BRIDGING THE DOMAIN GAP IN REAL WORLD SUPER-RESOLUTION

Charles Laroche, Matias Tassano

GoPro France

ABSTRACT

In this paper, we propose a state-of-the-art super-resolution algorithm and a framework to effectively train this architecture to handle real-world images. Recent learning-based, super-resolution methods have achieved impressive performance on ideal datasets. However, their performance plummets when tested on real images, as their assumed degradation model deviates from reality. Instead of assuming simplistic degradations, we make use of generative methods to mimic real degradations in order to narrow the domain gap between training data and testing data. In particular, we employ a novel Normalizing Flow architecture to replicate realistic noise and camera pipeline degradations. This framework, in conjunction with a carefully designed neural network architecture, results in a super-resolution algorithm which compares favorably to other methods. As for the model architecture, the combination between its performance on real images and lower computational load makes it attractive for practical super-resolution applications. We compare our method with different state-of-art algorithms, both on synthetic and real images. The experimental results demonstrate the remarkable performance of our super-resolution algorithm in real-world applications.

***Index Terms*—** Super-resolution, CNN, Normalizing Flow, Real-world data

1. INTRODUCTION

Single image super-resolution (SISR) methods aim to up-sample a low-resolution (LR) image into a high-resolution (HR) one. In other words, the goal of SISR is to multiply the size of an image by a given scale factor. The problem is ill-posed since there exists many solutions for a single low-resolution pixel. Recently, convolutional neural networks (CNN) have been used extensively to learn such a mapping [1, 2, 3]. Most of these methods assume that the downscaling kernel is known and fixed–e.g. bicubic kernel–, but the real downscaling kernels are usually unavailable and complex. Thus, the performance of learning-based methods is compromised when the pre-defined blur kernel differs from the real one [4, 5]. Additional challenges appear in the presence of degradations such as blur, noise, compression artifacts, etc. As these algorithms are trained with mainly clean and sharp images, there is a notable performance loss when they are tested on real world data. Our algorithm overcomes these limitations thanks to a rich training dataset which comprises a substantial collection of degradations. This dataset is constructed with a pipeline composed of different generative methods which produce realistic training samples. The improved fidelity of the training data results in a more robust reconstruction network. Regarding the downscaling, instead of assuming a unique and known kernel such as the bicubic kernel, we develop a refined version of

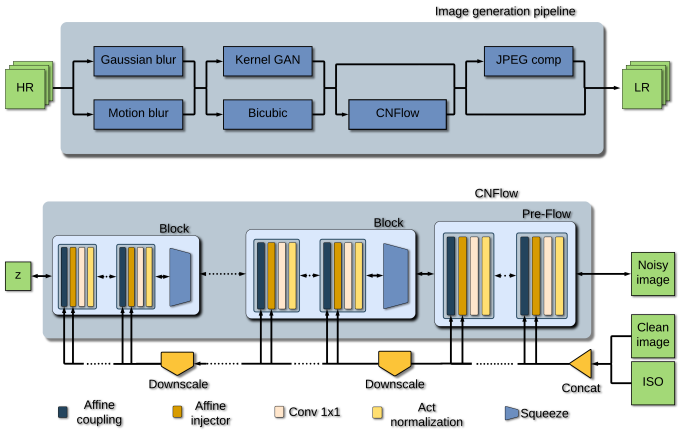


Fig. 1. Our image pipeline (top) and the architecture of CNFlow (bottom)

KernelGAN [6]. KernelGAN is based on the principle of maximizing the recurrence of patches across scales of the LR image [7, 8, 9]. This method proposes a fully unsupervised generative adversarial network (GAN) which learns the internal distribution of the patches of the LR image. Its GAN loss is optimized along with five additional regularization losses. We show that in reality these regularization terms prevent the network from converging to optimal kernel solutions. Our version of KernelGAN overcomes these limitations. Other degradations considered in our pipeline include blur, noise and compression artifacts. The employed blur kernels include isotropic and anisotropic Gaussian, and motion blur. As for the noise, a conditional normalizing flow [10] replicates the distribution of the noise and camera pipeline artifacts present in real image datasets. The source code is publicly available on the webpage of the algorithm [1].

1.1. Related Work

In this section, we briefly review relevant major SISR works and normalizing flows.

1.1.1. Deep SISR models

Several deep learning-based methods have been proposed to approach the SISR problem [11, 12, 13, 14, 15], which basically learn the mapping between a LR image and its HR version. In particular, ESRGAN [3] is a state-of-the-art SISR method which introduced a GAN-based loss to reconstruct high-frequency details. However, the algorithm struggles to generalize on real images, as their training dataset is built using bicubic downsampling. Non-blind methods

claroche@gopro.com

[1] https://github.com/claroche-r/Practical-Super-Resolution-with-CNFlow

assume that the information of the degradations is known and available. In this way, this information can be provided to the network in order to improve the restoration performance in the presence of multiple degradations. In particular, SRMD [16] introduced this idea by proposing to encode the blur kernel as an additional input to the network with principal components analysis.

1.1.2. Blind models

Blind models assume that the degradation kernels are unknown. [17] proposes a SISR dataset comprised of aligned LR/HR camera image pairs. Unfortunately, the construction of decent quality datasets of such type is impractical and time consuming, What is more, the application domain of the trained models is restricted to the domain defined by the LR images. Inspired by [8, 9], Shocher et al. [18] exploit the internal recurrence of information across scales inside an image and propose an unsupervised SISR method to compute the HR estimate with different blur kernels. To this end, they train a reduced CNN on patches extracted from the input image itself. Gu et al. [19] shows that when the estimated blur kernel is biased, the super-resolution estimation will not be accurate and may introduce artifacts. To remedy this issue, they propose an iterative method that jointly estimates the blur kernel and the super-resolved image. A clear disadvantage of methods such as the latter is the large runtime and computational burden. In a different vein, RealSR [20] illustrates the importance of accurate degradation modeling. Their approach employs realistic downsampling kernels obtained with KernelGAN instead of using bicubic downsampling to generate LR/HR pairs and train a ESRGAN architecture. In general, the existing literature assume the noise to be additive white Gaussian (AWGN). As this model does not correspond to real noise, these methods usually experience a performance drop when applied to real world images.

1.1.3. Normalizing Flows

Normalizing flows [10] are a family of invertible generative methods with tractable distributions. Both sampling and density evaluation can be exact and efficient. They have been applied to learn the SR space [21]. Other applications include image [22, 23, 24], audio [25], and point cloud [26] generation.

2. MODEL

2.1. Problem formulation

For our super-resolution model, we suppose that the low-resolution image is the blurry, downsampled and noisy version of an HR image. It can be written as follows:

$$y = (x \circledast k) \downarrow_s + \epsilon \tag{1}$$

where $k = k_{blur} \circledast k_{down}$ is the result of the convolution between the blur kernel k_{blur} and the downsampling kernel k_{down}, y and x be respectively the low and high resolution image, and ϵ is the noise. We further develop the different parts of our degradation model.

Noise: Classical SISR models consider ϵ to be AWGN. However, this approach does not match the real noise distribution. As discussed in [27], noise in raw images can be approximated with a Poissonian distribution. This noise distribution is transformed by the camera image processing pipeline, which renders the resulting distribution signal-dependant and correlated among neighboring pixels [28]. Based on this observation, we estimate the noise distribution directly from the data by means of a generative model.

Kernel: During the acquisition process, the image is subject to different blurring sources. First, the image is degraded by shooting conditions–e.g., camera-shake, exposition time, etc. We model these degradations using a blur kernel k_{blur}.

On the other hand, super-resolution aims to mimic the behavior of an optical zoom at scale s. An image at zoom 1 and at zoom s does not have the same sharpness. [7] models this phenomenon by a downsampling kernel called the instrinsic downsampling kernel. This downsampling kernel is referenced as k_{down} in our paper.

3. DATA GENERATION PIPELINE

Good performance of a super-resolution model relies upon training data. It is difficult to gather well-aligned low-resolution/high-resolution pairs, particularly in the presence of multiple degradations. We use synthetic data to overcome this problem. Our data generation pipeline mimics real world degradations using: a) A camera-shake model for realistic motion blur, b) a GAN-based realistic downsampling kernel generator, and c) a noise generator based on normalizing flows. Additionally, our framework applies JPEG compression of variable quality to simulate compression artifacts. The whole architecture of our data generator is summarized in Figure 1.

3.1. Blur kernel

Instead of considering the blur kernel k_{blur} to be Gaussian, we also consider motion blur kernels. These motion blur kernels are obtained using the realistic model proposed in [29]. We degrade 50% of our images using isotropic and anisotropic Gaussian blur and 50% using motion blur.

3.2. Downsampling kernel

The bicubic downsampling kernel is widely used in super-resolution due to its simplicity. Training a model on bicubic downsampling often leads to bad generalization properties on real world images. Realistic downsampling kernels can be directly estimated from the images. To do so, we use a modified version of KernelGAN [6]. We replace the Gaussian constraints from KernelGAN with a perceptual loss. Our loss can be expressed as follows:

$$\begin{aligned}\mathcal{L} &= \|1 - \sum_{i,j} k_{i,j}\|_1 + \sum_{i,j} \|k_{i,j}\|_1^{0.5} \\ &- E_{z \in \mathcal{Z}}[log(D((z \circledast k) \downarrow_s) + \|\Phi((z \circledast k) \downarrow_s) - \Phi((z) \downarrow_s)\|_1]\end{aligned} \tag{2}$$

where $(z \circledast k) \downarrow_s$ is the downsampled high-resolution patch z by a kernel k and stride s, and Φ are features extracted with a VGG-19 network [30]. The term $\|\Phi(G(z)) - \Phi((z) \downarrow_s)\|_1$, where $G(z) = (z \circledast k) \downarrow_s$, enforces the preservation of the elements in the image across the two scales. The term $\sum_{i,j} \|k_{i,j}\|_1^{0.5}$ enforces the sparsity of the kernel, and the constraint $\|1 - \sum_{i,j} k_{i,j}\|_1$ forces the sum of the elements of the kernel to be equal to one. Finally, the discriminator $D(.)$ ensures that the distributions of the generated LR and the small patches match. We find that our modified version of KernelGAN produces more diverse kernels and better convergence. It also allows non-Gaussian downsampling kernel estimation since we removed all the Gaussian constraints in KernelGAN. We train

this model on every image of the HR database to obtain a pool of realistic downsampling kernels $\mathcal{K}_{down}$.

3.3. Noise generation with CNFlow

We propose to learn the distribution of the noise from a dataset of noisy/clean pairs without any further knowledge of the camera pipeline by using a normalizing flow. This approach has three advantages. First, once the model is trained to replicate a given noise corresponding to a fixed camera, we can apply this noise to every image we need. Compared to other generative methods, our model is fully stochastic and learns the full noise distribution. Finally, our noise generator not only captures the distribution of the noise but also all the related image artifacts introduced by the camera pipeline blocks such as demosaicing.
The idea of normalizing flows is to train an invertible network to encode a given data distribution, in this case the image noise, into a Gaussian latent space by minimizing the negative log-likelihood. To construct our flow architecture, we modify the blocks introduced in [21] by injecting the clean image and the noise intensity into the network. The whole architecture is summarized in Figure 1. Given a noisy image y at a given ISO, its clean version x and an invertible flow f_θ such that $f_\theta(y; x, ISO) = z$, using the change of variable formula, the log-likelihood is defined as follows:

$$\begin{aligned}\mathcal{L}(\theta; x, y, ISO) = &-log(p_z(f_\theta(y; x, ISO))) \\ &- log(|det\frac{\partial f_\theta(y; x, ISO)}{\partial y}|)\end{aligned} \tag{3}$$

Using the fact that $f_\theta(.)$ is the composition of N layers with $h^{n+1} = f^n_\theta(h^n; x, ISO)$ and $h^0 = y$, by applying the chain rule and the multiplicative property of the determinant in the second term we obtain:

$$\begin{aligned}\mathcal{L}(\theta; x, ISO, y) = &-log(p_z(f_\theta(y; x, ISO))) \\ &- \sum_{i=1}^{n} log(|det\frac{\partial f^n_\theta(h^n; x, ISO)}{\partial h^n}|)\end{aligned} \tag{4}$$

Working with layers which have a determinant which is fast and easy to compute helps us to calculate this loss efficiently. Once the network is trained, we sample z from a Gaussian distribution and obtain a noisy image $\tilde{y}$ from its clean version with $\tilde{y} = f_\theta(z; x, ISO)$. By sampling the latent space, we can generate different noise samples. We train our flow architecture, CNFlow, on the SIDD [31] dataset using the clean/noisy pairs at ISO 400, 800, 1600, 3200. Figure 2 illustrates the fidelity of the noise generated with CNFlow (right) from the clean image (left) compared to the respective noisy SIDD image (middle).

(a) Clean (b) SIDD (c) CNFlow

Fig. 2. Comparison between SIDD noise and CNFlow output at ISO 1600

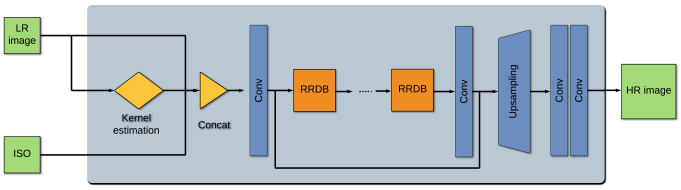


Fig. 3. Method architecture

3.4. Super-resolution model

3.4.1. Generator architecture

Our blind architecture can be considered as a non-blind architecture combined with a kernel prediction, as shown in Figure 3. Non-blind SR methods assume that the downsampling kernel and ISO level are known. In real world applications, the blur kernels is not available, so we build a kernel predictor which estimates the blur kernel on top of our non-blind model.

Non-blind: The non-blind architecture is based on RRDB blocks [3], and a kernel encoder which injects the blur kernel information into the SR network, reducing the dimension of the blur kernel to a smaller size $k_{featuresize}$. In [16], PCA is employed to encode the blur kernel, which works well in the case of Gaussian blur but fails to capture complex features from motion blur kernels. To solve this issue, we use a two layers linear neural network as our encoder. We train the kernel encoder in an end-to-end manner so that the model learns to extract only meaningful features to optimize SR results. The resulting encoded kernel is concatenated with the noise level information and then stretched to $k_{featuresize} + 1$ constant maps of the image size. These maps are then concatenated to the image and fed to the super-resolution network.

Blind: In real world applications the blur and downsampling kernels are unknown. Building an efficient non-blind network with a poor kernel prediction leads to artifacts in the super-resolved image [19]. Moreover, current kernel prediction methods such that KernelGAN are not robust to real noise. To remedy this issue, we train a kernel predictor on our dataset of blurry and noisy LR images, see Figure 3. Since our restoration network only takes as input the encoded version of the kernel, the kernel-predictor network predicts the encoded kernel directly instead of the full dimension kernel. This strategy yields an overall lighter SISR architecture. More specifically, let $E_{\hat{\theta}}(.)$ be the trained kernel encoder described in the non-blind section above, and $P(.)$ our kernel predictor. $P(.)$ is trained to minimize the following loss:

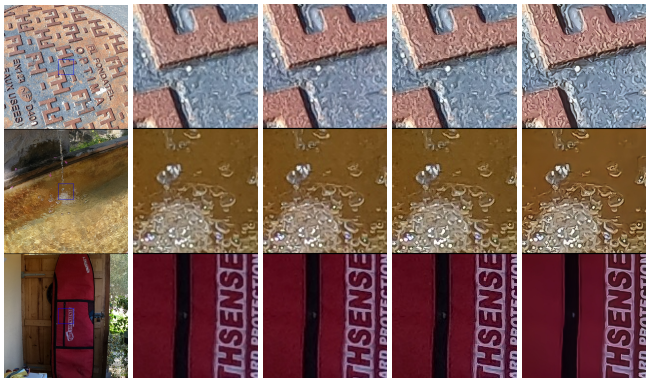
$$\mathcal{L}(HR) = \|P(LR) - E_{\hat{\theta}}(k)\|_2 \tag{5}$$

with $LR = (HR \circledast k)\downarrow_s + \epsilon(ISO)$.
Our kernel predictor is composed of 5 convolutional blocks followed by a global pooling layer. Once trained, the kernel predictor replaces the kernel encoder in our architecture and takes the LR image instead of the kernel as input.

4. EXPERIMENTS

In this section, we compare our model with ESRGAN [3], SRMD [16] and IKC [19]. We use both synthetic data and real images to compare the efficiency of our model. In particular, we demonstrate

that our model works on degraded images such as low-light images where all the other models fail to produce correct results. Please refer to the website of the article for an ablation study.



(a) LR (a) Bicubic (a) ESRGAN (a) IKC (a) Ours

Fig. 4. Comparison of results on GoPro images.

4.1. Testing dataset

SISR algorithms are often evaluated on synthetic datasets since capturing perfectly aligned HR/LR pairs is very challenging. These datasets are generally built using anistropic and istropic Gaussian blur, bicubic downsampling and AWGN. However, a model that performs well on synthetic images will not necessarily perform well on real images. Then, we evaluate our model both on synthetic data and real images. To test the robustness of our model to several degradations, we use the DIV2K [32] validation set and generate three sub-datasets of LR images. In particular, the three degradation types are:

- *Type 1)*: Anisotropic and Isotropic Gaussian blur + Bicubic downsampling
- *Type 2)*: Motion blur + Bicubic downsampling
- *Type 3)*: Anisotropic and Isotropic Gaussian blur + Bicubic downsampling + Camera Noise at ISO 800

We use bicubic downsampling for a fair comparison since the other models are trained only with bicubic downsampling. To test our model on images without downsampling, we use two different testsets: the SIDD_small and the GoPro_small datasets. SIDD_small is composed of noisy images at several ISOs. We use all the images with ISO values between 100 and 1600. The GoPro_small dataset contains images captured with a GoPro HERO8 camera. These images are very challenging for SR algorithms since they have natural blur and noise due to shooting conditions such as low-light or moving objects.

4.2. Results

Synthetic data: Table 1 shows PSNR, SSIM and LPIPS [33] scores of different methods on the 3 DIV2K synthetic datasets. Generally speaking, our algorithm performs similarly or better than the other state-of-the-art methods. On the *Type 1)* sub-dataset, IKC performs better on all metrics since it was trained on bicubic and Gaussian blur. It also benefits from its iterative scheme. On the other hand, our model outperfoms ESRGAN and SRMD in terms of PSNR,

Degradation	Method	PSNR↑	SSIM↑	LPIPS↓
Type 1	Bicubic x4	23.42	0.62	0.57
	ESRGAN x4	24.57	0.68	0.4
	SRMD x4	24.08	0.7	0.4
	IKC x4	25.41	0.75	0.37
	Ours x4	24.79	0.7	0.42
Type 2	Bicubic x4	22.62	0.58	0.59
	ESRGAN x4	23.16	0.61	0.39
	SRMD x4	19.26	0.52	0.48
	IKC x4	23.14	0.63	0.44
	Ours x4	23.25	0.62	0.46
Type 3	Bicubic x4	21.21	0.49	0.78
	ESRGAN x4	19.96	0.35	0.68
	SRMD x4	19.05	0.37	0.69
	IKC x4	20.9	0.44	0.73
	Ours x4	22.5	0.57	0.52

Table 1. Results on the different DIV2K synthetized datasets. Best scores are shown in red, whilst second best are in blue. See 4.1 for an explanation on the different degradations.

even though SRMD is trained on Gaussian blur and bicubic downsampling. This fact shows that our model achieves similar performance on Gaussian blur while being trained on more diverse degradations. On *Type 2)*, our model obtains the best PSNR. Although IKC presents the best SSIM and ESRGAN the best LPIPS, such models tend to produce artifacts while performing SR on motion-blurred images. Our model does not completely deblur the image but does not produce artifacts, which is a crucial aspect for practical use cases. Finally, on *Type 3)*, only our model is able to produce clean SR images. All other models are trained only on Gaussian noise which is ineffective in the case of real camera noise. They also tend to oversharp noise artifacts, which renders their results visually unappealing. On the contrary, our model is capable of jointly super-resolving and denoising the input image, and outperforms the other algorithms by a large margin. Another important point is that our architecture has roughly half the number of parameters than ESRGAN's and runs up to 8X faster than the iterative IKC.

Real world images: When the shooting conditions are not ideal, for example in low-light or when the camera is moving, many super-resolution algorithms fail to perform effective upsampling. Figure 4 demonstrates an example of such case with images from the GoPro_small dataset. Residual noise is notable in the results by bicubic interpolation, ESRGAN and IKC, whereas our model denoises the images well. Similar observations, in general favorable to our method, can be seen in images of the SIDD dataset, please see the webpage of the algorithm for more examples.

5. CONCLUSION

In this paper, we present a practical degradation pipeline for the training of SR models, which mimics degradations found in real world SR. Thanks to the increased fidelity of the generated training data, our SR algorithm is robust to degradations observed in real world applications. The combination between its restoration performance and lower computational load makes this algorithm attractive for practical super-resolution applications. In summary, the proposed approach offers a feasible solution toward practical super-resolution applications.

6. REFERENCES

[1] Y. Chen, Y. Tai, X. Liu, C. Shen, and J. Yang, "FSRNet: End-to-End Learning Face Super-Resolution with Facial Priors," in *CVPR*, 2018, pp. 2492–2501.

[2] Y. Tai, J. Yang, and X. Liu, "Image Super-Resolution via Deep Recursive Residual Network," in *CVPR*, 2017, pp. 2790–2798.

[3] W. Xintao, Y. Ke, W. Shixiang, G. Jinjin, L. Yihao, D. Chao, Q. Yu, and L. Chen Change, "Esrgan: Enhanced super-resolution generative adversarial networks," in *ECCV Workshops*, September 2018.

[4] C.Y. Yang, C. Ma, and M.H. Yang, "Single-image super-resolution: A benchmark," in *Lecture Notes in Computer Science*, 2014, vol. 8692 LNCS.

[5] N. Efrat, D. Glasner, A. Apartsin, B. Nadler, and A. Levin, "Accurate blur models vs. image priors in single image super-resolution," in *ICCV*, December 2013.

[6] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani, "Blind super-resolution kernel estimation using an internal-gan," *Advances in Neural Information Processing Systems*, vol. 32, pp. 284–293, 2019.

[7] T. Michaeli and M. Irani, "Nonparametric Blind Super-Resolution," in *ICCV*, 2013.

[8] Daniel Glasner, Shai Bagon, and Michal Irani, "Super-resolution from a single image," in *ICCV*, 2009, pp. 349–356.

[9] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, USA, 2011, CVPR '11, p. 977–984, IEEE Computer Society.

[10] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *TPAMI*, pp. 1–1, 2020.

[11] C. Dong, C. Change Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[12] L. Bee, S. Sanghyun, K. Heewon, N. Seungjun, and L. Kyoung Mu, "Enhanced deep residual networks for single image super-resolution," in *CVPR Workshops*, July 2017.

[13] Z. Yulun, L. Kunpeng, L. Kai, W. Lichen, Z. Bineng, and F. Yun, "Image super-resolution using very deep residual channel attention networks," in *ECCV*, 2018.

[14] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *CVPR*, June 2016.

[15] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *ICCV*, Oct 2017.

[16] Zhang Kai, Zuo Wangmeng, and Zhang Lei, "Learning a single convolutional super-resolution network for multiple degradations," in *CVPR*, 2018, pp. 3262–3271.

[17] 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 *ICCV*, October 2019.

[18] Assaf Shocher, Nadav Cohen, and Michal Irani, ""zero-shot" super-resolution using deep internal learning," in *CVPR*, June 2018.

[19] G. Jinjin, L. Hannan, Z. Wangmeng, and D. Chao, "Blind super-resolution with iterative kernel correction," in *CVPR*, June 2019.

[20] J. Xiaozhong, C. Yun, T. Ying, W. Chengjie, L. Jilin, and H. Feiyue, "Real-world super-resolution via kernel estimation and noise injection," in *CVPR Workshops*, June 2020.

[21] A. Lugmayr, M. Danelljan, L. Van Gool, and R. Timofte, "Srflow: Learning the super-resolution space with normalizing flow," in *ECCV*, 2020.

[22] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville, "Neural autoregressive flows," in *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause, Eds. 10–15 Jul 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 2078–2087, PMLR.

[23] Diederik P Kingma and Prafulla Dhariwal, "Glow: generative flow with invertible $1\times$ 1 convolutions," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 10236–10245.

[24] A. Abdelhamed, M. A. Brubaker, and M. S Brown, "Noise Flow: Noise Modeling with Conditional Normalizing Flows," in *ICCV*, 2019.

[25] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, Axel Chemla, et al., "Universal audio synthesizer control with normalizing flows," in *International Conference on Digital Audio Effects (DaFX 2019)*, 2019.

[26] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *CVPR*, 2019, pp. 4541–4550.

[27] Tamara Seybold, "Noise Characteristics and Noise Perception," in *Denoising of Photographic Images and Video: Fundamentals, Open Challenges and New Trends*, Marcelo Bertalmío, Ed., pp. 235–265. Springer International Publishing, 2018.

[28] Tobias Plotz and Stefan Roth, "Benchmarking denoising algorithms with real photographs," in *CVPR*, July 2017.

[29] F. Gavant, L. Alacoque, A. Dupret, and D. David, "A physiological camera shake model for image stabilization systems," in *SENSORS, 2011 IEEE*, 2011, pp. 1461–1464.

[30] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[31] A. Abdelhamed, S. Lin, and M.S. Brown, "A high-quality denoising dataset for smartphone cameras," in *CVPR*, June 2018.

[32] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *CVPR Workshops*, July 2017.

[33] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.