CDFI: Compression-Driven Network Design for Frame Interpolation

Tianyu Ding[1*] Luming Liang[2*†] Zhihui Zhu[3] Ilya Zharkov[2]

[1]Johns Hopkins University [2]Microsoft [3]University of Denver

tding1@jhu.edu, {lulian,zharkov}@microsoft.com, zhihui.zhu@du.edu

Abstract

DNN-based frame interpolation—that generates the intermediate frames given two consecutive frames—typically relies on heavy model architectures with a huge number of features, preventing them from being deployed on systems with limited resources, e.g.*, mobile devices. We propose a compression-driven network design for frame interpolation (CDFI), that leverages model pruning through sparsity-inducing optimization to significantly reduce the model size while achieving superior performance. Concretely, we first compress the recently proposed AdaCoF model and show that a 10× compressed AdaCoF performs similarly as its original counterpart; then we further improve this compressed model by introducing a multi-resolution warping module, which boosts visual consistencies with multi-level details. As a consequence, we achieve a significant performance gain with only a quarter in size compared with the original AdaCoF. Moreover, our model performs favorably against other state-of-the-arts in a broad range of datasets. Finally, the proposed compression-driven framework is generic and can be easily transferred to other DNN-based frame interpolation algorithm. Our source code is available at* https://github.com/tding1/CDFI.

1. Introduction

Video frame interpolation is a lower level computer vision task referring to the generation of intermediate (non-existent) frames between actual frames in a sequence, which is able to largely increase the temporal resolution. It plays an important role in many applications, including frame rate up-conversion [4], slow-motion generation [27], and novel view synthesis [20, 66]. Though fundamental, the problem is challenging in that the complex motion, occlusion and feature variation in real world videos are difficult to estimate and predict in a transparent way.

Figure 1. **A challenging example consists of large motion, severe occlusion and non-stationary finer details.** From top to bottom: the overlaid two inputs, the ground-truth middle frame, the frame generated by AdaCoF [32], the frame generated by the 10× compressed AdaCoF, and the frame generated by our method. The compressed AdaCoF even outperforms the full one in this case.

Recently, a large number of researches have been conducted in this area, especially those based on deep neural networks (DNN) for their promising results in motion estimation [18, 26, 55, 58], occlusion reasoning [2, 27, 46] and image synthesis [19, 20, 28, 30, 66]. In particular, due to the

*Equal contribution. This work was done when Tianyu Ding was an intern at Applied Sciences Group, Microsoft.

†Corresponding author.

rapid expansion in optical flow [1, 60], many approaches either utilize an off-the-shelf flow model [2, 41, 42, 61] or estimate their own task-specific flow [27, 36, 62, 63, 45] as a guidance of pixel-level motion interpolation. However, integrating a pre-trained flow model makes the whole architecture cumbersome, while with only pixel-level information the task-oriented flow alone is still insufficient in handling complex occlusion and blur. As opposed to this, kernel-based methods [43, 44, 46] synthesize the intermediate frames by convolution operations over local patches surrounding each output pixel. Nevertheless, it cannot deal with large motions beyond the kernel size and it typically suffers from high computational cost. There are also hybrid methods [2, 3] that combine the advantages of flow-based and kernel-based methods, but the networks are much heavier and thus limit their applications.

We observe a growing tendency that more and more complicated and heavy DNN-based models are designed for interpolating video frames. Most of the methods proposed in the past few years [2, 3, 12, 16, 27, 32, 44, 61] involve training and inference on DNN models consisting of over 20 million parameters. For example, the hybrid MEMC-Net [3] consists of more than 70 million parameters and requires around 280 megabytes if stored in 32-bit floating point. Normally, large models are difficult to train and inefficient during inference. Moreover, they are not likely to be deployed on mobile devices, which restricts their scenarios to a great extent. In the mean time, other work [14, 36, 62, 63] directly focus on simple and light-weight video interpolation algorithms. However, they either perform less competitively on benchmark datasets or are bound to specific design that lack of transferability.

In this paper, we propose a *compression-driven* network design for video interpolation (CDFI) that takes advantage of model compression [5, 13, 67]. To the best of our knowledge, we are the first to explore the *over-parameterization* issue appearing in the state-of-the-art DNN models for video interpolation. Concretely, we compress the recently proposed AdaCoF [32] via fine-grained pruning [67] based on sparsity-inducing optimization [7], and show that a 10× compressed AdaCoF is still able to maintain a similar benchmark performance as before, indicating a considerable amount of redundancy in the original model. *The compression provides us two direct benefits: (i) it helps us understand the model architecture in depth, which in turn inspires an efficient design; (ii) the obtained compact model makes more room for further improvements that could potentially boost the performance to a new level.* Towards justifying the latter point, observing that AdaCoF is capable of handling large motion while is short of dealing with occlusion or preserving finer details, we improve upon the compact model by introducing a multi-resolution warping module that utilizes a feature pyramid representation of the input frames to help with the image synthesis. As a result, our final model outperforms AdaCoF on three benchmark datasets with a large margin (more than 1 dB of PSNR on the Middlebury [1] dataset) while is only a quarter of its initial size. Note that typically it is difficult to implement the same improvements on the original heavy model. Experiments show that our model also performs favorably against other state-of-the-art methods.

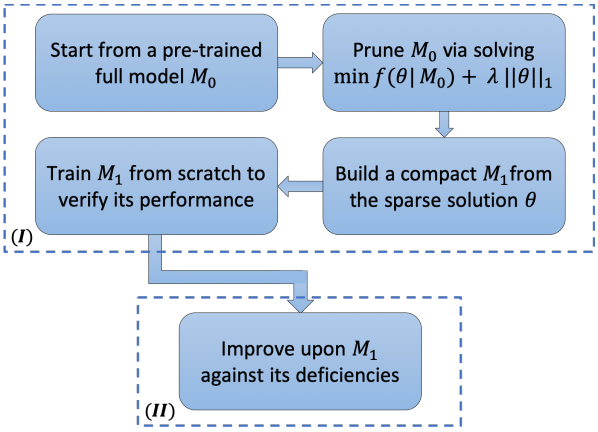

Figure 2. **Pipeline of CDFI.** Stage (I): compression of the baseline; Stage (II): improvements upon the compression.

In short, we present a compression-driven framework for video interpolation, in which we take a step back with reflections on over-parameterization. We first compress AdaCoF and obtain a compact model but performs similarly well, then we improve on top of it. The pipeline of CDFI is illustrated in Figure 2. This retrospective approach leads to superior performance and can be easily transferred to any other DNN-based frame interpolation algorithm.

2. Related work

2.1. Video frame interpolation

Conventional video frame interpolation is modeled as an image sequence problem, *e.g.*, the path-based [38] and phase-based approach [39, 40]. Unfortunately, these methods are less effective in complex scenes due to their incapability of accurately estimating the path (flow) or representing high-frequency components.

Convolutional neural network (CNN) has recently demonstrated its success in understanding temporal motion [18, 26, 49, 55, 58, 60] through predicting optical flow, leading to flow-based motion interpolation algorithms. [37] trains a deep CNN to directly synthesize the intermediate frame. [36] estimates the flow by sampling from the 3D spatio-temporal neighborhood of each output pixel. [27, 45, 62, 63] utilize bi-directional flows to warp frames and resort to additional modules to handle occlusion. [41, 42] integrate an off-the-shelf flow model [55] into the network. Also, quadratic [34, 61] and cubic [14] non-liner

models are proposed to approximate complex motions.

One major drawback of the flow-based methods is that only pixel-wise information is used for interpolation. In contrast, kernel-based methods propose to generate the image by convolving over local patches near each output pixel. For example, [44] estimates spatially-adaptive 2D convolution kernels and [43] improves its efficiency by using pairs of 1D kernels for all output pixels simultaneously. [2, 3] integrate both optical flow and local kernels; specifically [2] detects the occlusion with depth information. However, those methods only rely on local kernels and cannot deal with large motion beyond the rectangular kernel region.

Inspired by the flexible spatial sampling locations of deformable convolution (DConv) [17, 68], [32] proposes the AdaCoF model that utilizes a spatially-adaptive separable DConv to synthesize each output pixel. [51] generalizes it by allowing sampling in the full spatial-temporal space. [12] is similar to AdaCoF except that it estimates 1D separable kernels to approximate 2D kernels. [11] extends [12] to produce the intermediate frame at arbitrary time step. This paper is also based on AdaCoF; however, unlike the previous work, for the first time we explore the over-parameterization issue presenting in the existing DNN-based approaches, and show that a much smaller model performs similarly well through compression. Moreover, by addressing its drawbacks upon the compression, one can easily build a model (still small) so that it outperforms the original one to a large extent. This compression-driven network design is generic and can be transferred to any other DNN-based frame interpolation algorithms.

2.2. Pruning-based model compression

Model compression [5, 13] is particularly important to DNN models, which are known to suffer high cost of storage and computation. In general, model compression can be categorized into several types: pruning [67], quantization [48], knowledge distillation [25] and AutoML [24]. We adopt the pruning technique for its simplicity, which seeks to induce sparse connections. There are many hybrid pruning methods [10, 22, 56] that are suitable for model deployment, but they may be overkill for our purpose of searching and designing the architecture *after the compression*. That being said, compression plays a completely different role in our work, namely it works as a tool for a better understanding of the underlying architecture and makes room for further improvements. For this reason, we turn our attention to optimization-based sparsity-inducing pruning techniques [31, 33, 59, 65] which involve training with sparsity constraints, *e.g.* ℓ_0 or ℓ_1 regularizers. Specifically, we use a simple three-step pipeline (see Stage (I) in Figure 2) which is most similar to [8, 23] that involves: (i) training with ℓ_1-norm sparsity constraint; (ii) reformulating a small dense network according to the sparse structures identified in each layer; and (iii) retraining the small network to verify its performance. We will see shortly (Sec. 3.2) that its implementation and test is straightforward.

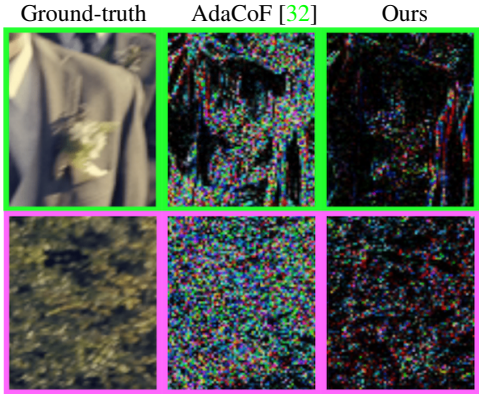

Figure 3. **Visualization of the difference between the interpolation and the ground-truth image.**

3. The proposed approach

Given two consecutive frames I_0 and I_1 in a video sequence, the goal of video frame interpolation is to synthesize an intermediate frame I_t, where $t \in (0, 1)$ is an arbitrary temporal position. A common practice is $t = 0.5$, that is synthesizing the middle frame between I_0 and I_1. We now introduce the proposed CDFI framework with the recently proposed AdaCoF [32] as an instance.

3.1. Motivation

To describe AdaCoF, we begin with the introduction of one of its key components, a spatially-adaptive separable DConv operation for synthesizing one image (denoted by I_{out}) from another one (denoted by I_{in}). Towards synthesizing I_{out} from I_{in}, the input image I_{in} is padded such that I_{out} preserves the original shape of I_{in}. For each pixel (i, j) in I_{out}, AdaCoF computes $I_{\text{out}}(i, j)$ by convolving a deformable patch surrounding the reference pixel (i, j) in I_{in}:

$$\sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{i,j}^{(k,l)} I_{\text{in}}\big(i + dk + \alpha_{i,j}^{(k,l)}, j + dl + \beta_{i,j}^{(k,l)}\big), \quad (1)$$

where F is the deformable kernel size, $W_{i,j}^{(k,l)}$ is the (k, l)-th kernel weight in synthesizing $I_{\text{out}}(i, j)$, $\vec{\Delta} := \big(\alpha_{i,j}^{(k,l)}, \beta_{i,j}^{(k,l)}\big)$ is the offset vector of the (k, l)-th sampling point associated with $I_{\text{in}}(i, j)$, and $d \in \{0, 1, 2, \cdots\}$ is the dilation parameter that helps to explore a wider area. Note that the values of F and d are pre-determined. For synthesizing each output pixel in I_{out}, a total number of F^2 points are sampled in I_{in}. With the offset vector $\vec{\Delta}$, the F^2 sampling points are not necessarily restricted inside a rigid rectangular region centered at the reference point. On the other hand, unlike the classic DConv, AdaCoF uses different ker-

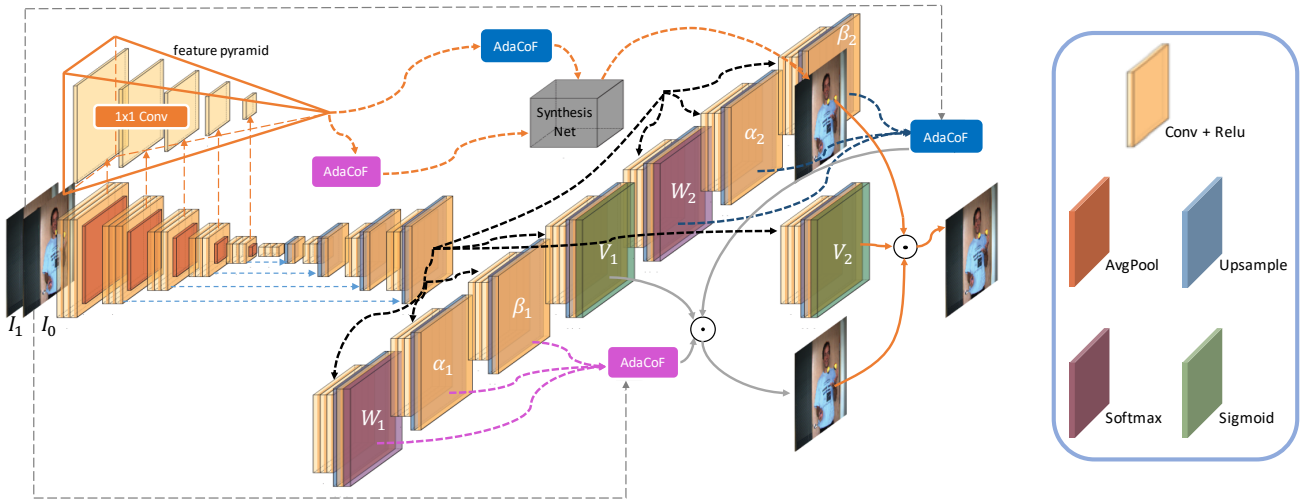

Figure 4. **Illustration of our architecture design based on the compressed AdaCoF [32].** The lower part (AdaCoF) consists of a U-Net, a group of sub-networks for estimating two sets of $\{W_i, \alpha_i, \beta_i\}$ in AdaCoF operation (1) correspond to backward/forward warping, and an occlusion mask V_1 for synthesizing one candidate intermediate frame $I_{0.5}^{(1)}$. The upper part (our design) extracts a feature pyramid representation of the input frames through 1-by-1 convolutions from the encoder of the U-Net, then the multi-scale features are warped by AdaCoF operation of learned backward/forward parameters, which are fed into a synthesis network to generate another candidate intermediate frame $I_{0.5}^{(2)}$. Note that the pink and blue AdaCoF modules are associated with $\{W_1, \alpha_1, \beta_1\}$ and $\{W_2, \alpha_2, \beta_2\}$, respectively. Finally, the network outputs the interpolation frame by blending $I_{0.5}^{(1)}$ and $I_{0.5}^{(2)}$ via an extra occlusion mask V_2.

nel weights across different reference pixels (i, j), indicated by $W_{i,j}^{(k,l)}$ in (1); hence the attribute "separable" [44].

Although AdaCoF is flexible in handling large and complex motion since the parameters $\{W_{i,j}^{(k,l)}, \alpha_{i,j}^{(k,l)}, \beta_{i,j}^{(k,l)}\}$ are computed specifically for each pair of input frames, it is unable to deal with severe occlusion and non-stationary finer details, as shown in Figure 1. We further visualize the difference between the interpolation and the ground-truth in Figure 3. AdaCoF is insufficient in preserving the contextual information because the interpolation is simply obtained by blending the two warped frames through a sigmoid mask (V_1), as demonstrated in Figure 4. A natural question to ask is that if we can make direct improvements on top of it. However, we find the architecture design of the AdaCoF model is relatively cumbersome, especially the encoder-decoder part. For example, six $512 \times 512 \times 3 \times 3$ convolutional layers are employed in the middle, which is an entire heuristic since it is unclear whether this design is sufficient or not for the interpolation task. The original AdaCoF model has 21.8 millions of parameters when $F = 5, d = 1$ and requires 83.4 megabytes if stored with PyTorch. Typically, such a large model takes a long time for training and validation, and thus prohibits direct improvements upon it. Towards better understanding the architecture and improving its performance, we propose a compression-driven approach described as follows.

3.2. First stage: compression of the baseline

As the first stage in our approach, we compress the baseline model, *i.e.*, AdaCoF here, by leveraging the fine-grained model pruning [67] through sparsity-inducing optimization [6]. Specifically, given a pre-trained full model M_0, we start by re-training (fine-tuning) its weights θ by imposing an ℓ_1 norm sparsity regularizer, and solve the following optimization problem:

$$\min_{\theta} \; f(\theta|M_0) + \lambda\|\theta\|_1, \tag{2}$$

where $f(\cdot)$ denotes the training objective for our task (see Sec. 3.4 for details) and $\lambda > 0$ is the regularization constant. It is known that with appropriately chosen λ the formulation (2) promotes a sparse solution, with which one can easily identify those important connections among neurons, namely the ones corresponding to non-zero weights. Towards solving (2), we utilize the newly proposed orthant-based stochastic method [7] for its efficient mechanism in promoting sparsity and less performance regression compared with other solvers. By solving the ℓ_1-regularized problem (2), we indeed perform a fine-grained pruning since zeros are promoted in an unstructured manner. Note that one can also impose group sparsity constraints [9, 31, 65], *e.g.*, mixed ℓ_1/ℓ_2, to prune the kernel weights in a group-wise fashion. We only adopt the ℓ_1 constraint in the presentation for its simplicity.

After obtaining a sparse solution $\hat{\theta}$, different than [23] that directly operates on the sparse network, we re-design a small dense network M_1 based on the sparsity computed in each layer. Given the l-th convolutional layer consisting of $K_l = C_l^{\text{in}} \times C_l^{\text{out}} \times q \times q$ parameters (denoted as $\hat{\theta}_l$), where C_l^{in} is the number of input channels, C_l^{out} is the number of output channels, $q \times q$ is the kernel size, then the sparsity s_l

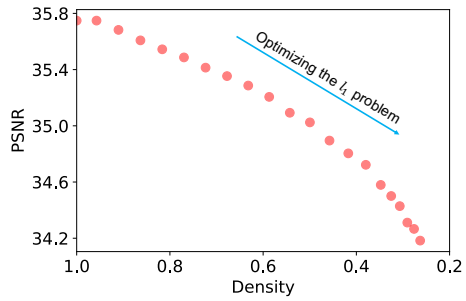

Figure 5. **Plot of PSNR against the density of AdaCoF, trained on Middlebury, within 20 epochs of optimizing equation (2).**

and density ratio d_l of this layer are respectively defined as

$$s_l := \big(\#\text{ of zeros in } \hat{\theta}_l\big)/K_l \quad \text{and} \quad d_l := 1 - s_l. \tag{3}$$

Inspired by [8], we use d_l as the *compression ratio* and compute $\widetilde{C}_l^{\text{in}} := \lceil d_l \cdot C_l^{\text{in}} \rceil$ as the number of kernels we actually need in that layer. The main intuition is that the density ratio d_l reflects the minimal amount of necessary information that needs to be encoded in that layer without affecting performance largely. Since $C_l^{\text{in}} \equiv C_{l-1}^{\text{out}}$, we also update $\widetilde{C}_{l-1}^{\text{out}} = \widetilde{C}_l^{\text{in}}$, then repeat the above process for computing the number of kernels in the $(l-1)$-th layer by $\widetilde{C}_{l-1}^{\text{in}} := \lceil d_{l-1} \cdot C_{l-1}^{\text{in}} \rceil$, and so on. In words, we reformulate a small network by updating the number of kernels in each convolutional layer according to its density ratio, and proceed from back to the front. Since AdaCoF is fully convolutional (see Figure 4), the above procedure can be easily implemented by reducing the number of input/output channels for each layer, leading to a much more compact architecture. In fact, the strategy does not bind to convolutional layers. One can also operate on a fully connected layer by re-computing its number of input/output features accordingly, making it extensible to other architectures.

Finally, we train the compressed model M_1 from scratch (without the ℓ_1 constraint) to verify its performance. Typically, it takes a significantly shorter time than that of the full model M_0 due to its compactness. The entire compression pipeline is illustrated as the Stage (I) of Figure 2. We remark that a pre-trained M_0 is not necessarily required for the sake of compression since problem (2) is suitable for a one-shot training/pruning, but M_0 allows us to make sure the compressed model works competitively as before.

Compression of AdaCoF. We now apply the compression strategy to the AdaCoF model [32]. We use the pre-trained model provided by the authors. Starting with the ℓ_1-regularized problem (2), where λ is set as 10^{-4}, we run the orthant-based stochastic solver [7] for 20 epochs by feeding the model with only 1000 video triplets from Vimeo-90K [62]. For each epoch, we record the network density and the PSNR evaluated on the Middlebury dataset [1], as plotted in Figure 5. One can see that the model performance declines as more sparsity is promoted. After 20 epochs of training, the density of the network is down to 26%. Interestingly, by examining the density ratio of each layer, we find that the six $512\times512\times3\times3$ convolutional layers in the middle of the U-Net are among the most redundant portions. In particular, the $512 \times 512 \times 3 \times 3$ convolutional layer following the upsampling layer achieves a density ratio of only 7%, which means 93% of the kernel is of little use. This observation confirms our previous conjecture that the original architecture design has a considerable amount of redundancy. Then we reformulate a compact network guided by the computed density ratio in each layer, as described before, and train it from scratch by using the entire taining set (51312 video triplets) of Vimeo-90K. In this case, the training of the compressed AdaCoF is about $5\times$ faster than previously. When it finishes, we compare the before-and-after models in Table 1, where PSNR and SSIM [57] are evaluated on the Middlebury dataset, and time and FLOPS are calculated in synthesizing a $3 \times 1280 \times 720$ frame on RTX 6000 Ti GPU. Note that although the PSNR drops below 34.2 during the ℓ_1 optimization, after formulating and training the compressed model rises again to 35.46, which is on par with the original uncompressed AdaCoF. We conclude that a $10\times$ compressed AdaCoF still maintains a similar performance as its original counterpart.

	Original AdaCoF ($F=5, d=1$)	After Compression
PSNR	**35.72**	35.43
SSIM	**0.96**	**0.96**
Size (MB)	83.4	**9.4**
Time (ms)	82.6	**60.4**
FLOPS (G)	359.2	**185.9**
Parameters (M)	21.8	**2.45**

Table 1. **The statistics of AdaCoF and the compressed version.**

3.3. Second stage: improve upon the compression

In the second stage, we improve upon the compression against its deficiencies. The point is that the compression makes room for further improvements due to its compactness, which is typically difficult if directly operating on the original large model, *e.g.*, the long training and validation time appears daunting in the first place.

Observing that AdaCoF is short of handling severe occlusion and preserving finer details, we design three specific components, *i.e.*, a feature pyramid, an image synthesis network and a path selection mechanism, on top of the compressed AdaCoF. Note that the improvements are case by case since different baselines have their own weakness.

Feature pyramid. In AdaCoF, the final interpolation frame is computed by blending the two warped frames through a single sigmoid mask V_1 (see Figure 4), which is a generalization of using a binary mask to determine the occlusion weights of the two warped frames for each output pixel. We argue that with only raw pixel information the loss of contextual details in the input frames is inevitable

	Vimeo-90K [62]			Middlebury [1]			UCF101-DVF [36]			Parameters (million)
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	
AdaCoF ($F = 5, d = 1$)	34.35	0.956	0.019	35.72	0.959	0.019	35.16	**0.950**	0.019	21.84
Compressed AdaCoF ($F = 5, d = 1$)	34.10	0.954	0.020	35.43	0.957	0.018	35.10	**0.950**	0.019	**2.45**
AdaCoF+ ($F = 11, d = 2$)	34.56	0.959	0.018	36.09	0.962	0.017	35.16	**0.950**	0.019	22.93
Compressed AdaCoF+ ($F = 11, d = 2$)	34.44	0.958	0.019	35.73	0.960	0.018	35.13	**0.950**	0.019	2.56
Ours: FP ($F = 5, d = 1$)	34.62	0.962	0.011	36.13	0.961	0.008	35.08	0.949	**0.015**	4.88
Ours: FP + 1x1 Conv ($F = 5, d = 1$)	34.82	0.963	0.011	36.52	0.964	0.008	35.11	0.949	**0.015**	4.72
Ours: FP + 1x1 Conv ($F = 11, d = 2$)	34.82	0.963	0.011	36.70	0.964	0.008	35.14	0.949	**0.015**	4.83
Ours: FP + 1x1 Conv + PS ($F = 11, d = 2$)	**35.17**	**0.964**	**0.010**	**37.14**	**0.966**	**0.007**	**35.21**	**0.950**	**0.015**	4.98

Table 2. **Ablation experiments on the architecture design of our approach.**

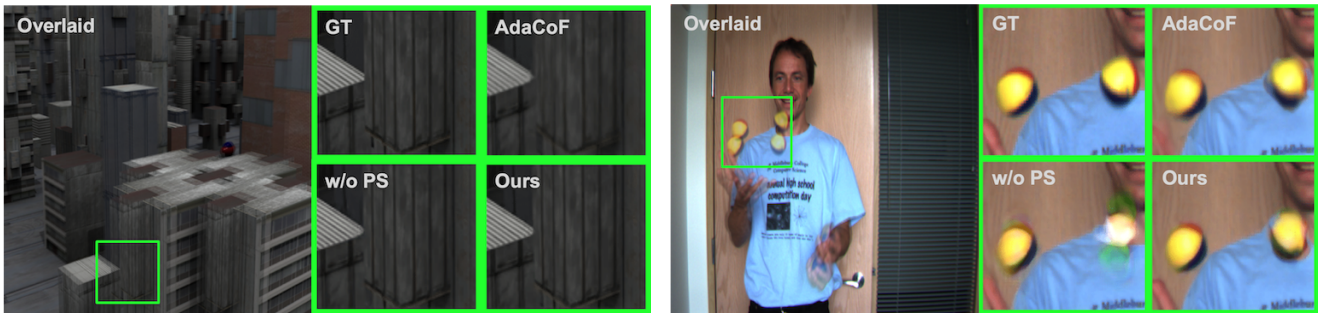

Figure 6. **Examples of adding the path selection (PS) mechanism in our design.**

since it lacks guidance from the feature space. Instead, we extract a feature pyramid representation [42] of the input frames from the encoder part of the U-Net. Specifically, it has five feature levels in accordance with the encoder, and for each level we utilize a 1-by-1 convolution to filter the encoder at multi-scale with $4, 8, 12, 16, 20$ output features (in descending order by the feature scale). The extracted multi-scale features are then warped by AdaCoF operation (1), which captures the motion in the feature space.

Image synthesis network. To better make use of the extracted multi-scale features, we resort to a GridNet [21] architecture with three rows and six columns in synthesizing the image, which is also employed in [41, 42] for its superiority in combining multi-scale information. Particularly, we feed the synthesis network with both the forward- and backward-warped multi-scale feature maps, generating a single RGB image that focuses on the contextual details.

Path selection. In order to take advantage of both AdaCoF (handling complex motion) and our own components (handling contextual details), we apply a path selection mechanism in generating the final interpolation result. As shown in Figure 4, one path leads to the output of the original AdaCoF (denoted as $I_{0.5}^{(1)}$), which is computed by blending two warped input frames using the occlusion mask V_1. Parallel to this, another path leads to the output of the synthesis network (denoted as $I_{0.5}^{(2)}$), which is computed by combining the warped multi-scale feature maps. In the end, we learn another occlusion module V_2 to synthesize the final result from $I_{0.5}^{(1)}$ and $I_{0.5}^{(2)}$, and we expect that $I_{0.5}^{(2)}$ can compensate for the lack of contextual information in $I_{0.5}^{(1)}$.

The above three specific components can not only be easily incorporated into the compressed AdaCoF, but also boost the performance to a large extent while still maintain the compactness (see Sec. 4).

3.4. Training

With the architecture described above, we train it using AdaMax [29] with $\beta_1 = 0.9, \beta_2 = 0.999$, an initial learning rate of 0.001 which decays half every 20 epochs, a mini-batch size of 8, and a maximum training epochs of 100.

Objective function. Given the interpolated frame I_{out} of our network and its ground truth I_{gt}, we first employ the Charbonnier penalty [36] as a surrogate for the ℓ_1 loss:

$$\mathcal{L}_{\text{Charbon}} = \rho(I_{\text{out}} - I_{\text{gt}}) \tag{4}$$

where $\rho(\boldsymbol{x}) = (\|\boldsymbol{x}\|_2^2 + \epsilon^2)^{1/2}$ and ϵ is set to 0.001. Next, we follow [32] and use a perceptual loss with feature ϕ extracted from `conv4_3` of the pre-trained VGG16 [52]:

$$\mathcal{L}_{\text{vgg}} = \|\phi(I_{\text{out}}) - \phi(I_{\text{gt}})\|_2. \tag{5}$$

Then, inspired by the implementation of AdaCoF, we use a total variation loss imposed on the offset vectors for ensuring spatial continuity and smoothness:

$$\mathcal{L}_{\text{tv}} = \tau(\boldsymbol{\alpha}_1) + \tau(\boldsymbol{\alpha}_2) + \tau(\boldsymbol{\beta}_1) + \tau(\boldsymbol{\beta}_2) \tag{6}$$

where $\tau(I) = \sum_{i,j} \rho(I_{i,j+1} - I_{i,j}) + \rho(I_{i+1,j} - I_{i,j})$, and $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2$ are the offsets modules computed within our network. Lastly, we formulate our final loss function as

$$\mathcal{L} = \mathcal{L}_{\text{Charbon}} + \lambda_{\text{vgg}}\mathcal{L}_{\text{vgg}} + \lambda_{\text{tv}}\mathcal{L}_{\text{tv}} \tag{7}$$

where we set $\lambda_{\text{vgg}} = 0.005, \lambda_{\text{tv}} = 0.01$ in the experiments.

	Training dataset	Vimeo-90K [62]			Middlebury [1]			UCF101-DVF [36]			Parameters (million)
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	
†SepConv - $\mathcal{L}_1$ [44]	proprietary	33.80	0.956	0.027	35.73	0.959	0.017	34.79	0.947	0.029	21.6
†SepConv - $\mathcal{L}_F$ [44]	proprietary	33.45	0.951	0.019	35.03	0.954	0.013	34.69	0.945	0.024	21.6
†CtxSyn - $\mathcal{L}_{Lap}$ [41]	proprietary	34.39	0.961	0.024	36.93	0.964	0.016	34.62	0.949	0.031	–
†CtxSyn - $\mathcal{L}_F$ [41]	proprietary	33.76	0.955	0.017	35.95	0.959	0.013	34.01	0.941	0.024	–
†SoftSplat - $\mathcal{L}_{Lap}$ [42]	Vimeo-90K	**36.10**	**0.970**	0.021	**38.42**	**0.971**	0.016	**35.39**	**0.952**	0.033	–
†SoftSplat - $\mathcal{L}_F$ [42]	Vimeo-90K	**35.48**	**0.964**	**0.013**	**37.55**	0.965	**0.008**	35.10	0.948	0.022	–
†DAIN [2]	Vimeo-90K	34.70	**0.964**	0.022	36.70	0.965	0.017	35.00	**0.950**	0.028	24.02
AdaCoF [32]	Vimeo-90K	34.35	0.956	0.019	35.72	0.959	0.019	35.16	**0.950**	**0.019**	21.84
AdaCoF+ [32]	Vimeo-90K	34.56	0.959	0.018	36.09	0.962	0.017	35.16	**0.950**	**0.019**	22.93
EDSC - $\mathcal{L}_C$ [11]	Vimeo-90K	34.86	0.962	0.016	36.76	**0.966**	0.014	35.17	**0.950**	**0.019**	8.9
EDSC - $\mathcal{L}_F$ [11]	Vimeo-90K	34.57	0.958	**0.010**	36.48	0.963	**0.007**	35.04	0.948	**0.015**	8.9
BMBC [45]	Vimeo-90K	35.06	**0.964**	0.015	36.79	0.965	0.015	35.16	**0.950**	**0.019**	11.0
CAIN [16]	Vimeo-90K	34.65	0.959	0.020	35.11	0.951	0.019	34.98	**0.950**	0.021	42.8
Ours	Vimeo-90K	35.17	**0.964**	**0.010**	37.14	**0.966**	**0.007**	**35.21**	**0.950**	**0.015**	**4.98**

Table 3. **Quantitative comparisons with state-of-the-art methods.** The results of methods marked with † are cloned from [42].

Training dataset. We use the Vimeo-90K dataset [62] for training, which contains 51312/3782 video triplets of size 256×448 for training/validation. We further augment the data by randomly flipping them horizontally and vertically as well as perturbing the temporal order.

Evaluation. Besides the validation set of Vimeo-90K, we also evaluate the model on the well-known Middlebury dataset [1] and UCF101 [36, 54]. The metrics we use are PSNR, SSIM [57] and LPIPS [64]. Note higher values of PSNR and SSIM indicate better performance, while for LPIPS a lower value corresponds to a better result.

4. Experiments

4.1. Ablation study

We analyze three components in our proposed method: model compression, feature pyramid, and path selection.

Model compression. As described in Sec. 3.2, we compress the baseline model to remove a large mount of redundancy, which also facilitates the training and inference. In Table 2, we compare the performance of the AdaCoF and the compressed counterpart. It shows that a 10× compressed model does not sacrifice much when evaluated on the three benchmark datasets in different settings of (F, d) (which are the parameters in (1)), revealing the redundancy in AdaCoF and the necessity of the compression stage.

Feature pyramid. In order to better capture the contextual details, we incorporate the feature pyramid (FP) module into the compressed AdaCoF followed by warping operations and an image synthesis network (see Sec. 3.3). We isolate its effect by training a network that only outputs the synthesized image without a path selection mechanism. It turns out that by using only FP module (see "Ours - FP", Table 2) on top of the compressed AdaCoF ($F = 5, d = 1$), we achieve visible improvements in terms of PSNR, SSIM and LPIPS on the Vimeo-90K and Middlebury datasets. Note that it substantially improves LPIPS on all the three benchmark datasets. Moreover, filtering the multi-scale feature maps with 1-by-1 convolutions leads to better PSNR and SSIM as well as a slightly smaller model size.

Path selection. Although by adding only the FP module (and 1-by-1 convolutions) we are able to achieve promising quantitative results as shown in Table 2, it does not take advantage of the capability of AdaCoF in handling complex motion, which can be integrated into our design with the proposed path selection (PS) mechanism. The left example in Figure 6 shows that, when there is only fine detail variations in the input frames, adding PS or not does not quite affect our interpolation performance since FP module is capable of synthesizing details (also note the output of AdaCoF is blurry due to the loss of information). On the other hand, the right example contains large motion of two balls, and with only FP module the model is difficult in capturing the motion of the right ball precisely. In contrast, our final model with the embedded PS mechanism can deal with the large motion very well (even sharper on the edges of the balls compared to AdaCoF). More importantly, our approach preserves the finger shape (see the bottom-left corner) while AdaCoF totally misses that part of information. In conclusion, our completed model with FP and PS can handle both fine details and large motion, and achieves significant improvements when evaluated quantitatively.

4.2. Quantitative evaluation

We evaluate our compression-driven approach based on AdaCoF with $F = 11, d = 2$ against the other state-of-the-art DNN methods in Table 3. Since SepCov [44], CtxSyn [41] and SoftSplat [42] are not open source, we directly copy their numerical results as well as DAIN's [2] from [42]. For the rest of the methods, we evaluate their pre-trained models on the three datasets. Note that "AdaCoF" corresponds to the setting of $F = 5, d = 1$ while "AdaCoF+" is associated with $F = 11, d = 2$.

As shown in Table 3, first note that our approach performs favorably against all the compared methods in terms of SSIM and LPIPS. As for PSNR, the proposed method also outperforms all the other methods with a large margin

Figure 7. **Visual comparisons on the DAVIS 2016 dataset [47].** Our compression-driven method not only outperforms the baseline model AdaCoF but also is more appealing compared with more recently proposed methods in handling large motion, occlusion and fine details.

except for SoftSplat [42]. Moreover, our model is significantly smaller than the other competitors. We remark that in the past there do exist some light-weight frame interpolation models, *e.g.*, DVF [36], ToFlow [62] and CyclicGen [35], but they fail to compete with SepConv [44] or CtxSyn [41] as reported in [42]. Lastly, we observe that AdaCoF [32] is only mediocre among those methods, but our final model which is based upon it has a significantly better performance while maintains compactness, indicating the superiority of the proposed CDFI design framework.

4.3. Qualitative evaluation

We demonstrate the visual comparisons on the DAVIS dataset [47] in Figure 7. The first and third example contain complex motion and occlusion, while the second example involves many non-stationary finer details. Note that AdaCoF+ [32] generates relatively blurry interpolation frame for all these examples (see the motorbike, house and swing stool). In contrast, our method built upon it predicts sharper and more realistic results due to our newly added FP module and PS mechanism. Furthermore, we compare with BMBC [45], CAIN [16] and EDSC [11], which are all newly developed within the year. In particular, similar to AdaCoF, EDSC relies on the deformable separable convolution but estimates an extra mask to help with the image synthesis. However, they are not as appealing as our method on the provided examples. One can see that their interpolations normally contain visible artifacts and are not capable of preserving clear details. Note that BMBC [45] occasionally synthesizes sharp results but is not as consistent as ours. We conjecture that the additional bilateral cost volume in BMBC benefits the intermediate motion estimations, which can also be incorporated into our design. Recall that the size of our model is the smallest among them, which again confirms the advantage of the CDFI network design.

5. Conclusions

We presented a compression-driven network design for frame interpolation (CDFI) that uses model compression as a guide in determining an efficient architecture and then improves upon it. For the first time, we considered the redundancy in the existing methods. As an instance, we showed that a much smaller AdaCoF model performs similarly as the original one, and with simple modifications it is able to outperform the baseline with a large extent and is also superior against other state-of-the-art methods. We emphasize that the optimization-based compression over a baseline model does not rely on particular design of the baseline. Therefore, we believe that our framework is generic to be applied to other models and provides *a new perspective* on developing efficient frame interpolation algorithms. In future work, it will be of interest to construct a better association between the compression and design stages which iteratively refines the underlying architecture.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[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.

[4] Wenbo Bao, Xiaoyun Zhang, Li Chen, Lianghui Ding, and Zhiyong Gao. High-order model and dynamic filtering for frame rate up-conversion. *IEEE Transactions on Image Processing*, 27(8):3813–3826, 2018.

[5] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[6] Tianyi Chen. *A Fast Reduced-Space Algorithmic Framework for Sparse Optimization*. PhD thesis, Johns Hopkins University, 2018.

[7] Tianyi Chen, Tianyu Ding, Bo Ji, Guanyi Wang, Yixin Shi, Sheng Yi, Xiao Tu, and Zhihui Zhu. Orthant based proximal stochastic gradient method for ℓ_1-regularized optimization. *arXiv preprint arXiv:2004.03639*, 2020.

[8] Tianyi Chen, Bo Ji, Yixin Shi, Biyi Fang, Sheng Yi, Tianyu Ding, and Xiao Tu. Neural network compression via sparse optimization. *arXiv preprint arXiv:2011.04868*, 2020.

[9] Tianyi Chen, Guanyi Wang, Tianyu Ding, Bo Ji, Sheng Yi, and Zhihui Zhu. Half-space proximal stochastic gradient method for group-sparsity regularized problem. *arXiv preprint arXiv:2009.12078*, 2020.

[10] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294, 2015.

[11] Xianhang Cheng and Zhenzhong Chen. Multiple video frame interpolation via enhanced deformable separable convolution. *arXiv preprint arXiv:2006.08070*, 2020.

[12] Xianhang Cheng and Zhenzhong Chen. Video frame interpolation via deformable separable convolution. In *AAAI*, pages 10607–10614, 2020.

[13] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[14] 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.

[15] 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.

[16] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *AAAI*, pages 10663–10671, 2020.

[17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[18] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[19] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[20] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5515–5524, 2016.

[21] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Tremeau, and Christian Wolf. Residual conv-deconv grid network for semantic segmentation. *arXiv preprint arXiv:1707.07958*, 2017.

[22] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[23] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[24] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision*, pages 784–800, 2018.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[27] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[28] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field

cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[31] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564, 2016.

[32] 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*, pages 5316–5325, 2020.

[33] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[34] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. *arXiv preprint arXiv:2009.04642*, 2020.

[35] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang. Deep video frame interpolation using cyclic frame generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8794–8802, 2019.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020.

[43] 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.

[44] 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.

[45] 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.

[46] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. Im-net for high resolution video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2398–2407, 2019.

[47] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[48] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[49] Lars Lau Rakêt, Lars Roholm, Andrés Bruhn, and Joachim Weickert. Motion compensated frame interpolation with a symmetric optical flow constraint. In *International Symposium on Visual Computing*, pages 447–457. Springer, 2012.

[50] 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.

[51] Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video interpolation via generalized deformable convolution. *arXiv preprint arXiv:2008.10680*, 2020.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[53] 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.

[54] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[55] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[56] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[57] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[58] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013.

[59] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[60] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. Optical flow guided tv-l 1 video interpolation and restoration. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 273–286. Springer, 2011.

[61] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *Advances in Neural Information Processing Systems*, pages 1647–1656, 2019.

[62] 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.

[63] 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.

[64] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[65] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[66] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.

[67] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[68] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019.