Channel Attention Is All You Need for Video Frame Interpolation

Myungsub Choi,[1]* **Heewon Kim,**[1] **Bohyung Han,**[1] **Ning Xu,**[2] **Kyoung Mu Lee**[1]
[1]Computer Vision Lab. & ASRI, Seoul National University, [2]Amazon Go
{cms6539, ghimhw, bhhan, kyoungmu}@snu.ac.kr, ninxu@amazon.com

Abstract

Prevailing video frame interpolation techniques rely heavily on optical flow estimation and require additional model complexity and computational cost; it is also susceptible to error propagation in challenging scenarios with large motion and heavy occlusion. To alleviate the limitation, we propose a simple but effective deep neural network for video frame interpolation, which is end-to-end trainable and is free from a motion estimation network component. Our algorithm employs a special feature reshaping operation, referred to as PixelShuffle, with a channel attention, which replaces the optical flow computation module. The main idea behind the design is to distribute the information in a feature map into multiple channels and extract motion information by attending the channels for pixel-level frame synthesis. The model given by this principle turns out to be effective in the presence of challenging motion and occlusion. We construct a comprehensive evaluation benchmark and demonstrate that the proposed approach achieves outstanding performance compared to the existing models with a component for optical flow computation.

Introduction

Various video processing problems—including video classification (Carreira and Zisserman 2017; Simonyan and Zisserman 2014a; Xie et al. 2018), object detection and segmentation (Cheng et al. 2017; Zhu et al. 2018; 2017), video prediction (Li et al. 2018; Reda et al. 2018), and others (Chen et al. 2017; Lai et al. 2018)—frequently depend on the accurate optical flow estimation since it leads to the success of the target tasks in terms of quantitative and qualitative performance. Although there exists another line of research that is free from optical flow estimation and predicts outputs directly without motion information (Long et al. 2016; Mathieu, Couprie, and LeCun 2016a; Ranzato et al. 2014; Srivastava, Mansimov, and Salakhudinov 2015), these methods have failed to present promising results compared with the techniques based on optical flow estimation while showing potential as generic methodologies.

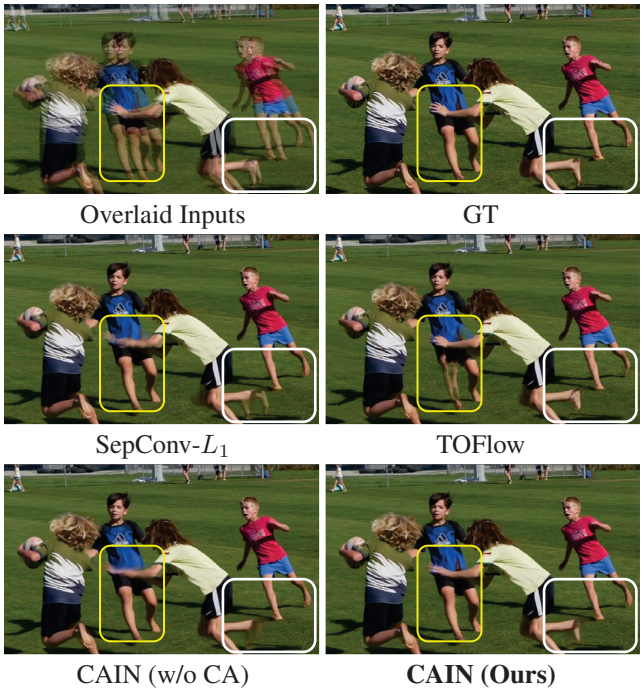


Figure 1: Advantage of using Channel Attention (CA) for video frame interpolation in an example with large motion. The proposed algorithm is also compared with the existing methods including SepConv-L_1 (Niklaus, Mai, and Liu 2017b) and TOFlow (Xue et al. 2018).

Video frame interpolation approaches (Jiang et al. 2018; Liu et al. 2017; Niklaus and Liu 2018; Xue et al. 2018) also follow this trend and typically work as follows: 1) estimating optical flow between two consecutive frames, 2) constructing initial estimates of intermediate frames by image warping with the optical flow, and 3) refining the initial interpolation results through high-level processing using a deep neural network. While this pipeline often works well in practice, it relies heavily on optical flow estimation that incurs substantial computational cost, in terms of time and memory, for motion estimation. Moreover, while the approach requires

*This work was partially done during the internship at Snap Research.

a large number of ground-truth annotations to learn optical flow, the supervision for motion estimation is tricky because real ground-truth motion is not available; researchers often use synthetic image pairs with known motion fields but they cannot model the real challenges observed in natural scenes.

We are motivated by the potential drawbacks in using optical flow, and propose a novel framework of video frame interpolation that replaces optical flow with simple feature map transformations, referred to as PixelShuffle (Shi et al. 2016). PixelShuffle gradually distributes the information related motion into multiple channels and constructs a transformed feature map, which is combined with a channel attention to capture the variations between the anchor frames including motion and synthesizes high-quality intermediate video frames without explicit motion estimation. The proposed approach is capable of handling large motion and heavy occlusion effectively, and outperforms the prior state-of-the-art methods.

Overall, our contributions are summarized as follows:

- We propose a novel end-to-end trainable network of video frame interpolation that synthesizes an intermediate frame effectively without the explicit estimation of motion.
- We empirically show that the motion estimation module can be replaced by a simple combination of PixelShuffle, parameter-free feature transformations, and channel attention successfully for video frame interpolation.
- We construct a more comprehensive benchmark for video frame interpolation and confirm that our model outperforms the existing methods by large margins.

The rest of the paper is organized as follows. We first review the related works and then describe our network architecture. After that, we discuss the reasons that our model can alleviate the need for explicit motion estimation by visualizing and analyzing each feature channels. Last, we present experimental results on a new benchmark dataset and compare our method with the existing ones.

Related Works

Video frame interpolation

Video frame interpolation approaches are typically based on optical flow estimation to handle time-varying information in videos (Baker et al. 2010; Werlberger et al. 2011; Yu et al. 2013). The standard pipeline is given by linear interpolation of optical flow, warping of input frames using the interpolated motion, and frame blending with occlusion reasoning and missing region completion. Recent advances in deep convolutional neural networks (CNNs) have enabled this process to be trained in an end-to-end manner, and many video frame interpolation techniques incorporate a CNN-based optical flow module into their core networks (Bao et al. 2018; Jiang et al. 2018; Liu et al. 2017; Niklaus and Liu 2018; Xue et al. 2018). However, the accurate estimation of optical flow is a challenging problem that is error-prone to many cases in real-world videos, especially when there are large motion and heavy occlusion. The errors in optical flow estimation may be propagated to the subsequent procedure for video frame interpolation and degrade the overall performance of algorithms.

On the other hand, several recent frame interpolation methods generate unseen frames without predicting explicit flow vectors. These approaches include phase-based and kernel-based methods while there exists a direct pixel-level synthesis technique (Long et al. 2016). The phase-based interpolation methods (Meyer et al. 2015; 2018) represent motion with the phase shift of individual pixels and construct intermediate frames using modified per-pixel phase. (Niklaus, Mai, and Liu 2017a) performs video frame interpolation using spatially-adaptive convolution filters whose weights are predicted by a CNN. The computational complexity of this idea has been greatly improved in (Niklaus, Mai, and Liu 2017b) by making the convolution kernels separable, but the amount of motion this approach can handle is still limited to the predefined kernel size. An earlier work by (Long et al. 2016) synthesizes video frames by interpolation to learn a CNN model for unsupervised optical flow estimation. Although the frames generated by the method look blurry and their quantitative results are not competitive, the approach has shown the potential of pixel-level synthesis without the direct supervision of motion.

While our algorithm is conceptually related to (Long et al. 2016), it introduces a new combination of simple modules to build an architecture appropriate for motion understanding and leads to significant performance boost in synthesizing intermediate video frames in challenging scenarios.

Attention mechanism

Attention mechanism makes a neural network focus on important regions of its feature representations. There is a wide range of applications taking advantage of attention in deep neural networks, which includes sequence-based models (Bahdanau, Cho, and Bengio 2014; Mnih et al. 2014; Vaswani et al. 2017), image classification (Hu, Shen, and Sun 2018; Wang et al. 2017; Woo et al. 2018), image localization (Cao et al. 2015; Jaderberg et al. 2015), and image super-resolution (Zhang et al. 2018b).

To maximize the benefit of attention, several recent works propose specialized architectures to effectively implement this feature. (Wang et al. 2017) proposes residual attention network, a powerful encoder-decoder style model, for image classification. (Hu, Shen, and Sun 2018) proposes squeeze-and-excitation (SE) module to focus on calculating inter-channel relationships, and improve classification accuracy with a more compact module. This approach has been extended in (Woo et al. 2018), which introduces an efficient combination of spatial and channel attention. (Zhang et al. 2018b) applies channel attention to a low-level vision problem of single image super-resolution. Since every spatial region is important for pixel-level synthesis, we do not explicitly consider spatial attention. Instead, we adopt the channel attention method proposed in (Zhang et al. 2018b) to our video frame interpolation framework.

Video Frame Interpolation Model

The goal of video frame interpolation is to synthesize a correct intermediate frame given two input video frames $\mathbf{I}_1$

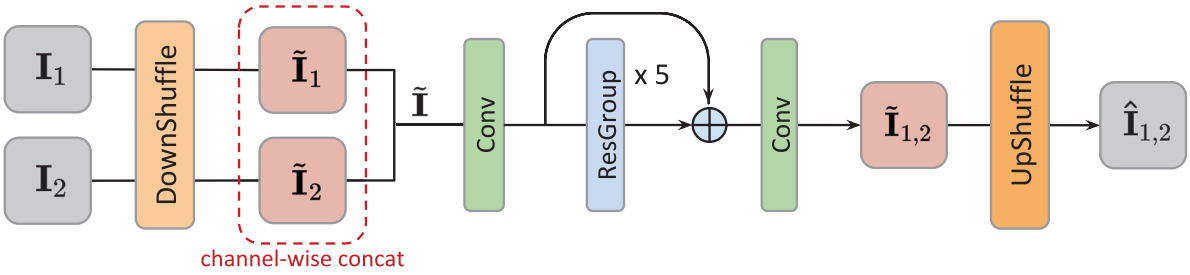


Figure 2: The overall architecture of our network. The two down-shuffled images $\tilde{\mathbf{I}}_1$ and $\tilde{\mathbf{I}}_2$ are concatenated in a channel direction to build $\tilde{\mathbf{I}}$. The channel dimension of $\tilde{\mathbf{I}}$ is reduced by the first convolution layer to match that of $\tilde{\mathbf{I}}_{1,2}$. A global identity connection is added around the 5 ResGroups. After the identity feature maps is combined with the residuals, we perform another convolution followed by up-shuffling operations to obtain the final output $\hat{\mathbf{I}}_{1,2}$.

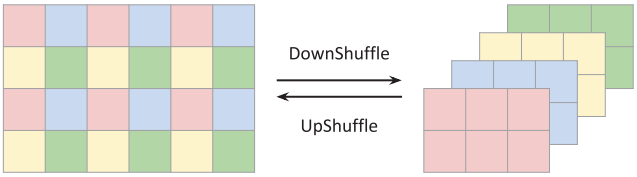
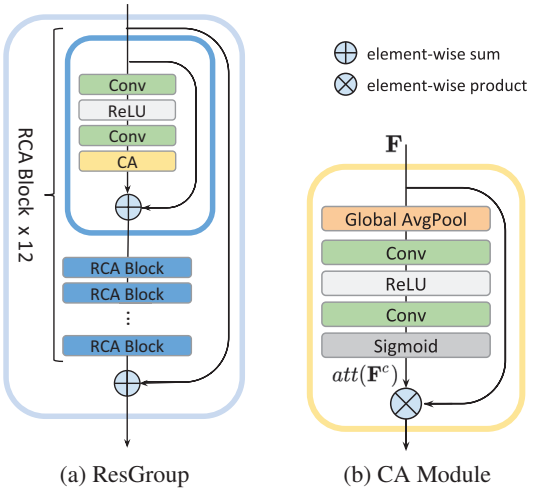


Figure 3: Visualization of the operations in PixelShuffle. Note that, contrary to the original version in (Shi et al. 2016), our algorithm does not apply convolutions after the shuffling operations.

and $\mathbf{I}_2$. The success of video frame interpolation algorithms obviously depends on the quality of high-level reasoning about motion and occlusion across the two frames. While most existing methods estimate bidirectional optical flow or its variants for motion estimation, we design a deep neural network architecture for accurate video frame interpolation through a simple but effective image representation via PixelShuffle (Shi et al. 2016) and an inter-frame low-level event modeling by channel attention (Hu, Shen, and Sun 2018; Zhang et al. 2018b). The proposed method allows to be free from learning or estimating optical flow during our training and testing procedure.

Overview of network architecture

Figure 2 illustrates the overall network architecture of the proposed model. Our network down-shuffles two input images, $\{\mathbf{I}_1, \mathbf{I}_2\}$ into $\{\tilde{\mathbf{I}}_1, \tilde{\mathbf{I}}_2\}$, respectively, which are concatenated in a channel direction to construct a combined image, $\tilde{\mathbf{I}}$. The transformed input $\tilde{\mathbf{I}}$ is passed through a series of residual blocks with channel attentions to synthesize the down-shuffled target image denoted by $\tilde{\mathbf{I}}_{1,2}$, which is up-shuffled to construct the final intermediate frame, $\hat{\mathbf{I}}_{1,2}$. Note that the procedure of down-shuffling followed by up-shuffling resembles encoder-decoder models, but does not involve any feature learning and introduce any parameter. We describe the details of the individual components next.



(a) ResGroup (b) CA Module

Figure 4: Illustration of ResGroup and CA module.

Main components

We now discuss two main components of our network, PixelShuffle and residual groups with channel attention.

PixelShuffle PixelShuffle (Shi et al. 2016) is the operation to reorganize the layout of an image (or a feature map) by pooling an image with several different switch variables and generating down-sampled images corresponding to the individual switch variables or performing its inverse procedure. The former is referred to as down-shuffling and the latter is called up-shuffling. The down-shuffle operation reduces the spatial dimension of an image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ by a factor of s in each down-shuffle layer, and results in a tensor in $\mathbb{R}^{H/s \times W/s \times s^2 C}$ while the up-shuffle performs the operation in the opposite direction. Figure 3 visualizes both operations when $s = 2$.

We apply PixelShuffle operations with $s = 8$, and generate two down-shuffled images by a factor of 8, denoted by $\tilde{\mathbf{I}}_1, \tilde{\mathbf{I}}_2 \in \mathbb{R}^{H/8 \times W/8 \times 192}$, from the corresponding input images, $\mathbf{I}_1, \mathbf{I}_2 \in \mathbb{R}^{H \times W \times 3}$. After that, $\tilde{\mathbf{I}}_1$ and $\tilde{\mathbf{I}}_2$ are concatenated in a channel direction and the resulting image has 384

Figure 5: Visualization of internal feature maps with their channel attentions. The leftmost images show our prediction results $\hat{\mathbf{I}}_{1,2}$ while the rightmost ones illustrate the reference optical flow map $\mathbf{M}_{1\to2}$ between two input images $\mathbf{I}_1$ and $\mathbf{I}_2$, calculated with PWC-Net (Sun et al. 2018). In the middle, the activation maps for the channels that have the highest attention scores are shown for each example. The regions in red have high activations while blue indicates low ones, and the heatmaps are super-imposed on the ground-truth intermediate image. We can clearly see that most regions with high feature responses in the feature maps with high channel attention scores correspond to the regions with large motion. It implies that channel attention is playing its role appropriately by focusing on these regions with high weights.

channels. We reduce the channel dimension back to 192 by applying a 3×3 convolution. At the end of the network, we reconstruct the image to its original size through a series of the inverse processing, up-shuffling.

The main reason to employ PixelShuffle instead of the standard encoder-decoder models such as U-Net (Ronneberger, Fischer, and Brox 2015) is to maintain a large receptive field size in the following convolution blocks and enable to handle various scene changes through the combinations of multiple channels with shifted features. Note that PixelShuffle is parameter-free, and it does not lose any local information when increasing the receptive field. Any kind of encoder and decoder can be integrated additionally to improve performance with extra cost, but it turns out that PixelShuffle is powerful enough to achieve competitive performance without a complex encoder-decoder pair.

Residual groups with channel attentions The down-shuffled feature maps are followed by five residual groups, referred to as *ResGroups* from now, each of which consists of 12 residual channel attention (RCA) blocks, having 60 RCA blocks in total. As illustrated in Figure 4a, each RCA block in ResGroup contains two 3×3 convolution layers with the rectified linear unit (ReLU) activation in between and a channel attention (CA) module is located before the residual connection. The architecture for the CA module is shown in Figure 4b. Denoting the input feature by $\mathbf{F} \in \mathbb{R}^{H' \times W' \times C'}$, global average pooling aggregates the statistics of a channel to obtain a descriptor $\mathbf{F}^c \in \mathbb{R}^{1 \times 1 \times C'}$. The following two 1×1 convolution layers, which are identical to fully-connected layers, are expected to capture the non-linear inter-channel relationships.

Formally, the channel attention weights are computed as:

$$\text{att}\left(\mathbf{F}^c\right) = \sigma\left(\mathbf{W}_1 * \left(\text{ReLU}\left(\mathbf{W}_0 * \mathbf{F}^c\right)\right)\right), \tag{1}$$

where $\sigma(\cdot)$ denotes a sigmoid function, and $\mathbf{W}_0$ and $\mathbf{W}_1$ are weights of the two 1×1 convolution layers. The final output of our CA module is then calculated as an element-wise product of the input feature $\mathbf{F}$ and the obtained attention weights $\text{att}(\mathbf{F}^c)$.

Loss

We train our base network illustrated in Figure 2 using a pixel reconstruction loss $\mathcal{L}_r$ that measures how close the reconstructed intermediate frame $\hat{\mathbf{I}}_{1,2}$ is to the ground-truth frame $\mathbf{I}_{\text{gt}}$ in RGB pixel space. The reconstruction loss is given by the ℓ_1 norm as

$$\mathcal{L}_r = \left\| \hat{\mathbf{I}}_{1,2} - \mathbf{I}_{\text{gt}} \right\|_1. \tag{2}$$

The loss based on ℓ_1 norm is known to produce less blurry results than the standard ℓ_2 loss (Goroshin, Mathieu, and LeCun 2015; Long et al. 2016; Mathieu, Couprie, and LeCun 2016b).

For better qualitative results, after convergence with $\mathcal{L}_r$, we fine-tune our model using another loss function with a perceptual loss term, $\mathcal{L}_p$, which is given by

$$\mathcal{L} = \lambda_1 \mathcal{L}_r + \lambda_2 \mathcal{L}_p, \tag{3}$$

where we use $\lambda_1 = 0.9$ and $\lambda_2 = 0.005$ to match the scale of both loss types. The perceptual loss is computed by

$$\mathcal{L}_p = \left\| \phi(\hat{\mathbf{I}}_{1,2}) - \phi(\mathbf{I}_{\text{gt}}) \right\|_2^2, \tag{4}$$

where $\phi(\cdot)$ is a function to extract conv5_4 features from the VGG-19 model pretrained on ImageNet dataset (Simonyan and Zisserman 2014b). As noted in (Zhang et al. 2018a), the perceptual loss greatly helps in synthesizing realistic frames.

Understanding Our Model

We investigate the benefit of channel attention in handling motion when combined with PixelShuffle. Note that, in our framework, the down-shuffling operation distributes spatial context to the channel axis, which makes it straightforward to model scene changes by attending to individual channels appropriately. Regarding this, we visualize the internal feature representations with channel attentions and their reconstruction during the interpolation process to gain more insight.

Internal Feature Visualization

We describe the relationships between feature activations of our fully trained model and the actual motion map obtained from an optical flow computation network. Figure 5 visualizes multiple activation maps in the channels that have the highest attention weights, where our final prediction results are presented along with the corresponding reference optical flow estimations. Note that the regions with high responses for each visualized feature map roughly match the moving regions or its subset. This shows how our channel attention module successfully identifies the right channels containing important information.

Reasoning about the correct motion is the most critical step in video frame interpolation. However, one caveat of using a motion estimation module is that even the exact optical flow does not necessarily guarantee an optimal interpolation, as discussed in (Xue et al. 2018). This is the main reason that we attempt to perform video frame interpolation without explicit motion estimation and jointly consider all the information needed to synthesize high-quality frames (including *e.g.* occlusion, fine details around the motion boundaries, brightness, color, etc.) as a whole. Therefore, our feature activations visualized in Figure 5 should also contain all kinds of information, which is why the visualizations may look a bit noisy compared to the flow map. Another reason for the misalignment with the motion map is that our model aims at generating a good interpolation by iteratively refining the motion through multiple ResBlocks. In other words, our approach estimates a latent motion field in multiple channels within a layer and handles the full motion along multiple layers (ResBlocks) in a progressive fashion. Figure 6 presents evidence that our model gradually improves motion estimation.

Intermediate image reconstruction

We follow the feedforward process of our network and visualize the interpolated frame from the intermediate feature maps in the layers of 4 ResGroups, from $G2$ to $G5$[1] in Figure 6. Each image is reconstructed by passing the first feature map in each ResGroup through the last convolution layer followed by up-shuffling layers. The results clearly illustrate that, while the reconstruction from an early layer (*e.g.,* $G2$) almost looks like an overlay of two input frames, the motion between two frames is gradually compensated to synthesize the interpolated frames accurately. The last row presents our final results from the full model.

[1]We omit the first ResGroup since the most of its activations resemble the low-level feature responses that are commonly observed in CNNs.

Figure 6: Visualization of the reconstructed images from the output feature representations of individual ResGroups. They clearly show that moving regions gradually merge together as the features pass through more layers. From top to bottom, we present the results from the 2nd to 5th ResGroups and the last row illustrates the final output.

Experiments

Datasets

We evaluate our model on three benchmark datasets commonly used in the recent works (Jiang et al. 2018; Liu et al. 2017; Niklaus and Liu 2018; Niklaus, Mai, and Liu 2017b; Xue et al. 2018): Middlebury optical flow (Baker et al. 2010), UCF101 (Soomro, Zamir, and Shah 2012), and Vimeo90K (Xue et al. 2018).

In addition to these three datasets, we create a more comprehensive benchmark dataset called SNU-FILM (SNU Frame Interpolation with Large Motion), to evaluate the performance of video frame interpolation methods with respect to the amount of motion; from almost static scenes to very challenging scenarios containing large motion and occlusion. Note that it is not appropriate to use the existing datasets comprising of 30 fps videos because controlling the amount of motion by dropping multiple frames in a 30 fps video results in a very large time gap between remaining frames. So, we construct our benchmark dataset based on high frame rate videos including 11 videos from the test set of GOPRO dataset (Nah, Kim, and Lee 2017) and 20 videos collected from YouTube. All the videos are of 240 fps and we evaluate on 10 different frames per sequence. We plan to release the dataset with the motion attribute annotations.

Our evaluation benchmark has four different settings—*Easy, Medium, Hard*, and *Extreme*—depending on the temporal gap between two input frames. For the *Easy* setting, we

Figure 7: Sample image and its flow maps from each of the test settings of the proposed SNU-FILM benchmark. Note the clear difference of the amount of motion for each setting. Optical flow is calculated with PWC-Net (Sun et al. 2018), and the maximum value of its magnitude is set to 20 for better visualization.

Table 1: Motion (flow magnitude) statistics for each setting.

Setting	Easy	Medium	Hard	Extreme
Mean	2.71	5.39	10.65	20.51
Maximum	15.71	31.38	62.26	120.23

drop every other frame from the original 240 fps videos and use the dropped frame as the target ground-truth interpolation, which formulates the video frame interpolation task from 120 fps to 240 fps. In the *Medium* setting, every fourth frame remains and the middle frame among the three dropped ones is employed as the target. which define a problem to double the frame rate from 60 to 120. Likewise, the *Hard* and the *Extreme* are for 30-to-60 fps and 15-to-30 fps, respectively.

Although the categorization in our synthetic dataset does not exactly correspond to the actual motion magnitude between two frames, they are highly correlated and the dataset is representative to evaluate video frame interpolation algorithms with respect to the size of motion. Table 1 presents the motion statistics for each setting, and Figure 7 illustrates the examples in our test set with different motion attributes. Note that, since we do not have the ground-truth optical flow for each scene, we use PWC-Net (Sun et al. 2018) for all optical flow calculation and use the results as ground-truths.

Implementation details

We use the training split of Vimeo90K (Xue et al. 2018) dataset for training, where our model is optimized by Adam (Kingma and Ba 2014) for 200 epochs (approximately 320K iterations); training is based on 256×256 patches and the batch size is 32. Random vertical and horizontal flipping along with random temporal order swapping between two input frames are adopted for data augmentation. The initial learning rate is 0.0001, which is reduced by a factor of 2 whenever the validation loss stops decreasing for more than 5 epochs. We clip the gradient norm to be less than 0.1, which handles the gradient explosion issue.

Our algorithm is implemented in PyTorch. A full training of our network takes about 4 days on a single Titan Xp GPU. The source code for our framework is made public along with the pretrained models to facilitate reproduction.[2]

[2]https://github.com/myungsub/CAIN

Comparison with the state-of-the-art

We compare our framework, named CAIN (Channel Attention for frame INterpolation), with state-of-the-art end-to-end models including deep voxel flow (DVF) (Liu et al. 2017), adaptive separable convolution (SepConv) (Niklaus, Mai, and Liu 2017b), task-oriented flow (TOFlow) (Xue et al. 2018), SuperSloMo (Jiang et al. 2018), CyclicGen (Liu et al. 2019), and DAIN (Bao et al. 2019). We also use recent PWC-Net (Sun et al. 2018) to compute bi-directional optical flow and use an occlusion reasoning algorithm (Baker et al. 2010) to construct a strong baseline for two-step frame interpolation algorithm for comparison.

For quantitative evaluation, we measure peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) index. Overall performance is summarized in Table 2. We use the official pretrained models provided by the authors if they are available. Note that, although our network is only trained on Vimeo90K dataset, it can still generalize well to other datasets and show outstanding performance. We observe significant gain in terms of both PSNR and SSIM on all settings of the new motion-focused evaluation benchmark, which indicates that our model handles a wide range of motion magnitude effectively. This is an exciting result, since many cases in *Hard* or *Extreme* setting have larger motion than that is present in Vimeo90K dataset. This demonstrates that our model, while synthesizing the pixel values directly without motion estimation, generalizes to the challenging scenarios with large motions better than the existing methods with explicit motion computation. While our model shows relatively higher interpolation error (IE) in the Middlebury benchmark, it is qualitatively much better in most cases.

Qualitative results are shown in Figure 8, where we compare with four recent methods that have their pretrained models available. In both examples, all compared methods failed to find the correct position of the moving object that leads to severe ghost artifacts. Our method, on the other hand, is not only good at finding the correct position of the objects even in the presence of large motion, but also has much less ghost artifacts due to better occlusion handling. The resulting interpolations are thus visually much more pleasing. For more qualitative comparison and full video demos, please also take a look at our supplementary document.

We analyze the computational complexity of our model in comparison with two recent state-of-the-art approaches, CyclicGen (Liu et al. 2019) and DAIN (Bao et al. 2019). For this experiment, we use HD resolution frame triplets

Table 2: Quantitative results (PSNR / SSIM) of different frame interpolation algorithms on the proposed SNU-FILM benchmark, Vimeo90K (Xue et al. 2018), and UCF101 (Soomro, Zamir, and Shah 2012; Liu et al. 2017) datasets. Note that SNU-FILM consists of 4 splits with respect to the amount of motion. Results for Middlebury dataset (Baker et al. 2010) show interpolation error (IE, the lower the better). **Bold** denotes the best performance.

Model	FILM(Easy)	FILM(Medium)	FILM(Hard)	FILM(Extreme)	Vimeo90K	UCF101	Middlebury
PWC-Net	36.42 / 0.983	33.09 / 0.960	27.72 / 0.888	23.81 / 0.806	31.36 / 0.939	33.60 / 0.963	2.24
DVF	25.10 / 0.848	23.31 / 0.809	21.68 / 0.768	19.86 / 0.720	31.54 / 0.946	34.12 / 0.963	4.04
SepConv-L_1	39.68 / **0.990**	35.07 / 0.976	29.39 / 0.926	24.32 / 0.845	33.79 / 0.970	**34.95** / 0.968	**2.05**
TOFlow+Mask	39.08 / 0.989	34.39 / 0.974	28.44 / 0.918	23.39 / 0.831	33.73 / 0.968	34.58 / 0.967	2.15
SuperSloMo	†37.28 / 0.986	†33.80 / 0.973	†28.98 / 0.925	†24.15 / 0.845	†33.15 / 0.966	34.75 / 0.967	†2.28
CAIN (w/o CA)	39.59 / **0.990**	35.34 / 0.976	29.56 / 0.926	24.48 / 0.846	34.25 / 0.970	34.75 / 0.968	2.36
CAIN (Ours)	**39.78 / 0.990**	**35.49 / 0.977**	**29.86 / 0.929**	**24.69 / 0.850**	**34.65 / 0.973**	34.91 / **0.969**	2.28

†: results from publicly available implementation of (Paliwal 2018).

Figure 8: Qualitative comparisons on Vimeo90K dataset (Xue et al. 2018) with recent frame interpolation algorithms. While the moving regions are not properly handled and sometimes disappear for other approaches, our method successfully finds the correct interpolated positions.

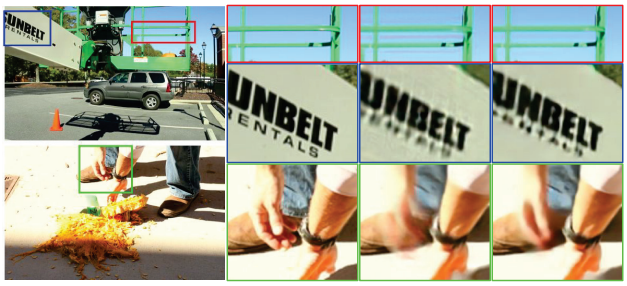
Table 3: Computational complexity for running HD resolution (1280 × 720) frames.

Model	CyclicGen	DAIN	CAIN (Ours)
# Parameters (M)	**3.05**	24.0	42.8
Run-time (ms)	1036	816	**64**
Memory (MB)	4025	6944	**492**
Vimeo90k	32.10 / 0.949	**34.72 / 0.976**	34.65 / 0.973
UCF101	**35.11** / 0.968	35.00 / 0.968	34.91 / **0.969**

with batch size of 1, and measure the peak GPU memory usage and the average running time for 100 batches. Table 3 summarizes time and memory complexity of each model. The number of parameters in our model is larger than other algorithms, but the proposed model has a clear advantage with respect to run-time and memory requirements. While the accuracy of CAIN is marginally worse than DAIN, it is faster by more than one order of magnitude with only about 7% of memory usage. Compared to CyclicGen, CAIN achieves competitive accuracy in the tested datasets, Vimeo90k and UCF101, with more than 16 times faster speed and 12.2% memory.

Ablation study

Comparison to the model without channel attention Figure 9 and Table 2 presents the qualitative and quantitative comparisons, respectively, of the models with and without channel attention. For the model without channel attention (w/o CA), we simply remove the CA module in Figure 4a to change the base building block from RCA blocks to a widely-used plain residual block (ResBlock) (He et al. 2016). The results show that integrating channel attention gives a significant PSNR gain of 0.4 dB in Vimeo90k dataset, and also show consistent boosts in PSNR and SSIM in our SNU-FILM benchmark across all motion groups. The proposed model with CA generates visually more pleasing frames with less artifacts; it demonstrates that channel attention plays a crucial role in handling motion and improving accuracy. Please refer to the supplementary materials for more qualitative results.



GT w/o CA w/ CA

Figure 9: Visual comparison for our models with and without channel attention. We illustrate the cropped regions for each image from Vimeo90K dataset (Xue et al. 2018). Our model with CA captures moving objects better and shows images with less motion blur.

Table 4: Quantitative results from several models in which the numbers of ResGroups and RCA blocks per ResGroup vary while their product is set to a constant. We show the performance for Vimeo90K dataset (Xue et al. 2018).

#ResGroups	#RCA Blocks	PSNR	SSIM
3	20	34.55	0.972
5	12	**34.65**	**0.973**
12	5	34.03	0.968

Effects of hyperparameters We present the influence of the number of ResGroups and RCA blocks using Vimeo90K dataset in Table 4. For fair comparisons, we fix the number of total RCA blocks in our model and modify the ratio between the number of groups and blocks, *i.e.*, (the number of groups) $\times$ (the number of blocks) $= 60$. All compared models therefore has the same number of parameters and running time. Our final model with 5 ResGroups outperforms the other model variants, but the differences in performance are not that significant when the number of RCA blocks becomes large. We also observe that as the number of groups increases a lot, training becomes more sensitive and does not converge at times.

Conclusion

We presented a novel framework for video frame interpolation that synthesizes high-quality images without explicit estimation of motion. The proposed model incorporates channel attention with PixelShuffle, and generalizes to unseen motions effectively. To effectively present the results, we build a new benchmark focused on evaluating video frame interpolation algorithms in the presence of the variations in motion magnitude. Both quantitative and qualitative results demonstrate the advantage of our approach over the existing state-of-the-art methods that are based on optical flow.

Acknowledgements This work was partially supported by National Research Foundation of Korea (NRF), and the Ministry of Science and ICT of Korea under MSIT/IITP grants [2017R1A2B2011862, 2017-0-01780].

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Baker, S.; Scharstein, D.; Lewis, J. P.; Roth, S.; Black, M. J.; and Szeliski, R. 2010. A database and evaluation methodology for optical flow. *IJCV* 92(1):1–31.

Bao, W.; Lai, W.-S.; Zhang, X.; Gao, Z.; and Yang, M.-H. 2018. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *arXiv preprint arXiv:1810.08768*.

Bao, W.; Lai, W.-S.; Ma, C.; Zhang, X.; Gao, Z.; and Yang, M.-H. 2019. Depth-aware video frame interpolation. In *CVPR*.

Cao, C.; Liu, X.; Yang, Y.; Yu, Y.; Wang, J.; Wang, Z.; Huang, Y.; Wang, L.; Huang, C.; Xu, W.; et al. 2015. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *ICCV*.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*.

Chen, D.; Liao, J.; Yuan, L.; Yu, N.; and Hua, G. 2017. Coherent online video style transfer. In *ICCV*.

Cheng, J.; Tsai, Y.-H.; Wang, S.; and Yang, M.-H. 2017. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*.

Goroshin, R.; Mathieu, M. F.; and LeCun, Y. 2015. Learning to linearize under uncertainty. In *NIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *CVPR*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *NIPS*.

Jiang, H.; Sun, D.; Jampani, V.; Yang, M.-H.; Learned-Miller, E.; and Kautz, J. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Lai, W.-S.; Huang, J.-B.; Wang, O.; Shechtman, E.; Yumer, E.; and Yang, M.-H. 2018. Learning blind video temporal consistency. In *ECCV*.

Li, Y.; Fang, C.; Yang, J.; Wang, Z.; Lu, X.; and Yang, M.-H. 2018. Flow-grounded spatial-temporal video prediction from still images. In *ECCV*.

Liu, Z.; Yeh, R. A.; Tang, X.; Liu, Y.; and Agarwala, A. 2017. Video frame synthesis using deep voxel flow. In *ICCV*.

Liu, Y.-L.; Liao, Y.-T.; Lin, Y.-Y.; and Chuang, Y.-Y. 2019. Deep video frame interpolation using cyclic frame generation. In *AAAI*.

Long, G.; Kneip, L.; Alvarez, J. M.; Li, H.; Zhang, X.; and Yu, Q. 2016. Learning image matching by simply watching video. In *ECCV*.

Mathieu, M.; Couprie, C.; and LeCun, Y. 2016a. Deep multi-scale video prediction beyond mean square error. In *ICLR*.

Mathieu, M.; Couprie, C.; and LeCun, Y. 2016b. Deep multi-scale video prediction beyond mean square error. In *ICLR*.

Meyer, S.; Wang, O.; Zimmer, H.; Grosse, M.; and Sorkine-Hornung, A. 2015. Phase-based frame interpolation for video. In *CVPR*.

Meyer, S.; Djelouah, A.; McWilliams, B.; Sorkine-Hornung, A.; Gross, M.; and Schroers, C. 2018. Phasenet for video frame interpolation. In *CVPR*.

Mnih, V.; Heess, N.; Graves, A.; et al. 2014. Recurrent models of visual attention. In *NIPS*.

Nah, S.; Kim, T. H.; and Lee, K. M. 2017. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*.

Niklaus, S., and Liu, F. 2018. Context-aware synthesis for video frame interpolation. In *CVPR*.

Niklaus, S.; Mai, L.; and Liu, F. 2017a. Video frame interpolation via adaptive convolution. In *CVPR*.

Niklaus, S.; Mai, L.; and Liu, F. 2017b. Video frame interpolation via adaptive separable convolution. In *ICCV*.

Paliwal, A. 2018. Pytorch implementation of super slomo. https://github.com/avinashpaliwal/Super-SloMo.

Ranzato, M.; Szlam, A.; Bruna, J.; Mathieu, M.; Collobert, R.; and Chopra, S. 2014. Video (language) modeling: a baseline for generative models of natural videos.

Reda, F. A.; Liu, G.; Shih, K. J.; Kirby, R.; Barker, J.; Tarjan, D.; Tao, A.; and Catanzaro, B. 2018. Sdc-net: Video prediction using spatially-displaced convolution. In *ECCV*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*.

Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A. P.; Bishop, R.; Rueckert, D.; and Wang, Z. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*.

Simonyan, K., and Zisserman, A. 2014a. Two-stream convolutional networks for action recognition in videos. In *NIPS*.

Simonyan, K., and Zisserman, A. 2014b. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*.

Srivastava, N.; Mansimov, E.; and Salakhudinov, R. 2015. Unsupervised learning of video representations using lstms. In *ICML*.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; and Tang, X. 2017. Residual attention network for image classification. In *CVPR*.

Werlberger, M.; Pock, T.; Unger, M.; and Bischof, H. 2011. Optical flow guided tv-l1 video interpolation and restoration. In *EMMCVPR*.

Woo, S.; Park, J.; Lee, J.-Y.; and Kweon, I.-S. 2018. Cbam: Convolutional block attention module. In *ECCV*.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*.

Xue, T.; Chen, B.; Wu, J.; Wei, D.; and Freeman, W. T. 2018. Video enhancement with task-oriented flow. In *CVPR*.

Yu, Z.; Li, H.; Wang, Z.; Hu, Z.; and Chen, C. W. 2013. Multi-level video frame interpolation: Exploiting the interaction among different levels. *IEEE Transactions on Circuits and Systems for Video Technology* 23(7):1235–1248.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018a. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*.

Zhang, Y.; Li, K.; Li, K.; Wang, L.; Zhong, B.; and Fu, Y. 2018b. Image super-resolution using very deep residual channel attention networks. In *ECCV*.

Zhu, X.; Wang, Y.; Dai, J.; Yuan, L.; and Wei, Y. 2017. Flow-guided feature aggregation for video object detection. In *ICCV*.

Zhu, X.; Dai, J.; Yuan, L.; and Wei, Y. 2018. Towards high performance video object detection. In *CVPR*.