GED-based Image Denoiser Performed on Residual Learning

Tianyu Zhang[*]
Department of Computer Science
New York University
New York, NY 10003
`tz1280@nyu.edu`

Sree Gowri Addepalli
Department of Computer Science
New York University
New York, NY 10003
`sga297@nyu.edu`

Hui Wang
Department of Computer Science
New York University
New York, NY 10003
`hw1832@nyu.edu`

Abstract

The project is a continuation of the topic in [1]. Note that the authors in [1] tackled the image denoising problem by training a Deep CNN model with blind Gaussian noise (i.e., Gaussian noise with random standard deviation). In our project, we use blind noise out of Generalized Gaussian distribution (GED). As such, our trained model is able to de-noise wider range of noisy images.

1 Introduction

In the paper [1], the authors studied a residual learning problem, i.e., separating the noise from a noisy image by feed-forward convolutional neural networsk (CNN). By residual learning, we refer to the process that, rather than the output being the clean image, the output of the CNN is the noise. The authors observed that, the residual learning together with batch normalization technique can enhance each other in terms of speeding up the training and boosting the denoising performance.

1.1 Blind Gaussian Denoising: the author's original input generation process

To train the CNN model for blind Gaussian denoising, the authors in paper [1] generated noise by Gaussian distribution with a random standard deviation in the range $\sigma \in (0, 55/255)$ and then add the noise into the image in the training sample.

1.2 Generalized Gaussian Distribution

In reality, it is not true that the noise follows Gaussian distribution. For example, white noise image are typically assumed to be independent random variables with uniform probability distribution. Therefore, we seek a class of random distribution which is more general and more representative to the reality. A good candidate is the Generalized Gaussian Distribution (GED). The following is the probablity density function of GED:

$$f(x; \beta, \alpha, \mu) = \frac{\beta}{2\alpha\Gamma(1/\beta)} e^{(-|x-\mu|/\alpha)^{\beta}}. \tag{1}$$

[*]Github: `https://github.com/tianyu-z/GEDImageDenoiser`

GED is a class of random distribution. Among all the other properties, GED is reduced to Gaussian when $\beta = 2$; GED can mimic the uniform distribution on the interval $(\mu - \alpha, \mu + \alpha)$ when β is large (empericall, $\beta \sim 5$).

1.3 The approach in this project

We therefore use GED to generate the noise with β in a discretized range in $(0.83, 6)$. This choice of β is motivated by the Kurtosis level of the GED, where Kurtosis is a measurement of tailness, i.e., the likelihood of a large random sample can occur.

2 Method

We agree with the paper [1] that CNN's deep archietecture, with Rectifier Linear Unit (ReLU) and batch normalization, has been proved to be effective and in the meanwhile flexible. In the project, we adapt the same architecture in the paper [1]. There are 3 types of layers:

- Conv+ReLU: This layer has 64 filters of size $3 \times 3 \times c$, which are used to generate 64 feature maps, and rectified linear units (ReLU, max(0, $\cdot$)) are then utilized for nonlinearity. Here c represents the number of image channels, i.e., c = 1 for gray image and c = 3 for color image. This this the first layer.
- Conv: for the last layer, c filters of size $3 \times 3 \times 64$ are used to reconstruct the output.
- Conv+BN+ReLU: for all the middle layers, 64 filters of size $3 \times 3 \times 64$ are used, and batch normalization is added between convolution and ReLU.

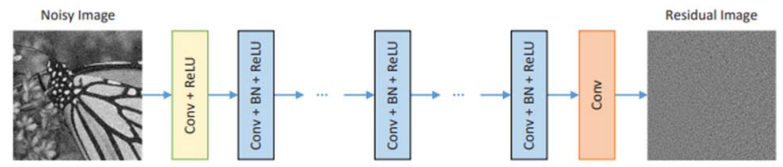


Fig. 1. The architecture of the proposed DnCNN network.

However, as mentioned in the Section 1, we do not necessary agree with the authors that all the noise are generated out of Gaussian distribution. We use the GED introduced in equation (1). More precisely, we choose 7 beta values that correspond to the different level of kurtosis, as shown in the table below.

Table 1: The selected β value

Kurtosis	-1	0	1	2	3	4	5
Beta	6.0	2.0	1.406	1.1483	1.0	0.902	0.831

Note that the relationship between Kurtosis and β is given as

$$Kurt = \frac{\Gamma(5/\beta)\Gamma(1/\beta)}{\Gamma(3/\beta)^2} - 3. \tag{2}$$

We know the kurtosis of Gaussian is 0, the kurtosis of uniform distribution is -1.2, and the kurtosis of t-distribution with degree of freedom 6 is 3. Therefore, our selection of β is wide enough to cover several typical random distributions.

For each β level, we train one CNN model. During the training of this model for given β level, the α is specified in such a way that the resulting standard deviation of the GED is randomly selected in the range of $(0, 55/255)$. Here we denote our models as β_1-model, ... β_7-model.

The reference characteristic that we use in training, validation, as well as testing is the Peak Signal-to-Noise Ratio (PSNR). In general PSNR in dB of a noisy image K and the clean image I is defined as

$$PSNR(K, I) = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{3}$$

where MSE is the mean square error between K and I, and MAX_I is a universal constant, maximum possible pixel value of the image I depending on how the pixels are represented. It is clear from the defintion of the PSNR that larger PSNR is desirable.

In the last step, a VGG-16 classification model is trained to predict the parameter beta in GED. For a given noisy image, the beta-level is the output of this classification model, and we select the corresponding model among the β_1-model, ... β_7-model to de-noise such an image.

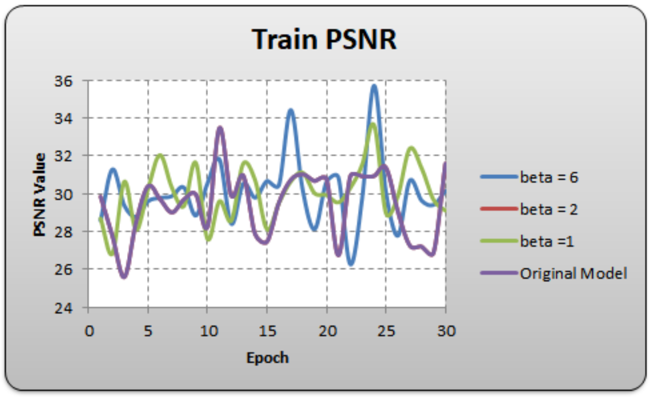
3 Experimental Results

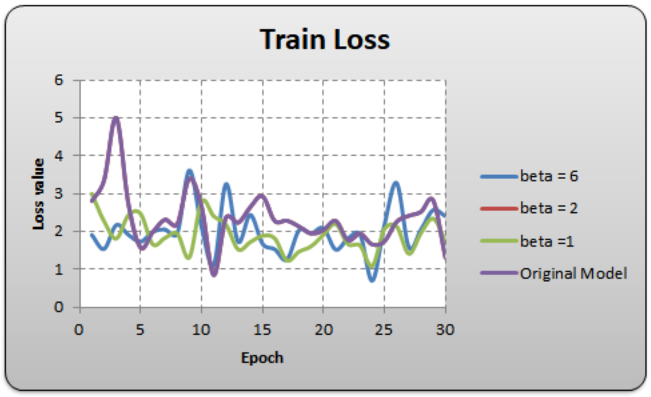
We note that currently the generalized Gaussian distribution is not yet well-suited for paralleled computation on GPU. As such, our data-preprocessing (i.e. adding noise to training dataset) is CPU based. Therefore, the following experiments are all based on 1/10th of the original dataset.

3.1 Comparison between different models

As noted in Table 1, the value of beta represents different level of kurtosis, out of the 7 models trained by us, we present here the representative case, which corresponds to Gaussian-Like noises, noises with fatter tails, noises with thinner tails.

In terms of the training PSNR and traing loss, our model is general better than the original model.



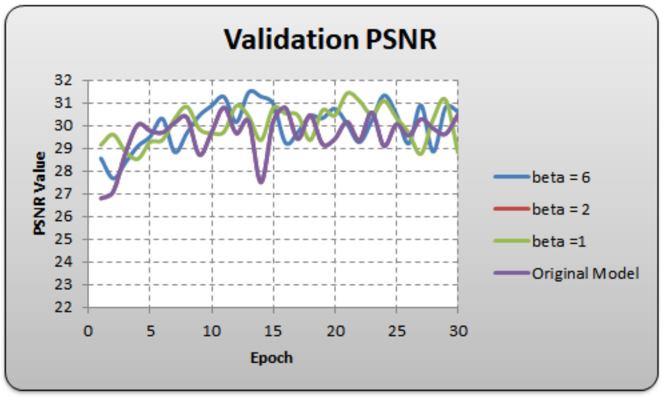


The table below shows that when we use our trained model on two different test datasets (the "dateset12" and "dataset68" as named by the original paper) with uniform noise, our model performs better.

Table 2: Average PSNR score on test data sets

	Our model beta = 6	original model
dataset12	29.8657	15.6502
dataset68	28.6719	8.1774

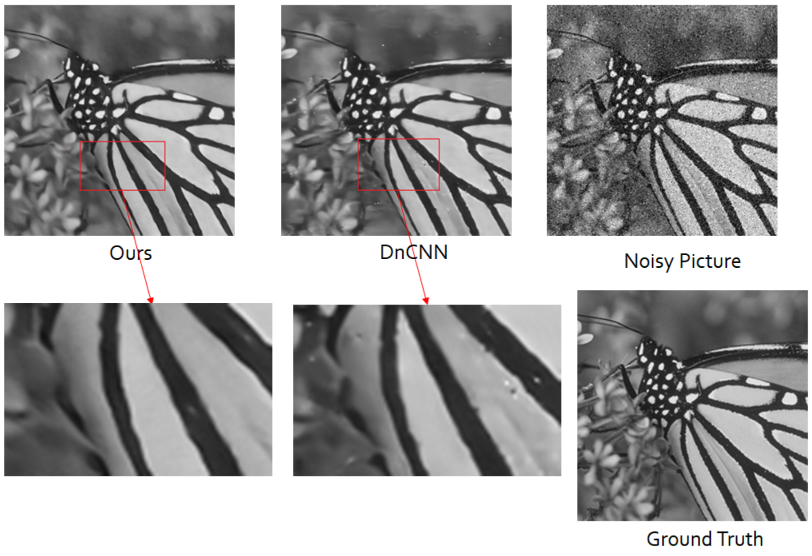
In terms of validation PNSR during the training, we can also tell that our model is in general better than the original model.



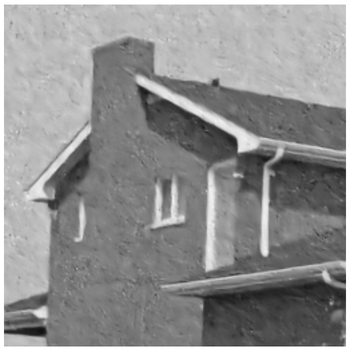
3.2 Outcome of sample images

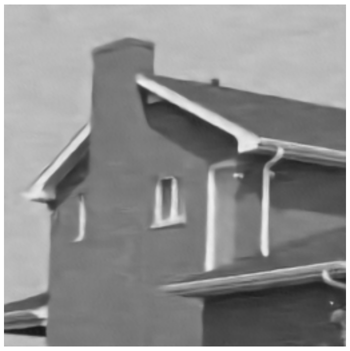
In this section, we visually compare our model with the original model.

We first tested the original model and our model with noise coming a heavy-tailed distribution with kurtosis 3. On the right we have the image with "ground truth" and the noisy image which is the ground truth image added with noise. On the left and on the middle, we compared the output of our model with the output of the original model. It can be seen that our model performs better when the noise has a Kurtosis higher than Gaussian. The baseline model from the orignal paper suffered from white spots in the denoised picture, while ours didn't have this problem.



Although our model is trained based on non-Gaussian noise, we would like to see the performance of our model on Gaussian noise given that Gaussian is the most common distribution. Because the original model is trained using Gaussian noise while our is not, it is to be expected that our model's output has less quality than the original ones. We note that the quality of our model's output is still reasonable.


(a) Our Model: Beta = 6


(b) Original Model

(c) Gaussian Noisy image

(d) Ground Truth

3.3 Two step approach

In the real world application, image comes with noise. Therefore, we won't be able to have the so-called "ground truth" that we see in section 3.2. It implies that the PSNR can not be calculated on the model output image. In the meanwhile, there are 7 models trained in Section 3.2. We need to know which model out of these 7 models need to be used. Our proposal for this problem is the two-step approach:

Phase one: VGG-16 classification model to predict the parameter beta in GED which indicates the Kurtosis of the noise in the picture. To train this classification model, we take the training sample and add GED noise with different levels of beta, which provides the labeled training sample. At this moment, the top-1 accuracy of this classification model is 55%. Although this accuracy is reasonable for multi-class classification problem, we believe this can be improved further.

Phase two: the residule learning model trained as in section 3.2 which is closest to the beta from Phase one is used to process the noisy image and output the clean image.

4 Discussions

Overall, in this project, we explored the possibility of extending the paper [1] to train CNN models capable of denosing images with non-Gaussian noise. Our conclusions can be twofold: For the non-Gaussian noise that follows the distribution not too far away with Gaussian in terms of kurtosis, our model can outperform the original model trained in [1], with the same training and testing dataset. For the non-Gaussian noise that are further substantially different with Gaussian, our trained model has relatively lower PSNR. This is due to the limitation of the CNN structure that proposed in [1]; we believe that to train a non-Gaussian denoiser, a new CNN archetecture needs to be explored.

We also wanted to remark that although PSNR is a convenient measure for model output quality, we shall not ignore the usefulness of visually checking model output picture. During our project, we noticed that two pictures can be very different visually while still have excellent PSNR score.

Lastly, as mentioned, we only used the 1/10th data during training due to computation power. On the other hand, GED distribution is a two-parameter family and requires more training sample than Gaussian Distribution. We believe using more training data will improve the performance of our model.

Acknowledgments

The authors would like to thank Prof. Rob Fergus for his valuable suggestions and discussions during the project.

References

[1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang, Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising, https://arxiv.org/pdf/1608.03981v1.pdf. TIP, 2017.

[2] https://github.com/SaoYan/DnCNN-PyTorch

[3] Nadarajah, Saralees, September 2005, A generalized normal distribution, *Journal of Applied Statistics.* **32** (7): 685–694.