Deep Bayesian Video Frame Interpolation

Zhiyang Yu[1,2†] , Yu Zhang[2✉], Xujie Xiang[2,3†], Dongqing Zou[2,4],

Xijun Chen[1✉] , and Jimmy S. Ren[2,4]

[1] Harbin Institute of Technology, Harbin, China
[2] SenseTime Research and Tetras.AI, Beijing, China
[3] Beihang University, Beijing, China
[4] Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, China
zhangyulb@gmail.com chenxijun@hit.edu.cn

Abstract. We present deep Bayesian video frame interpolation, a novel approach for upsampling a low frame-rate video temporally to its higher frame-rate counterpart. Our approach learns posterior distributions of optical flows and frames to be interpolated, which is optimized via learned gradient descent for fast convergence. Each learned step is a lightweight network manipulating gradients of the log-likelihood of estimated frames and flows. Such gradients, parameterized either explicitly or implicitly, model the fidelity of current estimations when matching real image and flow distributions to explain the input observations. With this approach we show new records on 8 of 10 benchmarks, using an architecture with half the parameters of the state-of-the-art model. Code and models are publicly available at https://github.com/Oceanlib/DBVI.

Keywords: Video Frame Interpolation · Deep Bayesian · Image Reconstruction

1 Introduction

To meet the requirement of power consumption and image quality, modern video sensors often work with limited frame aquisition rate. Video Frame Interpolation (VFI) [3,7,15,16,20,25,41] is an important computational approach that synthesizes the missing intermediate frames between consecutive frames of a low frame-rate video. When capable of performing interpolation at any continuous time step, VFI is able to temporally upsample a video to any desired resolution, creating smooth slow-motion effect at various degrees of natural movements [4,15].

Solving continuous-time VFI requires representing pixel movements continuously. It was commonly achieved by fitting parametric pixel trajectory models to the optical flows extracted from sparse observed frames [6,15,41]. Intermediate pixel flows are resampled at sub-frame time steps, providing warping fields to

† The work is done during an internship at SenseTime Research and Tetras.AI.
✉ Correspondence should be addressed to Yu Zhang (zhangyulb@gmail.com) and Xijun Chen (chenxijun@hit.edu.cn).

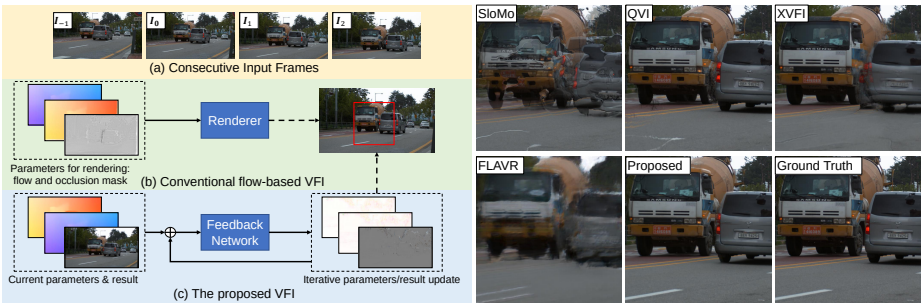

Fig. 1. Motivation of the proposed approach. Given input consecutive frames (a), most conventional approaches predict parameters for novel frame rendering in one pass (b) while ours generates learned updates in multiple iterations (c) to simplify prior modeling. This yields improved interpolation results in challenging scenarios compared with recent models like SloMo [15], QVI [41], XVFI [32], and FLAVR [16].

reorganize the pixels from known input frames to synthesize the intermediate view, possibly with external geometric guidance (*e.g.* depth [3]). However, as inputs of VFI are sparse scene captures with low temporal frequency, inferring the underlying full frequency motion is highly ill-posed. Several works tackle the ill-poseness by learning a function mapping the low frame-rate input to its high frame-rate counterpart, fitted from raw data (*e.g.* [7,11,16]). Yet without continuous representation, they lose the flexibility for continuous-time frame interpolation.

If viewed from optimization perspective, continuous-time VFI methods predict the parameters (*e.g.* intermediate scene geometries like flows and occlusion) of an inference model (*e.g.* image renderer with scene geometries as rendering parameters) to best explain the measurements (*e.g.* input frames). Since there are far more predictive parameters than measurements, effective priors are required. In most contemporary works [3,15,27,41], parameters are predicted from sparse measurements in one pass, having few chance to feedback and improve rendering errors (as in Fig. 1 (b)). Consequently, burden is left to the network to encode strong priors into the parameters (*e.g.* joint space of parameters of natural scene geometry), which is prohibitively difficult even for large architecture.

In this work we propose deep Bayesian video frame interpolation, a continuous-time VFI framework with iterative learned task priors. Our framework learns posterior distributions of the intermediate optical flows and frames to be interpolated, optimized via learned gradient descent [1,2] for fast convergence within a few iterations. Each iteration evaluates the gradients of the log-likelihood modelling fidelity of current estimations of image and flows. Such first-order gradients are altered by a network encoding learned prior of higher-order information, such that after updated with altered gradients, intermediate estimations fall effectively onto the manifold of natural images and flows. We show log-likelihood gradients *w.r.t.* interpolated images can be explicitly defined with intuitive in-

terpretations. The gradients *w.r.t.* interpolated optical flows are derived from an implicit posterior flow distribution jointly trained with other components to account for the complex real-world flow distribution. When unfolded, the learned gradient steps form an architecture that can be trained end-to-end.

Our approach amortizes learning the whole task priors in one pass into multiple iterations, while each iteration reduces to learning a simpler prior on how to manipulate gradients towards smaller rendering errors in a principled optimization framework. Evaluated with extensive experiments, this method sets new records on 8 of 10 public benchmarks, outperforms state-of-the-art VFI model [16] with half parameter size.

Our work aims to introduce 3 folds of contributions: 1) a principled contiuous-VFI framework, which produces high-quality interpolation results with less parameters; 2) a deep parameterization of gradients of arbitary posterior distributions that can be end-to-end optimized as a network component; 3) a unified and fair benchmarking of VFI models on standard datasets.

2 Related works

Video frame interpolation has became an active research topic in computer vision since more than one decade earlier [22,39,43]. Though, the performance of VFI methods are effectively boosted in recent year [15], with emergence of deep learning architectures and high-quality training data. Since then, various methods were proposed. Among them, a large body of works treat VFI as a novel view rendering problem by warping the observed video frames.

Flow-based rendering predicts pixel flows of intermediate time steps by analyzing the optical flows extracted from input video. As input frames are sparse, explicit flow priors are placed to yield continuous sampling of flows at arbitary time. Early work [15] imposes linear flow prior, which was further extended to quadratic [41] or even cubic [6] trajectory models to accurately represent local movements of pixels. A challenge here is that to get rid of occlusion and confliction in flow-based warping, flows starting from the intermediate frame to input frames need to be known, while the resampled flows are in reverse order. Heuristic flow reversing was thus proposed, by interpolating at occluded pixels with various local kernels [4,15,32,41] possibly guided by scene geometry like depth [3]. Instead of reversing flows, [24] addresses occlusion-ware warping with forward splatting, using color constancy as guidance. As an important component, learnable flow refinement modules were applied to improve the heuristically interpolated flows. To name a few, they include bilateral cost volumes [26,27], spatial pyramidal flow upsampling [32], temporal pyramidal refinement [6] and learnable median filtering [41].

Kernel-based rendering learns linear kernels to aggregate input pixels within a local spatiotemporal window to produce interpolation results. Such kernels were implemented as separable convolutions [25], deformable convolutions [5,20,30,31] and trilinear sampling kernels [21]. However, most such methods cannot address VFI at arbitary time steps as learned kernels are time-specific.

Instead of explicit frame rendering, several works learn direct mapping from low frame-rate video to its higher frame-rate counterpart from data. It makes VFI benefit from the strong pattern fitting ability of deep networks, implemented with channel attention [7] or 3D convolution [16]. Again, they do not address arbitary-time VFI for the descrete representations of neural network, and have risk of over-fitting to the low-level statistics of a dataset.

Our method starts from a different perspective, treating directly the inverse problem nature of VFI and solving it with gradient-based Bayesian optimization [1,2]. Though actively studied for low-level tasks like denoising [19,36], demosaicking [18] and 3D view synthesis [9], it is by the first time explored for VFI to the best of our knowledge. The method can be viewed as combining existing directions but in different way — it learns frame interpolation by fitting the data, while at the same time evaluates current rendering errors on the observed frames and flows and feedback such errors to boost next iteration. This makes it robust to overfitting, as each of its step is to fix local estimation errors instead of encoding all the appearance and motion dynamics in a single pass.

3 The Proposed Method

3.1 Background and Notations

Given consecutive frames $\mathbf{I}_{-1}, \mathbf{I}_0, \mathbf{I}_1, \mathbf{I}_2$, where $\mathbf{I}_i \in [0,1]^{H\times W\times 3}, \forall i \in \{-1,0,1,2\}$, we aim at interpolating intermediate frame $\mathbf{I}_t$ at any arbitrary time $t \in (0,1)$. Following prior works [15,32,41], we start by extracting optical flows from input frames, then fit a continuous motion model that can be resampled at time t to get flows $\mathbf{F}_{0\to t}, \mathbf{F}_{1\to t} \in \mathbb{R}^{H\times W\times 2}$. To render the target frame $\mathbf{I}_t$, such forward flows were heuristically reversed and then refined in previous works, yielding backward-stage flows $\mathbf{F}_{t\to 0}$, $\mathbf{F}_{t\to 1}$ so as to make rendering possible via warping:

$$\mathbf{I}_\mathbf{t}^* = (1-t)\odot\mathbf{M}\odot\phi_\mathcal{B}(\mathbf{I}_0,\mathbf{F}_{t\to 0}) + t\odot(1-\mathbf{M})\odot\phi_\mathcal{B}(\mathbf{I}_1,\mathbf{F}_{t\to 1}), \tag{1}$$

where $\mathbf{M}$ is the blending mask, $\odot$ is the Hadamard product, and ϕ_B stands for the backward warping function [14]. However, reversing forward flows and inferring occlusions at unknown time step are both challenging, while the rendering-driven scheme (1) poses strong assumption on the quality of them.

Differing from previous works, we solve VFI with a deep Bayesian approach, optimizing for results that can best explain the occurrence of input data regularized with priors. Specfically we learn directly the posterior distributions of $\mathbf{I}_t$, conditioned on the observed frames and flows:

$$\mathbf{I}_\mathbf{t}^* = \arg\max_{\mathbf{I}_t} P(\mathbf{I}_t|\mathbf{I}_0,\mathbf{I}_1,\mathbf{F}_{0\to t},\mathbf{F}_{1\to t}). \tag{2}$$

3.2 Deep Bayesian Video Frame Interpolation

To derive a tractable solution of (2), we make the following simplification

$$P(\mathbf{I}_t|\mathbf{I}_0,\mathbf{I}_1,\mathbf{F}_{0\to t},\mathbf{F}_{1\to t}) = \prod_{i\in\{0,1\}} P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t}), \tag{3}$$

where $\mathcal{I}_t$ is assumed that can be independently explained by image and flows at either frame 0 or 1. Since the forward flows $\mathbf{F}_{0\to t}$ and $\mathbf{F}_{1\to t}$ are estimated from low frame-rate video frames, errors inevitably exist. To account for it, we also incorporate the refined task-specific forward flows as latent variables:

$$\begin{aligned} P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t}) &= \int_{\Delta\mathbf{F}_{i\to t}} P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t},\Delta\mathbf{F}_{i\to t})P(\Delta\mathbf{F}_{i\to t}|\mathbf{I}_i,\mathbf{F}_{i\to t})d\Delta\mathbf{F}_{i\to t} \\ &\approx P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t})P(\Delta\hat{\mathbf{F}}_{i\to t}|\mathbf{I}_i,\mathbf{F}_{i\to t}), \end{aligned} \tag{4}$$

where refined flows are modelled by the residual flows $\Delta\mathbf{F}_{i\to t}$. The approximation step replaces the integral, which enumerates all possible $\Delta\mathbf{F}_{i\to t}$s to be calculated, with the mode of the conditional $\Delta\hat{\mathbf{F}}_{i\to t} = \arg\max_{\Delta\mathbf{F}_{i\to t}} P(\Delta\mathbf{F}_{i\to t}|\mathbf{I}_i,\mathbf{F}_{i\to t})$, for tractability. Integrating (4) into (3) and taking the negative logarithm, we arrive at the following minimization problem

$$\min_{\mathbf{I}_t,\{\Delta\hat{\mathbf{F}}_{i\to t}\}} -\sum_{i\in\{0,1\}} \left(\log P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t}) + \log P(\Delta\hat{\mathbf{F}}_{i\to t}|\mathbf{I}_i,\mathbf{F}_{i\to t})\right). \tag{5}$$

The non-linear problem (5) can be optimized via iterative gradient descent:

$$\mathbf{I}_t^{(k+1)} = \mathbf{I}_t^{(k)} - \lambda_{\mathbf{I}}\frac{\partial L}{\partial \mathbf{I}_t}, \quad \Delta\hat{\mathbf{F}}_{i\to t}^{(k+1)} = \Delta\hat{\mathbf{F}}_{i\to t}^{(k)} - \lambda_{\mathbf{F}}\frac{\partial L}{\partial \Delta\hat{\mathbf{F}}_{i\to t}}, \tag{6}$$

where $L(\mathbf{I}_t,\{\Delta\hat{\mathbf{F}}_{i\to t}\};\{\mathbf{I}_i\},\{\mathbf{F}_{i\to t}\})$ is the objective of (5) ($\{\cdot\}$ represents the collection of variables of different is for short), $\lambda_{\mathbf{I}}$ and $\lambda_{\mathbf{F}}$ are step sizes. For $\frac{\partial L}{\partial \mathbf{I}_t}$,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{I}_t} &= -\sum_{i\in\{0,1\}} \frac{\partial \log P(\mathbf{I}_t|\mathbf{I}_i,\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t})}{\partial \mathbf{I}_t} \\ &= -\sum_{i\in\{0,1\}} \left(\frac{\partial \log P(\mathbf{I}_i|\mathbf{I}_t,\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t})}{\partial \mathbf{I}_t} + \frac{\partial \log P(\mathbf{I}_t|\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t})}{\partial \mathbf{I}_t}\right), \end{aligned} \tag{7}$$

where the first term represents gradients of the *log-likelihood* that input frame $\mathbf{I}_i$ occurs given the estimations $\mathbf{I}_t$ and $\Delta\hat{\mathbf{F}}_{i\to t}$, while the second term are gradients of the *conditional prior* of the interpolated frames. Iteratively evaluating gradients and performing updates can take large numbers of steps to converge. We leverage recent advances on learned gradient descent [1], performing updates with learned gradients conditioned on the classical ones:

$$\mathbf{I}_t^{(k+1)} = \mathbf{I}_t^{(k)} + \mathcal{G}_{\mathbf{I}}\left(\left\{\frac{\partial \log P(\mathbf{I}_i|\mathbf{I}_t^{(k)},\mathbf{F}_{i\to t},\Delta\hat{\mathbf{F}}_{i\to t}^{(k)})}{\partial \mathbf{I}_t}\right\}, \mathbf{I}_t^{(k)}, \{\mathbf{F}_{i\to t}\}, \{\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}\}\right). \tag{8}$$

Note that conditional priors are implicitly folded into the learned gradient network $\mathcal{G}_{\mathbf{I}}$. Conditioning on the likelihood gradients encodes the local loss landscape into iterative minimization, where gradient directions and magnitudes signify the gap of current estimation to explain the given observations.

The partial gradients $\frac{\partial L}{\partial \Delta\hat{\mathbf{F}}_{i\to t}}$ in (5) are comprised exactly of the conditional likelihood and prior terms. Therefore, its learned update rule can be defined:

$$\Delta\hat{\mathbf{F}}_{i\to t}^{(k+1)} = \Delta\hat{\mathbf{F}}_{i\to t}^{(k)} + \mathcal{G}_{\mathbf{F}}\left(\frac{\partial \log P(\mathbf{I}_t^{(k)}|\mathbf{I}_i, \mathbf{F}_{i\to t}, \Delta\hat{\mathbf{F}}_{i\to t}^{(k)})}{\partial \Delta\hat{\mathbf{F}}_{i\to t}}, \Delta\hat{\mathbf{F}}_{i\to t}^{(k)}, \mathbf{I}_i, \mathbf{F}_{i\to t}\right). \quad (9)$$

With learned gradients, we unroll a small number of K update steps. The overall optimization procedure can be unfolded as a specific architecture, trained on a dataset with paired low frame-rate and high frame-rate video frames.

3.3 Formulating the gradients

Each iteration of (8) and (9) requires evaluating specific log-likelihood gradients. We show that gradients corresponding to the incremental image (8) has explicit and interpretable representations under simple distributions, resembling warped reconstruction errors with current flow estimations. For the flow log-likelihood gradients (9), we propose an implicit representation via deep parameterization to make them tractable and computationally efficient.

Explicit image log-likelihood gradients. Image likelihood in (8) models the occurrence of input video frame $\mathbf{I}_i$, given intermediate estimations $\mathbf{I}_t^{(k)}$ and $\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}$. We define this conditional likelihood as a pixel-independent Gaussian distribution, centered at the warped version of $\mathbf{I}_t^{(k)}$:

$$\mathbf{I}_i \sim \mathcal{N}\left(\phi_{\mathcal{B}}(\mathbf{I}_t^{(k)}, \mathbf{F}_{i\to t} + \Delta\hat{\mathbf{F}}_{i\to t}^{(k)}), \sigma_{\mathbf{I}}^2\right), \quad (10)$$

where $\phi_{\mathcal{B}}$ is the bilinear warping function, and $\sigma_{\mathbf{I}}$ is set to a constant. If we write $\hat{\mathbf{W}}_{i\to t}^{(k)}$ as the matrix absorbing bilinear warping weights from flows $\mathbf{F}_{i\to t} + \Delta\hat{\mathbf{F}}_{i\to t}^{(k)}$, then the warping can be represented by matrix multiplication $\hat{\mathbf{W}}_{i\to t}^{(k)}\mathbf{I}_t^{(k)}$. Gradients of this Gaussian likelihood are then $(\hat{\mathbf{W}}_{i\to t}^{(k)})^{\mathrm{T}}(\hat{\mathbf{W}}_{i\to t}^{(k)}\mathbf{I}_t^{(k)} - \mathbf{I}_i)$. It can be intuitively interpreted as evaluating the errors of explaining $\mathbf{I}_i$ when $\mathbf{I}_t^{(k)}$ is warped to time i, while gathering such errors at time t by reverse splatting.

Implicit flow log-likelihood gradients. The flow-side likelihood from (9) aims to explain the occurrence of $\mathbf{I}_t^{(k)}$ given $\mathbf{I}_i$ and estimated flows $\hat{\mathbf{F}}_{i\to t}^{(k)}$. Unlike image likelihood, defining it with simple distribution by image warping is tricky, as forward flows $\hat{\mathbf{F}}_{i\to t}^{(k)}$ do not convey sufficient scene geometry (*e.g.* occlusion relaionship) for warping. Instead of heuristic flow reversal [3,15,32,41], we model such distribution implicitly with learnable parameterization. By the Bayesian formula relating likelihood, posterior and prior, the likelihood gradients can be implicitly represented with the gradients derived from posterior and prior:

$$\frac{\partial \log P(\mathbf{I}_t^{(k)}|\cdot, \Delta\hat{\mathbf{F}}_{i\to t}^{(k)})}{\partial \Delta\hat{\mathbf{F}}_{i\to t}} = \frac{\partial \log P(\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)}, \cdot)}{\partial \Delta\hat{\mathbf{F}}_{i\to t}} - \frac{\partial \log P(\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}|\cdot)}{\partial \Delta\hat{\mathbf{F}}_{i\to t}}, \quad (11)$$

where we omit $\mathbf{I}_i$ and $\mathbf{F}_{i\to t}$ in the conditions to ease presentation. As prior is folded into gradient network, we propose to evaluate the log-posterior gradients (the first term of the righthand of (11)) instead. This resembles flow estimation from a frame pair $\mathbf{I}_t^{(k)}$ and $\mathbf{I}_i$, where in the existing literature, the flow posterior distribution is mostly a Laplacian of fixed variance (with ℓ_1 loss as negative log-density) [8,35,37]. However, to ensure such simple distribution works, it requires presence of ground-truth optical flows, which are absent in our task.

To learn flow posterior gradients without explicitly specifying its underlying distribution $P(\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)},\cdot)$, we propose a deep reparameterization approach by leveraging *normalizing-flow-based invertible layers* [17]. We assume an invertible, injective and twice-differentiable function $f:\mathbb{R}^m\mapsto\mathbb{R}^m$ transforms the flow maps $\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}$ to a multi-dimensional latent representation $\mathbf{X}_{i\to t}^{(k)}=f(\Delta\hat{\mathbf{F}}_{i\to t}^{(k)})$. By the change of variables and chain rule of gradients,

$$\frac{\partial\log P(\Delta\hat{\mathbf{F}}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)},\cdot)}{\partial\Delta\hat{\mathbf{F}}_{i\to t}}=\frac{\partial\log\left|\det\left(\frac{\partial\mathbf{X}_{i\to t}}{\partial\Delta\hat{\mathbf{F}}_{i\to t}}\right)\right|}{\partial\Delta\hat{\mathbf{F}}_{i\to t}}+\frac{\partial\mathbf{X}_{i\to t}}{\partial\Delta\hat{\mathbf{F}}_{i\to t}}\frac{\partial\log P(\mathbf{X}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)},\cdot)}{\partial\mathbf{X}_{i\to t}}. \tag{12}$$

As $\mathbf{X}_{i\to t}$ resides in deep latent space, we assume it follows multi-variate Gaussian $P(\mathbf{X}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)},\cdot)=\mathcal{N}\left(\mu(\mathbf{I}_t^{(k)},\cdot),\mathbf{\Sigma}(\mathbf{I}_t^{(k)},\cdot)\right)$, whose mean and covariance are predictive as intermediate network outputs. Gradients of Gaussian log-likelihood *w.r.t.* $\mathbf{X}_{i\to t}$ can be efficiently evaluated:

$$\frac{\partial\log P(\mathbf{X}_{i\to t}^{(k)}|\mathbf{I}_t^{(k)},\cdot)}{\partial\mathbf{X}_{i\to t}}=\mathbf{\Sigma}^{-1}(\mathbf{I}_t^{(k)},\cdot)\left(\mathbf{X}_{i\to t}^{(k)}-\mu(\mathbf{I}_t^{(k)},\cdot)\right). \tag{13}$$

To ensure PSD of covariance matrix and avoid inversion, we let $\mathbf{\Sigma}=\mathbf{S}\mathbf{S}^{\mathrm{T}}$ and parameterize $\mathbf{S}^{-1}$ instead of $\mathbf{\Sigma}$. Other gradient terms of (12) can be computed by once and twice differentiating though the layers of the function f[5].

The modeling (12) can be thought of transforming the flow predictions $\Delta\hat{\mathbf{F}}_{i\to t}$ into a deep embedded space such that the fidelity of flows are measureable with learned metric defined by a simple distribution. The first and second gradient terms of (12) serve as volume-preserving scaling and biases of the deep gradients, ensuring real gradients *w.r.t.* $\Delta\hat{\mathbf{F}}_{i\to t}$ are computed no matter what the transformation is. The transformation is not explicitly defined, but implicitly optimized towards explaining the training data. By stacking many invertible layers to form f, we theoretically can model the gradients of arbitarily complex posterior distributions of optical flows to match the real distribution.

One thing to take care of in practice is that the forward flows $\Delta\hat{\mathbf{F}}_{i\to t}$ have only two channel dimensions, which is too restricted to learn expressive deep latent representations (since invertible layers preserve dimensionality). We therefore lift $\Delta\hat{\mathbf{F}}_{i\to t}$ to an augmented M-dimensional space by adding more channels, while only the first two channels are used for flow modeling. Such additional channels are initialized with zeros at beginning, while learned during iterations.

[5] Please refer to the supplementary material for more discussions on implementation.

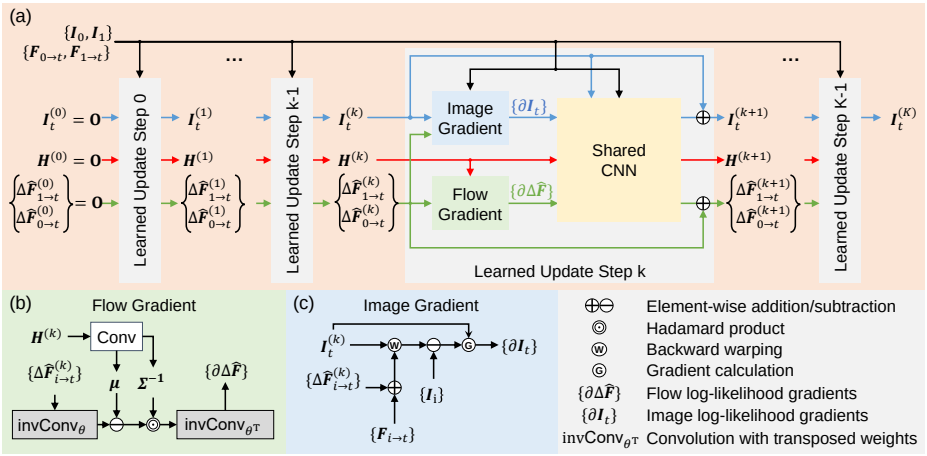

Fig. 2. Pipeline of our approach, best viewed in color. Please see text for details.

3.4 Implementation

Architecture. The per-iteration networks $\mathcal{G}_{\mathbf{I}}$ and $\mathcal{G}_{\mathbf{F}}$ have many shared inputs (*e.g.* $\{\mathbf{F}_{i\to t}\}$, $\{\Delta\hat{\mathbf{F}}^{(k)}_{i\to t}\}$ and $\{\mathbf{I}^{(k)}_t\}$). As shown in Fig.2, we therefore use a single network to reduce redundant calculation and explore the complementary cues. At each iteration, this shared network generates learned updates of image and flows, which are element-wise added to previous estimations to get current results $\mathbf{I}^{(k)}_t$ and $\{\Delta\hat{\mathbf{F}}^{(k)}_{i\to t}\}$. In addition to image and flow updates, the shared network also outputs additional intermediate features $\mathbf{H}^{(k)}$, which summarizes historical information and are passed across iterations, as suggested in [2,9].

In each iteration, $\mathbf{I}^{(k)}_t$, $\{\mathbf{I}_i\}$, $\{\Delta\hat{\mathbf{F}}^{(k)}_{i\to t}\}$, and $\{\mathbf{F}_{i\to t}\}$ are processed by the Image Gradient Module to get the gradients $\partial\mathcal{I}_t$, as described in Sect. 3.3. For flow gradients, first note that flows are lifted to high-dimensional space ($\Delta\hat{\mathbf{F}}_{i\to t}$ is set 16-dimensional in our implementation). The Flow Gradient Module then predicts the mean and inverse of co-variance (or precision) of 16-dimensional Gaussian from the last hidden features $\mathbf{H}^{(k)}$. For efficiency we assume the precision matrix is diagonal, using exponential layer on the top to ensure positiveness of precision. Remember that $\{\Delta\hat{\mathbf{F}}^{(k)}_{i\to t}\}$ go through invertible transformation for gradient evaluation. In practice, we model it with a single invertible convolution layer [17]. Though stacking more layers is advantageous for performance in principle, we find it already effective in experiments (Sect. 4) while highly efficient. As shown in Fig. 2 (b), differentiation through this transformation is simply a matrix operation with the transpose of the convolution weights.

After calculated, image and flow gradients are concatenated with other conditional inputs and fed into the shared CNN, whose configuration is shown in Fig. 3. After an input-transform convolution, two Residual-in-Residual UNet Blocks (RRUB) are adopted. RRUB is a simplified version of the RRDB block [38], where its dense block [13] is replaced with a light-weight UNet [29]. The UNet

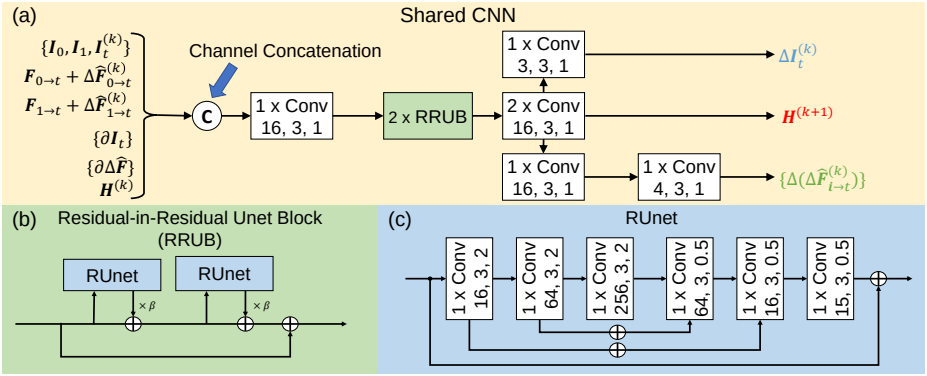

Fig. 3. Architecture of shared network. Convolutional layers are labeled with the number of blocks, number of filters followed by the kernel size and stride. Best viewed in color. Please see text for details.

has 3 scales, with skip connections as feature addition between encoder and decoder. Group normalization [40] is used with group size 4, accompanied with GELU activation [12], for all the convolutional layers in the shared network.

Initialization. At the beginning, $\mathbf{I}_t^{(0)}$, $\mathbf{H}^{(0)}$ and the lifted $\Delta\hat{\mathbf{F}}_{i\to t}$ are initialized with zeros. The forward flows $\{\mathbf{F}_{i\to t}\}$ are estimated based on the quadratic motion model proposed in [41], where optical flows are computed by RAFT [37].

Loss function. We use ℓ_1 loss only to jointly train all the iterations:

$$\mathcal{L} = \sum_{k=1}^{K} \alpha_k \left\| \mathbf{I}_t^{gt} - \mathbf{I}_t^{(k)} \right\|_1, \tag{14}$$

where α_k is the weight for kth iteration. Weights are set empirically, increasing monotonously with the iterations (please see Sect. 4.1 for detailed parameters).

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate extensively on 10 benchmarks: GoPro[23], Adobe240[34], X4K1000FPS[32], Vimeo-90K[42], UCF101[33], DAVIS[28] and the *easy*, *medium*, *hard* and *extreme* subsets from SNU-FILM[7]. Among them, GoPro, X4K1000FPS and Adobe240 are used to evaluate 8× interpolation, while others are 2×.

Unified evaluation protocols with guaranteed reproducibility[6]**.** For 8× interpolation, we follow the same settings in [16] and use the official train/test split of GoPro to perform evaluations on this dataset. For X4K1000FPS, we use both the model pretrained on GoPro and retrain it on X4K1000FPS to fairly

[6] In the supplementary material we provide a more detailed description.

Table 1. Quantitative results on 8× interpolation in terms of PSNR/SSIM on GoPro, X4K1000FPS and Adobe240 datasets. Best results highlighted in red.

	GoPro	X4K1000FPS	Adobe240	Param(M)	TFLOPS
SloMo [15]	29.71/0.924	25.07/0.795	29.63/0.927	39.61	0.624
QVI [41]	30.52/0.941	28.06/0.855	31.41/0.955	29.23	1.075
DAIN [3]	29.53/0.920	27.28/0.835	30.53/0.939	24.03	5.785
EDSC [5]	29.20/0.916	25.30/0.811	29.87/0.931	8.95	0.260
FLAVR [16]	31.10/0.942	24.50/0.791	30.92/0.938	42.06	3.793
XVFI [32]	29.80/0.925	28.42/0.881	29.74/0.930	5.61	0.676
Ours	31.73/0.947	31.10/0.928	33.28/0.965	15.18	1.284

Table 2. Quantitative results following benchmarking protocol of the X4K1000FPS dataset [32] for 8× interpolation. Best results highlighted in red.

	AdaCoF [20]	FeFlow [10]	DAIN [3]	SloMo [15]	FLAVR [16]	XVFI [32]	QVI [41]	Ours
PSNR	25.81	25.16	27.52	27.77	27.92	30.12	29.96	32.89
SSIM	0.772	0.783	0.821	0.849	0.853	0.870	0.892	0.939

compare with existing methods evaluated with these two different protocols. For Adobe, there is no standard test dataset. We thus follow [32] and randomly extract non-overlapping clips containing complex motions. Note our sampled test set is 3× larger than that of [32]. Downsampling as in [41] is not applied.

For 2× interpolation, our model is trained on the train split of Vimeo-90K. As various methods including ours require 4 consecutive frames as input, we use the *septuplet* subset for evaluation. The pre-trained model is tested on the test dataset of Vimeo-90K, UCF101 and DAVIS, which is the same setting of [16]. We follow [5,7,27] to report the results on the 4 subsets from SUN-FILM. The Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are used for quantitative evaluation, averaged across all the interpolated frames.

Note that evaluation protocols of previous VFI models for multiple interpolation are not totally unified. For example, some were trained with private training data [6,15,41], or tested on different subsets selected from a particular dataset [16,32]. As a contribution of this work, we present here a more unified benchmarking. Specifically, we retrained SloMo [15], QVI [41], DAIN [3], CAIN [7], FLAVR [16] and XVFI [32], which have open-source code, on different datasets with reproducibility guarantee. We reuse the results of several works (*e.g.* AdaCoF [20] and FeFlow [10]) in case that their evaluation protocols tightly follow the standard (and ours). However, for methods without training code or image results released, we either use their pretrained model (*e.g.* EDSC [38] and ABME [27]) or the reported results from related paper with the same evaluation protocol with ours (*e.g.* Softsplat [24] results are copied from [31]).

Training Details. For 8× MFI, we use $K = 4$ iterations, while the per-iteration weight α_k is $\{0.2, 0.4, 1.0, 1.0\}$. We train the network for 200 epochs with Adam optimizer and batch size 16. The learning rate is initialized as 5×10^{-4}

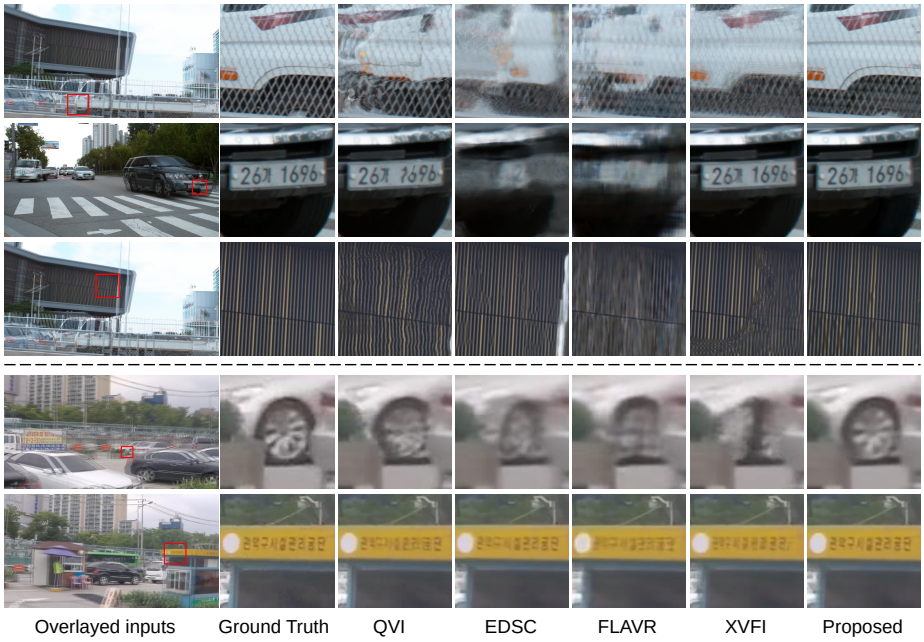

Fig. 4. Visual comparisons on X4K1000FPS (top 3 rows) and GoPro (bottom 2 rows) datasets, where each row shows an example. We overlay the nearest 2 input frames to illustrate the input motion. More results can be found in the supplementary material.

and decreased by a factor of 0.4 at the 80th, 120th and 160th epochs. Data augmentations like random cropping, flipping, and color jittering are adopted. For 2× VFI, we use $K = 6$ iterations with all corresponding α_k set to 1.0. The batch size is 32, and learning rate is decreased at the 110th and 135th epochs, respectively.

4.2 Comparisons with State-of-the-Art models

8× interpolation. We report quantitative results on 8× VFI on GoPro, Adobe240 and X4K1000FPS in Table 1, as well as TFLOPS and the model size in terms of number of parameters of different methods. Following [16], all the models for evaluation are trained on the GoPro dataset except EDSC, for which we use the officially pretrained model due to lacking the training code. The proposed approach achieves the best results across all the datasets, achieving at least 0.63dB improvement with comparable FLOPS and greatly reduced size of model (15M parameters compared with 42M of FLAVR and 29M of QVI). For those with less parameters than ours, we achieve nearly 2dB improvements at least.
8× interpolation trained on X4K1000FPS. We also make fair comparisons on the recently proposed X4K1000FPS dataset [32], by retraining our model on it and comparing with the benchmarking results from [32], which is summarized

Table 3. Quantitative results on 2× interpolation in terms of PSNR/SSIM with model size reported in number of parameters. Best results highlighted in red.

	Vimeo-90K (septulets)	UCF101	DAVIS	SNU-FILM				Param(M)
				Easy	Medium	Hard	Extreme	
SloMo [15]	34.43/0.969	32.45/0.967	26.10/0.862	36.12/0.984	33.44/0.972	29.17/0.928	24.14/0.843	39.61
QVI [41]	34.98/0.970	32.87/0.966	27.20/0.874	39.53/0.990	36.43/0.983	31.07/0.947	24.96/0.856	29.23
CAIN [7]	34.69/0.969	32.40/0.966	27.12/0.872	39.33/0.989	35.34/0.977	30.15/0.933	24.88/0.855	42.78
ABME [27]	35.67/0.972	32.81/0.969	27.00/0.868	39.59/0.990	35.77/0.977	30.58/0.936	25.42/0.864	18.1
EDSC [5]	34.52/0.967	32.67/0.968	26.28/0.849	40.01/0.990	35.37/0.978	29.59/0.926	24.39/0.843	8.95
XVFI [32]	35.21/0.970	32.68/0.968	26.89/0.868	39.21/0.989	34.96/0.977	29.43/0.928	24.02/0.841	5.61
DAIN [3]	33.57/0.964	31.65/0.963	26.61/0.867	38.53/0.988	34.34/0.974	29.50/0.930	24.54/0.851	24.03
BMBC [26]	34.76/0.965	32.61/0.955	26.42/0.868	39.90/0.991	35.34/0.978	29.34/0.927	23.65/0.837	11.01
Softsplat [24]	35.76/0.972	32.89/0.970	27.42/0.878	-	-	-	-	12.46
VFIT-B [31]	36.96/0.978	33.44/0.971	28.09/0.888	-	-	-	-	28.09
FLAVR [16]	36.30/0.975	33.33/0.971	27.44/0.873	40.44/0.991	36.37/0.981	30.87/0.942	25.18/0.862	42.06
Ours	36.17/0.976	33.01/0.970	28.61/0.905	40.46/0.991	36.95/0.985	31.68/0.953	25.90/0.876	21.69

in Table 2. This dataset is more challenging due to the large motion universally appeared. Our approach achieves 2.7dB improvement than previous leading result. Fig. 4 shows visual comparisons of different methods. Our approach shows advantage in challenging cases such as occlusion and high frequency textures, where many existing methods would fail. Due to space limit, please refer to our supplementary material for more results.

Single-frame (2×) interpolation. Though our method is for continuous-time VFI, we evaluate it on additional 7 datasets from the single frame interpolation literature for completeness. Results are shown in Table 3. Following the convention [16], the models are trained on the *septuplet* subset of Vimeo-90K. For EDSC and BMBC the official pretrained models are used due to lacking the training code. Results of Softsplat and VFIT-B are taken from the literature [31] due to absence of both source code and models. Though not designed for single-frame interpolation, the proposed method achieves best results on 5 of 7 datasets. Our method performs worse than FLAVR and VFIT-B, which adopt larger architectures than ours, on Vimeo-90K and UCF101. These two datasets have relatively slow motion, so that advantage of our method is less significant. While, we show much better results than them on DAVIS.

4.3 Performance Analysis

In the following, we conduct several experiments to analyze the performance of the proposed approach under several conditions. All the experiments are conducted on the GoPro dataset.

Ablation analysis. The aim of this experiment is to show impact of each proposed module. Here, excluding the Image/Flow Gradient module means that image/flow gradient calculation is removed when forming the input of each iteration. By comparing the 3rd, 5th and 7th rows, we conclude that dropping either module would cause loss of performance. We also evaluate the case that flow updates are not learned at each iteration. By comparing 4th and 5th rows, we show there will be 0.3dB PSNR loss. Finally, we evaluate the effectiveness of the proposed deep reparameterization of flow gradients. To this end, we place 2-variate

Table 4. Ablation analysis on the GoPro dataset.

Component				PSNR	SSIM
Image Gradient	Flow Gradient	Flow Update	Deep reparam.		
✗	✗	✗	✗	30.56	0.935
✗	✗	✓	✗	30.72	0.937
✗	✓	✓	✓	30.78	0.937
✓	✗	✗	✗	31.28	0.942
✓	✗	✓	✗	31.52	0.944
✓	✓	✓	✗	31.53	0.944
✓	✓	✓	✓	31.73	0.947

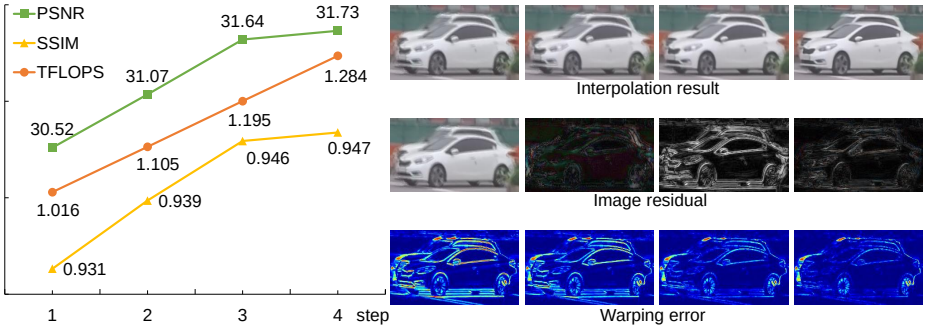

Fig. 5. Analyzing the performance of each iteration on the GoPro dataset. Left: quantitative performance in terms of PSNR, SSIM and TFLOPS as a function of iteration step. Right: the per-step interpolation result (top), learned image residual (middle) as well as the rendering error on one input image by warping the result (visualized as heatmap).

Gaussian on the 2-dimensional optical flows directly, instead of the transformed 16-dimensional deep features. The model capacity is preserved by adding extra convolutional layers when processing optical flows for fairer comparison. By comparing 6th and 7th rows, this variant results into 31.53dB PSNR with a loss of 0.2dB, demonstrating the effectiveness of deep gradient reparameterization.

Analyzing per-iteration performance. We evaluated the performance of per-iteration interpolation result in the left of Fig. 5. As expected, the result get consistently improved with more iterations. Two iterations of our model already surpasses all methods in Table 1 except FLAVR with comparable FLOPS and more iterations up to 4 can further outperform FLAVR and achieve the leading result. In the right of Fig. 5, we visualize the results on one example. As can be seen, the interpolation result gets consistently sharper, while the rendering error gets smaller, with iteration goes. The learned per-iteration image residuals identify pixels with high rendering errors in the previous step. It might be correlated with different pixels at different iterations, as a result of the quality

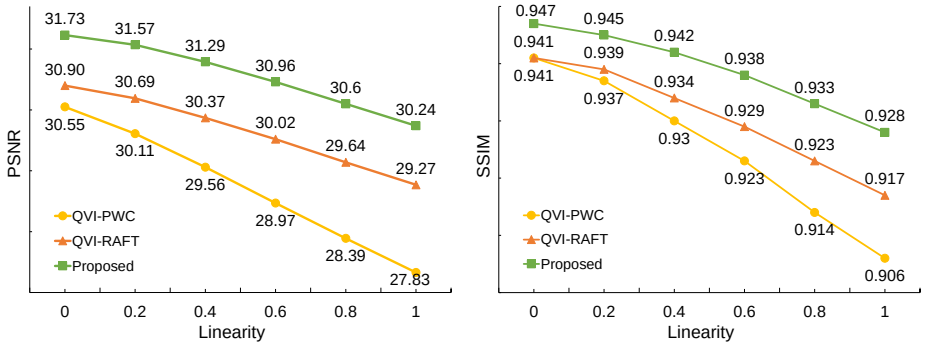

Fig. 6. Analyzing the robustness *w.r.t.* the quality of initial optical flows on different methods. PSNR and SSIM results are shown in the left and right, respectively.

of previously estimated image and flow results. However, at the last iteration, the error often converges and the residual becomes subtle.

Analyzing robustness to motion estimation. Many existing approaches and our method assume the input of high-quality optical flows. In this experiment, we aim to see what happens if the initially estimated pixel motion are of worse quality. For comparision we choose QVI as baseline, which proposed the quadratic model for intermediate flow estimation, as also followed by our work. Since QVI computes optical flows with PWCNet [35] while we use RAFT [37], for fairness we have also trained a variant, QVI-RAFT, which improves QVI with RAFT optical flows. For this evaluation, we replace the quadratic model with the simpler linear one as proposed in [44]. The interpolated forward flows from quadratic and linear models are alpha blended with different weights in $[0, 1]$. In this way, we artifically deteriorate the quality of initial flow estimations by setting a higher weight for the linear motion model. Fig. 6 shows the performance as a function of the linearity, measured as blending weight of forward flows generated by linear motion model. As expected, when motion estimation results get worse, so will be the final performance. However, our approach still achieves consistently the best results with initial motion of various quality.

5 Conclusion

In this work we present deep Bayesian video frame interpolation, a lightweight approach showing new records on 8 of 10 VFI benchmarks. Our approach by the first time formulates VFI with a posterior maximization framework optimized by learned gradient descent, whose gradient terms are principally defined.

In addition, we provide standard benchmarking results on the GoPro [23] and X4K1000FPS [32] datasets, and unified evaluation protocols on GoPro. We hope this facilitates future VFI research.

References

1. Adler, J., Öktem, O.: Solving ill-posed inverse problems using iterative deep neural networks. Inverse Problems **33**(12), 124007 (2017)
2. Adler, J., Öktem, O.: Learned primal-dual reconstruction. IEEE Transactions on Medical Imaging (TMI) **37**(6), 1322–1332 (2018)
3. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3703–3712 (2019)
4. Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **43**(3), 933–948 (2019)
5. Cheng, X., Chen, Z.: Multiple video frame interpolation via enhanced deformable separable convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2021)
6. Chi, Z., Nasiri, R.M., Liu, Z., Lu, J., Tang, J., Plataniotis, K.N.: All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In: European Conference on Computer Vision (ECCV). vol. 12372, pp. 107–123 (2020)
7. Choi, M., Kim, H., Han, B., Xu, N., Lee, K.M.: Channel attention is all you need for video frame interpolation. In: AAAI Conference on Artificial Intelligence (AAAI). pp. 10663–10671 (2020)
8. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). pp. 2758–2766 (2015)
9. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2367–2376 (2019)
10. Gui, S., Wang, C., Chen, Q., Tao, D.: Featureflow: Robust video interpolation via structure-to-texture generation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14004–14013 (2020)
11. Gupta, A., Aich, A., Roy-Chowdhury, A.K.: Alanet: Adaptive latent attention network for joint video deblurring and interpolation. In: ACM International Conference on Multimedia (ACMMM). pp. 256–264 (2020)
12. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). Arxiv preprint, 1606.08415 [cs.CV] (2016)
13. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
14. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2017–2025 (2015)
15. Jiang, H., Sun, D., Jampani, V., Yang, M., Learned-Miller, E.G., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9000–9008 (2018)

16. Kalluri, T., Pathak, D., Chandraker, M., Tran, D.: Flavr: Flow-agnostic video representations for fast frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
17. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 10236–10245 (2018)
18. Kokkinos, F., Lefkimmiatis, S.: Iterative joint image demosaicking and denoising using a residual denoising network. IEEE Transactions on Image Processing (TIP) **28**(8), 4177–4188 (2019)
19. Kokkinos, F., Lefkimmiatis, S.: Iterative residual cnns for burst photography applications. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5929–5938 (2019)
20. Lee, H., Kim, T., Chung, T.y., Pak, D., Ban, Y., Lee, S.: Adacof: Adaptive collaboration of flows for video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5316–5325 (2020)
21. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: IEEE International Conference on Computer Vision, (ICCV). pp. 4463–4471 (2017)
22. Mahajan, D., Huang, F.C., Matusik, W., Ramamoorthi, R., Belhumeur, P.: Moving gradients: a path-based method for plausible image interpolation. ACM Transactions on Graphics (TOG) **28**(3), 1–11 (2009)
23. Nah, S., Hyun Kim, T., Mu Lee, K.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3883–3891 (2017)
24. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5437–5446 (2020)
25. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: IEEE International Conference on Computer Vision (ICCV). pp. 261–270 (2017)
26. Park, J., Ko, K., Lee, C., Kim, C.: BMBC: bilateral motion estimation with bilateral cost volume for video interpolation. In: European Conference on Computer Vision (ECCV). vol. 12359, pp. 109–125 (2020)
27. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: IEEE International Conference on Computer Vision (ICCV). pp. 14539–14548 (2021)
28. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 724–732 (2016)
29. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI). pp. 234–241 (2015)
30. Shi, Z., Liu, X., Shi, K., Dai, L., Chen, J.: Video frame interpolation via generalized deformable convolution. IEEE Transactions on Multimedia (TMM) **24**, 426–439 (2021)
31. Shi, Z., Xu, X., Liu, X., Chen, J., Yang, M.H.: Video frame interpolation transformer. Arxiv preprint, 2111.13817 [cs.CV] (2021)
32. Sim, H., Oh, J., Kim, M.: Xvfi: Extreme video frame interpolation. In: IEEE International Conference on Computer Vision, (ICCV). pp. 14489–14498 (2021)

33. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. Arxiv preprint, 1212.0402 [cs.CV] (2012)
34. Su, S., Delbracio, M., Wang, J., Sapiro, G., Heidrich, W., Wang, O.: Deep video deblurring for hand-held cameras. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1279–1288 (2017)
35. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8934–8943 (2018)
36. Sun, L., Dong, W., Li, X., Wu, J., Li, L., Shi, G.: Deep maximum a posterior estimator for video denoising. International Journal of Computer Vision (IJCV) **129**(10), 2827–2845 (2021)
37. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European Conference on Computer Vision (ECCV). vol. 12347, pp. 402–419 (2020)
38. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Change Loy, C.: ESRGAN: enhanced super-resolution generative adversarial networks. In: European Conference on Computer Vision Workshops (ECCVW). vol. 11133, pp. 63–79 (2018)
39. Werlberger, M., Pock, T., Unger, M., Bischof, H.: Optical flow guided tv-l 1 video interpolation and restoration. In: Computer Vision and Pattern Recognition workshops (CVPRW). pp. 273–286 (2011)
40. Wu, Y., He, K.: Group normalization. In: European Conference on Computer Vision (ECCV). vol. 11217, pp. 3–19 (2018)
41. Xu, X., Siyao, L., Sun, W., Yin, Q., Yang, M.H.: Quadratic video interpolation. Advances in Neural Information Processing Systems (NeurIPS) **32**, 1645–1654 (2019)
42. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision (IJCV) **127**(8), 1106–1125 (2019)
43. Yu, Z., Li, H., Wang, Z., Hu, Z., Chen, C.W.: Multi-level video frame interpolation: Exploiting the interaction among different levels. IEEE Transactions on Circuits and Systems for Video Technology **23**(7), 1235–1248 (2013)
44. Yu, Z., Zhang, Y., Liu, D., Zou, D., Chen, X., Liu, Y., Ren, J.S.: Training weakly supervised video frame interpolation with events. In: IEEE International Conference on Computer Vision, (ICCV). pp. 14589–14598 (2021)