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 \( \Delta t \) 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 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.

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. [57] 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.
3.1. Propagation Network (PNet)

We use an encoder-decoder architecture for PNet. First, we design an encoder network \( E \) to extract features from the input frames \( \{x_t, x_{t+1}, \ldots, x_{t+n}\} \) 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 \( 3 \times 3 \). 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.

\[
\{u^l_t\}_{l=1}^k = E(x_t) \quad \{u^l_{t+n}\}_{l=1}^k = E(x_{t+n}) \quad (1)
\]

where \( l \) denotes a level in the feature pyramid with a total of \( k \) levels (\( k = 5 \) in our experiments) and \( u^l_t \) 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. \( \{u^1_t, \ldots, u^k_t\} \) and \( \{u^1_{t+n}, \ldots, u^k_{t+n}\} \), 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 (\( D_G \)) and local (\( D_L \)) motion decoders. \( D_G \) 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 \( D_G \).

\[
\{\theta_{R[T+i]}\}_{i=1}^m = D_G(x_t || u^l_{t+n}) \quad (2)
\]

\[
\{\hat{u}^l_{t+i}\}_{i=1}^m = \text{transform}(u^l_t, \{\theta_{R[T+i]}\}_{i=1}^m) \quad (3)
\]

where \( || \) denotes channel-wise concatenation, \( m \) refers to the number of features (frames) propagated from the anchor feature \( u_t \) and \( \hat{u}^l_{t+i} \) represents the output of \( D_G \) at time step \( t+i \) and feature level \( l \). As \( D_G \) is limited to learning only non-local motion, in order to capture the locally varying motion, we further refine the outputs of \( D_G \) with a local motion decoder (\( D_L \)). \( D_L \) inputs the globally transformed feature \( \hat{u}^l_{t+i} \) along with the anchor and reference features, and outputs the forecasted feature \( u^l_{t+i} \) (see Eq. (4)). \( D_L \) has 3 densely connected convolutional layers [13] each with kernel size \( 3 \times 3 \) and stride 1. As the forecasted feature \( u^l_{t+i} \) 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 \( D_L \), as shown in Eq. (4). A deconvolution layer of kernel size \( 4 \times 4 \) and stride size 2 is used to upsample (\( \times 2 \)) features.

\[
\hat{u}^l_{t+i} = D^l_L(\hat{u}^l_{t+i} || u^l_t || u^l_{t+n} || \uparrow_2(u^l_{t+i+1})) \quad (4)
\]
where \( i = \{1, \ldots, m\} \), up. stands for upsampling and \( u^l_{t+i} \) is the forecasted feature at level \( l \). In principle, \( D_L \) can decode both local and global motions. However, explicitly modelling global motions with \( D_L \) 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 \( f_{t+i} \rightarrow x_t \) and \( f_{u_{t+i} \rightarrow u_t} \) 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 \( f_{x_t \rightarrow x_{t+i}} \) 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 \( f_{u_{t+i} \rightarrow u_t} \), we design a feature flow network (FFNet) which inputs two sets of features, i.e. \( \{u^l_{t+i}\}_{l=1}^k \) and \( \{u^l_t\}_{l=1}^k \), 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 perform 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 \( u^l_{t+i} \). The first feature, the cost volume, and the upsampled flow are fed into FFNet to predict a flow as shown in Eq. (6).

\[
\begin{align*}
    u^l_{t+i} &= \text{backwarp}(u^l_t, \text{up.}(\hat{f}_{t+i}^{l+1} \rightarrow t)) \\
    \hat{f}_{t+i}^{l+1} &= \text{FFNet}(u^l_{t+i} \oplus \text{corr.}(u^l_{t+i}, w_{t+i}^l) \oplus \text{up.}(\hat{f}_{t+i}^{l+1} \rightarrow t))
\end{align*}
\]  

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.* \( D_G \) and \( D_L \) 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).

**Feature-to-Frame Decoding.** The forecasted features and optical flows are then used to decode frames. For this purpose, we design a frame decoder \( (D_P) \) which regresses frames from the corresponding forecasted features. When decoding the current frame, \( D_P \) 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 \( (\alpha) \) is a learnable, one-dimensional weight parameter with elements initially set to 1. For better reconstruction of occluded regions in the predicted frames, \( D_P \) 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 \( l \), \( D_P \) inputs the forecasted feature \( (u^l_{t+i}) \), attended past features \( (v^l_{t+i}) \), encoded features \( (u^l_t \text{ and } u^l_{t+i}) \) and \( \times 2 \) upsampled frame predicted from previous level \( l + 1 \) (see Eq. (8)). \( D_P \) is composed of 3 densely connected convolutional layers each with kernel size \( 3 \times 3 \) and stride 1, where the last layer outputs a frame.

\[
\begin{align*}
    u^l_{t+i} &= \sum_{j=t}^{t+i-1} \alpha_j \cdot \text{backwarp}(u^l_j, \hat{f}_{t+i-j}^{l+1}) \\
    \hat{x}_{t+i}^{l+1} &= D_P(u^l_{t+i} \parallel v^l_{t+i} \parallel u^l_t \parallel u^l_{t+i} \parallel \text{up.}(\hat{x}_{t+i}^{l+1}))
\end{align*}
\]

### 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: $(x_{i}, x_{i+n})$ // $m$ is the frame gap
Output: $\tilde{x}_{t+i}$, 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
    $\tilde{x}_{t+i} = \text{INet}(x_t, x_{t+n})$ for all $i$
  else if $\Delta t(n) \leq M$ then // propagate from $x_t$
    $\tilde{x}_{t+i} = \text{PNet}(x_t, x_{t+n})$
  else if $\Delta t(n) < i < n - \Delta t(n)$ then // propagate and interpolate
    $\tilde{x}_{t+\Delta t(n)} = \text{PNet}(x_t, x_{t+n})$
    $\tilde{x}_{t+n-\Delta t(n)} = \text{PNet}(x_{t+n}, x_t)$
    $\tilde{x}_{t+i} = \text{INet}(\tilde{x}_{t+\Delta t(n)}, \tilde{x}_{t+n-\Delta t(n)})$
  else // propagate from $x_{t+n}$
    $\tilde{x}_{t+i} = \text{PNet}(x_{t+n}, x_t)$
end


3.3. Loss Functions

We train our network in an end-to-end manner by jointly propagating and interpolating depending on the $\Delta t$ between in-put and output 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 $\Delta t$ 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 $\min([\frac{n-M}{2}], M)$, 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 $\frac{m(m+1)}{2} + 1$ flows are estimated. We also investigate training our network by selectively optimizing some of the flows (see Sec. 5).

$$L_{\text{M2FNet}} = \sum_{i=1}^{m} \sum_{l=1}^{k} \omega_{l}^{i} \left| f_{i}^{l} - \hat{f}_{i}^{l} \right|_{2}$$

where $\omega_{l}^{i}$ is a flow loss weight coefficient at level $l$. For sharp frame decoding, we train PNet with the multi-scale $\ell_{1}$ photometric loss. We also use gradient difference loss $\text{GDL}$ between the predicted frames and their ground truth to mitigate blurry predictions (see Eq. (10)).

$$L_{\text{PNet}} = \sum_{i=1}^{m} \sum_{l=1}^{k} \omega_{l}^{i} \left| x_{i}^{l} - \hat{x}_{i}^{l} \right|_{1} + \lambda \text{GDL}(x_{i}, \hat{x}_{i})$$

where $\omega_{l}^{i}$ 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).

$$L_{\text{total}} = \lambda_{1} L_{\text{M2FNet}} + \lambda_{2} L_{\text{PNet}} + \lambda_{3} L_{\text{INet}}$$

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 $448 \times 256$ 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 $\beta_{1}$, $\beta_{2}$ and weight decay fixed to 0.9, 0.999 and $4e-4$, respectively. The loss weight coefficients are set
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 \( L_{M2FNet} \) and \( L_{PNet} \). We train \( P\text{-INet} \) 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 \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) to 1. We use a mini-batch size of 4 and randomly crop image patches of size 256 \( \times \) 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 [3] 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 \( P\text{-INet} \), 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, \( P\text{-INet} \) 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 \( P\text{-INet} \). 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adobe240 [40]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GOPRO [27]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NIS [11]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Method</strong></td>
<td>PSNR</td>
<td>SSIM</td>
<td>PSNR</td>
<td>SSIM</td>
<td>PSNR</td>
</tr>
<tr>
<td></td>
<td><strong>30 fps</strong></td>
<td><strong>15 fps</strong></td>
<td><strong>8 fps</strong></td>
<td><strong>30 fps</strong></td>
<td><strong>15 fps</strong></td>
<td><strong>8 fps</strong></td>
</tr>
<tr>
<td>SepConv [30]</td>
<td>29.91</td>
<td>0.915</td>
<td>23.94</td>
<td>0.811</td>
<td>19.88</td>
<td>0.707</td>
</tr>
<tr>
<td>SloMo [17]</td>
<td>30.03</td>
<td>0.917</td>
<td>24.30</td>
<td>0.818</td>
<td>20.17</td>
<td>0.717</td>
</tr>
<tr>
<td>DAIN [3]</td>
<td>30.53</td>
<td>0.924</td>
<td>24.39</td>
<td>0.824</td>
<td>20.21</td>
<td>0.721</td>
</tr>
<tr>
<td>AdaCoF [19]</td>
<td>30.14</td>
<td>0.896</td>
<td>24.11</td>
<td>0.741</td>
<td>20.07</td>
<td>0.567</td>
</tr>
<tr>
<td>FeFlow [12]</td>
<td>30.48</td>
<td>0.902</td>
<td>24.19</td>
<td>0.737</td>
<td>20.04</td>
<td>0.576</td>
</tr>
<tr>
<td>P-INet</td>
<td>30.30</td>
<td>0.920</td>
<td>24.21</td>
<td>0.819</td>
<td>20.12</td>
<td>0.718</td>
</tr>
<tr>
<td></td>
<td>30.30</td>
<td>0.920</td>
<td>27.10</td>
<td>0.890</td>
<td>24.00</td>
<td>0.810</td>
</tr>
</tbody>
</table>

Figure 5. Qualitative comparison of our method and state-of-the-art VFI approaches for inputs with large temporal gap.
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 → very high fps up-sampling. We perform 10 fps → 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 from both sides. However, it can be noticed from Fig. 8 that there is a rapid performance drop for SloMo 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 $D_G$ and $D_L$ 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 $z_1 = \frac{1}{2}(\sum |\hat{f}_{t+i→l}| + \sum |\hat{f}_{t+n-i→t+n}|)$. 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. $D_G$ and $D_L$ 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.
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 → 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 the flow magnitude as a guidance for feature propagation, we forecast the flows without estimating flows and directly regress the optical flow as a guidance for feature propagation, we forecast the flows as a guidance for feature propagation, we forecast the flows as a guidance for feature propagation, we forecast the flows as a guidance for feature propagation.

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

Table 2. Ablation experiments

<table>
<thead>
<tr>
<th>Loss Functions</th>
<th>Adobe240 [40]</th>
<th>GOPRO [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o $\mathcal{L}_{M2FNet}$</td>
<td>25.09 0.730</td>
<td>25.16 0.728</td>
</tr>
<tr>
<td>w/o direction supervision</td>
<td>27.13 0.801</td>
<td>27.83 0.806</td>
</tr>
<tr>
<td>w/o $\mathcal{L}_{GDL}$</td>
<td>26.97 0.801</td>
<td>27.84 0.813</td>
</tr>
<tr>
<td>Frame Decoding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2FNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o $D_L$</td>
<td>25.13 0.734</td>
<td>25.71 0.760</td>
</tr>
<tr>
<td>w/o $D_G$</td>
<td>26.96 0.801</td>
<td>27.34 0.811</td>
</tr>
<tr>
<td>P-INet</td>
<td>27.70 0.816</td>
<td>28.43 0.843</td>
</tr>
</tbody>
</table>

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 the current frame gives a notably lower performance compared to attending all past features. Moreover, not attending to any past feature (excluding $v_{t+1}$ 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 → 240 fps up-conversion given an input pair of size $448 \times 256$, SloMo [17] takes 0.32 secs while P-INet takes 3.37 secs. We experimentally observed failure cases where 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.

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 optical flows.


