How Neural Network can Learn to Denoise an Image without any Training Data?

Ji, Hui (纪辉)
Department of Mathematics
National University of Singapore

08-July-2020
上海交通大学自然研究院
Reference and Co-authors

• Reference

• Main driving force behind the work

Quan, Yuhui  
SCUT, China

Chen, Mingqin  
SCUT, China

Pang, Tongyao  
NUS, Singapore
Image denoising

• Image denoising: A fundamental problem in imaging-relating applications
• Problem formulation

\[ y = x + n \]

\( y \): noisy image; \textbf{known} \hspace{0.5cm} \( x \): original image; \textbf{unknown} \hspace{0.5cm} \( n \): random noise; \textbf{unknown}

Removing noise from a noisy input image

Denoising
Review: From regularization to deep learning

• Regularization: Prominent technique image denoising in last decades
• Basic idea:
  – Imposing certain prior on image for separating noise and image
  – Formulating the problem into an optimization model with regularization term
• Two often-seen types of priors
  – Local prior
    • Modeling the image $\mathbf{x}$ as a function
    • Prior on local variations of the function
  – Global prior
    • Modeling the image $\mathbf{x}$ as the set of small image patches $\{x_i\}$
    • Statistical prior on the set of image patches $\{x_i\}$
Review: Regularization using local image prior

- Example
  Image is a piecewise smooth function $\ell_1$-norm relating regularizations

- TV (Total Variation) regularization
  $$\min_x ||y - x||_2^2 + \lambda ||\nabla x||_1, \quad \nabla = \left[ \frac{\partial}{\partial h}, \frac{\partial}{\partial v} \right]$$

- Wavelet/framelet regularization:
  $$\min_x ||y - x||_2^2 + \lambda ||Wx||_1, \quad W: \text{wavelet/framelet transform}$$
Review: From local intensity prior to global patch recurrence prior

• Example
  – Block-matching and 3D filtering (BM3D)

• Global recurrence prior on image patches
  \( \{x_i\} \in R^{m \times m} (R^{8 \times 8}) \)
  – Each image patch is likely to repeat itself for many times over the image \( x \)

• Procedure of BM3D
  – Grouping similar patches together
  – Processing the set of similar patches.

Red boxes; Green boxes; Blue boxes
Review: From pre-defined prior to data-driven sparsity prior

- Learning a sparse representation of image (image patches)
- K-SVD method for dictionary learning
  - Learning a dictionary for sparse representation of image patches
    \[
    \min_{D,c_j} \sum_j ||x_i - Dc_i||_2^2 + \lambda||c_i||_0, \quad \text{s.t.} \quad ||D_j||_2 = 1, 1 \leq j \leq J
    \]
    where \( D = [D_1, D_2, ..., D_J] \) is an over-complete dictionary
- Data-driven wavelet tight frame
  - Learning a wavelet tight frame for sparse representation of an image
    \[
    \min_{W^*,W} ||Wx - c||_2^2, \quad \text{s.t.} \quad ||c_i||_0 \leq L_0, 1 \leq i \leq L.
    \]
    where \( W \) is a wavelet tight frame transform
Review: From sparse-representation-learning to deep learning

- Deep learning: Training a deep neural network (DNN) to learn the mapping $f(\cdot; \theta)$ between noisy image $y$ and original image $x$

$$f(\cdot; \theta^*) : y \rightarrow x$$

- $\theta^*$: network weights learned on a set of noise/clean image pairs $\{y_i, x_i\}_{i=1}^N$

$$\theta^* = \arg \min_\theta \sum_{i=1}^N ||f(y_i; \theta) - x_i||_2^2$$

DnCNN: a denoising CNN Using residual learning, 2017
Review: Supervised learning is prominent in image denoising

- Most deep learning methods are **supervised** on the noisy/clean image pairs \( \{y_i, x_i\}_{i=1}^N \)

- **Pros:**
  - State-of-the-art performance, provided a large amount of training samples

- **Cons**
  - Collecting training samples can be challenging in many scenarios
  - The dataset can be biased, which leads to poor generalization.

- There is an increasing interest on relaxing requirement on trainings samples

- Noise2Noise (2018): training the NN on the set of noisy/noisy pairs \( \{y_i, \tilde{y}_i\}_i \)

\[
\begin{align*}
\{y_i = x_i + n_i \} & \quad \rightarrow \quad \min_{\theta} \sum_i \|f(y_i; \theta) - \tilde{y}_i\|_2^2 \\
\{\tilde{y}_i = x_i + \tilde{n}_i \}
\end{align*}
\]
Review: From supervised to unsupervised learning

• Consider a set of noisy images \( \{y_i\} \)
  \[
  \{y_i = x_i + n_i\}_{i=1}^{N}
  \]
  – Overfitting: Convergence to identity map \( I \)
  \[
  \min_{\theta} ||f(y_i; \theta) - y_i||_2^2 \quad \longrightarrow \quad f(\cdot; \theta) = I
  \]
• Noise2Void, Noise2self (2019): training the NN on the set of noisy images \( \{y_i\} \)
  – Blind-spot technique: using neighboring noisy pixels to predict centering pixels
  – The NN \( f(\cdot; \theta) \) is designed with blind spot, to avoid convergence to identity map
• Deep Image prior (2018): training the NN on the input \( y \) from random initial \( \epsilon_0 \)
  \[
  \{y = x + n\} \quad \longrightarrow \quad \min_{\theta} \sum_i ||f(\epsilon_0; \theta) - y_i||_2^2
  \]
  – Using early-stopping to prevent the NN converging to identity map
Review in summary

- Progress on methodology for image denoising
  - Tikhonov
  - Wiener
  - TV
  - Wavelet
  - Non-local means
  - BM3D

- Configuration on training samples for deep learning

<table>
<thead>
<tr>
<th>Training data</th>
<th>Noisy/Clean pairs</th>
<th>Noisy/Noisy pairs</th>
<th>Noisy images</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>Supervised denosing network</td>
<td>N2N</td>
<td>N2V, N2S SURE, GAN</td>
<td>DIP</td>
</tr>
</tbody>
</table>
Discussion

• Non-learning methods vs. deep learning methods
  – Deep learning: Great potential on performance
  – Non-learning methods: Easier to use as no need for training samples.
• There are few works on deep learning method with ZERO training sample
• Deep image prior (DIP)
  – Observation: NN is likely to generate regular structures before random noise during training
  – Idea: used early stopping to train an NN on the input image for image denoising
• Cons of DIP
  – Finding the optimal time to stop the training can be troublesome
  – The performance is not satisfactory, inferior to traditional methods (e.g., BM3D)
Self-supervised deep learning for image denoising

• **Question:** Without using any training sample, Is it possible for deep learning to solve the problem of image denoising with state-of-the-art performance?
• **Our hypothesis:** YES.
• **Rationale behind our hypothesis**
  1. Different from classification, image denoising does not need global feature.
  2. As shown in BM3D, image denoising can treat an image as the collections of many image patches \( \{x_\ell\} \)
  3. In image denoising, the network indeed can be viewed as learning a mapping from noisy patches to clean patches: \( \{y_\ell\}_{\ell=1}^N \rightarrow \{x_\ell\}_{\ell=1}^N \)
  4. Self-supervised learning is about learning the mapping on a dataset of noisy image patches without truth.
Self-supervised deep learning for image denoising

- **Problem setting**
  - Given only a noisy image $y$, related to clean image $x$ by
    \[
y = x + n, \quad n: \text{random noise}
    \]
  - Problem: how to train an NN $f(\cdot; \theta)$ w/o any additional info. to predict $x$ from $y$

- **Two key questions to answer**
  1. How to train an NN to predict $x$ from $y$, without any additional info?
    - Overfitting: Convergence to identity map
      \[
      \min_{\theta} ||f(y; \theta) - y||_2^2 \quad \longrightarrow \quad f(\cdot; \theta^*) = I \quad \longrightarrow \quad f(y; \theta^*) = y
      \]
  2. How to reduce large variance when only a single sample is available?
Data augmentation using $y$ for NN training

Question: How to train an NN to predict $x$ from $y$, without any additional info?

- Data augmentation by Bernoulli sampling.
  - The pairs of Bernoulli sampled instance and its counterparts
    \[ \{(\hat{y}_m = b_m \odot y, \quad \overline{y}_m = y - \hat{y}_m = (1 - b_m) \odot y)\} \]
  - $\odot$ denotes element-wise multiplication
  - $b_i$ denotes one instance of binary Bernoulli vector where each entry is drawn independently from a Bernoulli distribution with parameter $0 < p_0 < 1$.

\[
b_m[r] \sim \mathcal{B}(p_0), \text{ whose } \text{pr}(x) = \begin{cases} p_0, & x = 1; \\ 1 - p_0, & x = 0. \end{cases}
\]

- The network is trained to predict $\overline{y}_m$ from $\hat{y}_m$

\[ f(\cdot; \theta): \hat{y}_m \rightarrow \overline{y}_m, \quad 1 \leq m \leq M \]
Cost function

- Training samples: \( \{(\hat{y}_m = b_m \odot y, \ \bar{y}_m = y - \hat{y}_m = (1 - b_m) \odot y)\} \)

- Cost function for NN training

\[
\sum_m L(f(\hat{y}_m; \theta), \bar{y}_m) = L(\min_\theta \sum_m \|f(\hat{y}_m) - \bar{y}_m\|)_{\text{supp}(1-b_m)} \\
= \min_\theta \sum_m \|(1 - b_m) \odot (f(b_m \odot y) - y)\|_2^2
\]

– For each pair \( (\hat{y}_m, \bar{y}_m) \), the cost is the error using remaining pixel values of \( y \) in \( \hat{y}_m \) for predicting removed pixel values of \( y \) in \( \bar{y}_m \).
Understanding data augmentation

**Proposition.** Suppose the noise $n$ is i.i.d. random variables with zero mean and s.t.d. $\sigma$. The expectation of the loss function is then

$$
E \left[ \sum_{m=1}^{M} \|f(\hat{y}_m; \theta) - \bar{y}_m\|_{\text{supp}(1-b_m)} \right] = \sum_{m=1}^{M} (\|f(\hat{y}_m; \theta) - x\|_2^2 + q_m\sigma^2)
$$

where $q_m = \#(\text{supp}(1 - b_m))$.

- With enough samples $\{\hat{y}_m\}$, the proposed data augmentation enables training the NN on the pairs of noisy/clean image $x$

  $$
  \{(\hat{y}_m, x)\}_{m=1}^{M}
  $$

  $$
  \hat{y}_m = b_m \odot (x + n): \text{image with both noise } n \text{ and impulse noise } b_m
  $$
Variance reduction

• Self-supervised learning NN:
  – Many pairs of noisy images and the same truth image
    \[ \{(\hat{y}_i, x)\} \text{ (inexact)} \]

• Supervised learning
  – Many pair of noisy image and its corresponding truth image
    \[ \{(y_i, x_i)\}_{i=1}^{N} \]

• The key to fill the gap between supervised and self-supervised learning

How to reduce large variance caused by too few truth images?
An NN with Bernoulli dropout [Hinton et al. 2014]

Vanilla version
- $z^{(\ell+1)} = w_i^{(\ell+1)} y^{\ell} + b_i^{(\ell+1)}$
- $y_i^{(\ell+1)} = \phi(z_i^{(\ell+1)})$

Dropout version
- $r_i^{(\ell)} \sim Bernoulli(p_0)$
- $\tilde{y}^{(\ell)} = r_i^{(\ell)} \odot y^{(\ell)}$
- $z^{(\ell+1)} = w_i^{(\ell+1)} \tilde{y}^{\ell} + b_i^{(\ell+1)}$
- $y_i^{(\ell+1)} = \phi(z_i^{(\ell+1)})$
An NN with dropout

- Recall that the training samples are generated by Bernoulli sampling
  \[ \hat{y} = y \odot b_m, \quad \bar{y}_m = y \odot (1 - b_m), \quad b_m[i] \sim Bernoulli(p_0) \]

- Such Bernoulli sampling be implemented by adding a Dropout-layer in the first and in the end of an NN with specific connectivity.
- Dropout NN: A denoising NN with Bernoulli dropout
  \[ f (\cdot; \theta_1, \theta_2 \odot b), \quad b[i] \sim Bernoulli(p_0) \]
- A denoising network with dropout layers is trained as follows.
  \[ \{\theta_1^*, \theta_2^*\} := \min_{\theta_1, \theta_2} \sum_m L(f(y; \theta_1, \theta_2 \odot b_m), y) \]
Testing with dropout

- Most often-seen usage of the model trained with dropout
  - Training model: \( f (\cdot; \{\theta_1, \theta_2 \circ b\}) \)
  - Testing model: \( f (\cdot; \{\theta_1^*, p_0 \cdot \theta_2^*\}) \) (normalization)

- Our approach: running testing with dropout too
  - Generate \( N \) instances of dropout NN
    \[
    f_n = f (\cdot; \{\theta_1^*, \theta_2^* \circ r_n\}), \quad b_n[i] \sim Bernoulli(p_0), \quad n = 1,2,\ldots,N
    \]
  - Generating \( N \) estimates of clean image \( x \) from \( y \)
    \[
    \hat{x}_n := f_n(y) = f (y; \{\theta_1^*, \theta_2^* \circ b_n\})
    \]
  - Output
    \[
    \hat{x} := \frac{1}{N} \sum_{n=1}^{N} \hat{x}_n
    \]
Dropout: Bagging in Ensemble Learning

- Combining multiple simpler (base) models to a more powerful predictor

\[ \hat{x} := \frac{1}{N} \sum_{n} f_{m}(y) \]

Variance reduction by averaging

- Exploiting independence of base models for variance reduction

Dropout provided an effective and efficient way to construct many simpler NN models (with less nodes)
Bagging vs. Boosting

- Bagging: Dropout
- Boosting: Iterative method and unrolling-based deep learning

Average of \([x_1, x_2, x_3, \ldots, x_M]\)

Iterative refinement:
\[x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_M\]
Denoising with dropout: A Bayesian approximate MMSE estimator

- Bayes estimator with minimum mean squared error (MMSE)
  \[ x^* := \arg\min_x E[||\hat{x} - x||_2^2] = \text{bias}^2 + \text{variance} \]

- Suppose that NN is sufficiently parameterized such that
  \[ x = f(y; \phi), \quad \phi \sim q(\phi) \]

- The MMSE estimator
  \[ \hat{x} = E_{x|y} (x|y) = \int f(y; \phi)p(\phi|y)d\phi \]
  - The function \( p(\phi|y) \) is generally computationally intractable
  - One solution: Bayesian approximation to \( p(\phi|y) \)
Con’t

- Recall: The prediction of the average of the NN with dropout

\[ \bar{x} = \frac{1}{N} \sum_{n=1}^{N} f(y; \theta \odot b_n) \]

is a Monte-Carlo approximation to

\[ \bar{x} = E_r[f(y; \theta \odot b)] = \int f(y; \phi)p(\phi|\theta)d\theta \]

- MMSE estimator vs. Average prediction of NN with dropout

\[ \hat{x} = \int f(y; \phi)p(\phi|y)d\phi \quad \text{vs.} \quad \bar{x} = \int f(y; \phi)p(\phi|\theta)d\theta \]

- Using \( p(\phi|\theta) \) to approximate \( p(\phi|y) \)
- In dropout, \( \phi = \theta \odot b, \quad b \sim \text{Bernoulli}(p_0) \)
Con’t

• How well \( p(\phi|\theta) \) approx. \( p(\phi|y) \) determines how well \( \tilde{x} \) approx. MMSE \( \hat{x} \)

• In the presence of \( n \sim \mathcal{N}(0, \sigma^2 I) \), training the NN with dropout is minimizing the KL-divergence between two distributions \( p(\phi|y) \) and \( p(\phi|\theta) \)

\[
\min_{\theta} E_b[L(f(y; \theta \odot b) - y)] \equiv \min_{\theta} D_{KL}( p(\phi|y) \| p(\phi|\theta)),
\]

where \( D_{KL}(\cdot \| \cdot) \) denotes the KL-divergence

• In short, training and testing a dropout-NN is about finding a Bayesian approximation to the MMSE estimator of truth \( x \)

\[
x^* = E_b[f(y; \theta^* \odot b)], \quad \theta^* = \text{argmin}_{\theta} [L(f(y; \theta \odot b) - y))]
\]
NN implemented for experiments

An encoder-decoder NN
- six encoder’s blocks (EBs)
- Five decoder’s blocks (DBs)
- \( p_0 = 0.3 \)
Ablation study

- Quantitative metric
  - Peak signal-to-noise ratio (PSNR)
  - Structural Similarity Index (SSIM)

- Ablation study: keep all other the same; except removing
  - Dropout: Training the NN without dropout
  - Ensemble: Testing the NN without dropout
  - Sampling: Training the NN using the image without Bernoulli sampling

<table>
<thead>
<tr>
<th>Ablation (w/o)</th>
<th>dropout</th>
<th>ensemble</th>
<th>sampling</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR(dB)</td>
<td>24.08</td>
<td>29.69</td>
<td>23.09</td>
<td>31.36</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.67</td>
<td>0.92</td>
<td>0.75</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Performance impact by ensemble size of prediction

- Blue bars denote the PSNR results of individual inferences.
- Red curves denote the cumulative average PSNR.

Results on "Lena"

Results on "Peppers"
Performance impact by training iterations

- DIP is another unsupervised deep learning method w/o any external sample
  - Observation: regular image patterns appear before random noise during training
  - Technique: using early-stopping to prevent noise showing in the result

![Graph showing PSNR versus number of training iterations for DIP and Ours with different image datasets and noise levels.](graph.png)
### Removing Gaussian white noise from images (w/o quantification)

**Metric:** PSNR/SSIM

#### Single-image-based methods

<table>
<thead>
<tr>
<th>Dataset</th>
<th>sigma</th>
<th>(C)BM3D</th>
<th>N2V(1)</th>
<th>N2S(1)</th>
<th>DIP</th>
<th>DIP*</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set9</td>
<td>25</td>
<td>31.67/0.96</td>
<td>28.12/0.91</td>
<td>29.30/0.94</td>
<td>30.49/0.94</td>
<td>30.77/0.94</td>
<td>31.74/0.96</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>28.95/0.92</td>
<td>26.01/0.87</td>
<td>27.25/0.90</td>
<td>24.62/0.81</td>
<td>28.23/0.91</td>
<td>29.25/0.93</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>27.36/0.89</td>
<td>24.18/0.83</td>
<td>25.85/0.86</td>
<td>19.80/0.63</td>
<td>26.64/0.88</td>
<td>27.61/0.91</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>26.04/0.87</td>
<td>23.55/0.78</td>
<td>24.67/0.85</td>
<td>16.23/0.47</td>
<td>25.41/0.86</td>
<td>26.27/0.88</td>
</tr>
<tr>
<td>BSD68</td>
<td>25</td>
<td>28.56/0.80</td>
<td>25.34/0.68</td>
<td>27.19/0.77</td>
<td>27.47/0.74</td>
<td>27.96/0.77</td>
<td>28.70/0.80</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>25.62/0.69</td>
<td>23.85/0.62</td>
<td>24.53/0.64</td>
<td>21.45/0.42</td>
<td>25.04/0.65</td>
<td>25.92/0.70</td>
</tr>
</tbody>
</table>

#### Dataset-based deep learning methods

<table>
<thead>
<tr>
<th>Dataset</th>
<th>sigma</th>
<th>Ours</th>
<th>N2V</th>
<th>N2S</th>
<th>N2N</th>
<th>(C)DnCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set9</td>
<td>25</td>
<td>31.74/0.96</td>
<td>30.66/0.95</td>
<td>30.05/0.94</td>
<td>31.33/0.96</td>
<td>31.42/0.96</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>29.25/0.93</td>
<td>27.81/0.91</td>
<td>27.51/0.90</td>
<td>28.94/0.93</td>
<td>28.84/0.92</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>27.61/0.91</td>
<td>25.99/0.87</td>
<td>26.49/0.88</td>
<td>27.42/0.91</td>
<td>27.36/0.90</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>26.27/0.88</td>
<td>25.37/0.86</td>
<td>25.46/0.86</td>
<td>26.45/0.89</td>
<td>26.30/0.88</td>
</tr>
<tr>
<td>BSD68</td>
<td>25</td>
<td>28.70/0.80</td>
<td>27.72/0.79</td>
<td>28.12/0.79</td>
<td>28.86/0.82</td>
<td>29.14/0.82</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>25.92/0.70</td>
<td>25.12/0.68</td>
<td>25.62/0.68</td>
<td>25.77/0.70</td>
<td>26.20/0.71</td>
</tr>
</tbody>
</table>
More experiments on other noise types

• Removing pepper-and-salt noise

<table>
<thead>
<tr>
<th>Dropping Ratio</th>
<th>CSC</th>
<th>DIP</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>32.97/0.912</td>
<td>33.48/0.930</td>
<td><strong>35.14/0.954</strong></td>
</tr>
<tr>
<td>70%</td>
<td>28.44/0.855</td>
<td>28.50/0.848</td>
<td><strong>31.06/0.897</strong></td>
</tr>
<tr>
<td>90%</td>
<td>24.34/0.712</td>
<td>24.24/0.727</td>
<td><strong>25.91/0.792</strong></td>
</tr>
</tbody>
</table>

• Removing real noise from images

<table>
<thead>
<tr>
<th>Metric</th>
<th>CBM3D</th>
<th>TWSC</th>
<th>DIP</th>
<th>N2V</th>
<th>N2S</th>
<th>DnCNN</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>36.98</td>
<td>36.10</td>
<td>34.80</td>
<td>34.08</td>
<td>35.46</td>
<td>37.55</td>
<td>37.52</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
<td>0.95</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>
Visualization of some examples

- Removing Gaussian noise from image

PSNR
20.3 → 33.1
Visualization of some examples

- Removing pepper-and-salt noise (randomly missing pixels)

![Image of noise pattern and result](image-url-removed)

90% missing $\Rightarrow$ PSNR 25.7
Visualization of some examples

- Removing real noise from image

PSNR
33.5 → 34.7
Paper and code @ github

https://github.com/scut-mingqinchen/self2self

Self2Self With Dropout: Learning Self-Supervised Denoising From Single Image

In this repository we provide the official implementation of Self2Self with Dropout.

General Information

- Codename: Self2Self (CVPR 2020)
- Writers: Yuhui Quan (csyhquan@scut.edu.cn); Mingqin Chen (csmingqinchen@mail.scut.edu.cn);
  Tongyao Pang (matpt@nus.edu.sg); Hui Ji (matjh@nus.edu.sg)
- Institute: School of Computer Science and Engineering, South China University of Technology;
  Department of Mathematics, National University of Singapore

For more information please see:

- [paper]
- [supmat]
- [website]
Discussion

• A self-supervised deep learning method for image denoising
  – No need for any training data
  – Provide state-of-the-art performance
• Key idea in the method for self-supervised learning
  – Bernoulli-sampling-based data augmentation
  – Dropout-based NN training and testing
• One sentence to summarize

Without external training samples, un-supervised/self-supervised deep learning can be as powerful as supervised-learning method, in image denoising.
Any question?