

SUPPLEMENTARY MATERIAL: TEST-TIME ADAPTATION FOR REAL-WORLD DENOISING NETWORKS VIA NOISE-AWARE IMAGE GENERATION

Anonymous authors

Paper under double-blind review

A NAN ARCHITECTURE

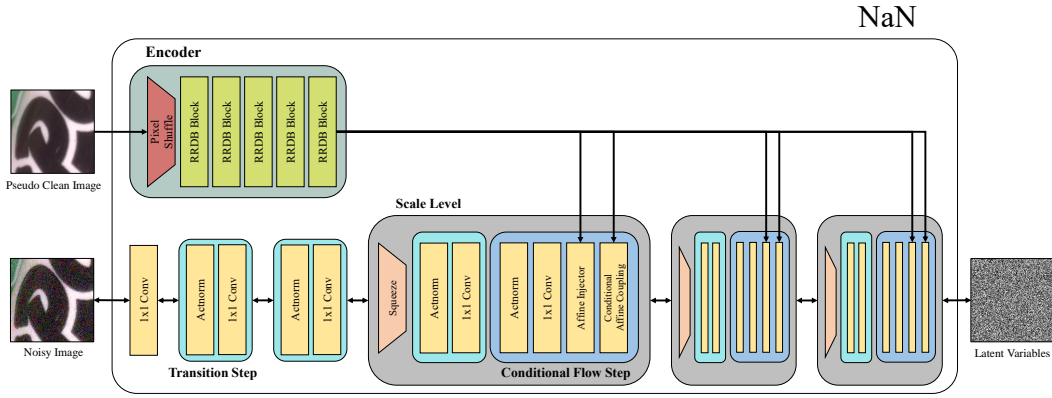


Figure S1: Architecture overview of NaN.

A.1 IMPLEMENTATION DETAILS

The overall architecture of our NaN framework is based on SRFlow (Lugmayr et al., 2020). SRFlow is one of the successful instances of a conditional normalizing flow (NF) framework designed for low-level computer vision tasks. We have modified the SRFlow network to generate noise conditioned on a noisy image and the corresponding pseudo clean image, as illustrated in Figure S1. In particular, our NaN network is composed of three scale level blocks ($L = 3$). Responsible for handling different scales of noise, each scale level block consists of one conditional flow step block ($K = 1$), one transition step block, and a squeeze operation. In turn, each transition step block consists of invertible 1x1 convolution (Kingma & Dhariwal, 2018) and ActNorm (Kingma & Dhariwal, 2018). Meanwhile, each conditional step block consists of Conditional Affine Injector (Lugmayr et al., 2020), Affine Injector (Lugmayr et al., 2020), and one transition step block. Similar to Low-Resolution Encoder in SRFlow, our conditional encoder is conditioned on a pseudo clean image to extract features, on which Affine Injector and Conditional Affine Coupling blocks are conditioned. The encoder consists of 5 RRDB blocks (Wang et al., 2018), along with a pixel shuffle operator (Shi et al., 2016) placed at the beginning to minimize the effect of remaining noise or artifact generated from the pseudo clean generator h . For more details on the pixel shuffle operator and its effect, refer to Section A.3. Furthermore, input noisy image is mapped from discrete space to continuous space via the addition of uniform noise from $U(0, \frac{1}{255})$ (Dinh et al., 2015). In contrast to SRFlow (Lugmayr et al., 2020), a squeeze operation is not used in each conditional flow step, and thus only one latent variable is used for simplicity.

Downsampling	SIDD validation		SIDD+		Nam		PolyU		Average	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$\times 1$	39.05	0.9156	36.20	0.9024	38.25	0.9579	38.14	0.9621	37.91	0.9345
$\times 2$	39.04	0.9155	36.23	0.9035	38.40	0.9581	38.19	0.9622	37.97	0.9348
$\times 4$	39.10	0.9160	36.31	0.9061	38.40	0.9560	38.20	0.9615	38.00	0.9349
$\times 8$	39.12	0.9158	36.48	0.9114	38.49	0.9579	38.21	0.9620	38.08	0.9368

Table S1: Denoising performance of our framework with different downscale factors ($\times 1$, $\times 2$, $\times 4$, $\times 8$) for pixel shuffle. Every model is trained using an equivalent setting of our final version, except for the pixel shuffle scale factor in Conditional Encoder of NaN.

A.2 TRAINING AND INFERENCE TIME

Our NaN has 8.2M parameters, which is comparable in size to other generative models, such as CycleISP (7.5M) and DANet (9.1M). Our final model is trained for 8 hours on a single Nvidia V100. As for inference, it takes 0.12 seconds to generate a single noisy image of size 160×160 .

A.3 IMPACT OF PIXEL SHUFFLE OPERATOR

As discussed in the main manuscript, ground-truth clean images are not available in the real scenario, and thus we use the pseudo clean image for conditioning the NF. In this work, we instantiate a pseudo clean generator as one of the conventional denoising networks, and the resulting image includes unexpected artifacts. To reduce such adverse effects within the pseudo clean image, we reshape the pseudo clean images using the pixel shuffle operator (Shi et al., 2016) which is frequently used in the image restoration tasks to enlarge the receptive field of the networks without information loss (Jin et al., 2018; Sajjadi et al., 2018). To better demonstrate the effect of the pixel shuffle operator, we provide denoising results on numerous datasets by changing the parameters for the reshaping in Table S1. We achieve the best results when we down-size the spatial resolution with $\times 8$ scaling factor and increase the channel dimension accordingly, and the experiments in the main manuscript are conducted with this setting.

B LATENT SPACE VISUALIZATION

To investigate the capability of NaN to learn the distribution of noises from different camera settings, we visualize the latent space of NaN (*i.e.*, $\{\mu_c\}$) in Figure S2. For visualization, we reduce the dimension of μ_c using principal component analysis (PCA) into 2D. Notably, latent variables corresponding to similar ISO settings are observed to be placed together (low ISO on the left and high ISO on the right). Another tendency can be observed based on the camera (*i.e.*, smartphone) model. For example, latent variables corresponding to iPhone tend to be on the negative side of PC2 axis (*i.e.*, bottom of the figure). Reasons for such observed trend are because ISO and camera models are among the most influential factors of noise. High ISO increases the sensitivity of camera sensors and thus increases the noise susceptibility as well. Furthermore, different smartphone (*i.e.*, camera) models have different amplifier, signal strength, and thus noise in imaging pipeline (Gow et al., 2007). Also, noise is affected by hardware and software factors which vary for each camera model (Abdelhamed et al., 2018).

C PSEUDO CLEAN GENERATOR

Since our framework relies on pseudo clean images to generate new noisy images, the quality of pseudo clean images and thus pseudo clean generators of NaN can affect the final denoising performance of our framework. To investigate the influence of pseudo clean generator on the denoising performance of our framework, we employ different denoising networks for pseudo clean generator and report the final denoising performance in Table S2. As one may expect, using a heavier network, RIDNet, for pseudo clean generator results in slightly better performance than DnCNN. However, the performance difference is relatively small, in comparison to the performance difference observed when different baselines g are used in Table 1 in the main manuscript. This results suggests that our framework is robust to the quality of pseudo clean images and pseudo clean generator to some ex-

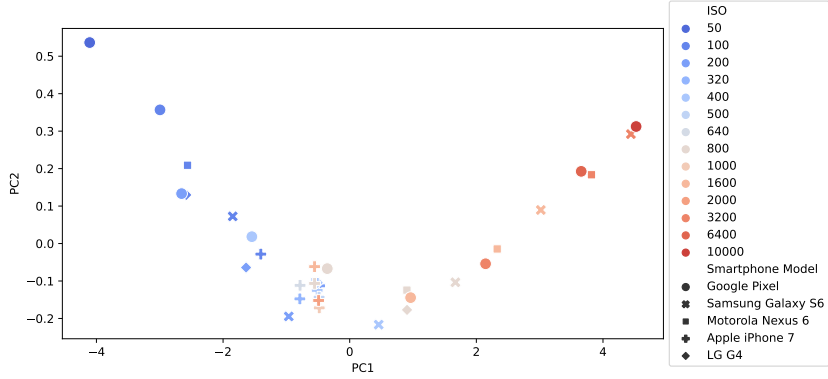


Figure S2: Visualization of the latent space ($\{\mu_c\}$) learned by our NaN in 2-D dimensional space. PCA is used to reduce the dimension of μ_c . Each point indicates the mean of normal distribution μ_c , while PC on each axis denotes each principle component of PCA.

tent. This is partly owed to pixel shuffle operator that reduces noise present in pseudo clean images, as discussed in Section A.3.

h	SIDD		SIDD+		Nam		PolyU		Average	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DnCNN	39.12	0.9158	36.48	0.9114	38.49	0.9579	38.21	0.9620	38.08	0.9368
RIDNet	39.12	0.9153	36.54	0.9128	38.54	0.9582	38.23	0.9619	38.11	0.9371

Table S2: Denoising performance of our framework with different pseudo clean generators h .

D KLD IMPLEMENTATION

KLD metric is sensitive to implementation setting such as the number of histogram bins and its calculation methods. Therefore, we describe our KLD implementation as follows:

$$D_{KL}(P||Q) = \sum_{i=0}^{255} P(i) \log\left(\frac{P(i)}{Q(i)}\right), \text{ where } P(i) \neq 0, Q(i) \neq 0. \quad (\text{A})$$

P and Q are probability density functions of real input noise and synthetic noise, respectively. Input noise and synthetic noise are obtained by subtracting ground-truth clean image from input noisy image and generated synthetic noisy image, respectively. Then, the probability density function of each noise is calculated using a histogram with 256 bins.

E TRAINING DETAIL

E.1 NAN

For training NaN, we use the SIDD sRGB train dataset with 34(=C) different camera configurations, and minimize the \mathcal{L}_{PNLL} loss in Equation using the Adam optimizer (Kingma & Ba, 2014) with initial learning rate 1e-4 which is reduced by half at 50k, 75k, 90k during 100k iterations. We use randomly cropped patches (160 x 160) and the mini-batch size of 8 for training. Note that, we use both the ground-truth clean and pseudo-clean images for conditioning while training the NaN, but only the pseudo clean image is used at the test-stage.

E.2 DENOISING NETWORK g_ϕ

In this paper, DnCNN, RIDNet, and HINet are used as our baseline denoisers. Since there are no official release of parameters trained on the SIDD dataset for DnCNN and RIDNet, we trained these networks on the SIDD dataset with the following training settings:

- **mini-batch size:** 8
- **patch-size:** randomly cropped 160×160 patches
- **loss function:** L1 loss minimization between the ground-truth clean image and network output
- **optimizer:** Adam (learning rate started from $1e-4$ and gradually decreased to $1e-6$ for RIDNet, and the learning rate fixed to $1e-5$ for DnCNN.)
- **augmentation:** flip, flop, and rotation

As for HINet, we use the available official parameters of network trained on the SIDD dataset.

E.3 MAML TRAINING WITH DENOISING NETWORK

In this work, we implement the MAML integrated algorithms using the Higher library (Grefenstette et al., 2019) with PyTorch. During training the denoising networks with MAML algorithm, we also use the Adam optimizer with inner-loop learning rate α and outer-loop learning rate β as follows:

- **DnCNN:** $\alpha = 1e-5, \beta = 1e-6$
- **RIDNet:** $\alpha = 5e-6, \beta = 1e-6$
- **HINet:** $\alpha = 5e-6, \beta = 1e-6$

F FAST TEST-TIME PARAMETER ADAPTATION (FPA)

Algorithm 1 Fast Test-time Parameter Adaptation (FPA)

Input: noisy input y

Require: meta-trained denoiser g_ϕ , adaptation number M , learning rate α

```

for  $i \leftarrow 1$  to  $M$  do
    Generate  $\tilde{y}$  from  $y$  according to NaN algorithm
     $\mathcal{L}_{N2N}(\phi^*) = ||g_{\phi^*}(\tilde{y}) - y||$ 
     $\phi = \phi - \alpha \nabla_{\phi^*} \mathcal{L}_{N2N}(\phi)$ 
end
return  $\phi$  // return input specific denoising network parameter

```

In Algorithm 1, we provide the pseudo-code for our FPA algorithm.

G QUALITATIVE RESULTS

We provide more visualizations of clean images recovered by AP-BSN and RIDNet baseline adapted with N2S and our algorithms (TPA and FPA) from real-world noisy images on SIDD+ dataset (Figure S3) and PolyU dataset (Figure S4). AP-BSN is shown to often produces images with too much blur while RIDNet adapted with N2S provides images with visible noise. On the other hand, RIDNet adapted with our algorithm produces sharper images with less noise.

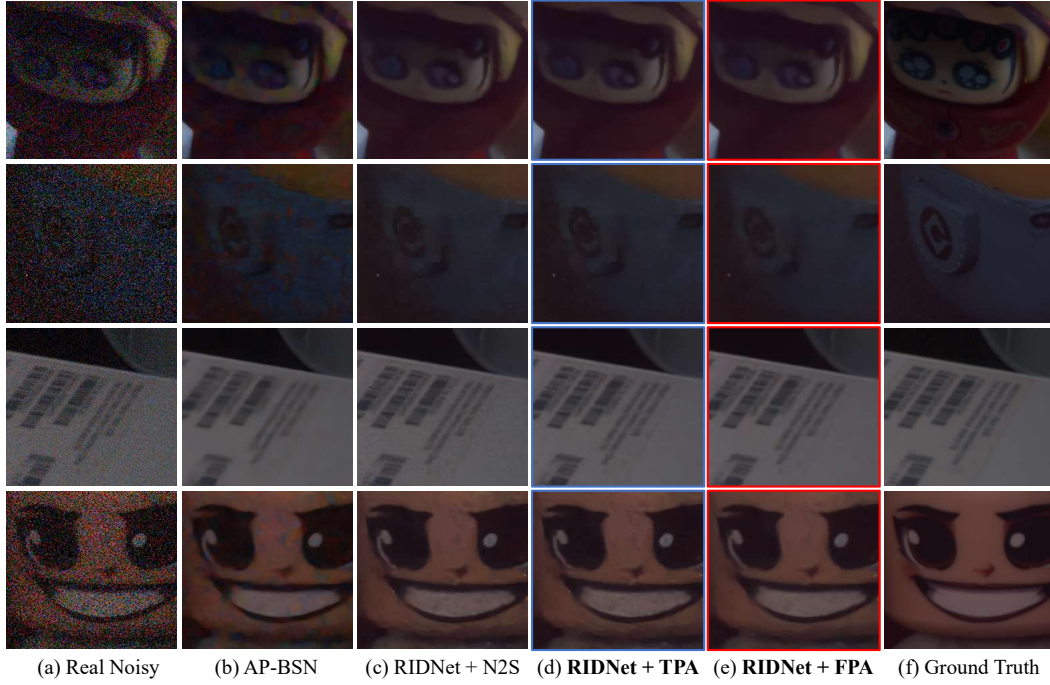


Figure S3: Qualitative results in RIDNet on SIDD+ datasets.

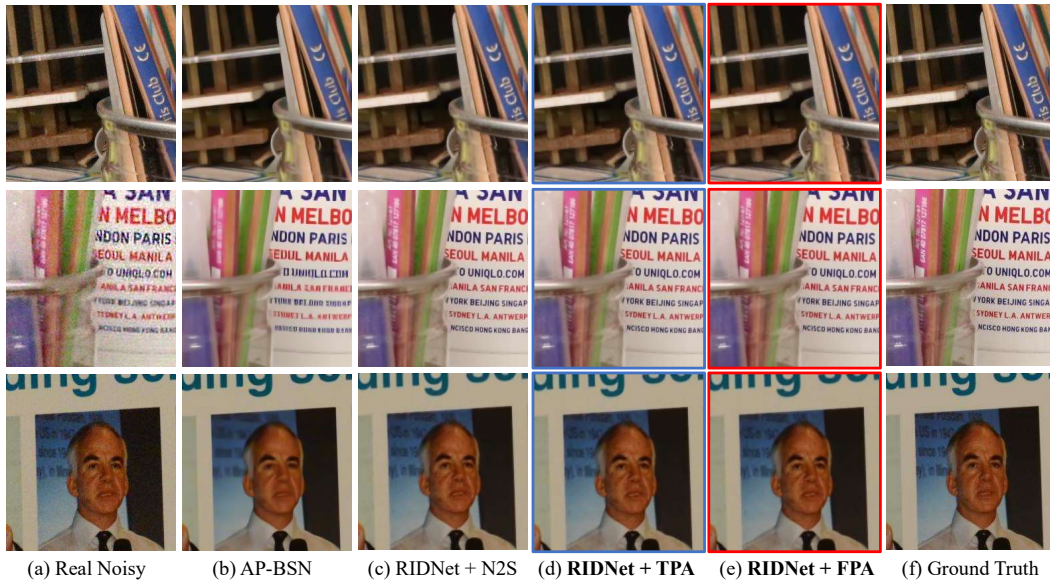


Figure S4: Qualitative results in RIDNet on PolyU datasets.

REFERENCES

- Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- 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.
- Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.
- Meiguang Jin, Michael Hirsch, and Paolo Favaro. Learning face deblurring fast and wide. 2018.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. SrfLOW: Learning the super-resolution space with normalizing flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2018.