Modeling sRGB Camera Noise with Normalizing Flows

Shayan Kousha[1, 3,*]; Ali Maleky[1, 3, *], Michael S. Brown[3], Marcus A. Brubaker[1,2,3]
[1]York University [2]Vector Institute [3]Samsung AI Center–Toronto

Abstract

Noise modeling and reduction are fundamental tasks in low-level computer vision. They are particularly important for smartphone cameras relying on small sensors that exhibit visually noticeable noise. There has recently been renewed interest in using data-driven approaches to improve camera noise models via neural networks. These data-driven approaches target noise present in the raw-sensor image before it has been processed by the camera's image signal processor (ISP). Modeling noise in the RAW-rgb domain is useful for improving and testing the in-camera denoising algorithm; however, there are situations where the camera's ISP does not apply denoising or additional denoising is desired when the RAW-rgb domain image is no longer available. In such cases, the sensor noise propagates through the ISP to the final rendered image encoded in standard RGB (sRGB). The nonlinear steps on the ISP culminate in a significantly more complex noise distribution in the sRGB domain and existing raw-domain noise models are unable to capture the sRGB noise distribution. We propose a new sRGB-domain noise model based on normalizing flows that is capable of learning the complex noise distribution found in sRGB images under various ISO levels. Our normalizing flows-based approach outperforms other models by a large margin in noise modeling and synthesis tasks. We also show that image denoisers trained on noisy images synthesized with our noise model outperforms those trained with noise from baselines models.

1. Introduction

Modeling and reducing noise are long-standing problems in computer vision and image processing with a rich history (*e.g.*, [10, 17, 18]). While simple models, like simple additive white Gaussian noise (AWGN), have often been used in testing denoising methods, they are well-known not to be realistic and serves only as a rough approximation for real-world camera noise. When realistic noise models are needed, more sophisticated models, such as Poisson-Gaussian [8] or Heteroscedastic Gaussian models [7, 19], are used to model the noise distribution observed on camera sensors. While such models are more realistic than AWGN, they too are often not able to fully capture real camera noise distributions.

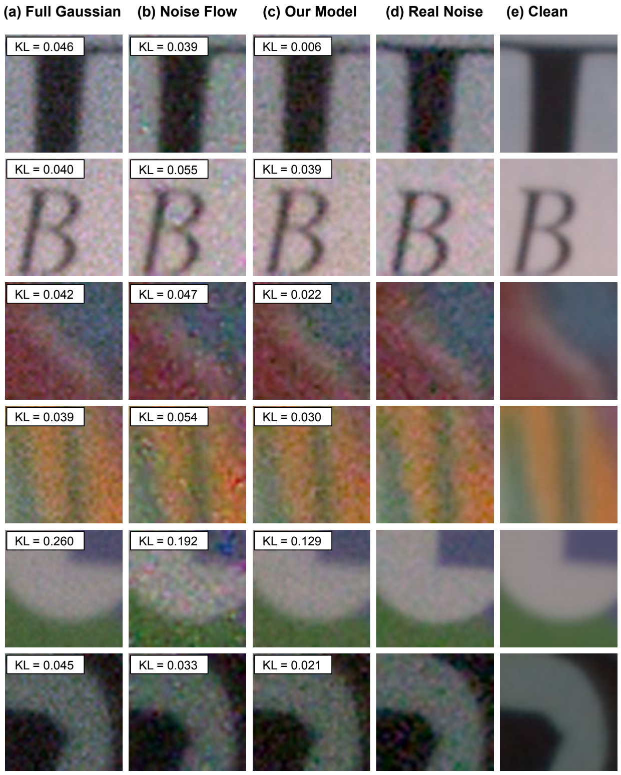


Figure 1. Noisy images generated by (a) a full covariance Gaussian model, (b) Noise Flow [2], and our model (c) compared with real images (d). Images are from the SIDD dataset [1].

In recent years, data-driven noise models have been proposed that learn the noise distribution directly from large datasets of noisy sensor images (*e.g.*, [2, 4, 9, 20, 21, 30, 31]). These methods focus on modeling the noise present in the raw sensor images. Modeling noise in the RAW-rgb domain is useful as denoising algorithms are typically applied by the camera's image signal processor (ISP) hardware. Such noise reduction is applied early in the ISP's processing pipeline (often in the Bayer processing stages)

[*] Work performed while interns at the Samsung AI Center–Toronto.

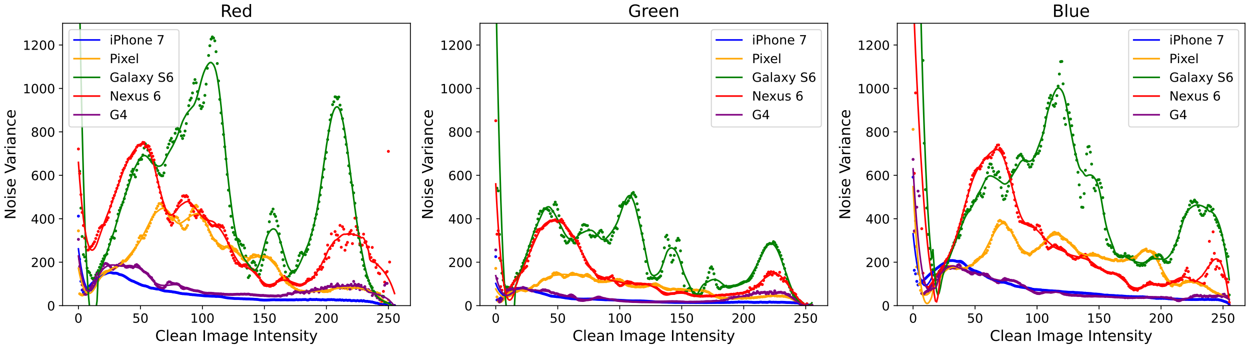


Figure 2. sRGB noise variance as a function of clean image intensity on SIDD [1]. These plots exhibit highly variable and unpredictable behavior, in contrast to RAW where there is typically a linear relation between noise variance and intensity. Consequently, we cannot use traditional image noise models like AWGN and NLF and instead propose a novel sRGB focused noise model based on normalizing flows.

before the image is rendered to its final sRGB representtion with tonal and color photo-finishing algorithms which are applied to improve the image's perceptual quality.

It is not unusual, however, for the in-camera denoising to be disabled, not present or insufficient. Further, most cameras do not save in RAW-rgb by default, making sRGB images far more ubiquitous. In such cases, the noisy sensor image is rendered through the camera's ISP, including the photo-finishing algorithms that apply complex, nonlinear operations to manipulate the image's tonal and color values. The resulting noise distribution in the final sRGB is notably more complex than in the unprocessed RAW-rgb space and existing RAW-rgb focused noise models become ineffective for modeling the sRGB noise, as shown in Fig. 1.

Contributions. We focus on modeling and synthesizing sRGB image noise, where the in-camera nonlinear processing has altered the noise characteristics from that of the camera's sensor. We begin with an analysis that shows that existing noise models targeting sensor noise in the RAW-rgb domain are not well suited for sRGB images. We then propose a generative model that combines recent advances in normalizing flows and captures effects of different gain (ISO) settings and camera types on sRGB image noise. We show that our sRGB noise model is superior to several baseline noise models. We further investigate our model's ability to synthesize noise by training a denoiser using noisy images sampled from the model. We show that this denoiser achieves significantly higher performance compared to denoisers trained on synthesized data from baseline models.

2. Related work

Additive white Gaussian noise (AWGN) [22, 25, 29] has long been used to model image noise. However, it is well known that real camera noise is non-Gaussian, in part because it fails to capture signal-dependence of the variance. A common and more realistic model is the heteroscedastic Gaussian model [19], defined as:

$$\mathbf{N} \sim \mathcal{N}(0, \beta_1 \mathbf{I} + \beta_2), \tag{1}$$

where $\beta_1, \beta_2 > 0$ are parameters that model the signal-dependent and signal-independent nature of noise observed on real camera sensors. Some cameras include a manufacturer-calibrated heteroscedastic Gaussian noise model in their saved RAW-rgb images, encoded in the DNG format [1, 2], although recent work [32] has suggested that such models are often not well calibrated. The heteroscedastic Gaussian noise model is relatively simple and has few parameters; however, it is still only an approximation of the real sensor noise [1, 7, 12, 24, 28].

Researchers have recently begun exploring data-driven approaches. For example, Abdelhamed et al. [2] proposed the Noise Flow model which combined the domain knowledge of signal and gain dependence with the expressiveness of learning-based generative models based on normalizing flows to capture more complex components of the noise. While the Noise Flow model was effective in simulating RAW-rgb noise, it relied on assumptions that do not apply in the sRGB color space. For example, the Noise Flow model builds off the heteroscedastic Gaussian model, which assumes the noise variance linearly depends on the underlying clean image intensity. However, this assumption no longer applies in the sRGB image domain (see Fig. 2) due to subsequent non-linear and potentially content-dependent processing of the RAW-rgb image. As a result, in our experiments we show that simply applying the Noise Flow model to sRGB data fails to capture the noise distribution.

There have been relatively few attempts to model noise from sRGB data. Nam et al. [21] introduced a model that is designed for the specific use case of modeling noise and other degradations caused by JPEG compression. More recently, the C2N [13] model attempted to model noise using unpaired clean and noisy images using a generative adversarial network (GAN).

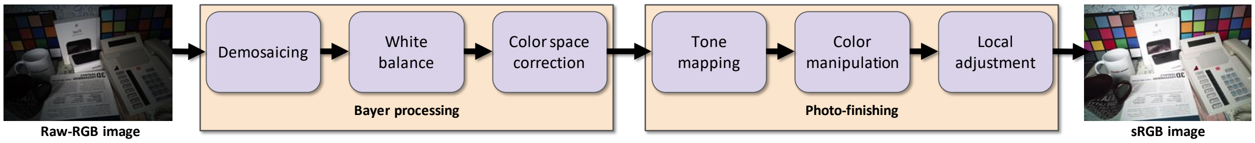


Figure 3. A typical camera ISP processing pipeline. This pipeline processes the RAW-rgb image to encode it in the sRGB domain. The nonlinear steps in the photo-finishing stages substantially complicate the noise distribution in the sRGB domain challenging.

In this paper, we introduce a data-driven model for image noise in the sRGB domain that conditions on camera settings and the underlying clean image. Like Noise Flow [2], the model is built using normalizing flows [6, 15, 16] but avoids making unrealistic assumptions that apply only to RAW-rgb. The resulting model is shown to have state-of-the-art noise modeling capabilities for sRGB noise. Further, when training a denoiser using noisy samples from the proposed noise model, we show that the resulting denoiser significantly out-performs those trained with existing sRGB noise models.

3. Preliminaries

Image noise is a combination of degradations introduced by multiple sources in the imaging process and physical limitations of sources like camera sensors. Many of these sources occur when first capturing the RAW-rgb image by the camera sensors. Subsequently, the RAW-rgb image (including noise) is subject to in-camera imaging process that transforms the image from the scene-referred RAW-rgb color space to the display-referred sRGB color space. Fig. 3 summarizes some of the main steps in the in-camera imaging pipeline. The results of the photo-finishing steps, many nonlinear in nature, introduce new sources of noise (*e.g*., from clipping and sharpening) while amplifying and distorting the original sensor's noise distribution. The noise from these multiple sources can be characterized as:

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{N}, \tag{2}$$

where $\tilde{\mathbf{I}}$ is the observed, noisy image, $\mathbf{I}$ is the true, underlying clean image, and $\mathbf{N}$ is the the noise whose distribution may depend on $\mathbf{I}$. In this work, we aim to design a generative model that captures the complexity of noise introduced by all sources.

Normalizing Flows

Normalizing flows are a family of generative models that have gained popularity in recent years. Due to their formulation, they admit both efficient sampling and exact evaluation of probability density in contrast to other generative models, like GANs and VAEs [3, 16]. Additionally, flow-based models do not suffer from issues like mode and posterior collapse that are commonly faced when training GANs and VAEs.

Below we briefly introduce normalizing flows. We refer the reader to recent review articles [16, 23] for a more extensive treatment. A normalizing flow consists of differentiable and bijective functions that learn a transformation $\mathbf{z} = f(\mathbf{x}|\Theta)$ with parameters Θ called a *flow*. A flow transforms data samples $\mathbf{x} \in \mathbb{R}^d$ from a complex distribution, $p_{\mathcal{X}}$, to some base space $\mathbf{z} \in \mathbb{R}^d$ with a known and tractable distribution and probability density function, $p_{\mathcal{Z}}$. Here, as is common, we will assume that $p_{\mathcal{Z}}$ takes the form of an isotropic Gaussian distribution with unit variance. The probability density function in the data space can then be found using the change of variables formula

$$p_{\mathcal{X}}(\mathbf{x}) = p_{\mathcal{Z}}(f(\mathbf{x}|\Theta)) \left|\det \mathbf{D}f(\mathbf{x}|\Theta)\right|, \tag{3}$$

where $\mathbf{D}f(\mathbf{x})$ is the Jacobian matrix of f at $\mathbf{x}$. The result is a model that, given a dataset $D = \{\mathbf{x}_i\}_{i=1}^{M}$, can be trained by using stochastic gradient descent to minimize the negative log likelihood of the data

$$-\sum_{i=1}^{M} \log p_{\mathcal{Z}}(f(\mathbf{x}_i|\Theta)) + \log\left|\det \mathbf{D}f(\mathbf{x}_i|\Theta)\right|, \tag{4}$$

with respect to the parameters Θ. Samples can be generated by sampling from the base distribution $\mathbf{z} \sim p_{\mathcal{Z}}$ and then applying the inverse flow $\mathbf{x} = f^{-1}(\mathbf{z})$.

Formally, normalizing flows define distributions over continuous spaces. To apply them to quantized data (*e.g*., as sRGB data which is typically truncated to 256 intensity levels) some attention must be paid to avoid a degeneracy [26] that occurs when fitting continuous density models to discrete data. Here we use uniform dequantization, which adds uniformly sampled noise to the images during training. More complex forms of dequantization are possible [11].

Constructing expressive, differentiable, and bijective functions is the primary research problem in normalizing flows, and there have been many attempts at this; see [16, 23] for a thorough review. A flow f is typically constructed by the composition of simpler flows—namely, $f = f_1 \circ ... \circ f_{N-1} \circ f_N$—since the composition of bijective functions is itself bijective. Analogous to adding depth in a neural network, composing flows can increase the complexity of the resulting distribution $p_{\mathcal{X}}$. Individual flows are typically constructed such that their inverse and Jacobian determinant are easily calculated. Next we review two common forms of bijection that we will use.

Affine Coupling Affine coupling flows [6] are a simple, efficient and widely used form of flow. They work by splitting the input dimensions, $\mathbf{x} = (\mathbf{x}^A, \mathbf{x}^B)$, into two disjoint subsets, $\mathbf{x}^A, \mathbf{x}^B$. Then, one subset, $\mathbf{x}^A$, is unmodified but used to compute scale and translation factors, which are applied to the other subset, $\mathbf{x}^B$. Formally, an affine coupling layer is defined as $\mathbf{y} = (\mathbf{y}^A, \mathbf{y}^B)$, where $\mathbf{y}^A = \mathbf{x}^A$ and

$$\mathbf{y}^B = \mathbf{x}^B \odot f_s(\mathbf{x}^A|\Theta) + f_t(\mathbf{x}^A|\Theta),$$

where $\odot$ is the element-wise product. The functions f_s and f_t compute the scale and translation factors and can be arbitrary, for example, deep neural networks. The inverse of this layer is easily computed as $\mathbf{x}^B = \left(\mathbf{y}^B - f_t(\mathbf{y}^A|\Theta)\right) \oslash f_s(\mathbf{y}^A|\Theta)$. Further, the log determinant of this transformation is efficiently calculated as $\sum \log f_s(\mathbf{x}^A|\Theta)$, where the sum is taken over the output dimensions of f_s.

1x1 Convolution Coupling layers must change the way dimensions are split between layers. This can be done by a random permutations [5, 6] but the Glow model [15] introduced the use of 1x1 convolutions as an invertible transformation. In essence, these layers are full linear transformations applied channel-wise to the inputs. The inverse is simply the inverse linear transformation applied channel-wise and the log determinant term is the number of pixels times the log determinant of the linear transformation.

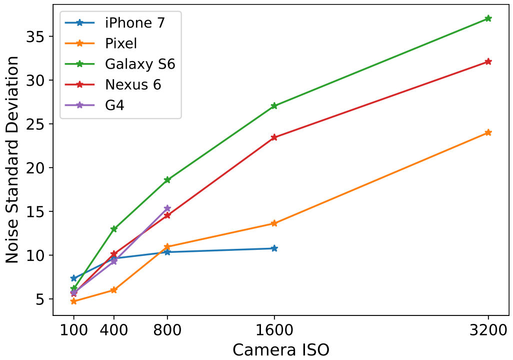


Figure 4. Real noise standard deviation changes as the sensitivity of the camera's sensor increases. The camera type is another factor affecting the noise behavior. For example, the noise standard deviation of images captured by Galaxy S6 is the highest under almost all ISO levels. Analysis done on the SIDD [1].

4. Normalizing Flows for sRGB Noise

Here we introduce our model of sRGB noise based on normalizing flows. Our analysis on the SIDD dataset [1], summarized in Fig. 2, confirms that real noise in the sRGB domain has a complex structure that is not captured by the standard, heteroscedastic-based noise models and varies significantly between cameras and even color channels. Inspired by this analysis, we introduce two new conditional flows in the following sections that allow the noise model to be conditioned on critical parameters, such as camera model, gain setting and clean intensity.

Figure 5 shows the proposed model architecture. Here we describe it as a transformation from the data (*i.e.*, a noisy, observed sRGB image $\tilde{\mathbf{I}}$) to the base space $\mathbf{z}$. First, input images are dequantized with uniform dequantization as mentioned above. Unlike RAW-rgb, which is typically represented as a floating point number, sRGB data is typically quantized to 256 intensity levels. Note that, because both the clean and observed images have been quantized, both need to be dequantized. Next, the clean image, $\mathbf{I}$, is subtracted from the observed image, $\tilde{\mathbf{I}}$, to get the noise image, $\mathbf{N}$. This is then followed by S flow blocks, which are responsible for learning a transformation from the noise to a sample in the base distribution, and vice versa. These flow blocks consist of one conditional linear (CL) flow followed by K conditional coupling steps (CCS), where a conditional coupling step consists of one invertible 1x1 convolution layer and one conditional affine coupling transformation. In our experiments, we use $S = 4$ and $K = 2$, unless otherwise specified. Next, we describe the conditional linear flow and conditional affine coupling layers.

4.1. Conditional Linear Flow

The nature of the noise and the subsequent non-linear processing is heavily determined by the specific camera and gain (or ISO) settings used. To account for this, we introduce a linear flow layer, which is conditioned on the camera, $\mathbf{c}$, and gain setting, $\mathbf{g}$, of the camera. This has the form

$$\mathbf{y} = \mathbf{x} \odot f_s(\mathbf{c}, \mathbf{g}) + f_t(\mathbf{c}, \mathbf{g}), \tag{5}$$

where $\odot$ is the element-wise product, $\mathbf{x}$ is the input, $\mathbf{y}$ is the output, and f_s and f_t are functions that output the scale and translation factors. The functions f_s and f_t can be arbitrarily complex and have no constraints other than that $f_s \neq 0$. See supplemental materials for architectural details of f_s and f_t. The inverse of this layer is easily calculated as:

$$\mathbf{x} = (\mathbf{y} - f_t(\mathbf{c}, \mathbf{g})) \oslash f_s(\mathbf{c}, \mathbf{g}), \tag{6}$$

where $\oslash$ is element-wise division. The log-determinant is given by $\sum \log f_s(\mathbf{c}, \mathbf{g})$, where the sum is taken over all dimensions of the input.

4.2. Conditional Affine Coupling

This layer is an extension of the affine coupling layer presented above. To capture the complex dependence of the noise distribution on both the underlying clean image and the camera and gain settings (see Figs. 2 and 4), we extend the coupling layers to take these values as input. The conditional affine coupling layer is similar to the standard affine

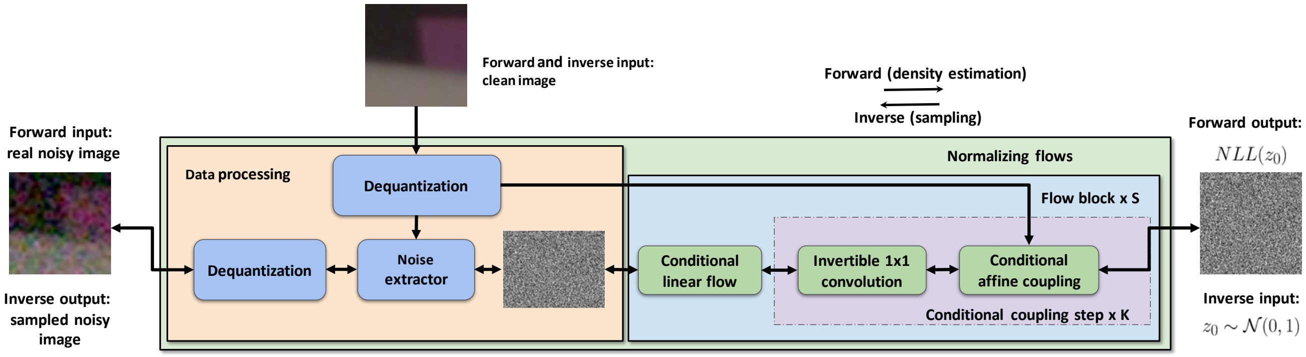


Figure 5. Our model consists of two main sections: (1) The data processing section is responsible for processing noisy and clean images. (2) The flow steps are responsible for learning the complex noise distribution.

coupling layer but differs in the the scale and translation factors. Specifically, the output of the layer is $\mathbf{y} = (\mathbf{y}^A, \mathbf{y}^B)$ with $\mathbf{y}^A = \mathbf{x}^A$ and

$$\mathbf{y}^B = \mathbf{x}^B \odot f_{cs}(\mathbf{x}^A|\mathbf{I}, \mathbf{c}, \mathbf{g}) + f_{ct}(\mathbf{x}^A|\mathbf{I}, \mathbf{c}, \mathbf{g}),$$

where $\odot$ is the element-wise product, $\mathbf{x} = (\mathbf{x}^A, \mathbf{x}^B)$ is the input, and f_{cs} and f_{ct} are functions that compute the conditional scale and translation based on the input clean image, $\mathbf{I}$, camera, $\mathbf{c}$, and gain setting, $\mathbf{g}$. The inverse and log determinant of this transformation can be easily calculated, analogous to the (unconditional) coupling layer. See the supplemental material for architectural details of f_{cs} and f_{ct}.

5. Experiments

To evaluate our model we use the SIDD dataset [1]. The SIDD-Medium split contains 320 noisy-clean image pairs captured under various ISO and lighting conditions taken by five different smartphones. While this dataset provides the data in both RAW and sRGB domains, here, we use only sRGB images. Note that this dataset used a simplified software ISP to render images from the captured RAW-rgb to sRGB instead of directly using the sRGB images produced by the camera. The software ISP applies the camera parameters stored in the raw's DNG file (*e.g.*, white-balancing, lens shading correction, color space mapping, custom tonemap, mapping to sRGB, and sRGB gamma). We extract approximately 3,000 patches of size 32x32 from each image. From these extracted patches, 80% are used for training and the remaining for validation. Patches are randomly distributed to ensure all cameras and ISO settings are fairly represented in both training and validation sets. For training we minimize the negative log likelihood (Eq. 4) using the Adam optimizer [14].

5.1. Metrics

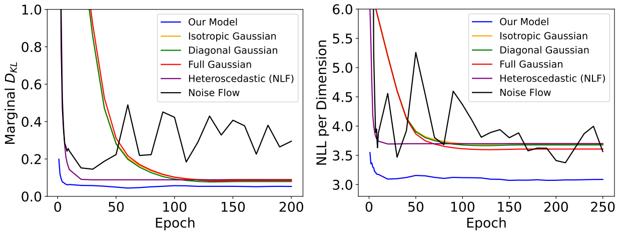
To quantitatively evaluate the model we consider two metrics. First, the negative log likelihood per dimension (NLL) on the test set is used as a direct evaluation of density estimation. Second, to better assess the quality of the sampled noise, we use the Kullback-Leibler (KL) divergence which was introduced in [2]. This metric computes the KL divergence between histograms of real and sampled noise. This metric is more sensitive to mismatches in the model's estimated variance than the NLL metric is.

5.2. Baselines

We explored a number of baseline sRGB noise models. We considered three variations of homoscedastic Gaussian noise: 1) AWGN, which assumes independent, isotropic noise at each pixel; 2) diagonal covariance Gaussian, which assumes independent but anisotropic noise at each pixel; and 3) full covariance Gaussian, which allows correlations between color channels. Note the full covariance Gaussian model was previously proposed for sRGB data by [21]. We also implemented a heteroscedastic Gaussian model, often referred to as the noise level function (NLF) and described in Eq. 1. While not expected to perform well based on, *e.g.*, Figure 2, it is a widely used and well known model of camera noise. Finally, we compared with a direct adaption of the Noise Flow model [2] to sRGB instead of RAW-rgb data. To do this we modified the number of channels that the architecture expected, but otherwise left it unchanged. Because of the strong assumptions made, Noise Flow has a small number of parameters. To make a more fair comparison, we also built a larger version of Noise Flow, referred to as Noise Flow-Large, that follows the architecture of Noise Flow but with the number of parameters increased to be comparable to our model. All baselines were implemented as normalizing flows using combinations of simple linear flows, signal-dependent flows, and gain-dependent flows [2] to ensure consistency.

5.3. Results

Table 1 shows the final test NLL and KL divergence for our model and all the baselines. Figures 6a and 6b show the



(a)

(b)

Figure 6. (a) Marginal KL divergence between the synthetic noise from the models and the real noise samples of the test set. (b) Testing NLL per dimension of our model compared to our baseline models. Our model not only performs by both metrics, it also converges faster.

Model	NLL	D_{KL}	$\#Params$
Isotropic Gaussian	3.703	0.091	50
Diagonal Gaussian	3.678	0.079	150
Full Gaussian	3.608	0.085	525
Heteroscedastic (NLF)	3.642	0.088	72
Noise flow [2]	3.311	0.198	2330
Noise Flow-Large	3.288	0.227	6618
Our model	**3.072**	**0.044**	6160

Table 1. Test NLL and marginal D_{KL} for our model and our baselines. The proposed model outperforms the baselines in both metrics by a large margin. Noise Flow-Large and Noise Flow models have the closest NLL to the proposed method, however, their relatively high D_{KL} indicates that these models fail to generate realistic noise samples. Together the results show that models originally developed for RAW-rgb are unlikely to be successful with sRGB.

KL divergence and NLL during training of all models. The results demonstrate that our model achieves a lower (better) NLL (3.072 vs. 3.311 nats/pixel for Noise Flow), and converges faster, requiring just a few epochs of training. During training the NLL of the Noise Flow model fluctuates significantly compared to the other baselines. We believe this is due to the assumptions made in the model architecture, specifically the signal-dependent layer, which do not hold in the sRGB domain and further demonstrate the need for a different approach to noise modeling in the sRGB domain.

In terms of marginal KL divergence our model also significantly improves over the baselines. Unlike with NLL, the closest performing baseline in terms of KL divergence was the diagonal Gaussian noise model, with a KL divergence of 0.079. In contrast, our model achieved a KL divergence of 0.044. For comparison, we also considered the recently proposed C2N [13] model, a GAN-based sRGB noise model. Their paper reported a KL divergence of 0.1638. However, we note that KL divergence is sensitive to the choice of histogram bins and other implementation details.

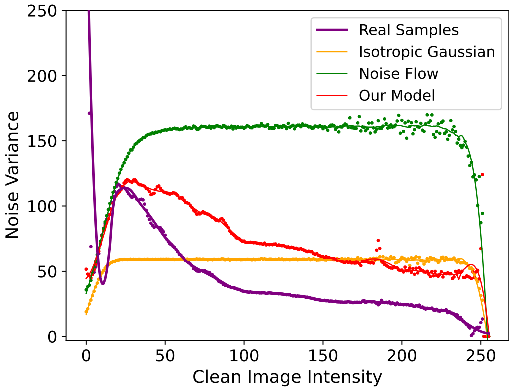


Figure 7. Red channel noise variance of real noise samples from iPhone7-ISO 100 setting and samples generated from our model and two of the baselines given clean images corresponding to the real noise samples. Our model success fully shows a similar noise variance trend to the real noise. However, the Noise Flow and Isotropic Gaussian models fail to learn this trend.

Interestingly, Table 1 shows that Noise Flow significantly outperformed the other baseline noise models in terms of NLL, but significantly underperformed them in terms of KL divergence. On further investigation, we found that the Noise Flow model was significantly overestimating the variance of the noise in many cases. For instance, Figure 7, similar to Figure 2, shows the variance of sRGB noise as a function of noise-free image intensity for real data, our model, Noise Flow, and the isotropic baseline for an iPhone7 with ISO level of 100. This graph shows that Noise Flow has badly overestimated the variance, even compared to the isotropic Gaussian model, suggesting that the difficulties in training seen earlier are preventing it from converging to a reasonable model. In contrast, while our proposed model slightly overestimates the noise as well, it much better captures the structure of the relationship.

Qualitative Comparison. To qualitatively compare the trained noise models Figure 8 shows samples of noise from our model and with two baseline models at different ISO levels. Out of the Gaussian-based models, the full covariance Gaussian Model achieves the best test NLL and has been shown to be a good fit for modeling noise in the sRGB space as its full covariance can learn dependencies between channels [21]. A more extensive set of samples is available in the supplemental materials. Samples from our model are generally more visually similar to those of real noisy images, particularly in comparison to the baselines. Noise Flow samples are too noisy and samples from full covariance Gaussian do not exhibit enough variance. For example, Noise Flow samples at ISO 800 are significantly less noisy compared to the real noisy image.

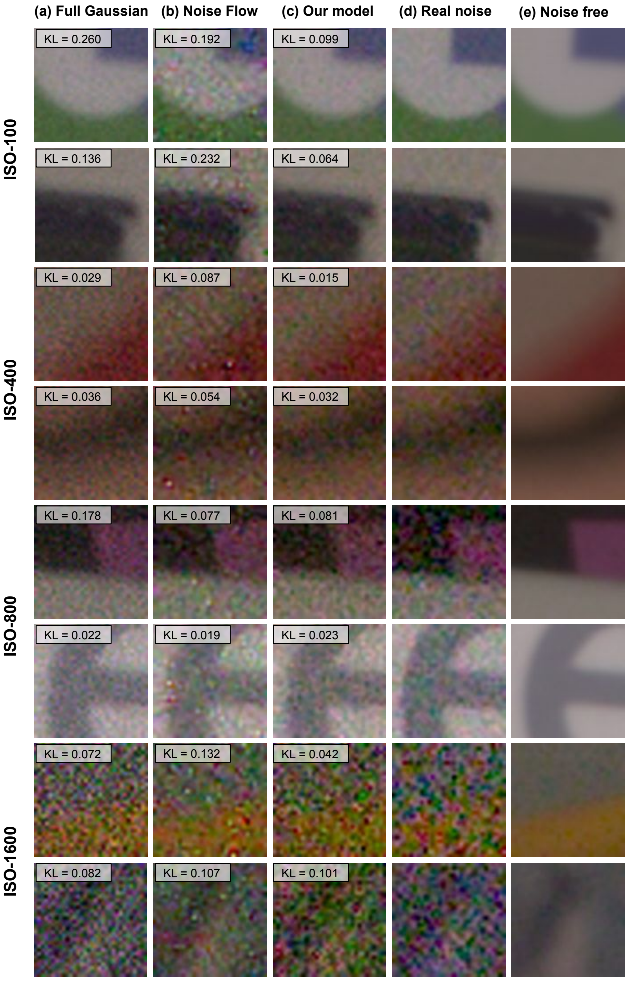
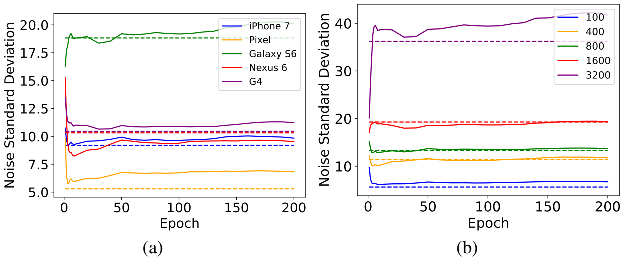


Figure 8. Generated samples from our model and two baselines. Samples from our model have noticeable visual similarities with the real noisy samples. Our samples achieve the lowest D_{KL} in almost all cases, showing its ability to generate realistic noise.

Modeling Different Cameras and ISO Settings. Figure 9 shows the learning of noise characteristics under different cameras and ISO settings. The dotted lines are the true noise standard deviation under each condition. The results show that the noise distribution changes drastically with different cameras and ISO levels. Further, our model is able to successfully capture this behavior to learn a more realistic noise model. The graphs also suggest that, while the model is quick to learn in the first few epochs and exhibits relatively little overfitting with the exception of ISO 3200. However we note that this ISO setting has a very limited number of samples in the training set.



(a) (b)

Figure 9. Standard deviation of learned noise model during training under (a) different cameras and (b) different ISO levels is close to the real ones. This shows the proposed conditional layers have learned to adjust the distribution based on those settings. Dotted lines are the true values.

Flow blocks	NLL	D_{KL}
CL	3.678	0.079
CCS$_{\text{ISO only}}$x2	3.726	0.154
CCS$_{\text{camera only}}$x2	3.609	0.216
CCS$_{\text{clean image only}}$x2	3.882	0.295
CCSx2	3.530	0.104
CL-CCSx2	3.398	0.075
(CL-CCSx2) x2	3.254	0.067
(CL-CCSx2) x4	**3.072**	**0.044**

Table 2. Test NLL and D_{KL} achieved by different flow steps. The symbols CL and CCS refer to the conditional linear and conditional coupling steps where a conditional coupling step is a combination of a 1x1 convolutional layer and a conditional affine coupling layer. The numbers next to x indicate the number of flow and coupling steps. Unless otherwise specified in the subscripts, the layers have the formulation mentioned in the methods sections. Last row is the architecture we use in other experiments.

Ablation Studies. Table 2 summarizes the performance of different architecture choices for the flow block of our normalizing flows model. The results show a significant improvement in noise modeling and noise synthesis when the coupling step conditions on all the important variables including the clean image, camera type, and ISO settings, rather than conditioning on only one of them. Additionally, we see an improvement by adding the conditional linear flow (CL) layer to the conditional coupling steps (CCS), showing the importance of having a direct way of transferring knowledge from camera types and ISO levels. Finally, we show the importance of having multiple flow blocks. The architecture of (CL-CCSx2) x4 with four flow blocks achieves the best performance in both metrics, NLL and D_{KL}. This is the architecture used in other experiments.

5.4. Application: sRGB Denoising

One of the main applications of noise modeling is to generate realistic noise to be used in downstream tasks, like denoising. Here we explore the training of the standard DnCNN denoiser [30] using samples generated from the learned noise model to test its noise generation capabilities.

Noise Model	$PSNR$	$SSIM$
Isotropic Gaussian	32.48	0.855
Diagonal Gaussian	33.34	0.867
Full Gaussian	32.72	0.873
Heteroscedastic Gaussian	32.24	0.849
Noise Flow	33.81	0.894
C2N*	33.76	0.901
Our model	**34.74**	**0.912**
Real Noise	36.51	0.922

Table 3. Denoiser performance when trained on samples from each model. Denoisers are evaluated on the SIDD Benchmark set. The denoiser trained on samples from our noise model achieves better performance compared to those trained on noise from the baselines. (*) Results are taken from [13].

Dataset To train DnCNN we use SIDD-Medium dataset. Clean images are from SIDD-Medium and the noisy images are either the real ones provided with the dataset or generated by either the proposed noise model or one of the baseline models as specified. We use noisy and clean images from SIDD-Validation and SIDD-Benchmark for validation and testing purposes, respectively.

Results Table 3 summarizes the result of our denoising experiment. The results show that a DnCNN model trained on the synthetic noises from our model achieves a significantly higher performance in terms of peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [27] on SIDD-Benchmark compared to the denoisers trained on the samples from baseline models. While the performance of our model does not exceed the performance of a denoiser trained with real data (*e.g.*, as was found with noise models in RAW-rgb [2]), it does significantly shrink the gap.

Figure 10 shows denoising results of the DnCNN model trained with real noisy images and three noise synthesis strategies including our model and two of our baselines. The figure also includes the input noisy image and the ground truth clean image for reference. For the full set of results, we refer the readers to the supplemental materials. The denoiser trained on noise samples from our model tends to produce denoised images that are closer to the ground truth clean images than when trained on samples from the baseline noise models. The model trained on noisy image samples from Noise Flow tends to not remove the noise fully, outputting images which still contain a significant amount of noise (*e.g.*, as in row 6). This is likely caused by the tendency of Noise Flow to significantly overestimate the noise variance as demonstrated in Figure 7. Finally, denoisers trained on Gaussian samples tend to produce overly smooth denoised images, (*e.g.*, as in row 4).

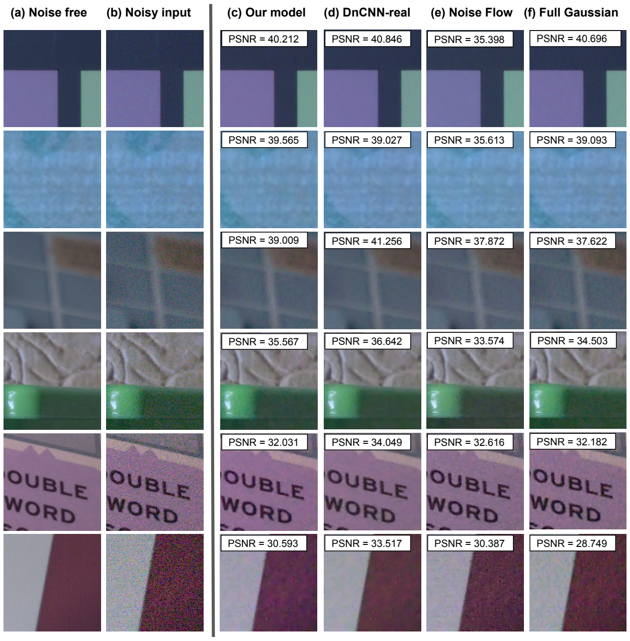


Figure 10. Denoising results on SIDD-Validation from denoisers trained on noisy images from (c) real noisy images of SIDD-Validation, (d) our model, and (e, f) two of our baselines.

6. Conclusion

We introduced a noise model specifically tailored to capture image noise in the sRGB domain. Because image noise in the sRGB domain exhibits significantly more complex noise distributions than those found in the unprocessed RAW-rgb domain, we showed that existing RAW-rgb noise models like heteroscedastic noise and the Noise Flow model [2] are ineffective at capturing noise in sRGB. To address this we described an architecture based on normalizing flows that can effectively model sRGB image noise, while capturing the complex dependencies on variables such as clean image intensity, camera model and ISO settings. We demonstrated the effectiveness of the proposed noise model directly (with NLL and KL divergence metrics) and by training image denoisers using image noise synthesized by our model. We showed that our image denoisers significantly outperform other denoisers trained on existing noise models for sRGB. Source code is available at `https://yorkucvil.github.io/sRGBNoise/`.

Acknowledgments

This work was done as part of an internship at the Samsung AI Center in Toronto, Canada. AM's internship was funded by a Mitacs Accelerate. SK's and AM's student funding came in part from the Canada First Research Excellence Fund for the Vision: Science to Applications (VISTA) programme and an NSERC Discovery Grant.

References

[1] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *CVPR*, 2018. 1, 2, 4, 5

[2] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, "Noise Flow: Noise Modeling with Conditional Normalizing Flows," in *ICCV*, 2019. 1, 2, 3, 5, 6, 8

[3] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," *arXiv preprint arXiv:2103.04922*, 2021. 3

[4] C. Chen, Z. Xiong, X. Tian, and F. Wu, "Deep boosting for image denoising," in *ECCV*, 2018. 1

[5] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," in *ICLR Workshop*, 2015. 4

[6] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," in *ICLR*, 2017. 3, 4

[7] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Processing*, vol. 89, pp. 2609–2629, 12 2009. 1, 2

[8] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *TIP*, vol. 17, no. 10, pp. 1737–1754, 2008. 1

[9] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *CVPR*, 2019. 1

[10] G. E. Healey and R. Kondepudy, "Radiometric ccd camera calibration and noise estimation," *TPAMI*, vol. 16, no. 3, pp. 267–276, 1994. 1

[11] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *ICML*, 2019. 3

[12] G. C. Holst, *CCD Arrays, Cameras, and Displays*. SPIE Optical Engineering Press, USA, second edition, 1996. 2

[13] G. Jang, W. Lee, S. Son, and K. M. Lee, "C2n: Practical generative noise modeling for real-world denoising," in *ICCV*, 2021. 2, 6, 8

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015. 5

[15] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018. 3, 4

[16] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *TPAMI*, vol. 43, no. 11, p. 3964–3979, 2021. 3

[17] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," *TPAMI*, vol. 7, no. 2, pp. 165–177, 1985. 1

[18] C. Liu, R. Szeliski, S. Bing Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *TPAMI*, vol. 30, no. 2, pp. 299–314, 2008. 1

[19] X. Liu, M. Tanaka, and M. Okutomi, "Practical signal-dependent noise parameter estimation from a single noisy image," *TIP*, vol. 23, no. 10, pp. 4361–4371, 2014. 1, 2

[20] Y. Liu, S. Anwar, L. Zheng, and Q. Tian, "Gradnet image denoising," in *CVPR Workshop*, 2020. 1

[21] S. Nam, Y. Hwang, Y. Matsushita, and S. J. Kim, "A holistic approach to cross-channel image noise modeling and its application to image denoising," in *CVPR*, 2016. 1, 2, 5, 6

[22] N. Ohta, "A statistical approach to background subtraction for surveillance systems," in *ICCV*, 2001. 2

[23] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021. 3

[24] T. Plötz and S. Roth, "Benchmarking denoising algorithms with real photographs," in *CVPR*, 2017. 2

[25] P. L. Rosin, "Thresholding for change detection," in *ICCV*, 1998. 2

[26] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *ICLR*, 2016. 3

[27] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *TIP*, vol. 13, no. 4, pp. 600–612, 2004. 8

[28] K. Wei, Y. Fu, J. Yang, and H. Huang, "A physics-based noise formation model for extreme low-light raw denoising," in *CVPR*, 2020. 2

[29] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *TPAMI*, vol. 19, no. 7, pp. 780–785, 1997. 2

[30] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *TIP*, vol. 26, no. 7, pp. 3142–3155, 2017. 1, 7

[31] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *TIP*, vol. 27, no. 9, pp. 4608–4622, 2018. 1

[32] Y. Zhang, H. Qin, X. Wang, and H. Li, "Rethinking noise synthesis and modeling in raw denoising," in *ICCV*, 2021. 2