LiteBokeh: Compact Model for Real-time Bokeh Rendering

Biao Wu, Yaowei Guo, Si Gao, Shaoli Liu, Chengjian Zheng, Diankai Zhang, Ning Wang
State Key Laboratory of Mobile Network and Mobile Multimedia Technology, ZTE, China
{wu.biao,guo.yaowei,gao.si,liu.shaoli,zheng.chengjian,zhang.diankai,wangning}@zte.com.cn

Abstract

Bokeh is an important artistic effect that highlights the main subject of a photo by blurring the out-of-focus areas. While DSLR cameras can achieve this effect naturally, mobile devices still find it challenging to reach a similar level due to their compact optics and tiny camera sensors. In this paper, we propose two solutions for generating bokeh images: one is designed for real-time applications, striking a balance between quality and efficiency, while the other prioritizes output quality without emphasizing efficiency optimization. During the model training phase, we combine L1 loss, perceptual loss, and SSIM loss to render realistic bokeh effects. To address the issue of foreground blurring in images generated by end-to-end depth-of-field rendering methods, we propose a enhancement network. This approach introduces a lightweight enhancement network—comprising only a residual block—built upon the initial rendering output, which significantly improves the clarity of foreground regions. Experiments show that our method can render high-quality bokeh effects on mainstream flagship smartphone chipsets and can process a 1024×1536 pixel image in real time. Our lightweight TinyUnet solution won first place in the 2026 Mobile AI Rendering Realistic Bokeh Challenge.

1. Introduction

The popular bokeh effect highlights a photograph's subject by blurring distracting background elements. Given the inherent optical constraints of smartphone cameras, producing true optical bokeh is challenging. Therefore, many devices now rely on computer vision algorithms to recreate the effect digitally.

Over the past few years, generating bokeh effects automatically has become a highly popular subject, and numerous solutions are now integrated into most smartphone camera apps. Modern mobile devices are equipped with powerful GPUs and NPUs, which are well-suited for executing the proposed deep learning models [12, 16]. Methods for rendering automatic bokeh effects are categorized into two main types based on whether prior information (such as segmentation masks or depth maps) is required. The first type relies on prior information, which can be further divided into segmentation mask-based methods and depth map-based methods. In segmentation mask-based methods, a segmentation network [17] first generates a foreground mask. The image is then Gaussian-filtered, and the original image is blended with the filtered version using the mask to achieve the blur effect. However, the background blur produced by this method lacks a sense of depth. In depth map-based methods, depth maps are used as prior information to apply varying degrees of blur to different areas, resulting in a bokeh effect with a certain hierarchical quality, similar to that of a DSLR camera. However, this solution is relatively complex and not easily suitable for real-time processing. The second type does not require prior information and directly achieves an end-to-end rendering of the automatic bokeh effect. In Mobile AI 2026 Rendering Realistic Bokeh Challenge, the target was to develop an efficient end-to-end AI-based bokeh effect rendering approach that is based on the TensorFlow Lite framework and capable of running on modern smartphone GPUs.

In this work, we propose two solutions for generating bokeh images: one is designed for real-time applications, striking a balance between quality and efficiency, while the other prioritizes output quality without emphasizing efficiency optimization. Our main contributions are:

- We design a TinyUnet network specifically for real-time applications, achieving a balance between quality and efficiency.
- Another network we designed is called Glass-Enc-Net, which prioritizes bokeh effect quality over efficiency optimization.
- During training, we used depth maps to assist training and assigned different weights to different regions in the loss function, so that the background is blurred while the foreground remains clear when the subject stands out.
- To address the issue of foreground blurring in end-to-end depth-of-field rendering, this paper proposes a

two-stage approach. It incorporates a lightweight enhancement network (with a single residual block) upon the initial render output, significantly improving the sharpness of the foreground region.

2. Related Work

In recent years, one prevalent technique has been to segment the central object from the scene and then blur everything else [1–3]. A different technique introduced for this purpose entails blurring the image based on the estimated depth map. Prior knowledge, including salient region detection and depth estimation, forms the basis for most of these methods. The method introduced in [4] builds upon the megaDepth [5] algorithm, renowned for its exceptional capabilities in single-image depth estimation. Subsequently, an efficient process is developed to apply selective defocusing, utilizing a Gaussian filter to blur areas outside the focal plane. A DDDF framework introduced in [6] employs both salient region segmentation [7] and depth estimation [5] as its priori knowledge. In addition, the PyNET approach [9] utilizes depth maps generated from MegaDepth. The authors note that while such depth information may not boost standard quantitative metrics like PSNR and SSIM, it still contributes to enhanced visual quality. Based on the discussed methods, it is found that integrating prior information—such as saliency detection or depth estimation maps—can partially enhance the visual quality of generated images. However, these approaches are critically reliant on such priors; when the priors are ineffective in certain contexts, the final output may encounter unpredictable flaws. Furthermore, the preprocessing involved in these methods is computationally expensive, requiring the models to run initially whenever a new image is inferred on a device.

In reference [10], a complete deep learning framework and its accompanying EBB! bokeh rendering dataset were introduced, with a neural network designed to automatically convert wide depth-of-field images into shallow depth-of-field versions. Numerous other approaches based on this dataset have been developed, as documented in [11–15]. BGGAN [13] introduced a novel generator named GlassNet that produces bokeh effects without requiring specialized hardware. During the fine-tuning stage, the method combines a GAN-based approach with perceptual loss to achieve realistic bokeh rendering. Furthermore, Instance Normalization (IN) is reformulated within the network, enabling the resulting tflite model to utilize IN while maintaining efficient acceleration on mobile GPUs.Although BGGAN generates realistic blurring effects, it can only achieve processing speeds in the order of seconds on smartphones, which does not meet the requirements for real-time processing.

3. Approach

3.1. Realistic Bokeh Rendering Challenge

The challenge aimed to achieve the optimal balance between MOS and latency in output bokeh images on mobile devices.

The final score for this challenge, as defined in Formula (1), is determined by two key metrics:

- The quality of the reconstructed results.
- The runtime of the model on the actual target mobile platform.

$$Score(MOS, runtime) = \frac{2^{2 \cdot MOS}}{runtime} \tag{1}$$

The scoring formula shows that doubling the computational efficiency on the target platform can result in a quality improvement equivalent to a 0.5-point increase in the MOS score. This necessitates a focus on balancing image quality restoration and computational performance optimization in network design.

3.2. Analysis

The goal of the Mobile AI challenge was to create an efficient, AI-powered end-to-end solution for rendering bokeh effects, capable of running on mobile GPUs via TensorFlow Lite. In previous MAI challenges [18, 19], most proposed methods relied on end-to-end deep learning solutions with multi-scale encoder-decoder architectures, which significantly improved runtime performance. Training loss typically used L1, SSIM/MSSSIM, VGG-based Sobel, and Charbonnier loss functions. However, BGGAN demonstrates that using adversarial loss can significantly improve the quality of bokeh effects. BGGAN utilizes GAN-based methods and perceptual loss during model training to render realistic bokeh effects. Furthermore, the instance normalization (IN) algorithm has been reimplemented in the network. While the BGGAN network produced good bokeh effects, its complexity was too great to run in real time on smartphones. The PyNET network utilizes prior information from the depth map, feeding both the original image and the depth map into the network, resulting in a decent bokeh effect. However, the bokeh effect is affected by the depth map, and because it requires prior information from the depth map, the network is difficult to simplify. The algorithms mentioned above share a common problem: the foreground in the bokeh effect is somewhat blurred compared to the original image. Therefore, our main improvements are building a lightweight network and making the foreground of the bokeh effect clearer.

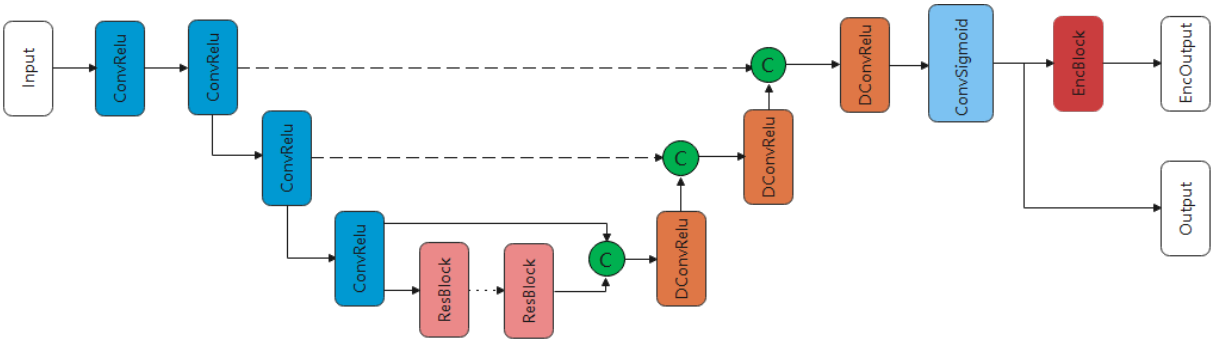

Figure 1. TinyUnet architecture overview.

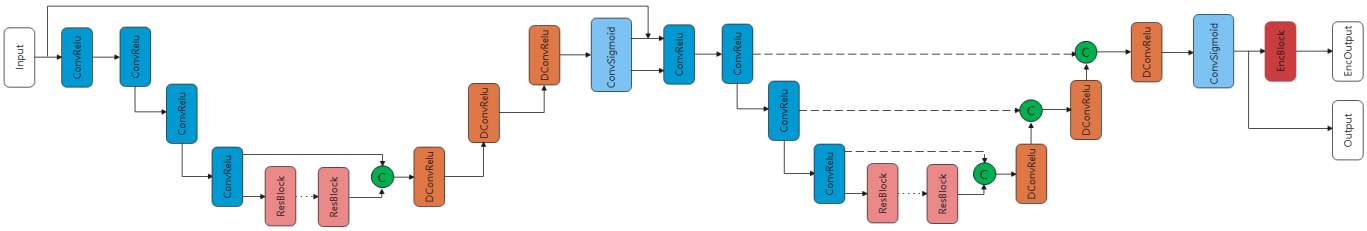

Figure 2. Glass-Enc-Net architecture overview.

3.3. Network Architecture

We proposed a two-stage network architecture: the first stage generates bokeh effects, and the second stage employs an enhancement network to further improve image quality. TinyUnet strikes a good balance between bokeh effect and running efficiency. Its network structure is shown in Figure 1. The encoder module comprises three downsampling layers, each implemented with a convolutional layer using a stride of 2. The features extracted by the encoder are then processed through four residual blocks. Each residual block consists of a sequence of layers: a convolutional layer, a ReLU activation layer, a standard normalization layer, another convolutional layer, and another ReLU activation layer. Additionally, a skip connection is added between the input and output of each residual block. The transformed feature maps from the residual blocks are passed to the decoder module, which includes three transposed convolutional layers with a stride of 2. All convolutional layers in the encoder and the transposed convolutional layers in the decoder employ the ReLU activation function. Finally, the output layer is a convolutional layer with a stride of 1 and the sigmoid activation function, followed by an enhancement network that uses only one residual block. The Glass-Enc-Net architecture is implemented without taking runtime efficiency into consideration, and it is fundamentally constructed based on the underlying BGGAN network framework. The sole modification introduced in this implementation is the incorporation of an additional enhancement network, as visually depicted and detailed in Figure 2. This specific network is designed with a primary focus on augmenting the overall model complexity, with the explicit objective of further improving the quality and realism of the bokeh effects generated.

3.4. Loss

The training was divided into two phases. The first phase involved training the bokeh model, using L1, SSIM, and perceptual loss. The perceptual loss is calculated using Euclidean loss on the feature map, located in the fourth ReLU5 layer of VGG19. The objective function is formally given by formula (2). The second stage involves training the enhancement model, primarily aimed at improving the foreground blur in the bokeh results. This stage seeks to achieve the sharpest possible enhancement while maintaining fidelity, therefore only L1 Loss was used.

$$\mathcal{L}_{total} = 10 \cdot \mathcal{L}_1 + 0.1 \cdot \mathcal{L}_{vgg19} + 10 \cdot (1 - \mathcal{L}_{SSIM}) \quad (2)$$

4. Experiments

In this part, we will describe the implementation details of our proposed method and report the results on the EBB! [10] dataset.

4.1. Datasets

The goal of this competition is to design an efficient, deep learning-based solution for rendering realistic bokeh effects. For this, the participants were provided with a newly collected large-scale dataset, EBB!, which comprises

Figure 3. Depth maps generated by Deep Anything v2. Zoom in for better visualization.

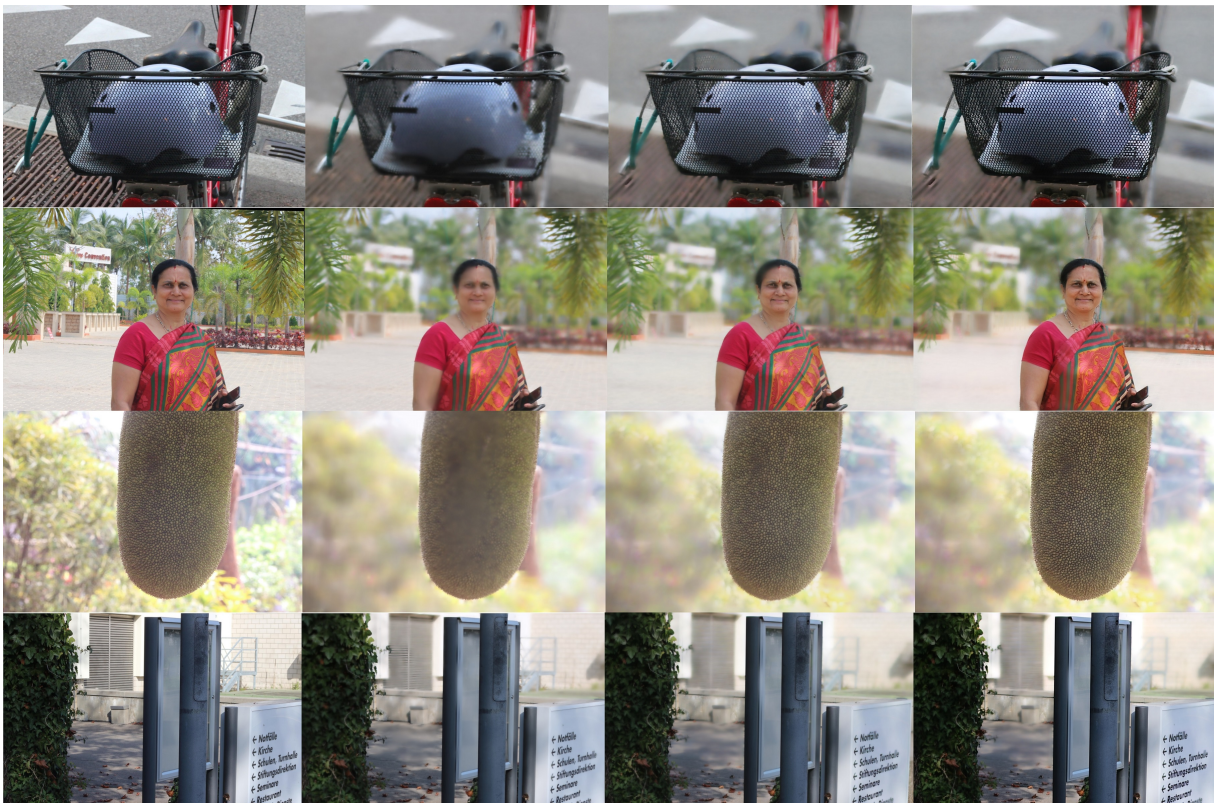

Figure 4. From left to right: input images, the results of not using a depth map during the training loss calculation, the results of using a depth map during the training loss calculation, The results of using depth maps in training loss calculation and adding an enhancement network during inference. Zoom in for better visualization.

a wide variety of RGB image content. This dataset contains approximately 5,000 pairs of aligned images captured with a Canon 70D DSLR camera, using different aperture settings to simulate normal photographs and photos with a bokeh (background blur) effect. All images in the dataset have a fixed height of 1024 pixels, while their widths vary. To address the foreground blurring artifact in bokeh synthesis, we trained an enhanced network using the original image of EBB! dataset. We constructed pairwise training sets for the enhanced network by applying degradation techniques—such as blurring, adding noise, and reducing contrast—to the ground truth (GT) data from the original image of EBB! dataset.

4.2. Training Configuration

A two-phase training strategy was employed for the process using TensorFlow. In the first stage, we used the EBB! dataset to train end-to-end bokeh rendering by randomly cropping images into 1024 × 1024 pixel blocks as input. The batch size was set to 16 limited by memory. The training phase utilized a composite loss function comprising L1 loss, SSIM loss, and perceptual loss and utilized the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with an initial learning rate of 1×10^{-4}. A decaying learning rate scheduler was used, decaying every 120 epochs, for a total of 600 epochs of training. In the second phase, to improve the end-to-end foreground blurring issue, We added an enhancement module to make the foreground clearer, using only one residual block. The training dataset for this augmentation model uses the ground truth (GT) data from the original image of EBB! dataset. Degradation techniques such as blurring, adding noise, and reducing contrast were employed. Pairwise training sets were constructed, and only L1 loss was used during training.

4.3. Depth Map

As mentioned earlier, PyNET inputs both the original image and the depth map into the network to generate higher-quality bokeh effects. However, this method significantly increases model processing time and cannot achieve real-time performance. Inspired by this, we considered using depth maps to assist model training, assigning different weights to different regions of the image when calculating the loss, thus playing a role in saliency detection. To generate stable and reliable depth maps, we used the open-source deep anything v2 model to generate corresponding depth

Figure 5. From left to right: input images, the results of BGGAN, the results of PyNET, the results of our TinyNet. Zoom in for better visualization.

Figure 6. From left to right: input images, the results of our TinyUnet, the results of our Glass-Enc-Net. Zoom in for better visualization.

maps for the EBB! dataset, as shown in Figure 3.

4.4. Ablation Studies

We present two approaches for producing bokeh effects: the first is tailored to real-time scenarios, ensuring a harmonious compromise between performance and visual fidelity,

Team	PSNR	MOS	Snapdragon 8 Elite Gen5 GPU,ms / NPU,ms	MediaTek Dimensity 9500 GPU,ms / NPU,ms	Score
ZX VIP	22.58	6.5	19.7 / 16.8	42.6 / 11	282
Motong-AI-Studio	23.83	6.7	1826 / error	2801 / error	4.8
AVC2	23.54	1.5	16.6 / 17.2	38.3 / error	0.32
Manis	19.2	1.1	129 / 43.2	191 / error	0.03
UCAS ICT	22.93	1.7	204053 / error	179710 / error	0

Table 1. Quantitative Evaluation of MAI 2026 Bokeh Effect Rendering Challenge.

and the second is dedicated to achieving superior image quality, with minimal concern for computational speed. We developed the TinyUnet model, as shown in Figure 1, tailored for real-time usage, to strike a balance between output quality and computational efficiency. In addition, we created a network called Glass-Enc-Net, as shown in Figure 2, which emphasizes the quality of the bokeh effect rather than optimizing efficiency. In the training process, depth information was utilized to support model learning, as shown in Figure 3, and the loss function incorporated spatially varying weights for different areas. This approach results in a blurred background with the foreground kept sharp when the main subject is distinct. As shown in Figure 4, depth maps are used to assist in loss calculation during training, and different regions are assigned different weights. This results in the trained model maintaining more complete foreground details in the bokeh effect. Nevertheless, the above results, using end-to-end bokeh, exhibit a slight foreground blur compared to the original image. Therefore, we added an enhancement model during inference, as shown in Figure 4. After adding the enhancement model, the foreground of the bokeh result is clearer.

4.5. Quantitative Evaluation

The organizers of the MAI2026 Realistic Bokeh Challenge conducted a comprehensive evaluation of the results submitted by each participating team on the EBB! test set. This evaluation employed established image quality metrics, including Mean Opinion Score (MOS), Peak Signal-to-Noise Ratio (PSNR), and other relevant quantitative and perceptual metrics. To evaluate the model's efficiency, we measured its inference latency on a 1024×1536×3 resolution image using AI Benchmark (FP16 precision with TFLite GPU and NPU delegates). The tests were conducted on platforms equipped with MediaTek Dimensity 9500 and Snapdragon 8 Elite Gen5 chipsets. The results are shown in Table 1. The final score is calculated by incorporating the average GPU inference latency from both test devices and the Mean Opinion Score (MOS). Our TinyUnet strikes a good balance between performance and bokeh, resulting in a higher overall score that significantly surpasses the scores of other teams' submissions.

4.6. Visual Comparision

Our proposed TinyUnet algorithm was compared and evaluated with BGGAN and PyNET (without computational limitations) in terms of subjective visual quality. Our TinyUnet has an advantage in bokeh result sharpness and can achieve real-time inference, as shown in Figure 5. Although the lightweight TinyNet network achieves good results in balancing bokeh effect and running efficiency, it has obvious defects on some images, as shown in Figure 6. Therefore, we tried to increase the network complexity without considering efficiency constraints. Our proposed Glass-Enc-Net shows significant improvement.

5. Conclusion

In this paper, we proposed two end-to-end solutions for bokeh rendering: TinyUnet for real-time performance and Glass-Enc-Net for high-quality results. To prevent foreground blurring, we utilized depth maps only during the training phase to spatially weight the loss function to achieve depth-aware rendering without adding computational overhead during inference.Additionally, a lightweight enhancement network based on a single residual block was introduced during inference to sharpen the foreground. In training, a combination of L1, perceptual, and SSIM loss functions was employed to achieve realistic bokeh effects. We evaluated the model's performance on smartphone platforms powered by MediaTek Dimensity 9500 and Snapdragon 8 Elite Gen5 chipsets, using GPU and NPU delegation methods respectively. The results show that this method achieves real-time performance on mainstream flagship smartphones and significantly improves image clarity and overall performance, outperforming uncompromising PyNET and BGGAN networks by several times. Our lightweight TinyUnet model placed first in the 2026 Mobile AI challenge for realistic bokeh rendering.

References

[1] Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E., Sachs, I. Automatic portrait segmentation for image stylization. In: Computer Graphics Forum. vol. 35, pp. 93–102. Wiley Online Library (2016)

[2] Zhu, B., Chen, Y., Wang, J., Liu, S., Zhang, B., Tang. Fast deep matting for portrait animation on mobile phone. In: Proceedings of the 25th ACM international conference on Multimedia. pp. 297–305 (2017).

[3] Shen, X., Tao, X., Gao, H., Zhou, C., Jia, J. Deep automatic portrait matting. In: European conference on computer vision. pp. 92–107. Springer (2016)

[4] Dutta, S. Depth-aware blending of smoothed images for bokeh effect generation. arXiv preprint arXiv:2005.14214 (2020).

[5] Li, Z., Snavely, N. Megadepth: Learning single-view depth prediction from internet photos. *In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2041*, 2018.

[6] Purohit, K., Suin, M., Kandula, P., Ambasamudram, R. Depth-guided dense dynamic filtering network for bokeh effect rendering. *In: 2019 IEEE/CVF International Conference on Computer Vision Workshop.*

[7] Hou, Q., Cheng, M.M., Hu, X., Borji, A., Tu, Z., Torr, P.H. Deeply supervised salient object detection with short connections. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3203–3212, 2017.

[8] Li, Z., Snavely, N. Deeply-recursive convolutional network for image super-resolution. 2016.

[9] Ignatov, A., Patel, J., Timofte, R. Rendering natural camera bokeh effect with deep learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 418–419, 2020.

[10] Ignatov, A., Patel, J., Timofte, R. Rendering natural camera bokeh effect with deep learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops.*, pp. 418–419 (2020)

[11] Ignatov, A., Patel, J., Timofte, R., Zheng, B., Ye, X., Huang, L., Tian, X., Dutta, S., Purohit, K., Kandula, P., et al. Aim 2019 challenge on bokeh effect synthesis: Methods and results. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW).*, pp. 3591–3598. IEEE (2019)

[12] Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., Van Gool, L. Ai benchmark: All about deep learning on smartphones in 2019. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW).*, pp. 3617–3635. IEEE (2019).

[13] IQian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J. Bggan: Bokehglass generative adversarial network for rendering realistic bokeh. In: *European Conference on Computer Vision.*, pp. 229–244. Springer (2020)

[14] Dutta, S. Depth-aware blending of smoothed images for bokeh effect generation. In: *Journal of Visual Communication and Image Representation 77, 103089 (2021).*

[15] Dutta, S., Das, S.D., Shah, N.A., Tiwari, A.K. Stacked deep multi-scale hierarchical network for fast bokeh effect rendering from a single image. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*, pp. 2398–2407 (2021).

[16] Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., Van Gool, L. Ai benchmark: Running deep neural networks on android smartphones. In: *Proceedings of the European Conference on Computer Vision (ECCV).*, pp. 0-0 (2018)

[17] Lin, S., Yang, L., Saleemi, I., and Sengupta, S. Robust High-Resolution Video Matting with Temporal Guidance. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 238-247.

[18] A. Ignatov et al. AIM 2020 Challenge on Rendering Realistic Bokeh. In: *arXiv:2011.04988*

[19] A. Ignatov et al. Realistic Bokeh Effect Rendering on Mobile GPUs, Mobile AI AIM 2022 challenge: Report. In: *arXiv:2211.06769, 2022.*