Frame Interpolation with Residual and Depth-Wise Networks with AdaSepConv

Arvindh A R, Bukkala Varun, Doma Pratheek, Hitesh Goyal, Subramanian V
VIT Chennai, Tamil Nadu, India 600127

***Abstract*—In the world of video processing, generating non-existing frames from a sequential video sequence has been an intriguing and demanding subject. Kernel-based interpolation algorithms often anticipate pixels using a single convolution process that works over source frames with spatially adaptive local kernels, avoiding the time-consuming, explicit motion estimation in the form of optical flow. However, when the scene motion exceeds the predefined kernel size, these approaches are prone to producing less believable results. Furthermore, because the learnt kernels are bound to the midway in time between the input frames, they cannot directly create a frame at any arbitrary temporal position.**

The huge memory demand of these approaches limits the number of pixels whose kernels may be estimated at once. These methods need large kernels to handle large motion. This study develops local separable convolution over input frames using pairs of 1D kernels as a solution to this issue. The 1D kernels require a great deal less parameters to be estimated than conventional 2D kernels. With our approach, a deep full convolution neural network is created that takes two input frames and simultaneously calculates pairs of 1D kernels for all pixels.

Our approach trains the neural network to create frames since it estimates kernels and synthesises the entire video frame all at once. Without any human annotation, this deep neural network is trained from beginning to end using readily accessible video data. Qualitative and quantitative tests demonstrate that our approach offers a workable remedy for low-memory video frame interpolation.

1. Introduction

Frame interpolation is a well-known computer vision issue that is useful in applications such as innovative view interpolation and frame rate conversion. Traditional frame interpolation methods consist of two steps: motion estimation (often optical flow) and pixel synthesis. Optical flow is sometimes difficult to assess in areas with occlusion, haze, and rapid brightness changes. The occlusion problem cannot be properly handled by flow-based pixel generation. Failure to complete either of these two processes will result in visible artefacts in interpolated video frames.

Our method proposes a robust video frame interpolation method that uses a deep convolutional neural network to produce frame interpolation without explicitly splitting it into different phases. Using a deep fully convolutional neural network, we estimate the spatially-adaptive convolutional kernel by considering pixel interpolation as convolution over matching picture patches in the two input video frames.

The construction of pixel interpolation as convolution over pixel patches rather than depending on optical flow is a key component of our technique. Motion estimation and pixel synthesis are combined in this convolution formulation. It allows us to create a deep fully convolutional neural network for video frame interpolation without breaking it up into distinct parts. This formulation is also more versatile than those based on optical flow and can handle more difficult frame interpolation conditions.

2. Related Work

2.1. Video Frame Interpolation via Adaptive Separable Convolution

This study describes a viable approach to high-quality video frame interpolation. By calculating spatially-adaptive separable kernels for each output pixel and convolving input frames with them to create the intermediate frame, the current technique integrates motion estimation and frame synthesis into a single convolution process. The key to making this convolution method work is to mimic entire 2D kernels with 1D kernels. The use of 1D kernels decreases the amount of kernel parameters and enables full-frame synthesis, which permits the application of perceptual loss to improve the visual quality of the interpolation results. Here trials indicate that the approach both numerically and qualitatively compares favourably to state-of-the-art interpolation findings and generates high-quality frame interpolation outcomes [2].

2.2. Depth-Aware Video Frame Interpolation

This paper introduces a revolutionary depth-aware video frame interpolation technique that identifies occlusion explicitly using depth information. presented a depth aware flow projection layer that favours sampling of nearby items over distant ones Furthermore, they use the contextual information from the learnt hierarchical features and depth maps to synthesise the intermediate frame. Instead, then managing occlusion implicitly through estimating occlusion masks, they detect occlusion directly with depth information. They employed an hourglass network trained on the Megadeth Dataset to predict depth maps from input frames. The flow estimation, depth estimation, context extraction,

kernel estimation, and frame synthesis networks are part of the proposed model. Some of the suggested method's shortcomings. When depth maps are not accurately assessed, this approach produces hazy results with less defined borders. Complicated implementation with multiple components and a convoluted process that was tough to completely comprehend. The proposed model is small and effective. Extensive quantitative and qualitative assessments show that the proposed technique outperforms existing frame interpolation methods on a variety of datasets. The suggested method's state-of-the-art achievement illuminates future research on utilising the depth cue for video frame interpolation [3].

2.3. RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation

In this work, they provide RIFE, an efficient and flexible VFI method. A second neural module, IFNet, estimates the intermediate optical fluxes directly, supervised by a privileged distillation technique in which the instructor model has access to the ground truth intermediate frames. Experiments show that RIFE can properly process films of various scenarios. In addition, an additional input with temporal encoding allows RIFE to do arbitrary-timestep frame interpolation. Because of its lightweight nature, RIFE is significantly more accessible for downstream activities. RIFE employs a neural network known as IFNet, which can estimate intermediate flows from coarse to fine with much greater speed than many recent flow-based VFI methods, which first estimate bidirectional optical flows, then scale and reverse them to approximate intermediate flows, resulting in artefacts on motion boundaries and complex pipelines. To keep the model lightweight, unlike most prior optical flow models, this model (IF Blocks) does not include expensive operators such as cost volume and instead relies on 3x3 convolution and deconvolution as building blocks, which have been demonstrated to be more efficient [4].

3. Method

3.1. Dataset

Our models are trained using the Vimeo-90K training dataset. Vimeo-90K septuplets dataset contains 91701 septuplets (each septuplets is a short RGB video sequence that consists of 7 frames) from 39k video clips with fixed resolution 448x256. This dataset is designed to video denoising, deblocking, and super-resolution. All the videos are downloaded from vimeo.com.Video Enhancement with task oriented flow

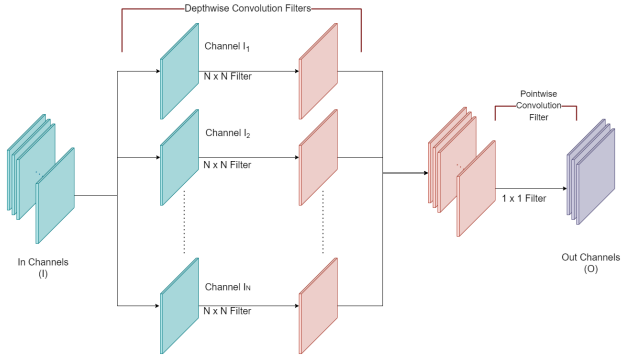

Diagram 1. Residual Block

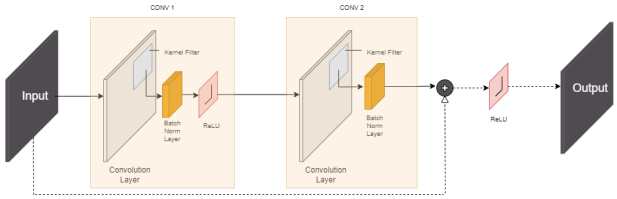

Diagram 2. Depth-Wise Block

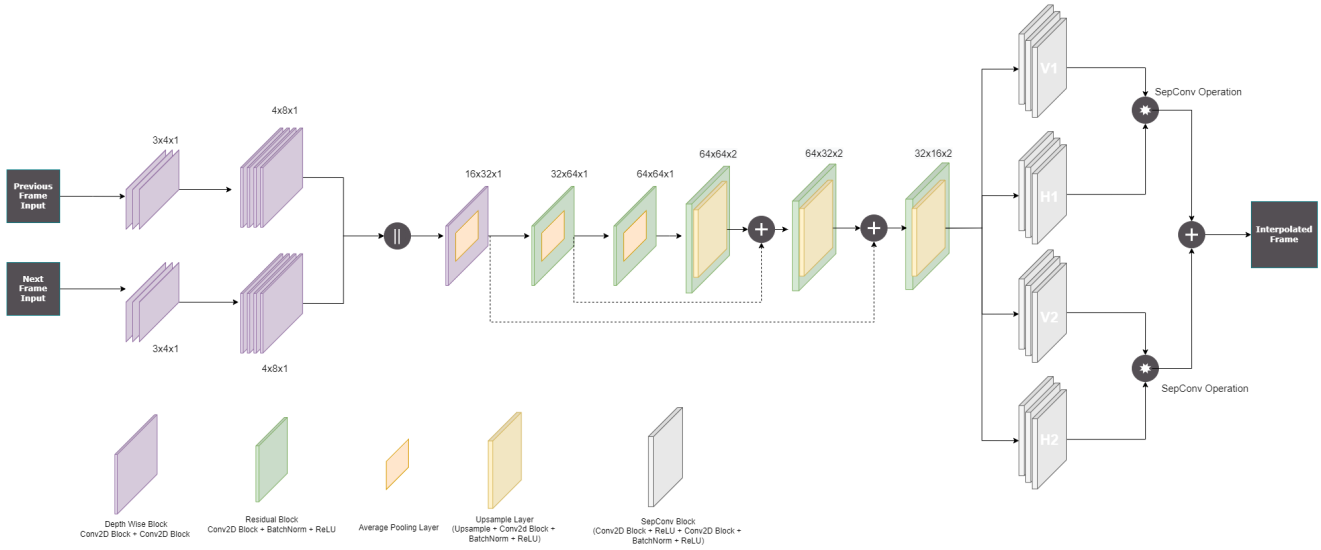

Diagram 3. Complete Architecture

3.2. Model Architectures

3.2.1. Residual Block. The ResidualBlock consists of two convolutional layers, each followed by batch normalisation, for an intermediate output. Addition operation is applied to this output and the 'residue', which is the initial input for this block, to produce its final output.

3.2.2. Depth-wise Block. The DepthWiseBlock consists of a depth-wise convolutional layer, where each operation by a filter channel is used only on one input channel with no change in number of output channels, and a point-wise convolution is applied to provide the block's result. This is done to increase the depth of the network without increasing the computation excessively.

3.2.3. AdaSepConv Block. This block is taken from the paper 'Video Frame Interpolation via Adaptive Separable Convolution' [2], where the separable convolution block takes an input and performs four different 1-d kernel operations, a horizontal and vertical strip on each input image. Each horizontal and vertical operation from each image is merged together, the resulting images merged again to form the final output to be generated by the model as prediction.

3.2.4. Downsample and Upsample. The Upsample block consists of a Pytorch Upsample layer by a factor of 2, followed by a convolution layer without a change of dimensions, and batch normalisation to produce this layer's output [5]. Down sampling is performed using an average pooling layer of 2 strides before passing it to a convolution layer. The process is repeated thrice until it is passed on to a series of up-sampling. Outputs of down sampling from different stages are merged with the final down-sample output to be passed on to respective stages of up sampling.

3.2.5. Final Architecture. Input images are processed for initial features using Depth-wise blocks. These processed images are concatenated to be passed through another combination of residual and depth-wise blocks. These features are then passed on to the down-sample and up-sample modules briefed in the previous section, and passed on to the AdaSepconv block to receive the final output.

3.3. Implementation

3.3.1. Training parameters. The training parameters for the neural network are their loss function and optimizer. The loss function used was Huber Loss [6] with delta value of 0.2 and other parameters as default. The optimizer used was Adamax [7] with a learning rate of 10^{-4} and other parameters as default.

3.3.2. Model training. The model was trained for approximately 21 hours, with a batch size of 16 and number of batches fixed at 64. It was trained on 12 GB Nvidia RTX3060 GPU with a 16 GB RAM and Intel i7 11th Gen processor. Each epoch is defined as 64 batches of 16 mini batches being trained once. Each epoch took approximately 95 ± 3 seconds to run.

3.4. Results

The table below shows the Loss and PSNR for Training and Testing

	Training	**Testing**
Loss	0.00120	0.00122
PSNR	28.0704	27.9536

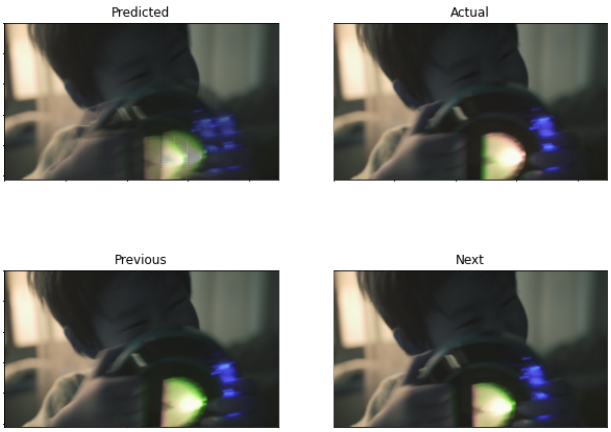

Figure 1. Result Image 1

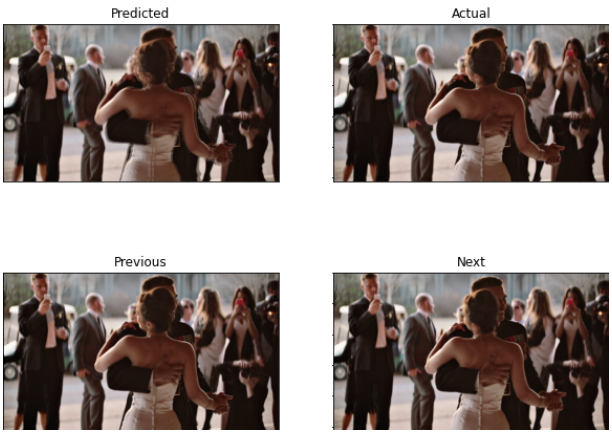

Figure 2. Result Image 2

4. Conclusion

Even though the output seems blurry in large movement, minor ones are predicted accurately. In case of videos, the output looks smoother than the original video making it useful for a low-memory use case. It may not be the most state-of-the-art result but for use cases like that of mobiles, it is one viable solution.

5. Future Work

Further work can be done to use loss functions like perceptual loss to get a sharper and clearer output. We can

try out deeper networks to see how they affect the accuracy or PSNR. We can use different loss functions which are more relevant to ensure a more direct relation between them and the image quality.

6. Acknowledgement

This work is supported by Samsung R&D Institute India-Bangalore (SRI-B) as part of Samsung PRISM (Preparing and Inspiring Student Minds) program. We would like to thank them for providing us with this opportunity.

We would also like to thank our Mentors, Ashwin Korwakar and Nitin Kumar Gautam along with our Professors, Dr. Maheshwari S and Dr. Leninisha Shanmugam for their help and support.

Finally we would like to thank our University, VIT Chennai, for providing us with all the resources without which we may never have completed this research.

7. References

[1] Xue, Tianfan, et al. "Video enhancement with task-oriented flow." International Journal of Computer Vision 127.8 (2019): 1106-1125.

[2] Niklaus, Simon, Long Mai, and Feng Liu. "Video frame interpolation via adaptive separable convolution." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[3] Bao, Wenbo, et al. "Depth-aware video frame interpolation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[4] Huang, Zhewei, et al. "Rife: Real-time intermediate flow estimation for video frame interpolation." arXiv preprint arXiv:2011.06294 (2020).

[5] Brownlee, Jason. "How to use the UpSampling2D and Conv2DTranspose Layers in Keras." Web blog post. machinelearningmastery. June 24, 2019.

[6] Rakhecha, Aditya. "Importance of Loss Function in Machine Learning." Web blog post. Towards Data Science. Sep 12, 2019.

[7] Brownlee, Jason. "Gradient Descent Optimization With AdaMax From Scratch." Web blog post. machinelearningmastery. June 7, 2021.