Non-parametric Sensor Noise Modeling and Synthesis (Supplementary Material)

Ali Mosleh[1], Luxi Zhao[1], Atin Singh[2], Jaeduk Han[2], Abhijith Punnappurath[1], Marcus A. Brubaker[1,3], Jihwan Choe[2], and Michael S. Brown[1]

[1] Samsung AI Center Toronto, Canada
[2] Samsung Electronics, South Korea
[3] York University, Canada

This supplementary material contains additional details regarding the proposed non-parametric sensor noise modeling method and experimental assessments we presented in the main document.

S1 Additional Details of Image Capturing Process

As explained in Sec. 4, our calibration process is performed in a controlled environment. Noise samples are collected from imaging of a single target scene rather than extensive captures of a wide variety of scenes. This allows us to automate collecting noise samples. Various pixel intensities are obtained via varying the exposure value (EV) of the camera with no need to change the target scene. We modified the Camera2API software (Fig. S1 shows its user interface) to take a list of ISOs (including ISO 50 to serve in preparing noise-free captures), the total number of EVs per ISO (e.g., 9 indicating a list of integers with step size 1 centred at 0 corresponding to EVs -4, -3,...,3, and 4), and the number of captures (e.g. 100) per ISO-EV setting. The capture process begins with a single click and iterates over these settings automatically. This results in sufficient noise samples (over 200K samples per intensity in our experiments) to generate noise PMFs with negligible labour cost.

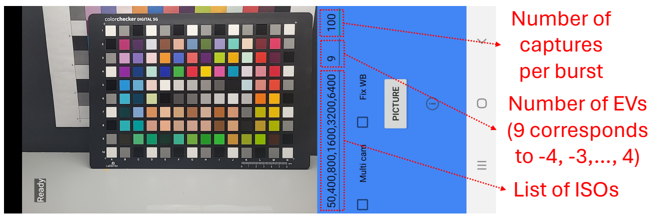
100
9
50,400,800,1600,3200,6400
Fix WB
Multi cam
PICTURE
Ready
Number of captures per burst
Number of EVs (9 corresponds to -4, -3,..., 4)
List of ISOs

Fig. S1: Capture of calibration data is a single click. This particular phone screenshot shows that the setup is ready to capture images with 6 different ISOs (separated by comma), 9 different exposure values (EVs) as -4, -3, ..., 3, 4 and bursts of 100 images per ISO-EV combination through a click on "PICTURE".

S2 Details of Inversion Sampling

Although random variable generation through inversion sampling is quite popular and straightforward [9], we discuss more details about RandomSampling(.) introduced in Eq. 3 to allow our work to be reproduced.

Inverse transform sampling, or inversion sampling, is a method for generating random numbers from any probability distribution by using its inverse cumulative distribution (CDF). Let $X \in \mathbb{R}$ denote a random variable whose distribution can be characterized as $f_X(x)$ with a CDF as $F_X(x)$. In order to randomly sample from the original distribution $f_X(x)$, we first form the inverse of the CDF denoted by $F_X^{-1}(x)$. Since the range of $F_X(x)$ is determined by probabilities (*i.e.*, $F_X(x) \in [0,1]$), the domain of $F_X^{-1}(x)$ is defined uniformly on $[0,1]$. Thus, assuming that $U \sim \text{Unif}[0,1]$ is a random variable from a uniform distribution on $[0,1]$, we can generate a random variable X as $X = F_X^{-1}(U)$.

The following Python code shows the inversion sampling process to synthesize one noise sample for a given noise probability mass function (PMF) computed from around 200000 noise samples:

Listing S1: Simplified illustrative Python code for noise synthesis through inversion sampling.

```
import numpy as np
from scipy.interpolate import  interp1d

num_bins = 240

# X: around 200000 noise samples
noise_histogram, bin_edges = np.histogram(X, bins=num_bins, density=True)
pmf_X = noise_histogram * np.diff(bin_edges)

# RandomSampling(pmf_X)
cdf_X = np.zeros(bin_edges.shape)
cdf_X[1:] = np.cumsum(pmf_X)
inv_cdf_X = interp1d(cdf_X, bin_edges)

n_samples = 1
U = np.random.uniform(0,1,n_samples)
X_prime = inv_cdf_X(U) # random noise
```

Fig. S2 shows the noise distribution obtained through our noise calibration setup for an instance of ISO/intensity level of S22+ sensor. The actual noise samples are used to build an inverse CDF of the noise distribution. Next, we use the inversion process to generate 10000 noise samples using the code in Listing S1. The distribution plot of the synthetic noise obtained through inversion sampling is quite similar to that of the actual sensor noise. See the plot corresponding X'. In this figure, we use a general form of notations for random variables, PMFs, etc. X and $f_X(\cdot)$, however, correspond to any arbitrary calibrated noise sample set and PMF, *i.e.*, ξ_κ^l and $p_{\xi_\kappa^l}(\cdot)$, respectively.

S3 Noise Synthesis Evaluation using Traditional KLD

To maintain consistency with traditional statistical evaluations for noise modeling, we also report the Kullback-Leibler Divergence (KLD) for our noise synthesis

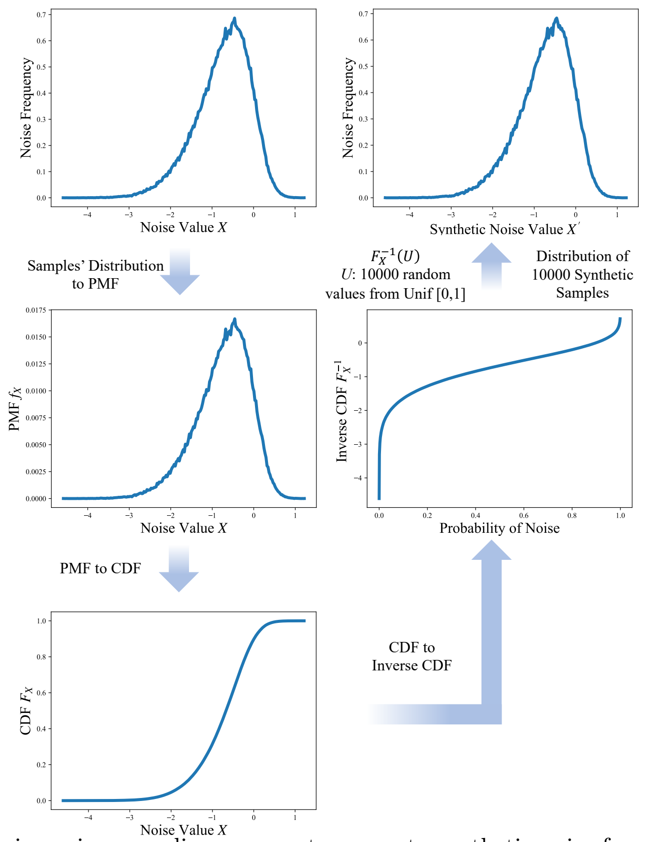
Noise Frequency
Noise Value X
Synthetic Noise Value X'
Samples' Distribution to PMF
$F_X^{-1}(U)$
U: 10000 random values from Unif [0,1]
Distribution of 10000 Synthetic Samples
PMF f_X
Inverse CDF F_X^{-1}
Probability of Noise
PMF to CDF
CDF to Inverse CDF
CDF F_X

Fig. S2: The inversion sampling process to generate synthetic noise from a calibrated noise model.

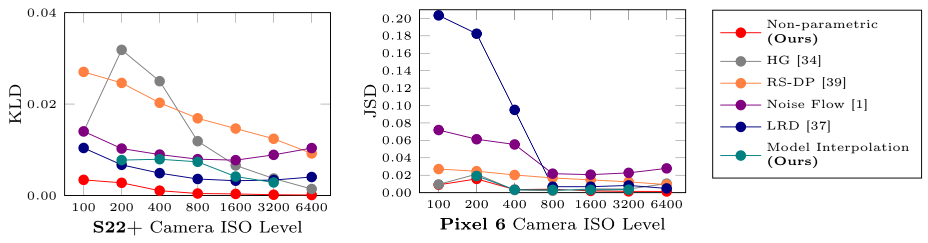

Fig. S3: Statistical analyses per ISO sensor noise modeling for both S22+ and Pixel 6 cameras using traditional KLD. KLD is measured relative to the real noise distribution obtained from homogeneous patches of the color checker chart images. (Examples of such images are shown in Fig. S4 and S5 for ISOs 1600, 3200, and 6400.)

experiments, as shown in Fig. S3. KLD has been traditionally used to quantify the similarity between the synthesized noise and real noise [1]. This is done by obtaining noise distributions in terms of PMFs, P and Q, for real and synthetic noise, respectively and finding the relative entropy of the two probability functions. Noise distributions P and Q are obtained as pixel values extracted from the 24 homogeneous patches of the color checker in the clean image subtracted from the corresponding pixels in the real and the synthetic images shown in Fig. S4. Therefore, KLD can be computed as

$$D_{KL}(Q||P) = -\sum_{n} Q(n) \ln \frac{P(n)}{Q(n)}, \tag{1}$$

where $D_{KL}(Q||P) \geq 0$ denotes KLD between the two probability distributions measured in nats. When the score is zero, it suggests that both P and Q are identical. A major drawback of KLD is that it is not symmetrical, *i.e.* $D_{KL}(Q||P) \neq D_{KL}(P||Q)$, which is counterintuitive considering the nature of noise distributions. Also, it does not have an upper bound. Thus, a more practical way of computing how the synthetic noise diverges from the real noise is the Jensen-Shannon divergence (JSD). JSD uses the KLD to compute a normalized and symmetrical score as

$$D_{JS}(Q||P) = 0.5D_{KL}(Q||M) + 0.5D_{KL}(P||M), \tag{2}$$

where $M = 0.5(P + Q)$ and $0 \leq D_{JS}(Q||P) \leq 1$. In our experiments, we use $\sqrt{D_{JS}(Q||P)}$ as a metric of the distance between two distributions as reported in Fig. 4.

The plots in Fig. S3 follow a similar trend compared to the plots of JSD in Fig. 4, except for the results obtained using Noise Flow for ISOs 100, 200, 400 in Pixel 6 experiments, where KLD values show a larger divergence compared to the JSD values.

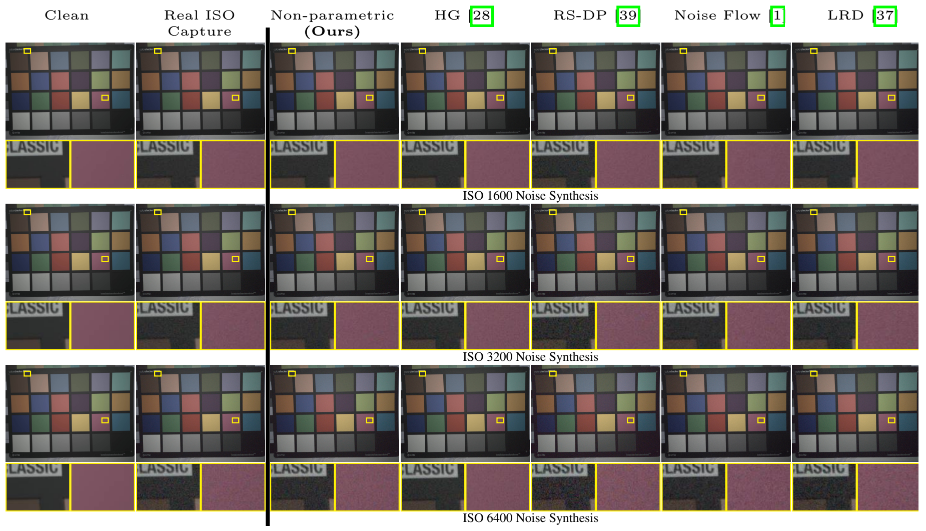

Fig. S4: Noise synthesis examples using different methods for **S22+** camera. Noise distribution is computed as pixel values extracted from the 24 homogeneous patches of the color checker in the clean image subtracted from the corresponding pixels in the real and synthesized noisy images. The plots of JSD in Fig. 4 and KLD in Fig. S3 are computed relative to real ISO noise distribution.

S4 More Details on Training Setup for Denoising-demosaicking

In Sec. 5.1, we explained how we generate synthetic training data using EXR graphics data and noise models. To further detail the training setup for the model in [7] to perform joint denoising-demosaicking for all experiments, including assessing the baseline noise models, per training iteration, we randomly pick and flip a batch of 16 EXR images and crop them to 512×512. We then perform raw simulations by applying randomly selected noise models calibrated for the seven ISO levels $\{2^m 100 | m \in \mathbb{N}, 0 \leq m \leq 6\}$. Fig. S6 shows an example of such data synthesis for one of the EXR images where Fig. S6a and Fig. S6c correspond to our ground-truth and input raw images, respectively.

In all experiments, we minimize the distance between the ground-truth RGB data and the output of the denoising-demosaicking model measured with the ℓ_1-norm. Our code-base is in Pytorch, and we use Adam optimizer, initiated with a learning rate of 5×10^{-4}, scheduled to decay by half after every 300 epochs. The training runs for a total of 2000 epochs for each experiment.

The only exception is the noise synthesis experiment using Noise Flow. Noise Flow was originally implemented in TensorFlow [1]. As a result, it is not straightforward to use Noise Flow directly in our Pytorch setup in the aforementioned noise synthesis process during training runs. Thus, we process all of the 292 EXR images directly using the Noise Flow model at seven ISO levels and save all the resulting 2044 simulated raw images paired with their corresponding ground-truth RGB images as our training set. To be consistent with the other training

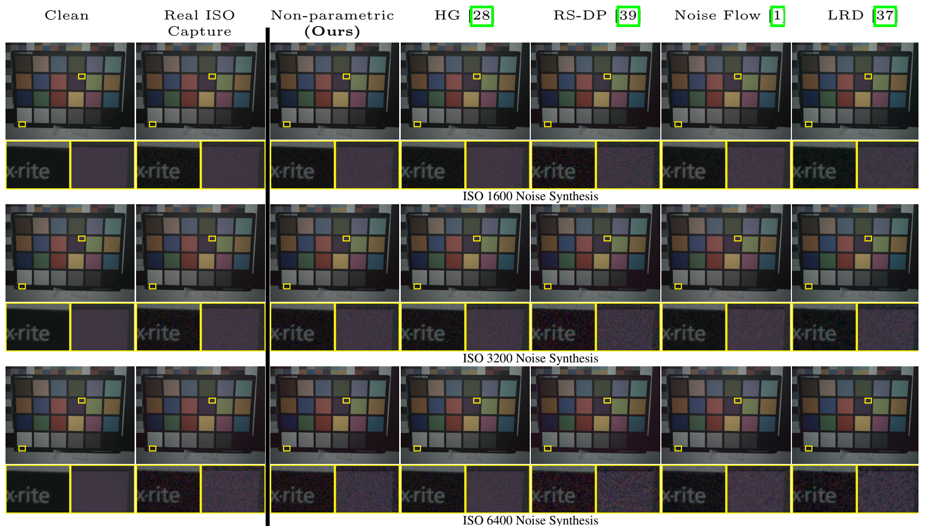

Fig. S5: Noise synthesis examples using different methods for **Pixel 6** camera. Noise distribution is computed as pixel values extracted from the 24 homogeneous patches of the color checker in the clean image subtracted from the corresponding pixels in the real and synthesized noisy images. The plots of JSD in Fig. 4 and KLD in Fig. S3 are computed relative to real ISO noise distribution.

experiments, we run 285 epochs over the 2044 raw images. All other training parameters such as batch size, crop size, learning rate, etc., are similar to other experiments. To account for the randomness, we run this experiment three times, and the best result is reported in Table 1.

S5 Additional Qualitative Results for Denoising-demosaicking Experiments

In Sec. 5.1 of the main paper, we showed three S22+ raw captures processed with the denoising-demosaicking model trained using different synthesized training sets (Fig. 5). In this section, we show a few more qualitative results from real evaluation sets for S22+ in Fig. S7 and Pixel 6 in Fig. S8.

S6 Additional Details of Applying Noise on Noisy Images

The steps of the process explained in Sec. 4.2 are listed in Alg. S1. For ISO κ our noise model is a set of PMFs per l, *i.e.*, $\{p_{\xi_\kappa^0}(n), \ldots, p_{\xi_\kappa^L}(n)\}$. The inputs to Alg. S1 are non-parametric noise models for two different ISOs ($\kappa = 1$ and $\kappa = 2$), the ISO 1 capture $\tilde{I}_1$, and thus no clean image is needed. Since an underlying/clean image I is not available, we need some approximation to infer the noise level from $\tilde{I}_1$. As a result, we train a raw-to-raw denoiser $D(\cdot)$ specific to ISO 1 using simulated ISO 1 images and $\{p_{\xi_1^0}(n_1), \ldots, p_{\xi_1^L}(n_1)\}$. In step 2, a clean

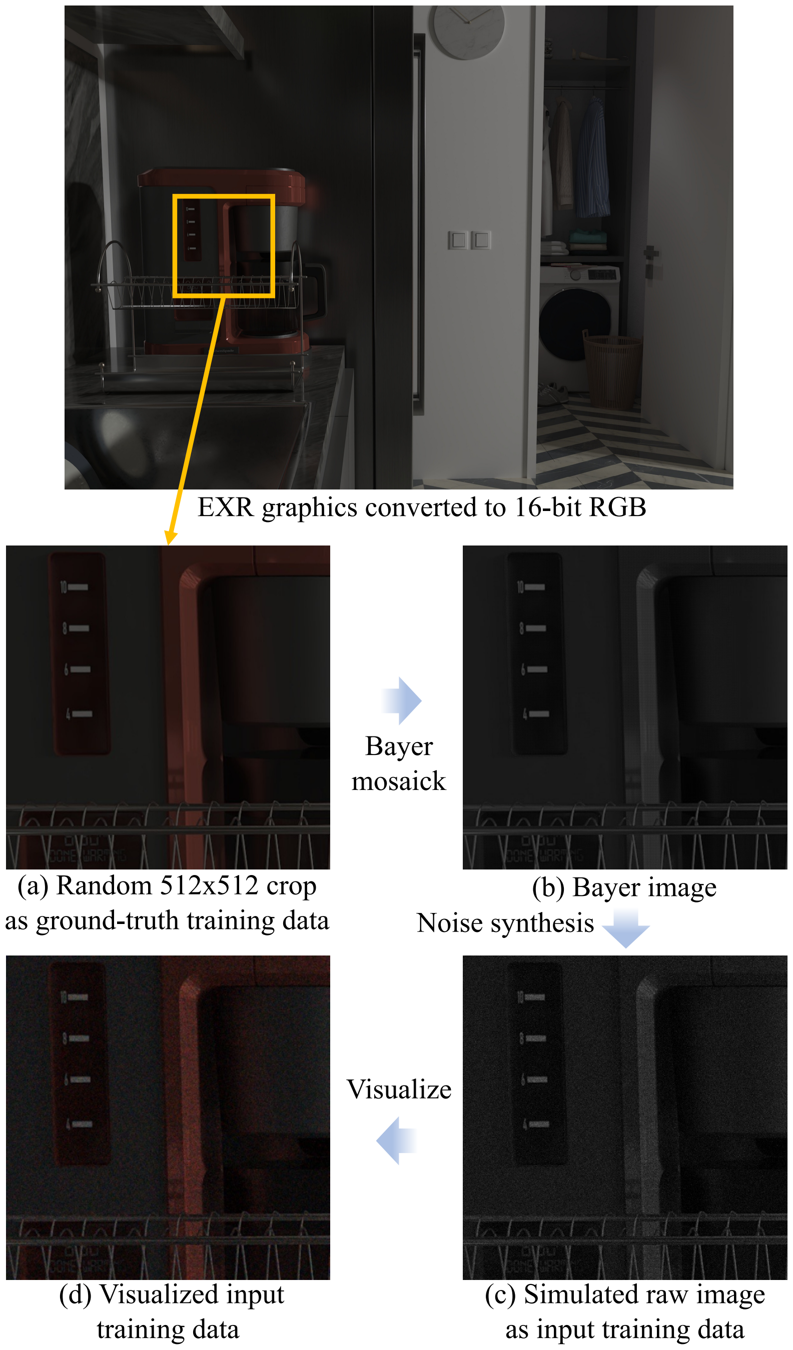
EXR graphics converted to 16-bit RGB
Bayer mosaick
(a) Random 512x512 crop as ground-truth training data
(b) Bayer image
Noise synthesis
Visualize
(d) Visualized input training data
(c) Simulated raw image as input training data

Fig. S6: Synthesizing raw training data for denoising-demosaicking [7] using EXR graphics images and calibrated sensor noise models. (The synthesized raw image is processed by a bi-linear demosaicker followed by white-balancing for visualization.)

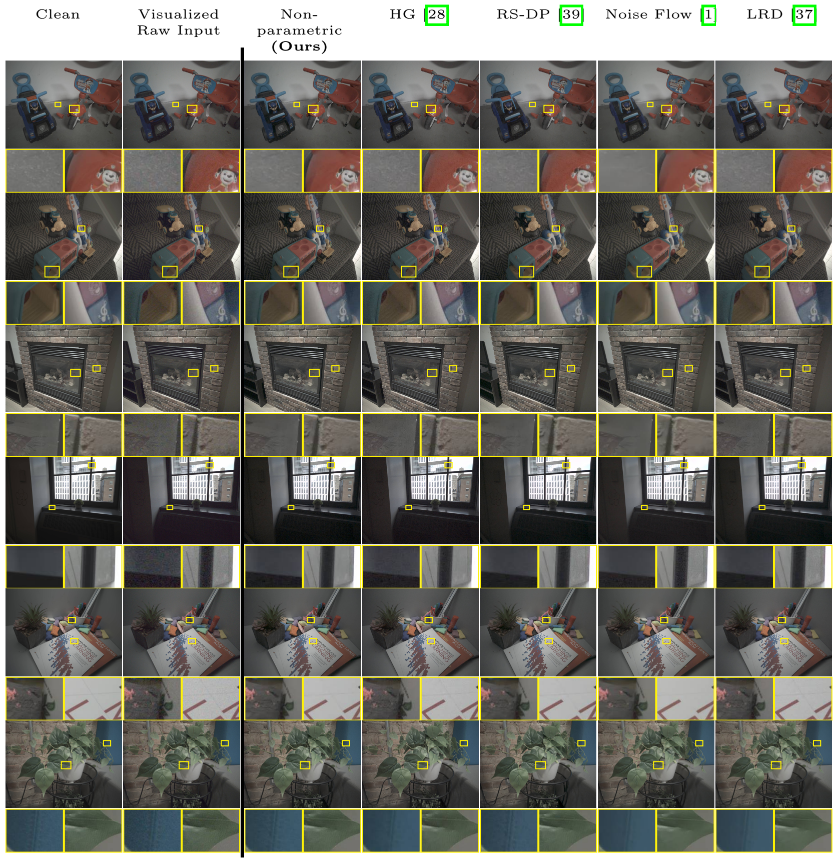

Fig. S7: Denoising-demosaicking qualitative results on **S22+** raw captures. We use different noise synthesis methods applied on EXR images [28] to create training images for the denoising-demosaicking model [7]. Our non-parametric noise models help the image reconstruction model remove more noise while preserving more details. Raw inputs are converted to RGB using bi-linear demosaicking for visualization here. Images in each row are white-balanced and gamma-corrected for better visualization.

estimate ($\hat{I} \approx I$) is obtained for inferring noise level and clean intensity (step 4). To synthesize noise per intensity l, the noise PMF of ISO 2 ($p_{\xi_2^l}(n_2)$) is shifted by the inferred noise level from ISO 1, a random sample is drawn from this new PMF (step 5). See Sec. S8, for experiments of different ways of obtaining $\hat{I}$. Note that We do not have access to the clean image I (unlike Eq. (3)), but an estimate $\hat{I}$. We showed that we cannot simply use ISO 2 models on denoised ISO 1 to synthesize ISO 2 images—*i.e.*, $\tilde{I}_2(i) = \hat{I}(i) + \text{RandomSampling}(p_{\xi_2^l}(n_2))$ is not equivalent to $\tilde{I}_2(i) = I(i) + \text{RandomSampling}(p_{\xi_2^l}(n_2))$ since $D(\cdot)$ may remove some details in $\hat{I}$. But, $\hat{I}$ can be used to infer the noise level and modify noise models as Eq. (7) and step 5 of Alg. S1.

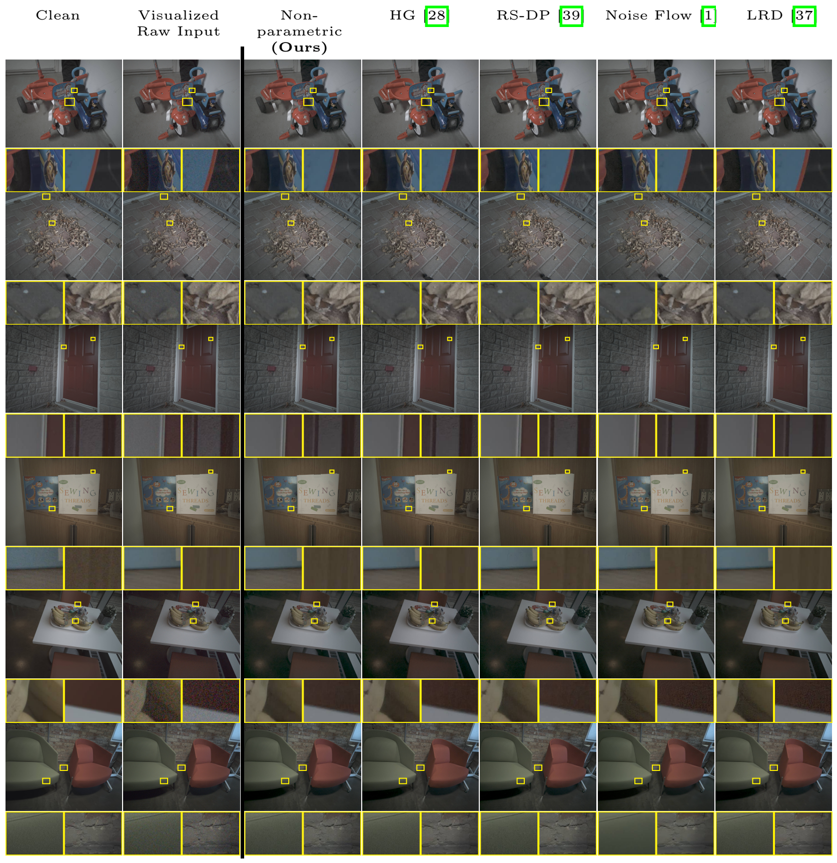

Fig. S8: Denoising-demosaicking qualitative results on **Pixel 6** real raw captures. We use different noise synthesis methods applied on EXR images [28] to create training images for the denoising-demosaicking model [7]. Our non-parametric noise models help the image reconstruction model remove more noise while preserving more details. Raw inputs are converted to RGB using bi-linear demosaicking for visualization here. Images in each row are white-balanced and gamma-corrected for better visualization.

S6.1 Details on Training Setup for Noise-on-noise Assessments

As mentioned in Sec. 5.3, we train a UNet to perform raw denoising for S20FE ISO 1600 raw images–*i.e.*, $D(\cdot)$. This denoiser is trained using simulated raw images using the EXR graphics data with ISO 1600 non-parametric noise models. Thus, a pair of ground-truth and raw data from the training set would be the images shown in Fig. S6b and Fig. S6c. Hence, the denoiser network takes a single-channel noisy raw input and outputs a single-channel raw. The other training parameters are similar to the ones described in Sec. S4.

For training the neural ISP, we once again use a UNet architecture. The UNet takes a noisy raw image as input and outputs an sRGB image. We trained the model for 4000 epochs with a learning rate of 0.0001 and a batch size of 8 on

Algorithm S1 Synthesizing noise on noisy raw images

```
Require: $\tilde{I}_1$, $\{p_{\xi_1^0}(n_1),\ldots,p_{\xi_1^L}(n_1)\}$, $\{p_{\xi_2^0}(n_2),\ldots,p_{\xi_2^L}(n_2)\}$
1: $\hat{I}=D(\tilde{I}_1)$                    ▷ Learning $D(\cdot)$ is done with simulated data only.
2: for all pixels indexed by $i$ do
3:     $l=\hat{I}(i)$ , $n_1=\tilde{I}_1(i)-\hat{I}(i)$
4:     $\tilde{I}_2(i)=\tilde{I}_1(i)+\text{RandomSampling}(p_{\xi_2^l}(n_2-n_1))$
5: end for
6: return $\tilde{I}_2$
```

patches of size 256×256 pixels. We used the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

S7 Additional Qualitative Results for Neural ISP

In this section, we provide more qualitative results regarding the experiment presented in Sec. 5.3. The qualitative results obtained for the neural ISP trained using different datasets are shown in Fig. S9. The inputs to the trained models are S20FE raw images captured at ISO 3200.

S8 Ablation Studies for our Noise-on-noise Method

In this section, we present different experiments to support the details discussed in Sec. 4.2 and Sec. 5.3 regarding our noise-on-noise augmentation approach.

We first replace the data-driven UNet denoiser with the local block-matching-based BM3D denoiser to evaluate the effects of the choice of denoiser to approximate clean intensity for this noise synthesis solution. As Table S1 (the second row) shows, using BM3D lowers the task performance, indicating that the proposed noise-on-noise model relies on accurate denoisers to approximate $\hat{I}$. However, these experiments also suggest that such accurate raw denoisers can be obtained without real captures, almost at no cost, using only graphics data and calibrated ISO 1600 noise models.

Since our approach employs a denoiser to approximate clean intensities, one may argue why we do not apply the calibrated ISO 3200 noise model directly on the approximated clean intensity images. Table S1 (the third row) shows the performance of the neural ISP once trained using such a synthetic dataset. The training set is generated using the calibrated ISO 3200 noise model applied on denoised ISO 1600 images. Not surprisingly, the performance is much lower than correctly applying the noise-on-noise model on ISO 1600 images. This low performance can be attributed to the loss of details and other perturbations imposed on the underlying image due to denoising.

Lastly, we highlight that the proposed solution to applying noise on top of noisy raw captures can be employed with any noise modeling approach that provides noise model PMFs, such as the HG model. However, the accuracy of noise synthesis using this approach depends on the accuracy of the calibrated

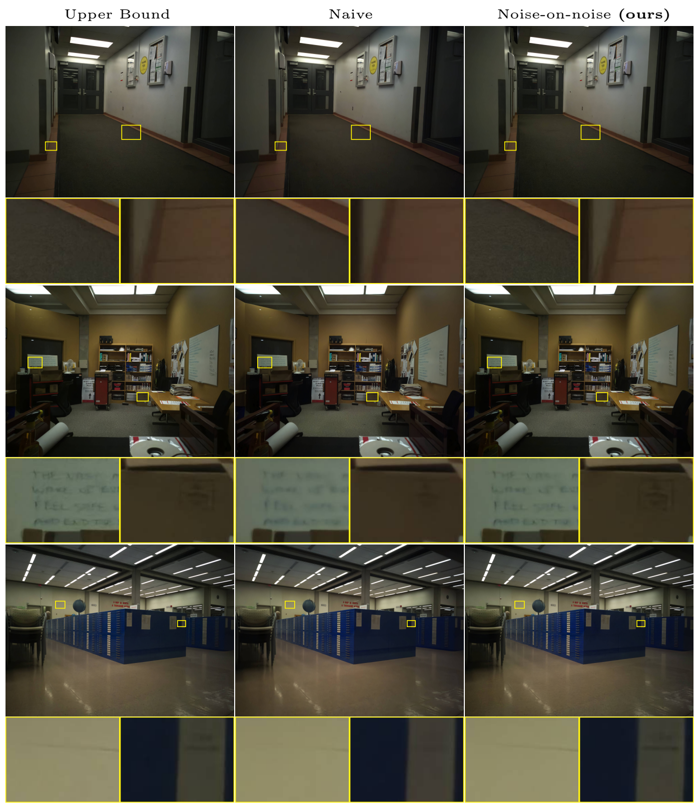

Fig. S9: Neural ISP applied on S20FE raw captures at ISO 3200. The upper bound corresponds to training with real captures at ISO 3200. We use different noise synthesis methods applied to ISO 1600 images to create ISO 3200 training images for raw to sRGB rendering. Our noise-on-noise model helps the neural ISP preserve more details compared to naively applying calibrated ISO 3200 noise models to ISO 1600 images. Note the over-smoothing effects and the loss of details in the experiment with directly applying ISO 3200 noise model on ISO 1600 raw images (Naive).

noise model. See the task performance metrics obtained for the neural ISP once trained with the images synthesized using the proposed noise-on-noise approach with HG noise PMFs (the last row in Table S1). These numbers and the JSD computed for this method show how the synthetic noise distribution diverges from the real captured noise compared to using our calibrated noise models.

References

1. Abdelhamed, A., Brubaker, M.A., Brown, M.S.: Noise flow: Noise modeling with conditional normalizing flows. In: ICCV (2019)
2. Abdelhamed, A., Lin, S., Brown, M.S.: A high-quality denoising dataset for smartphone cameras. In: CVPR (2018)
3. Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., Barron, J.T.: Unprocessing images for learned raw denoising. In: CVPR (2019)

Table S1: Ablation studies for generating S20FE camera ISO 3200 training data given ISO 1600 images and our non-parametric (NP) noise models.

Noise Synthesis Approach			Neural ISP		JSD
ISO 3200 Noise Model	Denoiser	Input Image	PSNR	SSIM	
NP Noise-on-noise	UNet	ISO 1600	39.05	0.939	0.008
NP Noise-on-noise	BM3D	ISO 1600	36.60	0.893	0.061
NP Calibrated	UNet	Denoised ISO 1600	37.34	0.922	0.056
HG Noise-on-noise	UNet	ISO 1600	38.15	0.929	0.031

4. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input / output image pairs. In: CVPR (2011)
5. Cao, Y., Liu, M., Liu, S., Wang, X., Lei, L., Zuo, W.: Physics-guided ISO-dependent sensor noise modeling for extreme low-light photography. In: CVPR (2023)
6. Chang, K.C., Wang, R., Lin, H.J., Liu, Y.L., Chen, C.P., Chang, Y.L., Chen, H.T.: Learning camera-aware noise models. In: ECCV (2020)
7. Chen, C., Chen, Q., Xu, J., Koltun, V.: Learning to see in the dark. In: CVPR (2018)
8. Chen, J., Chen, J., Chao, H., Yang, M.: Image blind denoising with generative adversarial network based noise modeling. In: CVPR (2018)
9. Devroye, L.: Non-Uniform random variate generation. Springer New York, NY (1986)
10. El Gamal, A., Fowler, B.A., Min, H., Liu, X.: Modeling and estimation of FPN components in CMOS image sensors. In: Solid State Sensor Arrays: Development and Applications II. vol. 3301, pp. 168–177 (1998)
11. Feng, H., Wang, L., Wang, Y., Fan, H., Huang, H.: Learnability enhancement for low-light raw image denoising: A data perspective. IEEE TPAMI pp. 1–18 (2023)
12. Foi, A.: Clipped noisy images: Heteroskedastic modeling and practical denoising. Signal Processing **89**(12), 2609–2629 (2009)
13. Foi, A., Trimeche, M., Katkovnik, V., Egiazarian, K.: Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. IEEE TIP **17**(10), 1737–1754 (2008)
14. Gow, R.D., Renshaw, D., Findlater, K., Grant, L., McLeod, S.J., Hart, J., Nicol, R.L.: A comprehensive tool for modeling CMOS image-sensor-noise performance. IEEE Transactions on Electron Devices **54**(6), 1321–1329 (2007)
15. Hasinoff, S.W.: Photon, poisson noise. Computer Vision, A Reference Guide **4**(16), 1 (2014)
16. Hwang, Y., Kim, J.S., Kweon, I.S.: Difference-based image noise modeling using skellam distribution. IEEE TPAMI **34**(7), 1329–1341 (2011)
17. Ignatov, A., Van Gool, L., Timofte, R.: Replacing mobile camera ISP with a single deep learning model. In: CVPRW (2020)
18. Jang, G., Lee, W., Son, S., Lee, K.M.: C2N: Practical generative noise modeling for real-world denoising. In: ICCV (2021)
19. Kim, D.W., Ryun Chung, J., Jung, S.W.: GRDN: Grouped residual dense network for real image denoising and GAN-based real-world noise modeling. In: CVPRW (2019)
20. Konnik, M., Welsh, J.: High-level numerical simulations of noise in CCD and CMOS photosensors: Review and tutorial. arXiv (2014)

21. Liu, X., Tanaka, M., Okutomi, M.: Practical signal-dependent noise parameter estimation from a single noisy image. IEEE TIP **23**(10), 4361–4371 (2014)
22. Makitalo, M., Foi, A.: Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. IEEE TIP **22**(1), 91–103 (2012)
23. Maleky, A., Kousha, S., Brown, M.S., Brubaker, M.A.: Noise2NoiseFlow: Realistic camera noise modeling without clean images. In: CVPR (2022)
24. Monakhova, K., Richter, S.R., Waller, L., Koltun, V.: Dancing under the stars: Video denoising in starlight. In: CVPR (2022)
25. Ploetz, T., Roth, S.: Benchmarking denoising algorithms with real photographs. In: CVPR (2017)
26. Punnappurath, A., Abuolaim, A., Abdelhamed, A., Levinshtein, A., Brown, M.S.: Day-to-night image synthesis for training nighttime neural ISPs. In: CVPR (2022)
27. Rim, J., Lee, H., Won, J., Cho, S.: Real-world blur dataset for learning and benchmarking deblurring algorithms. In: ECCV (2020)
28. Seo, D., Punnappurath, A., Zhao, L., Abdelhamed, A., Tedla, S.K., Park, S., Choe, J., Brown, M.S.: Graphics2RAW: Mapping computer graphics images to sensor raw images. In: ICCV (2023)
29. Suh, S., Itoh, S., Aoyama, S., Kawahito, S.: Column-parallel correlated multiple sampling circuits for CMOS image sensors and their noise reduction effects. Sensors **10**(10), 9139–9154 (2010)
30. Tran, L.D., Nguyen, S.M., Arai, M.: GAN-based noise model for denoising real images. In: ACCV (2020)
31. Wang, W., Chen, X., Yang, C., Li, X., Hu, X., Yue, T.: Enhancing low light videos by exploring high sensitivity camera noise. In: ICCV (2019)
32. Wei, K., Fu, Y., Yang, J., Huang, H.: A physics-based noise formation model for extreme low-light raw denoising. In: CVPR (2020)
33. Wei, K., Fu, Y., Zheng, Y., Yang, J.: Physics-based noise modeling for extreme low-light photography. IEEE TPAMI **44**(11), 8520–8537 (2021)
34. Yoshimura, M., Otsuka, J., Irie, A., Ohashi, T.: Rawgment: Noise-accounted raw augmentation enables recognition in a wide variety of environments. In: CVPR (2023)
35. Yue, Z., Zhao, Q., Zhang, L., Meng, D.: Dual adversarial network: Toward real-world noise removal and noise generation. In: ECCV (2020)
36. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: CycleISP: Real image restoration via improved data synthesis. In: CVPR (2020)
37. Zhang, F., Xu, B., Li, Z., Liu, X., Lu, Q., Gao, C., Sang, N.: Towards general low-light RAW noise synthesis and modeling. In: ICCV (2023)
38. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE TIP **26**(7), 3142–3155 (2017)
39. Zhang, Y., Qin, H., Wang, X., Li, H.: Rethinking noise synthesis and modeling in raw denoising. In: ICCV (2021)