DeepFovea++: Reconstruction and Super-Resolution for Natural Foveated Rendered Videos

Christoph Reich
TU Darmstadt
christoph.reich@stud.tu-darmstadt.de

Marius Memmel
TU Darmstadt
marius.memmel@stud.tu-darmstadt.de

Jonas Henry Grebe
TU Darmstadt
jonas.grebe@stud.tu-darmstadt.de

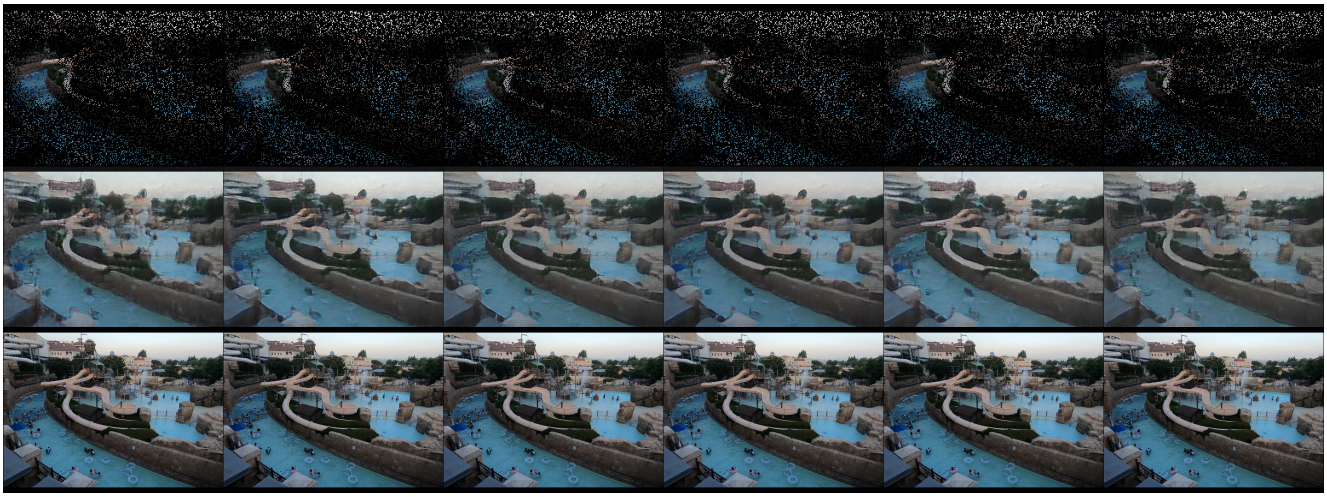

Figure 1. Results of our proposed DeepFovea++ (first setting) framework. The fovea sampled input sequences of low resolution (192×256) image frames can be seen on the top. The reconstructed super-resolution (768×1024) prediction sequence is shown in the middle, and the corresponding label at the bottom.

Abstract

Image super-resolution is a well-known problem in the computer vision community. Recent papers extended the problem of super-resolution to videos and showed amazing results. On the other hand deep learning based fovea sampled image reconstruction has drawn some popularity since the DeepFovea publication of Facebook AI. Even though DeepFovea showed outstanding results, the proposed reconstruction network was only able to reconstruct relatively low-resolution images of 128×128 *pixels. We revisit the proposed DeepFovea architecture to perform fovea sampled video reconstruction and super-resolution (*$192 \times 256 \rightarrow 768 \times 1024$*) at once. Our proposed architecture, DeepFovea++, first reconstructs a given video sequence by a recurrent U-Net architecture, and afterwards, the desired super-resolution is learned by deformable convolutions. We tested our DeepFovea++ architecture on the challenging REDS dataset. The code is available at* https://github.com/ChristophReich1996/DeepFoveaPP_for_Video_Reconstruction_and_Super_Resolution.

1. Introduction

A full immersion with virtual reality requires a very high image resolution, a low latency, and a high frame refresh rate [10]. Kaplanyan *et al*. [10] from Facebook Reality Labs approached this problem by making use of the fact that human perception has different levels of visual acuity across the retina. The closer regions in the field of view are to the point of maximum

focus - the fovea centralis - the more details from the incoming visual image are perceived. This makes many of the high-level details in image regions that are further away from the fovea point unnecessary, as they are not recognized by the human eye. Instead, the human brain infers this missing visual information based on experience it has made in the past.
The authors of DeepFovea [10] adopted this concept by dropping some of the peripheral pixels for the sake of reduced computational effort. They proposed a deep video reconstruction model that takes such foveated video frames and generates a plausible hypothesis for the missing peripheral pixels based on the remaining ones.
In this foveated rendering setup, recorded frames can be masked so that some of the pixels are dropped w.r.t. a probability that is higher the further away the respective pixel is from center of focus. This new masked frame requires less space. They were able to achieve a compression rate of up to 14x without a significant loss in perceived video quality.
However, as far as we know, the authors of the DeepFovea paper were working on relatively small images with a resolution of 128×128 pixels. This motivated us to go a step further and add a video super resolution extension to their basic DeepFovea architecture. We trained it in an end-to-end fashion and were able to generate upscaled and reconstructed output frames of a four times higher resolution.

2. Previous Work

2.1. DeepFovea

The DeepFovea [10] architecture by Kaplanyan *et al.* deals with foveated reconstruction i.e. the reconstructions of the most plausible peripheral video from a small portion of the pixels in each frame. This is done by finding the closest matching video. The input videos are foveated in stochastic manner to cover a wide variety. The network design is based on a U-Net i.e. a recurrent video encoder-decoder network with skip connections and is trained in an adversarial manner extending an adversarial loss with spectral normalization, perceptual spatial loss and an optical flow loss. To fit their constraint of being able to reconstruct the video in real-time on a head-mounted display, the network is only trained to reconstruct a 128×128 resolution. The dataset used for training was the YouTube-8M dataset [1] and contains a variety of natural content including people, animals, nature, *etc.*

2.2. Deformable Convolution

To overcome the geometric limitations of traditional 2d convolutions Dai *et al.* introduce deformable convolutions [6]. Instead of fixed sampling locations, the deformable convolution can learn an offset to the original sampling location. This is done by a separate traditional convolution. This improved operation showed performance benefits in multiple use-cases, like semantic segmentation or object detection. Dai *et al.* also published an advanced version, deformable convolution v2 [23]. This operation also learns a modulation mechanism, which further improved the performance of deformable convolutional neural networks. [6, 23]
The deformable convolution v2 can be expressed by the following formula:

$$Y(\boldsymbol{p}) = \sum_{k \in K} W_k \, X\left(\boldsymbol{p} + \boldsymbol{p}_k + \Delta \boldsymbol{p}_k\right) \Delta m_k. \qquad (1)$$

Where $\boldsymbol{p} \in \mathbb{R}^2$ is the current sampling location. Furthermore, $\boldsymbol{p}_k^{n \times n}$ represents the indexes of the $n \times n$ convolution kernel. The values $\Delta \boldsymbol{p}_k$ and Δm_k are the learnable offsets, obtained by a separate traditional convolution layer. To be able to sample a position $\boldsymbol{p} \in \mathbb{R}^2$ from a descried tensor bilinear interpolation us utilized. Finally, the resulting output feature for the position $\boldsymbol{p}$ is represented as $Y(\boldsymbol{p})$. [23]

2.3. Video Super-resolution

To achieve good results in video super-resolution, aligning the frames is of great importance. Tao *et al.* [18] introduce a 'sub-pixel motion compensation' layers and incorporate these into an end-to-end convolutional neural network (CNN) framework. Their approach yields an improvement in video super-resolution by '[fusing] multiple frames to reveal image details' 'without the need of parameter tuning'. [18]

Another approach regarding video super resolution is the attempt by Wang *et al.* [19] to further improve the aforementioned deformable convolutions towards solving a video restoration task. They propose enhanced deformable convolutions (EDVR) consisting of a pyramid, cascading and deformable alignment modules as well as a temporal and spatial attention fusion module. The former aligns the frames using deformable convolutions in a 'coarse-to-fine manner' and the latter introduces temporal and spatial attention for subsequent restoration. [19]

2.4. PWC-Net

PWC-Net [17] is a CNN model for optical flow estimation, which is widely used in cases where an optical

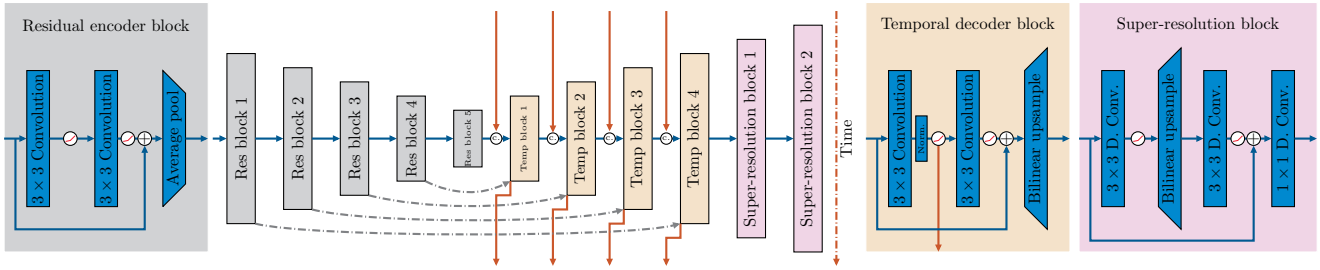

Figure 2. Reconstruction and super-resolution network architecture and the corresponding building blocks of the network in detail.

flow prediction is needed. It uses the established principles of pyramidal processing, warping, and the use of a cost volume. PWC-Net utilizes the current optical flow cast into a learnable feature pyramid to warp the CNN features of the second image. These features together with the the features of the first image are then used to construct a cost volume. A CNN is then used to estimate the optical flow on the basis of that volume. This pyramid approach outperformed more traditional encoder-decoder architectures like the famous FlowNet 2 [8] by Nvidia. [17]

2.5. General and Adaptive Robust Loss Function

The adaptive robust loss function by Barron [3] introduces robustness as a continuous parameter. In many deep learning use-cases, loss functions like the L1 or the mean squared error loss are utilized. However, choosing the right supervised loss function can be sometimes very difficult and computationally expensive. Barron introduced an adaptive robust loss function where the shape of the loss is a learnable parameter and thus not require manual tuning anymore. [3]

The general loss function Baron introduces, for the input $x = y - \hat{y}$, $x \in \mathbb{R}$, is defined as:

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{(\alpha/2)} - 1 \right). \quad (2)$$

Where $\alpha \in \mathbb{R}$ represents the shape parameter, which controls the robustness of the loss function. Furthermore, the parameter $c \in \mathbb{R}^+$ controls the size of the loss's quadratic bowl near $x = 0$. [3]
This general loss function can be parameterized to match existing loss functions like the mean squared error or the Huber loss. However, to be able to learn the parameters α and c the negative log-likelihood of the probability distribution that corresponds to the generalized loss function. This is needed because if the generalized loss function is optimized directly the loss function will end-up in a trivial solution. [3]
Learning the loss function shape showed improved performance in regression tasks like monocular depth estimation. [3]

3. DeepFovea++ Architecture

The DeepFovea++ reconstruction model is mainly based on two parts. First, a recurrent residual U-Net [15, 10, 22], and second, two super-resolution blocks based on deformable convolutions. We train both, the U-Net, and the super-resolution blocks, in an end-to-end setting. The whole reconstruction architecture consists of about 2.3M parameters. We used a relatively small reconstruction model to be able to fit the whole DeepFovea++ framework, consisting of the reconstruction model, a 3d discriminator, and a 3d FFT-discriminator, into GPU memory.

3.1. Reconstruction and Super-resolution Model

To reconstruct the fovea sampled input image sequence and achieve super-resolution we introduce a recurrent residual U-Net like architecture followed by two super-resolution blocks. Our architecture is highly inspired by the reconstruction network of the original DeepFovea paper. [10]
The recurrent residual U-Net uses five residual blocks in the encoding stage. Each residual block includes two 3×3 convolutions followed by an exponential linear unit (ELU) [5] activation function, respectively. The final operation of the residual block is an average pooling layer, which downscales the features. The residual mapping is realized by a 1×1 convolution, to deal with the changing number of feature dimensions.
In the encoding path of the recurrent residual U-Net, we utilize four temporal blocks. These include also two 3×3 convolution followed each by an ELU activation function. We introduce a layer normalization [2] operation after the first convolution and store the activated and normalized tensor as the recurrent state. This recurrent tensor is then concatenated in the next forward pass with the input of the corresponding tem-

poral block. This recurrent state can either be reset after each video sequence or be preserved. The recurrent state gets initialized with a random tensors sampled from $\mathcal{N}(0.0; 0.02)$. To upsample the output of the second convolution of the temporal block, we used a bi-linear upsampling operation. The residual mapping is also implemented in the form of a 1×1 convolution. To achieve the desired super-resolution ($4\times$) we utilize two super-resolution blocks. One super-resolution block consists of two 3×3 deformable convolutions, one bi-linear upsampling layer, two ELU activation functions, and a 1×1 deformable convolution. The residual mapping is utilized with standard 1×1 convolution followed by a bi-linear upsampling layer. We use deformable convolutions to be able to add more spatial details.
The whole network, as can be seen in figure 2, consists of 2.3M parameters. This is achieved by 32, 64, 128, 128, and 128 filters in each convolution in the corresponding encoding block. In the encoding blocks, we use 128, 128, 64, and 16 filters in the convolutions. The convolutions in the final super-resolution blocks consist of 8, and 18 filters respectively.

3.2. Discriminator Models

We utilize two discriminator networks. The first discriminator network receives the whole 3d video sequence. The second discriminator receives the 3d FFT spectrum of the video sequence, since natural images have characteristic statistics of a vanishing Fourier spectrum. [10]

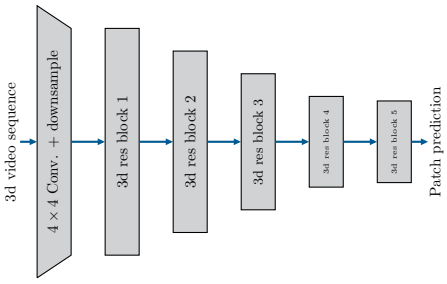

Figure 3. Architecture of the 3d discriminator network.

The 3d discriminator network, which can be seen in figure 3, has the same residual blocks as the reconstruction super-resolution network but uses a 3d convolutions instead of 2d convolutions. A schematic of the residual block can be seen in figure 4 (residual encoder block). In each operation, including the learnable parameters, we utilize spectral normalization [12] for a more stable adversarial training. To deal with the high-resolution input sequence, we introduce two downsampling operations before the 3d discriminator. First, we used a bi-linear downsampling layer, and second, we utilized a 4×4 convolution. The input sequence that is downscaled by a factor of four at each operation is concatenated and then fed into the 3d discriminator network. The final discriminator is a patch prediction as in [9].

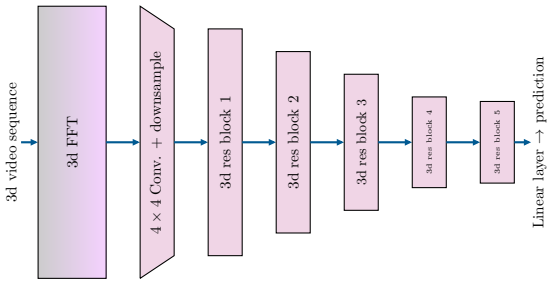

Figure 4. Architecture of the 3d FFT discriminator network.

The 3d FFT discriminator network consists of the same architecture as the 3d discriminator model. However, the input video sequence is transformed into a 3d spectrum. Furthermore, the final prediction is produced by a linear layer to achieve a scalar prediction, opposing the patch prediction of the 3d discriminator. [10]

3.3. Lossfunction

We train our reconstruction and super-resolution network on a weighted sum of multiple loss functions. The full loss function is defined as:

$$\mathcal{L} = w_{\text{sv}}\,\mathcal{L}_{\text{sv}} + w_{\text{flow}}\,\mathcal{L}_{\text{flow}} + w_{\text{adv}}\,\mathcal{L}_{\text{adv}} + w_{\text{adv fft}}\,\mathcal{L}_{\text{adv fft}} + w_{\text{flow}}\,\mathcal{L}_{\text{flow}} + w_{\text{LPIPS}}\,\mathcal{L}_{\text{LPIPS}} \quad (3)$$

For the supervised loss $\mathcal{L}_{\text{sv}}$ we used the general and adaptive robust loss function introduced in 2.5.[3]
The adversarial loss functions $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{adv fft}}$ represents the non-saturating generator GAN loss, which is defined as: [7]

$$\mathcal{L}_{\text{adv, adv fft}} = -\mathbb{E}\left[\log\left(\mathrm{D}\left(\hat{\mathbf{I}}\right)\right)\right]. \quad (4)$$

Where $\hat{\mathbf{I}}$ is the reconstructed super-resolution image and D the 3d discriminator or the 3d FFT discriminator. The discriminator networks are trained on the corresponding counter part to the non-saturating generator loss, which is defined as $-\mathbb{E}\left[\log\left(\mathrm{D}\left(\mathbf{I}\right)\right)\right] - \mathbb{E}\left[\log\left(1 - \mathrm{D}\left(\hat{\mathbf{I}}\right)\right)\right]$. Where $\mathbf{I}$ is the super-resolution ground truth label. [7]
Besides the supervised and adversarial losses, we also utilized the calibrated perceptual loss (LPIPS) [21] to achieve more natural-looking reconstructed images.

$$\mathcal{L}_{\text{LPIPS}} = \frac{1}{5}\sum_{i=1}^{5}\frac{1}{c_i\,h_i\,w_i}\left\|\text{VGG}_{i,2}\left(\hat{\mathbf{I}}\right) - \text{VGG}_{i,2}\left(\mathbf{I}\right)\right\|_1 \tag{5}$$

We take the outputs of the second convolution of the first five blocks of a pre-trained VGG-19 to compute the perceptual loss. This choice is inspired by the original DeepFovea framework. [16, 10]
To achieve consistency across frames we also utilized a flow loss $\mathcal{L}_{flow}$. This flow loss is defined as the L1 loss between one predicted frame and the corresponding backward wrapped subsequent frame. To be able to perform the backward wrapping operation we computed the optical flow between the two relevant frames by utilizing a pre-trained PWC-Net. [10, 17]

4. Experiments

We trained and tested our DeepFovea++ framework on the famous REalistic and Diverse Scenes (REDS) dataset, which was published in 2019. We analyze the performance of our framework with multiple metrics and discuss the occurring strengths and weaknesses.

4.1. REDS Dataset

In our experiments, we used the REDS dataset [13], which has been compiled in 2019 mainly for the tasks of video deblurring and video super-resolution. It totals 300 sequences of 100 high-quality RGB video frames each, with a resolution of 720×1280. These sequences are split into 240 videos for training, 30 for validation, and 30 for testing. However, only the training and test set are publicly available (as of July, 2020). In order to get input-output example pairs for supervised training, we followed the authors of the dataset by downscaling the target high-resolution images by a factor of 4 to obtain the unmasked low-resolution input frames. However, we used bi-linear instead of bi-cubic interpolation. [13]

4.2. Implementation Details

We implemented the whole DeepFovea++ framework in PyTorch 1.4.0 [14]. Our implementation is based on multiple existing implementations. First, we build our framework on top of the deformable convolution v2 [23] implementation included in the mmdetection toolbox [4]. Second, we used the correlation package [8] of Nvidia for implementing the optical flow loss. Furthermore, to estimate the optical flow for the corresponding optical flow loss, we utilized a pre-trained PWC-Net [17] by Nvidia research. Finally, our main supervised loss function is based on the adaptive robust loss function implementation by Jonathan T. Barron [3].

For optimizing all networks we utilized the Adam optimizer [11]. For the reconstruction model, we set the learning rate to 3×10^{-4}. The first and second-order running average factors were set 0.1 and 0.95, respectively. In both the 3d discriminator and the 3d FFT-discriminator we set the learning rate to 10^{-4}. The first and second-order running average factors were set to the same value as for the reconstruction model.
To produce the stochastic fovea mask we utilized the following formula.

$$p_{\boldsymbol{x}} = \begin{cases} 0.98 & \|\boldsymbol{x} - \boldsymbol{x}_{\text{f}}\|_2 < 20 \\ 0.15 & \|\boldsymbol{x} - \boldsymbol{x}_{\text{f}}\|_2 < 40 \\ -0.0415\|\boldsymbol{x} - \boldsymbol{x}_{\text{f}}\|_2 + 1.81 & \|\boldsymbol{x} - \boldsymbol{x}_{\text{f}}\|_2 \in [20, 40] \end{cases} \tag{6}$$

$p_{\boldsymbol{x}}$ represents the probability that the pixel of the image $\boldsymbol{I} \in \mathbb{R}^{3\times\text{height}\times\text{width}}$ at the given position $\boldsymbol{x} = [x_0, x_1] \in \mathbb{R}^2$ is not masked out. We mask out the image over all three RGB channels. The vector $\boldsymbol{x}_f = [x_{0,f}, x_{1,f}] \in \mathbb{R}^2$ represent the focus point of the fovea mask. This focus point is randomly generated by 2d uniform distribution which is defined as $\mathcal{U}\left([50, \text{height} - 50]\,; [50, \text{width} - 50]\right)$. To generate the final fovea mask we sample for each pixel from a uniform distribution $\mathcal{U}\,(0; 1)$ and threshold the resulting value with the corresponding $p_{\boldsymbol{x}}$ value to generate a binary mask. This binary mask is then applied to the input image.
The preprocessing in our asynchronous data loader contains five steps. In the first step, a batch of high-resolution images (720×1280) is loaded to construct a sequence of six frames. Then the images are downscaled by bi-linear interpolation to reach the resolution of 180×320 pixels. Next, both the inputs and the high-resolution labels are normalized to achieve pixel values between zero and one. Afterward, the fovea mask is generated and applied to the input image. In the final step, the fovea sampled images and the high-resolution label images are cropped and padded to a resolution of 192×256 and 768×1024, respectively. This is necessary to match the required input shape of the reconstruction network.
We set the weight factors of the loss function 3 to balance out the magnitudes between the different loss functions. The following weights were used: $w_{\text{sv}} = 5$, $w_{\text{adv}} = 0.01$, $w_{\text{adv ff}} = 0.01$, $w_{\text{flow}} = 2$ and $w_{\text{LPIPS}} = 1$.
We were able to fit one sequence consisting of six RGB frames on one Tesla V100 (16GB), at training time. We trained our whole framework for 15 epochs on two

GPUs with a batch size of two. The training process took us about two days.

4.3. Results

For analyzing our framework, results we calculated multiple metrics. First, we compute the common L1 metric and the Mean-Squared-Error (MSE, L2). In case of a reconstructed image $\hat{\mathbf{I}} \in \mathbb{R}^{c,h,w}$ and the corresponding label $\mathbf{I} \in \mathbb{R}^{c,h,w}$, the L1 and L2 loss is defined as

$$\mathrm{L1} = \frac{1}{c\,h\,w}\left\|\hat{\mathbf{I}} - \mathbf{I}\right\|_1 \tag{7}$$

$$\mathrm{L2} = \frac{1}{c\,h\,w}\left\|\hat{\mathbf{I}} - \mathbf{I}\right\|_2. \tag{8}$$

Additionally we compute the Peak-Signal-To-Noise (PSNR) and the Structural-Similarity-Image-Metric (SSIM) [20] to evaluate quality of prediction. The metrics are defined as follows:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\max\left\{\hat{\mathbf{I}}\right\}^2}{\mathrm{L2}\left(\hat{\mathbf{I}}, \mathbf{I}\right)}\right) \tag{9}$$

$$\mathrm{SSIM} = \frac{4\mathbb{E}\left[\hat{\mathbf{I}}\right]\mathbb{E}\left[\mathbf{I}\right]\mathrm{Cov}\left[\hat{\mathbf{I}}, \mathbf{I}\right]}{\left(\mathbb{E}\left[\hat{\mathbf{I}}\right]^2 + \mathbb{E}\left[\mathbf{I}\right]^2\right)\left(\mathrm{Var}\left[\hat{\mathbf{I}}\right] + \mathrm{Var}\left[\mathbf{I}\right]\right)}. \tag{10}$$

We tested our DeepFovea++ framework with two settings. In the first setting, we reset the recurrent state of the reconstruction model after each video. In the second setting, we preserve the recurrent state over the whole training and validation process. Our tests on the REDS dataset [13] led to the following results.

Rest re. st.	L1 ↓	L2 ↓	PSNR ↑	SSIM ↑
✓	0.0701	0.0117	22.6681	0.9116
✗	**0.061**	**0.009**	**23.8755**	**0.9290**

Table 1. Validation results of our DeepFovea++ framework on the REDS dataset.

From the results in table 1, it can be observed that resetting the recurrent state leads to a slight drop in performance compared to the setting where no reset is performed. We claim that the reconstruction model can generalize better if the recurrent state is preserved. However, if the recurrent state is not reset, a prediction can possibly depend on a different previous video. This fact could lead to unwanted side effects in predicting the first sequence of a new video.
Multiple validation sequences are visualized in the appendix 6. In the visual results, no significant performance difference between the two settings can be observed.

5. Conclusion

We propose an architecture to solve the novel problem of fovea sampled reconstruction and video super-resolution. According to our best knowledge, our framework is the first to tackle this problem. We think our framework can be seen as a baseline benchmark for future work. This claim is supported by the qualitative reconstruction results on the natural videos of the REDS dataset.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 2

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[3] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019. 3, 4, 5

[4] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. mmdetection. https://github.com/open-mmlab/mmdetection, 2018. 5

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 3

[6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 4

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 3, 5

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 4

[10] A. S. Kaplanyan, A. Sochenov, T. Leimkühler, M. Okunev, T. Goodall, and G. Rufo. Deepfovea: neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019. 1, 2, 3, 4, 5

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 4

[13] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. M. Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 5, 6

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019. 5

[15] 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. 3

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[17] 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. 2, 3, 5

[18] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4472–4480, 2017. 2

[19] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[20] 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. 6

[21] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 4

[22] Z. Zhang, Q. Liu, and Y. Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018. 3

[23] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 2, 5

6. Appendix

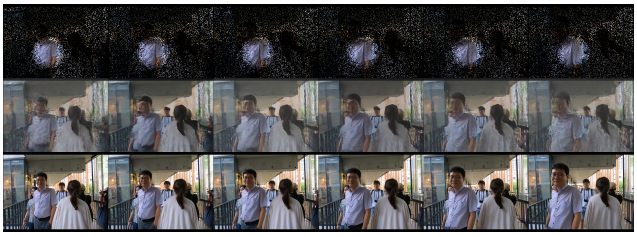

Figure 5. Results of the first DeepFovea++ setting (reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.

Figure 6. Results of the second DeepFovea++ setting (no reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.

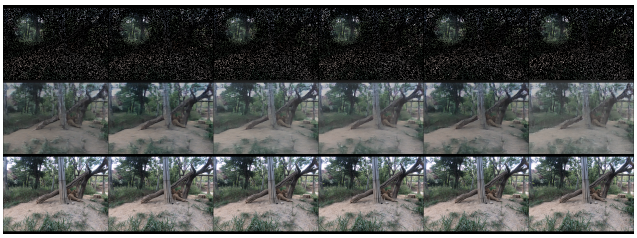

Figure 7. Results of the first DeepFovea++ setting (reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.

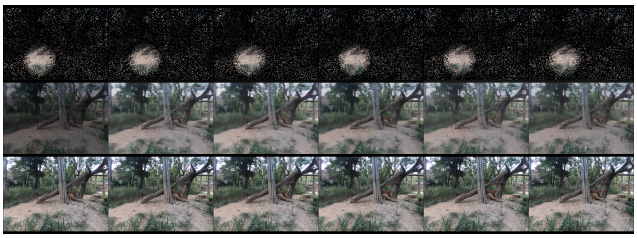

Figure 8. Results of the second DeepFovea++ setting (no reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.

Figure 9. Results of the first DeepFovea++ setting (reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.

Figure 10. Results of the second DeepFovea++ setting (no reset). The fovea sampled input sequences (192×256) on the top. The reconstructed super-resolution (768×1024) prediction sequence in the middle, and the corresponding label at the bottom.