SNPE-SRGAN: Lightweight Generative Adversarial Networks for Single-Image Super-Resolution on Mobile Using SNPE Framework

Hendrik Tampubolon[1,2], Aji Setyoko[2], Fanindia Purnamasari[3*]

[1] Department of Information System, Faculty of Engineering and Computer Science, Krida Wacana Christian University, Jakarta, Indonesia

[2] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

[3] Department of Information Technology, University of Sumatera Utara, Medan, Indonesia

Email:hendrik.tampubolon@ukrida.ac.id, aji.setyoko@gmail.com,fanindia@usu.ac.id

Abstract. An image resulting from a low-resolution (LR) camera on the mobile phone has lower quality than a high-resolution(HR) camera on a DSLR. Meanwhile, the HR camera is pricing if compared with the LR camera. How to achieve a single-image quality on LR camera likewise on HR camera becomes essential research in the past years. Addressing this issue can be done by upscaling a single LR image. Recently, the super-resolution generative adversarial network (SRGAN) model is one of the state-of-the-art super-resolution(SR)models employed on single-image SR. However, implementing a deep learning model like SRGAN on a mobile device is challenging in computation power and resources. This study aims to develop a smaller and lower resources model while preserving single-image SR quality on mobile devices. To meet these objectives, we convert, quantize, and compress the SRGAN model on Snapdragon Neural Processing Engine (SNPE) as an example. We then validate the SRGAN on the DIV2K dataset on which improves the model performances. Besides, we conduct experiments on GPU, DSP environment. The experimental result confirmed that SNPE-SRGAN capable of achieves not only HR images' quality but also low latency by 0.06 second and smaller model by 1.7 Mb size running on DSP. Also, the SRGAN-DLC-Quantized running on GPU has a smaller size by 1.7 Mb and lower latency by 1.151 seconds compared with Non-quantized SRGAN-TensorFlow by 9.1 Mb and 1.608 seconds latency.

1. Introduction

Images resulted from the mobile camera are usually lower resolution than the digital camera. Also, images with a long-distance shot have a low quality on a cheap phone's camera. Improving image quality on a phone device by embedding the camera's better hardware will increase the cost. Hence, this approach might not be practical due to the market demand for lower-cost tendency[1] [2]. Balancing the trade-off between cost and quality is continuously going on for years to come. The challenge is to improve image quality from a cheap camera on a mobile device while preserving low-cost development. In fact, low-resolution (LR) images can be enhanced by preprocessing LR into the higher resolution

(HR), where the images' quality will also be improved. This task can achieve through a generative model that fixes the image texture, referred to as the image super-resolution (SR) problem.

Many research works have done addressed SR problems through a deep learning (DL) model in the past years. For example, C. Dong et al. propose a deep convolutional neural network(CNN) to learn and map LR and HR images[3]. Their study reported a deep CNN model superior in restoration quality to the traditional model, like the sparse-coding-based SR approach. Although deep CNN achieves accurate and fast with a deeper CNN model, it might be less performed for texture details at large upscaling factors. This occasion occurs because most of the traditional model focuses on minimizing the error using mean square error(MSE) where the estimation has a maximum peak of the signal-to-noise ratio (PSNR) but deficient to a high frequency—as a consequence, deteriorating to meet the expected fidelity. To tackle this limitation, C. Ledig *et al.* introduce a generative adversarial network (GAN) with the perceptual loss to single-image SR, so-called SRGAN [4]. SRGAN capable of generating HR realistic images with four-time upscaling factors from extremely down-sampling images. Thus, SRGAN becomes state-of-the-art for SR challenges and has an enormous impact on single-image SR research and benchmark. After that, the research to improve SRGAN continuously conducted recently, such as enhanced SRGAN (ESRGAN) in [5] and also self-attention GAN in [6]

Dealing with the DL model on embedded systems or mobile devices with low-computation and limited resources face challenges because the DL model is typical computation-hungry and data-hungry. Therefore, to tackle this limitation, some research works have proposed which vary from compressions like SVD, network pruning[7], quantization [8], low-rank factorization, and knowledge distillation of the model [9-11]. Some extend the approaches mentioned above to SR on a mobile device (MobiSR) have also been studied [12]. Several CNN based model examined in their study. Similarly, we adopt those techniques in our study, but we focus on implementing the GAN-based model on Snapdragon SoC. This study aims to provide a lightweight SRGAN model for single-image SR that can work in the real-time processing on a resource-constrained like a mobile device, more specifically on Snapdragon SoC. To achieve the objectives, model compression, quantization, and optimization aforementioned are taking into account. In summary, the contribution of this work as the following:

- We propose a low-cost alternative to a hardware camera on a mobile device that can enhance LR camera image quality by developing the SRGAN model into a mobile device, more specifically SNPE runtime and environment.
- As stated in reference, training data and a suitable objective function are essential to enhance SRGAN. Thus, we train the SRGAN model to the more completed dataset like DIV2k [13]; therefore, accuracy and performance improved.
- We employ compression [7, 9], quantization in [8], and optimization techniques in which we achieve a smaller and faster SRGAN model for single-image SR while preserving the quality of the HR images. An experimental study also provided to investigate the GPU and DSP environment's performance.

2. SNPE-SRGAN for Single-Image Super-Resolution

2.1. SISR and GAN

Single Image Super-Resolution (SISR) is considered an image-to-image translation problem that maps LR into HR images. Figure 1 depicts the SISR model in general. Many approaches have done addressed to SISR in the past decades. Nevertheless, this study focuses more on deep-learning-based approaches with GAN that solve SISR. Before SRGAN, mapping LR images into HR images can be done by bicubic interpolation, CNN, and deep residual network (SRResNet) [3]. The typical optimization algorithms used in SISR is MSE on pixel-by-pixel between the output and the original image. Although using MSE has benefits to maximize PSNR, it strives to deal with uncertainty connatural in recovering lost high-frequency details. This occasion leads to overly smooth but low perceptually in restored images.

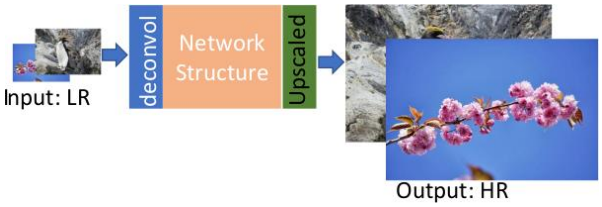

Figure 1. An illustration of the generic model in SISR

Therefore, C. Ledig et al. propose perceptual loss where PSNR is incorporated with structural similarity (SSIM) [4]. The loss function (L^{SR}) in SRGAN can be seen in equation 1: the weighted sum of content loss (L^{SR}_{MSE}) in MSE-based solution and adversarial loss ($10^{-3}L^{SR}_{GAN}$) in GAN model. For more detailed information on the equation derivation, please refer to SRGAN[4]

$$L^{SR} = L^{SR}_{MSE} + 10^{-3}L^{SR}_{GAN} \quad (1)$$

$$L^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I^{HR}_{x,y} - G_{\theta G}(I^{LR})_{x,y})^2 \quad (2)$$

$$L^{SR}_{GAN} = \sum_{n=1}^{N} -log\, D_{\theta D}(G_{\theta G}(I^{LR})) \quad (3)$$

As a typical GAN model, SRGAN architecture consists of the generator (G) and the discriminator (D) module. Each block has a generic structure of a deep network, such as the convolution layers (Conv), batch normalization (BN), and parameterized ReLU (PReLU) in G with additional pixel-shuffle. G also implements skip connections much the same as ResNet. However, in the discriminator (D), instead of using PReLU, it uses Leaky ReLU non-linearity. Moreover, the network structure in SRGAN is based on the structure of VGG [14] Network and DCGAN [15]. Figure 2 describes the network structure of the SRGAN.

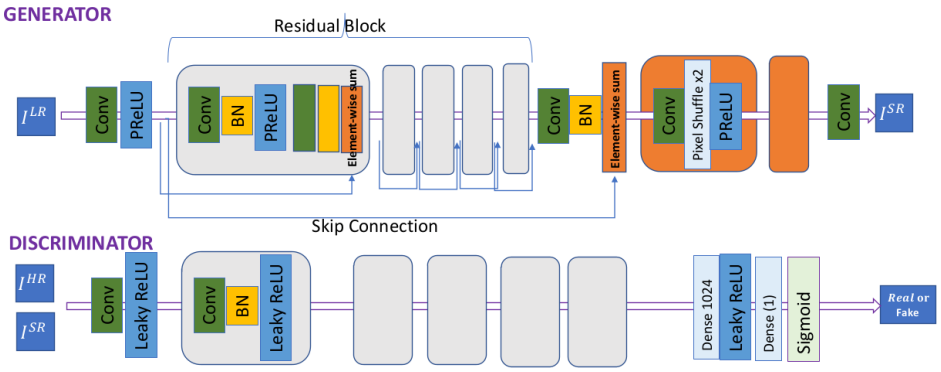

Figure 2. Network Architecture of SRGAN adopted in our study [4]

2.2. SNPE Framework

The Snapdragon Neural Processing Engine (SNPE) is a Qualcomm Snapdragon software accelerated runtime to execute deep neural networks. This framework allows the user to run the DL network arbitrarily. It is also capable of executing either CPU, GPU, and DSP environment. The quantization procedure has an 8-bit fixed point. Besides, the user enables to debug easily on offline host ubuntu before production. Unfortunately, the device supported by SNPE is still limited. Table 1 is the list of the device currently supported by SNPE. To develop DL on SNPE, it first converts the model to the DLC file and subsequently quantizes the DLC file on DSP.

Table 1. Snapdragon device support matrix

Snapdragon Device	CPU	GPU	DSP	AIP	Phone model
Qualcomm Snapdragon 845	Yes	Yes	CDSP	No	ASUS Zenfone 5Z, ASUS ROG phone, Sony Xperia XZ2, Sony Xperia XZ3, hTC U12+
Qualcomm Snapdragon 835	Yes	Yes	ADSP	No	hTC U11+ , Nokia 8, Nokia 8, Sirocco
Qualcomm Snapdragon 821	Yes	Yes	ADSP	No	hTC M10
Qualcomm Snapdragon 710	Yes	Yes	CDSP	No	Nokia 8.1
Qualcomm Snapdragon 660	Yes	Yes	CDSP	No	ASUS Zenfone 4, ASUS Zenfone Max Pro m2, Nokia 7 Plus
Qualcomm Snapdragon 636	Yes	Yes	No	No	ASUS Zenfone 5, Sony Xperia 10 plus, hTC U12 life, Nokia 6.1 Plus
Qualcomm Snapdragon 630	Yes	Yes	No	No	ASUS Zenfone 4, Sony Xperia 10 Sony Xperia XA2 Plus
Qualcomm Snapdragon 450	Yes	Yes	No	No	hTC Desire 12

2.3. Proposed SNPE-SRGAN Systems and Design

Overall, the proposed framework has three stages: the training model, optimization of inference on SNPE, and the online device's deployment (real mobile device). At the first stage, the training model follows the step as in the original SRGAN; however, train the model to different datasets DIV2K instead of Set5, Set14, and BSD100 datasets. After training, the model needs to be validated in the test set. This procedure is repeated until it satisfied the intended objectives. Subsequently, we freeze the weights and biases into a static file for the model compression. At the second stage, the model is ready to convert to the DLC file. The model optimization was done in this step like quantization and speeding up the model's performance on DSP. In this work, we employ Data-Free Quantization(DFQ) algorithm [8], which is a default algorithm deployed in SNPE. Debugging of the proposed model can be examined on the offline host before deployment. Finally, after the evaluation meets the threshold in terms of speed(latency) and the model's size, the model can be deployed onto real mobile devices. The workflow of the proposed SNPE-SRGAN systems is depicted in Figure 3.

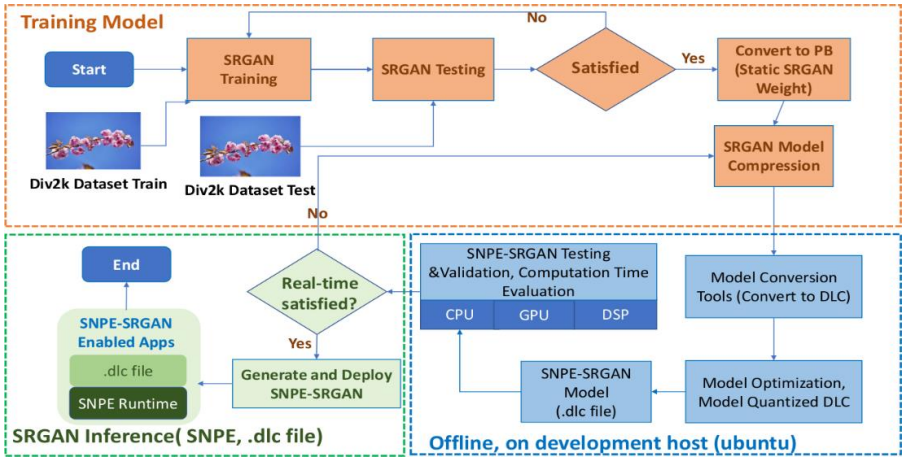

Figure 3. The processing Flow of Proposed SNPE-SRGAN Systems

- Model Compression

  After freezing the weight of trained SRGAN, the model is ready to compress. The Spatial Singular Value Decomposition (SSVD)[9] and Channel Pruning (CP)[7] method were adopted in this work. In SSVD decompose one large layer of the tensor of SRGAN into two smaller layers. For example, Given a convolution layer, with the kernel (m,n,h,w) where m is the input channels, n the output channels, and h, w giving the height and width of the kernel itself, Spatial SVD will decompose the kernel into two kernels. One of size (m,k,h,1) and one of size (k,n,1,w), where k is called the rank. The smaller the value of k, the larger the degree of compression achieved.

- Quantization Algorithm
  The quantization in SNPE is the process to enable Floating-point 32-bit (FP32) run in Fixed-point 8-bit (INT8) with very minimal loss of the model's accuracy. To achieve it, DFQ algorithm employed in SNPE. This step firstly equalizes the weight tensors to decrease the magnitude variation across the channels (Cross-Layer Equalization). Then, rectify a shift in layer output resulted from quantization (Bias Correction). Subsequently, simulate the on-target SRGAN inference accuracy. Finally, fine-tune the model, and repeat this process until the expected output satisfied. Please refer to DFQ paper [8] for detailed equation and process in Cross-Layer Equalization and Bias Correction method

3. Experiments

To evaluate the proposed SNPE-SRGAN framework, we then conduct experiments on BiCubic and on the SRGAN model. Then evaluate the accuracy by showing the PSNR and SSIM of the model trained on the DIV2K dataset. Besides, we also evaluate computational time (latency) through the different environments of CPU or DSP. Eventually, a comparison of the quantized model with the Non-quantized model was also provided in the following section.

3.1. Datasets

The dataset used in our study comes from DIVerse 2K resolution high-quality images as used for the challenges at NTIRE in CVPR 2017, CVPR 2018, and ECCV 2018 [13]. The dataset contains 1000 2k resolution images. We then divide the given dataset into 800 images for training and100 images for testing and validation.

3.2. Experimental Settings

- Tools and environment

All the environmental setup used in this study has done based on the following tools; Python 3.x programming language under TensorFlow 1.8 and SNPE 1.25.1.310 version. Also, offline hosts have done using ubuntu 18.04 with GPU NVIDIA GTX 1080 Ti available while Asus Zenfone 5z for deployment testing.

- SNPE-SRGAN training procedures

Overall, the training of the model was done by following the original network setup. The HR images were downsampling with factor r =4 bicubic kernel to obtain LR images. The LR images were standardized into a range [0,1], and HR images into a range [-1, 1]. MSE loss and VGG loss were scaled as the same as the reference. In general, the optimization used in our study follows the typical setting of DL, such as stochastic gradient descent, BN, ADAM = 0.9, and learning rate 10^{-4}. The generator G and discriminator D are updated based on [16], which is equivalent to k = 1. Residual block in generator G (B =16). The implementation of our SNPE-SRGAN was modified from the tensor layer version in [17, 18].

3.3. Result and Discussion

After choosing the model which have the same dependencies support by SNPE, we then retrain the model on the DIV2k dataset and evaluate using PSNR and SSIM metrics. Table 1 shows the experimental results. Also, the comparison of generated images can be seen in Figure 4. The left part is the input of LR images, the middle part is the output of generated HR images, and the right side is the ground truth of HR images.

Table 2. Comparison of bicubic and SRGAN model evaluated on DIV2k dataset

Model	PSNR (dB)	SSIM
SRGAN	23.043	23.892
Bi-CUBIC	23.812	23.033

The image itself has the following properties: the LR as the input (169 x 255 pixel), Bi-Cubic nd SRGAN output with 4x upscaling (676x1020), and the ground-truth (1356 x 2040 pixel). We evaluate the result using 53 test image by calculating PSNR and SSIM. The result of our SRGAN model 23.043 of PSNR and 23.892 of SSIM. Meanwhile, the Bi-Cubic model yielding 23.812 of PSNR and 23.033 of SSIM. Although the PSNR on our model slightly lower than Bi-Cubic, however, the images resulted from Bi-Cubic's are more blurry than our model.

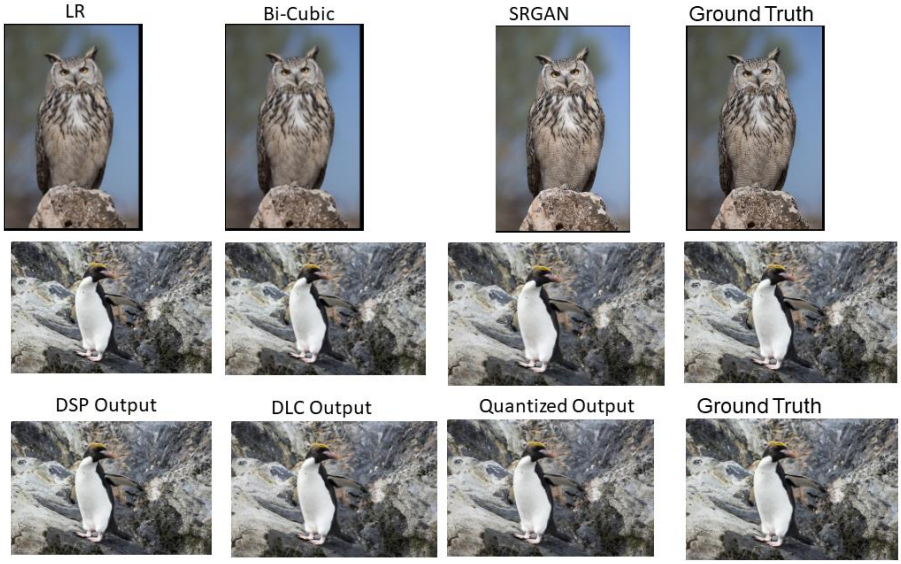

Figure 4. LR Images, Generated HR Images of Bi-Cubic and SRGAN Model, and Ground Truth Images and SRGAN output on DSP with Quantized and Non-quantized Images

Besides the PSNR and SSIM, the latency of SRGAN was also investigated on different mode and environment setup. To do that, we first save the weight of the best model as *npz* file, then freeze the layer using TensorFlow to save the model as PB file and after that, using *"snpe-tensorflow-to-dlc"* command to convert the .pb model into .dlc file format for android target. Furthermore, using *"snpe-dlc-quantized"* command to quantize the DLC format. Finally, a *"snpe-net-run"* to execute the model by given input file (raw file). Table 3 shows the model size and latency. As we can see in the Table, the SRGAN-Quantized version on the DSP environment gives us the best result with a smaller and faster performance. Based on this experimental result, we confirm that the SNPE-SRGAN model can perform in a timely and yet preserving the quality of HR images.

Table 3. Comparison of size of the model and latency

Model	Size (MB)	Latency (second)
SRGAN-TensorFlow	9.1	1.608
SRGAN-DLC	6.2	1.125
SRGAN-DLC-Quantized	1.7	1.151
SRGAN-DLC-Quantized-DSP	**1.7**	**0.06**

4. Conclusions

In conclusion, we validate that the SRGAN model in the DIV2K dataset improves the model performances. Besides, we conduct experiments on GPU, DSP environment of SNPE. The experimental result confirmed that SRGAN capable of achieves not only HR images' quality but also low latency by 0.06 second and smaller model by 1.7 Mb size running on DSP. Despite its performances, the improvement of SRGAN is still open research. In the future, the implementation of more advanced models like PFF, ESRGAN, and RCAN to SNPE would be interesting to be investigated. Moreover, it is also possible to apply SRGAN or enhance SRGAN to a real-time video on the embedded system.

References

[1] K. Bagchi, P. Kirs, and F. López, "The impact of price decreases on telephone and cell phone diffusion," *Information & Management,* vol. 45, no. 3, pp. 183-193, 2008/04/01/ 2008.

[2] T. Pettinger. (2012). *Falling Price of Mobile Phones*. Available: https://www.economicshelp.org/blog/6279/economics/falling-price-of-mobile-phones/

[3] 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*, 2014, pp. 184-199: Springer.

[4] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681-4690.

[5] X. Wang *et al.*, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0-0.

[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*, 2019, pp. 7354-7363: PMLR.

[7] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389-1397.

[8] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1325-1334.

[9] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 108, no. 4, pp. 485-532, 2020.

[10] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review,* vol. 53, no. 7, pp. 5113-5155, 2020/10/01 2020.

[11] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282,* 2017.

[12] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors," presented at the The 25th Annual International Conference on Mobile Computing and Networking, Los Cabos, Mexico, 2019. Available: https://doi.org/10.1145/3300061.3345455

[13] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126-135.

[14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," presented at the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015. Available: http://arxiv.org/abs/1409.1556

[15] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434,* 2015.

[16] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.

[17] H. Dong *et al.*, "Tensorlayer: a versatile library for efficient deep learning development," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1201-1204.

[18] Qualcomm. (2020, June, 20). *TensorFlow Model Conversion*. Available: https://developer.qualcomm.com/docs/snpe/model_conv_tensorflow.html