Scene-Adaptive Video Frame Interpolation via Meta-Learning

Myungsub Choi[1] Janghoon Choi[1] Sungyong Baik[1] Tae Hyun Kim[2] Kyoung Mu Lee[1]
[1]ASRI, Department of ECE, Seoul National University [2]Department of CS, Hanyang University
[1]{cms6539, ultio791, dsybaik, kyoungmu}@snu.ac.kr [2]taehyunkim@hanyang.ac.kr

Abstract

Video frame interpolation is a challenging problem because there are different scenarios for each video depending on the variety of foreground and background motion, frame rate, and occlusion. It is therefore difficult for a single network with fixed parameters to generalize across different videos. Ideally, one could have a different network for each scenario, but this is computationally infeasible for practical applications. In this work, we propose to adapt the model to each video by making use of additional information that is readily available at test time and yet has not been exploited in previous works. We first show the benefits of 'test-time adaptation' through simple fine-tuning of a network, then we greatly improve its efficiency by incorporating meta-learning. We obtain significant performance gains with only a single gradient update without any additional parameters. Finally, we show that our meta-learning framework can be easily employed to any video frame interpolation network and can consistently improve its performance on multiple benchmark datasets.

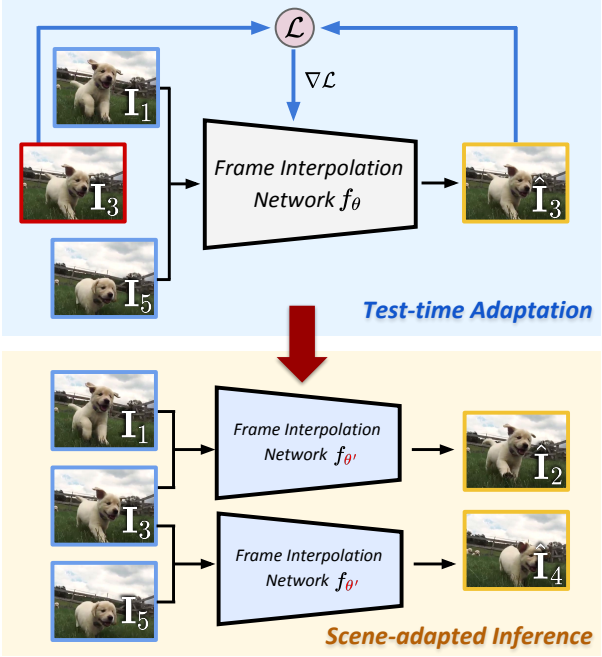

Figure 1. **Motivation of the proposed video frame interpolation method.** Our video frame interpolation framework incorporates a test-time adaptation process followed by scene-adapted inference. The adaptation process takes advantage of additional information from the input frames and is quickly performed with only a single gradient update to the network.

1. Introduction

Video frame interpolation aims to upscale the temporal resolution of a video, by synthesizing intermediate frames in-between the neighboring frames of the original input. Owing to its wide range of applications, including slow-motion generation and frame-rate up-conversion that provide better visual experiences with more details and less motion blur, video frame interpolation has gained substantial interest in the computer vision community. Recent advances of deep convolutional neural networks (CNNs) for video frame interpolation [16, 20, 29, 30, 31, 48] lead to a significant boost in performance. However, generating high-quality frames is still a challenging problem due to large motion and occlusion in a diverse set of scenes.

Previous approaches to video frame interpolation [16, 20, 29, 30, 31, 48], as well as other learning-based video processing models [6, 7, 40, 49, 50], typically require a huge amount of data for training. However, videos in the wild comprise of various distinctive scenes with many different types of low-level patterns. This makes it difficult for a single model to perform well on all possible test cases, even if trained with large datasets.

This problem can be alleviated by making the model adaptive to the specific input data. Utilizing the additional information only available at test time and customizing the model to each of the test data samples has shown to be effective in numerous areas. Examples include single-image super-resolution approaches exploiting self-similarities inherent in the target image [12, 14, 15, 24, 39], or many visual tracking methods where online adaptation to the input video sequence is crucial in performance [8, 10, 27]. However, most works either increase the number of parameters

or require considerable inference time for test-time adaptation of the network parameters.

Meta-learning, also known as *learning to learn*, can take a step forward to remedy current limitations in test-time adaptation. The goal of meta-learning is to design algorithms or models that can quickly adapt to new tasks from small set of training examples given during testing phase. It has been gaining tremendous interest in solving few-shot classification/regression problems as well as some reinforcement learning applications [11], but employing meta-learning techniques to low-level computer vision problems has yet to be explored.

To this end, we propose a scene-adaptive video frame interpolation algorithm that can rapidly adapt to new, unseen videos (or tasks, in meta-learning viewpoint) at test time and achieve substantial performance gain. A brief overview of the main idea of our approach is illustrated in Fig. 1. Using any off-the-shelf existing video frame interpolation framework, our algorithm updates its parameters using the frames only available at test time, and uses the adapted model to interpolate intermediate frames in the same way as the conventional approaches.Although the proposed method is not applicable for videos with their total length of less than 3 frames, most real-world scenarios have multiple consecutive frames that we can fully utilize for our meta-learning based test-time adaptation scheme.

Overall, our contributions are summarized as follows:

- We propose a novel adaptation framework that can further improve conventional frame interpolation models without changing their architectures.
- To the best of our knowledge, the proposed approach is the first integration of meta-learning techniques for test-time adaptation in video frame interpolation.
- We confirm that our framework consistently improves upon even the most recent state-of-the-art methods.

2. Related works

In this section, we review the extensive literature of video frame interpolation. Existing test-time adaptation schemes for other low-level vision applications and the history of meta-learning algorithms are also described.

Video frame interpolation: While video frame interpolation has a long-established history, we concentrate on recent learning-based algorithms, particularly CNN-based interpolation approaches.

The first attempt to incorporate CNNs to video frame interpolation was done by Long *et al*. [21], where interpolation is obtained as a byproduct of self-supervised learning of optical flow estimation. Since then, numerous approaches have focused on effectively modeling motion and handling occlusions. Meyer *et al*. [22, 23] represent motion as per-pixel phase shift, and Niklaus *et al*. [30, 31] model the sequential process of motion estimation and frame synthesis into a single spatially-adaptive convolution step. Choi *et al*. [9] handles motion with a simple feedforward network with channel attention.

Another line of research use optical flow estimation as an intermediate step (as a proxy) and warp the original frames with the estimated motion map for alignment, followed by further refinement and occlusion handling to obtain the final interpolations [3, 4, 16, 19, 20, 29, 47, 48]. These flow-based methods are generally able to synthesize sharp and natural frames, but some heavily depend on the pre-trained optical flow estimation network and show doubling artifacts in cases with large motion when flow estimation fail. Recently, Bao *et al*. [3] additionally use depth map estimation model to compensate for the missing information in flow estimation and effectively handle the occluding regions.

Test-time adaptation: Contrary to previous works, we explore an orthogonal area of research, adaptation to the inputs at test time, to further improve the accuracy of given video frame interpolation models. Our work is inspired by the success of self-similarity based approaches in image super-resolution [12, 14, 15, 24, 39]. Notably, recent zero-shot super-resolution (ZSSR) method proposed by Shocher *et al*. [39] has shown impressive results by incorporating deep learning. Specifically, ZSSR at test time extracts the patches only from the input image and trains a small image-specific CNN, thereby naturally exploiting the information that is only available after observing the test inputs. However, ZSSR suffers from slow inference time due to its self-training step, and it is prone to overfitting since using a pretrained network trained with large external datasets is not viable for internal training.

For video frame interpolation, Reda *et al*. [35] recently proposed the first approach to adapt to the test data in an unsupervised manner by using a cycle-consistency constraint. However, their method adapts to the general domain of the test data, and cannot adapt to each test sample. On the other hand, the proposed algorithm enables to update the model parameters *w.r.t.* each local part of the test sequence, thus better adapting to local motions and scene textures.

Meta-learning: To achieve test-time adaptation without susceptibility to overfitting and without greatly increasing the cost of computation, we turn our attention to meta-learning. Recently, meta-learning has gained a lot of attention for its high performance in few-shot classification, which evaluates the capability of the system to adapt to new classification tasks with few examples. Meta-learning aims to achieve such adaptation to new tasks (videos in our case) through learning prior knowledge across tasks. [5, 13, 37, 38, 45]. Broadly, one can categorize meta-

learning systems into three classes: metric-based, network-based, and optimization-based. The metric-based meta-learning manifests the prior knowledge by learning a feature embedding space, where different classes are placed far apart and similar classes are placed close to each other [18, 41, 44, 46]. The learned embedding space is then used to learn relationship between a query and support examples in few-shot classification. Network-based meta-learning achieves fast adaptation through encoding input-dependent dynamics into the architecture itself by generating input-conditioned weights [25, 32] or employing an external memory [26, 36]. On the other hand, optimization-based systems aim to encode the prior knowledge into optimization process for fast adaptation [11, 28, 34]. Among optimization-based systems, MAML [11] has greatly enjoyed the attention for its simplicity and generalizability, in contrast to the metric or network-based systems that suffer from the limitations in either applications or scalability issues. The generalizability of its model-agnostic algorithm motivates us to use MAML to integrate test-time adaptation into video frame interpolation.

3. Proposed Method

In this section, we first describe the general problem settings for video frame interpolation. Then, we empirically show the advantage of test-time adaptation with a feasibility test, and justify the need for meta-learning in this scenario.

3.1. Video frame interpolation problem set-up

The goal of video frame interpolation algorithms is to generate a high-quality, high frame-rate video given a low frame-rate input video by synthesizing intermediate frames between two neighboring frames. Standard settings for most frame interpolation models receive two input frames and output a single intermediate frame. Specifically, if we let $\mathbf{I}_1$ and $\mathbf{I}_3$ be the two consecutive input frames, our goal is to synthesize the middle frame $\hat{\mathbf{I}}_2$. Although recent frame interpolation models also consider more complex multi-frame interpolation problem where a frame of any arbitrary time step between two frames can be synthesized, we constrain our discussions to the single-frame interpolation models in this work. However, note that our proposed meta-learning framework described in Sec. 3.4 is model-agnostic and easily generalizable to different settings as long as the model is differentiable.

3.2. Exploiting extra information at test time

We demonstrate the effectiveness of test-time adaptation with a feasibility test and describe the details on our design choices. Starting from a baseline pre-trained frame interpolation model, we aim to fine-tune the model parameters at test time to improve its performance (for each test video sequence). To fine-tune the model, a frame triplet consisting

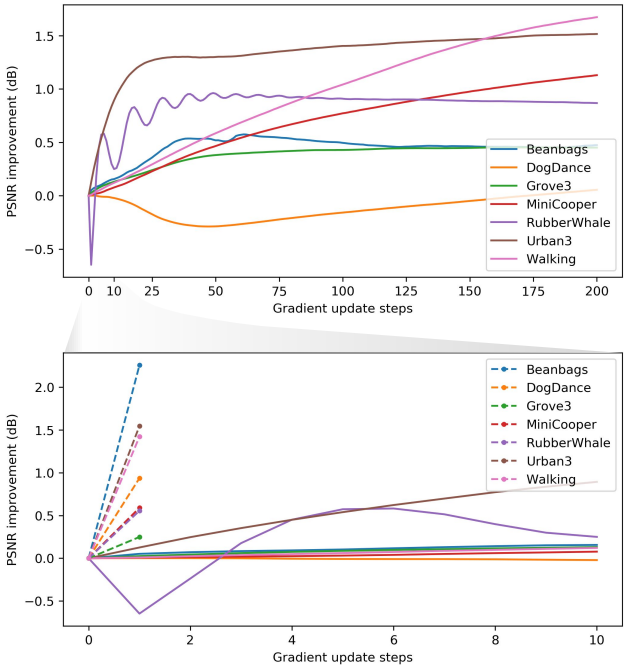

Figure 2. **Feasibility test for test-time adaptation.** Upper graph shows that fine-tuning with the test input data can improve performance in general, but the number of required steps greatly differs for each sequence. Lower graph shows a $\times 20$ zoomed in version of the upper graph, additionally denoting the large performance gain obtained with our *meta-learned* SepConv with a single gradient update.

of 3 consecutive frames are needed, where the first and last frames become the input and the middle frame becomes the target output. While training (fine-tuning) with triplets of a low frame-rate video may seem not beneficial due to the wider time gap, the overall interpolation performance boost has been observed, as shown in the following experiment. This implies the importance of the context and attributes of the given video, such as unique motion and occlusion, and signifies the benefit of test-time adaptation.

For a feasibility test on the effectiveness of test-time adaptation, we fine-tune a pre-trained SepConv [31] model on each sequence from Middlebury [2] dataset. Specifically, we choose 7 sequences from `OTHERS` set, and fine-tune the baseline model with Adamax [17] optimizer (which was used to train the original SepConv model) with a fixed learning rate of 10^{-5}. Batch construction for the gradient update is analogous to Fig. 1, but we increase the number of frames for test-time adaptation from 3 ($t = 1, 3, 5$) to 4 ($t = 1, 3, 5, 7$). In a sense, it can be seen as a 2-shot update, since we can build 2 triplets ($t = (1, 3, 5), (3, 5, 7)$) from the 4 input frames. Updating the model parameters with these 2 triplets for many iterations can tell whether or not this test-time adaptation scheme is advantageous. We measure the performance with peak signal-to-noise ratio (PSNR), and the results for PSNR difference with respect

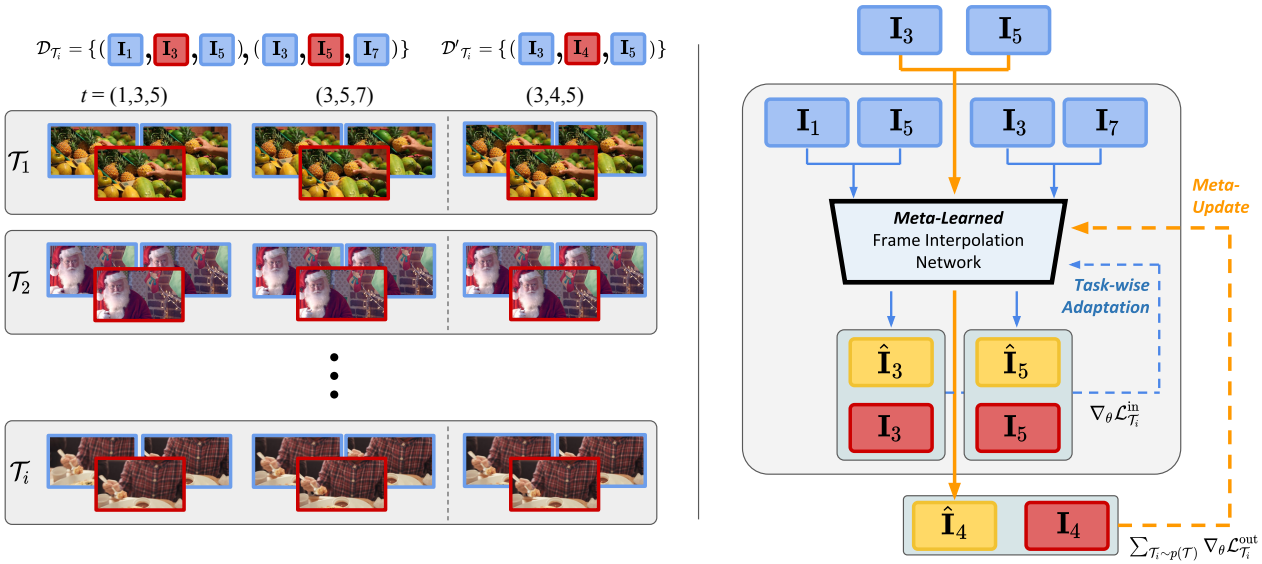

Figure 3. **Overview of the training process for the proposed video frame interpolation network.** ***Left***: Each task $\mathcal{T}_i$ consists of three frame triplets chosen from a video sequence where two are used for task-wise adaptation (*i.e.*, inner loop update) and one is used for meta-update (*i.e.*, outer loop update). ***Right***: Network parameters θ are adapted by gradient descent on loss $\mathcal{L}^{\text{in}}_{\mathcal{T}_i}$ using triplets in $\mathcal{D}_{\mathcal{T}_i}$ and stored for each task, and meta-update is performed by minimizing the sum of each loss $\mathcal{L}^{\text{out}}_{\mathcal{T}_i}$ using the triplets in $\mathcal{D}'_{\mathcal{T}_i}$ for all tasks.

to the number of gradient update steps are shown in Fig. 2.

The characteristics for performance improvements, shown in the upper graph of Fig. 2, greatly differs from sequence to sequence. While the PSNR scores for *Minicooper* and *Walking* steadily improve for 200 gradient updates and do not overfit even after over 1dB gain, updating with *DogDance* sequence hurts the original model's performance in its early stage. Notably, the graph for *RubberWhale* shows a strange characteristic, where the performance severely drops after the first gradient update but suddenly shifts back to the positive side after the subsequent steps. From these results, we can arguably conclude that test-time adaptation is beneficial for video frame interpolation, but how much to adapt (or not adapt at all to avoid overfitting) for each different sequence is hard to decide.

By incorporating meta-learning techniques, our method can enhance the original SepConv model to rapidly adapt to the test sequence, without changing any architectural choices or introducing additional parameters. With just a single gradient update at test time, our *meta-learned* SepConv can achieve large performance gain, as illustrated in the lower graph of Fig. 2. Compared to hundreds of iterations required for fine-tuning the baseline model, our meta-learned SepConv extremely reduces the computation time needed to obtain the same amount of performance boost.

3.3. Background on MAML

Meta-learning aims at rapidly adapting to novel tasks with only a few examples *i.e.* few-shot learning. Recent model-agnostic meta-learning (MAML) [11] approach achieve this goal with only a few gradient update iterations by preparing the model to be readily adaptable to incoming test data. In other words, MAML finds a good initialization of the parameters that are sensitive to changes in task, so that small updates can make large improvements on reducing the error measures and boosting the performance for each new task. Before diving into the main algorithm, we would first like to start with the formulation of the general meta-learning and MAML.

Under the assumption of the existence of task distribution, $p(\mathcal{T})$, the goal of MAML is to learn the initialization parameters that represent the prior knowledge that exists throughout the task distribution. In k-shot learning setting, $\mathcal{D}_{\mathcal{T}_i}$, a set of k number of examples, are sampled from each task, $\mathcal{T}_i \sim p(\mathcal{T})$. The sampled examples, along with its corresponding loss $\mathcal{L}_{\mathcal{T}_i}$, roughly represent the task itself and are used for the model to adapt to the task. In MAML, this is achieved by fine-tuning:

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta). \tag{1}$$

Once the model is adapted to each task, $\mathcal{T}_i$, new examples, $\mathcal{D}'_{\mathcal{T}_i}$, are sampled from the same task to evaluate the generalization of the adapted model on unseen examples. The evaluation acts as a feedback for MAML to adjust its initialization parameters to achieve better generalization:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}). \tag{2}$$

3.4. Meta-learning for frame interpolation

For video frame interpolation, we define a *task* as performing frame interpolation on a frame sequence (video). Fast adaptation to new video scenes via MAML introduces our scene-adaptive frame interpolation algorithm, which is described in detail later in this section.

We consider a frame interpolation model f_θ, parameterized by θ, that receives two input frames $(\mathbf{I}_t, \mathbf{I}_{t+2T})$ and outputs the estimated middle frame $\hat{\mathbf{I}}_{t+T}$ for any time step t and interval T. Thus, a training sample needed to update the model parameters can be formalized as a frame triplet $(\mathbf{I}_t, \mathbf{I}_{t+T}, \mathbf{I}_{t+2T})$. We define a task $\mathcal{T}$ as minimizing the sum of the losses $\mathcal{L} : \{(\mathbf{I}_t, \mathbf{I}_{t+T}, \mathbf{I}_{t+2T})\} \rightarrow \mathbb{R}$ for all time steps t in low frame-rate input video. In our scene-adaptive frame interpolation setting, each new task $\mathcal{T}_i$ drawn from $p(\mathcal{T})$ consists of frames in a single sequence, and the model is adapted to the task using a task-wise training set $\mathcal{D}_{\mathcal{T}_i}$, where training triplets are constructed only with frames existent in the low frame-rate input. Updating parameters at meta-training stage is governed by the loss $\mathcal{L}^{\text{out}}_{\mathcal{T}_i}$ for a task-wise test set $\mathcal{D}'_{\mathcal{T}_i}$, where the test triplets consist of two input frames and the target ground-truth intermediate frame that is non-existent in the low frame-rate input. In practice, we use 4 input frames $\{\mathbf{I}_1, \mathbf{I}_3, \mathbf{I}_5, \mathbf{I}_7\}$ as described in Sec. 3.2, and 1 target middle frame $\mathbf{I}_4$. The task-wise training and test set then become $\mathcal{D}_{\mathcal{T}_i} = \{(\mathbf{I}_1, \mathbf{I}_3, \mathbf{I}_5), (\mathbf{I}_3, \mathbf{I}_5, \mathbf{I}_7)\}$ and $\mathcal{D}'_{\mathcal{T}_i} = \{(\mathbf{I}_3, \mathbf{I}_4, \mathbf{I}_5)\}$. These configurations are illustrated in the left part of Fig. 3.

Given the above notations, we now describe the flow of our scene-adaptive frame interpolation algorithm in more detail. Since our method is model-agnostic due to integration with MAML, we can use any existing video frame interpolation model as a baseline. However, unlike MAML where the model parameters begin from random initialization, we initialize the model parameters from a pre-trained model that is already capable of generating sensible interpolations. Thus, our algorithm can also be viewed as a post-processing step, where the baseline model is updated to be readily adaptive to each test video for further performance boost.

The detailed flow of the algorithm is illustrated in the right part of Fig. 3. Let us denote the update iterations for each task as *inner loop* and the meta-update iterations as *outer loop*. For inner loop training, given two frame triplets from task-wise training set $\mathcal{D}_{\mathcal{T}_i}$ for each task $\mathcal{T}_i$, we first calculate the model predictions as

$$\hat{\mathbf{I}}_3 = f_\theta(\mathbf{I}_1, \mathbf{I}_5), \quad \hat{\mathbf{I}}_5 = f_\theta(\mathbf{I}_3, \mathbf{I}_7), \tag{3}$$

where the superscript i is hidden to reduce notation clutter. These outputs are then used to compute the loss for inner loop update $\mathcal{L}^{\text{in}}_{\mathcal{T}_i}(f_\theta)$, calculated as the sum of two losses as in

$$\mathcal{L}^{\text{in}}_{\mathcal{T}_i}(f_\theta) = \mathcal{L}_{\mathcal{T}_i}(\hat{\mathbf{I}}_3, \mathbf{I}_3) + \mathcal{L}_{\mathcal{T}_i}(\hat{\mathbf{I}}_5, \mathbf{I}_5). \tag{4}$$

Algorithm 1: Scene-Adaptive Frame Interpolation

Require: $p(\mathcal{T})$: uniform distribution over sequences
Require: α, β: step size hyper-parameters

Initialize parameters θ
while *not converged* **do**
  Sample batch of sequences $\mathcal{T}_i \sim p(\mathcal{T})$
  foreach *i* **do**
    Generate triplets $\mathcal{D}_{\mathcal{T}_i} = \{(\mathbf{I}_1, \mathbf{I}_3, \mathbf{I}_5), (\mathbf{I}_3, \mathbf{I}_5, \mathbf{I}_7)\}$ from $\mathcal{T}_i$
    Compute $\hat{\mathbf{I}}_3$, $\hat{\mathbf{I}}_5$ in Eq. (3)
    Evaluate $\nabla_\theta \mathcal{L}^{\text{in}}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{L}_{\mathcal{T}_i}$ in Eq. (4)
    Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}^{\text{in}}_{\mathcal{T}_i}(f_\theta)$
    Generate and save triplet $\mathcal{D}'_{\mathcal{T}_i} = \{(\mathbf{I}_3, \mathbf{I}_4, \mathbf{I}_5)\}$ from $\mathcal{T}_i$ for the meta-update
  end
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^{\text{out}}_{\mathcal{T}_i}(f_{\theta'_i})$ using each $\mathcal{D}'_{\mathcal{T}_i}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Eq. (5)
end

Next, we calculate the gradients for $\mathcal{L}^{\text{in}}_{\mathcal{T}_i}(f_\theta)$ and update θ with gradient descent to obtain customized parameters θ'_i for each task $\mathcal{T}_i$. Note that we can use any gradient-based optimizer (*e.g.* Adam [17]) for the updating step, and we choose the same optimization algorithm used to train the baseline pre-trained model in practice. Also note that the inner loop update can optionally consist of multiple iterations such that θ'_i is a result of k gradient updates from θ, where k is the number of iterations. We analyze the effect of hyperparameter k in Sec. 4.3, and choose $k = 1$ throughout our experiments for performance and simplicity (see Table 2). To further reduce computation, we employ a first-order approximation as suggested in [11] and avoid calculating the second-order derivatives required for the nested-loop updates in meta-training.

When training the outer loop, the parameters are updated to minimize the losses for $f_{\theta'_i}$ with respect to θ, on each of the task-wise test triplet $\{(\mathbf{I}_3, \mathbf{I}_4, \mathbf{I}_5)\} \in \mathcal{D}'_{\mathcal{T}_i}$. Loss function for the outer loop meta-update is defined as

$$\mathcal{L}^{\text{out}}_{\mathcal{T}_i}(f_{\theta'_i}) = \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}(\mathbf{I}_3, \mathbf{I}_5), \mathbf{I}_4), \tag{5}$$

and the summation of all losses for the sampled batch of sequences (tasks) $\mathcal{T}_i \sim p(\mathcal{T})$ are used to calculate the gradient and update the model parameters. The overall training process is summarized in Algorithm 1.

At test time, the base parameters θ for the outer loop are fixed, and only the inner loop update is performed to modify the parameter values to θ'_i for each test sequence $\mathcal{T}_i$. The final interpolations can then be obtained as the output of the adapted model $f_{\theta'_i}$.

Table 1. **Quantitative results for meta-training for recent frame interpolation algorithms**. We evaluate the benefits of our scene-adaptive algorithm on 3 datasets: VimeoSeptuplet [48], Middlebury-OTHERS [2], and HD [4] dataset. Performance is measured in PSNR (dB). Note how our *Meta-trained* performance consistently improves upon the *Baseline* or *Re-trained* correspondents.

	VimeoSeptuplet [48]			Middlebury-OTHERS [2]			HD [4]		
Method	Baseline	Re-trained	Meta-trained	Baseline	Re-trained	Meta-trained	Baseline	Re-trained	Meta-trained
DVF [20]	26.60	32.21	**32.27**	26.70	29.51	**29.70**	—	—	—
SuperSloMo [16]	30.85	32.76	**33.12**	30.28	33.54	**33.70**	26.05	29.66	**29.81**
SepConv [31]	33.70	33.72	**34.17**	35.14	34.90	**35.81**	30.04	30.01	**30.19**
DAIN [3]	34.73	34.86	**34.94**	**36.57**	36.50	36.50	30.35	30.45	**30.51**

Note that, the biggest difference from our algorithm from the original MAML is that the distributions for the task-wise training and test set, $\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$, are not the same. Namely, $\mathcal{D}_{\mathcal{T}_i}$ have a broader spectrum of motion and includes $\mathcal{D}'_{\mathcal{T}_i}$, since the time gap between the frame triplets are twice as large. Though this case with a distribution gap is an unexplored area in meta-learning literature, it shows an encouraging effect for the task of video frame interpolation; the model trained with our algorithm learns to update itself in considerably more difficult scenarios with larger motion, learning the overall context and motion present in the video as a result. Interpolations for the original input frames then become an easy task for our well-adapted model, which results in performance gain. Both quantitative and qualitative results in the experiments show that our algorithm actually improves the original model to better handle bigger motion.

4. Experiments

4.1. Settings

Datasets Most of the existing works on video frame interpolation use the video data pre-processed into frame triplets. Though our baseline model is pre-trained with conventional triplet datasets, it is not applicable for training the outer loop since multiple input frames are needed to construct the task-wise training samples for inner loop update. To this end, we use Vimeo90K-Septuplet (VimeoSeptuplet) dataset [48], which consists of 91,701 7-frame sequences with a fixed resolution of 448×256. Though this dataset is originally designed for video super-resolution or denoising / deblocking, it is also well suited for training video frame interpolation models that require multiple frames at test time, and we train all of our models with the training split of VimeoSeptuplet dataset. For evaluation, we use the test split of VimeoSeptuplet dataset, as well as sequences from Middlebury-OTHERS [2] and HD [4] dataset.

The OTHERS set from Middlebury contains 12 examples in total, with maximum resolution of 640×480. We use 10 sequences with multiple input frames and remove the other two that only have two input frames and are thus not suitable for test-time adaptation.

HD dataset proposed by Bao *et al*. [4] consists of relatively high-resolution frames, from 1280×544 to 1920×1080. Also, the length of the sequences in HD dataset is either 70 or 100, enabling test-time updates to our model.

Implementation details For our experiments, we use 4 conventional video frame interpolation models as baselines: DVF [20], SuperSloMo [16], SepConv [31], and DAIN [3]. We first initialize each model with pre-trained parameters, provided by the authors if possible.[1] We denote these models as *Baseline*. Then, since we use additional training set from VimeoSeptuplet for meta-training, we also fine-tune each *Baseline* models with VimeoSeptuplet training set, denoted as *Re-trained* models. For our final ***Meta-trained*** models, we start from the *Baseline* model parameters and follow the iterative steps for inner and outer loop training in Algorithm 1. The reported performance for *Meta-trained* models use a single inner loop update iteration at test time, and we examine the effects of increasing the number of gradient updates in the ablation study (Sec. 4.3).

We match the type of loss functions and optimization schemes for the gradient updates with the original methods used to train the *Baseline* models, which differs for each method. However, since we are fine-tuning from the pre-trained networks, we modify the inner/outer loop learning rates to be small and set $\alpha = \beta = 10^{-5}$. Throughout training, α is kept fixed, while β is decayed by a factor of 5 whenever validation loss does not decrease for more than 10,000 outer loop iterations. We do not crop patches and instead train with the full images of VimeoSeptuplet sequences with a mini-batch size of 4. While the number of training iterations differs for each interpolation model, the full meta-training step for any model requires less than a day with a single NVIDIA GTX 1080Ti GPU since we start from the baseline pre-trained network. The source code for our framework is made public[2] along with the pre-trained models to facilitate reproduction.

4.2. Video frame interpolation results

Quantitative results for all considered baseline frame interpolation models for all evaluated datasets are summarized in Table 1. For all experiments in this section, we standardize the evaluation metric to PSNR only. To check

[1] For SuperSloMo [16], we use the implementations and pre-trained models from [33].

[2] https://github.com/myungsub/meta-interpolation

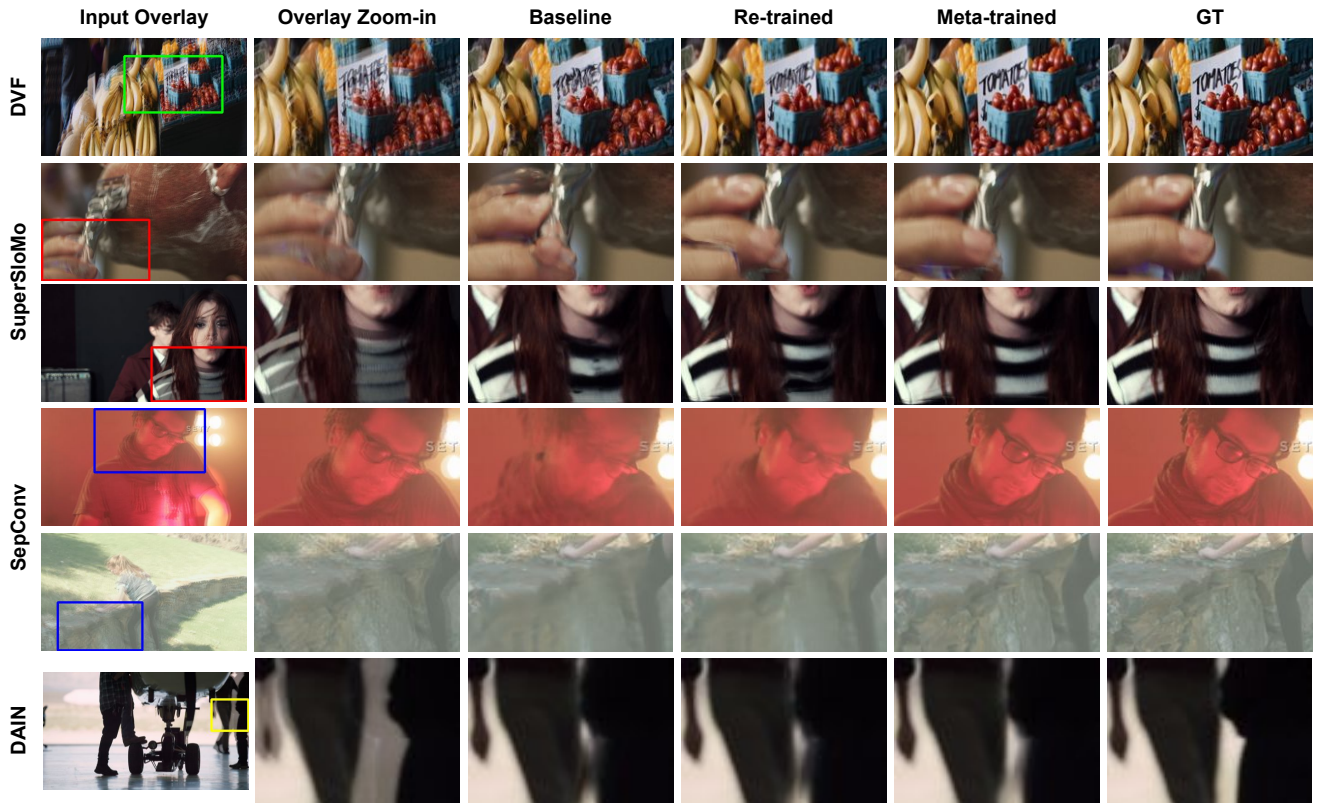

Figure 4. Qualitative results on VimeoSeptuplet [48] dataset for recent frame interpolation algorithms. Note how our *Meta-trained* outputs infer motion substantially better than the *Baseline* or *Re-trained* models, as well as generate realistic textures similar to the ground truth.

the results for other metrics such as interpolation error (IE) or structural similarity index (SSIM), we refer the readers to the supplementary materials.

In Table 1, note the consistent performance boost achieved by the *Meta-trained* model compared to both *Baseline* and *Re-trained* models, regardless of the method used for video frame interpolation. Also, even though meta-training for our scene-adaptive frame interpolation algorithm is only done in VimeoSeptuplet dataset, it generalizes well to the other datasets with different characteristics, presenting the benefits of test-time adaptiveness of our approach. Between two baselines, the *Re-trained* model generally performs better than the *Baseline* model. We believe this is due to the quality (*i.e.* degree of noise, artifacts, blurriness, *etc*.) of the training frames, since the frame sequences in VimeoSeptuplet are relatively clean. Since DVF is trained with videos from UCF-101 [42] dataset that has severe artifacts, its performance increase for fine-tuning to VimeoSeptuplet was the largest. The original training set, Adobe-240fps [43], for SuperSloMo [16] implementation also contains some degree of noise so that re-training helps to build a stronger baseline. An exception to this is SepConv [31], where re-training rather hurts the model's generalization capability to the other datasets. Nonetheless, our *Meta-trained* model considerably outperforms both baselines even for DAIN [3], the most recent state-of-the-art framework.

Qualitative results for VimeoSeptuplet dataset are shown in Fig. 4, where we compare the *Meta-trained* model with both *Baseline* and *Re-trained* models for each video frame interpolation algorithm. Note that our focus is on analyzing the benefits of *Meta-trained* models with its corresponding baselines, rather than comparison between different frame interpolation algorithms. For many cases where the baseline models fail due to large motion, our *Meta-trained* model adapts to the input sequence remarkably well to synthesize better texture and more precise position of the moving regions. In particular, the most notable improvements are shown for SepConv, which is the only model that does not utilize optical flow and the warping operation based on the predicted flow. Based on this evidence, we presume that explicit form of optical flow estimation constrains the possible performance gain obtainable by test-time adaptation. Additional qualitative results for HD dataset obtained with SepConv are presented in Fig. 5. Similar characteristics can be observed as in Fig. 4, and our *Meta-trained* model produces clearer interpolations with less artifacts. For more qualitative comparisons and the full video demos, please see the supplementary materials.

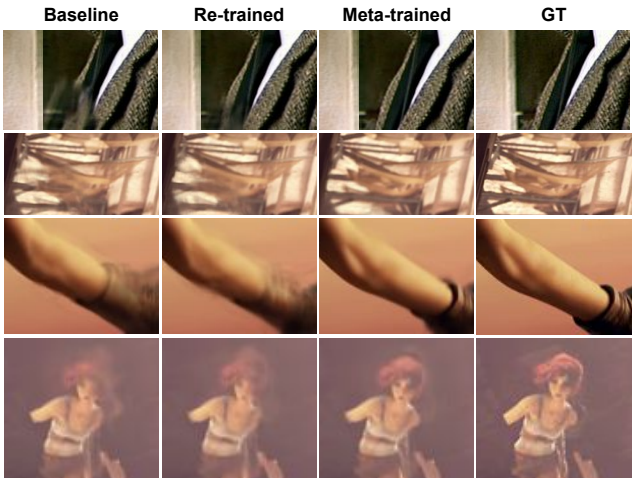

Figure 5. Qualitative results on HD [4] dataset for SepConv [31]. We show the cropped regions for *Shields*, *Alley2*, *Temple2*, and *Temple1* sequences.

Table 2. Effects on varying the the number of inner loop updates. Zero updates correspond to the *Re-trained* setting. PSNR (dB) for SepConv [31] is shown for Middlebury-OTHERS [2] dataset.

# gradient updates	0	1	2	3	5
Naive Fine-tune	34.90	34.90	34.95	34.99	35.03
Meta-trained	34.90	**35.81**	35.63	35.58	35.45
PSNR gain	—	**+0.91**	+0.68	+0.59	+0.42

4.3. Ablation studies

Effects on the number of inner loop updates We vary the number of iterations for test-time adaptation and analyze the effects. Table 2 demonstrates how the final performance changes while varying the number of inner loop updates from 1, 2, 3, and 5. We also show the results for naive test-time fine-tuning (from *Re-trained* model) along with our *Meta-trained* results, similar to the feasibility test in Sec. 3.2.

In summary, meta-training for just a single inner loop update, used in most of our experiment settings, shows the most PSNR gain, while increasing the number of updates did not have any benefits on performance. More updates even showed diminishing results, which is somewhat counter-intuitive compared to the tendency reported in MAML [11]. We believe there are two possible reasons for this phenomenon. First is overfitting to the data used for inner loop update ($\mathcal{D}_{\mathcal{T}_i}$). In Sec 3.2, we have shown that it is beneficial to use $\mathcal{D}_{\mathcal{T}_i}$ as a proxy for achieving good performance for $\mathcal{D}'_{\mathcal{T}_i}$ regardless of their distribution gap, but current ablation study suggests that *over*-fitting to $\mathcal{D}_{\mathcal{T}_i}$ can have negative effects on the final performance. This points out the need for finding the sweet spot in the trade-off between extracting from $\mathcal{D}_{\mathcal{T}_i}$ useful information that aids improving the interpolations in $\mathcal{D}'_{\mathcal{T}_i}$, and overfitting to $\mathcal{D}_{\mathcal{T}_i}$. For video frame interpolation, an example of common useful information can be the direction of existing motion or the details on background textures. If overfitting occurs, the inner loop may concentrate too much on handling the existing large motion and forget the generic prior knowledge learned by *Baseline* pre-trained model and its *Re-trained* version. Second reason is due to growing complexity of training as the number of gradient updates increase, which makes the model susceptible to falling into local minima [11, 28]. Presumably, incorporating recent techniques for adaptive learning rates [1] can help mitigate this issue, which remains as our future work.

Table 3. Effects on varying the learning rates for the inner loop updates. We use SepConv [31] framework for performance comparison on VimeoSeptuplet [48] dataset.

Learning rate α	0	10^{-6}	10^{-5}	10^{-4}
PSNR (dB)	33.72	34.10	**34.17**	34.15

Effects on inner loop learning rate Since our algorithm starts meta-training from a pre-trained video frame interpolation model, we believe that large learning rates for the inner loop update (α in Algorithm 1) can break the model's original performance at the early stage of training, while too small learning rates restrict the adaptive capability of the model. To support this claim, we report the performances on setting different values of α in Table 3 using SepConv. The final performance is maximized for the learning rate of 10^{-5}, with small gaps in PSNR compared to 10^{-4} or 10^{-6}. However, regardless of the values of α, the final performance is always better than when $\alpha = 0$, which demonstrates the effectiveness of our scene-adaptive frame interpolation algorithm via meta-learning.

5. Conclusion

In this paper, we introduced a novel method for video frame interpolation which aims to fully utilize the additional information available at test time. We employ a meta-learning algorithm to train the network that can quickly adapt its parameters according to the input frames for scene-adapted inference of intermediate frames. The proposed framework is applied to several existing frame interpolation networks and show consistently improved performance on multiple benchmark datasets, both quantitatively and qualitatively. Our scene-adaptive frame interpolation algorithm can be easily employed to any video frame interpolation network without changing its architecture or introducing any additional parameters.

Acknowledgements This work was supported by IITP grant funded by the Ministry of Science and ICT of Korea (No. 2017-0-01780), and Hyundai Motor Group through HMG-SNU AI Consortium fund (No. 5264-20190101).

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *ICLR*, 2019. 8

[2] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2010. 3, 6, 8

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *CVPR*, 2019. 2, 6, 7

[4] 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. *arXiv preprint arXiv:1810.08768*, 2018. 2, 6, 8

[5] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas, 1992. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1

[7] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, 2017. 1

[8] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time visual tracking based on target-specific feature space. *arXiv preprint arXiv:1712.09153*, 2017. 1

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

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017. 1

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 2, 3, 4, 5, 8

[12] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, 2009. 1, 2

[13] Sepp Hochreiter, A Younger, and Peter Conwell. Learning to learn using gradient descent. *Artificial Neural Networks, ICANN 2001*, pages 87–94, 2001. 2

[14] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 1, 2

[15] Jun-Jie Huang, Tianrui Liu, Pier Luigi Dragotti, and Tania Stathaki. Srhrf+: Self-example enhanced single image super-resolution using hierarchical random forests. In *CVPR Workshops*, 2017. 1, 2

[16] 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 *CVPR*, 2018. 1, 2, 6, 7

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 3, 5

[18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 3

[19] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang. Deep video frame interpolation using cyclic frame generation. In *AAAI*, 2019. 2

[20] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 1, 2, 6

[21] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *ECCV*, 2016. 2

[22] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *CVPR*, 2018. 2

[23] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *CVPR*, 2015. 2

[24] Tomer Michaeli and Michal Irani. Nonparametric blind super-resolution. In *ICCV*, 2013. 1, 2

[25] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017. 3

[26] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *ICML*, 2018. 3

[27] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1

[28] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. 3, 8

[29] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 1, 2

[30] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *CVPR*, 2017. 1, 2

[31] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 1, 2, 3, 6, 7, 8

[32] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NIPS*, 2018. 3

[33] Avinash Paliwal. Pytorch implementation of super slomo. https://github.com/avinashpaliwal/Super-SloMo, 2018. 6

[34] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 3

[35] Fitsum A Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised video interpolation using cycle consistency. In *ICCV*, 2019. 2

[36] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICLR*, 2016. 3

[37] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987. 2

[38] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. 2

[39] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *CVPR*, 2018. 1, 2

[40] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1

[41] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017. 3

[42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. 7

[43] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017. 7

[44] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 3

[45] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012. 2

[46] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 3

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

[48] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. In *CVPR*, 2018. 1, 2, 6, 7, 8

[49] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *CVPR*, 2018. 1

[50] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *ICCV*, 2017. 1