Disentangled Motion Modeling for Video Frame Interpolation

Jaihyun Lew[1], **Jooyoung Choi**[2], **Chaehun Shin**[2], **Dahuin Jung**[3,†], **Sungroh Yoon**[1,2,4,†]

[1]Interdisciplinary Program in AI, Seoul National University
[2]Department of Electrical and Computer Engineering, Seoul National University
[3]School of Computer Science and Engineering, Soongsil University
[4]AIIS, ASRI and INMC, Seoul National University
{fudojhl, jy_choi, chaehuny}@snu.ac.kr, dahuin.jung@ssu.ac.kr, sryoon@snu.ac.kr

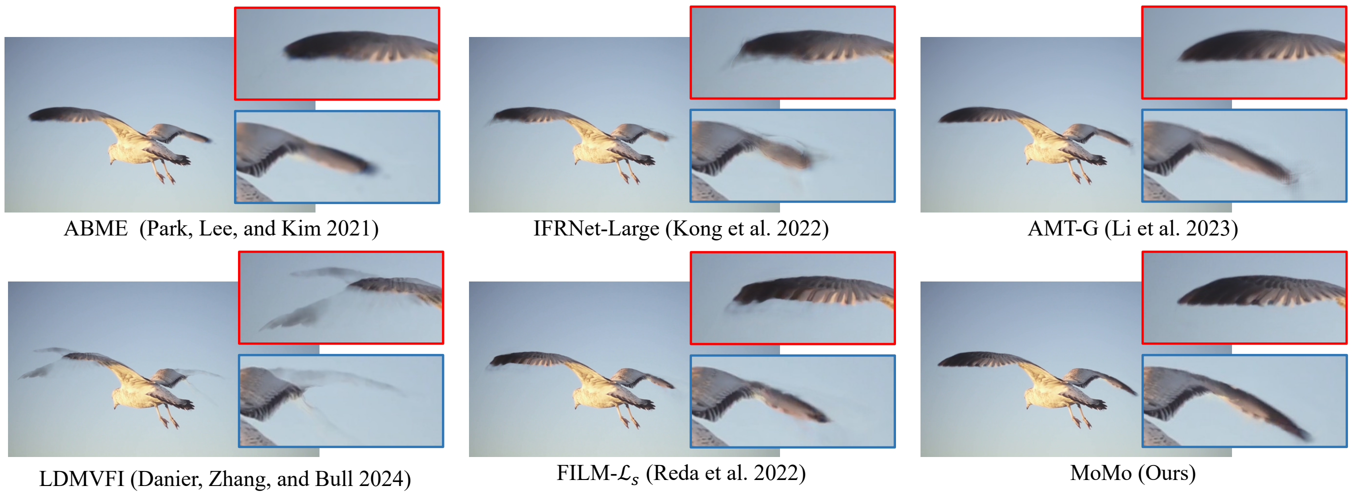

ABME (Park, Lee, and Kim 2021)
IFRNet-Large (Kong et al. 2022)
AMT-G (Li et al. 2023)
LDMVFI (Danier, Zhang, and Bull 2024)
FILM-$\mathcal{L}_s$ (Reda et al. 2022)
MoMo (Ours)

Figure 1: Video frame interpolation results of our proposed method called MoMo with comparison to state-of-the-art methods. MoMo produces the most visually pleasant result, owing to proper modeling of the intermediate motion.

Abstract

Video Frame Interpolation (VFI) aims to synthesize intermediate frames between existing frames to enhance visual smoothness and quality. Beyond the conventional methods based on the reconstruction loss, recent works have employed generative models for improved perceptual quality. However, they require complex training and large computational costs for pixel space modeling. In this paper, we introduce disentangled Motion Modeling (MoMo), a diffusion-based approach for VFI that enhances visual quality by focusing on intermediate motion modeling. We propose a disentangled two-stage training process. In the initial stage, frame synthesis and flow models are trained to generate accurate frames and flows optimal for synthesis. In the subsequent stage, we introduce a motion diffusion model, which incorporates our novel U-Net architecture specifically designed for optical flow, to generate bi-directional flows between frames. By learning the simpler low-frequency representation of motions, MoMo achieves superior perceptual quality with reduced computational demands compared to the generative modeling methods on the pixel space. MoMo surpasses state-of-the-art methods in perceptual metrics across various benchmarks, demonstrating its efficacy and efficiency in VFI.

1 Introduction

Video Frame Interpolation (VFI) is a crucial task in computer vision that aims to synthesize absent frames between existing ones in a video. It has a wide spectrum of applications, such as slow motion generation (Jiang et al. 2018), video compression (Wu, Singhal, and Krahenbuhl 2018), and animation production (Siyao et al. 2021). Its ultimate goal is to elevate the visual quality of videos through enhanced motion smoothness and image sharpness. Motions, represented by optical flows (Sun et al. 2018; Teed and Deng 2020) and realized by warping, have been central to VFI's development as recent innovations in VFI have mostly been accomplished along with advances in intermediate motion estimation (Xu et al. 2019; Chi et al. 2020; Park et al. 2020; Park, Lee, and Kim 2021; Sim, Oh, and Kim 2021; Reda et al. 2022; Jin et al. 2023).

However, these approaches often result in perceptually unsatisfying outcomes due to their reliance on L_1 or L_2 objectives, leading to high PSNR scores yet poor perceptual quality (Ledig et al. 2017; Zhang et al. 2018). To address this matter, recent advancements (Choi et al. 2020; Jiang et al. 2018; Niklaus and Liu 2020; Chen and Zwicker 2022) have explored the use of deep feature spaces to achieve improved quality in terms of human perception (Johnson, Alahi, and

[†] Corresponding authors
Code available at: https://github.com/JHLew/MoMo

Fei-Fei 2016; Zhang et al. 2018). Additionally, the integration of generative models into VFI (Danier, Zhang, and Bull 2024; Wu et al. 2024) has introduced novel pathways for improving the visual quality of videos but has primarily focused on modeling pixels or latent spaces directly, which demands high computational resources.

We introduce disentangled **Mo**tion **Mo**deling (MoMo), a perception-oriented approach for VFI, focusing on the modeling of intermediate motions rather than direct pixel generation. Here, we employ a diffusion model (Ho, Jain, and Abbeel 2020) to generate bi-directional optical flow maps, marking the first use of generative modeling for motion in VFI. We propose to disentangle the training of frame synthesis and intermediate motion prediction into a two-stage process: the initial stage includes the training of a *frame synthesis model* and fine-tuning of an *optical flow model* (Teed and Deng 2020). The frame synthesis model is designed to correctly synthesize an RGB frame given a pair of frames and their corresponding flow maps. In the subsequent stage of training, we train our *motion diffusion model*, which generates the intermediate motions the frame synthesis model uses to create the final interpolated frame during inference. In this stage of training, the optical flow model fine-tuned in the first stage serves as a teacher to provide pseudo-labels for the motion diffusion model. We also propose a novel architecture for our motion diffusion model, inspired by the nature of optical flows, enhancing both computational efficiency and performance.

Our experiments validate the effectiveness and efficiency of our proposed training scheme and architecture, demonstrating superior performance across various benchmarks in terms of perceptual metrics, with approximately 70$\times$ faster runtime compared to the existing diffusion-based VFI method (Danier, Zhang, and Bull 2024). By prioritizing the generative modeling of motions, our approach enhances visual quality, effectively addressing the core objective of VFI.

Our contributions can be summarized as follows:

- We introduce MoMo, a diffusion-based method focusing on generative modeling of bi-directional optical flows for the first time in VFI.
- We propose to disentangle the training of frame synthesis and intermediate motion modeling into a two-stage process, which are the crucial components in VFI.
- We introduce a novel diffusion model architecture suitable for optical flow modeling, boosting efficiency and quality.

2 Related Work

2.1 Flow-based Video Frame Interpolation

In deep learning-based Video Frame Interpolation (VFI), optical flow-based methods have recently become prominent, typically following a common two-stage process. First, the flows *to* or *from* the target intermediate frame is estimated, which involves warping of the input frame pair with the estimated flows. Then, a synthesis network merges the warped frames to produce the final frame. Recent advances in VFI quality have progressed with enhancements in intermediate flow predictions (Huang et al. 2022b; Kong et al. 2022; Lu et al. 2022; Zhang et al. 2023; Li et al. 2023), sparking specialized architectures to improve flow accuracy (Park et al. 2020; Park, Lee, and Kim 2021; Park, Kim, and Kim 2023). Following this direction of studies, our work aims to focus on improving intermediate flow prediction. Unlike most methods that heavily rely on reconstruction loss for end-to-end training, with optional flow distillation loss (Huang et al. 2022b; Kong et al. 2022) for stabilized training, our approach employs disentangled and direct supervision solely on flow estimation, marking an innovation in VFI research.

2.2 Perception-oriented Restoration

Conventional restoration methods in computer vision, including VFI, focused on minimizing L_1 or L_2 distances, often resulting in blurry images (Ledig et al. 2017) due to prioritizing of pixel accuracy over human visual perception. Recent studies have shifted towards deep feature spaces for reconstruction loss (Johnson, Alahi, and Fei-Fei 2016) and evaluation metrics (Zhang et al. 2018; Ding et al. 2020), demonstrating that these align better with human judgment. These approaches emphasize perceptual similarities over traditional metrics like PSNR, signaling a move towards more visually appealing, photo-realistic image synthesis.

Ever since the pioneering work of SRGAN (Ledig et al. 2017), generative models have been actively used to enhance visual quality in restoration tasks (Saharia et al. 2022b; Menon et al. 2020). The adoption of generative models has also been explored in VFI (Voleti, Jolicoeur-Martineau, and Pal 2022; Danier, Zhang, and Bull 2024; Wu et al. 2024). LDMVFI (Danier, Zhang, and Bull 2024), closely related to our work, use latent diffusion models (Rombach et al. 2022) to enhance perceptual quality. Our method aligns with such innovations but uniquely focuses on generating optical flow maps, differing from prior generative approaches that directly model the pixel space.

2.3 Diffusion Models

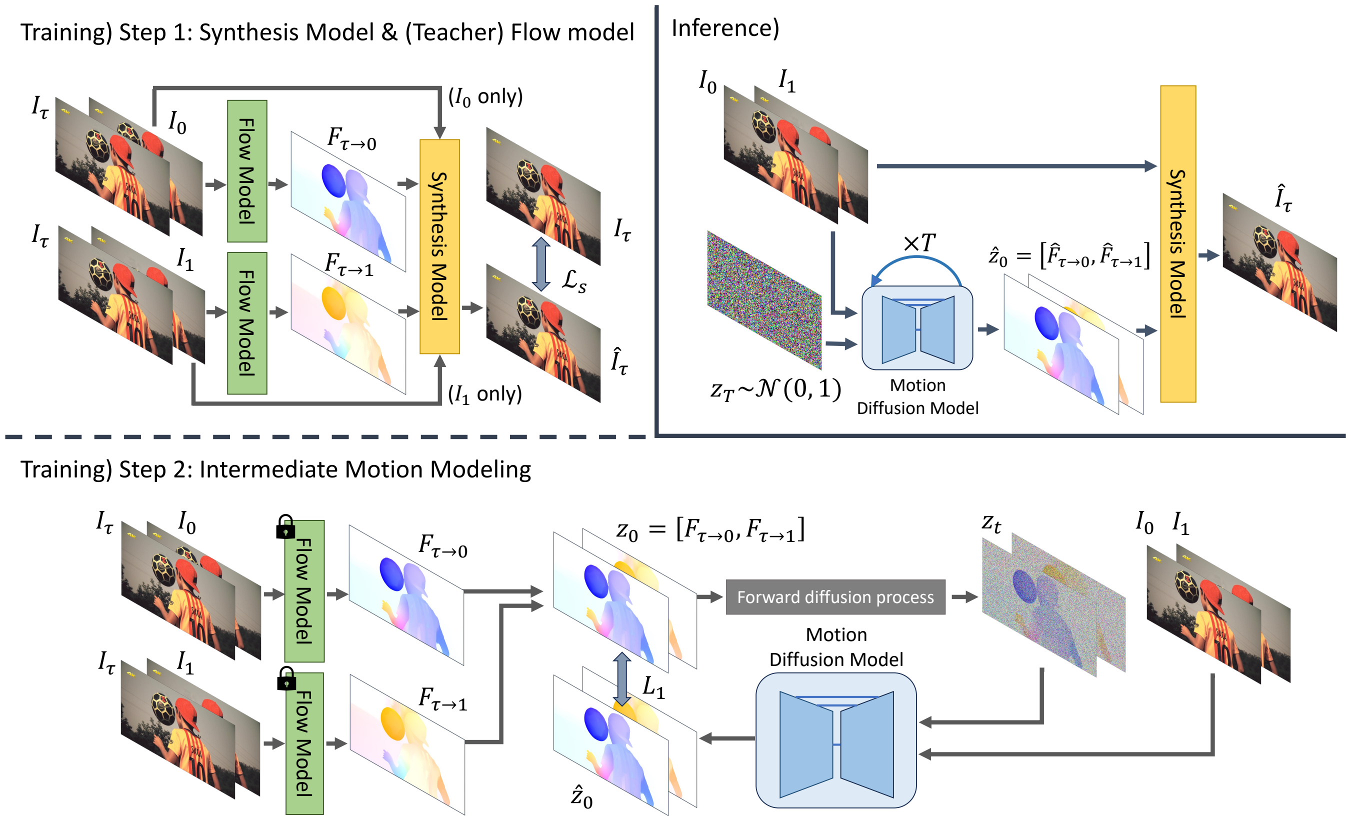
Diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020) are popular generative models that consist of forward and reverse process. Initially, the forward process incrementally adds noise to the data $\mathbf{x}_0$ over T steps via a predefined Markov chain, resulting in $\mathbf{x}_T$ that approximates a Gaussian noise. The diffused data $\mathbf{x}_t$ is obtained through forward process:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon, \tag{1}$$

where $\alpha_t \in \{\alpha_1, ..., \alpha_T\}$ is a pre-defined noise schedule. Then, the reverse process undoes the forward process by starting from Gaussian noise $\mathbf{x}_T$ and gradually denoising back to $\mathbf{x}_0$ over T steps. Diffusion models train a neural network to perform denoising at each step, by minimizing the following objective:

$$L = \mathbb{E}_{\mathbf{x}_0, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}), t\sim\mathcal{U}(1,T)} \left\| \epsilon - \epsilon_\theta\left(\mathbf{x}_t, t\right)\right\|_2 . \tag{2}$$

While a commonly used approach is to predict the noise as above (ϵ-prediction), there are some alternatives, such as $\mathbf{x}_0$-prediction (Ramesh et al. 2022) which predicts the data $\mathbf{x}_0$



Training) Step 1: Synthesis Model & (Teacher) Flow model
I_τ
I_0
Flow Model
$F_{\tau\to0}$
(I_0 only)
Synthesis Model
I_τ
$\mathcal{L}_s$
$\hat{I}_\tau$
I_1
$F_{\tau\to1}$
(I_1 only)
Inference)
I_0
I_1
$\times T$
$\hat{z}_0 = [\hat{F}_{\tau\to0}, \hat{F}_{\tau\to1}]$
Synthesis Model
$\hat{I}_\tau$
$z_T \sim \mathcal{N}(0,1)$
Motion Diffusion Model
Training) Step 2: Intermediate Motion Modeling
$z_0 = [F_{\tau\to0}, F_{\tau\to1}]$
Forward diffusion process
z_t
L_1
$\hat{z}_0$

Figure 2: Overview of our entire framework. The training procedure operates in two stages. Initially, we train a frame synthesis network and an optical flow model, with the latter providing pseudo-labels for the second stage. In the second stage of training, we focus on training a Motion Diffusion Model to predict bi-directional flow between frames. During inference, the Motion Diffusion Model generates flow fields given the input frame pair, which the frame synthesis model uses to generate the output.

itself or **v**-prediction (Salimans and Ho 2022), beneficial for numerical stability.

Diffusion model synthesizes the data in an iterative manner following the backward process, resulting in high perceptual quality of image samples or video samples (Saharia et al. 2022a; Ho et al. 2022). Further, we are motivated by optical flow modeling with diffusion models in other tasks (Saxena et al. 2023; Ni et al. 2023), and aim to leverage the benefit of diffusion models for optical flow synthesis in video frame interpolation. Although an existing work employs diffusion model for video frame interpolation task (Danier, Zhang, and Bull 2024), our method synthesizes the intermediate optical flows rather than directly synthesizing the RGB frames.

3 Method

3.1 Overview

In this paper, we focus on the goal of synthesizing an intermediate frame I_τ between consecutive frames I_0 and I_1, where $0 < \tau < 1$. Our method adopts a two-stage training scheme to disentangle the training of motion modeling and frame synthesis (Fig. 2). In the first stage, we train a frame synthesis network to synthesize an RGB frame from neighboring frames and their bi-directional flows. Then, we fine-tune the optical flow model to enhance flow quality. In the second stage, the fine-tuned flow model serves as a teacher for training the motion diffusion model. During inference, this motion diffusion model generates intermediate motion (bi-directional flow maps in specific), which the synthesis network uses to produce the final RGB frame.

3.2 Synthesis and Teacher Flow Models

We propose a synthesis network $\mathcal{S}$, designed to accurately generate an intermediate target frame using a pair of input frames and their corresponding optical flows from the target frames. Specifically, given a frame pair of I_0, I_1, and the target intermediate frame I_τ, we first use an optical flow model $\mathcal{F}$ to obtain the bi-directional flow from the target frame to the input frames:

$$F_{\tau\to i} = \mathcal{F}(I_\tau, I_i), i \in \{0, 1\}, \tag{3}$$

where i denotes the index of input frames. With the estimated flows and their corresponding frames from the target frame, we synthesize $\hat{I}_\tau$, which aims to recover the target frame I_τ.

$$\hat{I}_\tau = \mathcal{S}(I_{in}, F_\tau), \tag{4}$$

where I_{in} denotes the input frame pair $\{I_0, I_1\}$ and F_τ denotes the corresponding flow pair $\{F_{\tau\to0}, F_{\tau\to1}\}$.

We adopt pre-trained RAFT (Teed and Deng 2020) for optical flow model $\mathcal{F}$, and train the synthesis network $\mathcal{S}$ from

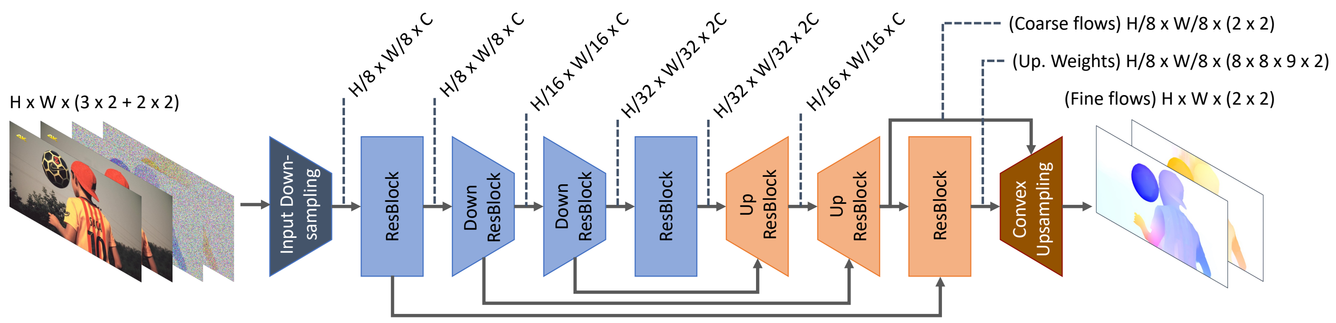


Figure 3: Architecture of our motion diffusion model. The input pair frames are downsampled to an $8\times$ smaller size and goes through a 3-level U-Net, which outputs a pair of coarse flow maps and their corresponding weight masks for upsampling. The convex upsampling layer takes the coarse flow maps and weight masks to return the full resolution flow maps.

scratch. We use an alternating optimization of two models $\mathcal{S}$ and $\mathcal{F}$. We first fix $\mathcal{F}$ to the pre-trained state, and train the synthesis network $\mathcal{S}$. Once the training of $\mathcal{S}$ converges, we freeze $\mathcal{S}$, and fine-tune $\mathcal{F}$. We fine-tune $\mathcal{F}$ so that it could provide better estimations as the teacher in the next stage of training. Note that the flow model $\mathcal{F}$ is not used during inference, but serves its purpose as the teacher for intermediate motion modeling described in Sec. 3.3.

Objective For optimization, we compute loss on the final synthesized output $\hat{I}_\tau$, with a combination of three terms. First, we use the pixel reconstruction error between the synthesized frame and the target frame: $\mathcal{L}_1 = ||I_\tau - \hat{I}_\tau||_1$. Following recent efforts (Danier, Zhang, and Bull 2024; Chen and Zwicker 2022), we adopt the LPIPS-based perceptual reconstruction loss $\mathcal{L}_p$ (Zhang et al. 2018), and also exploit the style loss $\mathcal{L}_G$ (Gatys, Ecker, and Bethge 2016) , as its effectiveness has been proved in a recent work (Reda et al. 2022). By combining the three loss terms, we define our perception-oriented reconstruction loss $\mathcal{L}_s$ for high quality synthesis:

$$\mathcal{L}_s = \lambda_1 \mathcal{L}_1 + \lambda_p \mathcal{L}_p + \lambda_G \mathcal{L}_G. \tag{5}$$

Further details on our perception-oriented reconstruction loss can be found in the Appendix.

Recurrent Synthesis We build our synthesis network $\mathcal{S}$ to be of recurrent structure, motivated by the recent trend in video frame interpolation (Sim, Oh, and Kim 2021; Jin et al. 2023; Reda et al. 2022), due to its great efficiency. The inputs I_{in} and F_τ are resized to various scales of lower-resolution, and by applying our synthesis module $\mathcal{G}$ recurrently from low-resolution and to higher resolutions, the output frame is synthesized in a coarse-to-fine manner. For our synthesis module $\mathcal{G}$, we use a simple 3-level hierarchy U-Net (Ronneberger, Fischer, and Brox 2015). Details on our recurrent synthesis procedure are described in the Appendix.

3.3 Intermediate Motion Modeling with Diffusion

With a synthesis network fixed, we focus on modeling the intermediate motions for VFI. We use our fine-tuned flow model $\mathcal{F}$ as the teacher to train our motion diffusion model $\mathcal{M}$, which generates bi-directional optical flows from the pair of input frames I_0 and I_1. Denoting concatenated flows $z_0 = \{F_{\tau\to 0}, F_{\tau\to 1}\}$, we train a diffusion model $\mathcal{M}$ by minimizing the following objective:

$$\mathcal{L}_m = \mathbb{E}_{z_0, t\sim\mathcal{U}(1,T)}[||z_0 - \mathcal{M}(z_t, t, I_0, I_1)||_1], \tag{6}$$

where z_t represents noisy flows diffused by Eq. 1. We concatenate I_0 and I_1 to z_t and keep the teacher $\mathcal{F}$ frozen during training of $\mathcal{M}$. While ϵ-prediction (Ho, Jain, and Abbeel 2020) and L_2 norm are popular choices for training diffusion-based image generative models, we found $\mathbf{x}_0$-prediction and L_1 norm to be beneficial for modeling flows. While image diffusion models utilize a U-Net architecture that employs input and output of the same resolution for noisy images that operates fully on the entire resolution, we introduce a new architecture for $\mathcal{M}$ aimed at learning optical flows and enhancing efficiency, which will be described in the following paragraph.

Architecture An overview of our proposed motion diffusion model architecture is provided at Fig. 3. In our novel diffusion model architecture designed for motion modeling, we begin by excluding attention layers, as we have found doing so saves memory without deterioration in performance. Observing that optical flow maps—our primary target—are sparse representations encoding low-frequency information, we opt to avoid the unnecessary complexity of full-resolution estimation. Consequently, we initially predict flows at $1/8$ of the input resolution and upsample them by $8\times$, a method mirroring the coarse-to-fine strategies (Teed and Deng 2020; Xu et al. 2022; Huang et al. 2022a), thus sidestepping the need for full-resolution flow estimation. We realize this by introducing *input downsampling* and *convex upsampling* (Teed and Deng 2020) into U-Net, making our architecture computationally efficient and well-suited to meet our resolution-specific needs. We elaborate them in the following paragraphs.

Input Downsampling Given an input $\{z_t, I_0, I_1\}$ of 10 channels, we downsample and encode it to $1/8$ resolution. Rather than using a single layer to directly apply on the 10 channel input, we separately apply layers $\mathcal{D}_I$ and $\mathcal{D}_z$ on the frames and the noisy flows, respectively:

$$I_0' = \mathcal{D}_I(I_0),\ I_1' = \mathcal{D}_I(I_1),\ z_t' = \mathcal{D}_z(z_t). \tag{7}$$

By sharing the parameters applied to I_0 and I_1, we could save the number of parameters required for the downsampling process, and make it invariant to the order of the two input frames. Once we obtain the downsampled features I'_0, I'_1, z'_t, we concatenate and project them to features:

$$p_t = \mathcal{D}_p([I'_0, I'_1, z'_t]), \tag{8}$$

where the projection layer $\mathcal{D}_p$ is implemented by a single 1×1 convolutional layer.

The projected features p_t is then given to a 3-level diffusion U-Net, which produces two outputs: a coarse estimation of flow maps and their corresponding upsampling weight masks. The two outputs are combined in the convex upsampling layer to obtain the final full-scale flow maps.

Convex Upsampling Given the coarse flow maps and their corresponding upsampling weight masks, both of size $H/8 \times W/8$, we attempt to upsample the coarse flows to the original $H \times W$ resolution using a weighted combination of 3×3 grid of each coarse flow neighbors, by integrating the convex upsampling layer (Teed and Deng 2020) to our architecture. Using the predicted upsampling weight masks of $8 \times 8 \times 9$ channels, we apply softmax on the weights of 9 neighboring pixels, and perform weighted summation with coarse flows to obtain the final upsampled flow map. An illustrated description is provided in the Appendix.

This method aligns with $\mathbf{x}_0$-prediction, promoting local correlations and differing from ϵ-predictions which necessitate locally independent estimations. By operating at a reduced resolution, specifically at $64\times$ smaller space, we achieve significant computation savings. Note that this upsampling layer does not involve any learnable parameters.

4 Experiments

4.1 Experiment Settings

Implementation Details We train our model on the Vimeo90k dataset (Xue et al. 2019), using random 256×256 crops with augmentations like 90° rotation, flipping, and frame order reversing. We recommend the reader to refer to the Appendix for further details.

Evaluation Protocol We evaluate on well-known VFI benchmarks: Vimeo90k (Xue et al. 2019), SNU-FILM (Choi et al. 2020), Middlebury (others-set) (Baker et al. 2011), and Xiph (Montgomery and Lars 1994; Niklaus and Liu 2020), chosen for their broad motion diversity and magnitudes. Following practices in generative models-based restoration (Danier, Zhang, and Bull 2024; Liang, Zeng, and Zhang 2022), we focus on perceptual similarity metrics LPIPS (Zhang et al. 2018) and DISTS (Ding et al. 2020), which highly correlates with human perception, for evaluation. While PSNR and SSIM are popular metrics, they have been known to differ from human perception in some aspects, sensitive to imperceptible differences in pixels and preferring blurry samples (Zhang et al. 2018). The full results can be found in the Appendix.

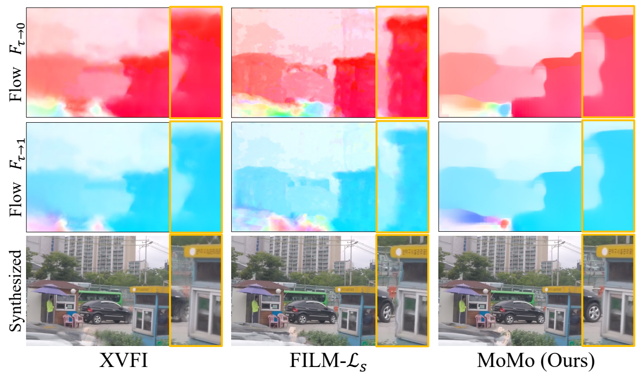


Figure 4: Visualized comparison of estimated intermediate flows against state-of-the-art methods. Our flow estimations show better-structured flow fields which leads to promising synthesis of frames.

4.2 Comparison to State-of-the-arts

Baselines We compare our method, MoMo, with state-of-the-art VFI methods which employ perception-oriented objectives in the training process: CAIN (Choi et al. 2020), FILM-$\mathcal{L}_{vgg}$, FILM-$\mathcal{L}_s$ (Reda et al. 2022), LDMVFI (Danier, Zhang, and Bull 2024) and PerVFI (Wu et al. 2024). Since there is a limited number of methods focused on perception-oriented objectives, we also include methods trained with the traditional pixel-wise reconstruction loss for comparison: XVFI (Sim, Oh, and Kim 2021), RIFE (Huang et al. 2022b), IFRNet (Kong et al. 2022), AMT (Li et al. 2023), EMA-VFI (Zhang et al. 2023), UPRNet (Jin et al. 2023).

Quantitative Results Tables 1 and 2 present our quantitative results across four benchmark datasets. MoMo achieves state-of-the-art on all four subsets of SNU-FILM, leading in both LPIPS and DISTS metrics. On Middlebury and Vimeo90k, it outperforms baselines in DISTS and closely trails FILM-$\mathcal{L}_s$ in LPIPS. MoMo also excels on both Xiph subsets, 2K and 4K, in both metrics. This highlights the effectiveness of our approach in generating well-structured optical flows for the intermediate frame through proficient intermediate motion modeling.

To support our discussion, we provide a visualization of flow estimations and the frame synthesis outcomes, with comparison to state-of-the-art algorithms at Fig. 4. Although XVFI and FILM takes advantage of the recurrent architecture tailored for flow estimations at high resolution images, they fail in well-structured flow estimations and frame synthesis. XVFI largely fails in flow estimation, which results in blurry outputs. The estimations by FILM display vague and noisy motion boundaries, especially in $F_{\tau\to1}$. Another important point to note is that the flow pair $F_{\tau\to0}$ and $F_{\tau\to1}$ of FILM do not align well with each other, causing confusion in the synthesis process.

Qualitative Results The qualitative results of MoMo with comparison to the state-of-the-art algorithms can be found at Fig. 1 and 5. In Fig. 1, MoMo reconstructs both wings with rich details, whereas methods of the top row, which greatly relies on the L_1 pixel-wise loss in training show

Method	Perception-oriented loss	FILM-easy		FILM-medium		FILM-hard		FILM-extreme	
		LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS
ABME (Park, Lee, and Kim 2021)	✗	0.0222	0.0229	0.0372	0.0344	0.0658	0.0496	0.1258	0.0747
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	0.0175	0.0181	0.0322	0.0276	0.0629	0.0414	0.1257	0.0673
IFRNet-Large (Kong et al. 2022)	✗	0.0203	0.0211	0.0321	0.0288	0.0562	0.0403	0.1131	0.0638
RIFE (Huang et al. 2022b)	✗	0.0181	0.0195	0.0317	0.0289	0.0657	0.0443	0.1390	0.0764
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	0.0184	0.0217	0.0315	0.0316	0.0568	0.0441	0.1060	0.0632
AMT-G (Li et al. 2023)	✗	0.0325	0.0312	0.0447	0.0395	0.0680	0.0506	0.1128	0.0686
EMA-VFI (Zhang et al. 2023)	✗	0.0186	0.0204	0.0325	0.0318	0.0579	0.0457	0.1099	0.0671
UPRNet-LARGE (Jin et al. 2023)	✗	0.0182	0.0203	0.0334	0.0327	0.0612	0.0475	0.1109	0.0672
CAIN (Choi et al. 2020)	✓	0.0197	0.0229	0.0375	0.0347	0.0885	0.0606	0.1790	0.1042
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	0.0123	0.0128	0.0219	0.0183	0.0443	0.0282	0.0917	0.0471
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	0.0120	0.0124	0.0213	0.0177	0.0429	0.0268	0.0889	0.0448
LDMVFI (Danier, Zhang, and Bull 2024)	✓	0.0145	0.0130	0.0284	0.0219	0.0602	0.0379	0.1226	0.0651
PerVFI (Wu et al. 2024)	✓	0.0142	0.0124	0.0245	0.0181	0.0561	0.0635	0.0902	0.0448
MoMo (Ours)	✓	**0.0111**	**0.0102**	**0.0202**	**0.0155**	**0.0419**	**0.0252**	**0.0872**	**0.0433**

Table 1: Quantitative experiments on the SNU-FILM benchmark (Choi et al. 2020). The best results are in **bold**, and the second best is underlined, respectively. Our method outperforms existing methods on all four subsets.

Method	Perception-oriented loss	Middlebury		Vimeo90k		Xiph-2K		Xiph-4K	
		LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS
ABME (Park, Lee, and Kim 2021)	✗	0.0290	0.0325	0.0213	0.0353	0.1071	0.0581	0.2361	0.1108
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	0.0169	0.0244	0.0229	0.0354	0.0844	0.0418	0.1835	0.0779
IFRNet-Large (Kong et al. 2022)	✗	0.0285	0.0366	0.0189	0.0325	0.0681	0.0372	0.1364	0.0665
RIFE (Huang et al. 2022b)	✗	0.0162	0.0228	0.0223	0.0356	0.0918	0.0481	0.2072	0.0915
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	0.0173	0.0246	0.0197	0.0343	0.0906	0.0510	0.1841	0.0884
AMT-G (Li et al. 2023)	✗	0.0486	0.0533	0.0195	0.0351	0.1061	0.0563	0.2054	0.1005
EMA-VFI (Zhang et al. 2023)	✗	0.0151	0.0218	0.0196	0.0343	0.1024	0.0550	0.2258	0.1049
UPRNet-LARGE (Jin et al. 2023)	✗	0.0150	0.0209	0.0201	0.0342	0.1010	0.0553	0.2150	0.1017
CAIN (Choi et al. 2020)	✓	0.0254	0.0383	0.0306	0.0483	0.1025	0.0533	0.2229	0.0980
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	0.0096	0.0148	0.0137	0.0229	0.0355	0.0238	0.0754	0.0406
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	**0.0093**	0.0140	**0.0131**	0.0224	0.0330	0.0237	0.0703	0.0385
LDMVFI (Danier, Zhang, and Bull 2024)	✓	0.0195	0.0261	0.0233	0.0327	0.0420	0.0163	0.0859	0.0359
PerVFI (Wu et al. 2024)	✓	0.0142	0.0163	0.0179	0.0248	0.0381	0.0153	0.0858	0.0331
MoMo (Ours)	✓	0.0094	**0.0126**	0.0136	**0.0203**	**0.0300**	**0.0119**	**0.0631**	**0.0274**

Table 2: Quantitative experiments on the three benchmarks, Middlebury (Baker et al. 2011), Vimeo90k (Xue et al. 2019) and Xiph-2K,4K (Montgomery and Lars 1994; Niklaus and Liu 2020). The best results are in **bold**, and the second best is underlined, respectively.

blurry results. Moreover, our result also outperforms state-of-the-art models designed particularly for perceptual quality, LDMVFI and FILM-$\mathcal{L}_s$, with well-structurized synthesis of both wings. Fig. 5 present additional results obtained from the 'extreme' subset of SNU-FILM and Xiph-4K set. MoMo consistently shows a superior visual quality, with less artifacts and well-structured objects. The interpolated video samples are provided in the supplementary material.

4.3 Ablation Studies

We conduct ablation studies to verify the effects of our design choices. We use the 'hard' subset of SNU-FILM dataset, unless mentioned otherwise. We start by studying the effects of the teacher optical flow model, used for training the motion diffusion model. We then experiment on the number of denoising steps used at inference time. Lastly, we study on the design choices in diffusion architecture.

Optical Flow Teacher We conduct a study on the teacher optical flow model. We choose RAFT (Teed and Deng 2020) as the teacher model, the state-of-the-art model for optical flow estimation. We test the teacher model of three different weights: 1) the default off-the-shelf weights provided by the Torchvision library (maintainers and contributors 2016). 2) Initialized with the pre-trained weights, weights trained jointly with our synthesis network in an end-to-end manner. 3) Optimized in an alternating manner with the synthesis network, initialized from pre-trained weights, as described in Sec. 3.2. We use these three different versions of RAFT as the teacher for the experiment.

The ablation study summarized in Table 4 shows that fine-tuning the flow model $\mathcal{F}$ after training the synthesis model $\mathcal{G}$ is the most effective. Fine-tuning of $\mathcal{F}$ enhances flow esti-

Data	Prediction	Architecture	LPIPS	DISTS	TFLOPs	Params. (M)	R-time (ms)
Latent	ϵ	Standard U-Net (Danier, Zhang, and Bull 2024)	0.0601	0.0379	3.25	439.0	10283.51
Flow	ϵ	Standard U-Net	0.4090	0.2621	8.08	71.1	603.64
Flow	$\mathbf{x}_0$	Standard U-Net	0.0460	0.0295	8.08	71.1	603.64
Flow	$\mathbf{x}_0$ (weighted)	Convex-Up U-Net (Ours)	0.0463	0.0298	**1.12**	**73.6**	**145.49**
Flow	$\mathbf{x}_0$	Convex-Up U-Net (Ours)	**0.0425**	**0.0257**	**1.12**	**73.6**	**145.49**
Flow	$\mathbf{x}_0$	Convex-Up U-Net (Ours) + Longer Training	0.0419	0.0252	1.12	73.6	145.49

Table 3: Ablation study on our motion diffusion model. Our design choice reaches the best performance with minimal computational needs and fastest runtime. The first row includes our baseline, LDMVFI (Danier, Zhang, and Bull 2024), for reference.

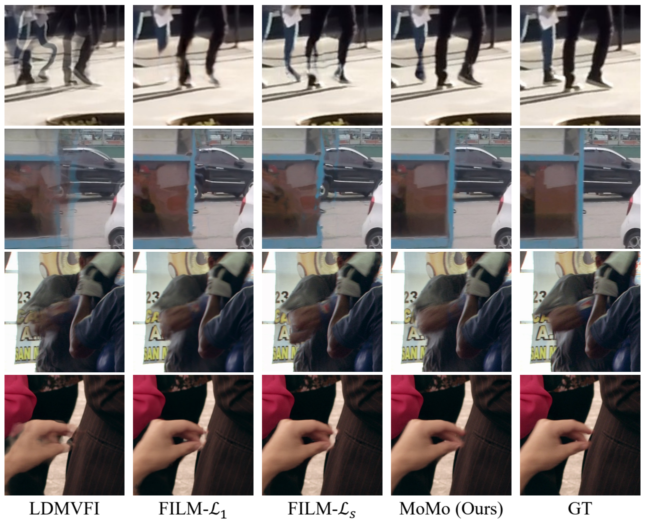


Figure 5: Qualitative comparison against state-of-the-art methods on 'extreme' subset of SNU-FILM and Xiph-4K. Our results show the least artifacts and generate well-structured images.

Teacher	Fine-tune $\mathcal{F}$	Train $\mathcal{S}$	LPIPS	DISTS
Pre-trained	✗	✗	0.0445	0.0284
End-to-End	✓	✓	0.0475	0.0287
Alternating (Ours)	✓	✗	**0.0419**	**0.0252**

Table 4: Experiments on the teacher flow model. We use RAFT (Teed and Deng 2020) with three different weights. The results show that alternating optimization, fine-tuning the flow model with $\mathcal{S}$ fixed, to be the most effective.

# of steps	LPIPS	DISTS
1 step ($\approx$ non-diffusion)	0.0892	0.0452
8 step (default)	**0.0872**	**0.0433**
20 step	0.0872	0.0433
50 step	0.0874	0.0435

Table 5: Experiment on the number of denoising steps for inference (on SNU-FILM-extreme). Our experiments show that about 8 steps is enough, and use of more steps exceeding this does not lead to a notable improvement.

mation and suitability for synthesis tasks (Xue et al. 2019). However, end-to-end training can cause the synthesis model to depend too heavily on estimated flows, risking inaccuracies from the motion diffusion model. Our results highlight that sequential training of the synthesis model and the flow estimator ensures optimal performance.

Diffusion Architecture In our ablation study, detailed in Table 3, we assess our motion diffusion model using the standard timestep-conditioned U-Net architecture (`UNet2DModel`) from the diffusers library (von Platen et al. 2022), alongside ϵ- and $\mathbf{x}_0$-prediction types. Contrary to the common preference for ϵ-prediction in diffusion models, our motion diffusion model favors $\mathbf{x}_0$-prediction.

We also explore our coarse-to-fine estimation using convex upsampling. This approach reduces computational costs and improves performance. Given that our architecture predicts values with a strong correlation between neighboring pixels, ϵ-prediction, which samples noise independently, proves less suitable. We experiment with a SNR-weighted $\mathbf{x}_0$-prediction (Salimans and Ho 2022), to make it equivalent to ϵ-prediction loss. Nonetheless, $\mathbf{x}_0$-prediction consistently outperforms, validating our architectural decisions.

Despite having a similar number of parameters as the standard U-Net, our Convex Upsampling U-Net significantly reduces floating point operations (FLOPs) by about $7.2\times$. Runtime tests on a NVIDIA 32GB V100 GPU for 256×448 resolution frames—averaged over 100 iterations—reveal that our Convex-Up U-Net processes frames in approximately 145.49 ms each, achieving a $4.15\times$ speedup over the standard U-Net and an $70\times$ faster inference speed than the LDMVFI baseline. This efficiency is attributed to our model's efficient architecture and notably fewer denoising steps.

Effectiveness of Diffusion Table 5 shows the effect of number of denoising steps in motion generation, experimented on the 'extreme' subset of SNU-FILM. We observe consistent improvement with more number of steps up to 8, with more steps not markedly improving performance. In contrast to image diffusion models (Rombach et al. 2022) and LDMVFI (Danier, Zhang, and Bull 2024) which requires over 50 steps, our method delivers satisfactory outcomes with far fewer steps, cutting down on both runtime and computational expenses. This is likely due to the simpler nature of flow representations compared to RGB pixels. This experiment shows the effectiveness of using diffusion models in motion modeling, as use of multiple steps guarantees better motion predictions.

5 Conclusion

In this paper, we proposed MoMo, a disentangled motion modeling framework for perceptual video frame interpolation. Our approach mainly focuses on modeling the intermediate motions between frames, with explicit supervision on the motions only. We introduced motion diffusion model, which generates intermediate bi-directional flows necessary to synthesize the target frame with a novel architecture tailored for optical flow generation, which greatly improves both performance and computational efficiency. Extensive experiments confirm that our method achieve state-of-the-art quality on multiple benchmarks.

Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A3B1077720), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(No. 2022R1A5A7083908), and the BK21 FOUR program of the Education and the Research Program for Future ICT Pioneers, Seoul National University in 2024.

References

Baker, S.; Scharstein, D.; Lewis, J.; Roth, S.; Black, M. J.; and Szeliski, R. 2011. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92: 1–31.

Chen, S.; and Zwicker, M. 2022. Improving the perceptual quality of 2d animation interpolation. In *European Conference on Computer Vision*, 271–287. Springer.

Chi, Z.; Mohammadi Nasiri, R.; Liu, Z.; Lu, J.; Tang, J.; and Plataniotis, K. N. 2020. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, 107–123. Springer.

Choi, M.; Kim, H.; Han, B.; Xu, N.; and Lee, K. M. 2020. Channel Attention Is All You Need for Video Frame Interpolation. In *AAAI*.

Danier, D.; Zhang, F.; and Bull, D. 2024. Ldmvfi: Video frame interpolation with latent diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 1472–1480.

Ding, K.; Ma, K.; Wang, S.; and Simoncelli, E. P. 2020. Image Quality Assessment: Unifying Structure and Texture Similarity. *CoRR*, abs/2004.07728.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2414–2423.

Ho, J.; Chan, W.; Saharia, C.; Whang, J.; Gao, R.; Gritsenko, A.; Kingma, D. P.; Poole, B.; Norouzi, M.; Fleet, D. J.; et al. 2022. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Huang, Z.; Shi, X.; Zhang, C.; Wang, Q.; Cheung, K. C.; Qin, H.; Dai, J.; and Li, H. 2022a. Flowformer: A transformer architecture for optical flow. In *European Conference on Computer Vision*, 668–685. Springer.

Huang, Z.; Zhang, T.; Heng, W.; Shi, B.; and Zhou, S. 2022b. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision*, 624–642. Springer.

Jiang, H.; Sun, D.; Jampani, V.; Yang, M.-H.; Learned-Miller, E.; and Kautz, J. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9000–9008.

Jin, X.; Wu, L.; Chen, J.; Chen, Y.; Koo, J.; and Hahm, C.-h. 2023. A Unified Pyramid Recurrent Network for Video Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1578–1587.

Johnson, J.; Alahi, A.; and Fei-Fei, L. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, 694–711. Springer.

Kong, L.; Jiang, B.; Luo, D.; Chu, W.; Huang, X.; Tai, Y.; Wang, C.; and Yang, J. 2022. Ifrnet: Intermediate feature refine network for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1969–1978.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690.

Li, Z.; Zhu, Z.-L.; Han, L.-H.; Hou, Q.; Guo, C.-L.; and Cheng, M.-M. 2023. AMT: All-Pairs Multi-Field Transforms for Efficient Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9801–9810.

Liang, J.; Zeng, H.; and Zhang, L. 2022. Details or artifacts: A locally discriminative learning approach to realistic image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5657–5666.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lu, L.; Wu, R.; Lin, H.; Lu, J.; and Jia, J. 2022. Video frame interpolation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3532–3542.

maintainers, T.; and contributors. 2016. TorchVision: PyTorch's Computer Vision library. https://github.com/pytorch/vision.

Menon, S.; Damian, A.; Hu, S.; Ravi, N.; and Rudin, C. 2020. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2437–2445.

Montgomery, C.; and Lars, H. 1994. Xiph. org video test media (derf's collection). *Online, https://media. xiph. org/video/derf*, 6.

Ni, H.; Shi, C.; Li, K.; Huang, S. X.; and Min, M. R. 2023. Conditional Image-to-Video Generation with Latent Flow Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18444–18455.

Niklaus, S.; and Liu, F. 2020. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5437–5446.

Park, J.; Kim, J.; and Kim, C.-S. 2023. BiFormer: Learning Bilateral Motion Estimation via Bilateral Transformer for 4K Video Frame Interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1568–1577.

Park, J.; Ko, K.; Lee, C.; and Kim, C.-S. 2020. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, 109–125. Springer.

Park, J.; Lee, C.; and Kim, C.-S. 2021. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14539–14548.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2): 3.

Reda, F.; Kontkanen, J.; Tabellion, E.; Sun, D.; Pantofaru, C.; and Curless, B. 2022. FILM: Frame Interpolation for Large Motion. In *European Conference on Computer Vision (ECCV)*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 234–241. Springer.

Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Denton, E. L.; Ghasemipour, K.; Gontijo Lopes, R.; Karagol Ayan, B.; Salimans, T.; et al. 2022a. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35: 36479–36494.

Saharia, C.; Ho, J.; Chan, W.; Salimans, T.; Fleet, D. J.; and Norouzi, M. 2022b. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4): 4713–4726.

Salimans, T.; and Ho, J. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*.

Saxena, S.; Herrmann, C.; Hur, J.; Kar, A.; Norouzi, M.; Sun, D.; and Fleet, D. J. 2023. The Surprising Effectiveness of Diffusion Models for Optical Flow and Monocular Depth Estimation. *arXiv preprint arXiv:2306.01923*.

Sim, H.; Oh, J.; and Kim, M. 2021. Xvfi: extreme video frame interpolation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 14489–14498.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Siyao, L.; Zhao, S.; Yu, W.; Sun, W.; Metaxas, D.; Loy, C. C.; and Liu, Z. 2021. Deep animation video interpolation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6587–6595.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265. PMLR.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8934–8943.

Teed, Z.; and Deng, J. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, 402–419. Springer.

Voleti, V.; Jolicoeur-Martineau, A.; and Pal, C. 2022. MCVD-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in Neural Information Processing Systems*, 35: 23371–23385.

von Platen, P.; Patil, S.; Lozhkov, A.; Cuenca, P.; Lambert, N.; Rasul, K.; Davaadorj, M.; and Wolf, T. 2022. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers.

Wu, C.-Y.; Singhal, N.; and Krahenbuhl, P. 2018. Video compression through image interpolation. In *Proceedings of the European conference on computer vision (ECCV)*, 416–431.

Wu, G.; Tao, X.; Li, C.; Wang, W.; Liu, X.; and Zheng, Q. 2024. Perception-Oriented Video Frame Interpolation via Asymmetric Blending. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2753–2762.

Xu, H.; Zhang, J.; Cai, J.; Rezatofighi, H.; and Tao, D. 2022. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8121–8130.

Xu, X.; Siyao, L.; Sun, W.; Yin, Q.; and Yang, M.-H. 2019. Quadratic video interpolation. *Advances in Neural Information Processing Systems*, 32.

Xue, T.; Chen, B.; Wu, J.; Wei, D.; and Freeman, W. T. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127: 1106–1125.

Zhang, G.; Zhu, Y.; Wang, H.; Chen, Y.; Wu, G.; and Wang, L. 2023. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5682–5692.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

A Implementation Details

A.1 Recurrent Synthesis

We build our synthesis network $\mathcal{S}$ to be of recurrent structure, motivated by the recent trend in video frame interpolation (Sim, Oh, and Kim 2021; Jin et al. 2023; Reda et al. 2022), due to its great efficiency. Let the number of recurrent process be $L-1$, and $\mathcal{S}$ can be expressed as recurrent application of a synthesis process $\mathcal{P}$:

$$\mathcal{S}(I_{in}, F_\tau) = \mathcal{P}^0(\cdots \mathcal{P}^{L-1}(\hat{I}^L_\tau, I^{L-1}_{in}, F^{L-1}_\tau)\cdots, I^0_{in}, F^0_\tau), \tag{9}$$

where I^l_{in} denotes the input image pair I_0, I_1 downsampled by a factor of $2^l\times$, and F^l_τ denotes the flow map pair downsampled likewise.

Our process $\mathcal{P}^l$ at level l is described as follows. $\mathcal{P}^l$ takes three components as the input: 1) frame $\hat{I}^{l+1}_\tau$, synthesized from the previous level $l+1$, 2) downsampled input frame pair $I^l_{in} = \{I^l_0, I^l_1\}$, and 3) the downsampled flow maps $F_\tau = \{F_{\tau\to 0}, F_{\tau\to 1}\}$. First, using the input frame pair $I^l_{in} = \{I^l_0, I^l_1\}$ and its corresponding flow pair $F_\tau = \{F_{\tau\to 0}, F_{\tau\to 1}\}$, we perform backward-warping ($\overleftarrow{\omega}$) on the two frames with their corresponding flows:

$$I^l_{\tau\leftarrow i} = \overleftarrow{\omega}(I^l_i, F^l_{\tau\to i}), i \in \{0, 1\}. \tag{10}$$

Next, we take frame $\hat{I}^{l+1}_\tau$ and use bicubic upsampling to match the size of level l, denoted as $\hat{I}^{l+1\to l}_\tau$. The upsampled frame $\hat{I}^{l+1\to l}_\tau$, along with the two warped frames $I^l_{\tau\leftarrow 0}, I^l_{\tau\leftarrow 1}$ and their corresponding flow maps $F^l_{\tau\to 0}, F^l_{\tau\to 1}$ are given to the synthesis module $\mathcal{G}$, which outputs a 4 channel output — 1 channel occlusion mask M_0 to blend $I_{\tau\leftarrow 0}$ and $I_{\tau\leftarrow 1}$, and 3 channel residual RGB values $\Delta\hat{I}_\tau$:

$$M^l_0, \Delta\hat{I}^l_\tau = \mathcal{G}(\hat{I}^{l+1\to l}_\tau, I^l_{\tau\leftarrow 0}, I^l_{\tau\leftarrow 1}, F^l_{\tau\to 0}, F^l_{\tau\to 1}). \tag{11}$$

Using these outputs, we obtain the output of $\mathcal{P}^l$, the synthesized frame at level l:

$$\mathcal{P}^l(\hat{I}^{l+1}_\tau, I^l_{in}, F^l_\tau) = I^l_{\tau\leftarrow 0}\odot M^l_0 + I^l_{\tau\leftarrow 1}\odot(1-M^l_0) + \Delta\hat{I}^l_\tau. \tag{12}$$

Note that $\hat{I}^L_\tau$ is not available for level $\mathcal{P}^{L-1}$, since it is of the highest level. Therefore we equally blend the two warped frames at level $l = L-1$ as a starting point: $\hat{I}^{L\to L-1}_\tau = I^{L-1}_{\tau\leftarrow 0}\odot 0.5 + I^{L-1}_{\tau\leftarrow 1}\odot 0.5$.

A.2 Perception-oriented Loss

As mentioned in Sec. 3.2, we specify the objective function we use for stage 1 training. Along with the L_1 pixel reconstruction loss, we use an LPIPS-based perceptual loss, which computes the L_2 distance in the deep feature space of AlexNet (Krizhevsky, Sutskever, and Hinton 2012). This loss is well-known for high correlation with human judgements. Next, we exploit the style loss (Gatys, Ecker, and Bethge 2016) $\mathcal{L}_G$ as its effectiveness has been proved in a recent work (Reda et al. 2022). This loss computes the L_2 distance of feature correlations extracted from the VGG-19 network (Simonyan and Zisserman 2014):

$$\mathcal{L}_G = \frac{1}{N}\sum_{n=1}^{N}\alpha_n||G_n(I_\tau) - G_n(\hat{I}_\tau)||_2. \tag{13}$$

Here, α_n denotes the weighting hyper-parameter of the n-th selected layer. Denoting the feature map of frame I_τ extracted from n-th selected layer of the VGG (Simonyan and Zisserman 2014) network as $\phi_n(I_\tau) \in \mathbb{R}^{H\times W\times C}$, the Gram matrix of frame I_τ at the n-th feature space, $G_n(I_\tau) \in \mathbb{R}^{C\times C}$ can be acquired as follows:

$$G_n(I_\tau) = \phi_n(I_\tau)^\top\phi_n(I_\tau). \tag{14}$$

Likewise, the Gram matrix of our synthesized frame, $G_n(\hat{I}_\tau)$, could be computed by substituting I_τ with $\hat{I}_\tau$ in Eq. 14.

A.3 Training Details

We elaborate on our training details mentioned in Sec. 4.1.

Stage 1 Training We employ the AdamW optimizer (Loshchilov and Hutter 2017), setting the weight decay to 10^{-4} and the batch size to 32 both in the training process of $\mathcal{G}$ and $\mathcal{F}$. We train the synthesis model $\mathcal{G}$ for a total of 200 epochs, with a fixed learning rate of 2×10^{-4}. For the first 150 epochs, we set the hyper-parameters to $\lambda_1 = 1, \lambda_p = 0, \lambda_G = 0$. After that, we use $\lambda_1 = 1, \lambda_p = 1, \lambda_G = 20$ for the last 50 epochs. Once the synthesis model is fully trained, we fine-tune the teacher flow model $\mathcal{F}$ for 100 epochs, with its learning rate fixed to 10^{-4}. We set hyper-parameters to $\lambda_1 = 1, \lambda_p = 1, \lambda_G = 20$. Both $\mathcal{G}$ and $\mathcal{F}$ benefit from an exponential moving average (EMA) with a 0.999 decay rate. We set the number of pyramids $L = 3$ during training, and use $L = \lceil\log_2(R/32)\rceil$ for resolution R at inference.

Stage 2 Training We train our diffusion model for 500 epochs using the AdamW optimizer with a constant learning rate of 2×10^{-4}, weight decay of 10^{-8}, and batch size of 64, applying an EMA with a 0.9999 decay rate. Given that diffusion models typically operate with data values between $[-1, 1]$ but optical flows often exceed this range, we normalize flow values by dividing them by 128. This adjustment ensures flow values to be compatible with the diffusion model's expected data range, effectively aligning flow values with those of the RGB space, which are similarly normalized. We utilize a linear noise schedule (Ho, Jain, and Abbeel 2020) and perform 8 denoising steps using the ancestral DDPM sampler (Ho, Jain, and Abbeel 2020) for efficient sampling.

A.4 Inference

Since our motion diffusion model is trained on a well-curated data ranging within 256 resolution, it could suffer from a performance drop when it comes to high resolution videos of large motions which goes beyond the distribution of the training data. To handle these cases, we generate flows at the training resolution by resizing the inputs, followed by post-processing of bicubic upsampling at inference time.

A.5 Input Downsampling

We provide illustrated description of input downsampling (Sec. 3.3) in Fig. A1.

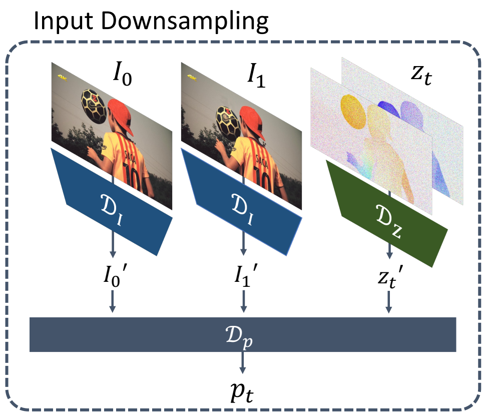


Figure A1: Visualized description of input downsampling in our motion diffusion model.

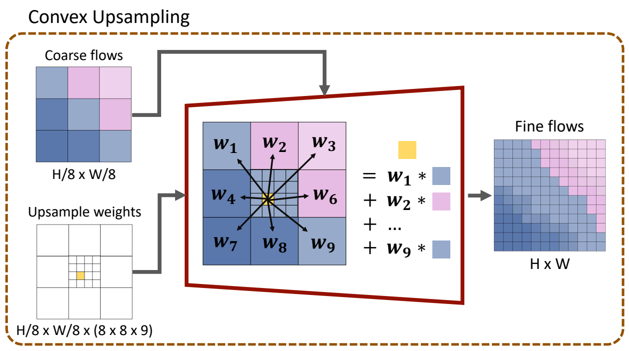


Figure A2: Visualization of convex upsampling layer. The dimensions of flows are omitted for simplicity. The upsampling weights are predicted for both directions and applied to the bi-directional flows in the same manner.

A.6 Convex Upsampling

We provide a illustrated description of convex upsampling (Sec. 3.3) in Fig. A2.

B Additional Experiments

B.1 Further Analysis on Denoising Steps

We experiment on the effect of different number of denoising steps for motion modeling, on the 'hard' subset of SNU-FILM.(Tab.A1) Although the performance does improve up to 8 steps, the increase is relatively marginal compared to the results on the 'extreme' subset. We speculate the reason for this result is due to the smaller ill-posedness of the 'hard' subset, which limits the diversity of feasible flows. We claim that the use of more steps and the design choice of diffusion models for motion modeling is more advantageous as the ill-posedness of motions gets larger.

B.2 Full Quantitative Results

We report the full quantitative results including the fidelity metrics such as PSNR and SSIM on the SNU-FILM (Tab. A2, A3), Middlebury, Vimeo90k (Tab. A4) and Xiph benchmarks (Tab. A5).

# of steps	SNU-FILM-hard		SNU-FILM-extreme	
	LPIPS	DISTS	LPIPS	DISTS
1 step	0.0421	0.0254	0.0892	0.0452
8 step (default)	**0.0419**	**0.0252**	**0.0872**	**0.0433**
20 step	0.0420	0.0253	0.0872	0.0433
50 step	0.0420	0.0254	0.0874	0.0435

Table A1: Experiment on the number of denoising steps at inference time. Our experiments show that about 8 steps is enough, and use of more steps exceeding this does not lead to a notable improvement considering the runtime tradeoff.

In addition to the fidelity metrics, we also include the results of a lighter version of our model with 10M parameters, denoted as **MoMo-10M**.

Method	Perception-oriented loss	SNU-FILM-easy				SNU-FILM-medium			
		PSNR	SSIM	LPIPS	DISTS	PSNR	SSIM	LPIPS	DISTS
ABME (Park, Lee, and Kim 2021)	✗	39.59	0.9901	0.0222	0.0229	35.77	0.9789	0.0372	0.0344
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	39.78	0.9865	0.0175	0.0181	35.36	0.9692	0.0322	0.0276
IFRNet-Large (Kong et al. 2022)	✗	40.10	0.9906	0.0203	0.0211	36.12	0.9797	0.0321	0.0288
RIFE (Huang et al. 2022b)	✗	40.06	0.9907	0.0181	0.0195	35.75	0.9789	0.0317	0.0289
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	39.74	0.9902	0.0184	0.0217	35.81	0.9789	0.0315	0.0316
AMT-G (Li et al. 2023)	✗	38.47	0.9880	0.0325	0.0312	35.39	0.9779	0.0447	0.0395
EMA-VFI (Zhang et al. 2023)	✗	39.52	0.9903	0.0186	0.0204	35.83	0.9795	0.0325	0.0318
UPRNet-LARGE (Jin et al. 2023)	✗	**40.44**	**0.9911**	0.0182	0.0203	**36.29**	**0.9801**	0.0334	0.0327
CAIN (Choi et al. 2020)	✓	39.89	0.9900	0.0197	0.0229	35.61	0.9776	0.0375	0.0347
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	39.79	0.9900	0.0123	0.0128	35.77	0.9782	0.0219	0.0183
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	39.68	0.9900	0.0120	0.0124	35.70	0.9781	0.0213	0.0177
LDMVFI (Danier, Zhang, and Bull 2024)	✓	38.68	0.9834	0.0145	0.0130	33.90	0.9703	0.0284	0.0219
PerVFI (Wu et al. 2024)	✓	38.02	0.9831	0.0142	0.0124	34.57	0.9662	0.0245	0.0181
MoMo (Ours)	✓	39.64	0.9895	**0.0111**	**0.0102**	35.45	0.9769	**0.0202**	**0.0155**
MoMo-10M (Ours)	✓	39.54	0.9896	0.0111	0.0103	35.36	0.9769	0.0204	0.0157

Table A2: Full quantitative results including the fidelity metrics (PSNR, SSIM) on the 'easy' and 'medium' subsets of SNU-FILM benchmark (Choi et al. 2020) The best results are in **bold**, and the second best is underlined, respectively.

Method	Perception-oriented loss	SNU-FILM-hard				SNU-FILM-extreme			
		PSNR	SSIM	LPIPS	DISTS	PSNR	SSIM	LPIPS	DISTS
ABME (Park, Lee, and Kim 2021)	✗	30.58	0.9364	0.0658	0.0496	25.42	0.8639	0.1258	0.0747
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	29.91	0.9073	0.0629	0.0414	24.67	0.8092	0.1257	0.0673
IFRNet-Large (Kong et al. 2022)	✗	30.63	0.9368	0.0562	0.0403	25.27	0.8609	0.1131	0.0638
RIFE (Huang et al. 2022b)	✗	30.10	0.9330	0.0657	0.0443	24.84	0.8534	0.1390	0.0764
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	30.42	0.9353	0.0568	0.0441	25.17	0.8593	0.1060	0.0632
AMT-G (Li et al. 2023)	✗	30.70	0.9381	0.0680	0.0506	**25.64**	**0.8658**	0.1128	0.0686
EMA-VFI (Zhang et al. 2023)	✗	30.79	**0.9386**	0.0579	0.0457	25.59	0.8648	0.1099	0.0671
UPRNet-LARGE (Jin et al. 2023)	✗	**30.86**	0.9377	0.0612	0.0475	25.63	0.8641	0.1109	0.0672
CAIN (Choi et al. 2020)	✓	29.90	0.9292	0.0885	0.0606	24.78	0.8507	0.1790	0.1042
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	30.34	0.9332	0.0443	0.0282	25.11	0.8557	0.0917	0.0471
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	30.29	0.9329	0.0429	0.0268	25.07	0.8550	0.0889	0.0448
LDMVFI (Danier, Zhang, and Bull 2024)	✓	28.51	0.9173	0.0602	0.0379	23.92	0.8372	0.1226	0.0651
PerVFI (Wu et al. 2024)	✓	29.68	0.9287	0.0561	0.0635	25.03	0.8120	0.0902	0.0448
MoMo (Ours)	✓	30.12	0.9312	**0.0419**	**0.0252**	25.02	0.8547	**0.0872**	**0.0433**
MoMo-10M (Ours)	✓	30.00	0.9308	0.0425	0.0257	24.91	0.8535	0.0882	0.0438

Table A3: Full quantitative results including the fidelity metrics (PSNR, SSIM) on the 'hard' and 'extreme' subsets of SNU-FILM benchmark (Choi et al. 2020). The best results are in **bold**, and the second best is underlined, respectively.

Method	Perception-oriented loss	Middlebury PSNR	Middlebury SSIM	Middlebury LPIPS	Middlebury DISTS	Vimeo90k PSNR	Vimeo90k SSIM	Vimeo90k LPIPS	Vimeo90k DISTS
ABME (Park, Lee, and Kim 2021)	✗	37.05	0.9845	0.0290	0.0325	36.18	0.9805	0.0213	0.0353
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	36.72	0.9826	0.0169	0.0244	35.07	0.9710	0.0229	0.0354
IFRNet-Large (Kong et al. 2022)	✗	36.27	0.9816	0.0285	0.0366	36.20	0.9808	0.0189	0.0325
RIFE (Huang et al. 2022b)	✗	37.16	0.9853	0.0162	0.0228	35.61	0.9780	0.0223	0.0356
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	37.37	0.9838	0.0173	0.0246	35.89	0.9796	0.0197	0.0343
AMT-G (Li et al. 2023)	✗	34.23	0.9708	0.0486	0.0533	**36.53**	**0.9819**	0.0195	0.0351
EMA-VFI (Zhang et al. 2023)	✗	**38.32**	**0.9871**	0.0151	0.0218	<u>36.45</u>	0.9811	0.0196	0.0343
UPRNet-LARGE (Jin et al. 2023)	✗	<u>38.09</u>	<u>0.9861</u>	0.0150	0.0209	36.42	<u>0.9815</u>	0.0201	0.0342
CAIN (Choi et al. 2020)	✓	35.11	0.9761	0.0254	0.0383	34.65	0.9729	0.0306	0.0483
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	37.28	0.9843	0.0096	0.0148	35.62	0.9784	0.0137	0.0229
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	37.38	0.9844	**0.0093**	<u>0.0140</u>	35.71	0.9787	**0.0131**	<u>0.0224</u>
LDMVFI (Danier, Zhang, and Bull 2024)	✓	34.03	0.9648	0.0195	0.0261	33.09	0.9558	0.0233	0.0327
PerVFI (Wu et al. 2024)	✓	35.00	0.9751	0.0142	0.0163	34.00	0.9675	0.0179	0.0248
MoMo (Ours)	✓	36.77	0.9806	<u>0.0094</u>	**0.0126**	34.94	0.9756	<u>0.0136</u>	**0.0203**
MoMo-10M (Ours)	✓	36.52	0.9801	0.0100	0.0139	34.82	0.9752	0.0138	0.0206

Table A4: Full quantitative results including the fidelity metrics (PSNR, SSIM) on Middlebury (Baker et al. 2011) and Vimeo90k (Xue et al. 2019) benchmarks. The best results are in **bold**, and the second best is <u>underlined</u>, respectively.

Method	Perception-oriented loss	Xiph-2K PSNR	Xiph-2K SSIM	Xiph-2K LPIPS	Xiph-2K DISTS	Xiph-4K PSNR	Xiph-4K SSIM	Xiph-4K LPIPS	Xiph-4K DISTS
ABME (Park, Lee, and Kim 2021)	✗	36.50	0.9668	0.1071	0.0581	33.72	0.9452	0.2361	0.1108
XVFI$_v$ (Sim, Oh, and Kim 2021)	✗	35.17	0.9625	0.0844	0.0418	32.45	0.9274	0.1835	0.0779
IFRNet-Large (Kong et al. 2022)	✗	36.40	0.9646	0.0681	0.0372	33.71	0.9425	0.1364	0.0665
RIFE (Huang et al. 2022b)	✗	36.06	0.9642	0.0918	0.0481	33.21	0.9413	0.2072	0.0915
FILM-$\mathcal{L}_1$ (Reda et al. 2022)	✗	36.53	0.9663	0.0906	0.0510	33.83	0.9439	0.1841	0.0884
AMT-G (Li et al. 2023)	✗	36.29	0.9647	0.1061	0.0563	<u>34.55</u>	<u>0.9472</u>	0.2054	0.1005
EMA-VFI (Zhang et al. 2023)	✗	<u>36.74</u>	<u>0.9675</u>	0.1024	0.0550	<u>34.55</u>	**0.9486**	0.2258	0.1049
UPRNet-LARGE (Jin et al. 2023)	✗	**37.13**	**0.9691**	0.1010	0.0553	**34.57**	0.9388	0.2150	0.1017
CAIN (Choi et al. 2020)	✓	35.18	0.9625	0.1025	0.0533	32.55	0.9398	0.2229	0.0980
FILM-$\mathcal{L}_{vgg}$ (Reda et al. 2022)	✓	36.29	0.9626	0.0355	0.0238	33.44	0.9356	0.0754	0.0406
FILM-$\mathcal{L}_s$ (Reda et al. 2022)	✓	36.30	0.9616	<u>0.0330</u>	0.0237	33.37	0.9323	<u>0.0703</u>	0.0385
LDMVFI (Danier, Zhang, and Bull 2024)	✓	33.82	0.9494	0.0420	0.0163	31.39	0.9214	0.0859	0.0359
PerVFI (Wu et al. 2024)	✓	34.69	0.9541	0.0381	<u>0.0153</u>	32.30	0.9149	0.0858	<u>0.0331</u>
MoMo (Ours)	✓	35.38	0.9553	**0.0300**	**0.0119**	33.09	0.9293	**0.0631**	**0.0274**
MoMo-10M (Ours)	✓	35.23	0.9548	0.0303	0.0120	32.97	0.9281	0.0638	0.0275

Table A5: Full quantitative results including the fidelity metrics (PSNR, SSIM) on Xiph-2K and Xiph-4K (Montgomery and Lars 1994; Niklaus and Liu 2020). The best results are in **bold**, and the second best is <u>underlined</u>, respectively.