Comparing supervised and unsupervised learning in image denoising

Hanyun Wang

Chongqing Foreign Language School, Chongqing, 400065, China

whywhy_Whyyyzbb@163.com

Abstract. Recent studies on unsupervised learning have attracted people's increasing attention. In particular, Deep learning has developed rapidly in recent years. With the development of media images, people's demand for image noise reduction is increasing, and the requirements are becoming more and more strict. The traditional methods used for image noise reduction are far from meeting people's requirements, and people are eager to find a more efficient image noise reduction technology. In recent years, the technology of using a convolutional neural network for image noise reduction has become more and more skilled. This paper explores the reliability of image noise reduction technology using a convolutional neural network as an autoencoder, and whether good performance is maintained without using clean images. The article aims to compare the performance with supervised learning and unsupervised learning by deep learning in image denoising.

Keywords: Unsupervised learning, Image noise reduction, Convolutional neural network, Performance.

1. Introduction

In today's era, whether it is in the field of streaming media, in the field of medicine, in the field of archaeology, etc., image noise reduction technology plays a vital role. People have become less tolerant of noisy images and crave clean images. In this experiment, we will explore the performance of n2n technology in the field of image noise reduction. In the traditional image noise reduction technology, the methods people use are far from meeting the current requirements. At present, most people use deep neural networks to denoise images. In the current research, most scholars use the supervised method to denoise the image, that is, give a noisy image to the model, and at the same time tell the model what the image should look like. This supervised method has achieved good results, but this also has a common disadvantage, that is, in practical applications, it is difficult to collect a rush image and a noisy image of the recording of the same thing [1]. In this experiment, we are more interested in unsupervised image denoising techniques, we want to know if denoising images in an unsupervised manner still has good performance.

2. Related work

2.1. Encoder-Decoder

Some scholars use autoencoder to denoise images. Encoder-Decoder mainly has two network structures, one is an encoder and the other is the decoder. In this network structure, the convolutional neural network is mainly used to operate the original image. After the encoder, the original image is embedded into the latent space, and the information in the latent space is mapped to the latent space through the decoder of the same number of layers, the original space. Chen et al. combining autoencoder, deconvolutional networks, and shortcut connections into a residual encoder-decoder convolutional neural network (RED-CNN) for low-dose CT imaging [2]. After patch-based training, the proposed RED-CNN achieves competitive performance against state-of-the-art methods in both simulated and clinical cases. Shan et al. use a conveying path-based convolutional encoder-decoder (CPCE) network in 2-D and 3-D configurations within the GAN framework for LDCT denoising [3].

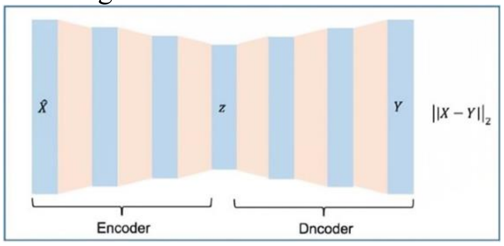

Figure 1. The autoencoder model.

In the autoencoder model, the input is a noisy image, where the Encoder maps the input image $\hat{X}$ to the latent space z, for the n2c method, the output Y is a clean target image, X and Y do a L_2 loss, for the n2n algorithm, the output Y is the target image with noise added, and X and Y do a L_2 loss (Figure1).

2.2. Fully convolution network

This means that the entire network uses convolutional layers, and the output image is subjected to convolution operations. As for the more specific convolution methods, different models have different methods, and some use the convolution kernel size of 3 or 5 [4]. Gholizadeh-Ansari et al., propose a deep neural network that uses dilated convolutions with different dilation rates instead of standard convolution helping to capture more contextual information in fewer layers and have employed residual learning by creating shortcut connections to transmit image information from the early layers to later ones [5].

2.3. Gan based algorithms

In this way, the network has two parts, a generator and a discriminator. Generation Using a convolutional neural network will generate a fake image, and the discriminator will identify whether the image is real or fake. This adversarial way of training forces the generator to produce images that are more and more similar to clean images throughout the training process. Some denoising method based on the generative adversarial network (GAN) with Wasserstein distance and perceptual similarity [4]. Some researchers train the FCN-based denoising network with fill-size images, which is computationally efficient due to the reuse of feature maps from the lower layers [6].

3. Method

In this part we will explore the specific methods of Noise2noise. In n2n, the network structure used by the model is a residual connected autoencoder. The autoencoder inputs the original image X to the Encoder, and through a series of convolution operations, maps the image information to the latent space z, and then restores the image through the same number of convolutional layer Decoders [7][8].

3.1. Prepare

First, let's introduce some operator symbols. We first define a noise image $\hat{X}=\{\hat{x}_1,\hat{x}_2,\dots\hat{x}_n\}$. Define $Y=\{y_i\}$ as clean target output and $\hat{Y}=\{\hat{y}_i\}$ as noise output.

3.2. Train

The purpose of our experiments is to compare the network performance of the supervised and unsupervised methods. So, for supervised networks, we feed the noisy samples $\hat{X}$ to the network, and compare the output of the network with the clean target output Y. We use the following loss to optimize the entire network.

$$\underset{\theta}{argmin} E_{\hat{x},y}\{L(f_\theta(\hat{x}),y)\} \tag{1}$$

Here the f network is the autoencoder, and θ is the parameter of the entire network. The goal is to optimize the autoencoder by minimizing this loss, and finally make the image output by the Decoder consistent with the clean target image.

Formula (1) mainly explores the performance of supervised learning in image noise reduction. As another unsupervised method, we input the noisy image to the Encoder, and let the Encoder map the image information into the latent space. After the Decoder Then restore the image, this time we use the noisy image $\hat{Y}$ as the output comparison, the loss is as follows:

$$\underset{\theta}{argmin} E_{\hat{x},\hat{y}}\{L(f_\theta(\hat{x}),\hat{y})\} \tag{2}$$

In formula (2), we keep making one noisy image close to another noisy image, but since the noise is irregular, the network cannot capture the two irregular noised images, mapping, but both images have latent representations that share common clean image semantic features. In this way, the network can also theoretically output a clean image.

In equations (1) and (2) above, we listed our goals for both methods, and now we write in detail how to use our network output to calculate the corresponding loss function. For formula (1), we input $\hat{X}$ and calculate the L_2 loss of output and $f_\theta(\hat{x}_i)$ and Y, the specific loss is as follows:

$$L_{n2c}=\frac{1}{n}\sum_{i=1}^{n}\|f_\theta(\hat{x}_i)-y_i\|_2 \tag{3}$$

For equation (2), we input $\hat{X}$, and compute the L_2 loss between the network's output $f_\theta(\hat{X})$ and the noise-added target output, The detailed loss is as follows:

$$L_{n2c}=\frac{1}{n}\sum_{i=1}^{n}\|f_\theta(\hat{x}_i)-\hat{y}_i\|_2 \tag{4}$$

Here n represents the number of training samples. The detailed algorithm pseudocode is as follows:

Algorithm 1 n2c Algorithm

```
Input: The set of noise image $\hat{X}=\{\hat{x}_i\}$ and $i=1,2\dots n$. The set of clean image targets $Y$.
Output: optimal $\theta$;
 1: initial Network parameters $\theta$;
 2: repeat
 3:    Enter $\hat{X}$ into the network;
 4:    Get the output of the network $f_\theta(\hat{X})$;
 5:    Calculate the $L_2$ loss between $Y$ and $f_\theta(\hat{X})$;
 6:    Optimize the network parameters $\theta$ according to equation(3);
 7: until convergence
```

We show in detail the steps of the n2c algorithm in Algorithm 1 and the steps of the n2n in Algorithm2.

Algorithm 2 n2n Algorithm

Input: The set of noise image $\hat{X} = \{\hat{x}_i\}$ $and\ i = 1,2 \dots n$. The set of clean image targets $\hat{Y}$.
Output: optimal θ;

Algorithm 2 (continue)

```
1: initial Network parameters $\theta$;
2: repeat
3:    Enter $\hat{X}$ into the network;
4:    Get the output of the network $f_\theta(\hat{X})$;
5:    Calculate the $L_2$ loss between $\hat{Y}$ and $f_\theta(\hat{X})$;
6:    Optimize the network parameters $\theta$ according to equation (3);
7: until convergence
```

3.3. Evaluate

3.3.1. PSNR. PSNR is an objective criterion for evaluating images. It is local, PSNR is the abbreviation of "Peak Signal to Noise Ratio". And ratio means ratio or ratio, psnr is generally used for an engineering project between maximum signal and background noise. Usually after image compression, the output image is usually different from the original image to some extent. In order to measure the image quality after processing, people usually refer to the PSNR value to measure whether a processing program is satisfactory. It is the logarithm of the mean square error between the original image and the processed image relative to $(2^n - 1)^2$ (the square of the maximum signal value, n is the number of bits per sample value), the unit is dB. Its formula is as follows:

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{5}$$

The formula for MSE is as follows:

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{6}$$

3.3.2. SSIM. SSIM is an indicator to measure the similarity of two pictures, and its formula is as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(\sigma_{xy} + c_2)}{(\mu_x^2\mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7}$$

Here μ_x and μ_y represent the average of x and y respectively, σ_x and σ_y represent the standard deviation of x and y respectively, σ_{xy} represents the covariance of x and y , c_1, c_2, c_3 represent constants, avoiding the denominator of 0.

4. Experiment

4.1. Dataset

In the experiments, we assess our result on two widely used digital image processing dataset:

- Kodak: A natural image dataset with 24 samples. Each image is either 768x512 or 512x768 in size. In this experiment, we will use Kodak only in the validation phase.
- BSD300: BSD is a dataset used frequently for image denoising and super-resolution. We use 200 datasets for training and 100 datasets for testing.
- COCO: We downloaded the validation set from the COCO dataset, which contains 5000 high-definition images, as a training set for our model.

In the processing stage of the data set, since the image size of the original data set is not fixed, it is impossible to use a general model to process images of different sizes, so we randomly crop the training set and randomly select each image. Cropped twice, each with a size of 256x256 for three channels. In the testing phase, we cannot use the cropping method to process the image, so we use the padding method for the image so that the size of the image can be restored to the same size image through the decoder after the encoder of the model. For the choice of noise, we choose Gaussian noise, salt and pepper noise, and Poisson noise, where Gaussian noise we choose = 0 and a = 25, We wanted to see if both methods performed well on different noises.

4.2. Noise

In order to complete this experiment, we first have to generate various kinds of noise on an image, we need:

- Gaussian noise: We need to write a function that randomly generates Gaussian noise with a given mean on an image.
- Poisson noise: We need a function that can generate Poisson noise on an image.
- Salt and pepper noise: We need a function that can generate salt and pepper noise on an image.

For the experimentally generated results, we use PSNR and SSIM to evaluate the performance of the model. A conceivable experimental result is that we run different data sets with two different models to generate evaluation indicators. We record the evaluation. indicators to compare the performance of the two models. This time we need, this time we use There are 3 kinds of noise, 2 training sets and 2 models, so a total of 16 experiments are required.

4.3. Experimental details

This experiment used the reproduced pytorch code. In the Encoder and Decoder stages, we used the convolutional neural network according to the original text. We used 5 block convolutional layers in the Encoder layer, each of which contains a 3x3 block convolutional layer. A convolution kernel with stride 1, pandding 1, and a LeakyReLU layer and a MaxPool layer. We use both L_2 loss and L_1 loss to perform Training. The learning rate is le-3, the learning rate is dynamically adjusted, and the optimizer uses Adam. We evaluate using PSNR and SSIM.

4.4. Experimental Results

4.4.1. LOSS. Figure2 shows the loss trend after model training to verify whether our model is trained normally.

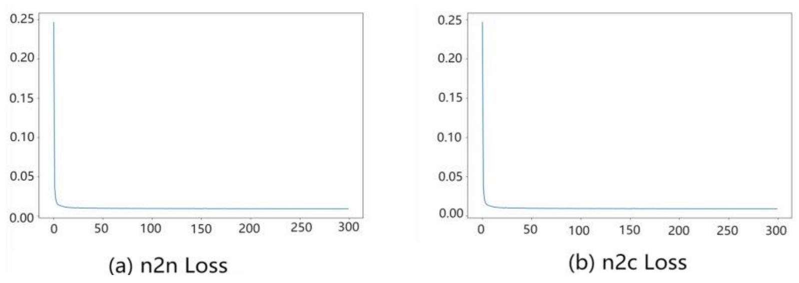

Figure 2. the loss trend after model training.

This is the loss map after training 300 epochs using the BSD300 dataset, (a) is the loss map when using a noisy target as the target, (b) is the loss map when using a clean target as the target

4.4.2. Evaluation. In order to see the results of using different training sets and using noisy images as targets and clean images as targets, the experimental results are as follows.

Table 1. Training results with BSD300 as the training set and Kodak as the validation set.

	noise	PSNR(top1)	SSIM(top1)
n2n	Gaussian	32.69	86.05
	Poisson	35.75	93.33

Table 1 . (continue)

	noise	PSNR(top1)	SSIM(top1)
n2c	Gaussian	32.03	85.61
	Poisson	35.32	92.45

Table 2. Training results with COCO as the training set and Kodak as the validation set.

	noise	PSNR(top1)	SSIM(top1)
n2n	Gaussian	33.95	89.88
	Poisson	37.65	94.42
n2c	Gaussian	33.65	89.43
	Poisson	36.84	94.32

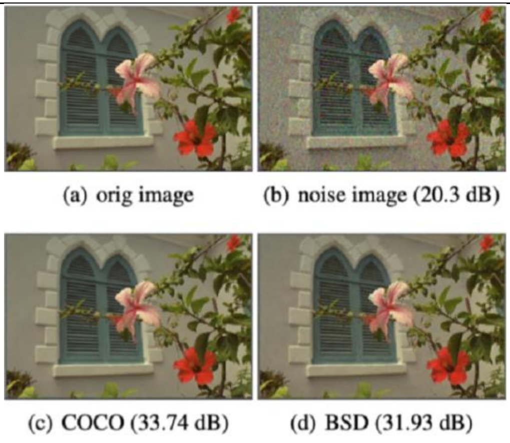

Figure 3. the denoising results of images when using different training sets.

We use two different training sets, BSD300 and COCO, and use Kodak for validation. We use Gaussian noise and $\sigma = 25$. (a) represents the original image, and (b) represents the image after Gaussian noise processing. (c): The result of denoising the image by using the COCO training set. (d): The result of denoising the image by using the BSD300 training set.

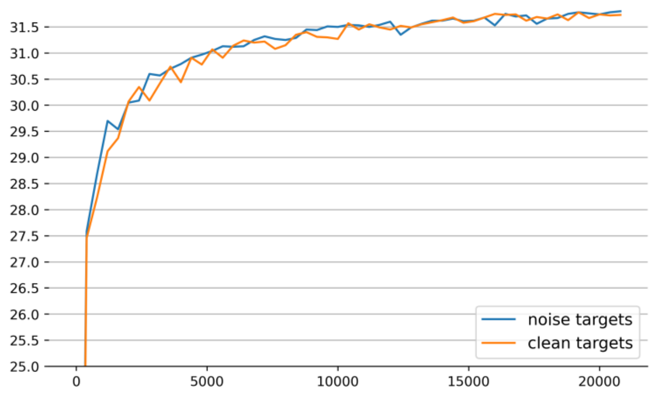

Figure 4. Denoising performance (dB in KODAK dataset) as a function of training epoch for additive Gaussian noise and $\sigma = 25$. White Gaussian noise, clean and noisy targets lead to very similar convergence speed and eventual quality (Figure 4).

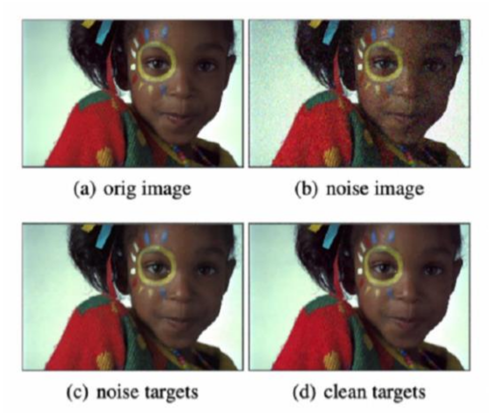

Figure 5. Denoising performance (dB in KODAK dataset) as a function of training epoch for additive Gaussian noise and $\sigma = 25$. In the figure 5, (a) represents the original image, and (b) represents the image after Gaussian noise processing. (c) and (d) represent the output of the two methods.

BSD300 data set, so the experimental result is that using COCO as the training set is more effective.

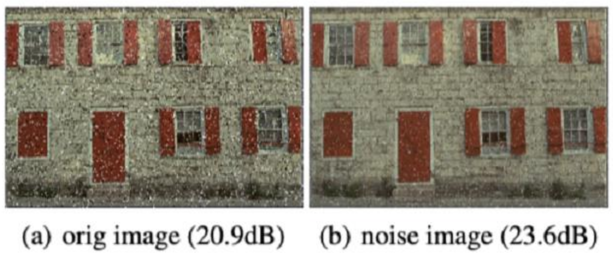

Figure 6. Denoising performance (dB in KODAK dataset) as a function of training epoch for additive Salt and peppemoise. In the figure, (a) represents the image after Salt and pepper noise processing, (b) represents the image after noise reduction.

We use BSD300 data and COCO dataset for training and Kodak dataset for test. We use clean images (clean targets) and noise images (noise targets) as outputs respectively, and compare the difference between them. And all the experimental results are visualized and tabulated to analyze the performance comparison and usage range of the two methods.

A clean image and a noise-processed image are used as the output of the network, respectively. The goal is to compare the performance of using the noised image as the output and using the clean image as the output directly, how. As shown in Figure 4, we add Gaussian noise to the image, and we use PSNR to evaluate the data during network training. It can be clearly seen that there is no significant difference between the two methods, even adding noise most of the time. The output looks better than a clean image as the output. It can be inferred that the performance of the method using the noised image as the output comparison is completely comparable to the performance of the method using the clean image as the output comparison.

For the two methods compared in this paper, the main purpose is to verify the effectiveness of these two methods for image noise reduction. From Figure 5, it is clear that there is not much difference in image noise reduction. Figure 6 shows the performance of Salt and pepper noise reduction by n2c (n2c is not shown, because the effect of n2n noise reduction is not much different from that of n2c). It is easy to see that the two methods of n2n and n2c are in The noise reduction effect under Salt and pepper noise is not very good, and it can be analyzed that this noise reduction method can only be applied to specific noises.

5. Conclusion

After a series of experiments, it can be concluded that in the same data set, by adding the same noise to the same image, the performance of n2n and n2c is not much different, but both methods have a certain scope for noise. Both methods have good performance for the noise reduction of Gaussian noise and Poisson noise, but not very good for the noise reduction of salt and pepper noise. At the same time, by comparing the training of the BSD300 and COCO datasets, it is obtained that the best trained network will have better noise reduction if a larger dataset is used for training.

References

[1] Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M.» and Aila, T. Noise2noise: Learning image restoration without clean data. Technical report, 2018.
[2] Chen, H., Zhang, Y., Kalra, M. K., Lin, F., Chen, Y.» Liao, R, Zhou, J., and Wang, G. Low-dose ct with a residual encoder-decoder convolutional neural network. Technical report, 2017.
[3] Shan, H., Zhang, Y. Yang, Q., Kruger, U., Kalra, M. K., Sun,
[4] Yang, Q・,Yan, P., Zhang, Y., Yu, H., Shi, Y., Mou, X., Kalra,
[5] Gholizadeh-Ansari, M.， Alirezaie, J., and Babyn, R Deep learning for low-dose ct denoising using perceptual loss and edge detection layer. Technical report, 2020.
[6] Choi, K., Kim, S. W., and Lim, J. S. Real-time image reconstruction for low-dose ct using deep convolutional generative adversarial networks (gans). Technical report, 2018.
[7] L.,Cong, W., and Wang, G. 3-d convolutional encoderdecoder network for low-dose ct via transfer learning from a 2-d trained network. Technical report, 2018.
[8] M. K., Zhang, Y., Sun, L.» and Wang, G. Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss. Technical report, 2018.