RenderSR: A Lightweight Super-Resolution Model for Mobile Gaming Upscaling

Tingxing(Tim) Dong, Hao Yan, Mayank Parasar, Raun Krisch
Samsung Austin Research Center
tim.dong, hao.yan, m.parasar, r.krisch@samsung.com

Abstract

Mobile game play can be a prime use case where an efficient SR network can lead to both performance boosts and power savings. In this paper, we present RenderSR (RSR), a bandwidth aware super-resolution network designed for use in mobile game upscaling. We explore how different factors affect the resulting image quality: color space, the inclusion of the depth channel, sharpening. With a 40K parameter size, RenderSR without sharpening achieves a PSNR value difference ranging -0.41 to 0.36dB from several much larger SR models. RenderSR with sharpening super resolved large objects such as rocks, buildings, tree trunks are almost identical to the ground truth. Based on our performance experiment, we propose that RenderSR upscales the GPU rendered image on NPU or DSP on the mobile SoC.

1. Introduction and background

Compared to a PC with a dedicated power supply, mobile phone with battery is a power constrained device. Previous research shows that the GPU has far higher power consumption than other components on a mobile SoC (System on a chip) when users are mobile gaming [23]. Recently, more and more phone companies have been releasing phones with high refresh rates, such as 120Hz, while most mobile games are still around 30-60 FPS. This large gap requires that the GPU must render fast enough to satisfy the display's throughput. One natural way is to let the GPU render at a lower resolution to catch up with the screen refresh rate. However, the number of pixels on mobile devices is also growing very fast, and more and more 2K (1440p) screen phones are on the market. For comparison, most PC games are on just 1080p or 2K resolution. Users require mobile phones to deliver both high quality and fast frames at the same time. Therefore, there is an increasing demand for image upscaling for mobile gaming.

Traditional upscaling methods are based on interpolation filters and are not able to deliver satisfying image quality (IQ). SRCNN [16], ESPCN [28], EDSR [24], RealSR [13] FALSR [14], MoreMNAS [15], ProSR [30], and ESRGAN [29], etc people resort to neural-networks (NN) to upscale images and achieve much better IQ. An overview of different NN models can be seen in [11], [31], [32]. There are models [25], [12], [17] optimized for mobile devices. An overview of mobile optimized models can be seen in 2021 Mobile AI workshop reports [21] and [20]. Such upscaling methods are commonly referred to as super-resolution. Most of them target camera photo datasets like DIV2K and assume the low-resolution (LR) and high-resolution (HR) images have a bicubic degradation.

However, the different resolution image users see in the game is not a simple pixel interpolation but the number of pixels rendering difference. Therefore, the dataset needs to be obtained from GPU actual pipelines. Unlike photo dataset which only has HR, and corresponding LR can be obtained easily by a bi-cubic downgrade. Obtaining LR and HR frame-to-frame mapping from GPU rendering is not easy but subject to various factors like various FPS, physical animation, etc. SR is a well-known ill-posed problem [26], when the mapping from LR-HR is not well established, this ill-posed problem will become even worse. In our work, we target on NN based super-resolution on mobile gaming.

2. RenderSR implementation

2.1. RenderSR model architecture: bandwidth aware design

While the accuracy and speed of super-resolution neural networks have improved drastically over recent years, the resulting networks like mentioned above are too large for use on mobile devices. Our ultimate goal is to achieve real-time rendering on mobile gaming. So RSR is designed to be lightweight. RenderSR is wider and shorter: four layers but the number of channels is relatively large in middle layers. Previous research [10] [27] show the wider version of the network can achieve similar and even better in

accurancy and outperform the deeper version of the network in performance as long as appropriate training. The wider and shorter version can maximize the underlying GPU parallelisim and minimize the latency. Since our intension is to upscale per frame, our batch has to be one during inference. Unlike other models who reply on increasing batch size to saturate GPU, RenderSR increases the number of channels.

Yet, increasing channels leads to a larger intermediate layer buffer which can be a challenge for mobile SoC's bandwidth. By channel 64, RenderSR's intermediate buffer is 128 MB assuming upscaling a 540p frame to 1080p. Each 128 MB buffer will be loaded (read or written) 5 times, so the memory footage is 640 MB. The flagship Qualcomm Snapdragon 888 memory bandwidth is 51.2 GB/s [4]. So the upper-bound time is 12.5ms by bandwidth which is the headroom left for RenderSR doubling native 1080p FPS which will be discussed in section 4 in details. Therefore, the hyperparameter, the number of channels cannot go beyond 64 which is the upper limit. Users can customize the model by going down if they have a lower end mobile SoC. In this paper, we use 64 by default.

The first two 3x3 convolution layers are for feature extracting and the transposed convolution layer for reconstruction as in Figure 1. The last step is a sharpen filter. The Rectified Linear Unit (Relu) activation function is used after each convolution. The transposed convolution is also referred to as a deconvolution layer in many literatures [33]. Instead of outputting smaller dimensional feature maps as in regular convolutions, transposed convolution layers generate larger dimensional feature maps. Transpose convolutiion determines the upscaling factor, in our paper, we use the upscaling factor to 2x2 (width x height). Transposed convolution is algorithmically similar to a 3x3 convolution and can be implemented using the im2col, GEMM. By using this transposed convolution layer to perform upscaling at the end, RenderSR saves on computation by performing the majority of its operations on the lower dimensional image space.

The sharpen filter is not trained by but a dedicated filter. The sharpen factor can be tuned by end users. If factor is 0, basically means sharpen is turned off. The input can be in the RGB or YCbCr color space. The output has the same color space. The inputs are normalized to be in the floating point [-1, 1] range in a pre-preprocessing step. The output is then denormalized back to the corresponding color space.

RenderSR reflects the latest progress in industry and academia. It is vastly different from 2014 SRCNN [16] in many aspects. SRCNN has 3 layers, and up-scales image first before convolution. RenderSR has 4 layers and convolution first and up-scales later. SRCNN reconstructs with convolution. RenderSR reconstructs with transposed convolution. SRCNN's filter size is big 9, 5 while RenderSR filter is small 3x3. SRCNN works on YCbCr space and only trains one channel Y, while RenderSR works on RGB color space and even depth channel and train all 3 or 4 channels. RenderSR has a sharpen layer while SRCNN does not. Mostly, RenderSR targets on GPU rendering frames instead of SRCNN's camera photos.

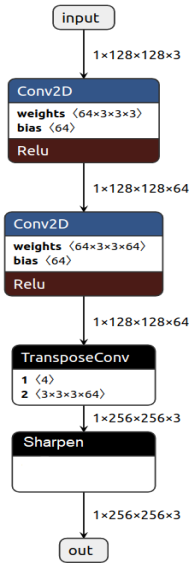

Figure 1. RenderSR architecture.

2.2. Training dataset collection

Our first challenge is training dataset collection. To our knowledge, there are no gaming datasets publicly available yet. Training datasets is critical, since without correct datasets models would not be able to learn and make correct predictions. For camera photo dataset, LR images can be easily obtained by down-scaling the correpsonding HR images. HR is also called ground truth (GT). We use both terms interchangeably throughout this article. However, the LR-HR mapping in games is not a simple pixel interpolation, and we want the NN models to learn real rendering rather than better interpolation. Therefore, the LR and HR frame pairs need to be obtained from actual rendering. In our experiment, we collected two training datasets: Aztec Ruin, Zen Garden.

Aztec Ruins (AR) comes with Gfxbench5 [3]. Users can download and run the benchmark at different resolutions and dump the screenshots in corresponding resolutions. However, your underlying GPU will not always give you the same FPS, and thus the frame screenshot obtained would be off in content and cause the training dataset to be misaligned and non-useable. Fortunately, by having a license to access their source code we were able to get around this issue by locking the FPS and time intervals and re-building and running the demo. After locking these values, each different

run was guaranteed to generate the same frame for each scene. We run on offline mode and dump the TGA format file from the frame buffer. We finally collected a dataset including about 10,000 images.

Zen Garden (ZG) is a game making engine UE4 product [2]. ZG has a lot of physical anomalies as other real games: random birds, flower petal dropping, swimming fish and 2D icons overlay. Achieving an LR-HR one to one exact mapping is even harder than AR. We do not have special license to get a special build but have to modify UE4 setting and examine ZG source code to disable these anomalies and 2D overlay to capture a consistent dataset. We re-built ZG in UE4 and packaged it into a game ready mode. Then we ran the game in benchmarking mode and dumped the screen shots. The entire dataset includes 800 images.

2.3. Training details

9/10 of the dataset is used for training. 1/10 is used for testing. Random images are selected to present the result. Each game has separate trained weights. We think it is a tiny overhead, as a game can be hundreds GBytes size, while the RenderSR wight size is only in KB.

During the training, the input image is converted to PNG format and randomly cropped size of 96x96 or 128x128. These tiles are augmented by horizontal flips, and 90-degree rotations to ensures our network's robustness and generalization capability. Users can choose a bigger tile size but we do not input the whole image into training because features learning and pixel-mapping are more local than global. Notice that the tile size does not affect the number of weight parameters which is determined only by kernel filter sizes and in/out channel sizes of each layer.

We trained our model with the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10e - 8$. We set the mini-batch size as 16 for training. The learning rate we initialized with piecewise constant decay with boundaries of 200000 and values [1e-3, 1e-5]. We compared with different initializers and found that Glorot uniform delivered the highest PSNR (peak signal-to-noise ratio) value. We chose L1 loss because it provided better convergence than L2 [11]. We implemented our networks on top of the Tensorflow framework with versions tf1.4 and later migrated to tf2.3. Our model was trained on an NVIDIA Quadro P5000 GPU. It takes over 8 hours to train a dataset with 300,000 training steps.

3. Experiments and Results

3.1. Y only training vs RGB color space

We experimented on the Aztec Ruins datasets with two types of color space: YCbCr and RGB. YCbCr training was first adopted in SRCNN, where the model was trained on only the Y channel of images as the authors claimed that human eyes are more sensitive to the luminance channel. Training on only the Y channel of images can save extra compute power compared to training on all three channels in the very first layer. We initially adopted this approach. However, compared to the model trained on all three RGB channels, the resulting IQ of Y channel-only is obviously worse than RGB as shown in Figure 2, the edge on leaves are jagged compared to the RGB color space results. While it would not be easy to distinguish the RGB results from the GT if they were not side by side. Therefore, we decided to train RGB color space in all of our later experiments.

3.2. RenderSR vs other methods

RenderSR vs Bicubic: We compared RenderSR with bicubic interpolation and GT in Figure 3, where RenderSR achieves much better image quality than bicubic interpolation. Bicubic result loses the shape of leaves and the edge is very blurry. While the RenderSR maintain the shape of leaf and the bar edge is distinct. The average PSNR of RenderSR is 28.11 far better than bicubic 23.45. We testified, even with a smaller set of parameters, RenderSR can achieve very much better visual quality than naive bicubic upscaing.

Table 1. Model name, number of parameters, and average PSNR on the Zen Garden dataset

Model	Parameters	Avg PSNR (dB)
Bicubic	N/A	23.45
RenderSR w/o sharpen	40k	28.11
RenderSR with depth buffer	41k	28.30
MoreMNAS-A	1039k	28.606
MoreMNAS-B	1118k	28.593
FALSR-A	1021k	28.524
FALSR-B	326k	28.540
FALSR-C	408k	28.557
EDSR-16	1369k	28.711
EDSR-4	483k	28.489
EDSR-2	335k	28.279
EDSR-1	262k	27.94

RenderSR vs other large NN models: Because other models do not have sharpen stage, we turn off RenderSR sharpening here. We compared the parameter sizes and PSNR values on the ZenGarden dataset in Table 1. RenderSR's PSNR is only slightly smaller than the other large models. EDSR-X [24] and FALSR-X from Xiaomi [14] have multiple residual blocks configuration, where X represents number of residual blocks. They are significantly bigger and thus we do not think fit power-constrained devices. A randomly picked tile is shown in Figure 4. By zooming in, we found FALSR-A has a better contrast

Figure 2. Left to right: Y channel, RGB, GT. Y channel-only model gives aliased leaf edges, while RGB model and GT are almost identical.Best viewed on a screen.

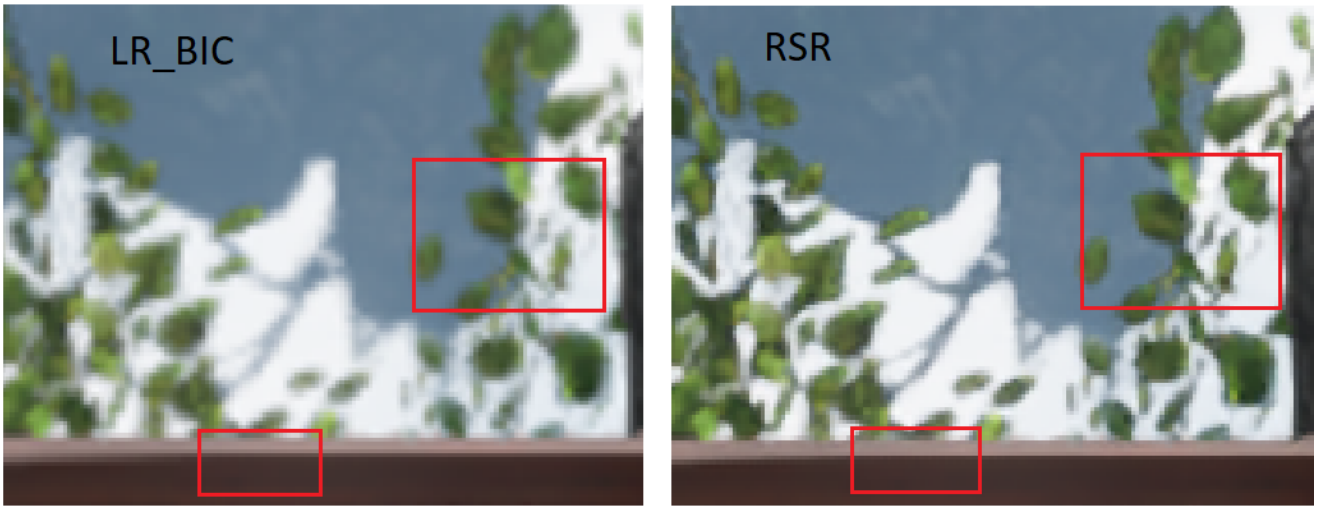

Figure 3. A random clip by zooming in very details. Best viewed on a screen

and details which is not so surprising, since FALSR-A is 25x larger in size than RenderSR. But there are artifacts on the branches, and more alias on white strips which are less apparent in RenderSR. Overall, FALSR-A has better details but appears more alias. RenderSR is more smooth. Artifacts are a common issue for SR espeically for large models which tend to learn too many features [29] and fill the details with the features they have learned. To improve RenderSR's contrast, we instroduce depth buffer and sharpen filter in the next two sections.

3.3. RenderSR with depth buffer

The pixel depth [1] records the distance between camera and objects and can be extracted from the GPU rendering pipeline. This is a major difference than photo dataset where depth information is almost impossible to obtain from RGB pictures without metadata attached in general. Figure 5 shows results with and without the depth channel. The depth channel improves the PSNR value by 0.19dB and provide better contrast, like the branch is darker and the red flower pedal is more saturated. Also the pedal and branch are less fuzzy. We calculated the RMS contrast value in Table 2. By including depth channel, the RMS contrast value increases by 0.29 and comes closer to GT. So training with depth channel can be a promising techniques. But including depth needs extra storage and increase the model size (only in the first layer) slightly.

Table 2. RMS (root mean square) contrast value of each image

Bicubic	RSR w/o depth	RSR w/t depth	GT
38.67	39.70	39.99	40.69

3.4. RenderSR with sharpening

The reason of adding a sharpeing layer is that the first three layers of RSR optimizes toward a norm loss and tends to mix the image's low frequency and high frequency signals. The sharpening filter can filter high frequency out and blocks certain low high frequency which makes the image crisper. Users can replace with any sharpen filters they want. Here, we chose the Contrast Adaptive Sharpening (CAS) filter originally from AMD [6]. CAS is contrast aware algorithm, where areas that are already sharp are sharpened less, while areas that lack detail can

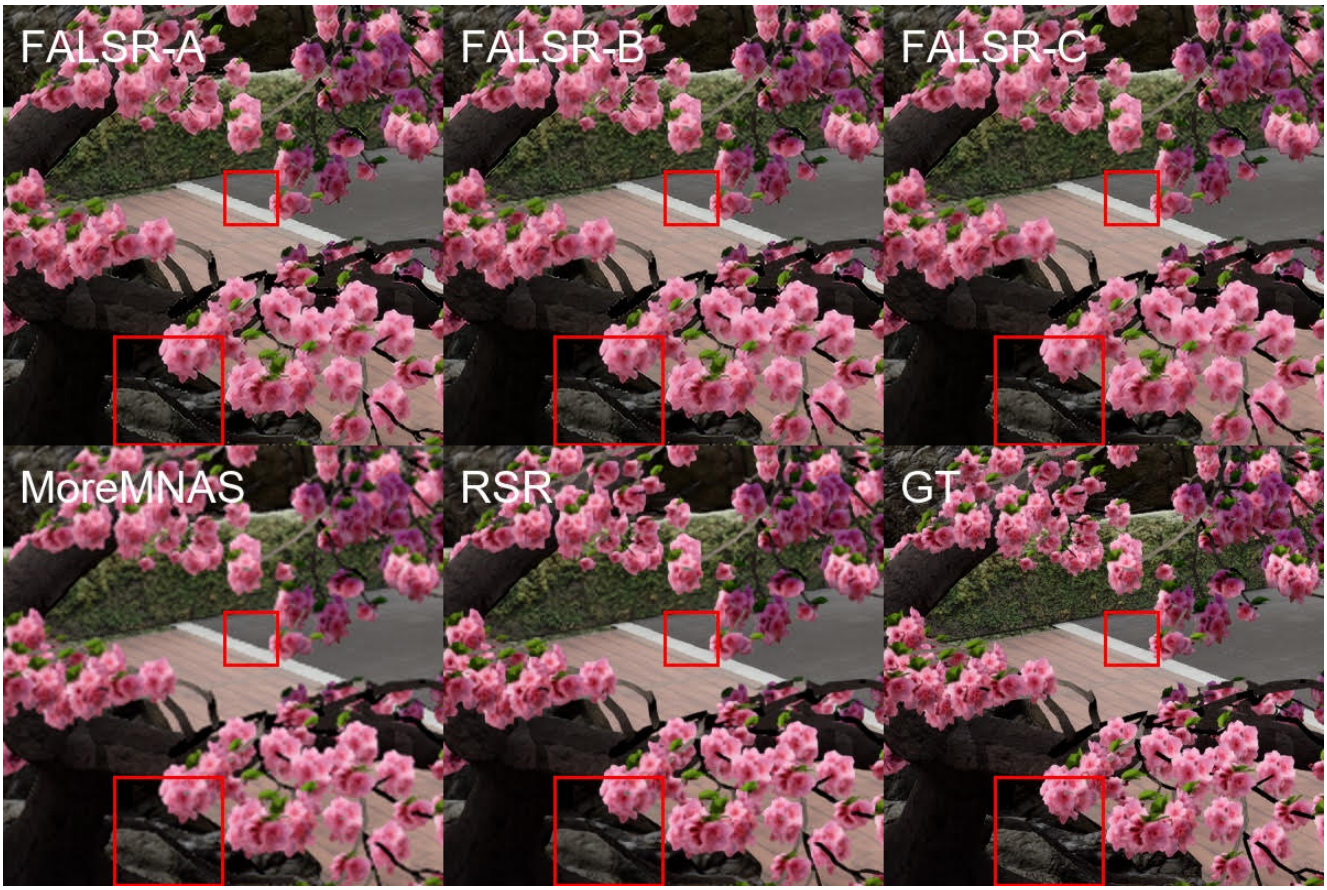

Figure 4. A random clip of Zen Garden. Best viewed on a monitor

Figure 5. Results of adding scene depth as the fourth channel besides RGB.Best viewed on a monitor and zoom in

be sharpened more. In this way, we gain both contrast and sharpeness for RenderSR.

During this paper writing, AMD enhanced CAS to create FidelityFX™ Super Resolution (FSR) by adding an edge adaptive spatial upsampling pass [8] . Different from NN models, FSR is not learning based but interpolation based algorithm. AMD open-sourced their tool [7] so we can compare our RSR result with FSR. FSR provide a scaling option raning from 1.3 to 2 per dimension. Here, we compare FSR 2x2 upscaling since RenderSR upscales 2x2.

Figure 6 shows a random scene of Zen Garden: AMD FSR upscaling, RenderSR without Sharpen, and RenderSR

with CAS sharpen and GT. Figure 7 shows RenderSR with sharpen vs GT. Notice FSR also has a sharpening stage.

Large objects: like buildings, rocks, tree trunk, cloud and mountains. In our opinions, large objects are most gamers' first focus when they are actively gaming. We first ask our reached viewers' subjective judgement for Figure 6. Everyone reaches a consensus: RSR w/t sharpen >RSR w/o sharpen >FSR. The top left FSR is the most blurry one and lack most texture and details (zoom in 3 rectangles). Compared to RenderSR with the same level of details (top right), a sharpen stage (bottom left) block certain low frequencies and increase the clearness of the texture on the rock and pedals and make it more crispy (zoom in rectangle box with rocks and flowers). Someone even say RenderSR is better than GT. The sharpening factor is in a tunable range of [0, 1.0]. Sharpen will cause certain info lose, so users need to find a sweet spot. We tuned between several factors, 0.0, 0.3, 0.5, 0.7 and 1.0. We finally found 0.7 is a good one for ZG.

Tiny objects: By zooming in the rectangle containing the leaves in Figure 6. RenderSR fills in much more details than FSR. RenderSR recognized more distinct leaves than FSR which blend them together. But only the GT has the most details and can distingush every single leaf. Although RenderSR with sharpen is crispy but still lack GT's level details. RSR is not as good at distinguishing very tiny objects as large ones , especially when they are clustered together like grass as in Figure 7. The reason is due to RenderSR's capacity, it is very hard for a 40K NN network to extract such tiny features.

PSNR vs human perception: Figure 6 also shows the PSNR value in bracket. The human being's perception order is different than PSNR value order. That indicates agian that PSNR is not a reliable metric for human being's subjective image quality as also reported by other researches [19], [18], [9], etc. The reason is PSNR is based on pixel-pixel comparision but not feature level human can perceive. Yet, PSNR can be as a sanity check as it usually first increases with human being's perception and then no longer correlates.

4. Performance requirement

Not every game is equal in the cost of rendering. Super resolution target for "hard" rendering games. When the native rendering is expensive, gamers gain benefits from SR. Here we assume the native rendering is only at 20 FPS (50ms) and 1/4 of the resolution is 80 FPS (12.5ms), and end game users wish to achieve 40 FPS (25ms) in SR, considering human needs at least 30 FPS to feel comfortable. This gives SR a headroom of 12.5ms.

Desktop GPU: we first implement a RenderSR inference engine with OpenCL 2.0 on a discreet entry level AMD Navi 5500XT GPU with datatype fp32. To upscale a 540p frame to 1080p, RenderSR's takes 13.88 ms with each layer break-down as shown in Table 5. 13.88 ms is quite close to the headroom metioned above. So a human eye friendly 1080p SR gaming (assume this game's native 1080p is only 20 FPS) is quite feasible on a desktop GPU, especially considering that 5500XT is an entry level card. Navi 5500XT's fp32 performance is 5.2 TFLOPs.

Mobile SoC: Table 3 is the result of RenderSR on Snapdragon 865 via AIBenchmark v5.0.1 by running custom model [22]. We can choose CPU, TFLite GPU Delegate and NNAPI for acceleration, and the other options like Qualcomm QNN or other vendors libraries all gray out and cannot be enabled. Due to the time limit, we have not get a chance to successfully quantized our model to int8. Table 3 shows the current optimal path is via GPU delegate fp16. AIbenchmark reports data of upscaling a 128x128 image to 256x256 while RenderSR aims to upscale image 960x540 to 1920x1080. If assuming the time increases linearly with image size, RenderSR needs 200ms to upscale 540p. Linear is a conservative estimation, since some overhead like kernel launching is constant. Yet even exclusing overhead factors, there is a still big gap between mobile GPU's hundred ms to desktop's 13.8ms.

Table 3. AI Benchmark timing results of RenderSR: upscale 128 to 256

Processors	fp16 (ms)	fp32 (ms)
CPU	22.9	30
TFLite GPU Delegate	6.33	8.31
NNAPI	10	11

NN is a compute intensive workload. The flagship mobile GPU on Qualcomm Snapdragon is 1.37 TFLOPs in compute power [4] still far behind entry level desktop GPU. Such existing gap turns us to other acceleartors on the SoC. Recently mobile SoC vendors intergrate dedicated AI acceleartors such as Samsung Exynos neural processing unit (NPU) [5] or Qualcomm hexagon digital singnal processor (DSP) [4] to do AI workload. NPUs and DSPs are capable of dozens of TFLOPs at a lower power budget as in Table 4. By delegating workload to NPU (or DSP), the GPU can be freed and previous idle NPU will be utilized. Compare to GPU only solution, GPU-NPU solution can save both time and power by careful pipelining. The main issue is the latency between the GPU and NPU. NPU takes input from GPU, upscales it and hands over to GPU to perform the folloswing pipeline. The headroom left is only a few milliseconds and the synchronization overhead between them must remain low. Developing such an inference engine on NPU or DSP is beyond this paper's

Figure 6. PSNR is in bracket. The human being perception order is different than PSNR order. Best viewed on a monitor and zoom in.

scope currently and need specific implementation for each vendor.

Conclusions

We designed a bandwidth-aware super-resolution model targeting mobile phone gaming. To make sure our model learn rendered pixel difference rather than interpolation, we collected two datasets from GPU rendering pipeline

Table 4. Some hardware TFLOPs capability

Navi 5500XT	mobile GPU Qual S888 [4]	mobile NPU [5]
5.2	1.37	> 10

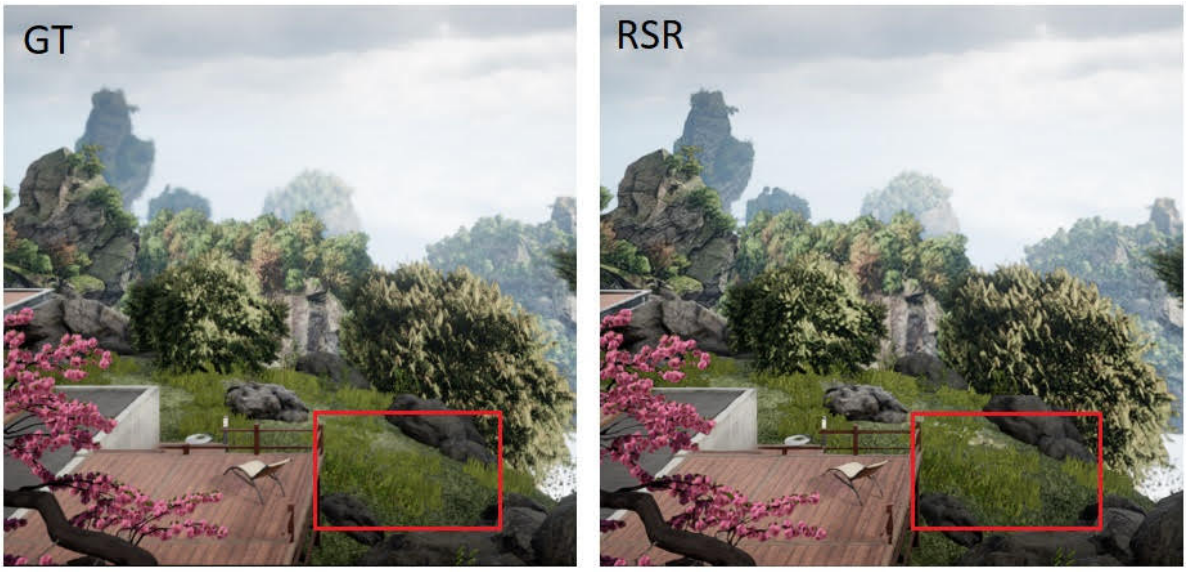

Figure 7. Left to Right: GT, RenderSR. RSR is excellent with big objects like rocks, tree trunks, clouds on the sky, and buildings but not as good for tiny grasses clustered together. Best viewed on a monitor

Table 5. RenderSR (OpenCL) on AMD Navi 5500XT GPU: upscale a 540p to 1080p frame

Layers	Time (us)
Layer 1	3322
Layer 2	4303
Layer 3	6201
Sharpen Layer	54.8
Total	13.88 (ms)

where LR-HR pairs were aligned by controlled setitng up and modifying source code of game engine. We explored a set of techniques including color space, depth buffers, sharpening to optimize the PSNR, and human perceptual image quality. We found that by including depth buffer, the image constrast can be effectively improved. We demonstrated by just 40K parameters (compared to GB sized games), RenderSR can achieve much better visual quality than bicubic and other interpolation based filters like FSR. Compared to many large NN models, RenderSR can achieve reasonably good image quality and has less artifacts. For large objects, RenderSR with sharpening delivers close to ground truth quality.

References

[1] Depth expressions. 4

[2] Epic zen garden in epic content - ue marketplace. 3

[3] Gfxbench - unified cross-platform 3d graphics benchmark database. 2

[4] Qualcomm details the snapdragon 888. 2, 6, 7

[5] Samsung introduces game changing exynos 2200 processor with xclipse gpu powered by amd rdna 2 architecture. 6, 7

[6] Amd fidelityfx - contrast adaptive sharpening, Dec 2021. 4

[7] https://github.com/gpuopen-effects/fidelityfx-fsr, Dec 2021. 5

[8] https://gpuopen.com/fidelityfx-superresolution/, Dec 2021. 5

[9] Adel Almohammad and Gheorghita Ghinea. Stego image quality and the reliability of PSNR. In Khalifa Djemal and Mohamed A. Deriche, editors, *2nd International Conference on Image Processing Theory Tools and Applications, IPTA 2010, 7-10 July, 2010, Paris, France*, pages 215–220. IEEE, 2010. 6

[10] Md. Zahangir Alom, Theodore Josue, Md Nayim Rahman, Will Mitchell, Chris Yakopcic, and Tarek M. Taha. Deep versus wide convolutional neural networks for object recognition on neuromorphic system. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–8. IEEE, 2018. 1

[11] Saeed Anwar, Salman H. Khan, and Nick Barnes. A deep journey into super-resolution: A survey. *ACM Comput. Surv.*, 53(3):60:1–60:34, 2020. 1, 3

[12] M. Ayazoglu. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2472–2479, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. 1

[13] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019,*

Seoul, Korea (South), October 27 - November 2, 2019, pages 3086–3095. IEEE, 2019. 1

[14] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*, pages 59–64. IEEE, 2020. 1, 3

[15] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Multi-objective reinforced evolution in mobile neural architecture search. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August, 2020, Proceedings, Part IV*, volume 12538 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2020. 1

[16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016. 1, 2

[17] Zongcai Du, Jie Liu, Jie Tang, and Gangshan Wu. Anchor-based plain net for mobile image super-resolution, 2021. 1

[18] Fernando A. Fardo, Victor H. Conforto, Francisco C. de Oliveira, and Paulo S. Rodrigues. A formal evaluation of PSNR as quality measurement parameter for image segmentation algorithms. *CoRR*, abs/1605.07116, 2016. 6

[19] Quan Huynh-Thu and Mohammed Ghanbari. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. *Telecommun. Syst.*, 49(1):35–48, 2012. 6

[20] Andrey Ignatov, Andres Romero, Heewon Kim, Radu Timofte, Chiu Man Ho, Zibo Meng, Kyoung Mu Lee, Yuxiang Chen, Yutong Wang, Zeyu Long, Chenhao Wang, Yifei Chen, Boshen Xu, Shuhang Gu, Lixin Duan, Wen Li, Wang Bofei, Zhang Diankai, Zheng Chengjian, Liu Shaoli, Gao Si, Zhang Xiaofeng, Lu Kaidi, Xu Tianyu, Zheng Hui, Xinbo Gao, Xiumei Wang, Jiaming Guo, Xueyi Zhou, Hao Jia, and Youliang Yan. Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge: Report, 2021. 1

[21] Andrey Ignatov, Radu Timofte, Maurizio Denna, Abdel Younes, Andrew Lek, Mustafa Ayazoglu, Jie Liu, Zongcai Du, Jiaming Guo, Xueyi Zhou, Hao Jia, Youliang Yan, Zexin Zhang, Yixin Chen, Yunbo Peng, Yue Lin, Xindong Zhang, Hui Zeng, Kun Zeng, Peirong Li, Zhihuang Liu, Shiqi Xue, and Shengpeng Wang. Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report, 2021. 1

[22] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635, 2019. 6

[23] Tianxing Jin, Songtao He, and Yunxin Liu. Towards accurate gpu power modeling for smartphones. page 7, 05 2015. 1

[24] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1132–1140. IEEE Computer Society, 2017. 1, 3

[25] Shaoli Liu, Chengjian Zheng, Kaidi Lu, Si Gao, Ning Wang, Bofei Wang, Diankai Zhang, Xiaofeng Zhang, and Tianyu Xu. Evsrnet: Efficient video super-resolution with neural architecture search. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2480–2485, 2021. 1

[26] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. PULSE: self-supervised photo upsampling via latent space exploration of generative models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2434–2442. Computer Vision Foundation / IEEE, 2020. 1

[27] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1

[28] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. pages 1874–1883, 2016. 1

[29] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: enhanced super-resolution generative adversarial networks. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11133 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 2018. 1, 4

[30] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. A fully progressive approach to single-image super-resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 864–873. Computer Vision Foundation / IEEE Computer Society, 2018. 1

[31] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. Deep learning for image super-resolution: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(10):3365–3387, 2021. 1

[32] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Trans. Multim.*, 21(12):3106–3121, 2019. 1

[33] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2018–2025. IEEE Computer Society, 2011. 2