Appendix:
Robust Image Denoising through Adversarial Frequency Mixup

Donghun Ryou[2] Inju Ha[1] Hyewon Yoo[2] Dongwan Kim[1] Bohyung Han[1,2]
Computer Vision Laboratory, [1]ECE & [2]IPAI, Seoul National University
{dhryou, hij1112, yoohyewony, dongwan123, bhhan}@snu.ac.kr

A. Mixup Design Choice

Unlike the conventional mixup [12], which typically operates on two distinct images, we design our Adversarial Frequency Mixup to mix the noisy image I with the denoised image $\hat{I}$. This is because using I and $\hat{I}$ in the frequency mixup can be interpreted as creating a new image with a transformed noise component (Eq. (5) of the main paper).

Let us consider the case where we follow conventional mixup to employ two distinct images from the dataset, I_1 and I_2, instead of I and $\hat{I}$. Following Eq. (5), the frequency mixup on I_1 and I_2 can be summarized as:

$$\begin{aligned}\tilde{I} &= \mathcal{F}^{-1}\left(\mathcal{F}(I_1)\odot\boldsymbol{m}+\mathcal{F}(I_2)\odot(\mathbf{1}-\boldsymbol{m})\right),\\ &= \mathcal{F}^{-1}\left(\mathcal{F}(I_2)+\boldsymbol{m}\odot(\mathcal{F}(I_1)-\mathcal{F}(I_2))\right)\\ &= I_2+\mathcal{F}^{-1}\left(\boldsymbol{m}\odot\mathcal{F}(I_1-I_2)\right)\\ &= I_2+\mathcal{F}^{-1}(\boldsymbol{m})*(I_1-I_2)\\ &= I_2+\boldsymbol{h}*(I_1-I_2)\\ &= I_2+\boldsymbol{h}*\Delta I_{1,2},\end{aligned} \tag{S.1}$$

where $\Delta I_{1,2}$ denotes the difference between I_1 and I_2. As shown above, when we use I_1 and I_2, frequency mixup can is interpreted as a transformation on $\Delta I_{1,2}$. However, this does not align with our original motivation, which was: to generate a new image with some noise that is characterized by a distribution distinct from the original noisy image, while also resembling noise encountered in real-world scenarios. In contrast, Eq. (5) aligns well with our motivation, and thus, we employ I and $\hat{I}$ for frequency mixup.

B. Architecture details

B.1. Denoising network architecture

In this section, we elucidate the details of the denoising architectures chosen as the base in our experiments. Beginning with the architecture DnCNN [13], we initially remove all biases following Mohan *et al*. [6]. Additionally, in our experiments, we observe improved performance when batch normalization is removed in DnCNN. Therefore, we utilize DnCNN without bias and batch normalization as our baseline architecture.

We also reproduce NAFNet [3] and MPRNet [11] both in our training setting with and without AFM. To minimize training costs in NAFNet, we decrease the channel width and the number of blocks to 32 and 18, respectively. In MPRNet, we reduce the number of channels to 20, and as the layers deepened, we scale the increase in channels to one-fourth of the original configuration.

Table S.1 shows the evaluation results of original off-the-shelf networks and the networks we reproduced. We observe that the denoising performance in OOD scenarios does not significantly change even with reduced network sizes, and interestingly, in the case of NAFNet, an improvement is actually observed.

B.2. Mask generation module

AFM-E utilizes pixel segmentation to create a mask that matches the dimensions of the input image. The input to the model comprises a channel-wise concatenation of four components: I, $\hat{I}$, $|\mathcal{F}(I)|$, and $|\mathcal{F}(\hat{I})|$. This concatenated input is then processed through a UNet-based encoder-decoder architecture, which features both upsampling and downsampling operations,

Table S.1. Quantitative comparisons on the SIDD validation set [1] and the other real noise datasets. Networks marked with * indicate results from off-the-shelf networks, while the others represent our reproduced results with reduced size and according to our training schedule with and without AFM-B. We present performance in terms of PSNR↑ (dB) and SSIM↑.

		In-distribution	Out-of-distribution							
Architecture	Metric	SIDD [1]	Poly [8]	CC [7]	HighISO [9]	iPhone [5]	Huawei [5]	OOD Avg	Params (M)	MACs (G)
MPRNet* [11]	PSNR	39.71	37.50	35.94	38.04	40.21	38.35	38.20	15.74	588.14
	SSIM	0.9586	0.9769	0.9767	0.9732	0.9743	0.9672	0.9737		
MPRNet†	PSNR	39.55	37.54	35.96	38.01	40.41	38.17	38.02	0.99	39.89
	SSIM	0.9572	0.9795	0.9792	0.9753	**0.9771**	0.9683	0.9759		
MPRNet-Ours	PSNR	39.41	**37.92**	**36.56**	**39.11**	**40.62**	**38.60**	**38.56**	0.99	39.89
	SSIM	0.9568	**0.9807**	**0.9806**	**0.9788**	0.9765	**0.9689**	**0.9771**		
NAFNet* [3]	PSNR	40.30	36.08	34.40	37.92	36.60	36.18	36.23	115.7	63.3
	SSIM	0.9614	0.9618	0.9786	0.9772	0.8903	0.9389	0.9494		
NAFNet†	PSNR	39.84	37.10	35.66	38.10	37.75	37.65	37.27	7.5	8.8
	SSIM	0.9592	0.9788	0.9807	**0.9774**	0.9111	0.9679	0.9631		
NAFNet-Ours	PSNR	39.81	**37.70**	**36.56**	**38.34**	**40.10**	**38.64**	**38.27**	7.5	8.8
	SSIM	0.9591	**0.9792**	**0.9823**	0.9760	**0.9723**	**0.9684**	**0.9756**		

Table S.2. Effect of varying hyperparameters λ and γ in terms of PSNR↑ (dB) while γ and λ are fixed to 0.3 and 0.8, respectively. The results are OOD average denoising performances of DnCNN [13] trained with AFM-B.

λ ($\gamma = 0.3$)	0.2	0.5	0.8	1.0
OOD Avg.	38.13	38.44	**38.56**	38.33
γ ($\lambda = 0.8$)	0.0	0.2	0.3	0.4
OOD Avg.	38.30	38.40	**38.56**	38.04

interconnected by skip connections. Each encoder and decoder block in this architecture consists of two consecutive blocks of: a convolutional layer, followed by batch normalization and a ReLU activation layer. In the first encoder block, the number of channels is augmented from an initial count of 12 to 32. Subsequently, this quantity is doubled in each successive encoder layer. The escalation continues until the channel count reaches a maximum of 256. Thereafter, in the decoding phase, the number of channels is halved in every decoder layer, progressively reducing until it reverts to the original count of 32 channels. This network culminates in a depth-wise convolutional layer, which is then succeeded by a sigmoid activation function to constrain the output values within the range of 0 to 1.

The AFM-B architecture starts with of three consecutive blocks of: convolutional layer, ReLU activation function and a pooling layer. This structure culminates in two fully-connected layers. After the final fully connected layer, we extract N values, each as if it were a classification task. For instance, to obtain v_1, we apply softmax to 100 output values of final fc layer and implement weighted sum with values from 0 to 1, devided into equal 100 parts. This same method is applied to each v_n. This approach is chosen to acquire values more stably than regression. These values are used to construct a mask comprising N circular bands, which are uniformly distributed along the polar axis. Within the scope of each band, every element corresponds to a mixup value generated by the network. Consequently, the aggregation of these N circular bands forms a comprehensive Mask, aligning with the dimensions of the input image. In our experiments, we set the N by 100.

C. Additional results

C.1. hyperparameters

We evaluate the OOD average denoising performances by varing the hyperparameters λ and γ as presented in Eq. (8) and (9), and the results are reported in Table S.2.

C.2. Generalization on Synthetic Noise

We evaluate the denoising performance of DnCNN [13] trained on SIDD Medium [1] dataset with and without proposed AFM-B in the presence of synthetic noise, specifically Gaussian and Poisson noise. To achieve this, Gaussian or Poisson noise was artificially introduced to the ground truth images from the SIDD validation dataset. As shown in Table S.3, the results obtained from the DnCNN model trained using the AFM-B approach demonstrates a notable improvement in

Table S.3. Quantitative comparisons on the Gaussian and Poisson noise of noise level $\sigma \in [0, 55]$ with DnCNN [13]. We generated synthetic noisy images by adding Gaussian and Poisson noise to ground truth images from the SIDD validation set [1], in order to evaluate the denoising performance. We present performance in terms of PSNR↑ (dB) and SSIM↑.

	Gaussian		Poisson	
	PSNR	SSIM	PSNR	SSIM
Normal Training	27.85	0.6959	27.47	0.6711
AFM-B (Ours)	**31.80**	**0.8675**	**31.86**	**0.8524**

Table S.4. Comparing the out-of-distribution performance of AFM-E and AFM-B with other conventional generalization methods on the DnCNN architecture. We present performance in terms of PSNR↑ (dB) and SSIM↑. The bolded and underlined values represent the best and second best values in each column, respectively. This Table is an extended version of Table 3 in the main paper.

		In-distribution	Out-of-distribution					
Algorithm	Metric	SIDD [1]	Poly [8]	CC [7]	HighISO [9]	iPhone [5]	Huawei [5]	OOD Avg.
Normal Training	PSNR	38.62	37.36	35.69	37.85	39.87	38.26	37.81
	SSIM	0.9501	0.9740	0.9755	0.9703	0.9688	0.9654	0.9708
Dropout [4]	PSNR	37.27	37.20	35.17	37.46	39.88	38.13	37.57
	SSIM	0.9294	0.9730	0.9722	0.9693	0.9696	0.9643	0.9697
Input mask [2]	PSNR	37.83	36.67	34.54	37.75	39.67	37.75	37.28
	SSIM	0.9441	0.9759	0.9774	0.9755	0.9738	0.9631	0.9731
CutMix [10]	PSNR	38.59	37.42	36.23	38.00	39.75	38.39	37.96
	SSIM	0.9500	0.9777	0.9771	0.9712	0.9684	0.9655	0.9719
Adversairal Training	PSNR	38.43	37.05	34.87	36.70	39.26	38.00	37.18
	SSIM	0.9483	0.9732	0.9697	0.9635	0.9638	0.9639	0.9668
Random Freq. Mixup	PSNR	38.53	37.53	36.39	38.41	40.02	**38.49**	38.17
	SSIM	0.9493	0.9765	0.9790	0.9744	0.9705	0.9665	0.9734
ASM-E	PSNR	38.46	37.31	36.13	38.26	39.65	38.23	37.92
	SSIM	0.9490	0.9753	0.9772	0.9739	0.9675	0.9644	0.9717
AFM-E (Ours)	PSNR	38.41	<u>37.73</u>	<u>36.78</u>	**39.18**	<u>40.39</u>	<u>38.46</u>	<u>38.51</u>
	SSIM	0.9485	<u>0.9802</u>	**0.9832**	<u>0.9800</u>	<u>0.9756</u>	<u>0.9682</u>	<u>0.9774</u>
AFM-B (Ours)	PSNR	38.35	**37.75**	**36.84**	<u>39.17</u>	**40.65**	38.39	**38.56**
	SSIM	0.9478	**0.9804**	<u>0.9830</u>	**0.9801**	**0.9777**	**0.9683**	**0.9779**

denoising performance when compared to the model trained via conventional supervised method. Specifically, there is an increase of 3.95 dB and 4.39 dB in the PSNR for Gaussian and Poisson noise, respectively. Consequently, it shows that our AFM training methodology effectively mitigates the issue of overfitting to specific noise distributions.

C.3. Conventional Generalization Methods

Table S.4 is the extended version of Table 3 in the main paper, which includes the PSNR and SSIM scores for each individual OOD dataset.

C.4. Visualization results

Figure S.1 effectively visualizes the functionality of our AFM. The generated I_{hard} cases are challenging for the denoising network trained with conventional supervised methods, while the generated I_{easy} show ease in denoising. This indicates that our AFM module successfully creates adversarial noisy images as intended. Figure S.2 shows the additional qualitative results of our AFM-B.

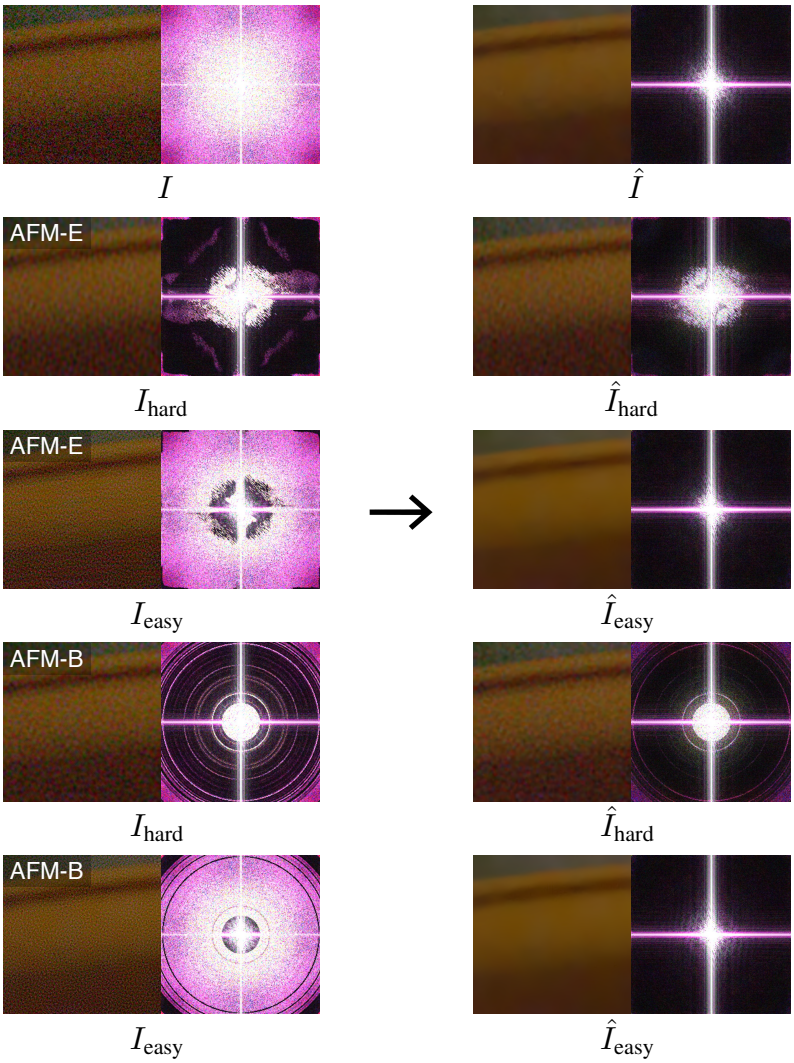

Figure S.1. Visualization of images generated within the AFM training framework. This includes hard and easy images, their denoised outputs, and the corresponding frequency magnitudes of each image. Images in the second and third rows are generated by AFM-E. Images in the fourth and fifth rows are generated by AFM-B. Each denoised image is generated using DnCNN, trained with a standard supervised method on the SIDD dataset. In both AFM-B and AFM-E, it can be observed that denoising is less effective in hard cases, while it is more successful in easy cases.

D. Limitations

Our experiments employing DnCNN with AFM, across various random seeds, demonstrated a uniform average enhancement in performance, evidenced by an increment with a standard deviation of 0.11 dB in PSNR and 0.0010 in SSIM. However, while the average performance enhancements remained consistent, we observed discrepancies in variance across different datasets. Our model generally enhances performance; however, ensuring consistency across all unseen Out-Of-Distribution (OOD) datasets presents a challenge.

Furthermore, while AFM does not affect inference cost, it inevitably leads to increase in computational cost during the training phase, as a compromise for enhancing the model's robustness. The DnCNN training times with and without AFM-B are 0.22s and 0.92s per iteration with NVIDIA RTX A5000. The potential of minimizing training time while enhancing a model's robustness remains an area for future exploration.

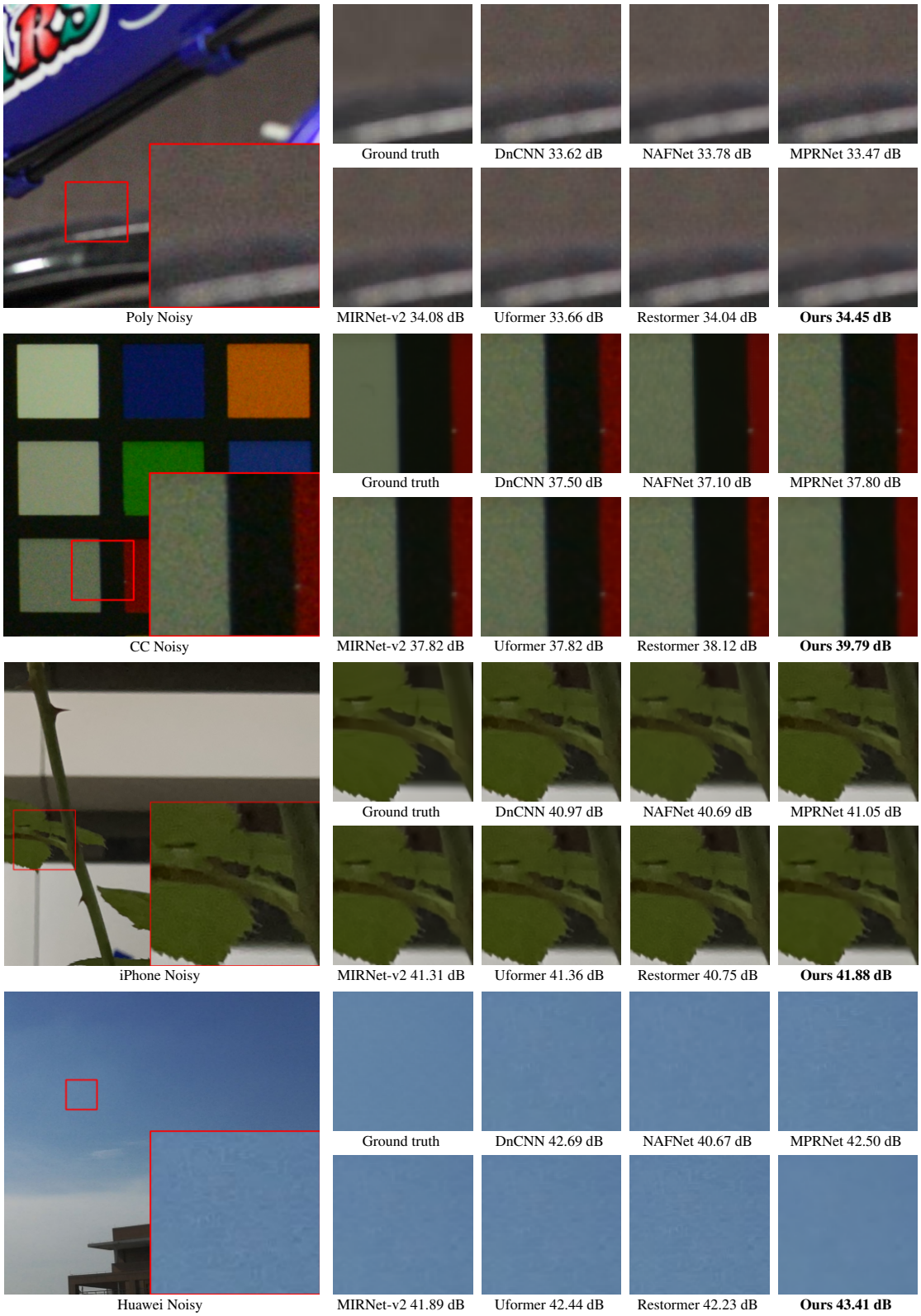

Figure S.2. Comparing the denoised outputs of various denoising networks including ours (DnCNN trained with AFM-B), on out-of-distribution (OOD) datasets. DnCNN with AFM-B displays cleaner outputs compared to other networks that are not trained with AFM.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, pages 1692–1700, 2018. 2, 3

[2] Haoyu Chen, Jinjin Gu, Yihao Liu, Salma Abdel Magid, Chao Dong, Qiong Wang, Hanspeter Pfister, and Lei Zhu. Masked image training for generalizable deep image denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1692–1703, 2023. 3

[3] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *ECCV*, pages 17–33, 2022. 1, 2

[4] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6002–6012, 2022. 3

[5] Zhaoming Kong, Fangxi Deng, Haomin Zhuang, Jun Yu, Lifang He, and Xiaowei Yang. A comparison of image denoising methods, 2023. 2, 3

[6] Sreyas Mohan, Zahra Kadkhodaie, Eero P. Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *ICLR*, 2020. 1

[7] Seonghyeon Nam, Youngbae Hwang, Yasuyuki Matsushita, and Seon Joo Kim. A holistic approach to cross-channel image noise modeling and its application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1683–1691, 2016. 2, 3

[8] Jun Xu, Hui Li, Zhetong Liang, David Zhang, and Lei Zhang. Real-world noisy image denoising: A new benchmark. *arXiv preprint arXiv:1804.02603*, 2018. 2, 3

[9] Huanjing Yue, Jianjun Liu, Jingyu Yang, Truong Q Nguyen, and Feng Wu. High iso jpeg image denoising by deep fusion of collaborative and convolutional filtering. *IEEE Transactions on Image Processing*, 28(9):4339–4353, 2019. 2, 3

[10] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3

[11] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *CVPR*, pages 14821–14831, 2021. 1, 2

[12] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 1

[13] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 1, 2, 3