Flowception: Temporally Expansive Flow Matching for Video Generation

Supplementary Material

A. Full derivations

In this section, we provide the derivations for our model. We start with a brief summarization of the Edit Flows framework in video space before deriving the interleaved time schedule for concurrent frame insertions and denoising and training losses.

A.1. Edit flows and frame insertions

Setup. As explained in the main manuscript, we model videos as sequences of frames from the space $\mathcal{X}$. We use a blank token $\varnothing$ to mark empty positions in a sequence of videos. Let $\mathcal{Z} = \bigcup_{n=0}^{N} (\mathcal{X} \cup \{\varnothing\})^n$ define the space were the (augmented) videos live and $f_{\text{strip}} : \mathcal{Z} \to \mathcal{X}$ the mapping from augmented to observable space where f_{strip} removes all blanks (i.e $X_t = f_{\text{strip}}(Z_t)$), and define the product delta on sequences $\delta_{z_1}(z_2) = \prod_i \delta_{z_1^i}(z_2^i)$.

Under this parameterization, a sample $Z_0 \in \mathcal{Z}$ in augmented space is a series of noise frames interleaved with blank tokens at random locations, as illustrated in Figure 9.

Conditional probability path. We prescribe a coupling between source and target distributions. We use the standard independent coupling where each clean frame is paired with an independent Gaussian noise frame. Concretely, for the source we take

$$X_0 \sim \prod_{i=1}^{k} \mathcal{N}(X_0^i; 0, I),$$

and use an augmented variable $Z_t \in (\mathcal{X} \cup \{\varnothing\})^n$ to model masked insertions. At $t = 0$, we start from the all-blank sequence $Z_0 = (\varnothing, \ldots, \varnothing)$ and gradually reveal the clean frames X_1 according to the scheduler κ_t.

Given $X_1 \sim p_{\text{data}}$ a video with n frames, we define a conditional masked path over $Z_t \in (\mathcal{X} \cup \{\varnothing\})^n$ interpolating between $X_0 \in \mathcal{N}(0, I)^k$ (with $k \leq n$) and X_1 where transitions from blank frames to real frames follow the probability law:

$$\begin{aligned} p_t(X_t, Z_t \mid X_1) &= p_t(X_t \mid Z_t)\, p_t(Z_t \mid Z_1) \qquad (13) \\ &= \delta_{f_{\text{strip}}(Z_t)}(X_t) \prod_{i=1}^{n} \Big[(1-\kappa_t)\, \delta_{\varnothing}(Z_t^i) + \kappa_t\, \delta_{X_1^i}(Z_t^i) \Big] \end{aligned}$$

with $\kappa_0 = 0$, $\kappa_1 = 1$. Each token in Z_t is blank with probability $1 - \kappa_t$ or equals X_1^i with probability κ_t.

Continuous-time Markov chain (CTMC). We describe the binary reveal process for frame insertions by a CTMC in augmented space. As described previously, let $Z_t \in \mathcal{Z}$ denote the augmented sequence with blanks, and $X_t = f_{\text{strip}}(Z_t)$ its observable subsequence. The CTMC acts only on the discrete reveal decisions in Z_t; continuous evolution of frame contents is handled separately by the flow-matching ODE which we develop later on.

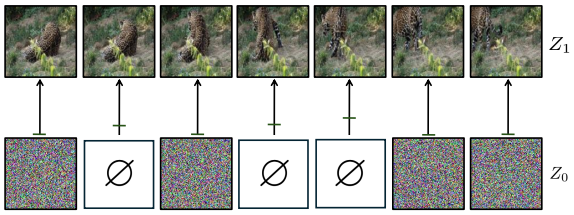

Figure 9. **Illustration of the coupling between source and target distributions in augmented space $\mathcal{Z}$.** Starting frames are initialized as noise vectors $\epsilon \sim \mathcal{N}(0, I)$ while others are blank tokens in augmented space which are transformed into noise vectors at their corresponding insertion time. Horizontal lines on the arrows indicate the time when a frame is revealed in the schedule.

Marginally over Z_t, the induced evolution on observable sequences has the infinitesimal transition kernel

$$\mathbb{P}(X_{t+h} \mid X_t) = \delta_{X_t}(X_{t+h}) + h\, u_t(X_{t+h} \mid X_t) + o(h), \qquad (15)$$

where u_t is the marginal insertion rate obtained from the underlying CTMC on (X_t, Z_t).

Conditional CTMC rate. As demonstrated in [14], a conditional CTMC that samples from (13) can be written

$$\begin{aligned} u_t(x, z \mid X_t, Z_t, X_1) = \qquad (14) \\ \left(\sum_{i=1}^{n} \frac{\dot{\kappa}_t}{1-\kappa_t} \big[\delta_{X_1^i}(z^i) - \delta_{Z_t^i}(z^i) \big] \right) \delta_{f_{\text{strip}}(z)}(x), \end{aligned}$$

where $x = \text{ins}(X_t, i, a)$ for some $i \in [n], a \in \mathcal{X}$. This gives the infinitesimal probability shift from (X_t, Z_t) to (x, z), restricted to next states x that differ by *one insertion* at most. Intuitively, any masked position i transitions from $\varnothing$ to X_1^i with a rate $\dot{\kappa}_t/(1-\kappa_t)$, ensuring that a sample started at $Z_0 = [\varnothing, \ldots, \varnothing]$ reaches X_1 as $t \to 1$.

Training loss. We train a model that transports sequences via insertions, $u_t^\theta(x \mid X_t)$, where $x = \text{ins}(X_t, i, \epsilon)$ for some i, ϵ, by marginalizing the auxiliary Z_t and the data X_1.

As shown by [14], the marginalized ground-truth rate $\bar{u}_t(\cdot \mid X_t) = \sum_z \mathbb{E}_{p_t(z_t \mid X_t)}\, u_t(\cdot, z \mid X_t, z_t, X_1)$ *generates* $p_t(X_t)$, any Bregman divergence $D_\phi(a, b) = \phi(a) - \phi(b) - \langle a - b, \nabla\phi(b) \rangle$ can be used to regress the marginal rate. Following [14, 17, 24], we use a Bregman divergence

between measures over next states and marginalize over all z such that $x = f_{\text{strip}}(z)$:

$$\mathbb{E}_{X_1 \sim p_{\text{data}}} E_{(X_t, Z_t) \sim p_t(X_t, Z_t \mid X_1)} D_\phi \left(\sum_z u_t(\cdot, z \mid X_t, Z_t, X_1), u_t^\theta(\cdot \mid X_t) \right). \quad (15)$$

Choosing the entropy as a potential function $\phi(u) = \langle u, \log u \rangle$ gives the explicit loss (see Theorem B.2 for the derivation of this term)

$$\mathcal{L} = \mathbb{E}_{t, X_1, X_t, Z_t} \left[\sum_{x \neq X_t} u_t^\theta(x \mid X_t) - \sum_{i=1}^{n} \mathbf{1}(Z_t^i = \varnothing) \frac{\dot{\kappa}_t}{1 - \kappa_t} \log u_t^\theta\big(\text{ins}\big(X_t, j, X_1^i\big) \mid X_t\big) \right], \quad (16)$$

where j is the slot in X_t corresponding to the first non-$\varnothing$ coordinate to the left of Z_t^i. This alignment ensures that inserting at position i corresponds to changing $Z_t^i : \varnothing \to X_1^i$.

Loss simplification. Following [24], we adopt an equivalent t-independent parameterization for one-insertion moves. Rather than modeling separate rates for each individual missing frame, we define a single slot-level insertion rate. For a one-insertion move $x = \text{ins}(X_t, i, \epsilon)$ (inserting a fresh noise frame ϵ at slot i), we write

$$u_t^\theta\big(\text{ins}(X_t, i, \epsilon) \mid X_t\big) = \frac{\dot{\kappa}_t}{1 - \kappa_t} \lambda^i(X_t), \quad (17)$$

where $\lambda^i(X_t) \geq 0$ is the *total insertion rate* at slot i. The actual frame content is always sampled from a fixed noise prior (e.g. $\epsilon \sim \mathcal{N}(0, I)$) and is not parameterized.

Let $\mathcal{A}_j$ denote the set of missing frames associated with slot j, i.e. those to be inserted "to the right" of j at time t under the alignment. Substituting this parameterization into Equation (16) and collecting the θ-dependent terms yields

$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{(\cdot)} \frac{\dot{\kappa}_t}{1 - \kappa_t} \sum_{j=1}^{\ell(X_t)} \left(\lambda^j(X_t) - \sum_{a \in \mathcal{A}_j} \log \lambda^j(X_t) \right) \\ &= \mathbb{E}_{(\cdot)} \frac{\dot{\kappa}_t}{1 - \kappa_t} \sum_{j=1}^{\ell(X_t)} \Big(\underbrace{\lambda^j(X_t) - |\mathcal{A}_j| \log \lambda^j(X_t)}_{\text{Poisson NLL}} \Big) + \text{const.} \end{aligned} \quad (18)$$

where the expectation is over $\tau, X_1 \sim p_{\text{data}}$ and $(X_t, Z_t) \sim p_t(X_t, Z_t \mid X_1)$, while $t = \text{clip}(\tau, 0, 1)$. As pointed out by [14], keeping the factor $\frac{\dot{\kappa}_t}{1 - \kappa_t}$ preserves a direct ELBO interpretation, while removing it can be more stable in practice; both choices recover the familiar Poisson loss over insertion counts. So in practice, the loss we use is

$$\mathcal{L}_{\text{ins}}(\theta) = \mathbb{E}_{(\cdot)} \left[\sum_{j=1}^{\ell(X_t)} \big(\lambda^j(X_t) - |\mathcal{A}_j| \log \lambda^j(X_t) \big) \right]. \quad (19)$$

A.2. Velocity flow-matching objective

Now we have the loss for the insertion term (where to insert and with with what probability), now we need to model how to update the currently active frames. We briefly recall the derivation of the velocity loss used for denoising active frames.

Following rectified flow matching, for $t_i \in [0, 1]$, each frame i follows a a linear probability path t_i

$$X_{t_i}^i = t_i X_1^i + (1 - t_i) X_0^i \sim p_t^i. \quad (20)$$

Consider a rectified flow coupling between source $X_0 \sim p_0$ and target $X_1 \sim p_1$ governed by the ODE

$$\frac{dX_t}{dt} = v(X_t, t), \qquad t \in [0, 1]. \quad (21)$$

Under this coupling, the population-optimal velocity field satisfies

$$v^\star(X_t, t) = \mathbb{E}\left[X_1 - X_0 \;\middle|\; X_t, t \right]. \quad (22)$$

The Conditional Flow Matching (CFM) objective then trains a neural velocity field v_θ to regress onto this target:

$$\mathcal{L}_{\text{vel}} = \mathbb{E}_{\tau, X_0, X_1} \Big[\mathbf{1}_{[0,1)}(\tau) \big\| v_\theta(X_t, t) - (X_1 - X_0) \big\|^2 \Big], \quad (23)$$

where $t = \text{clip}(\tau, 0, 1)$. As shown by [22], this CFM loss has the same optimum as the original Flow Matching objective and is minimized uniquely by $v_\theta = v^\star$ in function space.

In Flowception we apply this objective at the *frame* level. Each frame i has a local time $t_i \in [0, 1)$ induced by the extended-time construction described above, and we write $X_{t_i}^i$ for its state along the linear path. The loss in Equation (23) is evaluated only on *active* frames, i.e. those with $\tau_i \in [0, 1]$, and we mask out both frozen frames ($\tau_i < 0$) and terminal frames ($\tau_i \geq 1$) during training and sampling.

A.3. Interleaved time schedule for frame insertions

We now derive the interleaved schedule used to concurrently insert new frames and denoise existing frames, ensuring that training and sampling observe the same joint law over local times.

Design choice. At the instant of insertion we can either (i) fully or partially denoise the frame, or (ii) insert pure noise and denoise afterwards. We adopt (ii) for concurrency and parallelism: a single forward pass handles both velocity prediction for present frames and insertion decisions, while newly inserted frames start at local time 0 and are denoised in context thereafter.

Since the source $p_{\text{src}} = \mathcal{N}(0, I)$ and target p_{data} distributions of the data are usually decoupled, any sample from p_{src} is valid for the inserted frame.

Global and local times. Let $t_g \in [0,1]$ denote the *sequence* (global) time that advances monotonically during generation. Each frame i has a local time $t_i \in [0,1]$ (used by the rectified flow coupling). We must respect the causal constraint that a frame cannot be more denoised than the sequence has progressed, i.e., $t_g \geq t_i$ for all frames i currently present.

Insertion-time law and inverse-CDF sampling. Let $\kappa : [0,1] \to [0,1]$ be the monotone reveal scheduler with $\kappa_0 = 0$, $\kappa_1 = 1$, and define the hazard $\rho_\kappa(t) = \dot{\kappa}_t/(1-\kappa_t)$. We model insertion times by the density $p(t_{\text{ins}}) = \dot{\kappa}_t$, equivalently $t_{\text{ins}} = \kappa^{-1}(u)$, $u \sim \text{Unif}(0,1)$.

For a frame inserted at sequence time t_g, we set its local time to zero: $t_i \leftarrow 0$. Hence the instantaneous offset between the global and local times is distributed as

$$t_g - t_i = t_{\text{ins}}, \qquad 0 \leq t_g, t_i, t_{\text{ins}} \leq 1, \tag{24}$$

so that local time always lags behind global time by a random, scheduler-consistent delay.

Multi-frame generalization. To make Equation (24) hold during training for all frames, we lift time to an extended interval and tie each frame to an independent offset. Let

$$\tau_g \in [0,2], \qquad t_g = \text{clip}(\tau_g), \qquad t = \text{clip}(\tau, 0, 1). \tag{25}$$

For every potential frame index i, draw $u_i \sim \text{Unif}(0,1)$ and define the *extended local time*

$$\tau_i = \tau_g - \kappa^{-1}(u_i), \qquad t_i = \text{clip}(\tau_i). \tag{26}$$

This yields three phases per frame:

- (frozen): $\tau_i < 0$
- (flowing): $\tau_i \in [0,1]$
- (terminal): $\tau_i > 1$

Let $A = \{i : \tau_i \in [0,1]\}$ denote the active set $\tau_i \in [0,1)$. During training we sample $(\tau_g, \{u_i\})$, form $(t_g, \{t_i\})$, delete frames with $\tau_i < 0$, and apply the flow matching loss only to indices in A. By construction, the marginal law of $(t_g, \{t_i\})$ exactly matches that encountered at sampling, thus samples remain in-distribution.

B. Additional derivations and remarks

B.1. Deriving the insertion rate

Let $\kappa : [0,1] \to [0,1]$ be a nondecreasing insertion schedule with $\kappa(0) = 0$, $\kappa(1) = 1$, differentiable almost everywhere. For a single frame, define its reveal time T with cumulative density function (CDF) $F_T(t) = \mathbb{P}[T \leq t] = \kappa(t)$ on $[0,1]$ and survival $S(t) = 1 - F_T(t) = 1 - \kappa(t)$.

Lemma B.1 (Instantaneous reveal hazard)**.** *The instantaneous reveal hazard for insertion schedule κ is given by*

$$\begin{aligned} \rho_\kappa(t) &= \lim_{\Delta \to 0^+} \frac{\mathbb{P}\big(T \in [t, t+\Delta) \mid T > t\big)}{\Delta} \\ &= \frac{\dot{\kappa}(t)}{1-\kappa(t)}, \end{aligned} \tag{27}$$

for $t \in (0,1)$ and $\rho_\kappa(t) = 0$ outside $(0,1)$.

Proof. By definition, for $[t, t+\Delta) \subset [0,1)$, we have

$$\begin{aligned} \mathbb{P}(T \in [t, t+\Delta) \mid T > t) &= \frac{\mathbb{P}(T \in [t, t+\Delta), T > t)}{\mathbb{P}(T > t)} \\ &= \frac{F_T(t+\Delta) - F_T(t)}{S(t)} \\ &= \frac{\kappa(t+\Delta) - \kappa(t)}{1-\kappa(t)}. \end{aligned} \tag{28}$$

Plugging this in Equation (27) results in $\rho_\kappa(t) = \frac{\dot{\kappa}(t)}{1-\kappa(t)}$. For $t \notin (0,1)$, F_T is constant, hence $\rho_\kappa(t) = 0$. □

B.2. From Bregman divergence to the insertion loss

Let $\varphi(z) = z \log z - z$ on $\mathbb{R}_{\geq 0}$ and $D_\varphi(u \| \lambda) = \sum_j \{u_j \log(u_j/\lambda_j) + \lambda_j - u_j\}$.

Proposition B.2 (Bregman objective)**.** *For a fixed snapshot,*

$$\sum_j \big[\lambda_j - u_j \log \lambda_j\big] \quad \textit{and} \quad D_\varphi(u \| \lambda)$$

differ by a constant in u; thus they have the same minimizers. The training objective

$$\mathcal{L}_{\text{ins}}(\theta) = \mathbb{E}_{(\cdot)}\Big[\sum_j \lambda_{t,j}(X_t) - u_{t,j}(X_t) \log \lambda_{t,j}(X_t)\Big]$$

is therefore a valid form to learn that converges towards the marginal expectation while only the conditional expectation is used.

Corollary B.3 (Pointwise optimum)**.** *The per-snapshot objective in Prop. B.2 is strictly convex in each $\lambda_j > 0$ and is minimized uniquely at $\lambda_j = u_{t,j}$.*

Proof. Similarly to Edit Flows [14], we make use of the cross-entropy generator function to define the potential function for the Bregman divergence :

$$\forall (z \geq 0), \quad \phi(z) = z \log z - z.$$

Given ground-truth nonnegative targets $u = \{u_j\}_j$ and model predictions $\lambda = \{\lambda_j\}_j$, the separable Bregman divergence is

$$D_\phi(u \| \lambda) = \sum_j \Big(\phi(u_j) - \phi(\lambda_j) - \phi'(\lambda_j)\,(u_j - \lambda_j)\Big).$$

Which simplifies for each coordinate j to

$$\begin{aligned}&\phi(u_j) - \phi(\lambda_j) - \phi'(\lambda_j)(u_j - \lambda_j)\\ =&\big(u_j \log u_j - u_j\big) - \big(\lambda_j \log \lambda_j - \lambda_j\big) - (\log \lambda_j)\,(u_j - \lambda_j).\end{aligned} \tag{29}$$

Collecting terms gives

$$\begin{aligned}D_\phi(u\|\lambda) &= \sum_j \Big(u_j \log \tfrac{u_j}{\lambda_j} + \lambda_j - u_j\Big)\\ &= \sum_j \Big(\lambda_j - u_j \log \lambda_j\Big) + \underbrace{\sum_j \big(u_j \log u_j - u_j\big)}_{\text{constant in } \lambda}.\end{aligned} \tag{30}$$

Since $\sum_j (u_j \log u_j - u_j)$ does not depend on λ, minimizing $\mathbb{E}[D_\phi(u\|\lambda)]$ over model parameters is equivalent to minimizing $\mathbb{E}\Big[\sum_j \lambda_j - u_j \log \lambda_j\Big]$.

For the reveal schedule $\kappa : [0,1] \to [0,1]$ with hazard $\rho_\kappa(t) = \dot\kappa(t)/(1-\kappa(t))$, the true marginal insertion rate at snapshot (X_t, t) is

$$u_{t,j}(X_t) = \rho_\kappa(t)\, K_j(X_t)\, \mathbf{1}_{[0,1]}(t),$$

where $K_j(X_t)$ is the pending-count in slot j. The general form of the insertion loss then becomes

$$\begin{aligned}\mathcal{L}_{\text{ins}}(\theta) &= \mathbb{E}\left[\sum_j \lambda_{t,j}(X_t) - u_{t,j}(X_t)\, \log \lambda_{t,j}(X_t)\right]\\ &= \mathbb{E}\left[\sum_j \lambda_{t,j}(X_t) - \rho_\kappa(t) K_j(X_t) \mathbf{1}_{[0,1]}(t) \log \lambda_{t,j}(X_t)\right].\end{aligned} \tag{31}$$

In particular, when using the linear scheduler $\kappa(t) = t$, then $\rho_\kappa(t) = \frac{1}{1-t}$ and

$$\mathcal{L}_{\text{ins}}(\theta) = \mathbb{E}\Big[\textstyle\sum_j \lambda_{t,j}(X_t) - \frac{K_j(X_t)\,\mathbf{1}_{[0,1]}(t)}{1-t} \log \lambda_{t,j}(X_t)\Big] \tag{32}$$

□

B.3. Generalized insertions with Poisson thinning

From Bernoulli to Poisson. The Bernoulli-thinning sampler in Algorithm 1 draws a single bit per slot and per step, which permits at most one insertion in slot j during the step of size Δ. As $\Delta \to 0$, the sum of independent Bernoulli micro-trials with success probability $1 - \exp(-\lambda\,\Delta)$ converges in law to a Poisson random variable with mean $\int \lambda(s)\, ds$. This suggests a finite-step scheme that explicitly allows multiple insertions.

Poisson process. A time-inhomogeneous Poisson process on $[t, t+\Delta)$ with instantaneous rate $\rho(s) \geq 0$ has independent increments and satisfies

$$N([t, t+\Delta)) \;\sim\; \text{Poisson}\left(\int_t^{t+\Delta} \rho(s)\, ds\right).$$

If $\rho(s)$ is approximately constant on the step, then $N \sim \text{Poisson}(\rho(t)\,\Delta)$. Conditioned on $N = n$, the event times are i.i.d. uniform in the interval.

Drawing from Poisson process. To enable multiple insertions per slot, we replace the Bernoulli draw with a Poisson draw

$$\begin{aligned}N_{t,j} \,\big|\, (X_t, t) \;&\sim\; \text{Poisson}\big(\Lambda_{t,j}\big),\\ \Lambda_{t,j} \;&\approx\; \Delta\, u_{t,j}(X_t) \;=\; \Delta\, \rho_\kappa(t)\, K_j(X_t).\end{aligned} \tag{33}$$

This enables us to insert $N_{t,j}$ new elements into slot j (each initialized with independent base noise) and doing this in parallel across all slots. This preserves the expected number of births per step, $\mathbb{E}[N_{t,j}] \approx \Delta\, u_{t,j}(X_t)$, and removes the at most one insertion per slot constraint.

C. Discussions

C.1. Baseline implementation details

Full-sequence. For the full-sequence model training, we follow standard setups [13, 36] and sample timesteps during training according to a lognorm schedule. Similarly to Flowception, we expand the channel dimension of the input by a factor two, and use the second half to encode the (clean) context frames, allowing to support the image-to-video framework. For training on variable length videos, we experiment with two strategies. (1) In each batch we have videos of different length, and we mask out the loss on padded frames. (2) in each batch we collect videos of the same length only. We find (2) to perform more favorably in preliminary experiments and consequently use it for our remaining experiments.

Autoregressive. We also use a lognorm schedule for the timestep sampler, while the number of context frames is sampled randomly with uniform probability across the length of the video.

C.2. Finetuning Pre-trained Models

In order to finetune open-source models with our method, there are some changes that need to be done to the architecture to adapt to our framework, we describe these changes below.

- Framewise time embeddings: In Flowception, each frame in the video can have different noise level associated to a timestep t_i, we therefore adapt the AdaLN layers to modulate each frame independently according to its noise level.
- Variable length: we extend the model to support variable length sequences per-batch by feeding a frame activity mask which is responsible for the masking padding frames from the attention.
- Insertion rate tokens: We append one learnable frame token per frame to the sequence (after patch embedding and

before the transformer stack). This token participates in all self and cross-attention layers, takne the rope coordinates corresponding to the gap between layers at the center spatial position $(c_x, c_y, c_t = W/2, H/2, t_i)$, allowing it to aggregate frame-level information to predict the insertion rates. After the transformer layers, a lightweight head followed by an exponential activation is used to predict the insertion rates.

C.3. Flowception as implicit temporal compression

When a frame is inserted as pure noise ($x_{\text{new}} = \varepsilon,\ t_{\text{new}} = 0$), its clean identity among the (K) pending in-slot frames is unresolved at birth. Under the masked Flow Matching objective, the population-optimal first velocity is the conditional mean *over the posterior of the missing frames* (integrating both which clean frame it will become and that frame's content):

$$v^{\star}_{\text{new}} = \mathbb{E}_{\text{miss}}\big[X_{1,\text{new}} \mid X_t, t, M\big] - \varepsilon \tag{34}$$

$$= \mathbb{E}_{\text{miss}}\big[Z \mid X_t, t, M\big] - \varepsilon, \tag{35}$$

where Z is a random clean frame drawn from the posterior over the K not-yet-revealed frames in that slot induced by the snapshot law and the insertion-rate hazard $u_{t,j}(X_t) = \rho_\kappa(t), K_j(X_t)$, with $\rho_\kappa(t) = \dot\kappa(t)/(1-\kappa(t))$, and $\rho_\kappa(t) = 1/(1-t)$ for linear κ. Thus the first update points from noise toward a *group-wise conditional expectation over the missing frames*, not toward a single target.

In full-sequence flow matching, by contrast, all frames are present (as noise) at every step, so there is no identity ambiguity: the optimal direction for index j is the per-index conditional mean $\mathbb{E}[X_{1,j} \mid X_t, t] - X_{0,j}$, i.e., no marginalisation over missing content. The Flowception update therefore acts like an implicit temporal aggregator early on: active tokens move toward expectations that average over as-yet unseen in-between motion, while the unrevealed frames are integrated out.

D. Efficiency Comparison

We now study the impact of using Flowception on sampling efficiency compared to full-sequence diffusion and autoregressive paradigms. Let $L = H \times W$ denote the number of tokens per frame, and n the number of frames. The total sequence length is therefore nL. For this analysis we disregard the text tokens and the per-frame extra rate token for Flowception since we are only interested in orders of magnitude.

Full-sequence. Full-sequence diffusion and flows evolve all frames simultaneously, sharing a global timesteps between the n frames. At each sampling step all nL tokens are active, and self-attention dominates with quadratic cost. The total complexity over T_{full} steps is therefore

$$\mathcal{C}_{\text{full}} \approx T_{\text{full}}\,(nL)^2.$$

Autoregressive (no caching). In autoregressive diffusion/flow-matching, frames are generated sequentially, one at a time, each conditioned on all previously generated frames. A generation step involves appending a noise frame to the end of the sequence before evolving it using flow matching, which requires T_{AR} inner steps in order to evolve the noise sample into a valid frame by predicting the velocity field for that frame. At step j, the active sequence length is jL, yielding a cumulative cost

$$\mathcal{C}_{\text{AR}} \approx T_{\text{AR}} \sum_{j=1}^{n} (jL)^2 \le \tfrac{1}{3}\, T_{\text{AR}}\, L^2 n^3.$$

This cubic dependence on n makes autoregressive diffusion substantially more expensive than full-sequence diffusion.

Autoregressive diffusion (with caching). With key-value (KV) caching, past tokens do not need to be recomputed. At step j, attention is computed only between the L new tokens and the cached jL past tokens, for cost $\mathcal{O}(jL^2)$. Summing across n frames yields

$$\mathcal{C}_{\text{AR+cache}} \approx T_{\text{AR}} \sum_{j=1}^{n} jL^2 \approx \tfrac{1}{2}\, T_{\text{AR}}\, L^2 n^2.$$

Flowception. When starting from the empty sequence and under a linear insertion scheduler, the active fraction is $\kappa(\tau)=\tau$ and the (expected) active sequence length art any point is $R_\tau = \tau nL$. Averaging the quadratic self-attention cost over the trajectory yields

$$\mathcal{C}_{\text{flow}} \approx T_{\text{flow}}\,(nL)^2\,\mathbb{E}_\tau[\tau^2] = \tfrac{1}{3}\, T_{\text{flow}}\,(nL)^2.$$

Allowing Flowception to take α times more steps than the baseline ($T_{\text{flow}} = \alpha T_{\text{full}}$), to account for the delayed denoising of frames interted later, gives

$$\mathcal{C}_{\text{flow}} \approx \tfrac{\alpha}{3}\, T_{\text{full}}\,(nL)^2.$$

In our experiments we set $\alpha = 2$ to roughly allow the same number of denoising steps per frame as the full-sequence model, even for frames inserted close to $t_g = 1$.

Comparison. The asymptotic speedups between the different methods are

$$\text{speedup}_{\text{FC vs. Full}} = \frac{\mathcal{C}_{\text{full}}}{\mathcal{C}_{\text{flow}}} \approx \frac{3}{\alpha}$$

$$\text{speedup}_{\text{FC vs. AR}} \approx \frac{n}{\alpha}\,\frac{T_{\text{AR}}}{T_{\text{full}}}$$

$$\text{speedup}_{\text{FC vs. AR+cache}} \approx \frac{3}{2\alpha}\,\frac{T_{\text{AR}}}{T_{\text{full}}}$$

Flowception achieves computational efficiency by interleaving discrete frame insertions with continuous denoising, resulting in an implicit temporal compression that reduces

Table 5. **Comparison of expected FLOPs during sampling.**

	Full-seq	AR	AR+cache	Flowception
Complexity	$T_{\text{full}}(nL)^2$	$\frac{n}{3}T_{\text{AR}}(nL)^2$	$\frac{1}{2}T_{\text{AR}}(nL)^2$	$\frac{\alpha}{3}T_{\text{full}}(nL)^2$

the average sequence length per sampling step. Unlike AR without caching models which exhibit cubic complexity $\mathcal{O}(n^3)$ in the number of frames n, Flowception scales quadratically, $\mathcal{O}(n^2)$, providing a significant asymptotic speedup. Compared to full-sequence diffusion, the method achieves a theoretical speedup of approximately $3/\alpha$ (yielding $1.5\times$ for $\alpha = 2$) by leveraging a linear insertion schedule that avoids redundant computation on noise-only tokens. We summarize the complexities of the different frameworks in Table 5.

E. Algorithms & implementation

We provide algorithms for Flowception training and sampling procedures.

In Algorithm 1, we provide a sketch of the sampling algorithm, assuming a number of starting frames n_{start} and a step size h that is shared between insertions and flow matching. For simplicity, we do not include context frames in the sketch of the algorithms. We start with $t_i = 0$ for the starting frames. Each sampling step iterates two operations, flow matching on the current set of frames, followed by insertions to the right of each frame $i \in \{1, \dots, \ell(X)\}$, which happens with probability $h_i \lambda_i \frac{\dot{\kappa}(t_g)}{1-\kappa(t_g)}$ where λ_i is the rate associated with the frame i. When a new frame is inserted, we insert a new frame as a pure noise $\text{ins}(X, i, \epsilon), \epsilon \sim \mathcal{N}(0, I)$.

We detail the training algorithm in Algorithm 2. First, we sample a set of timesteps according to the Flowception schedule, $\tau_i,\ i \in \{0, \dots, n\}$, we map these timesteps to deletion operations according to the insertion schedule $X \leftarrow f_{\text{strip}}(X_{\text{target}}, M), t \leftarrow f_{\text{strip}}(t, M)$. Next, the remaining frames are noised according the rectified flow matching schedule $X = tX + (1-t)X_0$. The model is then fed these noised frames and their associated timesteps, predicting their associated velocities and insertion rates. The training losses are detailed in the main manuscript.

Time sampling. using the time sampling in Algorithm 2, it can happen that for the sampled τ_g all frames in a video are already denoised (in particular when all frames are inserted early), rendering the video useless for training. To ensure that there is at least one evolving frame per video, we can instead first sample the terminal time for the last frame in the video (when the last flow step happens), before deriving τ_g and sampling the individual offsets. To do this we proceed in the following manner:

1. Sample the insertion times $t^i_{\text{ins}} \sim \text{Unif}(0, 1)$
2. Compute maximum $\tau_{\max}$ that we need: $\max\{t^i_{\text{ins}}\} + 1$
3. Sample $\tau_g \sim \text{Unif}(0, \tau_{\max})$
4. Compute $\tau_i = \tau_g - t^i_{\text{ins}}$

Additionally, following other works [13, 24, 36], we used

Algorithm 1 Flowception generation procedure

function FLOWCEPTIONGENERATION(step size h)
  $X \sim \prod_{i=1}^{n_{\text{start}}} \mathcal{N}(X^i; 0, I)$
  $t \leftarrow [0, \dots, 0] = [0]^{n_{\text{start}}}$ ▷ per-frame times
  $t_g \leftarrow 0$ ▷ global time
  while $\min\{t_i\} < 1$ **do** ▷ iterate until all frames are clean
    $X, t, t_g \leftarrow$ FLOWCEPTIONSTEP(X, t, t_g, h)
  end whilereturn X
end function
function FLOWCEPTIONSTEP(X, t, t_g, h)
  $v, \lambda \leftarrow \texttt{FlowceptionModel}(X, t; \theta)$
  $h_i = \min\{h, 1 - t_i\}$ ▷ clean frames are frozen
  $X \leftarrow X + hv$ ▷ apply flow step to denoise
  $t \leftarrow \text{clip}(t + h, 0, 1)$
  $t_g \leftarrow \max(t)$ ▷ update time trackers
  ▷ all insertions are implemented in parallel
  for all $i \in \{1, \dots, \ell(X)\}$ **do**
    with probability $h_i \lambda_i \frac{\dot{\kappa}(t_g)}{1-\kappa(t_g)}$:
      ▷ inserted frames are set to pure noise
      $X = \text{ins}(X, i, \varepsilon)$ where $\varepsilon \sim \mathcal{N}(0, I)$
      ▷ inserted time values are set to zero
      $t = \text{ins}(t, i, 0)$
  end for
  return X, t
end function

Algorithm 2 Flowception training procedure

Require: scheduler κ
function FLOWCEPTIONTRAININGSTEP(κ)
  $X_{\text{target}} \sim p_{\text{target}}$
  $\tau_g \sim p(\tau_g)$ ▷ can choose e.g. logit normal
  $u_i \sim \text{Unif}(0, 1)$
  $\tau_i \leftarrow \tau_g - \kappa^{-1}(u_i)$ ▷ per-frame extended times
  $M_i \leftarrow \mathbb{1}_{[\tau_i \geq 0]}$ ▷ i-th frame is deleted if $\tau_i < 0$
  ▷ sample noisy frames
  $t_i \leftarrow \text{clip}(\tau_i)$
  $X_0 \sim \mathcal{N}(0, I)$
  $X = tX_{\text{target}} + (1-t)X_0$
  ▷ remove deleted frames
  $X \leftarrow f_{\text{strip}}(X, M)$
  $t \leftarrow f_{\text{strip}}(t, M)$
  $v, \lambda \leftarrow \texttt{FlowceptionModel}(X, t; \theta)$
  $\mathcal{L} \leftarrow \dots$ ▷ compute insertion and velocity losses
  $\theta \leftarrow \texttt{optimizer\_step}(\theta, \nabla\mathcal{L})$
end function

a lognorm global schedule during training to be beneficial $\tau_g \sim \text{lognorm}(0, 1) \cdot \tau_{\text{max}}$.

Loss reduction. Another important point is about the loss reduction for both the velocity and Poisson likelihood, since each sample in the training batch can have a varying number of frames which are still evolving, i.e. with $0 \leq \tau_i < 1$, a question arises around how to reduce the velocity loss across these frames. We experimented with both per-sample mean reduction (each video contributes equally independently from the number of active frames) and a mean reduction across all active frames in the batch (all frames in the batch contribute equally, so shorter videos contribute less). We found the latter to be more stable, especially for longer sequences while the former tends to over-optimize for short sequences (early stages of sampling or shorter videos) which in turns biases the model to under-insert.

Local sampling schedules. In video generation, early diffusion timesteps are particularly important, as they establish global structure, motion, and temporal alignment before later steps primarily refine appearance. This effect is amplified in our setting, where frames are inserted asynchronously into an evolving sequence: immediately after insertion, a frame is highly uncertain and must rapidly become consistent with its temporal neighbors. To address this, the denoising time steps can be biased towards the start of the process. For example, [28] uses a linear-quadratic scheduler where the first portion of sampling follows a linear schedule with a small step size $\approx 1/1000$, while the remaining time steps follow a quadratic schedule to arrive at $t = 1$.

To prioritize this post-insertion regime in Flowception, we introduce a *framewise time reparameterization*: each frame i maintains its own solver coordinate $u_i \in [0, 1]$ and physical diffusion time $t_i \in [0, 1]$, linked by a strictly increasing function $t_i = f(u_i)$. During sampling, we advance the solver coordinates by a fixed step Δu for all active frames, compute the corresponding physical increments $\Delta t_i = f(u_i + \Delta u) - f(u_i)$, and update $x_i \leftarrow x_i + \Delta t_i \, v_\theta(x_i, t_i)$ in the same scalar time variable $t \in [0, 1]$ used during training. This preserves consistency with the training setup while allowing the effective step size in diffusion time to depend on a frame's "age" since insertion. In practice, we instantiate this family with a power schedule $t_i = u_i^\gamma$ (with $\gamma > 1$), which yields small Δt_i for newly inserted frames (small u_i) and larger Δt_i for older, well-established frames. As a result, the sampler allocates more computation to the crucial early timesteps after each insertion and fewer steps to later, easier denoising phases, leading to improved temporal coherence and fewer artifacts compared to a uniform schedule ($\gamma = 1$) under the same compute budget.

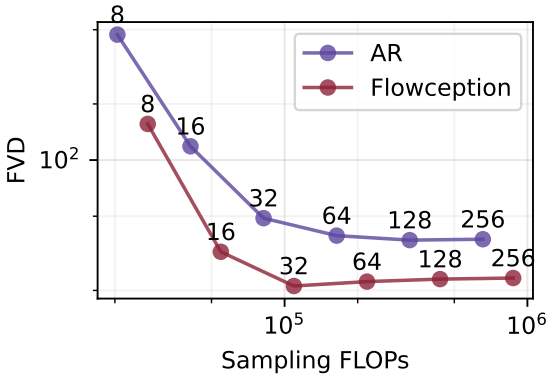

Figure 10. **Efficiency comparison.** We compare the sampling efficiency of autoregressive model (with caching) with Flowception, we plot the FVD on RealEstate10K as a function of the FLOPs used for sampling when varying the number of sampling steps.

F. Additional experiments

In this section we provide additional experimental results to complement those in the main paper.

Efficiency comparison. We perform a sweep over the number of sampling steps for both the autoregressive model and Flowception to compute how FVD performance changes as a function of the number of sampling steps and FLOPs. We report the results in Figure 10. First, we find that the autoregressive model has a somewhat lower number of FLOPs for a given number of sampling steps per frame, this is because Flowception denoise frames asynchronously so the total number of sampling steps is larger than the number of per-frame sampling steps. Second, we find that Flowception obtains significantly better FVD for a given number of FLOPs, with FVD plateauing at around 32 denoising steps per frame, where the autoregressive model continues to improve at least up to 64 steps, but without closing the gap with Flowception.

Length modeling. Here we assess the ability of Flowception to model the length distribution of the sequences in the training set. We create a toy dataset where the number of frames is either 15, 20, 25 or 30. Each sample is a 3 pixel video where the middle pixel makes a discrete jump between two pixel values, while the boundary pixels make up a gradient between two colors that moves along the boundary with constant speed in order to achieve an integer number of rotations. After training Flowception on this dataset, we compare the histogram of ground truth and generated video lengths and plot them in Figure 11. As expected the generated video lengths follow a similar distribution to the data distribution, with peaks around 15, 20, 25 and 30, while rarely generating videos with lengths outside these four modes.

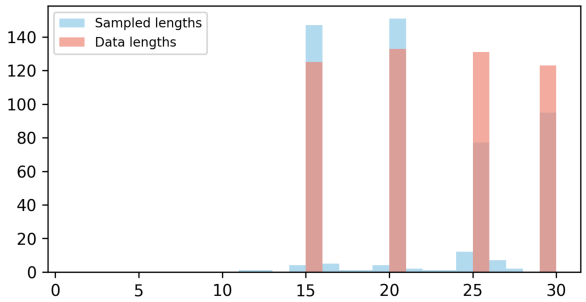

Figure 11. **Video length matching.** Our framework is able to accurately reproduce the length of videos from the toy dataset.

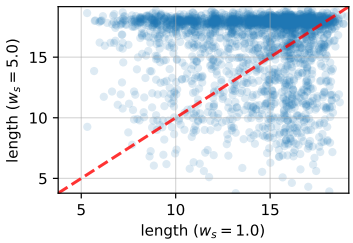

Figure 12. **Impact of rate guidance on I2V video length.** Samples obtained without guidance ($w_s = 1$, horizontal) and with guidance ($w_s = 5$, vertical) on RealEstate10K, using the same seed for each conditioning frame.

Insertion guidance. Classifier-free guidance (CFG) is commonly used to achieve better prompt alignment and image quality in diffusion and flow models [15]. Similarly, in Flowception, we can perform CFG on the insertion rates. Following CFG, we randomly drop the conditional information c during training, and define the guided update as

$$\lambda^{\text{cfg}}(X_t|c) = \lambda(X_t|c)^{w_s}\lambda(X_t)^{1-w_s}, \tag{36}$$

where $w_s \geq 1$ is the guidance scale for the insertion rate. Figure 12 presents a scatter plot comparing video lengths for I2V obtained with and without guidance, initiating generation from the same seed for each starting frame. For efficiency we limit the number of frames to 20 in this experiment. The result clearly indicates that using rate guidance ($w_s = 5$) biases the model towards generating longer videos.

Without guidance, or using lower values, we find insertion guidance to prevent problems of under-insertions which can result in a choppy transition between two frames. This is confirmed in Table 6 where motion smoothness increases while dynamic degree decreases as w_s increases.

Long video generation. While our main experiments focus on clips within the model context length, Flowception can be extended to longer videos with minimal changes. We consider two complementary approaches. First, we fine-tune the model on longer clips when available, which improves long-horizon stability while preserving short-clip performance. Second, to generate videos beyond the training/context window, we adopt a block-wise sampling strategy: we partition the target video into temporal blocks and sequentially sample each block conditioned on the previously generated frames. Concretely, we generate the first block from the text prompt, then iteratively generate the next block while conditioning on a short prefix of past frames (temporal overlap) to maintain appearance and motion continuity. This enables minute-scale rollouts without modifying the backbone architecture. We detail this sampling mechanism in Algorithm 3. In Section H we provide examples of such long rollouts.

Table 6. **Effect of rate guidance on motion smoothness.** We report dynamic degree and motion smoothness for various values of the insertion rate guidance parameter w_s.

w_s	FVD	Motion	Dynamic
1.0	21.80	99.30	78.59
2.0	22.69	99.31	78.61
5.0	25.30	99.33	77.78

Algorithm 3 Chunked Flowception generation

function CHUNKEDFLOWCEPTIONGENERATION(step size h, chunks N, window L, overlap O)
  Initialize an extended sequence X with n_{start} noisy frames (rest is padding)
  $t \leftarrow [0, \ldots, 0]$ for valid frames ▷ per-frame times
  $t_g \leftarrow 0$ ▷ global time
  Initialize chunk boundaries $\{[s_c, e_c)\}_{c=1}^{N}$ to cover valid frames (length $\leq L$, overlap O)
  while $\min\{t_i\} < 1$ **do** ▷ iterate until all frames are clean
    Extract chunk views $\{(X^{(c)}, t^{(c)})\}_{c=1}^{N}$ using current boundaries $\{[s_c, e_c)\}$
    for all chunks $c \in \{1, \ldots, N\}$ **in parallel do**
      $X^{(c)}, t^{(c)}, t_g \leftarrow$ FLOWCEPTIONSTEP$(X^{(c)}, t^{(c)}, t_g, h)$
    end for
    Write chunk updates back into X, t (blend overlap regions)
    if new frames were inserted in this step **then**
      Update chunk boundaries $\{[s_c, e_c)\}_{c=1}^{N}$ to again cover all valid frames (length $\leq L$, overlap O)
    end if
  end whilereturn X
end function

G. Broader societal impact

We recognize that our work could lead to potential negative societal impacts, as our method can help generate photorealistic videos, especially if combined with conditioning on real photos or videos. Nonetheless, our work also paves the way for efficient and flexible video generation that can be beneficial in domains such as the entertainment or film industry, as well as world modeling frameworks. As an example, animators could create coherent animations by providing a set of frames (either drawings or AI generated), which can significantly speed-up animation work flows. Our method can be used to hierarchically generate very long videos of high quality, at a lower computational cost by adopting local attention variants, thereby reducing the energy footprint of generative video models.

H. Additional qualitative examples

The following are the captions used in Figure 1:

1. A yellow taxi moving through new york streets on a rainy night.
2. A cute puppy wearing a yellow hat happily strolling in a field of roses.

In Figure 15, we provide additional qualitatives for the text-to-video generations obtained by finetuning the LTX model. Below are the captions used:

1. A compact humanoid robot with a smooth white helmet-like head, dark face panel, and a black armored body with glowing yellow/orange light accents walks through a busy New York City street at night. The scene feels like Times Square with bright billboards and crowds in the background.
2. Pretty farmer. agriculture business concept. farmer girl examines the rose crops at sunset. farmer walk agriculture lifestyle rose concept. farmer works in a field with roses at sunset.
3. Third person view of a man riding a boat in the sea towards a deserted island. A dolphin jumps in front of the boat.
4. A wild horse walking by a lake. Beautiful scene.
5. A cute bear in a knitted blue sweater and round glasses lounges on a sofa, reading a newspaper. A packed bookshelf and a crackling fireplace glow warmly in the background.
6. A lone reindeer stands on a windswept snowy mountain ridge at sunrise. The camera slowly pushes in, revealing vast icy peaks and a glowing sky while the reindeer takes a few careful steps through fresh powder.

In Figure 16, we provide examples of long generation rollouts using our chunked sampling algorithm. We used the following prompts:

1. A man wearing a black leather jacket walks through the crowded streets of Venice at noon, then arrives at a small park with a flowing water fountain.
2. A continuous cinematic shot of a wolf walking across an open field toward a small lake. The camera follows steadily from the side as the wolf moves through the grass and arrives at the water.

In Figure 13 we compare generations of Flowception and the autoregressive and Full-Sequence baselines trained on the Tai-Chi-HD dataset for 300k iterations, the generations are of 145 frames with an FPS of 16. For the autoregressive model, we observe that later frames suffer from drift due to error accumulation, hindering their quality (see,e.g., the legs). For the full-sequence model, the model struggles to accurately generate the high-frequency details of the video accurately (see, e.g., the face and foliage in the background). In contrast, Flowception results in a sharp video without error accumulation as the video progresses. In Figure 14 and Figure 19 we provide further examples of image-to-video results obtained with Flowception on the Tai-Chi-HD and RealEstate10k datasets respectively.

In Figure 17 and Figure 18, we provide additional video interpolation results obtained with Flowception on the Kinetics 600 and RealEstate10K datasets, respectively; extending the results in Figure 7 of the main paper. We always provide the first and last frames as context, plus at most two additional intermediate frames.

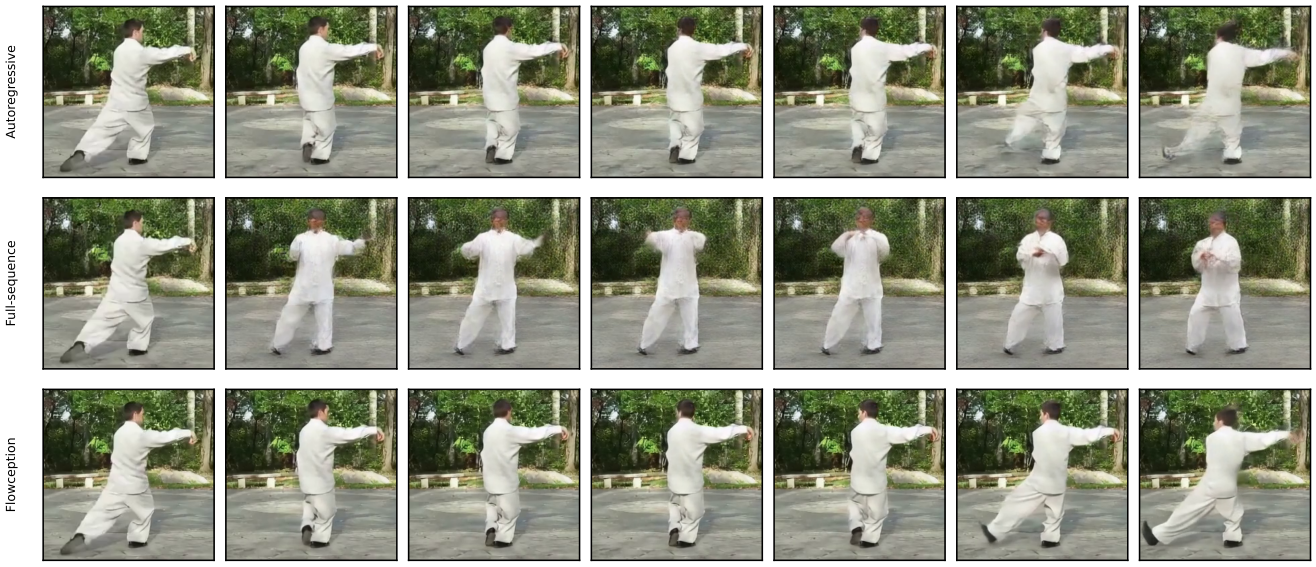

Figure 13. **Comparing different methods using models trained on the Tai-Chi-HD dataset.** Using the same input frame (left) and random seed, we compare generations with the autoregressive, full-sequence and Flowception models.

Figure 14. **Additional qualitative examples on Taichi.** Each row corresponds to a different video obtained with our method for image-to-video generation

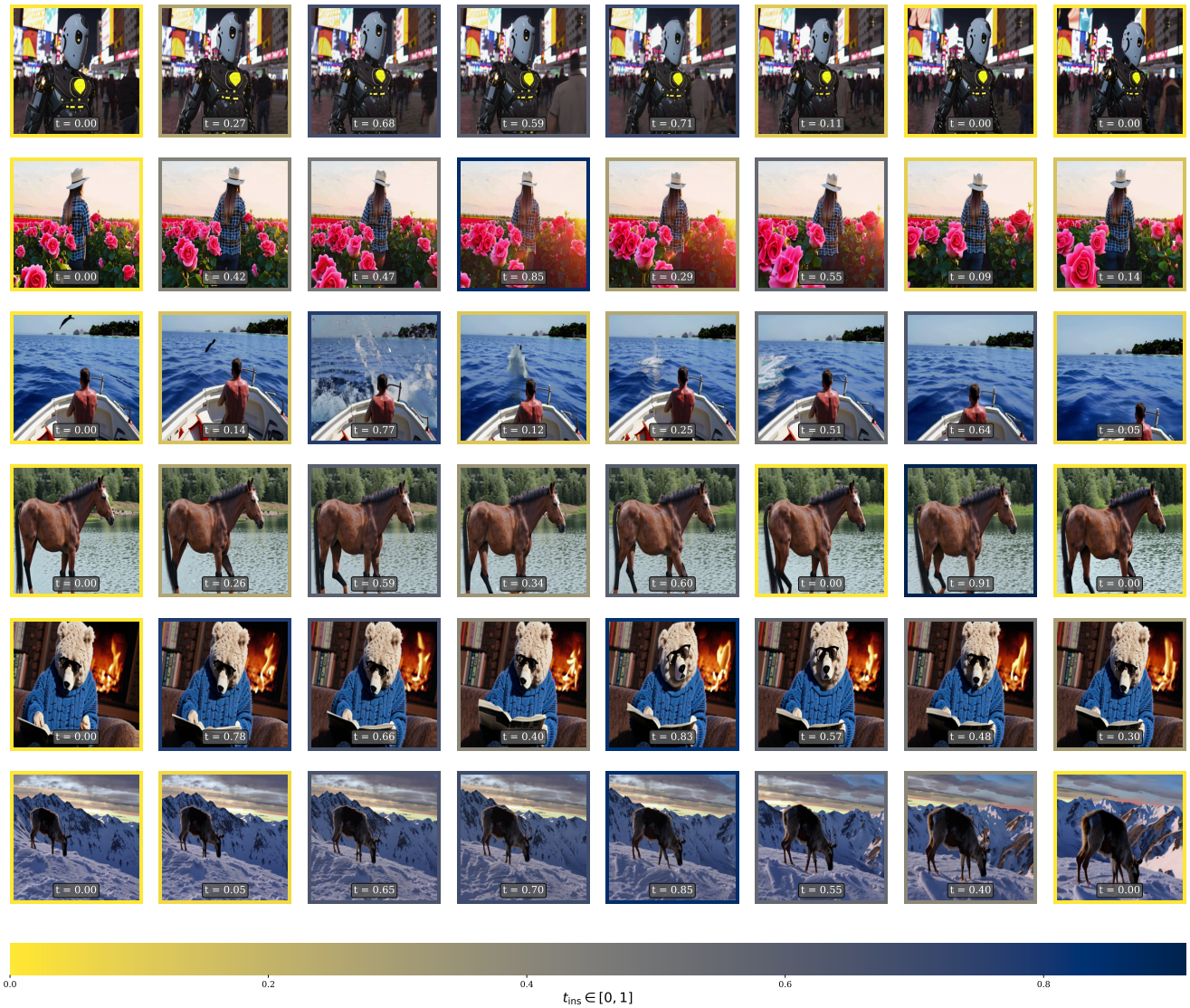

Figure 15. **Text-to-video generations with Flowception.** Additional qualitatives for the LTX-2B model finetuned on our internal data split of text-video pairs.

Figure 16. **Chunked generation.** We demonstrate results obtained with the chunked sampling algorithm (detailed above) to generate videos $4\times$ longer than what the model was trained with. This method allows us to seamlessly rollout generations for up to a minute at 24FPS. We indicate the frame index on the top left corner of every generated frame.

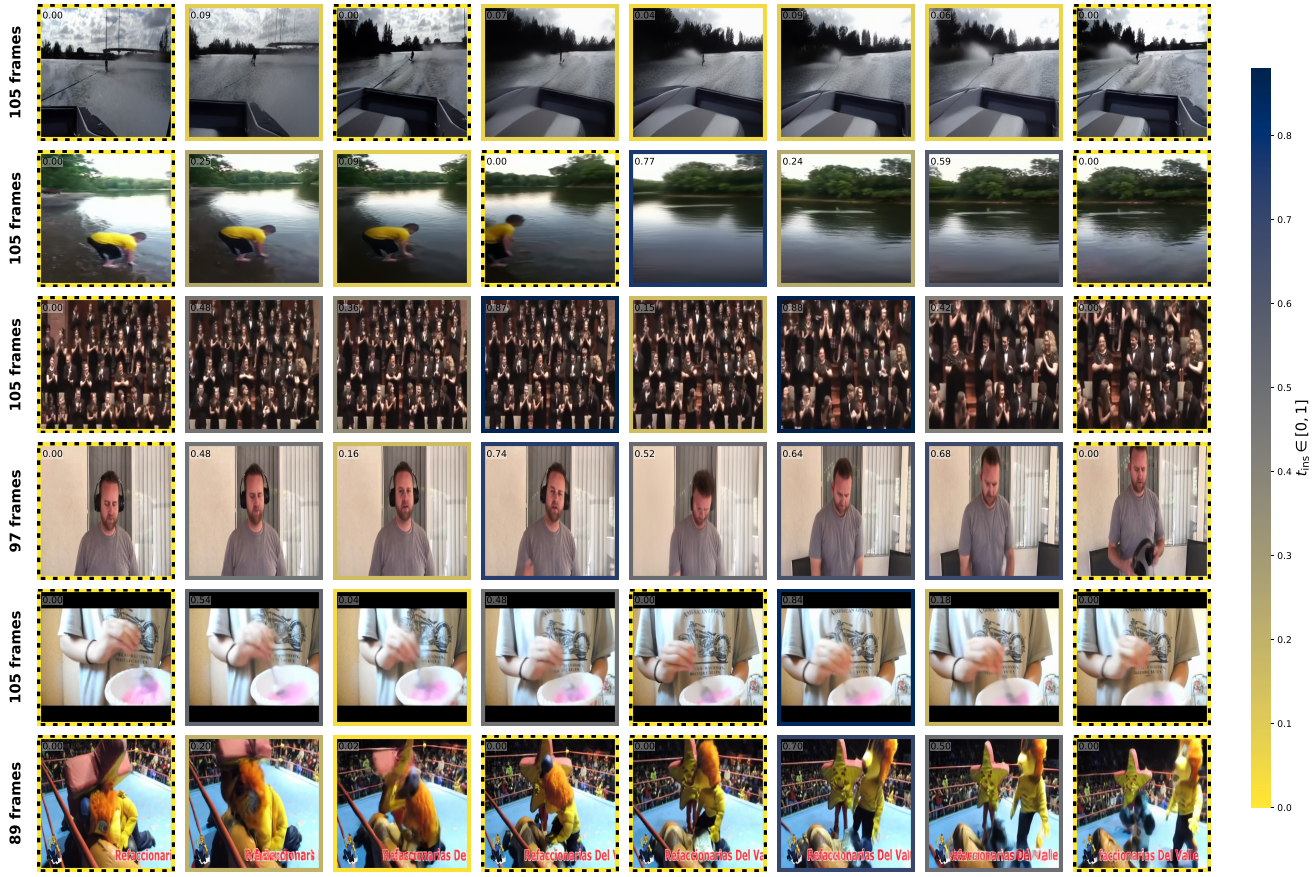

Figure 17. **Additional qualitative examples on Kinetics 600 interpolation.** Each row corresponds to a different video where the first and last frame are given and up to two extra middle frames are also given. Insertion time is highlighted in the bodrer color of each frame, context frames are highlighted with dashed lines.

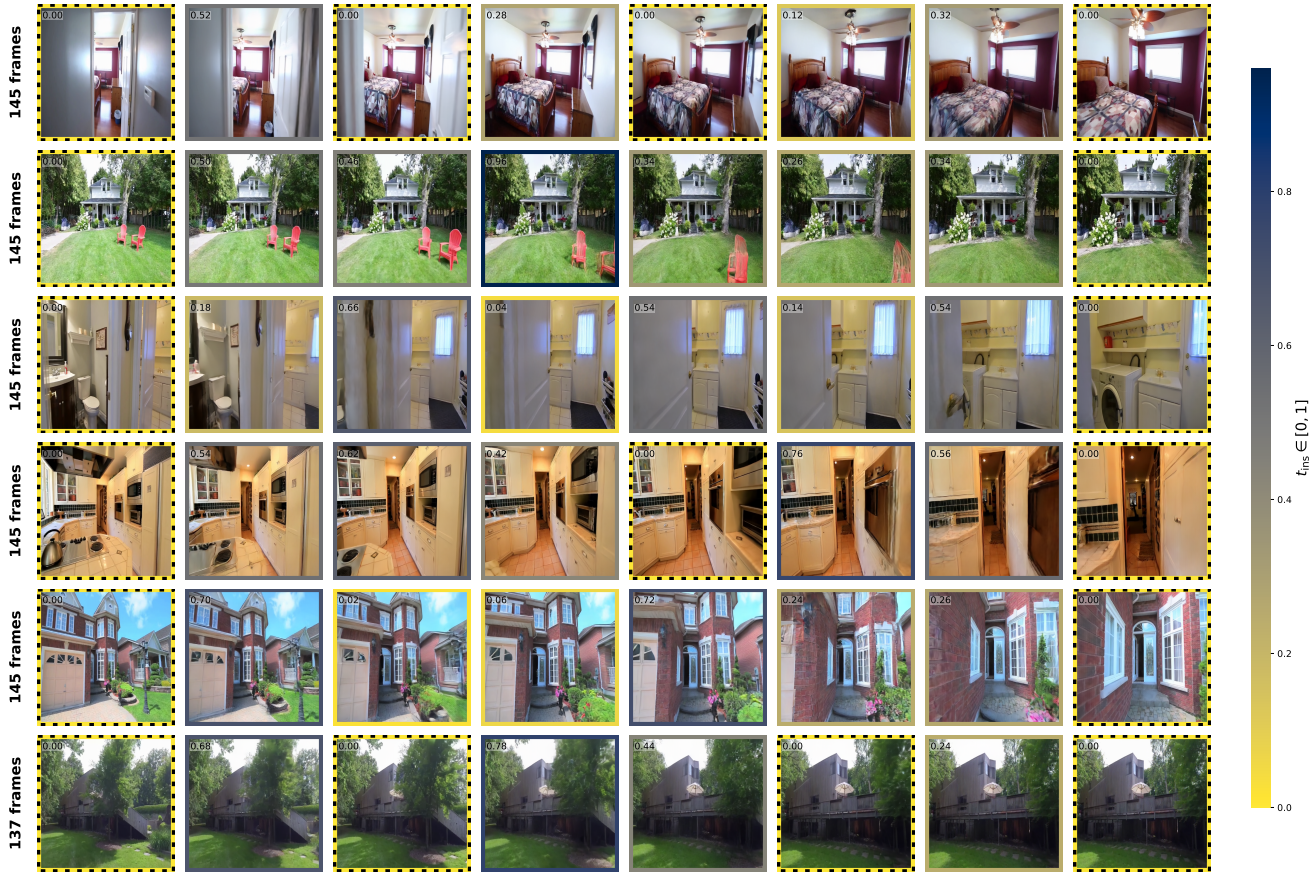

Figure 18. **Additional qualitative examples on RealEstate10K interpolation.** Each row corresponds to a different video where the first and last frame are given and up to two extra middle frames are also given. Insertion time is highlighted in the border color of each frame, context frames are highlighted with dashed lines.

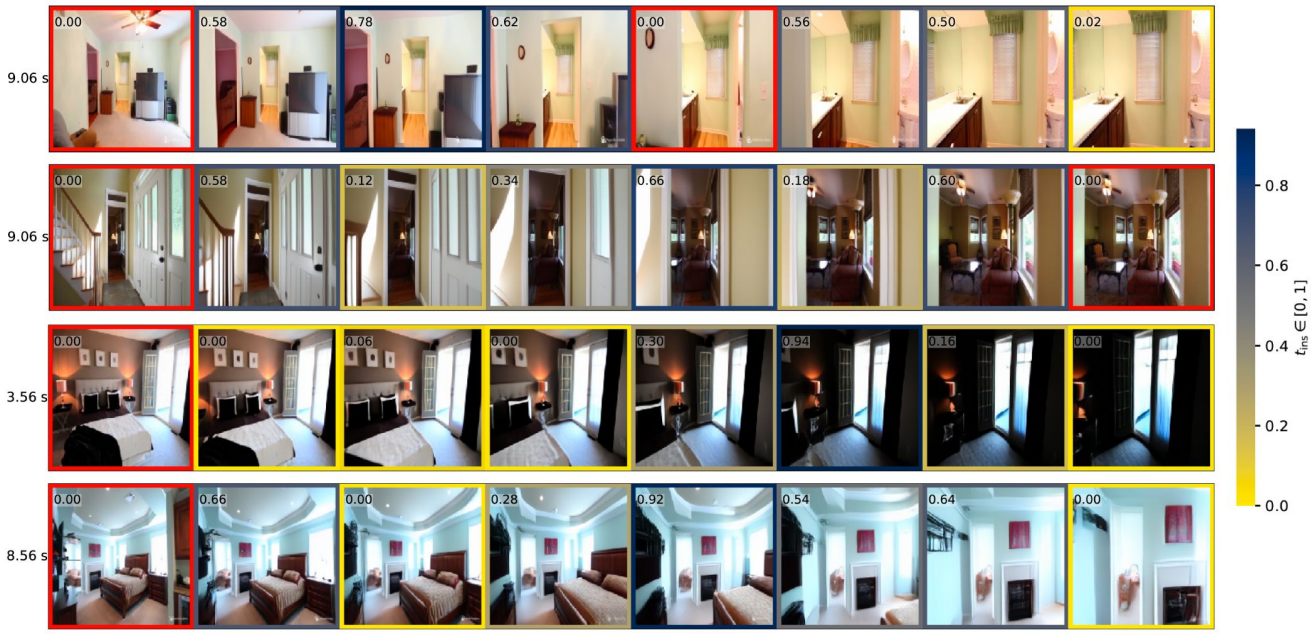

Figure 19. **Qualitative examples of Image-to-Video generation.** Using Flowception trained on the RealEstate10K dataset. First shown frame is given as context. Given the initial frame, we generate videos of at most 145 frames at 16 FPS, corresponding to 9.06 secs.

References

[1] NVIDIA: Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical AI. *arXiv preprint*, 2501.03575, 2025. 3

[2] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015. 2

[3] Tariq Berrada, Pietro Astolfi, Melissa Hall, Reyhane Askari-Hemmat, Yohann Benchetrit, Marton Havasi, Matthew Muckley, Karteek Alahari, Adriana Romero-Soriano, Jakob Verbeek, and Michal Drozdzal. On improved conditioning mechanisms and pre-training strategies for diffusion models. In *Advances in Neural Information Processing Systems*, 2024. 2

[4] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning*, 2024. 1

[5] Shengqu Cai, Ceyuan Yang, Lvmin Zhang, Yuwei Guo, Junfei Xiao, Ziyan Yang, Yinghao Xu, Zhenheng Yang, Alan Yuille, Leonidas Guibas, Maneesh Agrawala, Lu Jiang, and Gordon Wetzstein. Mixture of contexts for long video generation. In *International Conference on Learning Representations*, 2026. 3

[6] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv preprint*, 1808.01340, 2018. 6

[7] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, 2025. 3, 7

[8] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International Conference on Learning Representations*, 2024. 2

[9] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. In *International Conference on Learning Representations*, 2025. 3, 7

[10] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021. 2

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024. 1, 2, 3, 5

[12] Weichen Fan, Chenyang Si, Junhao Song, Zhenyu Yang, Yinan He, Long Zhuo, Ziqi Huang, Ziyue Dong, Jingwen He, Dongwei Pan, Yi Wang, Yuming Jiang, Yaohui Wang, Peng Gao, Xinyuan Chen, Hengjie Li, Dahua Lin, Yu Qiao, and Ziwei Liu. Vchitect-2.0: Parallel transformer for scaling up video diffusion models. *arXiv preprint*, 2501.08453, 2025. 3

[13] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. LTX-Video: Realtime video latent diffusion. *arXiv preprint*, 2501.00103, 2024. 1, 3, 5, 6, 7, 15, 17

[14] Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Variable length discrete flow matching with sequence-level edit operations. In *Advances in Neural Information Processing Systems*, 2025. 3, 12, 13, 14

[15] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021. 19

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. 2

[17] Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. In *International Conference on Learning Representations*, 2025. 12

[18] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Conference on Computer Vision and Pattern Recognition*, 2024. 6

[19] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations*, 2025. 2, 3

[20] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 1, 3

[21] Mark Levy, Bruno Di Giorgi, Floris Weers, Angelos Katharopoulos, and Tom Nickson. Controllable music production with diffusion models and guidance gradients. In *Advances in Neural Information Processing Systems*, 2023. 2

[22] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Machine Learning*, 2023. 2, 3, 5, 13

[23] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, 2024. 2

[24] John Nguyen, Marton Havasi, Tariq Berrada, Luke Zettlemoyer, and Ricky T. Q. Chen. OneFlow: Concurrent mixed-modal and interleaved generation with edit flows. *arXiv preprint*, 2510.03506, 2025. 4, 5, 12, 13, 17

[25] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, 2023. 5

[26] Xiangyu Peng, Zangwei Zheng, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, Yuhui Wang, Anbang Ye, Gang Ren, Qianran Ma, Wanying Liang, Xiang Lian, Xiwen Wu, Yuting Zhong, Zhuangyan Li, Chaoyu Gong, Guojun Lei, Leijun Cheng, Limin Zhang, Minghao Li, Ruijie Zhang, Silan Hu, Shijie Huang, Xiaokang Wang, Yuanheng Zhao, Yuqi Wang, Ziang Wei, and Yang You. Open-Sora 2.0: Training a commercial-level video generation model in $200k. *arXiv preprint*, 2503.09642, 2025. 3

[27] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024. 2, 3

[28] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie Gen: A cast of media foundation models. *arXiv preprint*, 2410.13720, 2025. 2, 18

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition*, 2022. 2

[30] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Advances in Neural Information Processing Systems*, 2019. 6

[31] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised

learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015. 2

[32] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. In *International Conference on Machine Learning*, 2025. 1, 3, 7

[33] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2

[34] Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang, Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. MAGI-1: Autoregressive video generation at scale. *arXiv preprint*, 2505.13211, 2025. 3

[35] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2019. 6

[36] 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*, 2503.20314, 2025. 1, 3, 7, 15, 17

[37] Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, and sheng zhao. Audit: Audio editing by following instructions with latent diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 2

[38] Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Jian Tong, Haodong Duan, Qipeng Guo, Jiaqi Wang, et al. VideoRoPE: What makes for good video rotary position embedding? In *International Conference on Machine Learning*, 2025. 5

[39] Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, Chong Luo, Yueyi Zhang, and Zhiwei Xiong. ART-V: Auto-regressive text-to-video generation with diffusion models. In *CVPR Workshop on Generative Models for Computer Vision*, 2024. 3

[40] Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. In *CVPR workshop on AI for Creative Visual Content Generation Editing and Understanding*, 2025. 3

[41] Lvmin Zhang and Maneesh Agrawala. Packing input frame contexts in next-frame prediction models for video generation. *arXiv preprint*, 2504.12626, 2025. 2, 3

[42] Canyu Zhao, Mingyu Liu, Wen Wang, Weihua Chen, Fan Wang, Hao Chen, Bo Zhang, and Chunhua Shen. MovieDreamer: Hierarchical generation for coherent long visual sequence. In *International Conference on Learning Representations*, 2025. 3

[43] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-Sora: Democratizing efficient video production for all. *arXiv preprint*, 2412.20404, 2024. 1, 3

[44] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 6