Modeling sRGB Camera Noise with Normalizing Flows —Supplemental Material—

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

1. Scale and Translation Functions

Here we provide more details about the normalzing flow architecture used in the main paper.

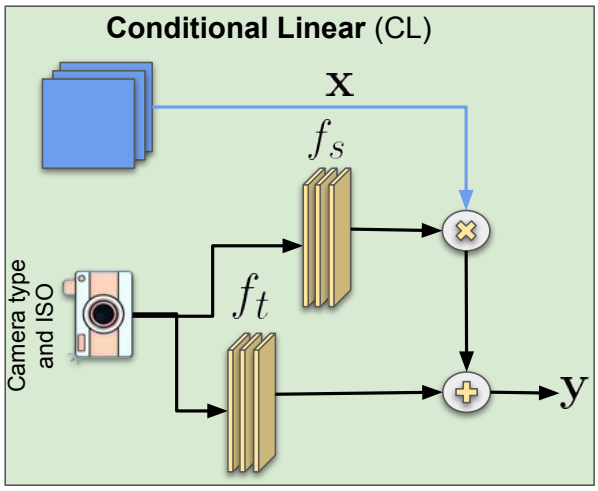

Figure 1. Forward pass of the conditional linear flow. Functions f_s and f_t are responsible for calculating the scale and bias terms, respectively. These terms are later applied to the input image $\mathbf{x}$ to calculate the output of this layer.

Conditional Linear Flow To condition on camera types and ISO levels, this layer creates a one-hot encoding of the camera-ISO pairs. For example, in our training and testing steps, we use images from five smartphones with five different ISO levels. This means the one-hot encoding of the pairs is a vector of size 25. $f_s : \mathbb{R}^{25} \rightarrow \mathbb{R}^3$ and $f_t : \mathbb{R}^{25} \rightarrow \mathbb{R}^3$ functions have one scale and bias parameter for each channel for a given camera-iso representation. Since we have 25 camera-ISO pairs and are working with three-channel sRGB data, each of the two functions has a set of parameters with shape $(25, 3)$. The two functions use the one-hot vectors to index into these parameters. As a result, the output of each function is a vector of size 3 (one value per channel).

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

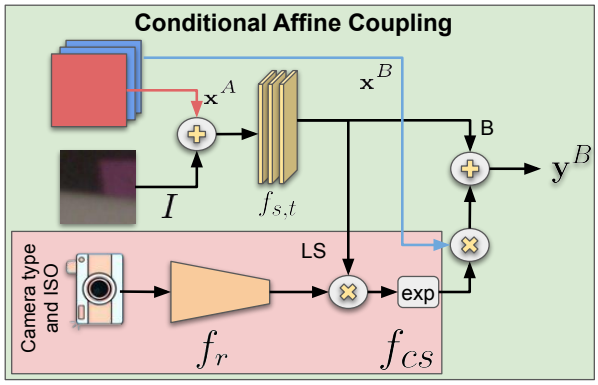

Figure 2. Forward pass of the conditional affine coupling layer. $f_{s,t}$ calculates the bias (B) and log_scale (LS) from the clean image and one channel of the input data. A rescaling term is calculated from the camera type and ISO setting and applied to LS by f_{cs} function. Later the rescaled LS is used along with the bias term and the remaining two channels of the input image to calculate $\mathbf{y}^B$. $\mathbf{y}^B$ is one part of the output.

The values corresponding to each channel are later applied to all pixels in that channel.

Conditional Affine Coupling As shown in Figure 2, to condition on all three variables, namely the clean image, camera type, and ISO level, f_{cs} and f_{ct} functions consider these variables in two steps. First, the clean image is concatenated with one subset of the input data and passed to a CNN model ($f_{s,t}$) to calculate the log scale and bias factors. Then, the camera type and gain setting are encoded into one-hot vectors separately. In our case, each encoding vector has a size of 5. The one-hot encoding vectors are concatenated and passed to a residual network ($f_r : \mathbb{R}^{10} \rightarrow \mathbb{R}$) to calculate a rescaling factor. This rescaling factor is multiplied by the log scale factor calculated earlier to get the final scaling values. The bias term is returned untouched.

These functions have the form:

$$\mathbf{LS}, \mathbf{B} = split(f_{s,t}(\mathbf{x}^A, \mathbf{I}))$$
$$f_{ct}(\mathbf{x}^A|\mathbf{I}, \mathbf{c}, \mathbf{g}) = \mathbf{B}$$
$$f_{cs}(\mathbf{x}^A|\mathbf{I}, \mathbf{c}, \mathbf{g}) = \exp(\mathbf{LS} * f_r(\mathbf{c}, \mathbf{g}))$$

where f_e is the one-hot encoder, and $\mathbf{LS}$ and $\mathbf{B}$ are the log scale and bias factors, respectively.

2. Inverse Gamma

RAW-rgb images go through an in-camera imaging pipeline that transforms the image from the RAW-rgb space to the sRGB domain. The steps in this pipeline introduce non-linearities that result in a complex noise distribution in the sRGB space. One of the main nonlinear steps in this pipeline is gamma correction which is an invertible process. Its inverse is commonly used to approximately linearize sRGB data and can easily be implemented as a normalizing flow transformation. Inverse gamma is defined as $\mathbf{y} = \mathbf{x}^{\mathbf{gamma}}$ where the default value of $\mathbf{gamma}$ is 2.2. In this section, we explore the effects this layer has on Noise Flow and our proposed model when added to the data processing step.

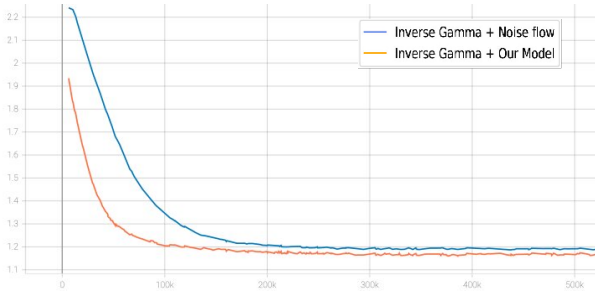

Figure 3. $\mathbf{gamma}$ value changes in the training process. $\mathbf{gamma}$ converges to 1.2 for both models.

Model	NLL	D_{KL}	$\#Params$
Noise flow	3.311	0.198	2330
Our model	**3.072**	**0.044**	6160
Inverse gamma + Noise flow	3.322	0.181	2332
Inverse gamma + Our model	3.092	0.061	6162

Table 1. Models with the inverse gamma transformation added to their data processing step have similar performances as the original models.

Table 1 shows that using the inverse gamma layer does not improve the modeling and sampling performances. Figure 3 shows that $\mathbf{gamma}$ gamma converges to 1.2 from the initial value of 2.2 resulting in a near identity transformation for both models.

3. Architecture Search

There are many more approaches and flow based transformations that can be used to form the flow block of our model. Here we introduce a few more novel conditional transformations and explore their modeling capabilities.

3.1. Conditional Transformations

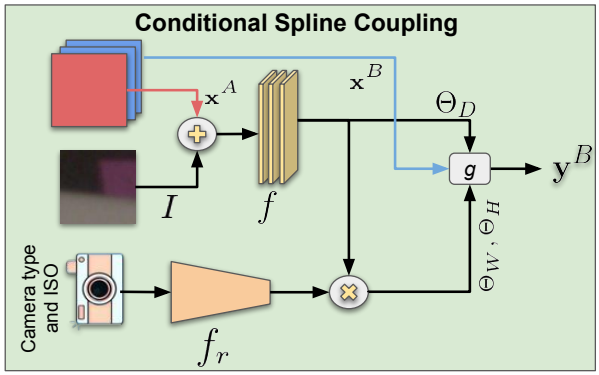

Figure 4. Forward pass of the conditional spline coupling layer. Function f calculates the bins' parameters from the clean image and one of the input channels. Later some of these parameters are rescaled using the rescale term calculated by f_r from the one-hot encodings of camera and gain settings. Finally, the updated parameters are passed to function g introduced in the neural spline paper [1]. This function is responsible for calculating the output of this layer.

Conditional Spline Coupling (CSC) This layer is an extension of the neural spline layer [1]. We extend the neural spline transformation to condition on the clean image, camera type and gain setting. Here, we outline the procedure to condition on these variables:

As shown in Figure 4, the clean image, I, is concatenated channel-wise with $\mathbf{x}^A$ and passed to function f to calculate the parameters of the bins. Then, similar to the conditional affine coupling layer, the camera and gain settings are encoded into one-hot vectors separately. $f_r : \mathbb{R}^{10} \rightarrow \mathbb{R}$, which is an arbitrary function, takes the resulting vectors as input and outputs a scale factor. In our experiments we use a residual network architecture. The scale factor is later used to rescale the bins' width and height parameters. The conditional spline coupling layer is defined as follows:

$$\Theta_W, \Theta_H, \Theta_D = f(\mathbf{x}^A, I)$$
$$\Theta_W, \Theta_H = (\Theta_W, \Theta_H) * f_r(\mathbf{c}, \mathbf{g})$$
$$\mathbf{y}^B = g(\mathbf{x}^B|\Theta_W, \Theta_H, \Theta_D)$$

where function g is the same function used in the standard neural spline flows. The inverse and log determinant of this

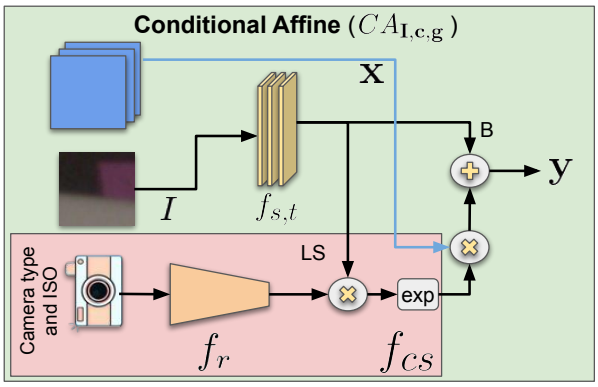

Figure 5. Forward pass of the conditional affine layer conditioned on the clean image, camera type and gain signal. $f_{s,t}$ calculates the bias (B) and log_scale (LS) from the clean image. The camera type and ISO setting are used to learn a rescaling factor which is later applied to LS by f_{cs} function. Finally, the rescaled LS is used along with the bias term and the input image to calculate $\mathbf{y}$.

layer can be calculated by following the same procedure used by the unconditional neural spline layer.

Similar to the affine coupling layer, the step that conditions on the clean image is independent of the step that conditions on the camera and gain. This allows us to use this layer in multiple ways in our experiments in Section 3.2.

Conditional Affine (CA) The nature of the noise and the subsequent non-linear processing is heavily determined by the underlying noise free image and the specific camera and gain (or ISO) settings used. To account for this, we introduce linear flow layers, which are conditioned on combinations of important variables including the camera, $\mathbf{c}$, gain setting, $\mathbf{g}$, and the underlying clean image, $\mathbf{I}$. This layer comes in two forms depending on which subset of variables it is considering:

Conditioned on Clean Image, Camera and ISO ($CA_{\mathbf{I},\mathbf{c},\mathbf{g}}$): This layer is similar to the conditional affine coupling layer. The only difference is that this layer does not split the input dimension. It calculates the scale and bias and applies them to all input dimensions. This layer has the following form:

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

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output of this layer, respectively. f_{cs} and f_{ct} are similar to the functions used by the affine coupling layer with not taking $\mathbf{x}^A$ as an input being the only difference. As a result, $f_{s,t}$, which is defined earlier, only uses the clean image to calculated the log scale and bias terms calculated. Later a rescaling factor is generated from the encoding of the camera type and ISO setting and applied to the log scale term. Similar to the CAC layer,

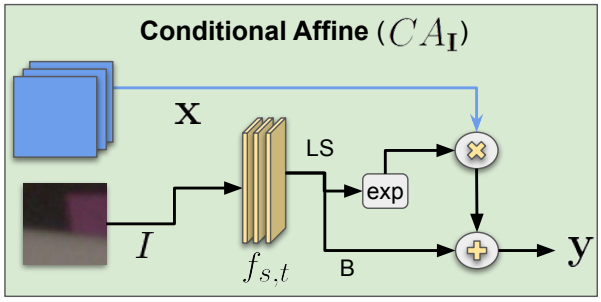

Figure 6. Forward pass of the conditional affine layer conditioned on the clean image. $f_{s,t}$ calculates the bias (B) and log_scale (LS) from the clean image. These factors are applied directly to the input image to calculate the output of this layer.

the inverse and log determinant of this transformation can be easily calculated.

Conditioned on Clean Image ($CA_{\mathbf{I}}$): f_{cs} and f_{ct} of $CA_{\mathbf{I},\mathbf{c},\mathbf{g}}$ use a two step process to calculate the scale and bias terms. First, they calculated the log scale and bias terms. Then, a rescale term is learned from variables such as gain and applied to the scale factor. These two steps are independent of each other and can be applied separately. In this layer, we only use the first step to learn the log scale and bias terms from the clean image. This layer has the following form:

$$\mathbf{LS}, \mathbf{B} = split(f_{s,t}((\mathbf{x}^A, \mathbf{I}))$$
$$\mathbf{y} = \mathbf{x} \odot \exp(\mathbf{LS}) + \mathbf{B}$$

Since the second step of f_{cs} and f_{ct} is not applied in this layer, information learned from the clean image is directly used to scale and shift the input data with out being rescaled. In Section 3.2 we investigate whether such direct transfer of information from the underlying clean representation to the noise image is beneficial.

3.2. Experiments

In this section, we consider multiple architectures for the flow block . The goal is to understand what combination of layers results in better modeling and generative performance. In our experiments, unless otherwise specified, all coupling layers are preceded by an invertible 1x1 convolution layer. Additionally, the coupling layers come in pairs. As a result, we have "x2" next to the coupling layers in our tables.

First, we investigate the effects of unconditional layers. Table 2 summarizes the results of experiment. The model defined in the first row uses two spline coupling layers as the unconditional step followed by some conditional layers. The model described in row two is similar to the first model with the exception that unconditional layers are eliminated. Even though this model has fewer parameters due

Uncond.	g and c	I	I, g, and c	NLL	D_{KL}
SCx2	CL	$\text{CA}_{\mathbf{I}}$	CSCx2	4.238	0.229
-	CL	$\text{CA}_{\mathbf{I}}$	CSCx2	**3.076**	0.141
ACx2	CL	$\text{CA}_{\mathbf{I}}$	CACx2	4.147	0.183
-	CL	$\text{CA}_{\mathbf{I}}$	CACx2	3.602	**0.104**

Table 2. **Unconditional layer experiment.** Models in this table use 1 flow block (S = 1). Removing the unconditional layer and keeping conditional layers untouched improves the performance of the models. This trend is independent of the type of unconditional layer used in these models. Models that use unconditional/conditional affine coupling layers achieve better D_{KL} compared to the models that use unconditional/conditional spline coupling layers. A lower D_{KL} suggests better generated samples.

to the lack of unconditional layers, it outperforms the first model in both metrics. Eliminating the unconditional layers results in a 1.162 nats/pixel improvement of NLL. The third row replaces the unconditional/conditional spline coupling transformations with unconditional/conditional affine coupling layers. A similar pattern of improvement emerges when this model is compared with the last row where the unconditional layers are removed. Our last model achieves an NLL of 3.602 which translates to a 0.545 nats/pixel improvement. Additionally, when compared to the second row, the last row reveals that the conditional affine coupling (CAC) layer is a better fit in terms of D_{KL} than the conditional spline couplin (CSC) layer for conditioning on the clean image, camera type, and gain setting. As a result, we use the model in row four as a base for our next experiment.

The next experiment is designed to show the effects of layers only conditioned on the clean image. The three models shown in Table 3 are identical with their choice of layer for conditioning on the clean image being the only difference. The results suggest removing the layer that only conditions on the clean image improves the performance in terms of D_{KL} but might worsen the NLL results. We find D_{KL} to be a better indicator of the quality of the samples and prefer simpler model. Therefore, we conclude that there is no need to have a specialized layer to solely condition on the clean image. As shown in Table 3 the model defined in row three achieves a D_{KL} value of 0.077 which is the best result achieved so far. As a result, we include this model in Table 4 to help with our next experiment.

The previous experiments show that having an unconditional layer and a layer that only conditions on the clean image is not necessary. The third experiment focuses on the conditional transformations that either utilize ISO and camera type information or use all three variables. Table 4 summarizes the experiment and shows CL_CACx2 model and CL_$\text{CA}_{\mathbf{I},\mathbf{c},\mathbf{g}}$ with one flow block achieve the best D_{KL} performances with values of 0.077 and 0.063, respectively. Given the architectural design choices we made, the models

g and c	I	I, g, and c	NLL	D_{KL}
CL	$\text{CA}_{\mathbf{I}}$	CACx2	3.602	0.104
CL	CSCx2	CACx2	**2.901**	0.163
CL	-	CACx2	3.473	**0.077**

Table 3. **Clean image only experiment.** Models in this table use 1 flow block (S = 1). Adding a specialized layer to only condition on the clean image harms D_{KL} but it might improve (reduce) NLL. We consider two specialized layers for conditioning on the clean image, namely $\text{CA}_{\mathbf{I}}$ and CSCx2. They both fail to improve the model performance in terms of D_{KL}.

ISO and Cam	Clean, ISO, and Cam	NLL	D_{KL}
$\text{CA}_{\mathbf{c},\mathbf{g}}$	CACx2	3.473	0.077
$\text{CA}_{\mathbf{c},\mathbf{g}}$	CSCx2	**3.064**	0.159
$\text{CA}_{\mathbf{c},\mathbf{g}}$	$\text{CA}_{\mathbf{I},\mathbf{c},\mathbf{g}}$	3.636	**0.063**
CACx2	$\text{CA}_{\mathbf{I},\mathbf{c},\mathbf{g}}$	3.522	0.099

Table 4. **Conditioning on all variables.** Models in this table use 1 flow block (S = 1). The choice of transformation for conditioning on ISO, camera type, and clean image has a significant impact on the overall performance of the normalizing flows model. $\text{CA}_{\mathbf{c},\mathbf{g}}$ seems to be the best choice for conditioning on gain and camera as the models containing this layer achieve the best NLL and D_{KL}.

# Flow Blocks (S)	g and c	I, g, and c	NLL	D_{KL}
1	$\text{CA}_{\mathbf{c},\mathbf{g}}$	CACx2	3.473	0.077
1	$\text{CA}_{\mathbf{c},\mathbf{g}}$	$\text{CA}_{\mathbf{I},\mathbf{c},\mathbf{g}}$	3.636	0.063
4	$\text{CA}_{\mathbf{c},\mathbf{g}}$	CACx2	**3.072**	**0.044**
4	$\text{CA}_{\mathbf{c},\mathbf{g}}$	$\text{CA}_{\mathbf{I},\mathbf{c},\mathbf{g}}$	3.639	0.060

Table 5. **Model depth.** Increasing the number of flow blocks improves the noise modeling capabilities of the NF models. The model in row three achieves the best performance in terms of D_{KL} compared to other models.

can easily be made deeper by increasing the number blocks.

Table 5 shows the performance of these two models when the number of blocks is 4. The results suggest making a model deeper improves the results. For example, CL_CACx2 with 4 repeated flow blocks achieves the best NLL with 0.401 nats/pixel improvement over the same model architecture with only one flow block. CL_CACx2 model with four blocks outperforms the other models in this table in both metrics. It achieves NLL and D_{KL} of 3.072 and 0.044, respectively. This is the same model as the model we introduced in the paper.

References

[1] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows," in *Neurips*, 2019. 2

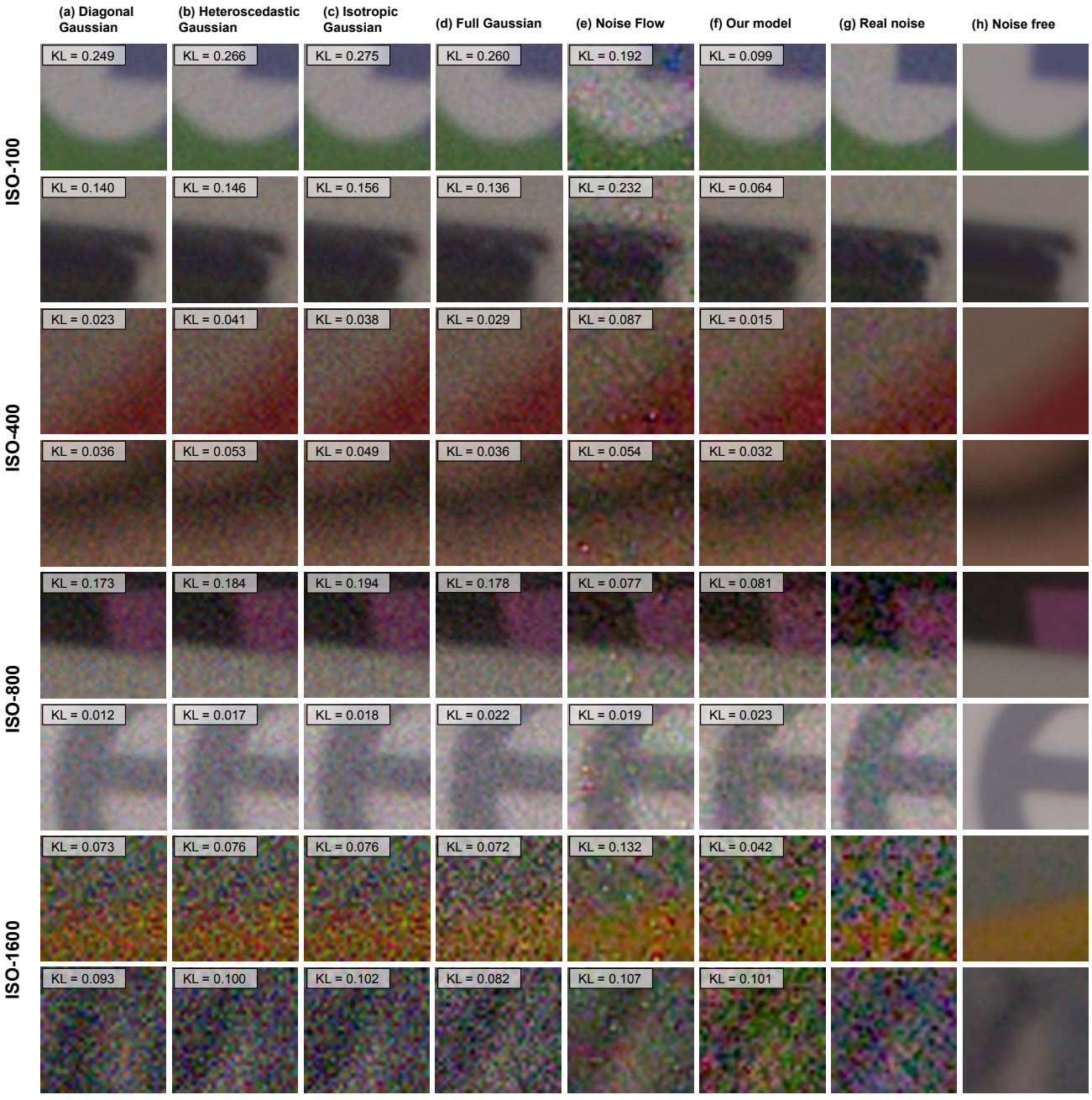

Figure 7. Generated samples from our model and all baselines. Samples from our model have noticeable visual similarities with the real noisy samples. Our samples achieve the lowest D_{KL} in almost all cases, showing its ability to generate realistic noise.

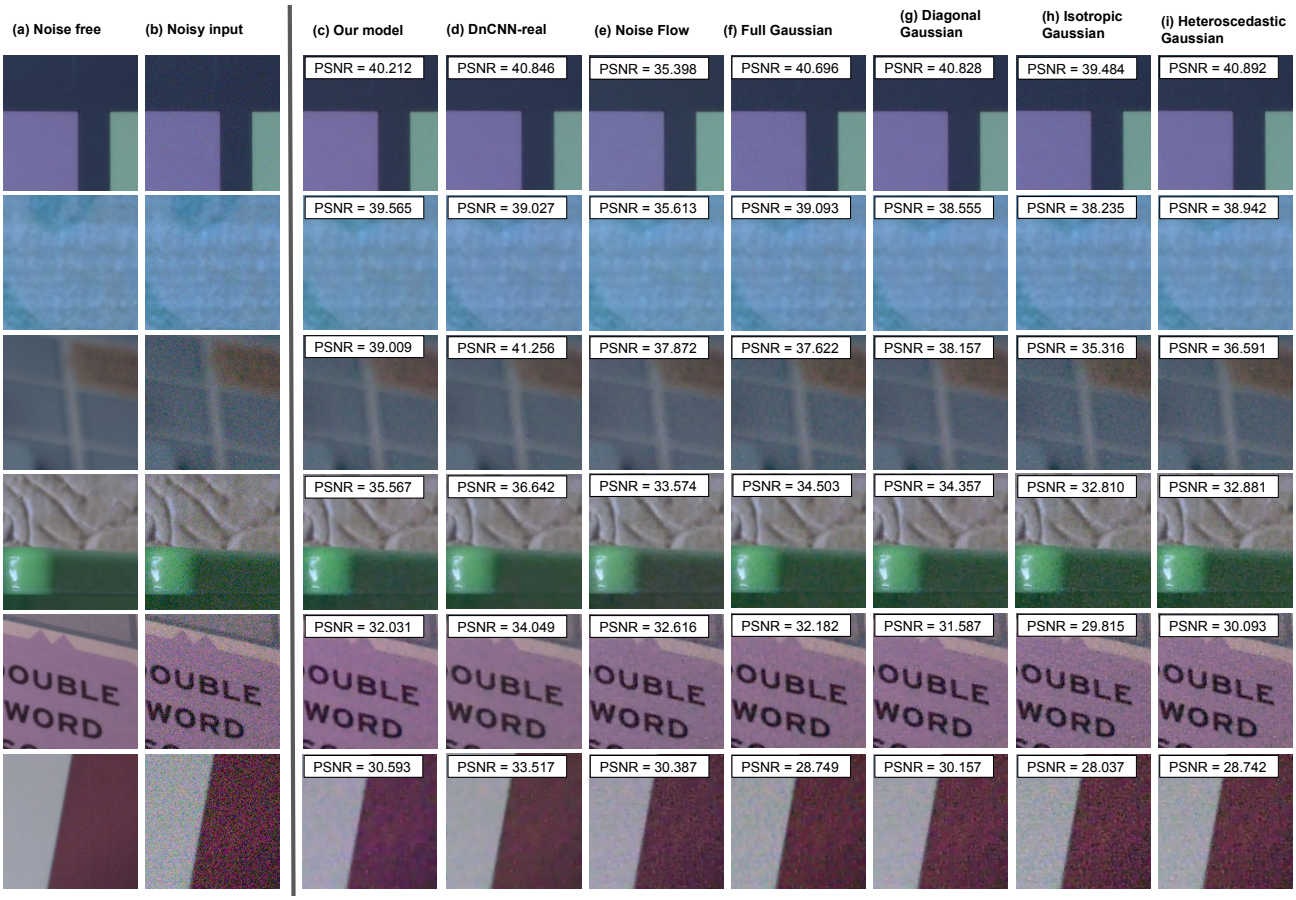

Figure 8. Denoising results on SIDD-Validation from denoisers trained on noisy images from (c) real noisy images of SIDD-Validation, (d) our model, and (e, i) all of our baselines.