**NEURAL ADAPTIVE IMAGE DENOISER**

Sungmin Cha and Taesup Moon

Department of Electrical and Computer Engineering
Sungkyunkwan University (SKKU)

**Abstract**

We propose a novel neural network-based adaptive image denoiser, dubbed as Neural AIDE. Unlike other neural network-based denoisers, which typically apply supervised training to learn a mapping from a noisy patch to a clean patch, we formulate to train a neural network to learn context-based offline mappings that get applied to each noisy pixel. Our formulation enables using SURE (Stein's Unbiased Risk Estimator)-like estimated losses of those mappings as empirical risks to minimize. In results, we can combine both supervised training of the network parameters from a separate dataset and adaptive fine-tuning of them using the given noisy image subject to denoising. Our algorithm with a plain fully connected architecture is shown to attain a competitive denoising performance on benchmark datasets compared to the strong baselines. Furthermore, Neural AIDE can robustly correct the mismatched noise level in the supervised learning via fine-tuning, of which adaptivity is absent in other neural network-based denoisers.

**Introduction**

- **Grayscale image denoising**
  - Various denoising methods have been proposed
    - EX) BM3D, WNNM, EPLL, MLP and DnCNN
  - Especially, CNN based image denoising methods recently surpassed the previous state-of-the-arts
- **Drawback of CNN based image denoising methods**
  - Solely based on offline batch training
  - Lacks adaptivity to the given noisy image

![Diagram](Image)

**Problem setting**

- $x^{n \times n}$: the clean grayscale image, and each pixel $x_i \in [0,255]$
  - Each pixel is corrupted by an independent additive noise to result in a noisy pixel $z_i, i.e., z_i = x_i + n_i$
  - $E(N_i) = 0, E(N_i^2) = \sigma^2$

**Estimated loss function for affine denoiser**

- **Affine denoiser**
  \[
  \hat{x}_i(z_i) = a(z_i) + b(z_i)
  \]

- **Estimated loss function for single letter setting**
  \[
  \hat{z}_i(z_i) = a(z_i) + b(z_i)
  \]

- **For each location $i$, given $z_i$**
  \[
  \mathcal{L}(z_i(a(z_i), b(z_i))) = (z_i - (a(z_i) + b(z_i)))^2 + 2\alpha \sigma^2
  \]

- **Neural AIDE**
  \[
  \hat{x}_i(z_i) = a(w, c_i^{(1)}) + b(w, c_i^{(1)})
  \]

- **Two steps for training Neural AIDE**
  1. Supervised training: minimize
  \[
  \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(z_i(a(w, c_i^{(1)}), b(w, c_i^{(1)})))
  \]
  - $(\hat{z}, \hat{x})$: Collected abundant clean and noisy image pairs
  2. Adaptive train with even noisy image: minimize
  \[
  \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(z_i(a(w, c_i^{(1)}), b(w, c_i^{(1)})))
  \]
  - $z_i$: Given noisy image
  - **The reconstruction at location $i$ by Neural AIDE**
  \[
  \hat{x}_i(z_i) = a(w, c_i^{(1)}) + b(w, c_i^{(1)})
  \]

**Experimental results**

- **Experimental settings**
  - **Training data:** 2000 training images
  - **Test data:** The 11 standard benchmark images, the 68 Berkeley test images
  - **Context size:** $17 \times 17 - 1$
  - **Model:** 9 FC layers with 512 nodes
  - **Keras with tensorflow backend**

**The 11 standard benchmark images**

![Table](Image)

**Neural AIDE considerably preserve PSNR in mismatched cases!**

**The 68 Berkeley test images**

![Table](Image)