Multi-Scale Frame Interpolation

Kartik Gupta Sridhar Thiagarajan
Oregon State University
guptak@oregonstate.edu
thiagars@oregonstate.edu

Abstract

Video interpolation aims at generating a number of intermediate frames given two frames, based on the desired increase in frame rate of the video. Although most work in this field predict exactly one intermediate frame, recent work by Nvidia focus on generating variable number of frames. In this work, we take inspiration from the optical flow literature which use coarse-to-fine approaches to refine the optical flow, and integrate that with Nvidia's approach. We also propose to use a refinement GAN on the output of our system in order to generate more visually pleasing images. We conduct experiments and demonstrate results on the Adobe240 FPS dataset.

1. Introduction

The problem of video frame interpolation for generating slow-motion video has been extensively studied in the field of computer vision. Video interpolation can also be used to generate smooth transitions between two views. The task of interpolation has also been used as a task during self-supervised learning in order to learn optical flow between images [7] [6]. Although majority of the work focuses on generating a single intermediate frame, the recent work Super-Slomo by Nvidia [4] propose a novel methodology to generate multiple intermediate frames based on our need. They first compute bi-directional optical flow between the input images using a U-Net like architecture. A combination of these flows as an approximation to the true flow, however, only works well in locally smooth regions and produce artifacts around motion boundaries. To address this issue, they employ another UNet to refine the approximated flow and also predict something they term as 'visibility maps'. Finally, the two input images are warped and linearly fused to form each intermediate frame.

In traditional computer vision techniques tasks like object detection or optical flow estimation, things are often done in a multi-scale fashion. For example, in the seminal optical flow paper [10], they use a multi-level optical flow framework which estimate and combine optical flow at different resolution. As the video frame interpolation task is heavily reliant of optical flow, we posit that combining this approach along with Nvidia's superslomo work will give superior results. Although this multi-scale approach has been tried in a video interpolation setting, their work is limited to generating a single intermediate frame [12].

Generative adversarial networks [2] [3] are nowadays the go-to methods for image generation. They have also been used in the context of image refinmenet recently, mostly notably in the same related work which used the multi-scale loss for estimating optical flow. Traditional losses like the L1-loss often ignore small defects as it does not contribute to the loss term as much as more important large scale features. It is well known that adding an adversary solves this problem satisfactorily [5], and hence we feel adding a image-to-image GAN at the end of our network will result in superior image quality.

2. Related Work

Deep Optical Flow Estimation : Over the past few years, several works have cropped up in the field of optical flow estimation using deep learning [11] [1]. These methods rely on large datasets in order to learn to estimate optical flow between two input images. The field of video frame interpolation borrows a lot from literature in this area. Straightaway using optical flow in order to compute intermediate frames doesn't take into account any occlusion reasoning, and hence we need more intelligent methods.

Super-Slomo Video interpolation Nvidia's [4] is the most relevant work to our project. Let $F_{t\to 0}$ and $F_{t\to 1}$ denote the optical flow from I_t to I_0 and I_t to I_1, where I_t and I_0 denote the images at time t and time 0 respectively. After computing these two flow fields, they synthesize the intermediate image $\hat{I}_t$ by an adaptive combination of these fields and the images as follows

$$\hat{I}_t = \alpha_0 \odot g(I_0, F_{t\to 0}) + (1-\alpha_0) \odot g(I_1, F_{t\to 1}), \quad (1)$$

where $g(\cdot,\cdot)$ is a *backward warping* function, which is using bilinear interpolation and is differentiable. The parame-

ter α_0 controls the contribution of the two input images and depend on two factors: temporal consistency and occlusion reasoning. $\odot$ denotes element-wise multiplication, implying content-aware weighting of input images. For temporal consistency, the closer the time step $T = t$ is to $T = 0$, the more contribution I_0 makes to $\hat{I}_t$; a similar property holds for I_1. On the other hand, an important property of the video frame interpolation problem is that if a pixel p is visible at $T = t$, it is most likely *at least visible in one of the input images*,[1] which means the occlusion problem can be addressed. They also introduce a concept called *visibility maps* $V_{t\leftarrow 0}$ and $V_{t\leftarrow 1}$. $V_{t\leftarrow 0}(p) \in [0, 1]$ denotes whether the pixel p remains visible (0 is fully occluded) when moving from $T = 0$ to $T = t$. Combining the temporal consistency and occlusion reasoning, the output image is given as follows.

$$\hat{I}_t = \frac{1}{Z} \odot \big((1-t)V_{t\leftarrow 0} \odot g(I_0, F_{t\rightarrow 0}) + tV_{t\leftarrow 1} \odot g(I_1, F_{t\rightarrow 1})\big),$$

where $Z = (1 - t)V_{t\rightarrow 0} + tV_{t\rightarrow 1}$ is a normalization factor which is added to ensure brightness consistency.

Multi-Scale Flow estimation Multi-Scale Optimization has traditionally been performed using a standard incremental multi-resolution technique to estimate flow fields with large displacements. The optical flow estimated at a coarse level is used to warp the second image toward the first at the next finer level, and a flow increment is calculated between the first image and the warped second image. Most notably, this was used in [12] in order to do coarse-to-fine estimation of optical flow for image interpolation. We use a similar approach in our work, where the optical flow output at one resolution is upsampled using a bilinear upsampling, and then refined further at the higher scale. This technique has been used very commonly to handle large displacements.

Generative Adversarial Networks In the loss functions used in the super-slomo work, there is no mechanism to avoid solutions that may not be visually pleasing. A successful approach used in research to output realistic solutions can be to add adversarial losses. . It is well known that adversarial training results in images with photo-realistic properties such as improved sharpness and textures [5].

In GANs, there are two main components, the generator and the discriminator. The generator is the module which outputs an image, either from random noise or from a preconditioned image. It's goal is to output a realistic looking image.

The discriminator tries to output the probability of an image being fake or real, and it is trained to discriminate the generator's output from the real data. The loss for the discriminator is (write discriminator loss).

[1] It is a rare case but it may happen that an object appears and disappears between I_0 and I_1.

3. Methodology

3.1. Proposed Approach - Multi Scale, then GANs

Our approach, as discussed, is similar to Nvidia's work, but we do this at multiple-scales. We use three resolution scales - 0.25, 0.5 and 1. The complete architecture can be seen in figure 2. We propose to follow Nvidia's approach at the smallest resolution of 0.25 and have 2 UNet modules - Flow Estimate and Flow Refinement. The Flow Estimate is UNet model with 3 down and up stages. It takes as input the image frames I_0 and I_1 and outputs two optical flows $F_{1\rightarrow 0}$ and $F_{0\rightarrow 1}$. The flow refinement module has similar architecture, just with different input and output channels. It takes as input the I_0, I_1, $F_{t\rightarrow 0}$ and $F_{t\rightarrow 1}$ (interpolated from $F_{1\rightarrow 0}$ and $F_{0\rightarrow 1}$) ,*g(I_0,$F_{t\rightarrow 0}$)* and *g(I_1,$F_{t\rightarrow 1}$)*. g(I_1,$F_{t\rightarrow 1}$) denotes the image at time *t* generated using backward warp of image I_1 using $F_{t\rightarrow 1}$. It generates the refinement for flows $F_{t\rightarrow 0}$ and $F_{t\rightarrow 1}$ along with the visibility maps.All this is generated at a resolution of 0.25 times the full image resolution.

The next Flow Refinement performs similarly to the refinement module at 0.25 resolution, with a few differences. Firstly, this module generates the flow and visibility maps at a resolution of 0.5 times the full image resolution. Secondly, the input to this network does not come from a similar resolution flow estimate module; instead it is generated using bilinear interpolation of the outputs from the modules at resolution 0.25. Also, this module consists of 4 down and up stages in the UNet. Similarly,the next flow refinement module works to generate optical flows at full image resolution using upsampled outputs from the 0.5 resolution modules. This final network is a UNet model with 5 down and up stages.

We use the generator in order to refine the output of our super-slomo, after the entire super-slomo module is trained end-to-end. In our work, the generator's loss is a combination of an L1 loss and a discriminator loss (adverserial loss). This is to make sure that the content is generates is close to the true refined frame, and the adverserial loss ensures that the image is realistic looking. We use a traditional GAN proposed initially, and also attempted to get a Wasserstein GAN to work.

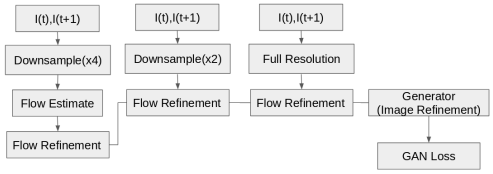

Figure 1: The complete network layout

3.2. Stage-Wise training

As there are several modules we are training, involving networks estimating flow at multiple scales, it would be difficult to training it completely end-to-end from the start. This might result in a vanishing gradient problem as the entire network as a whole is too deep.

Hence, we employ a stage-wise training process, where we first train the module which estimates the flow and interpolated images at a down-sampled resolution. We initially start with a resolution of 0.25 times the original image resolution. After this, we move to a resolution of 0.5, then to the full resolution. Finally, we train the entire network end-to-end in order to fine tune the weights.

We switch to a higher resolution after training a resolution one after a fixed number of epochs, chosen by pre-inspection. It would be interesting to explore better methods to do this, like the slope of the validation PSNR curve.

3.3. Dataset and preprocessing steps

To train our network, we use the 240-fps videos Adobe dataset taken with hand-held cameras [9]. We used a similar data generating process to Nvidia's work, except minor modifications where necessary. During training, all video clips are first divided into shorter ones with 12 frames in each and there is no overlap between any of two clips. For data augmentation, we randomly reverse the direction of entire sequence and select 9 consecutive frames for training just like their work. On the image level, each video frame is resized to have a shorter spatial dimension of 360 and a random crop of 352x352 plus horizontal flip are performed.

The main difference was that we needed training data at multiple scales, hence we used resizing using bilinear interpolation to get ground truth data at several resolutions.

4. Results

We base our implementation off an open-source implementation of the super-slomo work [8]. To evaluate the results we track the PSNR value. We train the network in multiple stages -
i) Train only the Flow Estimate Network(0.25x resolution) and Flow Refinement Network(0.25 resolution),
ii) Train only FLow Refinement Network(0.5x resolution),
iii) Train entire network end-to-end,
iv)Train only Flow Refinement Network (full resolution),
v)Train entire network end-to-end.

The complete training was performed for 350 epochs with 150,75,25,75 and 25 epochs at each stage respectively. We wished to perform more training at full resolution compared to lower stages. However, it proved to be computationally prohibitive. The learning was performed using a batch size of 8, once again constrained by our GPU memory. Results provided in Table 1. For qualitative results, please see Fig 3

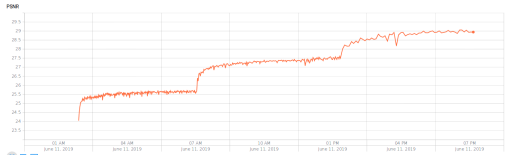

Figure 2: PSNR calculated on validation dataset during training. The steep jumps in the curve correspond to addition of the higher resolution module to the network.

Approach	SuperSloMo [4]	SuperSloMo (reproduction by [8]	ours
PSNR	31.1	29.8	29.5

Table 1: PSNR values calculated on the Adobe240fps dataset. Please note that the results provided by the authors are much higher compared to the results obtained using their evaluation script.

5. Conclusion and Future Work

In conclusion, we used a coarse-to-fine approach for optical flow estimation and combined this approach with Nvidia's super slomo work. We also propsoed to add a generative adversarial network to the output inorder to refine the image output, similar to prior work. We discuss the stage-wise training process we used in detail in order to train the network which estimates the flow at multiple scales.

As we could not get the image refinement GAN to work satisfactorily, a future avenue of work is to try more reliable GAN training algorithms [3] We attempted to get this working, but could not due to lack of time. Also, we used the original hyperparameters for balancing the loss components, and this might not be optimal for our case. We used this values as we did not have enough compute to do a thorough hyperparameter search. This may lead to a significant boost in performance in terms of PSNR. It would also be interesting to use different loss function, like the Charbonier loss, which have been shown to be more robust for optical flow estimation [10].

References

[1] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 1

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Gen-

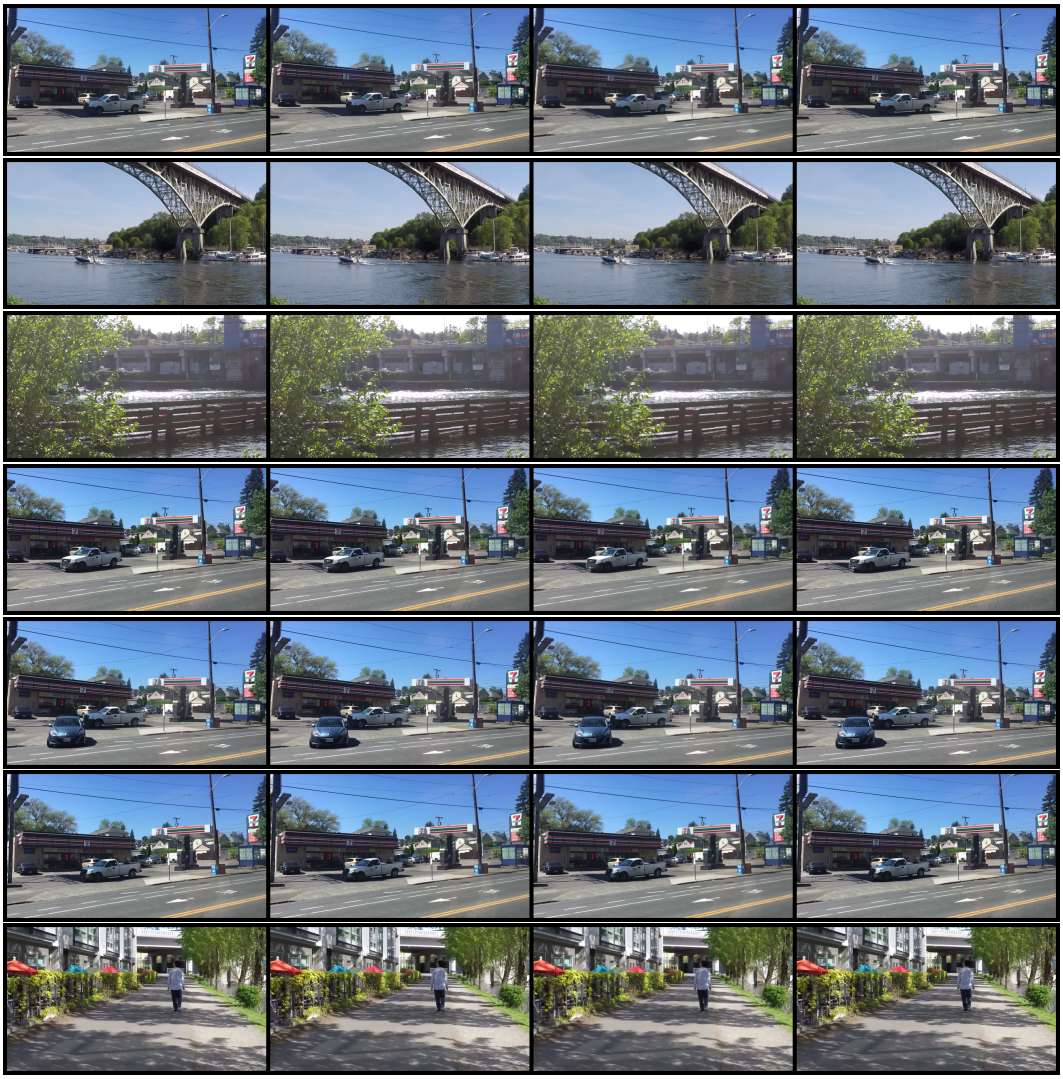

Figure 3: From left to right: 'Image at T=0, at T=t (ground truth), T=t(generated), T=1

erative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 1, 3

[4] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. 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. 1, 3

[5] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 1, 2

[6] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471, 2017. 1

[7] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016. 1

[8] A. Paliwal. Super slomo implementation. https://github.com/charlespwd/project-title, 2013. 3

[9] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring for hand-held cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1279–1288, 2017. 3

[10] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *2010 IEEE computer so-*

ciety conference on computer vision and pattern recognition, pages 2432–2439. IEEE, 2010. 1, 3

[11] D. Sun, X. Yang, M.-Y. Liu, and J. 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. 1

[12] J. van Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero. Frame interpolation with multi-scale deep loss functions and generative adversarial networks. *arXiv preprint arXiv:1711.06045*, 2017. 1, 2