Long-term Video Frame Interpolation via Feature Propagation

Dawit Mureja Argaw In So Kweon
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
{dawitmureja, iskweon77}@kaist.ac.kr

Abstract

Video frame interpolation (VFI) works generally predict intermediate frame(s) by first estimating the motion between inputs and then warping the inputs to the target time with the estimated motion. This approach, however, is not optimal when the temporal distance between the input sequence increases as existing motion estimation modules cannot effectively handle large motions. Hence, VFI works perform well for small frame gaps and perform poorly as the frame gap increases. In this work, we propose a novel framework to address this problem. We argue that when there is a large gap between inputs, instead of estimating imprecise motion that will eventually lead to inaccurate interpolation, we can safely propagate from one side of the input up to a reliable time frame using the other input as a reference. Then, the rest of the intermediate frames can be interpolated using standard approaches as the temporal gap is now narrowed. To this end, we propose a propagation network (PNet) by extending the classic feature-level forecasting with a novel motion-to-feature approach. To be thorough, we adopt a simple interpolation model along with PNet as our full model and design a simple procedure to train the full model in an end-to-end manner. Experimental results on several benchmark datasets confirm the effectiveness of our method for long-term VFI compared to state-of-the-art approaches.

1. Introduction

Video frame interpolation (VFI) aims at predicting one or more intermediate frames from a given frame sequence. Given inputs $\langle x_t, x_{t+n}\rangle$, where n is the frame gap between the inputs, existing VFI works generally follow two steps. First, they estimate the motion between x_t and x_{t+n} using off-the-shelf motion estimation modules or by imposing motion constraints. Then, they warp the inputs to the target time and synthesize an intermediate frame.

VFI works target temporal super-resolution on a premise that the frame rate of the input sequence is often already sufficiently high. We have experimentally verified this notion by evaluating several state-of-the-art VFI methods [2, 12, 17, 19, 30] on input sequences sampled at different frame rates. Even though a reasonable performance decrease is an expected phenomenon, we observed a significant drop in performance when the frame rate of the input sequence decreases (see Table 1), highlighting that interpolating frames becomes very challenging as the temporal distance between consecutive frames increases. Moreover, far less attention has been given to this problem in past literature as most evaluations have been done on videos with fixed frame rate (mostly 30 fps). We argue that the main reason behind this limitation is partly associated with the working principle of VFI works. If the estimated motion between inputs is inaccurate, then the interpolated frame synthesized by time warping the inputs with the estimated motion will also likely be inaccurate. This is particularly problematic when the temporal gap between input frames is large as existing flow or kernel based motion estimation modules can not effectively handle large motions.

In this work, we tackle the long-term video interpolation problem and propose a general VFI framework robust to relatively low frame rates. Specifically, when there is a large gap between input frames, instead of predicting the motion between the inputs which will likely be imprecise and eventually lead to inaccurate interpolation, we conjecture that we can safely propagate from one side of the input to a reliable extent of time using the other input as a useful reference, *i.e.* given $\langle x_t, x_{t+n}\rangle$, we propagate up to $x_{t+\Delta t}$ from the side of the first input x_t and we similarly propagate up to $x_{t+n-\Delta t}$ from the side of the second input x_{t+n}, where Δt is the extent of propagation. This is intuitive because the intermediate frames in the neighborhood of x_t will most likely depend on x_t compared to x_{t+n}, and vice versa. Once we propagate to a reliable time frame from both sides, the rest of the intermediate frames between $x_{t+\Delta t}$ and $x_{t+n-\Delta t}$ can be interpolated using existing interpolation approaches as the temporal gap is now reduced to $n - 2\Delta t$.

To this end, we propose a propagation network (PNet) that predicts future frames by relying more on one of the inputs while attending the other. We accomplish this by extending the classic feature-to-feature (F2F) forecasting

[5,8,36,37,43,44] with a novel motion-to-feature (M2F) approach, where we introduce optical flow as another modality to guide the propagation of features and to enforce temporal consistency between the propagated features. Unlike feature supervision which makes a network more dependent on the semantics of input frames, our motion supervision allows the network to focus on the motion between inputs and ensures features are propagated accordingly irrespective of the contents of the images. Moreover, while most F2F works focus on predicting task-specific outputs such as segmentation maps, we perform RGB forecasting by designing a frame synthesis network that reconstructs frames from the propagated features in a coarse-to-fine manner.

We experimentally show that the proposed PNet can be used as a plug in module to make existing state-of-the-art VFI approaches [2, 12, 17, 19, 30] robust particularly when there is a considerable temporal gap between inputs. To be thorough, we adopt a light version of SloMo [17] along with PNet as our full model and devise a simple, yet effective, procedure to successfully train the full model in an end-to-end manner. We comprehensively analyze our work and previous methods on several widely used datasets [11, 27, 40] and confirm the favorability of our approach. Moreover, we carry out ablation experiments to shed light on the network design and loss function choices.

2. Related Works

Video Frame Interpolation. Early conventional methods [23, 49] relied on the optical flow between inputs and the given image formation model for synthesizing intermediate frames. Recently, several deep network based VFI approaches have been proposed. While some works [7, 21] directly predict intermediate frames, most existing approaches embed motion estimation modules in their framework. According to the type of the motion estimation module used, VFI works can be broadly categorized as: phase-based, kernel-based, flow-based and a mix of the last two. Early phase-based works [25, 26] formulated the temporal change between the inputs as phase shifts. On the other hand, kernel-based methods such as AdaConv [31] and SepConv [30] estimate spatially adaptive 2D and 1D kernels, respectively. Meanwhile, due to the significant progress achieved in optical flow estimation research, flow-based interpolation approaches [17, 20, 28, 29, 32, 34, 35, 39, 46, 47] have grown to be popular. DVF [20] and SloMo [17] estimated flows between input frames and directly warped them to the target intermediate time while [28, 29, 32, 46] used a trainable frame synthesis network on the warped frames to predict the intermediate frame.

DAIN [2] and MEMC-Net [3] combined kernel-based and flow-based models. AdaCoF [19] proposed a generalized warping module via adaptive collaboration of flows for VFI. Recently, several works [1,4,6,12,34,35,38,45,48] have focused on addressing the different limitations of the VFI approaches discussed thus far. However, most existing works assume that the input sequence frame rate is often already sufficiently high, hence, long-term VFI has received far less attention in the past literature. Our work tackles this problem by proposing a novel framework that combines frame propagation and interpolation.

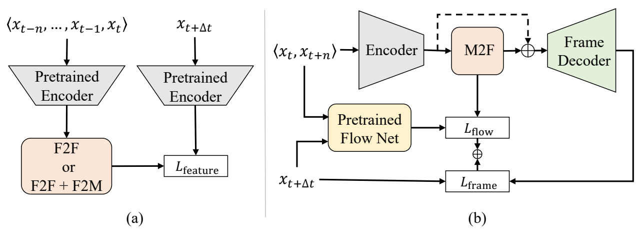

Figure 1. (a) Overview of previous feature-level propagation formulation, (b) Proposed problem formulation.

Feature Propagation. Feature-to-feature (F2F) forecasting inputs intermediate features of the past frames and anticipates their future counter parts. This approach have been previously used for action recognition [43], instance segmentation [8, 22, 42] and semantic segmentation [5, 36, 37, 44] tasks. Recently, Šarić *et al.* [37] proposed a feature-to-motion (F2M) module to compliment the classic F2F approach. Previous F2F or F2F + F2M based works use the encoder part of task-oriented pretrained models (*e.g.* semantic segmentation) to extract intermediate features of a set of inputs and use the extracted features to forecast features of future frame. The forecasting module is trained by optimizing the loss between the forecasted features and the extracted features of the future frame (see Fig. 1a). During inference, the task-specific output (*e.g.* segmentation map) is obtained by feeding the forecasted feature into the decoder part of the pretrained model.

In this work, we extend feature-level propagation to a relatively unexplored task, *i.e.* long-term VFI, by presenting a novel motion-to-feature (M2F) approach. Our approach is different from previous feature-level propagation methods in the following aspects. First, we introduce motion (in the form of optical flow) as another modality to guide the forecasting of features and to enforce temporal consistency between forecasted features. Second, we perform RGB forecasting by designing a frame synthesis network that outputs future frames from forecasted features. The proposed formulation is summarized in Fig. 1b.

3. Methodology

Given a pair of consecutive frames $\langle x_t, x_{t+n} \rangle$ from a low-frame-rate video, we aim to generate a high quality, high-frame-rate sequence $\{x_t, x_{t+1}, \ldots, x_{t+n-1}, x_{t+n}\}$ by jointly optimizing interlinked propagation and interpolation networks in an end-to-end manner. The overview of our proposed framework is shown in Fig. 2.

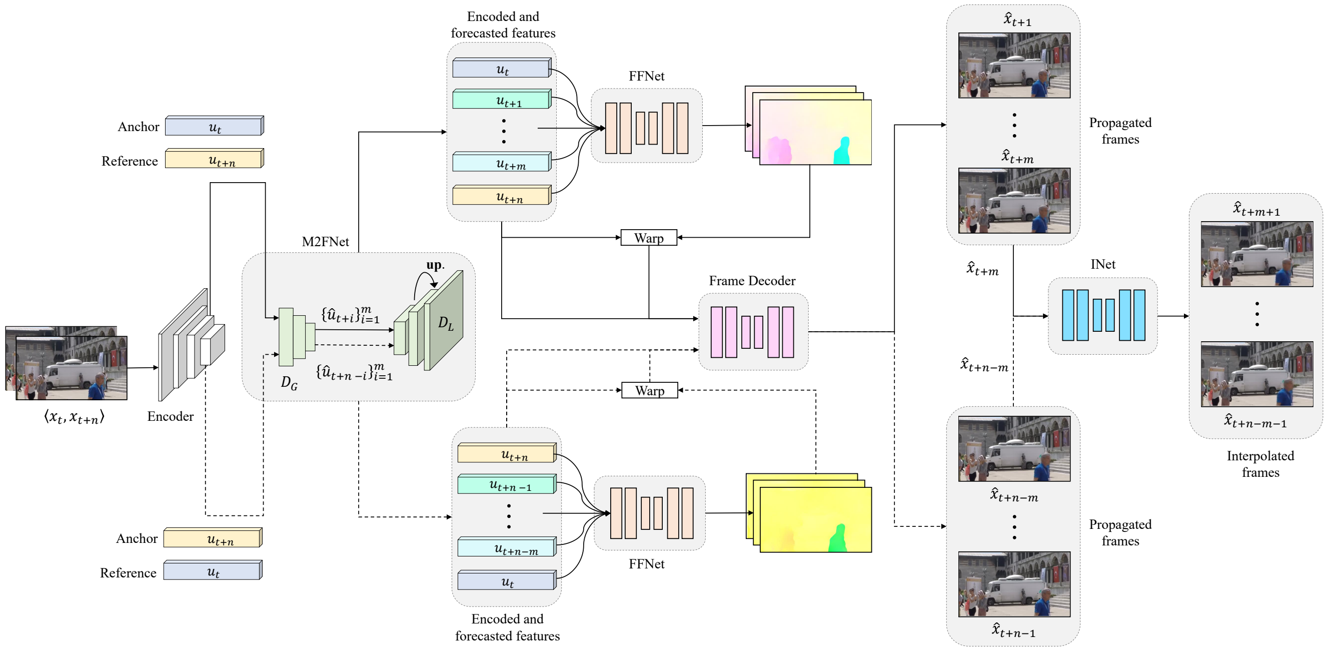

Figure 2. Overview of the proposed propagation-interpolation network (P-INet). The propagation network (PNet) consists of an encoder network for feature extraction, M2FNet to bidirectionally propagate features using the encoded features as anchor and reference features, FFNet to estimate optical flow between features for motion supervision, and a frame decoder to reconstruct frames from propagated features. An interpolation network (INet) is used to interpolate the intermediate frames between the end propagated frames.

3.1. Propagation Network (PNet)

We use an encoder-decoder architecture for PNet. First, we design an encoder network $\mathcal{E}$ to extract features from the input frames $\langle x_t, x_{t+n}\rangle$ in a top-down manner (Eq. (1)). The encoder is a feed-forward network with 5 convolutional blocks, each block containing 2 layers of convolution with kernel size 3×3. Except for the first block, features are downsampled to half of their spatial size and the number of channels is doubled after each convolutional block.

$$\{u_t^l\}_{l=1}^k = \mathcal{E}(x_t) \quad \{u_{t+n}^l\}_{l=1}^k = \mathcal{E}(x_{t+n}) \tag{1}$$

where l denotes a level in the feature pyramid with a total of k levels ($k = 5$ in our experiments) and u_t^l denotes an encoded feature of the first input x_t at level l. To propagate to the frame $x_{t+\Delta t}$ (from x_t side), we first perform feature-level forecasting using the encoded features of x_t and x_{t+n}, *i.e.* $\{u_t^1, \dots, u_t^k\}$ and $\{u_{t+n}^1, \dots, u_{t+n}^k\}$, as *anchor* and *reference* features, respectively. We then use a decoder network to reconstruct $x_{t+\Delta t}$ from the propagated features in a bottom-up manner.

Motion-to-Feature Forecasting. We design a motion-to-feature network (M2FNet) to forecast the future counterparts of the encoded features. M2FNet takes the anchor and reference features as inputs and anticipates the motion to propagate the anchor feature to its future counterparts. Then, it transforms the anchor feature according to the estimated motion. To take the complex motion dynamics between the input frames into account and better exploit the inter-frame spatio-temporal correlation, we propagate to multiple frames simultaneously. M2FNet has 2 components: global ($\mathcal{D}_G$) and local ($\mathcal{D}_L$) motion decoders. $\mathcal{D}_G$ learns the global motion between the encoded features, and predicts affine transformation parameters $\theta_{[R|T]}$ to spatially transform the anchor feature to its future counterparts (see Eq. (2) and Eq. (3)). We use spatial transformer network [16] for for $\mathcal{D}_G$.

$$\left\{\theta_{[R_{t+i}|T_{t+i}]}\right\}_{i=1}^m = \mathcal{D}_G^l\left(u_t^l \mid\mid u_{t+n}^l\right) \tag{2}$$

$$\left\{\hat{u}_{t+i}^l\right\}_{i=1}^m = \mathsf{transform}\left(u_t^l,\ \left\{\theta_{[R_{t+i}|T_{t+i}]}\right\}_{i=1}^m\right) \tag{3}$$

where $\mid\mid$ denotes channel-wise concatenation, m refers to the number of features (frames) propagated from the anchor feature u_t and $\hat{u}_{t+i}^l$ represents the output of $\mathcal{D}_G$ at time step $t+i$ and feature level l. As $\mathcal{D}_G$ is limited to learning only non-local motion, in order to capture the locally varying motion, we further refine the outputs of $\mathcal{D}_G$ with a local motion decoder ($\mathcal{D}_L$). $\mathcal{D}_L$ inputs the globally transformed feature $\hat{u}_{t+i}$ along with the anchor and reference features, and outputs the forecasted feature u_{t+i} (see Eq. (4)). $\mathcal{D}_L$ has 3 densely connected convolutional layers [13] each with kernel size 3×3 and stride 1. As the forecasted feature u_{t+i} is decoded in a coarse-to-fine manner, a residual connection is built by feeding the the upsampled decoded feature from previous feature level $l+1$ into $\mathcal{D}_L$ as shown in Eq. (4). A deconvolution layer of kernel size 4×4 and stride size 2 is used to upsample ($\times 2$) features.

$$u_{t+i}^l = \mathcal{D}_L^l\left(\hat{u}_{t+i}^l \mid\mid u_t^l \mid\mid u_{t+n}^l \mid\mid \mathsf{up}.(u_{t+i}^{l+1})\right) \tag{4}$$

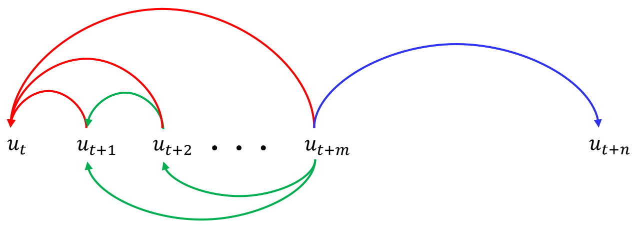

Figure 3. Optical flow estimation pattern when propagating to m future counterparts.

where $i = \{1, \dots, m\}$, up. stands for upsampling and u^l_{t+i} is the forecasted feature at level l. In principle, $\mathcal{D}_L$ can decode both local and global motions. However, explicitly modelling global motions with $\mathcal{D}_G$ is shown to be effective for the task at hand (see Sec. 5).

Optical Flow Estimation. M2FNet learns to propagate features via motion supervision. For instance, to ensure that the forecasted feature u_{t+i} can be reconstructed to x_{t+i}, we constrain the endpoint error between the flows $f_{x_{t+i} \to x_t}$ and $\hat{f}_{u_{t+i} \to u_t}$, which are computed between frames $\langle x_t, x_{t+i} \rangle$ and features $\langle u_t, u_{t+i} \rangle$, respectively. As the ground truth flow $f_{x_{t+i} \to x_t}$ does not exist for real high-speed videos, we generate a pseudo-ground truth flow using pretrained state-of-the-art optical flow models [15,41]. To estimate the flow $\hat{f}_{u_{t+i} \to u_t}$, we design a feature flow network (FFNet) which inputs two sets of features, *i.e.* $\{u^l_{t+i}\}^k_{l=1}$ and $\{u^l_t\}^k_{l=1}$, and regresses a flow in a coarse-to-fine manner. We use the architecture of the optical flow estimator in PWC-Net [41] for FFNet. To predict the flow between the anchor feature u_t and a forecasted feature u_{t+i}, we preform the following steps. First, at each level l, we backwarp the second feature u^l_t (to the first feature u^l_{t+i}) with $\times 2$ upsampled flow from previous level $l+1$ (see Eq. (5)). A correlation layer [10, 14,41] is then used to compute the cost volume between the first feature u^l_{t+i} and the backwarped feature w^l_{t+i}. The first feature, the cost volume and the upscaled flow are feed into FFNet to predict a flow as shown in Eq. (6).

$$w^l_{t+i} = \text{backwarp}(u^l_t, \text{up}.(\hat{f}^{l+1}_{t+i \to t})) \tag{5}$$

$$\hat{f}^l_{t+i \to t} = \text{FFNet}(u^l_{t+i} \oplus \text{corr}.(u^l_{t+i}, w^l_{t+i}) \oplus \text{up}.(\hat{f}^{l+1}_{t+i \to t})) \tag{6}$$

Fig. 3 depicts the flow estimation pattern when propagating to m future counterparts of the anchor feature u_t. We compute several optical flows to ensure that features are propagated by anticipating the complex motion between the input frames and not simply in a linear manner. Specifically, we estimate the flow between the anchor feature and each forecasted feature (shown in red in Fig. 3) so that M2FNet learns to forecast features according to their proximity to the anchor feature *i.e.* $\mathcal{D}_G$ and $\mathcal{D}_L$ decode smaller motions for features close to the anchor feature and larger motions for those further away. We also estimate flow between the forecasted features themselves (depicted in green) in order to account for inter-frame motion between propagated features. To address any potential ambiguity in the direction of propagation, we compute the flow between the last forecasted feature and the reference feature (shown in blue).

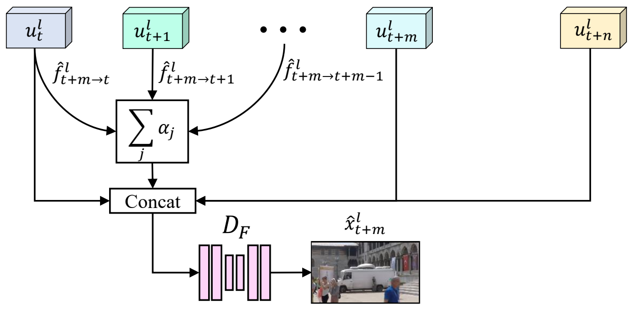

Figure 4. Frame synthesis of $\hat{x}_{t+m}$ at feature level l.

Feature-to-Frame Decoding. The forecasted features and optical flows are then used to decode frames. For this purpose, we design a frame decoder ($\mathcal{D}_F$) which regresses frames from the corresponding forecasted features. When decoding the current frame, $\mathcal{D}_F$ incorporates contextual and temporal information from the past frames via attention mechanism (see Fig. 4). This is accomplished by warping the past features into the current time step with the corresponding estimated optical flows and combining the warped features using attention weights as shown in Eq. (7). The attention vector (α) is a learnable, one-dimensional weight parameter with elements initially set to 1. For better reconstruction of occluded regions in the predicted frames, $\mathcal{D}_F$ also uses the anchor and reference features. Like the feature forecasting and flow estimation steps, frames are decoded in a coarse-to-fine manner. At each feature level l, $\mathcal{D}_F$ inputs the forecasted feature (u^l_{t+i}), attended past features (v^l_{t+i}), encoded features (u^l_t and u^l_{t+n}) and $\times 2$ upscaled frame predicted from previous level $l+1$ (see Eq. (8)). $\mathcal{D}_F$ is composed of 3 densely connected convolutional layers each with kernel size 3×3 and stride 1, where the last layer outputs a frame.

$$v^l_{t+i} = \sum_{j=t}^{t+i-1} \alpha_j.\text{backwarp}(u^l_j, \hat{f}^l_{t+i \to j}) \tag{7}$$

$$\hat{x}^l_{t+i} = \mathcal{D}^l_F(u^l_{t+i} \,||\, v^l_{t+i} \,||\, u^l_t \,||\, u^l_{t+n} \,||\, \text{up}.(\hat{x}^{l+1}_{t+i})) \tag{8}$$

3.2. Propagation-Interpolation Network (P-INet)

The proposed propagation network (PNet) can be used either as a stand-alone model or a plug-in module with existing VFI works (see Sec. 4). However, we have experimentally observed few trade-offs. First, when the temporal gap between inputs is small, PNet gives a sub-optimal performance compared to the state-of-the-art VFI approaches. This is mainly because PNet, by design nature, propagates

Algorithm 1: Training strategy for the P-INet

```
Input : $\langle x_t, x_{t+n} \rangle$          // $n$ is the frame gap
Output : $\hat{x}_{t+i}$, where $1 < i < n$
Let $N$ be the maximum frame gap in the dataset, $M$ be the upper
  limit for small frame gap, and $\Delta t(n)$ be a reliable time frame of
  propagation which is dependent on $n$
foreach input sample do
    if $n \leq M$ then // small gap
        $\hat{x}_{t+i}$ = INet $(x_t, x_{t+n})$ for all $i$
    else // large gap ($M < n \leq N$)
        if $i \leq \Delta t(n)$ then // propagate from $x_t$
            $\hat{x}_{t+i}$ = PNet $(x_t, x_{t+n})$
        else if $\Delta t(n) < i < n - \Delta t(n)$ then
            // propagate and interpolate
            $\hat{x}_{t+\Delta t(n)}$ = PNet $(x_t, x_{t+n})$
            $\hat{x}_{t+n-\Delta t(n)}$ = PNet $(x_{t+n}, x_t)$
            $\hat{x}_{t+i}$ = INet $(\hat{x}_{t+\Delta t(n)}, \hat{x}_{t+n-\Delta t(n)})$
        else // propagate from $x_{t+n}$
            $\hat{x}_{t+i}$ = PNet $(x_{t+n}, x_t)$
    end
end
```

to future counterparts of the anchor frame by using the other input as a reference, *i.e.* it relies more on one of the inputs by default. This leads to a performance trade-off since an interpolation method, intuitively speaking, should evenly rely on both inputs when there is a small gap between them. Second, as expected, the quality of frames eventually deteriorates as we propagate further away from the anchor frame. For completeness of the proposed approach, thereby alleviating the observed trade-offs, we adopt a light version of SloMo [17] as an interpolation network (INet) along with PNet as our full model, *i.e.* P-INet. The SloMo used in our work contains 50% fewer weight parameters compared to the model used in [17]. We train P-INet in an end-to-end manner by guiding it to propagate, interpolate or propagate and interpolate depending on the *temporal gap* between inputs and the *timestamp* of the intermediate frame to be predicted as summarized in Algo. 1.

We use a bidirectional propagation and interpolation scheme since it gives us the flexibility to experiment with long-range temporal gaps. The reliable time frame of propagation $\Delta t(n)$ is defined as $\min(\lceil (n-M)/2 \rceil, M)$, where $n > M$. In other words, PNet adaptively propagates until the temporal gap between the end propagated frames is less than or equal to M. As most VFI works conduct experiments at 30 fps by downsampling 240 fps videos ($\approx$ frame gap of 8) and because our approach uses 3 intervals, we set $M = 8$ and $N = 24$ during training. We experiment with up to 30 frame gaps during testing to analyze if our approach extends to even larger gaps (see Sec. 4).

3.3. Loss Functions

We train our network in an end-to-end manner by jointly optimizing the estimated flows, propagated frames and interpolated frames. To train M2FNet, we compute the end-point error between the estimated flows and pseudo-ground truth flows across different levels as shown in Eq. (9). To propagate to m features, a total of $\frac{m}{2}(m+1)+1$ flows are estimated. We also investigate training our network by selectively optimizing some of the flows (see Sec. 5).

$$\mathcal{L}_{\text{M2FNet}} = \sum_{i=1}^{\frac{m}{2}(m+1)+1} \sum_{l=1}^{k} \omega_1^l \left| f_i^l - \hat{f}_i^l \right|_2 \tag{9}$$

where ω_1^l is a flow loss weight coefficient at level l. For sharp frame decoding, we train PNet with the multi-scale ℓ_1 photometric loss. We also use gradient difference loss [24] ($\mathcal{L}_{\text{GDL}}$) between the predicted frames and their ground truth to mitigate blurry predictions (see Eq. (10)).

$$\mathcal{L}_{\text{PNet}} = \sum_{i=1}^{m} \sum_{l=1}^{k} \omega_2^l \left| x_i^l - \hat{x}_i^l \right|_1 + \mathcal{L}_{\text{GDL}}(x_i, \hat{x}_i) \tag{10}$$

where ω_2^l is a frame loss weight coefficient at level l. We use the training loss of SloMo discussed in Section 3.3 of [17] to train INet. We refer the reader to [17] for details. The total training loss for P-INet is defined as weighted sum of all losses as shown in Eq. (11).

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{M2FNet}} + \lambda_2 \mathcal{L}_{\text{PNet}} + \lambda_3 \mathcal{L}_{\text{INet}} \tag{11}$$

4. Experiment

Datasets. Most existing VFI works use Vimeo-90K [46] dataset which has 51312 triplets, where each triplet contains 3 consecutive video frames. However, as this dataset is not applicable to train a network for long-term VFI, we generate a dataset by sampling frames at different fps from high-speed video datasets. For this purpose, we use Adobe240 [40], GOPRO [27] and Need-for-Speed (NfS) [11] datasets, which contain 133, 33 and 100 videos, respectively. These datasets provide 240 fps videos which capture diverse combination of camera and object motions in real-world scenarios, and thus are suitable for the task at hand. A majority of the videos, however, have less than 1000 frames which makes it challenging to extract enough training samples with large temporal gaps. Hence, instead of training separately on each dataset, we used a total of 176 videos (103 from Adobe240, 3 from GOPRO and 70 from NfS) for training. The remaining 90 videos (30 from each dataset) are used for testing. We prepare train and test sets by extracting samples with variable length ranging from 9 to 31 consecutive frames in a video. In other words, we sample video clips at different frame rates in the range of approximately 30 fps to 8 fps, respectively. Following [46], we resize each frame in the dataset to a resolution of 448×256 to suppress noise and create consistency in size across videos.

Implementation Details. We implement our network in PyTorch [33] and optimize it using Adam [18] with parameters β_1 , β_2 and *weight decay* fixed to 0.9, 0.999 and $4e-4$, respectively. The loss weight coefficients are set

Table 1. Quantitative comparison at different fps. The numbers in red and blue represent the best and second best results, respectively.

Method	Adobe240 [40]						GOPRO [27]						NfS [11]					
	30 fps		15 fps		8 fps		30 fps		15 fps		8 fps		30 fps		15 fps		8 fps	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SepConv [30]	29.91	0.915	23.94	0.811	19.88	0.707	28.64	0.871	23.23	0.694	19.74	0.560	31.84	0.915	26.73	0.811	23.00	0.707
SloMo [17]	30.03	0.917	24.30	0.818	20.17	0.717	29.03	0.917	23.58	0.818	19.99	0.718	31.83	0.917	26.95	0.818	23.19	0.717
DAIN [2]	30.53	0.924	24.39	0.824	20.21	0.721	29.25	0.924	23.63	0.824	20.18	0.721	32.46	0.924	27.19	0.824	23.36	0.720
AdaCoF [19]	30.14	0.896	24.11	0.741	20.07	0.567	29.05	0.876	23.49	0.701	19.89	0.571	32.28	0.919	27.05	0.819	23.23	0.719
FeFlow [12]	30.48	0.902	24.19	0.737	20.04	0.576	29.30	0.921	23.51	0.822	19.82	0.724	32.42	0.921	27.05	0.822	23.16	0.724
INet	30.30	0.920	24.21	0.819	20.12	0.718	29.17	0.919	23.59	0.821	20.04	0.722	32.03	0.920	26.99	0.822	23.27	0.721
P-INet	30.30	0.920	27.10	0.890	24.00	0.810	29.17	0.919	26.45	0.879	23.90	0.804	32.03	0.920	28.98	0.874	26.23	0.798

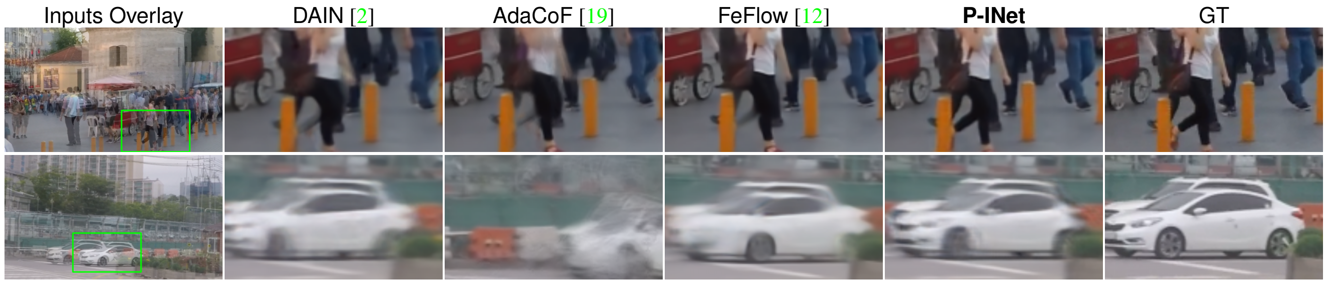

Figure 5. Qualitative comparison of our method and state-of-the-art VFI approaches for inputs with large temporal gap.

to $\omega^5 = 0.08$, $\omega^4 = 0.04$, $\omega^3 = 0.02$, $\omega^2 = 0.01$ and $\omega^1 = 0.005$ from the lowest to the highest resolution, respectively, for both $\mathcal{L}_{\text{M2FNet}}$ and $\mathcal{L}_{\text{PNet}}$. We train P-INet for 200 epochs with the learning rate initially set to $\lambda = 1e-4$ and gradually decayed by half at 100, 150 and 175 epochs. For the first 40 epochs, we only train the M2FNet by setting $\lambda_1 = 1$, $\lambda_2 = 0$ and $\lambda_3 = 0$ to facilitate motion estimation and feature propagation. For the remaining epochs, we fix λ_1, λ_2 and λ_3 to 1. We use a mini-batch size of 4 and randomly crop image patches of size 256×256 during training. The pseudo-ground truth optical flows for supervising M2FNet are computed on-the-fly using FlowNet 2 [15].

4.1. Experimental Results

In this section, we comprehensively analyze our work and several state-of-the-art VFI approaches for which open source implementations are available. These include SepConv [30], SloMo [17], DAIN [2], AdaCoF [19] and FeFlow [12]. For fair comparison, we retrain these models using our training set by following their official code. We deploy a multi-frame interpolation training scheme for P-INet, SloMo [17] and DAIN [2] as it is possible while we use single-frame interpolation scheme for others. For quantitative evaluation, we use PSNR and SSIM metrics.

Temporally Robust VFI. Here, we analyze the robustness of different VFI models for input sequences with different temporal gaps. In Table 1, we compare our approach and state-of-the-art VFI methods on single frame interpolation of test videos sampled at 3 different frame rates: 30 fps, 15 fps and 8 fps. As can be inferred from Table 1, P-INet performs competitively for smaller temporal gaps and significantly better than SOTA approaches for larger temporal gaps. For instance, our approach outperforms the second best method, *i.e.* DAIN [2], by an average margin of of 2.44 dB and 3.51 dB at 15 fps and 8 fps, respectively. Moreover, the performance gap for DAIN between 30 fps and 8 fps is 9.50 dB on average. By contrast, the performance gap for our model is 5.79 dB. This shows the effectiveness of our approach for low-frame-rate videos. It can also be noticed from Table 1 that the joint training of PNet and INet is beneficial even for smaller frame gaps. For instance, INet outperforms SloMo [17] by an average margin of 0.2 dB at 30 fps. In Fig. 5, we qualitatively compare the frames interpolated by our method and SOTA VFI approaches for input samples with large temporal gap. As can be seen from the figure, our approach interpolates sharper images with clearer contents compared to other VFI approaches.

PNet with VFI Methods. To highlight the versatility of the proposed PNet for long-term VFI, we couple PNet with VFI approaches and perform intermediate frame interpolation for input sequences with relatively large frame gap ranging from 11 to 30. Following the procedure in Algo. 1, we first propagate bidirectionally using PNet from a pretrained P-INet. Then, we interpolate an intermediate frame between the propagated frames using state-of-the-art VFI methods [9, 12, 17, 19]. The averaged results on the 3 datasets are plotted in Fig. 6. As can be inferred from the figure, the cascade models consistently outperform their vanilla baseline by a notable margin. The qualitative analysis in Fig. 7 also shows that, when the temporal distance between the inputs is large, incorporating PNet results in interpolated frames with more *accurate* contents compared to directly using SOTA VFI approaches. The static regions in Fig. 7 appear slightly less sharp for cascade models most

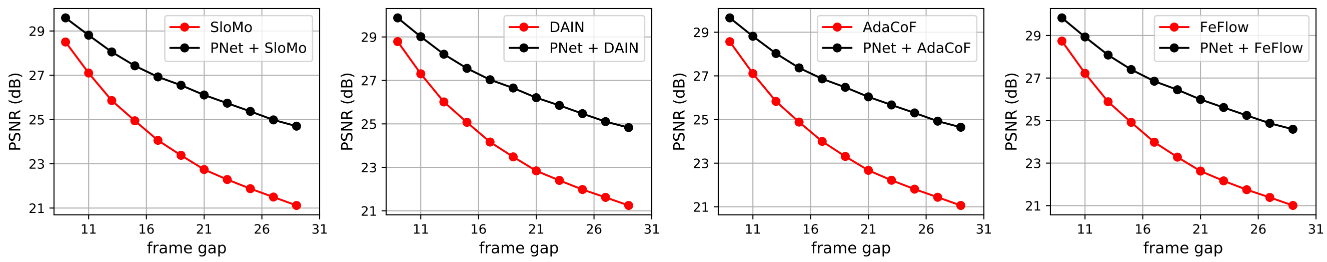

Figure 6. Quantitative analysis of PNet with state-of-the-art VFI approaches.

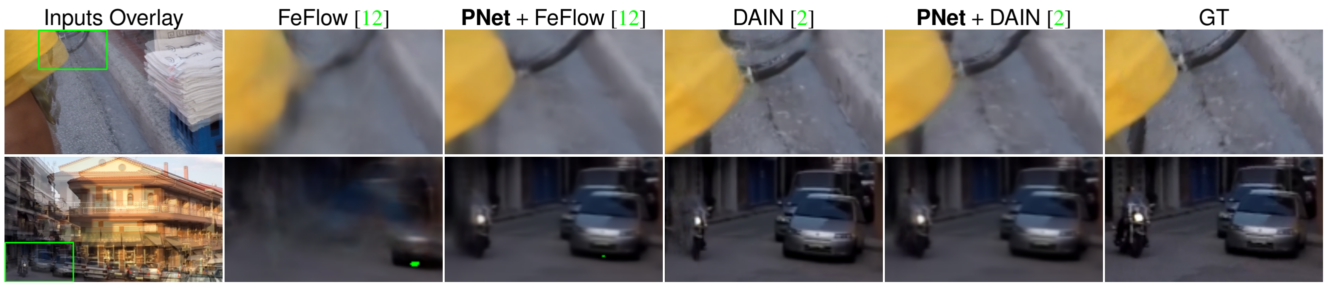

Figure 7. Qualitative analysis of PNet cascaded with state-of-the-art VFI approaches.

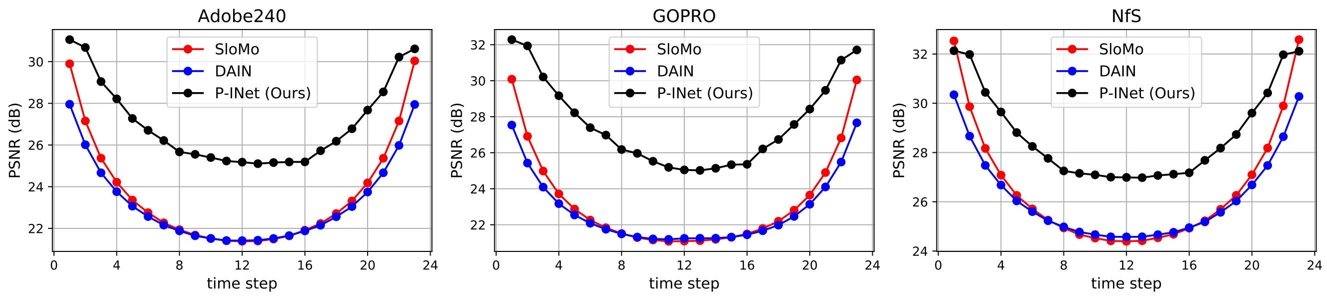

Figure 8. Quantitative analysis of intermediate frames at different time steps for long-term VFI.

likely because the interpolation model uses the output of PNet as inputs rather than the raw input frames.

Long-term Multi-Frame Interpolation. Beyond evaluating the robustness of VFI approaches at different frame rates, we analyze the quality of the intermediate frames interpolated during a direct *very low* fps $\rightarrow$ *very high* fps upsampling. We perform 10 fps $\rightarrow$ 240 fps up-conversion in a single pass and measure the quality of the interpolated frames at each time step. In Fig. 8, we compare our approach with SloMo [17] and DAIN [2] since they are also capable of multi-frame interpolation. As expected, performance generally decreases as we move to the middle time step from both sides. However, it can be noticed from Fig. 8 that there is a rapid performance drop for SloMo and DAIN compared to P-INet. For instance, the average performance range, *i.e.* the difference between largest and smallest PSNR values averaged over the 3 datasets, for SloMo is 8.62 dB. By contrast, the average performance range for P-INet is 6.11 dB. Instead of interpolating frames based on pre-computed motion that will likely be inaccurate due to large motion, our model adapts to propagate and interpolate frames, which explains the significant performance gain achieved over state-of-the-art approaches particularly for central time steps.

Optical Flow. Fig. 9 depicts the feature flows estimated by our network in comparison with the corresponding pseudo-ground truth (p-GT) flows. As can be seen from Fig. 9, our model reasonably anticipates the accurate motion to propagate features. To further confirm if $\mathcal{D}_G$ and $\mathcal{D}_L$ in the M2FNet properly learned to decode motions for feature propagation, we quantitatively analyze the optical flows estimated between the *anchor* and the *forecasted* features. To purely evaluate the magnitude of motion, we compute the sum of the absolute value of the estimated flows. In Fig. 10, we plot a heat map of the magnitude of the estimated flows (rescaled between 0 & 1) for different temporal gaps, where $z_i = \frac{1}{2}(\sum|\hat{f}_{t+i\rightarrow t}| + \sum|\hat{f}_{t+n-i\rightarrow t+n}|)$. We can infer two key things from Fig. 10. First, the proximity of the forecasted feature to the anchor is directly related to the magnitude of the estimated flow, *i.e.* $\mathcal{D}_G$ and $\mathcal{D}_L$ decode smaller motions for closer features and larger motion for those that are far. Second, M2FNet is implicitly aware of the relative temporal distance between inputs, *i.e.* the magnitude of the forecasted flows increases for increasing frame gap.

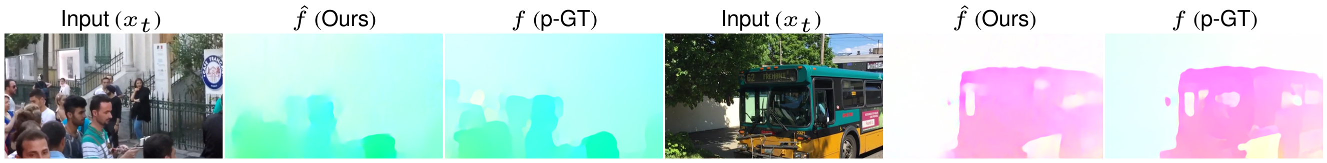

Figure 9. Qualitative analysis of the estimated optical flows between features in comparison with the pseudo-ground truth (p-GT) flows.

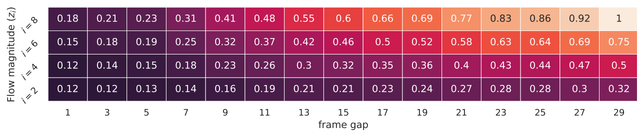

Figure 10. Quantitative analysis of the estimated flows.

5. Ablation Studies

Here, we present ablation experiments on different components of P-INet. We evaluate the quality of all *propagated* frames during long-term VFI (10 fps $\rightarrow$ 240 fps) for Adobe240 [40] and GOPRO [27] test videos (see Table 2).

Loss Functions. To highlight the importance of using optical flow as a guidance for feature propagation, we forecast features without estimating flows and directly regress frames from the respective forecasted features, *i.e.* P-INet is trained without $\mathcal{L}_{\text{M2FNet}}$. A network trained without motion supervision performed significantly worse compared to a model trained with motion supervision. We also confirmed the contribution of the different groups of flows estimated in Sec. 3. It can be inferred from Table 2 that estimating optical flows between the forecasted features is crucial as a network trained without inter-frame motion supervision (shown in green in Fig. 3) gives a subpar performance. Moreover, we studied the importance of addressing potential directional ambiguity by constraining the optical flow between the end propagated feature and the reference feature (shown in blue in Fig. 3). As can be seen from Table 2, training a network without direction supervision results in a performance decrease of 0.48 dB. We analyze the benefit of the gradient difference loss [24] ($\mathcal{L}_{\text{GDL}}$) in mitigating blurry frame predictions. It can be noticed from Table 2 that training our model with $\mathcal{L}_{\text{GDL}}$ improves performance by an average margin of 0.65 dB.

M2FNet. We examine the importance of global ($\mathcal{D}_G$) and local ($\mathcal{D}_L$) motion decoders in M2FNet. First, we only use $\mathcal{D}_G$ for decoding motion. This resulted in a subpar network performance as $\mathcal{D}_G$ is limited to anticipating only global motion at a feature-level and the local motion apparent in the test videos can not be effectively forecasted. In theory, $\mathcal{D}_L$ can decode both local and global motions (without the need to explicitly model global motions with $\mathcal{D}_G$) as CNNs are effective in motion estimation tasks [15, 41]. This is also empirically evident as a network trained only using $\mathcal{D}_L$ gives a competitive performance. However, using $\mathcal{D}_G$ to forecast global motions proved to give a considerable performance boost of 0.92 dB.

Table 2. Ablation experiments

	Adobe240 [40]		GOPRO [27]	
Loss Functions	PSNR	SSIM	PSNR	SSIM
w/o $\mathcal{L}_{\text{M2FNet}}$	25.09	0.730	25.16	0.728
w/o inter-frame motion	25.81	0.776	26.11	0.776
w/o direction supervision	27.13	0.801	27.83	0.806
w/o $\mathcal{L}_{\text{GDL}}$	26.97	0.801	27.84	0.813
M2FNet				
w/o $\mathcal{D}_L$	25.13	0.734	25.71	0.760
w/o $\mathcal{D}_G$	26.96	0.801	27.34	0.811
Frame Decoding				
only warping u_t in Eq. (7)	26.82	0.793	27.41	0.812
excluding v_{t+i} from Eq. (8)	26.03	0.781	26.57	0.789
P-INet	**27.70**	**0.816**	**28.43**	**0.843**

Frame Decoding. We study the importance of incorporating features of past frames when decoding the current frame in PNet. As can be inferred from Table 2, only attending to the *anchor* feature (only warping u_t in Eq. (7)) when synthesizing frames gives a notably lower performance compared to attending all past features. Moreover, not attending to any past feature (excluding v_{t+i} from Eq. (8)) during frame decoding performs significantly worse.

6. Conclusion

Our work introduces a temporally robust VFI framework by adopting a feature propagation approach. The proposed motion supervision tailors the network for the task at hand as it enforces features to be propagated according to the motion between inputs irrespective of their contents. The adaptive cascading of PNet with a simple interpolation backbone has significantly improved the interpolation quality for low frame rate videos as briefly analyzed in Sec. 4.

Limitations. The multi-scale approach along with aggregated motion estimation significantly increases the time complexity of our model. For instance, during 10 fps $\rightarrow$ 240 fps up-conversion given an input pair of size 448×256, SloMo [17] takes 0.32 secs while P-INet takes 3.37 secs. We experimentally observed failure cases when there is a fast-moving *small* object in the foreground of a scene with a relatively large, dynamic background. In this scenario, PNet fails to detect and anticipate the motion of such objects, and instead imitates the input feature during propagation. This results in temporal jittering artifact in the interpolated video. Improving this limitation using a detection module [47] or an attention mechanism [7] would be an interesting future direction.

References

[1] Dawit Mureja Argaw, Junsik Kim, Francois Rameau, and In So Kweon. Motion-blurred video interpolation and extrapolation. In *AAAI Conference on Artificial Intelligence*, 2021. 2

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 6, 7

[3] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 2

[4] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. *arXiv preprint arXiv:2007.11762*, 2020. 2

[5] Hsu-kuang Chiu, Ehsan Adeli, and Juan Carlos Niebles. Segmenting the future. *IEEE Robotics and Automation Letters*, 5(3):4202–4209, 2020. 2

[6] Myungsub Choi, Janghoon Choi, Sungyong Baik, Tae Hyun Kim, and Kyoung Mu Lee. Scene-adaptive video frame interpolation via meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9444–9453, 2020. 2

[7] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *AAAI*, 2020. 2, 8

[8] Camille Couprie, Pauline Luc, and Jakob Verbeek. Joint future semantic and instance segmentation prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 2

[9] Shengyang Dai and Ying Wu. Motion from blur. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 6

[10] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015. 4

[11] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. *arXiv preprint arXiv:1703.05884*, 2017. 2, 5, 6

[12] Shurui Gui, Chaoyue Wang, Qihua Chen, and Dacheng Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14004–14013, 2020. 1, 2, 6, 7

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3

[14] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8981–8989, 2018. 4

[15] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 4, 6, 8

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. 3

[17] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE Conferene on Computer Vision and Pattern Recognition*, 2018. 1, 2, 5, 6, 7, 8

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 5

[19] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 6

[20] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471, 2017. 2

[21] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016. 2

[22] Pauline Luc, Camille Couprie, Yann Lecun, and Jakob Verbeek. Predicting future instance segmentation by forecasting convolutional features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018. 2

[23] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Transactions on Graphics (TOG)*, 28(3):1–11, 2009. 2

[24] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 5, 8

[25] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 498–507, 2018. 2

[26] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1418, 2015. 2

[27] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 5, 6, 8

[28] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1710, 2018. 2

[29] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020. 2

[30] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017. 1, 2, 6

[31] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017. 2

[32] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. *arXiv preprint arXiv:2007.12622*, 2020. 2

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019. 5

[34] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. Im-net for high resolution video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[35] Fitsum A Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised video interpolation using cycle consistency. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 892–900, 2019. 2

[36] Josip Šarić, Marin Oršić, Tonći Antunović, Sacha Vražić, and Siniša Šegvić. Single level feature-to-feature forecasting with deformable convolutions. In *German Conference on Pattern Recognition*, pages 189–202. Springer, 2019. 2

[37] Josip Saric, Marin Orsic, Tonci Antunovic, Sacha Vrazic, and Sinisa Segvic. Warp to the future: Joint forecasting of features and feature motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10648–10657, 2020. 2

[38] Wang Shen, Wenbo Bao, Guangtao Zhai, Li Chen, Xiongkuo Min, and Zhiyong Gao. Blurry video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5114–5123, 2020. 2

[39] Sanghyun Son, Jaerin Lee, Seungjun Nah, Radu Timofte, and Kyoung Mu Lee. Aim 2020 challenge on video temporal super-resolution. *arXiv preprint arXiv:2009.12987*, 2020. 2

[40] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 5, 6, 8

[41] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 4, 8

[42] Jiangxin Sun, Jiafeng Xie, Jian-Fang Hu, Zihang Lin, Jianhuang Lai, Wenjun Zeng, and Wei-shi Zheng. Predicting future instance segmentation with contextual pyramid convlstms. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2043–2051, 2019. 2

[43] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–106, 2016. 2

[44] Suhani Vora, Reza Mahjourian, Soeren Pirk, and Anelia Angelova. Future segmentation using 3d structure. *arXiv preprint arXiv:1811.11358*, 2018. 2

[45] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *NeurIPS*, 2019. 2

[46] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 2, 5

[47] Liangzhe Yuan, Yibo Chen, Hantian Liu, Tao Kong, and Jianbo Shi. Zoom-in-to-check: Boosting video interpolation via instance-level discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12183–12191, 2019. 2, 8

[48] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A flexible recurrent residual pyramid network for video frame interpolation. In *European Conference on Computer Vision*, 2020. 2

[49] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM transactions on graphics (TOG)*, 23(3):600–608, 2004. 2