Supplementary Materials: GradNet Image Denoising

Yang Liu[1,2], Saeed Anwar[1,2], Liang Zheng[1], Qi Tian[3]
Australian National University[1], Data61-CSIRO[2], Huawei Noah's Ark Lab[3]
yang.liu3@anu.edu.au, saeed.anwar@csiro.au, liang.zheng@anu.edu.au, tianqi1@huawei.com

1. Training Details of Concept Proof

These four networks are trained on the NYUv2 dataset [2] with AWGN and the real noise SIDD dataset for 50 epochs. The learning rate is initialized as 10^{-3} and decays to 2×10^{-4} after 30 epochs. For all the three datasets, a crop size of 80×80 is employed during training.

2. Fusion of approximate clean gradient

As discussed in Section 4.2, fusing the approximate clean gradient is different from fusing the clean one. One might doubt whether fusing in the early layer is also the most effective way when it comes to the approximate gradient. In this part, we implement an experiment similar to the one in Section 3.2 with approximate gradient fused.

We use DnCNN as the naive denoise method. SKDnCNN is employed as the main denoise block, the same as the experiment in Section 3.2. We also train 50 epochs with 10^{-3} as the initialized learning rate which decays to 2×10^{-4} after 30 epochs. A crop size of 80×80 is applied. We show the results in Fig. 1.

Although the improvements on these three datasets by GradNet-II are not as high as those with clean image gradient fused (shown in Fig, 3 in the paper), GradNet-II still boosts +0.33dB, +0.47dB on NYUv2 with σ=15 and 50 and +1.11dB on SIDD. Comparing with the other three architectures (GradNet-I, GradNet-III, GradNet-IV), GradNet-II is still the best. The reported results validate that **fusing in the shallow layer is also the most effective way to exploit the approximate clean gradient**.

3. Sobel Filters vs. Grad Filters

One might also question that the pre-defined grad filters are fixed filters, which can be learned in convolutional networks. We agree that the network can learn filters like Sobel Filter automatically under ideal circumstances. However, our experiment in proof-of-concept suggests that it's not so obvious. Adding the image gradient extracted by a fixed filter (we use Sobel Filter as an example here) can be treated as adding a prior to the network, which can boost the denoising performance. If a network can learn Sobel Filter or

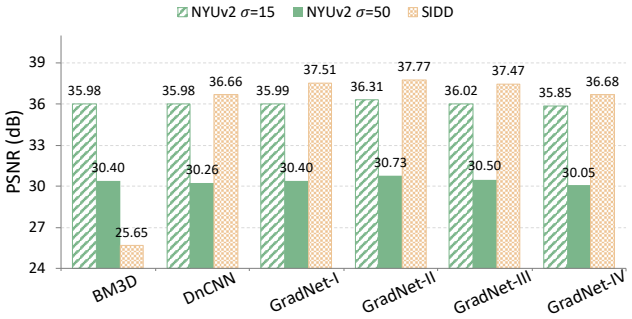

Figure 1: Fusing approximate gradient to different layers the same way as Fig. 2 shows in the paper. The approximate clean image is obtained by DnCNN. The results verify that GradNet-II is also the most effective architecture when it comes to approximate gradient.

other filters which are better than Sobel Filter automatically, the performance of adding Sobel Filter explicitly will be not higher than the original network in the optimization process. During training, the gradient-based update of parameters of our network is easy to fall in local optimal instead of finding the global optimal. Adding a fixed filter layer like Sobel Filter can help to jump out of local optimality to some extent.

4. Mathematical Expressions of the Details of GradNet

Image gradient fusion. Suppose $\boldsymbol{y} \in \mathbb{R}^{H \times W}$ is the contaminated input image. Let $\hat{\boldsymbol{x}}$ to be the naive denoised image. Let $\psi(\cdot)$ denotes the function that the naive denoising perform, then the edge and texture extraction operation can be expressed as $G(\hat{\boldsymbol{x}}) = G(\psi(\boldsymbol{y}))$.

The Main branch. As illustrated in Fig. 4 in the paper, the main branch of GradNet includes feature extraction, denoise and reconstruction blocks.

Feature extraction block. This feature extraction layer is composed of a convolutional layer $c(\cdot)$ which is followed by a ReLU layer $\tau(\cdot)$, denoted as $\boldsymbol{f_0} = \tau(c(\boldsymbol{y}))$. Then we concatenate the gradient of the naive denoised image $G(\hat{\boldsymbol{x}})$ with the extracted features as the input of the main denois-

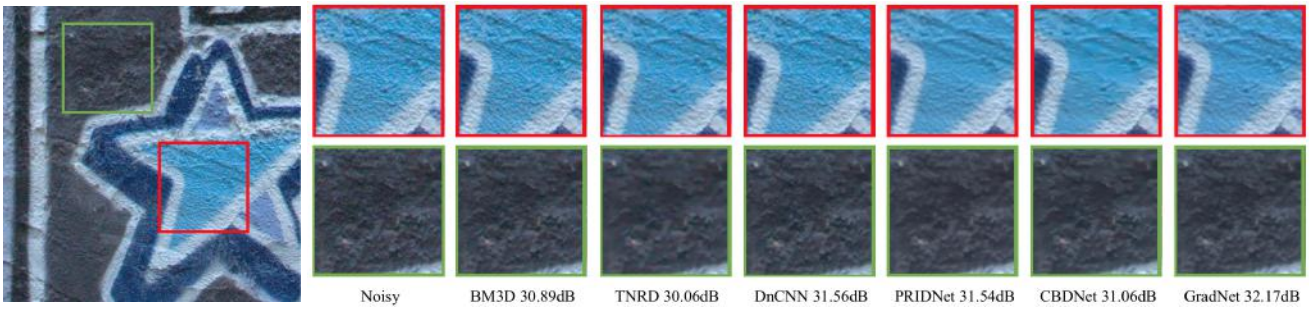

Figure 2: An example from the DnD dataset [3]. Zoom-in is needed to check the details. GradNet reserves fine scale details while removing noise, without generating artifacts.

(a) Noisy 20.11dB (b) DnCNN 28.75dB (c) FFDNet 21.68dB (d) CBDNet 28.44dB (e) GradNet 34.10dB (f) Clean

Figure 3: An example from the SIDD validation dataset [1]. FFDNet fails to remove all the noise. CBDNet leaves some artifacts while removing noise. DnCNN changes the color of the image while denoising. Both quantitative and qualitative results of GradNet outperform other methods to a great extent.

ing block $\boldsymbol{g_0} = \boldsymbol{f_0} \oplus G(\hat{\boldsymbol{x}})$, where $\oplus$ is the concatenation operator.

MSKResnet as the denoise block. As shown in Fig. 5, MSKResnet cascades several residual modules with a long skip connection. Each of the residual modules contains several ResUnits with short skip connection, a medium skip connection, and an attention module.

The particular operation in **ResUnit**, denoted by $R(\cdot)$, can be expressed as

$$\boldsymbol{f_{i+1}^k} = R(\boldsymbol{f_i^k}) = \tau(c(\tau(c(\boldsymbol{f_i^k}))) + \boldsymbol{f_i^k}), \tag{1}$$

where $\boldsymbol{f_i^k}$ and $\boldsymbol{f_{i+1}^k}$ are the input and output of the (i+1)-th ResUnit in the k-th residual module.

Suppose there are m ResUnits in each residual module and the input to the k-th **residual module** is $\boldsymbol{f_0^k}$, the output $\boldsymbol{f_m^k}$ after a sequence of operations of ResUnits is

$$\boldsymbol{f_m^k} = \underbrace{R_m(\dots R_2(R_1(\boldsymbol{f_0^k})))}_{\text{m}}. \tag{2}$$

We use an **attention module** after the concatenation of $\boldsymbol{f_0^k}$ and $\boldsymbol{f_m^k}$, *i.e.*, $\boldsymbol{f_0^k} \oplus \boldsymbol{f_m^k}$, in the k-th residual module. The attention module we use consists of an average pooling layer $\phi(\cdot)$, a shrinkage convolutional layer $c_D(\cdot)$ followed by ReLU $\tau(\cdot)$, and a reconstruction convolutional layer $c_U(\cdot)$ with Sigmoid $\sigma(\cdot)$ as the activate function. Suppose $\boldsymbol{f_0^k} \oplus \boldsymbol{f_m^k} \in \mathbb{R}^{C\times H\times W}$, where C is the number of channels and $H \times W$ is the size of each feature map. The size of the feature maps is reduced from $C \times H \times W$ to $C \times 1 \times 1$ after average pooling $\phi(\cdot)$.

$$\phi(\boldsymbol{f_0^k} \oplus \boldsymbol{f_m^k}) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \boldsymbol{f_0^k} \oplus \boldsymbol{f_m^k}, \tag{3}$$

$$\boldsymbol{f_a^k} = \sigma(c_U(\tau(c_D(\phi(\boldsymbol{f_0^k} \oplus \boldsymbol{f_m^k}))))), \tag{4}$$

where $\boldsymbol{f_a^k}$ is the output of the attention module, which is also the output of the k-th residual module. It is passed to the (k+1)-th residual module as input $\boldsymbol{f_0^{k+1}}$.

Assume N residual modules are contained in our MSKResnet, the whole process of this structure can be represented as

$$\boldsymbol{f_a^N} = \underbrace{\mathcal{M}_N(\dots \mathcal{M}_2(\mathcal{M}_1(\boldsymbol{g_0})))}_{\text{N}}, \tag{5}$$

where $\boldsymbol{g_0}$ is the features extracted by feature extraction layer, $\mathcal{M}(\cdot)$ is the residual module and $\boldsymbol{f_a^N}$ is the features from the final residual module.

The reconstruction layer. At last, the output of the main denoising block $\boldsymbol{f_a^N}$ is passed to a reconstruction layer. The final denoised image $\widetilde{\boldsymbol{x}} = c(\boldsymbol{f_a^N}) + \boldsymbol{y}$.

5. Visual Results

Two visual examples from DnD and SIDD are represented in Fig. 2 and Fig. 3.

References

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

[2] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1

[3] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *CVPR Workshops*, 2017. 2