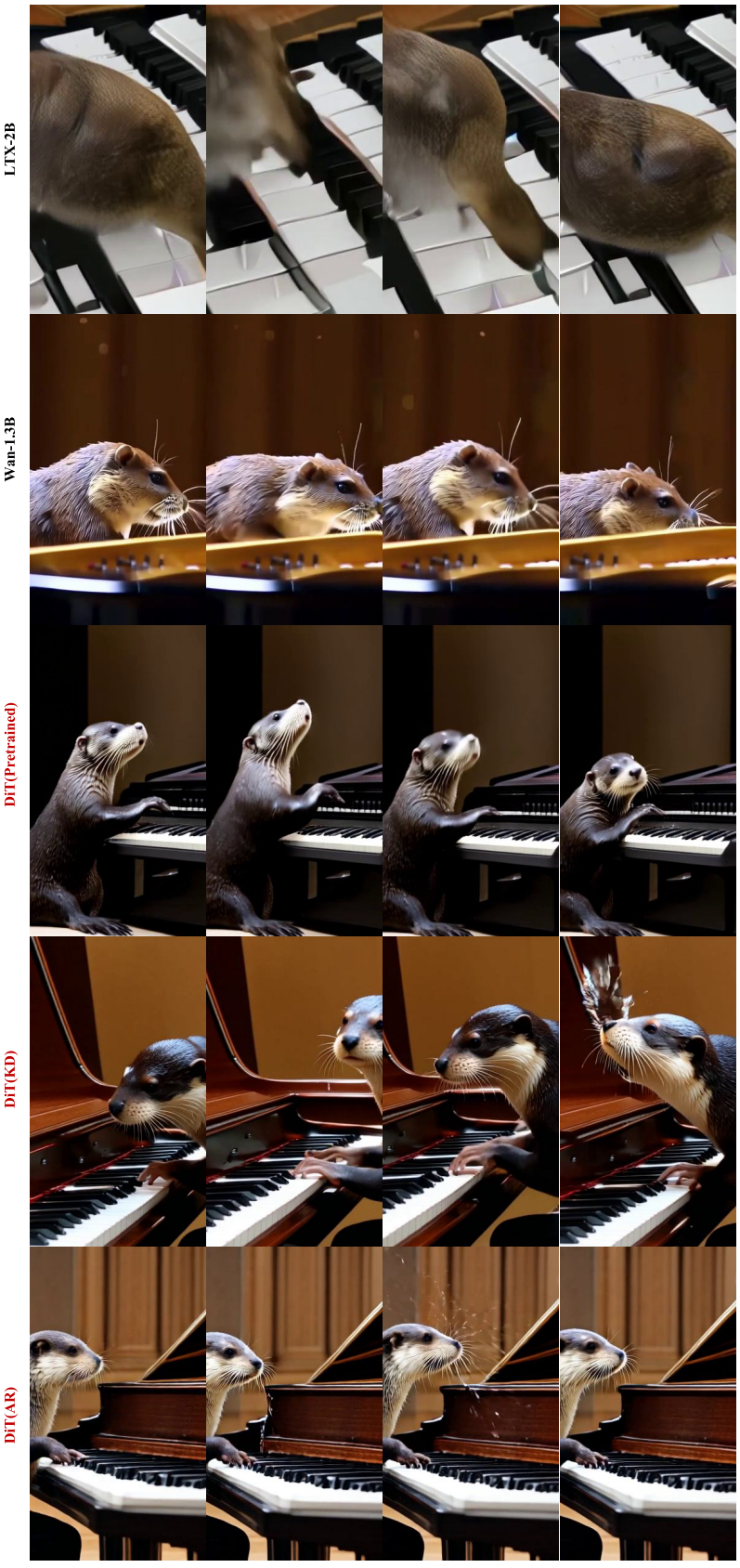

Figure A3. **Qualitative comparison on vertical videos.**

Figure A4. **Qualitative comparison on vertical videos.**