Integrating Disparate Sources of Experts for Robust Image Denoising

Joon Hee Choi  Omar Elgendy  Stanley H. Chan
Purdue University, West Lafayette, IN 47907, USA
{choi240, oelgendy, stanchan}@purdue.edu

Abstract

We study an image denoising problem: Given a set of image denoisers, each having a different denoising capability, can we design a framework that allows us to integrate the individual denoisers to produce an overall better result? If we can do so, then potentially we can integrate multiple weak denoisers to denoise complex scenes. The goal of this paper is to present a meta-procedure called the Consensus Neural Network (ConsensusNet). Given a set of initial denoisers, ConsensusNet takes the initial estimates and generates a linear combination of the results. The combined estimate is then fed to a booster neural network to reduce the amount of method noise. ConsensusNet is a modular framework that allows any image denoiser to be used in the initial stage. Experimental results show that ConsensusNet can consistently improve denoising performance for both deterministic denoisers and neural network denoisers.

1. Introduction

1.1. Motivation

While image denoising algorithms over the past decade have produced very promising results, it is also safe to say that thus far no single image denoiser can perform uniformly better than other denoisers. In fact, any image denoiser, either deterministic or learning-based, has an implicit prior model that determines its denoising characteristics. Since a particular prior model encapsulates the statistics of a limited set of imaging conditions, the corresponding denoiser is only an expert for the images that it is designed to handle. We refer to this gap between the imaging model and the denoising task as a model mismatch.

Model Mismatch is common in practice. For example, in Figure 1 we show two standard testing images denoised by BM3D [7] and one of the latest neural network denoisers DnCNN [22]. For Boat512, it is clear that DnCNN performs better. However, for Barbara512, BM3D actually outperforms DnCNN due to the weak oscillating pattern on the cloth, a rare feature that is difficult to learn.

Another example is the mismatch in noise levels for neural network image denoisers. Suppose that a denoiser is trained for i.i.d. Gaussian noise of standard deviation $\sigma = 20$. This denoiser will work well for noisy images with noise level exactly at $\sigma = 20$. But as soon as the noise level deviates from $\sigma = 20$, the performance will start to degrade. See Section 2.1 for an example.

Class-specific denoiser is also common: A denoiser could be well suited for a particular class of images (e.g., text images), but it may not work for other classes. Intuitively, one can argue that if high level vision tools are available, e.g., scene classification, then the problem could be solved. However, scene classification itself is an open problem and there is no unified opinion on how to achieve it. If possible, it would be more convenient if the denoiser can automatically pick a class that gives the best performance without seeking classification algorithms.

In all these examples, we can ask a question: If we have a set of denoisers each having a different characteristic, how do we integrate them to produce a better result? This problem is non-trivial because the denoisers could be trained at different noise levels, image classes, or built from different
1.2. Contributions

There are two innovations of ConsensusNet. First, ConsensusNet comprises a weighting stage that weights the relevance of the individual denoisers. The weighting mechanism uses a deterministic approach of the Stein’s Unbiased Risk Estimator (SURE) [19] to estimate the mean squared error (MSE). We also propose a new method to translate the estimated MSEs to the combination weights.

Second, ConsensusNet comprises a boosting neural network (or booster in short). The booster is essential as the weighted average of the initial estimates is typically over-smoothed. We use the booster to recover the lost features and to improve contrast. Our results show that the booster has significant impact to the denoising performance.

We will present applications of ConsensusNet on various scenarios. These examples include the integration of denoisers with (1) different noise levels; (2) different image classes; and (3) different denoiser types.

1.3. Related Work

The context of ConsensusNet is image denoising, which includes deterministic denoisers such as BM3D [7], non-local means [4], total variation [2], EPLL [25] and WNNM [10], and neural network denoisers such as MLP [5], SSDA [21], REDNet [14], DenoiseNet [18], DnCNN [22], NLNet [12] and FFDNet [24].

Methods seeking linear combination of image denoisers for better performance have been discussed but are scattered in the literature. In [6], the authors presented a SURE-based method to linearly combine two bilateral filters. We argue that ConsensusNet is significantly more general as it is applicable to any denoiser. In [1], the authors proposed to learn the weights using an auto-encoder. The difference with ConsensusNet is that we use a deterministic method to estimate the weights, and we have a booster network which [1] does not have.

The noise-level mismatch is discussed more often in the neural network literature. Conventional approach is to either truncate the noise level to the nearest trained level [23] or to train the network with a large number of examples covering all noise levels [22]. A more recent approach is to feed a noise map to the network and train the network to recognize the noise level [24]. However, this approach requires a redesign of the network structure. In contrast, ConsensusNet uses the same structure for all initial denoisers.

2. Consensus Neural Network

2.1. Model Mismatch

As we discussed in the introduction, model mismatch appears when the denoiser model fails to correspond to the actual imaging model. The followings are the cases we will address in this paper:

- Noise-level Mismatch: Happens when a denoiser assumes a noise level \( \hat{\sigma} \) that is different from the actual noise level \( \sigma \). For example, the denoiser could be trained at \( \hat{\sigma} \) but is used for actual noise level \( \sigma \).
- Image-class Mismatch: The network is trained using generic databases, but is used to denoise images with specific content such as face or building.
- Denoiser-model Mismatch: Every image denoiser has an underlying model characterizing its denoising behavior. However, there are always images that favor one denoiser but not another. Figure 1 is a typical example.

To illustrate further the behavior of the model mismatch, we consider a noise-level mismatch example using DnCNN [22] and BM3D [7], with results shown in Figure 2. In both cases, we use 10 standard Kodak images added with i.i.d. Gaussian noise of noise levels \( \sigma \in \{10, 20, 30, 40, 50\} \) in terms of true noise levels \( \{10, 50\} \) on 10 standard Kodak images.

![Figure 2: Illustration of noise-level mismatch.](image)

Figure 2: Illustration of noise-level mismatch. We compare BM3Ds and DnCNNS at noise levels \( \hat{\sigma} \in \{10, 20, 30, 40, 50\} \) in terms of true noise levels \( \{10, 50\} \) on 10 standard Kodak images. To illustrate further the behavior of the model mismatch, we consider a noise-level mismatch example using DnCNN [22] and BM3D [7], with results shown in Figure 2. In both cases, we use 10 standard Kodak images added with i.i.d. Gaussian noise of noise levels \( \sigma \) in the interval of \( \{10, 50\} \). Then, we train five DnCNNS at noise levels \( \hat{\sigma} \in \{10, 20, 30, 40, 50\} \). For BM3D, we fix the denoising parameter as \( \hat{\sigma} \in \{10, 20, 30, 40, 50\} \) so that the denoiser only has five denoising strengths.

There are two observations we can see from Figure 2. First, when \( \sigma \) matches with \( \hat{\sigma} \) exactly, DnCNN performs better than BM3D as it is trained to handle that noise level. However, when \( \hat{\sigma} \) deviates from \( \sigma \), BM3D demonstrates a more robust behavior. Second, at any \( \sigma \), the best performing
denoiser among the five is the one with \( \hat{\sigma} = \sigma \). However, if \( \sigma \) is a value between two consecutive \( \hat{\sigma} \)'s, we need a method to interpolate the result.

### 2.2. Overview of ConsensusNet

ConsensusNet is a general framework for integrating a disparate sources of denoising experts. The architecture of ConsensusNet is shown in Figure 3. The framework consists of a set of \( K \) pre-trained denoisers \( D_1, \ldots, D_K \). These denoisers can be both deterministic or learning-based. We assume that these denoisers are pre-defined and will not be altered during the denoising process.

![Figure 3: Structure of ConsensusNet.](image)

When a noisy image \( y \in \mathbb{R}^n \) is fed to ConsensusNet, it is first denoised individually by the \( K \) initial denoisers. This generates \( K \) estimates \( \hat{z}_1, \ldots, \hat{z}_K \):

\[
\hat{z}_k = D_k(y), \quad k = 1, \ldots, K.
\]

Once \( \hat{z}_1, \ldots, \hat{z}_K \) become available, we compute a weighted average of the results, yielding

\[
\hat{z} = \sum_{k=1}^{K} w_k \hat{z}_k, \quad (2)
\]

where the weights \( \{w_k\}_{k=1}^{K} \) sum to unity: \( \sum_{k=1}^{K} w_k = 1 \). We will discuss the weighing mechanism in Section 2.3.

The second stage of ConsensusNet is a booster neural network that takes \( \hat{z} \) and generates the final denoised output \( \hat{z}_{\text{final}} \). Denoting the booster neural network as \( F \), we have that

\[
\hat{z}_{\text{final}} = F(\hat{z}). \quad (3)
\]

Note that the booster network has to be trained for every collection of initial denoisers. We argue that this is reasonable in practice because the user usually has a set of candidate denoisers in mind. Once the candidate denoisers are chosen, we can then train a booster network. We will discuss the details of the booster network in Section 2.4.

### 2.3. Determining the Weights

#### A. Why Combine \( \hat{z}_k \)?

Before we address the problem of how to determine the weights \( w_k \), we must first discuss why we want to do so. That is, why should we send the linear combination \( \hat{z} \) in (2) to the booster network \( F \), and not the 3D-stack of \( \{\hat{z}_1, \ldots, \hat{z}_K\} \) to the booster network \( F \) directly? This problem is important because ideally we would want the neural network to learn the weights automatically. However, there are several difficulties.

First, while the initially denoised results \( \hat{z}_1, \ldots, \hat{z}_K \) do form a stack of images, these images are un-ordered. For example, if the input noisy image contains heavy noise, then a denoiser \( D_k \) with stronger denoising strength will produce a better result. In this case, the corresponding weight has to be large. However, the same weighting configuration cannot be used for low noise images because low noise images prefer weak denoisers. As a result, there is no single ordering of the weights that can support all noise levels.

Second, if we form a linear combination \( \hat{z} \), the booster \( F \) can be trained more easily. In contrast, if we ask \( F \) to recognize the \( K \) initially denoised results \( \hat{z}_1, \ldots, \hat{z}_K \), we need a significantly more complex network and significantly more amount of data to train the booster.

#### B. Oracle Case

Ideally, a weight \( w_k \) should be large if the corresponding denoiser is good. Thus, we first need a metric to measure the quality of the denoised image. Since most methods use PSNR as the quality metric for the final output, we consider the MSE difference between an estimate \( \hat{z}_k \) and the mean squared error (MSE) in designing the weights. The MSE between an estimate \( \hat{z}_k \) and the ground truth is:

\[
\text{MSE}_k = \frac{1}{N} \| \hat{z}_k - z \|^2 \quad (4)
\]

where \( N \) is the number of pixels of \( \hat{z}_k \).

The weight \( w_k \) in our method consists of two components. The first component measures the similarity between \( \hat{z}_k \) and the best performing denoiser:

\[
w_k = \exp \left\{ \frac{\text{MSE}_k - \min_k \{\text{MSE}_k\}}{c_r \sigma_r} \right\}, \quad (5)
\]

where \( \sigma_r = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (\epsilon_k - \bar{\epsilon})^2} \) is the standard deviation of the MSE difference \( \epsilon_k \) and \( \bar{\epsilon} = \frac{1}{K} \sum_{k=1}^{K} \epsilon_k \) is the average of the \( \epsilon_k \)'s. The constant \( c_r \) determines the strength of weight. Typically, we have \( 0.1 < c_r < 0.5 \).

The second component of the weight measures the difference between the noise levels. Given the actual noise standard deviation \( \sigma \) and the denoising strengths \( \hat{\sigma}_k \) for each denoiser, we define

\[
w_k = \exp \left\{ \frac{|\hat{\sigma}_k - \sigma|}{c_s \sigma_s} \right\}, \quad (6)
\]

where \( \sigma_s = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (s_k - \bar{s})^2} \) is the standard deviation of the noise differences \( s_k = \hat{\sigma}_k - \sigma \), and \( \bar{s} = \frac{1}{K} \sum_{k=1}^{K} s_k \).
is the average. The constant $c_w$ determines the strength of weight, with a typical value of $0.1 < c_w < 0.5$.

The overall weight $w_k$ is defined as the product of $w_k^h$ and $w_k^r$, with a normalization so that they sum to one:

$$w_k = w_k^h w_k^r / \sum_{k=1}^{K} w_k^h w_k^r. \quad (7)$$

The weight defined in (7) can be used for both noise-level mismatch and image-class mismatch problems. For the noise-level mismatch, the presence of (6) ensures that the weight is aware of how far the denoising strength deviated from the actual noise level $\sigma$. If $\hat{\sigma}_k$ is exactly $\sigma$, then $w_k^h = 1$. This happens for image-class mismatch because in the image-class mismatch we assume $\hat{\sigma}_k = \sigma$, for all $k$.

C. Practical Case

In the absence of the ground truth $z$, we can use Stein’s Unbiased Risk Estimator (SURE) [19] to estimate the MSE. Given a denoiser $D_k$, SURE is defined as

$$\text{SURE}_k = \frac{1}{N} \| y - \hat{z}_k \|^2 - \sigma^2 + \frac{2\sigma^2}{N} \sum_{i=1}^{N} \partial \hat{z}_{k,i} \partial y_i, \quad (8)$$

where $\hat{z}_{k,i}$ is the $i$-th component of the estimate $\hat{z}_k$. SURE is an unbiased estimator of the MSE, meaning that

$$\mathbb{E}[\text{SURE}_k] = \text{MSE}_k. \quad (9)$$

The accuracy of the estimator SURE$_k$ is determined by the number of pixels $N$. Thus, as $N$ grows, SURE$_k$ becomes more accurate.

In evaluating (8), the sum of the partial derivatives is also known as the divergence of the denoiser, $\text{div}D_k$. The divergence $\text{div}D_k$ is a measure of the sensitivity of the denoiser with respect to the noisy input. If $D_k$ is sensitive to $y$, then a small perturbation in the input will cause a significant perturbation in the denoised output.

Evaluating the divergence $\text{div}D_k$ can be done using a numerical scheme [17]. The idea is to define a random vector $b \sim \mathcal{N}(0, I)$ and define

$$y' = y + \epsilon b,$$

where $\epsilon < 1$ is a very small constant. Then, one can show that the divergence can be estimated as

$$\text{div}D_k \equiv \sum_{i=1}^{N} \frac{\partial \hat{z}_{k,i}}{\partial y_i} \approx \lim_{\epsilon \to 0} \mathbb{E} \left[ b^T \left( \frac{D_k(y') - D_k(y)}{\epsilon} \right) \right], \quad (10)$$

which is essentially the finite difference approximation to the actual derivative.

It should be noted that since the divergence is calculated using the first order approximation of the derivatives, it is possible that the estimated divergence is different from true divergence. This is especially true for highly nonlinear denoisers such as neural networks. In this case, the overall SURE estimate could be negative. However, this does not affect the performance of our proposed weight estimation method because what matters in defining (5) is the ordering and the gap between the MSEs.

D. Effectiveness

To illustrate the effectiveness of the weighting scheme, we conduct an experiment using the non-local means denoiser [4]. The non-local means has an intensity parameter $h$ determining the intensity cutoff of the non-local weight. In the original paper by Buades et al., this parameter $h$ is manually tuned to form a look-up table. The goal of this experiment is to see whether the weighting scheme can return a reliable estimate.

We construct five non-local mean denoisers with $h \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. The weights $w_k$’s are computed using (7), leading to the combined estimate $\hat{z}$. We test the denoiser for noise levels $\sigma = 10, 20, 30, 40$. The results are shown in Table 1. It is clear that $\hat{z}$ is able to pick the best denoiser. It also adds some minor PSNR improvement.

| $\sigma$ | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 |
|---------|------|-----|------|-----|------|
| $h$     |      |     |      |     |      |
| 10      | 31.76| 32.51| 31.04| 29.71| 28.70|
| 20      | 26.30| 29.47| 28.95| 28.01| 27.25|
| 30      | 21.18| 25.24| 27.14| 27.22| 26.80|
| 40      | 18.94| 23.48| 25.78| 26.02| 25.74|

Table 1: Example of denoiser-model mismatch. We construct five non-local mean denoisers with $h \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. The weights $w_k$’s are computed using (7), leading to the combined estimate $\hat{z}$. We test the denoiser for noise levels $\sigma = 10, 20, 30, 40$. The results are shown in Table 1. It is clear that $\hat{z}$ is able to pick the best denoiser. It also adds some minor PSNR improvement.

E. Other Possible Metrics

In measuring the quality of an image, the structural similarity index measure (SSIM) is the typical option besides MSE [20]. SSIM is better than MSE as it takes into account the quality perception of the human visual system (HVS) [9, 11]. However, as a full-reference metric, SSIM requires the ground truth which is impractical in our problem. Alternatively, we can use no-reference quality metrics such as [13, 15]. These metrics are based on natural scene statistics such as contrast, sharpness, and entropy. However, these metrics are not guaranteed to maximize PSNR.

2.4. The Booster Network $\mathcal{F}$

The second component of ConsensusNet is the booster network $\mathcal{F}$. In choosing $\mathcal{F}$, we must ask ourselves what functionality we expect $\mathcal{F}$ to have.

Recall the combined image $\hat{z}$ defined in (2), which is the linear combination of the $K$ initially denoised results $\hat{z}_k$. 


Since each $\tilde{z}_k$ is a denoised image, $\tilde{z}$ should also be a denoised image. The task of the booster $F$ is thus to remove the so called “method noise” remaining in $\tilde{z}$ [3]. The distribution of the method noise is typically Laplace. For example, Figure 4 shows the distribution of the method noise using DnCNN on Barbara512.

Figure 4: Distributions of the residues $y - z$ (red line) and $\tilde{z} - z$ (blue line) for actual noise levels $\sigma = 17, 27, 37, 47$. Note that after the initial denoising, the method noise becomes significantly more concentrated, thus making the training of the booster network $F$ feasible.

An important observation of Figure 4, besides the Laplacian distribution, is the concentration of the residue before and after initial denoising. Before the initial denoising, the distribution of the residue $y - z$ is Gaussian with a significant variation of the variance. However, after the initial denoising, the residue $\tilde{z} - z$ becomes concentrated. This makes the booster network $F$ easy to train because the variation of $\tilde{z}$ is significantly smaller than that of $y$.

The architecture of the booster network is shown in Figure 5. The network is a mixture of convolutional layers and deconvolutional layers. We use standard $3 \times 3$ kernels for both convolutional and deconvolutional layers. The presence of the deconvolutional layers is essential for the booster because the combined image $\tilde{z}$ is typically over-smoothed. Therefore, while in convolutional layers we aggregate multiple activations to a single activation (thus achieving feature extraction), in deconvolutional layers we disseminate single activation to multiple activations (thus enhancing the details). The usage of both convolutional and deconvolutional networks has been reported in [16]. Our booster network has 15 convolutional layers and 15 deconvolutional layers. We also adopt a skip connection method proposed in [14]. The idea is to ensure that the residue lost at the convolutional layer can be retained in the deconvolutional layer. If skip connections are not used, then noise will be sequentially eliminated and features cannot be retained.

Figure 5: Network structure of the proposed booster network. The network contains 15 convolutional layers followed by 15 deconvolutional layers. Skip connections are used to enforce symmetry of the network.

To train the booster network, we extract the non-overlapping patches of size $50 \times 50$ from each training dataset. For each patch we generate 8 variations by flipping and rotating at $0^\circ, 90^\circ, 180^\circ$ and $270^\circ$. The cost function we use in training the booster network is the standard $L_2$-loss. We have also tried using the $L_1$-loss but the improvement is marginal. During the training, we use ADAM optimizer with learning rate $10^{-4}$ and 50 epochs.

The effectiveness of the booster can be seen in Figure 6, where we crop a sub-region of an image from BSD500. In this example, we consider four different image denoisers (See Section 3.3 for experimental details), with noise level $\sigma = 50$. It is clear from the figure that fine details and contrast are improved by the booster.

Figure 6: Example showing the effectiveness of the booster in improving the details and contrast of the combined result.

3. Experiments

We consider three experiments to evaluate ConsensusNet. Unless provided by the authors, all neural networks are built using Tensorflow and are run on Intel(R) Core(TM) i5-4690K CPU 3.50GHz with an Nvidia Titan-X GPU.

3.1. Experiment 1: Noise-Level Mismatch

There are two objectives in this experiment. First, we want to evaluate the effectiveness of ConsensusNet in interpolating denoising performance when the initial denoisers are not trained for every noise level. Second, we want to compare the performance of ConsensusNet with existing blind denoisers such as [22] because these denoisers are trained to handle any noise level.
Regarding training and testing, we divide the Berkeley Segmentation Database (BSD500) into 300 training and 200 testing images. We train five initial denoisers $D_1, \ldots, D_5$ using two neural network denoisers: DnCNN [22] and REDNet [14]. For each denoiser, the denoising strength is set as one of the values $\sigma = 10, 20, 30, 40, \text{ and } 50$. When testing, we use a noise level of $\sigma \in [10, 50]$.

The results of this experiment are shown in Table 2, Figure 7, and Figure 8. Table 2 shows the comparison with REDNet as initial denoisers, whereas Figure 7 shows a visual comparison of an image in the BSD500 dataset. Figure 8 shows the trend of the PSNR as $\sigma$ increases.

There are three observations in this experiment. First, for each $\sigma$, the best performing REDNet is the one with $\hat{\sigma}$ right above $\sigma$. This result is consistent with the suggestion made in [22]. However, ConsensusNet is able to boost the performance by an average of 0.45dB for $\sigma = 15, 25, 35, 45$.

Second, comparing individual neural networks with ConsensusNet in Figure 8, more improvement is made when $\sigma$ lies in the midway of two consecutive $\hat{\sigma}$. When $\sigma = \hat{\sigma}$ exactly, the gain becomes insignificant. One should also pay attention to the case of $\sigma = 15$ in Figure 8. At low noise, the performance of REDNet (and DnCNN) changes more rapidly as $\sigma$ decreases. This is due to the fact that features are better retained at low noise, and thus the neural network has better discriminative power. One simple way to improve ConsensusNet is to make a finer interval of $\hat{\sigma}$.

Third, compared to blind DnCNN, we observe that ConsensusNet generally has a similar performance (except for $\sigma = 15$). The advantage of ConsensusNet, however, is that it is a modular framework that allows us to plug in any initial image denoiser. This offers the flexibility to handle other types of problems beyond noise level interpolation.
Table 3: Example of different image classes. Class-specific REDNets have better performance than BM3D, DnCNN (generic) and REDNet (generic). ConsensusNet selects the best class. We use 10 images from ImageNet for testing.

|                | BM3D (Generic) | DnCNN (Generic) | REDNet (Generic) | REDNet (Building) | REDNet (Face) | REDNet (Flower) | Ours +REDNet (Ours-Max) |
|----------------|----------------|-----------------|------------------|-------------------|--------------|----------------|-----------------------|
| Building       | 29.8571        | 30.3223         | 30.6248          | 30.9851           | 29.5022      | 29.8887        | 31.1045               | 0.1194                |
| Face           | 30.5405        | 30.9995         | 30.5405          | 30.8688           | 31.0916      | 31.0155        | 31.3293               | 0.2377                |
| Flower         | 31.6868        | 32.5036         | 32.5745          | 32.3602           | 32.2557      | 32.699         | 32.8324               | 0.1625                |

3.2. Experiment 2: Different Image Classes

The objective of this experiment is to evaluate the performance of ConsensusNet when the initial denoisers are trained for different image classes. To this end, we fix the type of initial denoisers as REDNet, and train three different REDNets using three classes of images: Flower, Face and Building. We have experimented with other initial denoisers such as DnCNN, but the results are similar. In training the initial denoisers, we manually select 200 class-specific images for each class from the ImageNet [8]. We fix the noise level as $\sigma = 20$ to eliminate the complication of having uncertainty in both noise levels and image classes.

The result of this experiment is shown in Table 3 with a few representative examples in Figure 9. We observe that denoisers trained with generic database such as DnCNN and REDNet perform worse than class-specific denoisers. For example, in the Building image, DnCNN and REDNet attain 30.84dB and 31.10dB respectively. A REDNet trained with Building class has a PSNR of 31.50dB, approximately 0.4dB above the generic REDNet. For Face and Flower classes, the same observation can be found, although the gap is less substantial. One reason is that for Building class, the vertical and horizontal features learned by the network are less common in generic images.

ConsensusNet is able to select the best denoiser in this experiment. However, the booster performs less aggressively than the other two experiments. The reason is that going across the class, the gap between the initial denoisers is typically large. As a result, it is more difficult to expect the other two initial denoisers to offer new information. For example, Face has a larger improvement because two of the initial denoisers have similar performance.
3.3. Experiment 3: Different Denoiser Types

The objective of this experiment is to evaluate ConsensusNet for different types of initial denoisers. To this end, we consider four denoisers running at specific noise levels \( \hat{\sigma} \) that match with the actual noise level \( \sigma \). These denoisers are BM3D [7], DnCNN [22], REDNet [14] and FFDNet [24]. We use the original implementation by the authors for DnCNN and FFDNet, and build our own REDNet.

The results of this experiment are shown in Table 4 with a visualization in Figure 10. Among the four denoisers, FFDNet and REDNet have comparable performance at the top, followed by DnCNN and then BM3D. For the five noise levels we tested, ConsensusNet consistently improves the performance. The PSNR gain with respect to the best denoiser is less significant for small \( \sigma \), but becomes more substantial for large \( \sigma \). One reason is that for high noise the initial denoisers tend to oversmooth. The boosting of the ConsensusNet is therefore very effective. The result in Figure 10 is also interesting. It is a case where BM3D performs better than neural networks. Nevertheless, ConsensusNet improves the performance by more than 1dB.

4. Discussion and Conclusion

We conclude the paper by discussing a few limiting factors of ConsensusNet. First, the number of initial denoisers \( K \) cannot be too large because in practice it is unlikely to have an excessive number of denoisers. On the other hand, \( K \) cannot be too small, for otherwise we will not be able to weigh out the less performing denoisers. From our experience, a reasonable \( K \) varies from 4 to 10, depending on the type of problem. The quality of denoisers is also important. Ideally, the optimal configuration of the initial denoisers is to have them perform similarly. The presence of an outlier typically hurts the performance.

An immediate extension of ConsensusNet is to handle multiple objects and classes in an image. This can be done by applying ConsensusNet on sub-regions of the image. In this case, the limiting factor will become the accuracy of SURE as SURE performs better for larger number of pixels \( N \), which could be satisfied for large-size images.

Acknowledgement. We thank Nvidia for their generous donation of the GeForce Titan-X GPU for this research.
References

[1] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive multicol-umn deep neural networks with application to robust image denoising. In Advances in Neural Information Processing Systems 26, pages 1493–1501. Curran Associates, Inc., 2013. 2

[2] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Trans Image Process, 18(11):2419–2434, Nov. 2009. 2

[3] A. Buades, B. Coll, and J. M. Morel. On image denoising methods. Technical report, Technical Note, CMLA (Centre de Mathematiques et de Leurs Applications, 2004. 5

[4] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In Proc. IEEE Conf Computer Vis Pattern Recogn, volume 2, pages 60–65 vol. 2, Jun. 2005. 2, 4

[5] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In Proc. IEEE Conf Computer Vis Pattern Recogn, pages 2392–2399, Jun. 2012. 2

[6] K. N. Chaudhury and K. Rithwik. Image denoising using optimally weighted bilateral filters: A sure and fast approach. In Proc. IEEE Int Conf Image Process, pages 108–112, Sep. 2015. 2

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Trans Image Process, 16(8):2080–2095, Aug. 2007. 1, 2, 8

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In Proc. IEEE Conf Computer Vis Pattern Recogn, 2009. 7

[9] K. Egiazarian, M. Ponomarenko, V. Lukin, and O. Ieremeiev. Statistical evaluation of visual quality metrics for image denoising. Available online at: https://arxiv.org/abs/1711.00693, Nov. 2017. 4

[10] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In Proc. IEEE Conf Computer Vis Pattern Recogn, pages 2862–2869, Jun. 2014. 2

[11] A. Hore and D. Ziou. Image quality metrics: PSNR vs. SSIM. In Proc. 20th Int. Conf. Pattern Recogn, pages 2366–2369, Aug. 2010. 4

[12] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In Proc. IEEE Conf Computer Vis Pattern Recogn, Jul. 2017. 2

[13] W. Lin and C.-C. J. Kuo. Perceptual visual quality metrics: A survey. J Vis Commun Image Represent, 22(4):297 – 312, 2011. 4

[14] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In Advances in Neural Information Processing Systems 29, pages 2802–2810. Curran Associates, Inc., 2016. 2, 5, 6, 8

[15] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. IEEE Trans Image Process, 21(12):4695–4708, Dec. 2012. 4

[16] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In Proc. IEEE Int Conf Computer Vis, pages 1520–1528, Dec. 2015. 5

[17] S. Ramani, T. Blu, and M. Unser. Monte-carlo sure: A black-box optimization of regularization parameters for general denoising algorithms. IEEE Trans Image Process, 17(9):1540–1554, Sep. 2008. 4

[18] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein. Deep class-aware image denoising. In Proc. Int Conf Sampling Theory and Applications (SampTA), pages 138–142, Jul. 2017. 2

[19] C. M. Stein. Estimation of the mean of a multivariate normal distribution. The Annals of Statistics, 9(6):1135–1151, Nov. 1981. 2, 4

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process, 13(4):600–612, Apr. 2004. 4

[21] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 341–349, 2012. 2

[22] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Trans Image Process, 26(7):3142–3155, Jul. 2017. 1, 2, 5, 6, 8

[23] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep CNN denoiser prior for image restoration. In Proc. IEEE Conf Computer Vis Pattern Recogn, 2017. 2

[24] K. Zhang, W. Zuo, and L. Zhang. FFDNet: Toward a fast and flexible solution for CNN based image denoising. Available online at: https://arxiv.org/abs/1710.04026, Oct. 2017. 2, 8

[25] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In Proc. Int Conf Computer Vis, pages 479–486, Nov 2011. 2