ReFIn: A Refinement Approach for Video Frame Interpolation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Video Frame Interpolation is an important video enhancement problem which aims to generate one or multiple frames between consecutive frames in video. Optical flow-based frame interpolation approaches estimate intermediate optical flow from interpolated frame to input frames and warped frames are fused to generate interpolated frame. However, intermediate flow estimates can itself be erroneous leading to inaccurate interpolation results. In this work, we improve an flow-based intrtpolation algorithm, Super-SloMo by residual refinement. Specifically, we feed intermediate flowmaps, visibility map, warped input frames and intermediate interpolation estimate to a refinement network to predict a frame residual. We have also experimented with different architecture choices to be used in different modules to further improve the results. We found out that GridNet with four pyramid levels achieves the best results whereas UNet++ performs moderately well with significantly less number of parameters.

1 Introduction

Video frame interpolation algorithms generate one or multiple frames between two consecutive frames in a video. These algorithms have applications in frame-rate upscaling, video compression-decompression [28], slow-motion video generation, novel view synthesis [7], medical imaging [15, 36] and so on. Optical flow based interpolation methods aim to first estimate intermediate flow i.e. flow between interpolated frame and source frames. Then the source frames are warped using the estimated intermediate flow. Finally, the warped frames are used to synthesize the interpolated result. Liu et al. [20] directly blend two warped frames. Niklaus et al. [23] feed the warped frames and warped context maps to generate interpolated frame. SuperSloMo [13] blend the warped frames using an estimated visibility map. In our work, we generate the intermediate frame in two steps. First, an estimate of intermediate frame is obtained through intermediate flowmap and visibility map estimation similar to [13]. Then, this estimate is further refined using a refinement network. A frame residual is estimated by the refinement network which is added with the intermediate frame estimate to generate the final interpolated frame. Our network performs better than multiple state-of-the-art methods on Vimeo-Triplet and UCF101 datasets.

2 Related Works

Early works in Video Frame Interpolation are often based on optical flow estimation and interpolation accuracy is used to compute quality of optical flow [1, 4]. The rise of optical flow estimation algorithms using Deep Learning has contributed a lot in the progress of Frame Interpolation algorithms [20, 13, 23, 33, 27, 9]. Kernel based frame interpolation algorithms [24, 25, 17] model each pixel in interpolated frame as linear combination of input patches convolved using spatially adaptive kernels.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Phase based frame interpolation algorithms [22, 21] perform phase decomposition of source images to generate intermediate frame. A few approaches [3, 2] in the literature tries to combine optical flow based and kernel based methods for frame interpolation.

3 Proposed Method

Given two consecutive frames I_0 and I_1 in a video, video interpolation algorithm predicts I_t where $t \in (0, 1)$. Our method is based upon a state-of-the-art Video Interpolation algorithm, SuperSloMo [13]. For the sake of completeness, we will describe the complete algorithm in this section. Our approach is summarized in Figure 1.

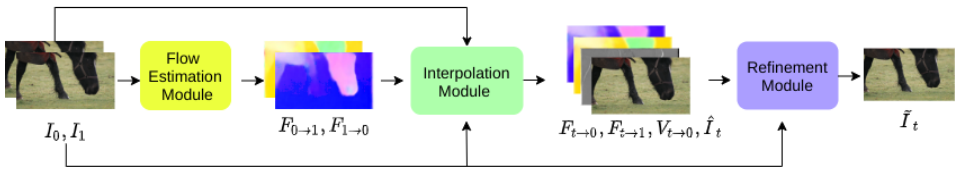


Figure 1: Overview of our refinement based frame interpolation approach.

Flow Estimation module: First step of our algorithm is to estimate bidirectional optical flowmaps between input frames. Given two input images I_0 and I_1, Flow Estimation module will compute bidirectional flowmaps $F_{0\to1}$ and $F_{1\to0}$. [13] estimate optical flow in unsupervised manner using a UNet. Instead of using UNet for flow estimation as in SuperSloMo, we use pretrained state-of-the-art flow estimator PWCNet [31].

Interpolation module: After computing bidirectional optical flows $F_{0\to1}$ and $F_{1\to0}$ using Flow Estimation module, we compute approximation of flow from intermediate image to input images, $\hat{F}_{t\to0}$ and $\hat{F}_{t\to1}$ using the following equations [13].

$$\begin{aligned} \hat{F}_{t\to0} &= -(1-t)tF_{0\to1} + t^2F_{1\to0} \\ \hat{F}_{t\to1} &= (1-t)^2F_{0\to1} - t(1-t)F_{1\to0} \end{aligned} \tag{1}$$

However, this approximation is not true in case of motion boundaries. Interpolation module acts as a correction step for computing final intermediate flowmaps. This module generates residual flowmaps to refine the intermediate flowmaps in Equation 1. Additionally, it computes soft visibility maps required for fusing the warped input frames. Interpolation network takes in input images I_0, I_1, approximate intermediate flows $\hat{F}_{t\to0}$ and $\hat{F}_{t\to1}$ and input images warped with $\hat{F}_{t\to0}$ and $\hat{F}_{t\to1}$ and predicts flow residuals $\triangle\hat{F}_{t\to0}$, $\triangle\hat{F}_{t\to1}$ and visibility map $V_{t\to0}$. Visibility map $V_{t\to1}$ is calculated from $V_{t\to0}$, since these two visibility maps should complement each other, i.e. $V_{t\to1} = 1 - V_{t\to0}$. So, the final estimation of intermediate flows are given by,

$$\begin{aligned} F_{t\to0} &= \hat{F}_{t\to0} + \triangle\hat{F}_{t\to0} \\ F_{t\to1} &= \hat{F}_{t\to1} + \triangle\hat{F}_{t\to1} \end{aligned} \tag{2}$$

We linearly blend the warped input images to synthesize predicted frame $\hat{I}_t$ as,

$$\hat{I}_t = \frac{(1-t)V_{t\to0} \odot bw\left(I_0, F_{t\to0}\right) + tV_{t\to1} \odot bw\left(I_1, F_{t\to1}\right)}{(1-t)V_{t\to0} + tV_{t\to1}} \tag{3}$$

where $bw(.,.)$ is the backward warping function [12].

Refinement Module: Although the formulation of intermediate frame in Equ-3 is compact, the synthesized frame can still have erroneous predictions. These errors can subject to large motion and occlusion. Hur et al. [11] has shown that iterative residual refinement of optical flow predictions can improve optical flow accuracy. Inspired by this, we introduce one step of refinement. We compute residual of the predicted frame $\triangle\hat{I}_t$ using a Refinement module. As we already have an approximation of the interpolated frame, we don't need to generate the image from scratch and compute the residual image only. Input images (I_0, I_1) , refined flow maps $(F_{t\to0}, F_{t\to1})$, visibility map $V_{t\to0}$, warped input images $bw(I_0, F_{t\to0}), bw(I_1, F_{t\to1})$ and predicted frame $\hat{I}_t$ are fed into this network. So, the final synthesized frame is given by,

$$\tilde{I}_t = \hat{I}_t + \triangle\hat{I}_t \tag{4}$$

Architectures used in Interpolation and Refinement modules: We use (a) UNet [29], (b) UNet++ [34], (c) GridNet [8, 23] architecture in interpolation and refinement modules. We try both 3-level and 4-level Gridnet in our experiments. We refer GridNet with 3 levels as GridNet-3 and GridNet with 4 levels as GridNet-4 for brevity. Details of these architectures are discussed in Appendix. Among the mentioned architectures, GridNet-4 achieves the best performance. Please refer to Section 4.1 for analysis.

4 Experiments

Datasets. We have used the Vimeo-Triplet [33] and UCF101 datasets [30, 20] in our experiments.

Training and Evaluation details. We use initial learning rate 10^{-4} for both Interpolation and Refinement modules. As we use pretrained PWCNet[1] as our Flow Estimation module, initial learning rate for this module is 10^{-6} and we keep it fixed throughout the training. We gradually decrease the learning rate to 10^{-6} for Interpolation and Refinement modules. We use Adam optimizer [16] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a batch size of 4.

We use train subset of Vimeo-Triplet dataset as our training dataset and evaluate on the test set of the same dataset. Videos in UCF101 dataset are of lower quality than Vimeo-Triplet sequences. Hence, for evaluation on test sequences of UCF, we fine-tune our model on our prepared UCF training data for one epoch. As we only predict the middle frame, t is set to 0.5 for both training and evaluation. We use Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) [32] metrics for evaluation on both the datasets.

Loss functions. Our loss function for stage-1 is given by,

$$\mathcal{L} = \lambda_r \mathcal{L}_r + \lambda_p \mathcal{L}_p + \lambda_w \mathcal{L}_w + \lambda_s \mathcal{L}_s \tag{5}$$

where, $\mathcal{L}_r$ is Reconstruction (L_1) loss, $\mathcal{L}_p$ is perceptual loss [14], $\mathcal{L}_w$ is Warping loss [13] and $\mathcal{L}_s$ is Smoothness loss [13]. In our experiments, we use $\lambda_r = 204$, $\lambda_p = 0.005$, $\lambda_w = 102$ and $\lambda_s = 1$. In stage-2 of training, we finetune the network using only reconstruction loss.

Comparison with other methods. We compare the performance of our algorithm with the following state-of-the-art models: TOFlow [33], SepConv [25], SuperSloMo [13], CtxSyn [23], CyclicGen [19], MEMC-Net [3], DAIN [2], CAIN [6], BMBC [27] and AdaCoF [18]. For comparison with SuperSloMo [13], we use unofficial pretrained models[2] since official pretrained models were not released. For rest of the models, official pretrained models were used for comparison. Table 1 summarizes the quantitative results of state-of-the art models. Our algorithm performs best among the mentioned methods on Vimeo-Triplet dataset and comparable against other state-of-the-art models in UCF101 dataset. Note that, we have gained 1.71 dB and 0.87 dB improvement on PSNR scores in Vimeo-Triplet and UCF101 datasets respectively over SuperSloMo, which we have considered as the base framework. We provide visual comparisons of interpolated images in Figure 2.

Method	Vimeo-Triplet		UCF101	
	PSNR	SSIM	PSNR	SSIM
TOFlow	33.73	0.9682	34.58	0.9667
SepConv	33.79	0.9702	34.78	0.9669
SuperSloMo	33.44	0.9673	34.09	0.9655
CtxSyn	34.39	0.9610	34.62	0.9490
CyclicGen	32.10	0.9490	35.11	0.9680
MEMC-Net	34.29	0.9739	34.96	0.9682
DAIN	34.71	0.9756	34.99	0.9683
CAIN	34.65	0.9730	34.91	0.9690
BMBC	35.01	0.9764	35.15	0.9689
AdaCoF	34.35	0.9714	35.16	0.9689
Ours (GridNet-4)	35.15	0.9773	34.96	0.9690

Table 1: Comparison with other state-of-the-art architectures in Vimeo-Triplet and UCF101 dataset.

Figure 2: Qualitative comparison with state-of-the-art models.

[1] https://github.com/sniklaus/pytorch-pwc

[2] https://github.com/avinashpaliwal/Super-SloMo

4.1 Ablation Study

Importance of Refinement Module: To show the significance of the Refinement Module, we train basic SuperSloMo Framework both with and without Refinement module. Please note, PWCNet is used as Flow Estimation module and UNet is used in Interpolation and Refinement modules in these experiments. Quantitative comparison is shown in Table 2. It can be seen in Table 2 that addition of refinement module improves the PSNR metric by 0.30 dB on Vimeo-Triplet dataset. Figure 3 shows significance of Refinement module qualitatively. We can see that rear light of the car is reconstructed when refinement module is used in Figure 3.

	PSNR	SSIM
W/o Refinement Module	34.27	0.9728
W/ Refinement Module	34.57	0.9744

Table 2: Results on Vimeo-Triplet dataset on Basic SuperSloMo Framework with and without Refinement module.

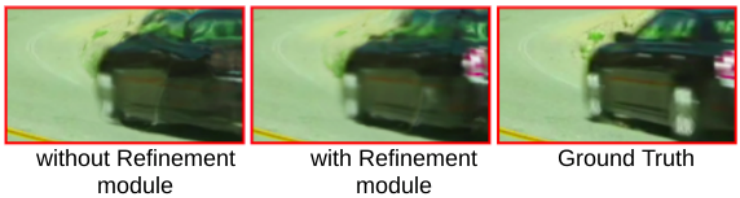


Figure 3: Effect of Refinement module.

Choice of architecture: We have experimented with four architectures in Interpolation and Refinement modules: UNet, UNet++, GridNet-3 and GridNet4. Quantitative results and number of parameters for each architecture is shown in Table 3. We observe that the two GridNet architectures perform better than UNet and UNet++ and GridNet-4 performs the best among the discussed architectures. It is worth mentioning that UNet++ gives good performance considering it has the least number of parameters among the said architectures. Qualitative comparison between the discussed architectures is shown in Figure 4. We can see that GridNet-4 as interpolation and refinement module produces superior results.

	# Parameters(M)		Performance	
	Interpolation Module	Refinement Module	PSNR	SSIM
UNet	19.8100	19.8157	34.57	0.9744
UNet++	**1.1764**	**1.1775**	34.83	0.9755
GridNet-3	2.2493	2.2497	34.96	0.9764
GridNet-4	5.0813	5.0817	**35.15**	**0.9773**

Table 3: Performance of different architectures in Interpolation and Refinement module on Vimeo-Triplet dataset.

Figure 4: Qualitative comparison between different architectures in Interpolation and Refinement modules.

5 Conclusion

In this work, we adopt a state-of-the-art Video Frame Interpolation framework SuperSloMo [13] and improve interpolation accuracy with the help of a refinement network. We show this refinement module can help in case of incorrect intermediate flow estimation. It can also help in enhancing edges and textures in the generated frame. We have explored a variety of network architectures to be used in Interpolation and Refinement modules and our experiments show that GridNet-4 performs better than the rest. On the other hand, UNet++ achieves good performance while being parameter-efficient.

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[2] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[3] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. 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] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[5] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[6] M. Choi, H. Kim, B. Han, N. Xu, and K. M. Lee. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10663–10671, 2020.

[7] J. Flynn, I. Neulander, J. Philbin, and N. 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.

[8] D. Fourure, R. Emonet, É. Fromont, D. Muselet, A. Trémeau, and C. Wolf. Residual conv-deconv grid network for semantic segmentation. *arXiv preprint arXiv:1707.07958*, 2017.

[9] S. Gui, C. Wang, Q. Chen, and D. Tao. Featureflow: Robust video interpolation via structure-to-texture generation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] J. Hur and S. Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019.

[12] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2017–2025, 2015.

[13] 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.

[14] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[15] A. Karargyris and N. Bourbakis. Three-dimensional reconstruction of the digestive wall in capsule endoscopy videos using elastic video interpolation. *IEEE transactions on Medical Imaging*, 30(4):957–971, 2010.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee. Learning spatial transform for video frame interpolation. *arXiv preprint arXiv:1907.10244*, 2019.

[18] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. 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.

[19] Y.-L. Liu, Y.-T. Liao, Y.-Y. Lin, and Y.-Y. Chuang. Deep video frame interpolation using cyclic frame generation. In *AAAI Conference on Artificial Intelligence*, 2019.

[20] 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.

[21] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 498–507, 2018.

[22] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1418, 2015.

[23] S. Niklaus and F. Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1710, 2018.

[24] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[25] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[26] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[27] J. Park, K. Ko, C. Lee, and C. S. Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *16th European Conference on Computer Vision, ECCV 2020*, pages 109–125.

Springer Science and Business Media Deutschland GmbH, 2020.
[28] R. Pourreza and T. S. Cohen. Extending neural p-frame codecs for b-frame coding. *arXiv preprint arXiv:2104.00531*, 2021.
[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
[30] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
[31] 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.
[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
[33] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.
[34] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11. Springer, 2018.
[35] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*, 2019.
[36] S. Zinger, D. Ruijters, L. Do, and P. H. de With. View interpolation for medical images on autostereoscopic displays. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(1):128–137, 2011.

A Architecture details

UNet: UNet was first proposed for medical image segmentation [29]. However, UNet has gained popularity in computer vision community and often used in image-to-image translation tasks. We use 23-layer encoder-decoder architecture used in SuperSloMo [13]. The encoder part consists of 12 convolutional layers and 5 average pooling layers. The decoder part consists of 10 convolutional layers and 5 bilinear upsampling layers. In encoder, first two convolutional layers have kernel size of 7×7. Next two layers in encoder use 5×5 kernels. Bigger kernels in initial layers is an important design choice to handle large motion. Kernels of size 3×3 are used in rest of the network. Skip connections (feature concatenation) are used from encoder to decoder for restoring the fine grained details in decoding part. Our UNet architecture is shown in Figure 5.

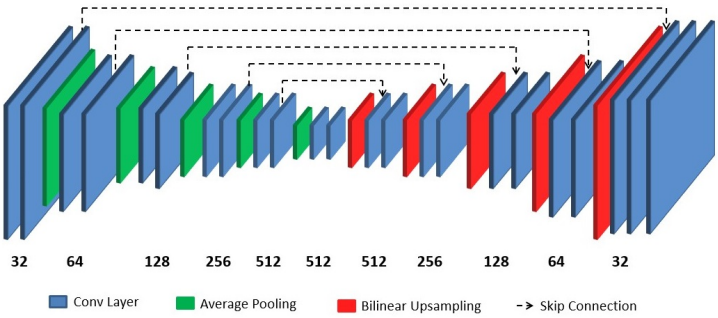


Figure 5: Model architecture of UNet. Number of output channels in convolutional layers are shown below.

UNet++: UNet++ is a Nested UNet architecture which was originally used for medical image segmentation [34, 35]. UNet++ aims to reduce the semantic gap between features from encoder branch and decoder branch using dense skip connections. We have used 4-level UNet++ architecture instead of 5 levels as used in original paper to reduce the model parameters and computational complexity. Maxpooling and bilinear upsampling is used for feature downsampling and upsampling respectively. Our UNet++ architecture has 4,3,2 and 1 Convolutional blocks from top to bottom levels respectively. Convolutional blocks except the output block has two convolutional layers with 3×3 kernels. Output block has only one convolutional layer with 1×1 kernel size. Output channel sizes in convolutional blocks from top pyramid level to bottom pyramid level are $32, 64, 96, 128$ respectively. Leaky ReLU is used after each convolutional layer. Model diagram of UNet++ is shown in Figure 6.

GridNet: GridNet is another encoder-decoder architecture where encoder and decoder blocks are laid out in a grid-like fashion which allows the network to combine features from different scales.

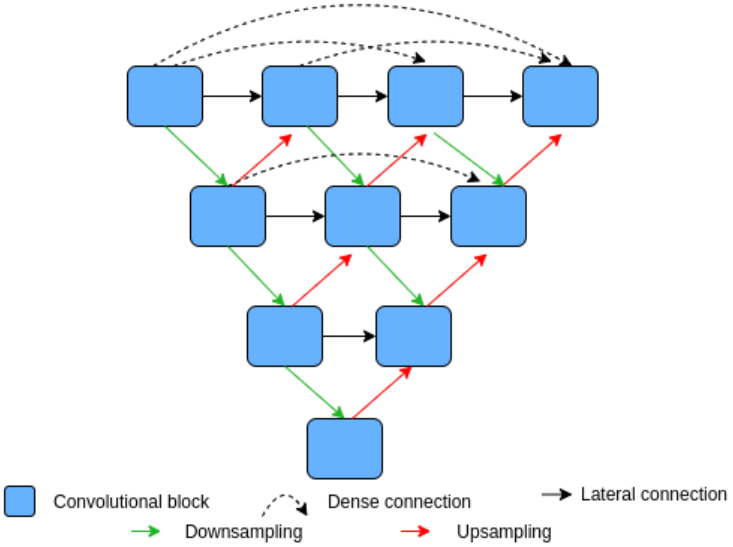


Figure 6: Model architecture of UNet++.

GridNet is a multi-scale architecture, where at each scale we have a number of lateral blocks. Number of output channels in this lateral blocks are consistent in same scale. Lateral blocks consists of convolutional layers with PReLU activation [10] and residual connection.

Features from lateral blocks on one level interact with features from other levels using upsampling and downsampling blocks and residual addition. Upsampling block consists of upscaling layer, convolutional layers and PReLU activation. Following [23], we use bilinear upsampling instead of using deconvolution in upsampling layer as Transposed convolution produces checkerboard artifacts [26]. Downsampling blocks use strided convolution as downsampling layer along with a convolutional layer and PReLU activation.

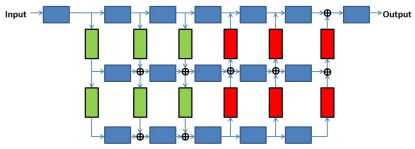
We try both 3-level and 4-level Gridnet in our experiments. We refer GridNet with 3 levels as GridNet-3 and GridNet with 4 levels as GridNet-4 for brevity. Output channel dimensions used in GridNet-3 are $32, 64, 96$ and GridNet-4 are $32, 64, 96, 128$ respectively from top to bottom. Architectures of GridNet-3, GridNet-4 and its basic components are shown in Figure 7.

B Edge and texture enhancement using Refinement module

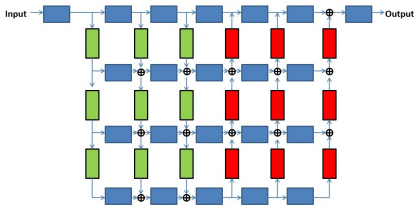
Refinement module can also help in better restoration of edges and textures. Figure 8 shows edge maps extracted by Canny edge detector [5] for interpolated frames generated when refinement module is used and when not used. We can observe that use of refinement module produces more clear edge and textures in interpolated image.

C Requirement for Stage-2 of training

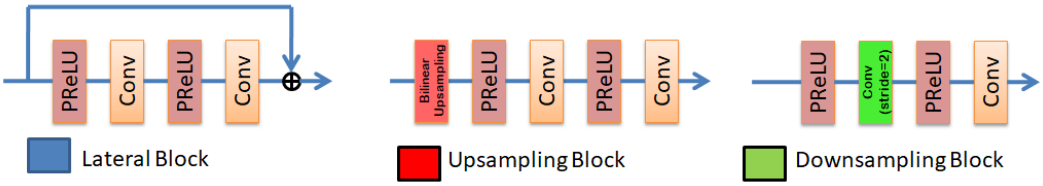
We train our model in two stages. In the first stage, we use Reconstruction loss, Perceptual loss, Warp loss and Smoothness loss. After the training is converged for first stage, we further fine-tune the whole model with just Reconstruction loss at a lower learning rate ($1e-6$ in our experiments). First stage of training gives us control over what is learnt by each module, whereas in the second stage, we can focus on only improving interpolation accuracy. The second stage of training helps to improve the performance as supported by our experiments in Table 4. PSNR is improved by 0.10 dB and 0.13 dB in case of without and with Refinement module respectively.



(a) GridNet-3

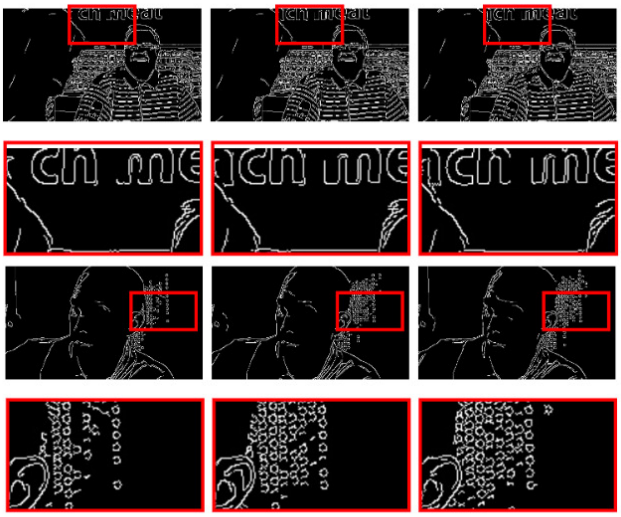


(b) GridNet-4



(c) Basic building blocks of GridNet

Figure 7: GridNet architectures used in our work and its constituting components.



Without Refinement module | With Refinement module | Ground Truth

Figure 8: Visualization of extracted edge maps for interpolated results with and without refinement module, along with ground truth edge map.

	Without Refinement Module		With Refinement Module	
	PSNR	SSIM	PSNR	SSIM
After Stage-1 training	34.17	0.9723	34.44	0.9737
After Stage-2 training	34.27	0.9728	34.57	0.9744

Table 4: Importance of Stage-2 of training.