Noise Flow: Noise Modeling with Conditional Normalizing Flows

Abdelrahman Abdelhamed[1,2] Marcus A. Brubaker[2] Michael S. Brown[1,3]
[1]York University [2]Borealis AI [3]Samsung AI Center, Toronto

{kamel,mbrown}@eecs.yorku.ca, marcus.brubaker@borealisai.com

Abstract

Modeling and synthesizing image noise is an important aspect in many computer vision applications. The long-standing additive white Gaussian and heteroscedastic (signal-dependent) noise models widely used in the literature provide only a coarse approximation of real sensor noise. This paper introduces Noise Flow, a powerful and accurate noise model based on recent normalizing flow architectures. Noise Flow combines well-established basic parametric noise models (e.g., signal-dependent noise) with the flexibility and expressiveness of normalizing flow networks. The result is a single, comprehensive, compact noise model containing fewer than 2500 parameters yet able to represent multiple cameras and gain factors. Noise Flow dramatically outperforms existing noise models, with 0.42 nats/pixel improvement over the camera-calibrated noise level functions, which translates to 52% improvement in the likelihood of sampled noise. Noise Flow represents the first serious attempt to go beyond simple parametric models to one that leverages the power of deep learning and data-driven noise distributions.

1. Introduction

Image noise modeling, estimation, and reduction is an important and active research area (e.g., [7, 14, 28, 29]) with a long-standing history in computer vision (e.g., [12, 18, 19, 24]). A primary goal of such efforts is to remove or correct for noise in an image, either for aesthetic purposes, or to help improve other downstream tasks. Towards this end, accurately modeling noise distributions is a critical step.

Existing noise models are not sufficient to represent the complexity of real noise [1, 26]. For example, a univariate homoscedastic Gaussian model does not represent the fact that photon noise is signal-dependent—that is, the variance of the noise is proportional to the magnitude of the signal. In turn, the signal-dependent heteroscedastic model [7, 8, 22], often referred to as the noise level function (NLF), does not represent the spatial non-uniformity of noise power (e.g., fixed-pattern noise) or other sources of noise and non-linearities, such as amplification noise and quantization [13]. See Figure 2. In spite of their well-known limitations, these models are still the most commonly used. More complex models, such as a Poisson mixture [15, 32], exist, but still do not capture the complex noise sources mentioned earlier.

Figure 1: Synthetic noisy images generated by (a) a Gaussian model, (b) a heteroscedastic signal-dependent model represented by camera noise level functions (NLF), and (c) our Noise Flow model. Synthetic noise generated from Noise Flow is consistently the most similar to the real noise in (d), qualitatively and quantitatively (in terms of KL divergence relative to the real noise, shown on each image). (e) Reference clean image. Images are from the SIDD [1].

Contribution We introduce Noise Flow, a new noise model that combines the insights of parametric noise models and the expressiveness of powerful generative models. Specifically, we leverage recent normalizing flow architec-

tures [17] to accurately model noise distributions observed from large datasets of real noisy images. In particular, based on the recent Glow architecture [17], we construct a normalizing flow model which is conditioned on critical variables, such as intensity, camera type, and gain settings (i.e., ISO). The model can be shown to be a strict generalization of the camera NLF but with the ability to capture significantly more complex behaviour. The result is a single model that is compact (fewer then 2500 parameters) and considerably more accurate than existing models. See Figure 1. We explore different aspects of the model through a set of ablation studies. To demonstrate the effectiveness of Noise Flow, we consider the application of denoising and use Noise Flow to synthesize training data for a denoising CNN resulting in significant improvements in PSNR. Code and pretrained models for Noise Flow are available at: https://github.com/BorealisAI/noise_flow.

2. Background and Related Work

Image noise is an undesirable by-product of any imaging system. Image noise can be described as deviations of the measurements from the actual signal and results from a number of causes, including physical phenomena, such as photon noise, or the electronic characteristics of the imaging sensors, such as fixed pattern noise.

Given an observed image $\tilde{\mathbf{I}}$ and its underlying noise-free image $\mathbf{I}$, their relationship can be written as

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{n}, \tag{1}$$

where $\mathbf{n}$ is the noise corrupting $\mathbf{I}$. Our focus in this work is to model $\mathbf{n}$.

Several noise models have been proposed in the literature. The simplest and most common noise model is the homoscedastic Gaussian assumption, also known as the additive white Gaussian noise (AWGN). Under this assumption, the distribution of noise in an image is a Gaussian distribution with independent and identically distributed values:

$$n_i \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

where n_i is the noise value at pixel i and follows a normal distribution with zero mean and σ^2 variance.

Despite its prevalence, the Gaussian model does not represent the fact that photon noise is signal-dependent. To account for signal dependency of noise, a Poisson distribution $\mathcal{P}$ is used instead:

$$n_i \sim \alpha \mathcal{P}(\mathbf{I}_i) - \mathbf{I}_i, \tag{3}$$

where $\mathbf{I}_i$, the underlying noise-free signal at pixel i, is both the mean and variance of the noise, and α is a sensor-specific scaling factor of the signal.

Neither the Gaussian nor the Poisson models alone can accurately describe image noise. That is because image noise consists of both signal-dependent and signal-independent components. To address such limitation, a Poisson-Gaussian model has been adapted [7, 8, 22], where the noise is a combination of a signal-dependent Poisson distribution and a signal-independent Gaussian distribution:

$$n_i \sim \alpha \mathcal{P}(\mathbf{I}_i) - \mathbf{I}_i + \mathcal{N}(0, \delta^2). \tag{4}$$

A more widely accepted alternative to the Poisson-Gaussian model is to replace the Poisson component by a Gaussian distribution whose variance is signal-dependent [20, 23], which is referred to as the heteroscedastic Gaussian model:

$$n_i \sim \mathcal{N}(0, \alpha^2 \mathbf{I}_i + \delta^2). \tag{5}$$

The heteroscedastic Gaussian model is more commonly referred to as the noise level function (NLF) and describes the relationship between image intensity and noise variance:

$$\text{var}(n_i) = \beta_1 \mathbf{I}_i + \beta_2, \qquad \beta_1 = \alpha^2, \beta_2 = \delta^2. \tag{6}$$

Signal-dependent models may accurately describe noise components, such as photon noise. However, in real images there are still other noise sources that may not be accurately represented by such models [1, 7, 26]. Examples of such sources include fixed-pattern noise, defective pixels, clipped intensities, spatially correlated noise (i.e., cross-talk), amplification, and quantization noise. Some attempts have been made to close the gap between the prior models and the realistic cases of noise—for example, using a clipped heteroscedastic distribution to account for clipped image intensities [7] or using a Poisson mixture model to account for the tail behaviour of real sensor noise [32]. Recently, a GAN was trained for synthesizing noise [3]; however, it was not clear how to quantitatively assess the quality of the generated samples. To this end, there is still a lack of noise models that capture the characteristics of real noise. In this paper, we propose a data-driven normalizing flow model that can estimate the density of a real noise distribution. Unlike prior attempts, our model can capture the complex characteristics of noise that cannot be explicitly parameterized by existing models.

2.1. Normalizing Flows

Normalizing flows were first introduced to machine learning in the context of variational inference [27] and density estimation [5] and are seeing increased interest for generative modeling [17]. A normalizing flow is a transformation of a random variable with a known distribution (typically Normal) through a sequence of differentiable, invertible mappings. Formally, let $\mathbf{x}_0 \in \mathbb{R}^D$ be a random variable with a known and tractable probability density function $p_{\mathcal{X}_0} : \mathbb{R}^D \to \mathbb{R}$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be a sequence of random variables such that $\mathbf{x}_i = f_i(\mathbf{x}_{i-1})$ where

$f_i : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a differentiable, bijective function. Then if $\mathbf{n} = f(\mathbf{x}_0) = f_N \circ f_{N-1} \circ \cdots \circ f_1(\mathbf{x}_0)$, the change of variables formula says that the probability density function for $\mathbf{n}$ is

$$p(\mathbf{n}) = p_{\mathcal{X}_0}(g(\mathbf{n})) \prod_{j=1}^{N} \left|\det \mathbf{J}_j(g(\mathbf{n}))\right|^{-1} \tag{7}$$

where $g = g_1 \circ \cdots \circ g_{N-1} \circ g_N$ is the inverse of f, and $\mathbf{J}_j = \partial f_j / \partial \mathbf{x}_{j-1}$ is the Jacobian of the jth transformation f_j with respect to its input $\mathbf{x}_{j-1}$ (i.e., the output of f_{j-1}).

Density Estimation A normalizing flow can be directly used for density estimation by finding parameters which maximize the log likelihood of a set of samples. Given the observed data, $\mathcal{D} = \{\mathbf{n}_i\}_{i=1}^{M}$, and assuming the transformations $f_1, \ldots, f_N$ are parameterized by $\Theta = (\theta_1, \ldots, \theta_N)$ respectively, the log likelihood of the data $\log p(\mathcal{D}|\Theta)$ is

$$\sum_{i=1}^{M} \log p_{\mathcal{X}_0}(g(\mathbf{n}_i|\Theta)) - \sum_{j=1}^{N} \log \left|\det \mathbf{J}_j(g(\mathbf{n}_i|\Theta), \theta_j)\right| \tag{8}$$

where the first term is the log likelihood of the sample under the base measure and the second term, sometimes called the log-determinant or volume correction, accounts for the change of volume induced by the transformation by the normalizing flows.

Bijective Transformations To construct an efficient normalizing flow we need to define differentiable and bijective transformations f. Beyond being able to define and compute f, we also need to be able to efficiently compute its inverse, g, and the log determinant $\log|\det \mathbf{J}|$, which are necessary to evaluate the data log likelihood in Equation 8. First consider the case of a linear transformations [17]

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{9}$$

where $\mathbf{A} \in \mathbb{R}^{D \times D}$ and $\mathbf{b} \in \mathbb{R}^D$ are parameters. For f to be invertible $\mathbf{A}$ must have full rank; its inverse is given by $g(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{b})$ and the determinant of the Jacobian is simply $\det \mathbf{J} = \det \mathbf{A}$.

Affine Coupling To enable more expressive transformations, we can use the concept of coupling [5]. Let $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_B)$ be a disjoint partition of the dimensions of $\mathbf{x}$ and let $\hat{f}(\mathbf{x}_A|\theta)$ be a bijection on $\mathbf{x}_A$ which is parameterized by θ. Then a coupling flow is

$$f(\mathbf{x}) = (\hat{f}(\mathbf{x}_A; \theta(\mathbf{x}_B)), \mathbf{x}_B) \tag{10}$$

where $\theta(\mathbf{x}_B)$ is *any* arbitrary function which uses only $\mathbf{x}_B$ as input. The power of a coupling flow resides, largely, in the ability of $\theta(\mathbf{x}_B)$ to be arbitrarily complex. For instance, shallow ResNets [11] were used for this function in [17].

Inverting a coupling flow can be done by using the inverse of $\hat{f}$. Further, the Jacobian of f is a block triangular matrix where the diagonal blocks are $\hat{\mathbf{J}}$ and the identity. Hence the determinant of the Jacobian is simply the determinant of $\hat{\mathbf{J}}$. A common form of a coupling layer is the *affine coupling layer* [6, 17]

$$\hat{f}(\mathbf{x}; \mathbf{a}, \mathbf{b}) = \mathbf{D}\mathbf{x} + \mathbf{b} \tag{11}$$

where $\mathbf{D} = \text{diag}(\mathbf{a})$ is a diagonal matrix. To ensure that $\mathbf{D}$ is invertible and has non-zero diagonals it is common to use $\mathbf{D} = \text{diag}(\exp(\mathbf{a}))$.

With the above formulation of normalizing flows, it becomes clear that we can utilize their expressive power for modeling real image noise distributions and mapping them to easily tractable simpler distributions. As a by-product, such models can directly be used for realistic noise synthesis. Since the introduction of normalizing flows to machine learning, they have been focused towards image generation tasks (e.g., [17]). However, in this work, we adapt normalizing flows to the task of noise modeling and synthesis by introducing two new conditional bijections, which we describe next.

3. Noise Flow

In this section, we define a new architecture of normalizing flows for modeling noise which we call Noise Flow. Noise Flow contains novel bijective transformations which capture the well-established and fundamental aspects of parametric noise models (e.g., signal-dependent noise and gain) which are mixed with more expressive and general affine coupling transformations.

3.1. Noise Modeling using Normalizing Flows

Starting from Equations 1 and 8, we can directly use normalizing flows to estimate the probability density of a complex noise distribution. Let $\mathcal{D} = \{\mathbf{n}_i\}_{i=1}^{M}$ denote a dataset of observed camera noise where $\mathbf{n}_i$ is the noise layer corrupting a raw-RGB image. Noise layers can be obtained by subtracting a clean image from its corresponding noisy one. As is common, we choose an isotropic Normal distribution with zero mean and identity covariance as the base measure. Next, we choose a set of bijective transformations, with a set of parameters Θ, that define the normalizing flows model. Lastly, we train the model by minimizing the negative log likelihood of the transformed distribution, as indicated in Equation 8.

We choose the Glow model [17] as our starting point. We use two types of bijective transformations (i.e., layers) from the Glow model: (1) the affine coupling layer as defined in Equation 11 that can capture arbitrary correlations between image dimensions (i.e., pixels); and (2) the 1×1 convolutional layers that are used to capture cross-channel correlations in the input images.

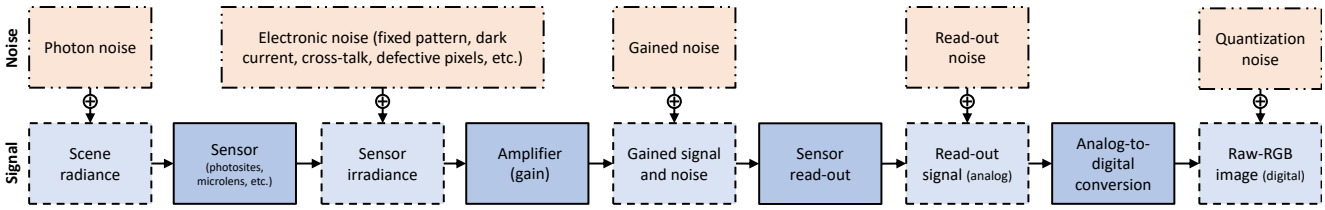


Figure 2: A simplified model of an imaging pipeline showing imaging processes (in the bottom row) and the associated noise processes (in the top row). Model adapted from [9, 10, 12, 19].

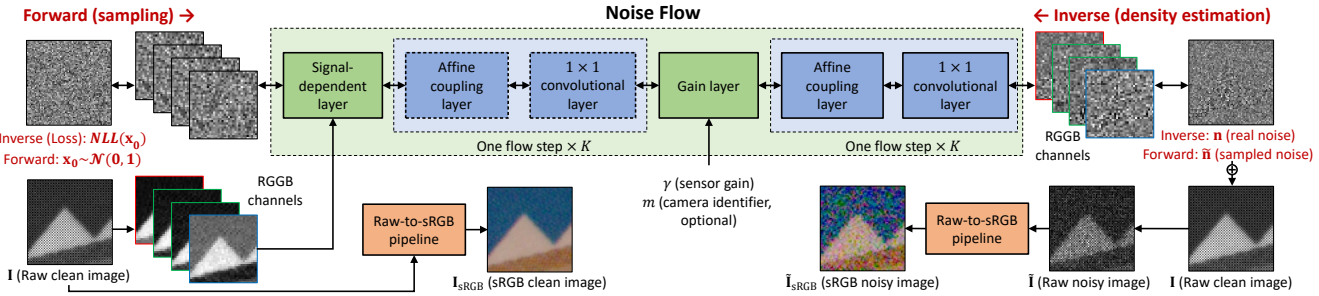


Figure 3: The architecture of our Noise Flow model. The affine coupling and 1×1 convolutional layers are ported from [17]. The signal-dependent and gain layers are newly proposed. The Raw-to-sRGB pipeline is ported from [1].

3.2. Noise Modeling using Conditional Normalizing Flows

Existing normalizing flows are generally trained in an unsupervised manner using only data samples and without additional information about the data. In our case, we have some knowledge regarding the noise processes, such as the signal-dependency of noise and the scaling of the noise based on sensor gain. Some of these noise processes are shown in Figure 2 along with their associated imaging processes. Thus, we propose new normalizing flow layers that are conditional on such information. However, many noise processes, such as fixed-pattern noise, cannot be easily specified directly. To capture these other phenomena we use a combination of affine coupling layers (Equations 10 and 11) and 1×1 convolutional layers (a form of Equation 9) which were introduced by the Glow model [17].

Figure 3 shows the proposed architecture of our noise model (Noise Flow). Noise Flow is a sequence of a signal-dependent layer; K unconditional flow steps; a gain layer; and another set of K unconditional flow steps. Each unconditional flow step is a block of an affine coupling layer followed by a 1×1 convolutional layer. The term K is the number of flow steps to be used in the model. In our experiments, we use $K = 4$, unless otherwise specified. The model is fully bijective—that is, it can operate in both directions, meaning that it can be used for both simulating noise (by sampling from the base measure $\mathbf{x}_0$ and applying the sequence of transformations) or likelihood evaluation (by using the inverse transformation given a noise sample $\tilde{\mathbf{I}}$ to evaluation of Equation 7). The Raw-to-sRGB rendering pipeline is imported from [1]. Next, we discuss the proposed signal-dependent and gain layers in details.

3.2.1 Signal-Dependent Layer

We construct a bijective transformation that mimics the signal-dependent noise process defined in Equation 5. This layer is defined as

$$f(\mathbf{x}) = \mathbf{s} \odot \mathbf{x}, \qquad \mathbf{s} = (\beta_1 \mathbf{I} + \beta_2)^{\frac{1}{2}}. \tag{12}$$

The inverse of this layer is given by $g(\mathbf{x}) = \mathbf{s}^{-1} \odot \mathbf{x}$, where $\mathbf{I}$ is the latent clean image, and $\odot$ is point-wise multiplication. To account for volume change induced by this transformation, we compute the log determinant as

$$\log |\det \mathbf{J}| = \sum_{i=1}^{D} \log(s_i) \tag{13}$$

where s_i is the ith element of $\mathbf{s}$ and D is the dimensionality (i.e., number of pixels and channels) of $\mathbf{x}$. The signal-dependent noise parameters β_1 and β_2 should be strictly positive as the standard deviation of noise should be positive and an increasing function of intensity. Thus, we parameterize them as $\beta_1 = \exp(b_1)$ and $\beta_2 = \exp(b_2)$. We initialize the signal-dependent layer to resemble an identity transformation by setting $b_1 = -5.0$ and $b_2 = 0$. This way, $\beta_1 \approx 0$ and $\beta_2 = 1.0$, and hence the initial scale $\mathbf{s} \approx 1.0$.

3.2.2 Gain Layer

Sensor gain amplifies not only the signal, but also the noise. With common use of higher gain factors in low-light imaging, it becomes essential to explicitly factor the effect of gain in any noise model. Hence, we propose a gain-dependent bijective transformation as a layer of Noise Flow. The gain layer is modeled as a scale factor γ of the corresponding ISO level of the image, and hence the transformation is

$$f(\mathbf{x}) = \gamma(\text{ISO}) \odot \mathbf{x}, \qquad \gamma(\text{ISO}) = u(\text{ISO}) \times \text{ISO}, \quad (14)$$

where $u(\text{ISO}) > 0$ allows the gain factors to vary somewhat from the strict scaling dictated by the ISO value. The inverse transformation is $g(\mathbf{x}) = \gamma^{-1}(\text{ISO}) \odot \mathbf{x}$, where u is parameterized to be strictly positive and is initialized to $u \approx 1/200$ to account for the typical scale of the ISO values. Finally, the log determinant of this layer is

$$\log|\det \mathbf{J}| = D \log(\gamma(\text{ISO})), \quad (15)$$

where D is the number of dimensions (i.e., pixels and channels) in $\mathbf{x}$. There are many ways to represent $u(\text{ISO})$. However, since the available dataset contained only a small set of discrete ISO levels, we chose to simply use a discrete set of values. Formally $u(\text{ISO}) = \exp(v_{\text{ISO}})$ where the exponential is used to ensure that $u(\text{ISO})$ is positive. We use a single parameter for each ISO level in the dataset (e.g., $\{v_{100}, \ldots, v_{1600}\}$). The values of v_{ISO} are initialized so that $\exp(v_{\text{ISO}}) \approx 1/200$ to account for the scale of the ISO value and ensure the initial transformation remains close to an identity transformation.

Different cameras may have different gain factors corresponding to their ISO levels. These camera-specific gain factors are usually proprietary and hard to access but may have a significant impact on the noise distribution of an image. To handle this, we use an additional set of parameters to adjust the gain layer for each camera. In this case, the above gain layer is adjusted by introducing a camera-specific scaling factor. That is,

$$\gamma(\text{ISO}, m) = \psi_m \times u(\text{ISO}) \times \text{ISO}, \quad (16)$$

where $\psi_m \in \mathbb{R}^+$ is the scaling factor for camera m. This is a simple model but was found to be effective to capture differences in gain factors between cameras.

4. Experiments

To assess the performance of Noise Flow, we train it to model the realistic noise distribution of the Smartphone Image Denoising Dataset (SIDD) [1] and also evaluate the sampling accuracy of the trained model.

4.1. Experimental Setup

Dataset We choose the SIDD for training our Noise Flow model. The SIDD consists of thousands of noisy and corresponding ground truth images, from ten different scenes, captured repeatedly with five different smartphone cameras under different lighting conditions and ISO levels. The ISO levels ranged from 50 to 10,000. The images are provided in both Raw-RGB and sRGB color spaces. We believe this dataset is the best fit to our task for noise modeling, mainly due to the great extent of variety in cameras, ISO levels, and lighting conditions.

Data preparation We start by collecting a large number of realistic noise samples from the SIDD. We obtain the noise layers by subtracting the ground truth images from the noisy ones. In this work, we use only raw-RGB images as they directly represent the noise distribution of the underlying cameras. We avoid using sRGB images as rendering image into sRGB space tends to significantly change the noise distribution [25]. We arrange the data as approximately $500,000$ image patches of size 64×64 pixels. We split the data into a training set $\mathcal{D}_{\text{r}}$ of approximately 70% of the data and a testing set $\mathcal{D}_{\text{s}}$ of approximately 30% of the data. We ensure that the same set of cameras and ISO levels is represented in both the training and testing sets. For visualization only, we render raw-RGB images through a color processing pipeline into sRGB color space.

The SIDD provides only the gain amplified clean image $\mathbf{I}_\gamma$ and not the true latent clean image $\mathbf{I}$. To handle this, we use the learned gain parameter γ to correct for this and estimate the latent clean image as $\mathbf{I} = \mathbf{I}_\gamma / \gamma$ when it is needed in the signal-dependant layer.

Loss function and evaluation metrics We train Noise Flow as a density estimator of the noise distribution of the dataset which can be also used to generate noise samples from this distribution. For density estimation training, we use the negative log likelihood (NLL) of the training set (see Equation 8) as the loss function which is optimized using Adam [16]. For evaluation, we consider the same NLL evaluated on the test set.

To provide further insight in the differences between the approaches, we also consider the Kullback-Leibler (KL) divergence of the pixel-wise marginal distributions between generated samples and test set samples. Such a measure ignores the ability of a model to capture correlations but focuses on a model's ability to capture the most basic characteristics of the distribution. Specifically, given an image from the test set, we generate a noise sample from the model and compute histograms of the noise values from the test image and the generated noise and report the discrete KL divergence between the histograms.

Baselines We compare the Noise Flow models against two well-established baseline models. The first is the ho-

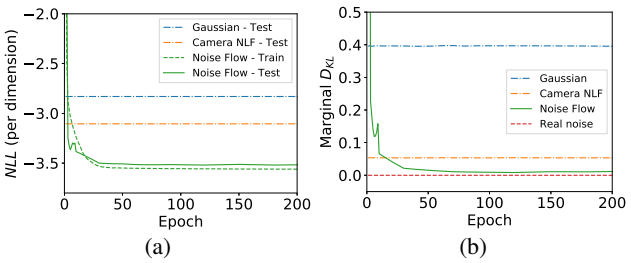


Figure 4: (a) NLL per dimension on the training and testing sets of Noise Flow compared to (1) the Gaussian model and (2) the signal-dependent model as represented by the camera-estimated NLFs. (b) Marginal KL divergence (D_{KL}) between the generated and the real noise samples.

	Gaussian	Cam. NLF	Noise Flow
NLL	-2.831 (99.4%)	-3.105 (51.6%)	**-3.521**
D_{KL}	0.394 (97.9%)	0.052 (84.1%)	**0.008**

Table 1: Best achieved testing NLL and marginal D_{KL} for Noise Flow compared to the Gaussian and Camera NLF baselines. Relative improvements of Noise Flow on other baselines, in terms of *likelihood*, are in parentheses.

moscedastic Gaussian noise model (i.e., AWGN) defined in Equation 2. We prepare this baseline model by estimating the maximum likelihood estimate (MLE) of the noise variance of the training set, assuming a univariate Gaussian distribution. The second baseline model is the heteroscedastic Gaussian noise model (i.e., NLF), described in Equations 5 and 6, as provided by the camera devices. The SIDD provides the camera-calibrated NLF for each image. We use these NLFs as the parameters of the heteroscedastic Gaussian model for each image. During testing, we compute the NLL of the testing set against both baseline models.

4.2. Results and Ablation Studies

Noise Density Estimation Figure 4a shows the training and testing NLL on the SIDD of Noise Flow compared to (1) the Gaussian noise model and (2) the signal-dependent noise model as represented by the camera-estimated noise level functions (NLFs). It is clear that Noise Flow can model the realistic noise distribution better than Gaussian and signal-dependent models. As shown in Table 1, Noise Flow achieves the best NLL, with 0.69 and 0.42 nats/pixel improvement over the Gaussian and camera NLF models, respectively. This translates to 99.4% and 51.6% improvement in likelihood, respectively. We calculate the improvement in likelihood by calculating the corresponding improvement in $\exp(-NLL)$.

Noise Synthesis Figure 4b shows the average marginal KL divergence between the generated noise samples and the corresponding noise samples from the testing set for the three models: Gaussian, camera NLF, and Noise Flow. Noise Flow achieves the best KL divergence, with 97.9% and 84.1% improvement over the Gaussian and camera NLF models, respectively, as shown in Table 1.

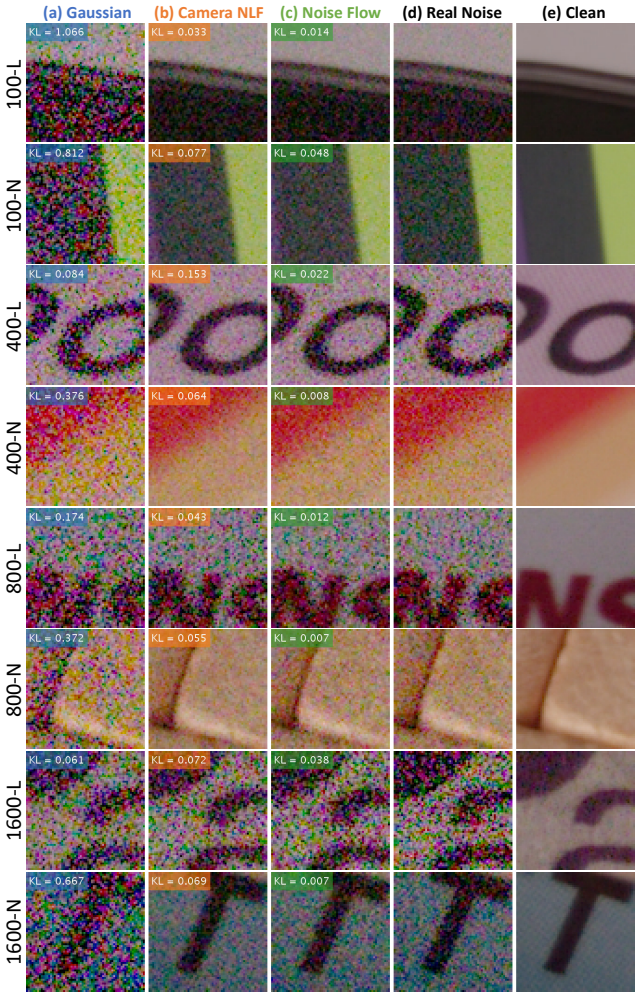
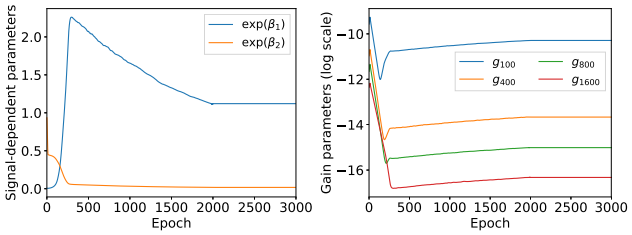


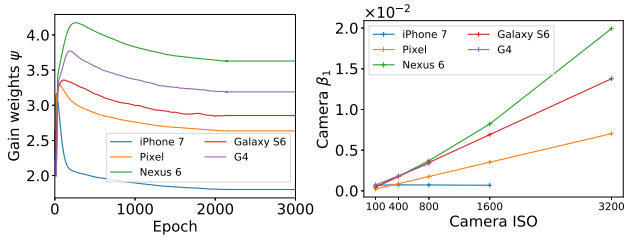
Figure 5: Generated noise samples from (c) Noise Flow are much closer, in terms of marginal KL divergence, to (d) the real samples; compared to (a) Gaussian and (b) camera NLF models. (e) Clean image. Corresponding ISO levels and lighting conditions are on the left.

Figure 5 shows generated noise samples from Noise Flow compared to samples from Gaussian and camera NLF models. We show samples from various ISO levels $\{100, \ldots, 1600\}$ and lighting conditions (N: normal light, L: low light). Noise Flow samples are the closest to the real noise distribution in terms of the marginal KL divergence. Also, there are more noticeable visual similarities between Noise Flow samples and the real samples compared to the Gaussian and camera NLF models.



(a) Signal-dependent parameters (b) Gain parameters

Figure 6: (a) Signal-dependent noise parameters β_1 and β_2 are consistent with the signal-dependent noise model where β_1 is dominant and β_2 is much smaller. (b) The gain parameters, in log scale, are consistent with the corresponding ISO levels shown in the legend.



(a) Camera gain weights (b) Camera β_1 values

Figure 7: (a) Learned camera-specific weights for the shared gain layer indicates differences in the gain of different cameras. These gains are correlated with the cameras' different values for NLF parameter β_1 shown in (b). The iPhone and G4 cameras have smaller ranges of ISO values in the SIDD dataset and hence their correlation with the gains is not clear.

Learning signal-dependent noise parameters Figure 6a shows the learning of the signal-dependent noise parameters β_1 and β_2 as defined in Equation 6 while training a Noise Flow model. The parameters are converging towards values that are consistent with the signal-dependent noise model where β_1 is the dominant noise factor that represents the Poisson component of the noise and β_2 is the smaller factor representing the additive Gaussian component of the noise. In our experiments, the shown parameters are run through an exponential function to force their values to be strictly positive.

Learning gain factors Figure 6b shows the learning of the gain factors as defined in Equation 14 while training a Noise Flow model. The gain factors $\{\gamma_{100}, \ldots, \gamma_{1600}\}$ are consistent with the corresponding ISO levels indicated by the subscript of each gain factor. This shows the ability of the Noise Flow model to properly factor the sensor gain in the noise modeling and synthesis process. Note that we omitted ISO level 200 from the training and testing sets because there are not enough images from this ISO level in the SIDD.

Model	NLL	D_{KL}
S-G	-3.431 (9.42%)	0.067 (88.1%)
S-G-CAM	-3.511 (1.01%)	0.010 (20.0%)
S-Ax1-G-Ax1-CAM	-3.518 (0.30%)	0.009 (11.1%)
S-Ax4-G-Ax4-CAM (Noise Flow)	**-3.521**	**0.008**

Table 2: Best achieved testing NLL and marginal D_{KL} for different layer architectures. The symbols S, G, CAM, Ax1, and Ax4 indicate a signal layer, gain layer, camera-specific parameters, one unconditional flow step, and four unconditional flow steps, respectively. Relative improvements of Noise Flow on each of the other architectures, in terms of likelihood, are in parentheses.

Learning camera-specific parameters In our Noise Flow model, the camera-specific parameters consist of a set of gain scale factors $\{\psi_m\}$, one for each of the five cameras in the SIDD. Figure 7 shows these gain scales for each camera in the dataset during the course of training. It is clear that there are differences between cameras in the learned gain behaviours. These differences are consistent with the differences in the noise level function parameter β_1 of the corresponding cameras shown in Figure 7b and capture fundamental differences in the noise behaviour between devices. This demonstrates the importance of the camera-specific parameters to capture camera-specific noise profiles. Training Noise Flow for a new camera can be done by fine-tuning the camera-specific parameters within the gain layers; all other layers (i.e., the signal-dependent and affine coupling layers) can be considered non-camera-specific.

Effect of individual layers Table 2 compares different architecture choices for our Noise Flow model. We denote the different layers as follows: G: gain layer; S: signal-dependent layer; CAM: a layer using camera-specific parameters; Ax1: one unconditional flow step (an affine coupling layer and a 1×1 convolutional layer); Ax4: four unconditional flow steps. The results show a significant improvement in noise modeling (in terms of NLL and D_{KL}) resulting from the additional camera-specific parameters (i.e., the S-G-CAM model), confirming the differences in noise distributions between cameras and the need for camera-specific noise parameters. Then, we show the effect of using affine coupling layers and 1×1 convolutional layers in our Noise Flow model. Adding the Ax1 blocks improves the modeling performance in terms of NLL. Also, increasing the number of unconditional flow steps from one to four introduces a slight improvement as well. This indicates the importance of affine coupling layers in capturing additional pixel-correlations that cannot be directly modeled by the signal-dependency or the gain layers. The S-Ax4-G-Ax4-CAM is the final Noise Flow model.

5. Application to Real Image Denoising

Preparation To further investigate the accuracy of the Noise Flow model, we use it as a noise generator to train an image denoiser. We use the DnCNN image denoiser [33]. We use the clean images from the SIDD-Medium [1] as training ground truth and the SIDD-Validation as our testing set. The SIDD-Validation contains both real noisy images and the corresponding ground truth. We compare three different cases for training DnCNN using synthetically generated noise: (1) DnCNN-Gauss: homoscedastic Gaussian noise (i.e., AWGN); (2) DnCNN-CamNLF: signal-dependent noise from the camera-calibrated NLFs; and (3) DnCNN-NF: noise generated from our Noise Flow model. For the Gaussian noise, we randomly sample standard deviations from the range $\sigma \in [0.24, 11.51]$. For the signal-dependent noise, we randomly select from a set of camera NLFs. For the noise generated with Noise Flow, we feed the model with random camera identifiers and ISO levels. The σ range, camera NLFs, ISO levels, and camera identifiers are all reported in the SIDD. Furthermore, in addition to training with synthetic noise, we also train the DnCNN model with real noisy/clean image pairs from the SIDD-Medium and no noise augmentation (indicated as DnCNN-Real).

Results and discussion Table 3 shows the best achieved testing peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [30] of DnCNN using the aforementioned three noise synthesis strategies and the discriminative model trained on real noise. The model trained on noise generated from Noise Flow yields the highest PSNR and SSIM values, even slightly higher than DnCNN-Real due to the relatively limited number of samples in the training dataset. We also report, in parentheses, the relative improvement introduced by DnCNN-NF over the other two models in terms of root-mean-square-error (RMSE) and structural dissimilarity (DSIMM) [21, 31], for PSNR and SSIM, respectively. We preferred to report relative improvement in this way because PSNR and SSIM tend to saturate as errors get smaller; conversely, RMSE and DSSIM do not saturate. For visual inspection, in Figure 8, we show some denoised images from the best trained model from the three cases, along with the corresponding noisy and clean images. DnCNN-Gauss tends to over-smooth noise, as in rows 3 and 5, while DnCNN-CamNLF frequently causes artifacts and pixel saturation, as in rows 1 and 5. Although DnCNN-NF does not consistently yield the highest PSNR, it is the most stable across all six images. Noise Flow can be used beyond image denoising in assisting computer vision tasks that require noise synthesis (e.g., robust image classification [4] and burst image deblurring [2]. In addition, Noise Flow would give us virtually unlimited noise samples compared to the limited numbers in the datasets.

Model	PSNR	SSIM
DnCNN-Gauss	43.63 (43.0%)	0.968 (75.6%)
DnCNN-CamNLF	44.99 (33.4%)	0.982 (56.0%)
DnCNN-NF	**48.52**	**0.992**
DnCNN-Real	47.08 (15.3%)	0.989 (27.5%)

Table 3: DnCNN denoiser [33] trained on synthetic noise generated with Noise Flow (DnCNN-NF) achieves higher PSNR and SSIM values compared to training on synthetic noise, from a Gaussian model or camera NLFs, and real noise. Relative improvements of DnCNN-NF over other models, in terms of RMSE and DSSIM, are in parentheses.

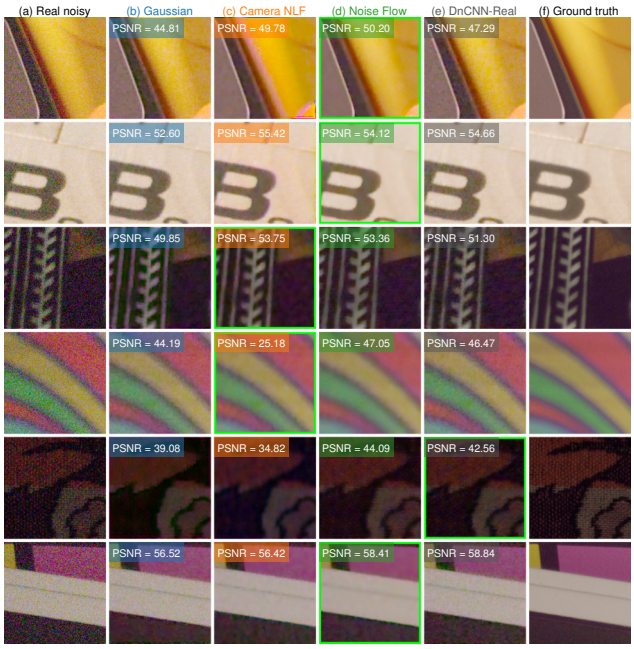


Figure 8: Sample denoising results from DnCNN trained on three different noise synthesis methods: (b) Gaussian; (c) camera NLF; and (d) Noise Flow. (e) DnCNN trained on real noise. (a) Real noisy image. (f) Ground truth.

6. Conclusion

In this paper, we have presented a conditional normalizing flow model for image noise modeling and synthesis that combines well-established noise models and the expressiveness of normalizing flows. As an outcome, we provide a compact noise model with fewer than 2500 parameters that can accurately model and generate realistic noise distributions with 0.42 nats/pixel improvement (i.e., 52% higher likelihood) over camera-calibrated noise level functions. We believe the proposed method and the provided model will be very useful for advancing many computer vision and image processing tasks. The code and pre-trained models are publicly available at: https://github.com/BorealisAI/noise_flow.

Acknowledgments

This work was supported by Mitacs through the Mitacs Accelerate Program as part of an internship at Borealis AI. This study was also funded in part by the Canada First Research Excellence Fund for the Vision: Science to Applications (VISTA) programme and an NSERC Discovery Grant. Dr. Brown contributed to this article in his personal capacity as a professor at York University. The views expressed are his own and do not necessarily represent the views of Samsung Research. Abdelrahman is partially supported by an AdeptMind scholarship.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A High-Quality Denoising Dataset for Smartphone Cameras. In *CVPR*, 2018. 1, 2, 4, 5, 8

[2] Miika Aittala and Frédo Durand. Burst Image Deblurring Using Permutation Invariant Convolutional Neural Networks. In *ECCV*, 2018. 8

[3] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image Blind Denoising with Generative Adversarial Network Based Noise Modeling. In *CVPR*, 2018. 2

[4] Steven Diamond, Vincent Sitzmann, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty Pixels: Optimizing Image Classification Architectures for Raw Sensor Data. *arXiv preprint arXiv:1701.06487*, 2017. 8

[5] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. In *ICLR Workshop*, 2015. 2, 3

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation using Real NVP. In *ICLR*, 2017. 3

[7] Alessandro Foi. Clipped Noisy Images: Heteroskedastic Modeling and Practical Denoising. *Signal Processing*, 89(12):2609–2629, 2009. 1, 2

[8] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data. *TIP*, 17(10):1737–1754, 2015. 1, 2

[9] Ryan D. Gow, David Renshaw, Keith Findlater, Lindsay Grant, Stuart J. McLeod, John Hart, and Robert L. Nicol. A Comprehensive Tool for Modeling CMOS Image-Sensor-Noise Performance. *IEEE Transactions on Electron Devices*, 54(6):1321–1329, 2007. 4

[10] Samuel W Hasinoff, Frédo Durand, and William T Freeman. Noise-Optimal Capture for High Dynamic Range Photography. In *CVPR*, 2010. 4

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 3

[12] Glenn E. Healey and Raghava Kondepudy. Radiometric CCD Camera Calibration and Noise Estimation. *TPAMI*, 16(3):267–276, 1994. 1, 4

[13] Gerald C Holst. *CCD Arrays, Cameras, and Displays*. SPIE Optical Engineering Press, USA, second edition, 1998. 1

[14] Youngbae Hwang, Jun Sik Kim, and In So Kweon. Difference-based Image Noise Modeling Using Skellam Distribution. *TPAMI*, 34(7):1329–1341, 2012. 1

[15] Xiaodan Jin and Keigo Hirakawa. Approximations to Camera Sensor Noise. In *Image Processing: Algorithms and Systems XI*, 2013. 1

[16] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 5

[17] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*, 2018. 2, 3, 4

[18] Darwin T Kuan, Alexander A Sawchuk, Timothy C Strand, and Pierre Chavel. Adaptive Noise Smoothing Filter for Images with Signal-Dependent Noise. *TPAMI*, 7(2):165–177, 1985. 1

[19] Ce Liu, Richard Szeliski, Sing Bing Kang, C. Lawrence Zitnick, and William T. Freeman. Automatic Estimation and Removal of Noise from a Single Image. *TPAMI*, 30(2):299–314, 2008. 1, 4

[20] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical Signal-Dependent Noise Parameter Estimation from a Single Noisy Image. *TIP*, 23(10):4361–4371, 2014. 2

[21] Artur Łoza, Lyudmila Mihaylova, David Bull, and Nishan Canagarajah. Structural Similarity-based Object Tracking in Multimodality Surveillance Videos. *Machine Vision and Applications*, 20(2):71–83, 2009. 8

[22] Markku Mäkitalo and Alessandro Foi. Optimal Inversion of the Generalized Anscombe Transformation for Poisson-Gaussian Noise. *TIP*, 22(1):91–103, 2013. 1, 2

[23] Amr M. Mohsen, Michael F. Tompsett, and Carlo H. Sèquin. Noise Measurements in Charge-Coupled Devices. *IEEE Transactions on Electron Devices*, 22(5):209–218, 1975. 2

[24] Firouz Naderi and Alexander A Sawchuk. Estimation of Images Degraded by Film-Grain Noise. *Applied Optics*, 17(8):1228–1237, 1978. 1

[25] Seonghyeon Nam, Youngbae Hwang, Keti Yasuyuki Matsushita, and Seon Joo Kim. A Holistic Approach to Cross-Channel Image Noise Modeling and its Application to Image Denoising. In *CVPR*, 2016. 5

[26] Tobias Plötz and Stefan Roth. Benchmarking Denoising Algorithms with Real Photographs. In *CVPR*, 2017. 1, 2

[27] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *ICML*, 2015. 2

[28] Tamara Seybold, Christian Keimel, Marion Knopp, and Walter Stechele. Towards an Evaluation of Denoising Algorithms with Respect to Realistic Camera Noise. In *IEEE International Symposium on Multimedia*, 2013. 1

[29] H Joel Trussell and R Zhang. The Dominance of Poisson Noise in Color Digital Cameras. In *ICIP*, 2012. 1

[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. Technical Report 4, 2004. 8

[31] Andrew R Webb. *Statistical Pattern Recognition*. John Wiley & Sons, 2003. 8

[32] Jiachao Zhang and Keigo Hirakawa. Improved Denoising via Poisson Mixture Modeling of Image Sensor Noise. *TIP*, 26(4):1565–1578, 2017. 1, 2

[33] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *TIP*, 26(7):3142–3155, 2017. 8