Image Denoising using Deep Residual Convolutional Neural Network

Merlin Carson
Department of Computer Science
Portland State University, Portland, OR, USA
mpc6@pdx.edu

***Abstract*—Noise is a common problem in signal processing. This is especially noticeable in images. Noise is generally a sensing problem, but can also be caused by low lighting, lighting effects, or data compression. Noise can not only cause a depreciation of the quality of an image, but it can also interfere with many computer vision applications. Deep convolutional neural networks have recently been used to achieve state of the art results in image denoising. Using residual blocks I'm able to train a significantly deeper network than other algorithms. This achieves a nearly 33% improvement over other well known deep learning approaches.**

***Keywords*—image denoising, computer vision, deep convolutional neural network, residual learning**

I. Introduction

Traditional image denoising algorithms use a type of Gaussian smoothing [1]. These algorithms can leave blurry, blocking and ringing artifacts. More recently, intelligent algorithms have been developed that reduce the amount of these noticeable artifacts [2], [3]. Now state of the art image noise reduction is done with deep neural networks (DNN) [4], and deep convolutional neural networks (CNN) [5]. One of the most notable CNN used for image denoising is DnCNN [6]. This model works through residual learning, predicting the noise in an input image and then subtracting the prediction from the input to produce a denoised image. The DnCNN is a 20-layer deep fully convolutional neural network.

Researchers at Microsoft did tests to determine the effect of the depth of a neural network on error rates. They found that past about 20 layers the error rate begins to increase. So they developed what they called a residual learning block [7] and were able to successfully train a CNN that had more than 1200 layers. Ultimately they settled on a 152-layer CNN as the best trade off for accuracy vs. efficiency and were able to win the 2015 Image Net competition with their ResNet-152 network.

I would like to introduce the DnCNN-ResNet. Using the residual learning blocks from the ResNet, I was able to extend the depth of the DnCNN to an 82-layer CNN. With this model I am able to achieve an almost 33% increase in the peak signal to noise ratio (PSNR) improvement over the DnCNN on the same benchmark dataset used in the paper.

II. Methods

A. Pre-Processing

The DnCNN-ResNet model is trained on 50x50 overlapping patches from the images in the training set. A 50x50 window with a stride of 10 is passed over each image, see Fig. 1, in the dataset and saved as a Torch Tensor in an HDF5 file for fast access during batch training.

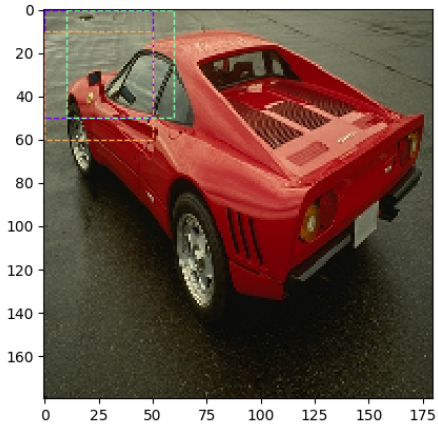

Fig. 1: Training data created from sliding an overlapping window across each image in the dataset to create patches.

B. Data Augmentation

Each image in the dataset has several types of data augmentation applied to increase the size and diversity of the training examples. Each image is scaled to 4 different sizes, 100%, 90%, 80%, 70%, as seen in Fig. 2. This allows the model to learn how to denoise features at different distances given the same size receptive field within the model.

Further data augmentation was done to each of the scaled images by mirroring, flipping, and 90° rotations in both directions, as seen in Fig. 3.

The patches for the training set were taken over each image with each type of data augmentation, producing over 800,000 examples given a 400 image dataset.

(a) 100%

(b) 90%

(c) 80%

(d) 70%

Fig. 2: Multiple scaling factors were applied to each image in the dataset to augment the training data.

(a) mirror

(b) flip

(c) rotate left

(d) rotate right

Fig. 3: Transformations to images in dataset to augment training data.

C. The Model

The DnCNN-ResNet is an 82-layer deep CNN. The input layer is a 2D convolutional layer with a ReLU activation. This is followed by 40 DnCNN residual blocks. Each block consists of a 2D convolutional layer followed by a batch normalization layer, followed by a ReLU activation, followed by a 2nd 2D convolutional layer, followed by another batch normalization layer. The output of each block is the summation of the output of the 2nd batch normalization layer and the input signal to the block, passed through a ReLU activation. The output of the model is a single 2D convolutional layer with a linear output. All convolutional layers consist of 64 3x3 filters with a stride of 1 and have a padding of 1 to maintain the height and width of the example throughout the model. In total the DnCNN-ResNet as 2.96 million trainable parameters.

III. Experiments

The model was trained on the Berkely segmentation dataset, BSDS500. This dataset consists of 400 clean images with a large variety of features and color palettes. The dataset was split into 3 parts, 380 for training, 10 for validation and 10 for testing. The validation set consisted of approximately 1,900 50x50 overlapping parches with no data augmentation. During training the PSNR of the validation set was taken to determine the best model and avoid overfitting.

To train the model, batches of 128 training examples were fed to it between each error backpropagation. This model was trained in blind mode, where varying levels of noise are added to the clean patches, across each batch. This is accomplished by applying a uniform distribution of noise across the batch. When training with additive Gaussian noise, the noise was sampled from a Gaussian distribution with a $\mu = 0$ and $\sigma = [0, 55]$. When training with uniform or pepper noise 0% to 25% of the pixels were corrupted. Examples of these noises at varying levels can be seen in Fig. 4. The model was trained using Mean Squared Error with the error calculated sample-wise, not pixel wise, between the predicted noise and the generated noise.

(a) Gaussian noise

(b) Uniform noise

(c) Pepper noise

Fig. 4: Different types of noises used to train the model.

The model was trained with an exponential learning rate decay of 0.91. This reduced the learning rate by 9% each epoch. The learning rate began at 0.01 to quickly find a good local minima and decayed to 1e-05 over 80 epochs, as can be seen in Fig. 5. This allowed the model to fine tune its search of the gradient of the error space as it got closer to convergence.

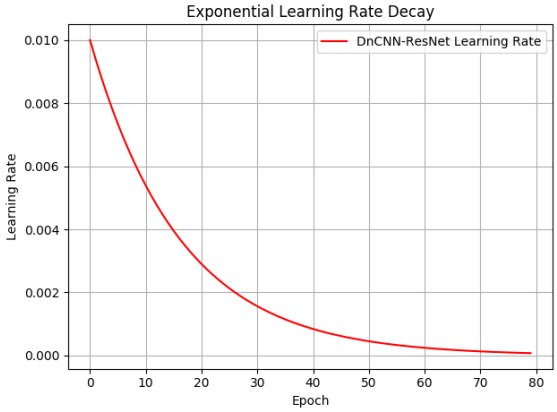

Fig. 5: Exponential learning rate decay used to train models.

Given 4 2080 Ti Nvidia GPUs, the model took approximately 30 hours to train to completion.

The best model achieved a PSNR of approximately 30.9 on the validation set, seen in Fig. 6. Overfitting appeared around epoch 37, so the model saved at that epoch was chosen as the best model. The Berkeley segmentation dataset CBSD68 was used to benchmark the model since this was what was used in the original DnCNN paper to compare it to other state of the art denoising algorithms. The comparisons can be seen in the table below along with several DnCNN-ResNet examples.

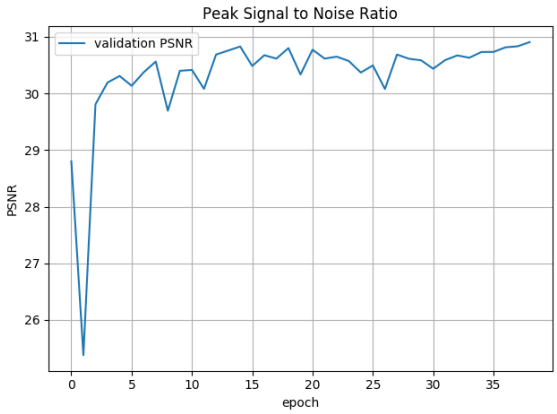

Fig. 6: Peak signal to noise ratio per-epoch during training.

Noise Level	BM3D	TRND	DnCNN-B	DnCNN-ResNet
$\sigma = 15$	31.07	31.42	31.61	33.91/+9.30
$\sigma = 25$	28.57	28.92	29.16	31.27/+11.09
$\sigma = 50$	25.62	25.97	26.23	28.03/+13.88

(a) Clean Image

(b) Clean Image with gaussian noise $\sigma = 25$

(c) Denoised image

Fig. 7: Results of denoising on image with distant features.

(a) Clean Image

(b) Clean Image with gaussian noise $\sigma = 25$

(c) Denoised image

Fig. 8: Results of denoising on image with near features.

(a) Clean Image

(b) Clean Image with 15% of pixels corrupted with pepper noise

(c) Denoised image

Fig. 9: Results of denoising on image with distant features.

(a) Clean Image

(b) Clean Image with 15% of pixels corrupted with pepper noise

(c) Denoised image

Fig. 10: Results of denoising on image with near features.

IV. Conclusion

By adding residual blocks to the DnCNN architecture, I was able to successfully train a significantly deeper model. With this model, the DnCNN-ResNet, I was able to achieve almost a 33% PSNR improvement over the original DnCNN. The model also showed remarkable results when trained on pepper noise, successfully differentiating between small shadows in the original image that resemble this type of noise and the actual additive noise, see Fig. 10. Future experiments will involve super resolution, since this can also be modeled as a subtractive problem. Images can be downsampled, then resized back to their original size to create blocky representations. Then the model can be given these lower quality images and optimized to predict the difference between the lower quality image and the original image. Thus, when the model correctly predicts this difference, it can be subtracted from the input image to produce the original image. Therefore the model would be able to not only denoise and image, but also upsample low quality images with the same operation.

Acknowledgement

I'd like to thank my professor Feng Liu for his instruction and helpful advice during this project. I'd also like to thank teuscher.: lab for the use of the training equipment.

References

[1] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Adaptive wiener denoising using a gaussian scale mixture model in the wavelet domain," in *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, vol. 2, pp. 37–40 vol.2, , 2001.

[2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D Image Denoising with Shape-Adaptive Principal Component Analysis," in *SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations*, R. Gribonval, Ed. Saint Malo, France: Inria Rennes - Bretagne Atlantique, 2009.

[3] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *CoRR*, vol. abs/1508.02848, 2015.

[4] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2392–2399, , 2012.

[5] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[6] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.