FFmpegSR: A General Framework Toward Real-Time 4K Super-Resolution

Na Li
Rutgers University
na.li@rutgers.edu

Yao Liu
Rutgers University
yao.liu@rutgers.edu

***Abstract*—With the explosive growth of online video content, the demand for high-quality video is ever-rising. To take advantage of recent advances in deep learning, in this paper, we propose and implement a framework, FFmpegSR, that applies deep learning-based super-resolution into an FFmpeg filter to implement real-time 4K video super-resolution. FFmpegSR applies super-resolution to the Y channel only, allowing reduced inference time while maintaining good inference quality. To further improve the inference speed, we also develop a patch-based solution that uses saliency detection to select regions of interest on the video frame. This allows us to achieve faster inference on key patches only instead of full video frames. We used videos from a public dataset for evaluation. Results show that FFmpegSR can achieve real-time super-resolution to 4K with high visual quality.**

I. Introduction

With the rapid growth of video content consumption, nowadays it is crucial to deliver high-quality video to the users. However, due to constraints with network bandwidth, it can be infeasible for users to smoothly stream videos in high quality. Existing research works have proposed to employ an end-to-end video quality enhancement solution using deep neural network (DNN)-based super-resolution models, e.g., [1]–[3]. The main idea with these approaches is to pre-train super-resolution models on a per-video basis (or even finer granularity). During video streaming, users will download both a low resolution version of the video and a super-resolution model for enhancing the resolution and quality of the video frames before displaying them to the users. Most existing works focus on upscaling low resolution input videos. e.g., 240p, 260p, 720p, to output videos up to 1080p. However, this can be insufficient for users' demands for videos in resolution as high as 4K.

Targeting output video frames at 4K means that a significant amount of computation is required for per-frame super-resolution. However, despite advanced hardware resources, e.g., GPUs, that are available to users, super-resolution at such high resolution can still take a long time (e.g., over 400 ms) to upscale one single image. On the other hand, video streaming requires frames to be processed at a high throughput to meet real-time requirements. For example, the per-frame processing time should not exceed 40 ms for a video with a frame rate of 25 frames per second (fps).

In this paper, we design and implement FFmpegSR – a framework for upscaling high resolution videos with super-resolution model, leveraging the open-source FFmpeg project [4]. FFmpegSR can be configured to use any super-resolution model, e.g., SRCNN [5], FSRCNN [6], ESPCN [7], and EDSR [8]. To ensure real-time performance, however, it is preferable to use models with smaller sizes. To improve super-resolution inference speed, we take advantage of video saliency detection and implement patch-based video quality enhancement. Evaluation results show that FFmpegSR is able to achieve real-time video super-resolution at 4K. In addition, we compare our FFmpegSR framework with the Super-Resolution Filter (SRF) [9] implemented in the FFmpeg repository. Since SRF does not support super-resolution on patches, we compare the inference time performance of full-frame super-resolution only. Results also show that our proposed framework can achieve up to 1.8x speedup.

II. Design of FFmpegSR

Unlike many previous works, we aim to perform super-resolution on a per-frame basis. To this end, we first propose and implement a basic deep learning-based super-resolution framework, which we refer to as FFmpegSR-Basic. FFmpegSR-Basic integrates super-resolution models with the popular cross-platform video processing pipeline of FFmpeg. Motivated by the results obtained from FFmpegSR-basic, we then propose to improve the performance of real-time super-resolution in three aspects: channel reduction, model selection, and patch selection.

A. *FFmpegSR-Basic*

The gray-shaded boxes in Figure 1 represent components in the FFmpegSR-Basic framework. Since many existing super-resolution models are trained input data with R/G/B channels, FFmpegSR-Basic takes input videos in the GBRP pixel format. To perform super-resolution on each video frame decoded by FFmpeg, we implemented an FFmpeg video filter. Consider a pre-trained deep learning-based super-resolution model, e.g., ESPCN [7], we have to transform it to the serialized TorchScript model for use with FFmpeg. To do so, we used the `torch.jit.trace` function. We then used CMake and the PyTorch C++ API – LibTorch [10] to generate a shared library file, `libffmpegsr.so`, that is able to load and execute the TorchScript model. Output from the super-resolution model will then be written to the output of the video filter we implemented. Our FFmpegSR-Basic framework can

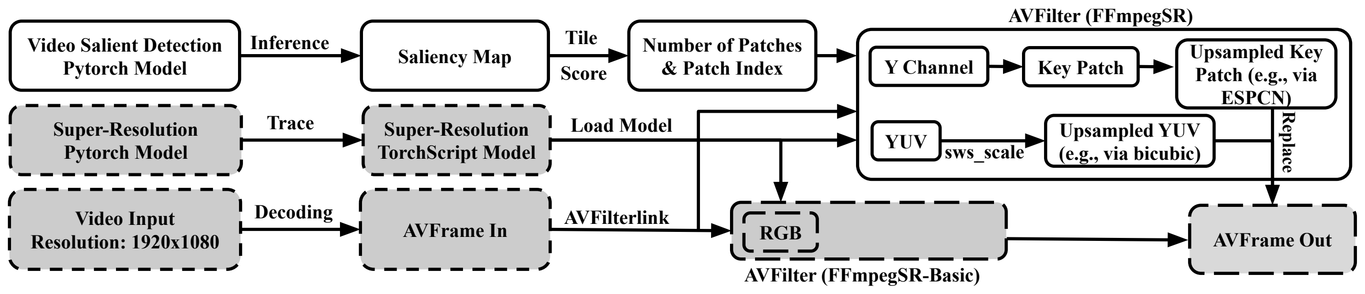
AVFilter (FFmpegSR)
Video Salient Detection Pytorch Model
Inference
Saliency Map
Tile Score
Number of Patches & Patch Index
Y Channel
Key Patch
Upsampled Key Patch (e.g., via ESPCN)
Super-Resolution Pytorch Model
Trace
Super-Resolution TorchScript Model
Load Model
YUV
sws_scale
Upsampled YUV (e.g., via bicubic)
Replace
Video Input Resolution: 1920x1080
Decoding
AVFrame In
AVFilterlink
RGB
AVFrame Out
AVFilter (FFmpegSR-Basic)

Fig. 1: Overview of the FFmpegSR framework. The gray-shaded boxes represent components in our basic deep learning-based super-resolution framework, FFmpegSR-Basic.

TABLE I: Performance of three deep learning-based super-resolution models. Here, model size is represented in bytes. The input videos are in 1920x1080 resolution (2K) at 25 frames-per-second (fps). x2 super-resolution is applied, and the output resolution is 3840x2160 (4K). "speed" is relative to the frame rate, i.e., 25 fps.

RGB inputs	# parameters	model_size	speed
FSRCNN	27,267	133,869	0.523x
ESPCN	26,796	119,254	0.636x
EDSR_baseline	1,369,859	5,605,671	0.093x

TABLE II: Performance of three deep learning-based super-resolution models when super-resolution is only performed on the Y-component of the YUV representation of decoded video frames. Model size is represented in bytes. "speed" is relative to the frame rate, i.e., 25 fps.

model	# parameters	model size	speed
FSRCNN	24,067	121,069	0.489x
ESPCN	21,284	97,238	1.060x
EDSR_baseline	1,367,553	5,596,455	0.103x

be configured to load any traced and serialized TorchScript model for performing per-frame super-resolution.

Naively applying super-resolution for each video frame using FFmpeg-Basic, however, takes a long time. Table I shows the performance of three deep learning-based super-resolution models on a desktop computer with Nvidia RTX 3080 Ti GPU. Note that the EDSR_baseline model has 16 resblocks with feature dimension of 64, while EDSR [8] has 64 resblocks with the feature dimension of 256. The input video of these experiments is the "RaceNight" video from the UVG Dataset [11] in RGB format. We preprocessed this video to 1920x1080 resolution (we also refer to this as 2K resolution throughout the paper) at 25 frames-per-second (fps). These models applied x2 super-resolution, and the resolution of output frames is 3840x2160. (We also refer to this as 4K resolution hereinafter.)

In this table, we compare different models by listing their number of parameters and traced model sizes. For performance metrics, we compare each model's super-resolution "speed", represented as the overall speed recorded by the FFmpeg pipeline against the frame rate (e.g., 25 fps).

Results show that even with ESPCN which has the least number of parameters and the smallest model size, it still takes more than 50 ms to process a video frame. These results indicate that even with a powerful GPU as we used in this experiment, it is infeasible to achieve real-time enhancement of video frames to 4K quality by naively using super-resolution models and settings. Next, we describe three approaches for improving the real-time performance of deep learning-based video frame super-resolution.

B. Channel Reduction

Most deep learning-based super-resolution models use the RGB format. However, video frames are typically converted to the YUV format, e.g., yuv420p, before being encoded by video codecs such as H.264. In addition, human eyes are most sensitive to the luminance component, which indicates that the Y channel is the most important component. This led us to save the computation by performing super-resolution on the Y channel only.

Using the same input video (1920x1980 input resolution, 25 fps) as Table I, Table II shows the model performances when super-resolution is only performed for the Y channel of the YUV representation of the decoded video frame. With reduced input channel, the number of parameters and model size of all models have slightly reduced compared to Table I. The performance of ESPCN substantially improved after channel reduction: the super-resolution speed can now reach 1.06x, faster than the video frame rate. For EDSR_baseline, however, operating on the Y-channel as opposed to 3 channels only improved the speed from 0.093x to 0.103x. This is because its performance constraints are the number of resblocks and the high dimension of feature space. To meet real-time super-resolution demands, we need to further adapt its model architecture.

C. Model Selection

The EDSR model has a large number of parameters. To further improve its real-time performance, we reduce its number of resblocks and feature dimensions and evaluate its super-resolution quality. We chose the following settings: `r2, r4, r8`, selecting the number of resblocks to be 2, 4, and 8, respectively, and `f8, f16`, selecting the feature dimensions to be 8 and 16, respectively. Following our finding

TABLE III: This table compares the performance of different models. Here, `r2, r4, r8` represents the number of resblocks in the adapted EDSR model is 2, 4, and 8, respectively, and `f8, f16` represents the feature dimension in the adapted EDSR model is 8 and 16, respectively.

model	# of params	size (bytes)	PSNR	SSIM	full frame	patch only
ESPCN	21,284	97,383	43.86	0.9788	1.09x	2.05x
EDSRr2f8	5,409	48,753	44.42	0.9791	1.01x	2.06x
EDSRr4f8	7,745	71,793	44.35	0.9790	0.92x	1.92x
EDSRr4f16	30,465	163,207	44.62	0.9806	0.80x	1.87x
EDSRr8f16	49,025	267,211	44.80	0.9808	0.61x	1.67x

with "channel reduction", we configure these models to take Y channel input only.

Table III compares the number of parameters, storage sizes in bytes, PSNR, structural similarity (SSIM) [12], and inference time results of these 4 EDSR-based models. Note that we also include ESPCN results in the table for comparison. The models evaluated in this table are trained using the testing video itself ("RaceNight") for 10 epochs. The visual quality results (PSNR and SSIM) presented in this table are obtained by comparing the deep learning-based super-resolution output (full frame) with the groundtruth video frame in 4K.

For EDSR models with different configurations, it is not surprising that the super-resolution quality generally increases with the increase of resblocks and feature dimensions, and the inference time increases with the number of resblocks and feature dimensions. When performing full frame super-resolution from 2K to 4K (i.e., 1920x1080 to 3840x2160), only ESPCN and EDSRr2f8 can achieve more than 1x speed, meeting the real-time requirement.

D. Patch Selection

Only 2 out of 5 models in Table III can meet the real-time requirements when performing 2x super-resolution on 2K frames. To further improve the inference speed, we propose to enhance the quality via super-resolution of "key patches" on the frame only, instead of the full frame.

To identify key patches in each frame, in this work, we propose to use video salient object detection (VSOD [13]). We carefully selected a state-of-art real-time video saliency detection model, STVS [14]. STVS proposes a novel spatio-temporal network with an extremely lightweight temporal unit. It can achieve high-quality video salient object detection at 50 fps in real-time applications and can be directly included in our framework.

Based on saliency maps generated by STVS, we tile each saliency map into 6x6 patches. For a 1920x1080 input frame, each patch is 320x180. For each patch, we calculate its saliency score and normalize it into $[0, 1]$. We then use a threshold to identify and select key patches in a frame. Here, we set the threshold as 0.5.

Inference time improvements of using key patches are shown in Table III. When super-resolution is only performed on selected key patches, all 5 models can achieve real-time inference.

III. FFmpegSR Implementation

Figure 1 shows an overview of our FFmpegSR framework that is fully integrated with the FFmpeg video processing pipeline. With a pre-trained super-resolution model, we can trace it to a TorchScript model for use in FFmpeg. Meanwhile, we set up the state-of-the-art pre-trained video salient object detection model for real-time saliency detection. It allows us to obtain the saliency map for each frame in the video. We spatially divide each frame into 6x6 patches. Based on the saliency map, we can obtain information about key patches for each frame. That is, the number of key patches and their corresponding patch indices for each frame in a video. With this information, we can achieve patch-based video quality enhancement in the filter, which further saves computation and inference time.

Within the FFmpegSR video filter, we extract the Y channel information from each decoded frame, obtain information about each frame's key patches, and apply deep learning-based super-resolution model to these key patches. In the meantime, we also directly apply bicubic interpolation, supported by the FFmpeg `libswscale` library, to upsample the full decoded YUV frame according to the scaling factor (e.g., x2). Key patches returned from super-resolution models are then used to replace bicubic-upsampled Y channel data. That is to say, we perform deep learning-based super-resolution on key patches in the Y channel, while the remaining patches in the Y channel as well as data in the U and V channels are upsampled using bicubic interpolation. In this way, fewer (but highly salient) areas of the full video frame are processed by the super-resolution model, thereby improving the inference time performance.

IV. Evaluation

A. Dataset Selection

For evaluation, we used videos from the UVG dataset [11]. This dataset includes 16 4K (3840x2160) test video sequences captured at 50/120 fps, 5 seconds and 12 seconds long. In our experiments, we used videos that are 50 fps and 12 seconds long. Additionally, we changed the fps of these videos from 50 to 25. Among the 7 videos that are 12 seconds long, we selected 4 most representative videos with moving scenes to evaluate our framework: RaceNight (`v1`) with fast and plenty of motion, Twilight (`v2`) with slow and plenty of motion, Flowerkids (`v3`) with slow and plenty of motion, and Riverbank (`v4`) with slow and little motion.

B. Inference Time Comparison

We evaluate our framework on all 4 videos on 2 different GPUs (3080 Ti and 2080 Ti). We consider two input frame resolutions, 1K (960x540) and 2K (1920x1080), and a scaling factor of x2. That is, super-resolution is used for transforming frames from 1K to 2K and from 2K to 4K. Results are shown in Table IV. Based on video salient object detection results, overall, we have 1481, 2291, 729, and 470 key patches for RaceNight, FlowerKids, Twilight, and Riverbank, respectively, which corresponds to the amount of motion in these videos. In

TABLE IV: Evaluation results based on 4 videos, 3 models, 2 scaling resolutions, and 2 GPUs.

video	model	reso-lution	speed on 3080Ti	speed on 2080Ti	PSNR	SSIM
v1	ESPCN	1K-2K	3.34x	2.89x	38.23	0.970
		2K-4K	2.12x	1.68x	38.25	0.962
	EDSRr2f8	1K-2K	3.22x	2.91x	38.99	0.971
		2K-4K	2.04x	1.64x	38.58	0.962
	EDSRr4f16	1K-2K	3.17x	2.67x	40.26	0.976
		2K-4K	1.92x	1.39x	38.86	0.965
v2	ESPCN	1K-2K	3.41x	3.11x	49.08	0.992
		2K-4K	2.25x	1.76x	46.78	0.987
	EDSRr2f8	1K-2K	2.21x	3.04x	47.81	0.990
		2K-4K	3.30x	1.85x	46.23	0.987
	EDSRr4f16	1K-2K	2.13x	2.97x	52.02	0.992
		2K-4K	3.28x	1.63x	48.54	0.989
v3	ESPCN	1K-2K	3.45x	3.13x	33.12	0.861
		2K-4K	2.37x	1.86x	34.72	0.898
	EDSRr2f8	1K-2K	3.42x	3.16x	33.57	0.847
		2K-4K	2.27x	1.85x	35.25	0.888
	EDSRr4f16	1K-2K	3.26x	2.97x	35.39	0.892
		2K-4K	2.27x	1.75x	36.77	0.917
v4	ESPCN	1K-2K	3.24x	2.78x	40.02	0.975
		2K-4K	1.93x	1.51x	41.24	0.979
	EDSRr2f8	1K-2K	3.09x	2.88x	44.01	0.979
		2K-4K	1.82x	1.50x	44.76	0.973
	EDSRr4f16	1K-2K	3.04x	2.53x	44.56	0.981
		2K-4K	1.71x	1.22x	46.94	0.980

all evaluated scenarios, our FFmpegSR framework can perform super-resolution to 4K at real-time speed and high visual quality.

C. Comparison with FFmpeg-provided Super-Resolution

The FFmpeg repository includes an implementation of a super-resolution video filter (SRF) [9]. It supports two models: SRCNN and ESPCN. In addition, SRF can perform full-frame super-resolution only as it does not support patch-based super-resolution. For this reason, we are only able to compare our FFmpegSR with SRF when both solutions are using ESPCN to operate on full frames. In addition, for fair comparison, both FFmpegSR and SRF perform super-resolution on the Y channel only. Note that we focus our comparison on inference speed. The visual quality of super-resolution depends on the trained model, and when using the same model, e.g., ESPCN, with the same parameters, the visual quality results are expected to be very similar.

We present the inference speed results for the video "RaceNight" in Table V. Our FFmpegSR can achieve up to 1.8x speed up compared to SRF. Note that this speedup is calculated based on full-frame super-resolution. FFmpegSR can achieve even faster inference speed with patch-based super-resolution.

TABLE V: This table compares the full-frame super-resolution speed of our FFmpegSR and the super-resolution video filter (SRF) in the FFmpeg repository. In this comparison, both SRF and FFmpegSR use the same ESPCN model for super-resolution.

	3080 Ti	2080 Ti
SRF (in FFmpeg repository)	3.22x	1.28x
FFmpegSR (ours)	5.73x	1.59x

V. Conclusion

In this paper, we present an efficient super-resolution framework called FFmpegSR. It integrates FFmpeg with super-resolution models such as ESPCN and can perform real-time per-frame deep learning-based super-resolution at 4K quality. To meet the real-time requirement, we improve the inference time of FFmpegSR from three aspects, channel deduction, model selection, and saliency-detection-based patch selection. We evaluate our framework on 4 videos using 3 super-resolution models with 2 scaling resolutions on 2 different GPUs. Results shows that our framework can achieve real-time high-quality video enhancement under different settings and can outperform the speed of the existing super-resolution filter in the FFmpeg repository by 1.8x.

VI. Acknowledgment

We appreciate constructive comments from anonymous referees. This work is partially supported by NSF under grant CNS-2200042.

References

[1] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 645–661.

[2] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: enabling neural-enhanced video streaming on commodity mobile devices," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[3] D. Baek, M. Dasari, S. R. Das, and J. Ryoo, "dcSR: practical video quality enhancement using data-centric super resolution," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 336–343.

[4] "FFmpeg," http://www.ffmpeg.org/.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[6] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[7] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[8] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[9] "FFmpeg sr Filter," https://ffmpeg.org/ffmpeg-filters.html#sr-1.

[10] "INSTALLING C++ DISTRIBUTIONS OF PYTORCH," https://pytorch.org/cppdocs/installing.html.

[11] A. Mercat, M. Viitanen, and J. Vanne, "Uvg dataset: 50/120fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 297–302.

[12] 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*, vol. 13, no. 4, pp. 600–612, 2004.

[13] H. J. Seo and P. Milanfar, "Static and space-time visual saliency detection by self-resemblance," *Journal of vision*, vol. 9, no. 12, pp. 15–15, 2009.

[14] C. Chen, G. Wang, C. Peng, Y. Fang, D. Zhang, and H. Qin, "Exploring rich and efficient spatial temporal interactions for real-time video salient object detection," *IEEE Transactions on Image Processing*, vol. 30, pp. 3995–4007, 2021.