Multi-Level Wavelet-CNN for Denoising Natural Noise Benchmarks

Arjun Arora
Stanford
450 Serra Mall, Stanford, CA 94305

aarora52@stanford.edu

Abstract

Denoising images is a common application of optimization algorithms. In this course we primarily covered methods for solving the deconvolution problem by modeling the inverse of the transfer function that produced the current noisy output. Generally, these methods rely on strong priors (TV, NLM, etc) in order to produce useful results. Neural networks on the other hand, require the use of a dataset and architecture, but are known to be more generalizable than purely convex methods. Neural networks of sufficient depth are known to be universal function approximators, which gives them a strong representational power for denoising tasks [1]. In this work we use one such network on a variety of different noise types to explore the effects of natural noise models and the efficacy of the Multi-Level Wavelet-CNN to denoise them.

1.Introduction and Motivation

U-Nets, a particular type of neural network with contracting and expanding networks, have been shown to do well in denoising tasks, specifically in biological scenarios [3]. The model used in this work is one such variant of the U-Net, the Multi-level Wavelet-CNN. To reduce parameter size and to extract more information, this model replaces the typical downsampling and upsampling layers of the U-Net with the Discrete Wavelet Transform and Inverse Discrete Wavelet Transform [4]. These transforms are much less destructive than a normal bilinear downsampling. However, compared to the typical Conv-Transpose layers used to upsample they do not require any parameters and do not produce dilated artifacts. Finally, while these models have proven to be useful in denoising images with gaussian white noise, such noise is not the only type of noise one faces with when using cameras. As we have discussed in this course, both shot noise and fixed pattern noise are common types of noise that cannot be modeled by a simple independent gaussian. To model these types of noise, one must take into account the image statistics.

2. Related Work

Deep learning models, and in particular U-Nets, have long been used for denoising various types of images [5]. Due to the inherent size of the parameter spaces, deep learning models can create transforms that are much more non-linear than simple matrix priors even with non-linear regularization schemes, as we have presented in class. U-Nets, in particular, allow the models to explicitly learn high level and low level detail. In the compression network, the model is incentivized to learn fine feature detail in the first depth, and subsequently coarser information as the images are fed into the subsequent depths. This is primarily due to the downsampling operations that the U-Net uses. This inherently allows the model to focus on higher level semantics of the image, similar in style to auto-encoder networks that learn dense embeddings from high level representations [1]. The expanding network focuses on rebuilding detail from the lowest depth back up to the first depth. These layers, along with the skip connections that provide feature information from the contracting networks on the respective depth, help the model learn a type of “non-linear” upsampling. In this way we can go from a dense feature representation back towards the sparse one without losing information. In comparison with a conventional DCNN, U-Nets provide an inbuilt hierarchical prior that must be learned in other CNN based models.

However, the common trend of these denoising papers is that they model all noise independent of the images in their dataset. We find this to be insufficient in modeling the entirety of noise found in the output image of a DSLR, which is the primary use case for these denoising models. For these reasons, we propose the following experimental method.

3. Methods

In this work, we trained and tested our model on a supervised dataset consisting of a mix of three different image datasets: the Berkeley Segmentation Dataset, DIV2K, and the Waterloo Exploration Database. This

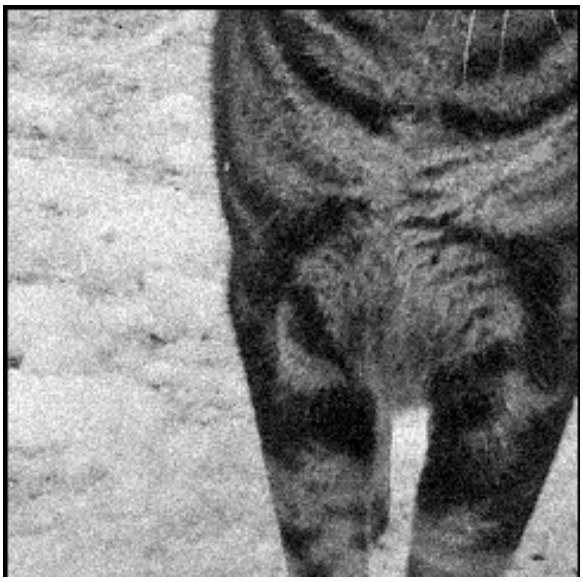
follows the exact dataset that Liu et al [3] used in their paper. The images found in these datasets were of various file types, from JPEGs to PNGs, which meant that some of the images in the dataset were already lossy images. Moreover, the images ranged in size from 240 x 240 pixels up to 2000 x 2000 pixels. Some images in the dataset were in color, while others were already in grayscale. This diversity in image quality, size, and color is representative of the wide quality of images that may be passed through our model.

Each image in this dataset (~5700 images) was broken up into a maximum of twenty-four 240 x 240 pixel patches and converted to grayscale. This allowed our full dataset to be ~137k images. However, due to GPU/RAM constraints on our testing machine, we could only load in approximately 24000 images as our dataset for training, validation, and testing. The dataset was split into a training, validation, and test set with a ratio of 8:1:1 respectively. The entire model was trained on a GTX 1070 GPU using Cuda and PyTorch 1.0.1.

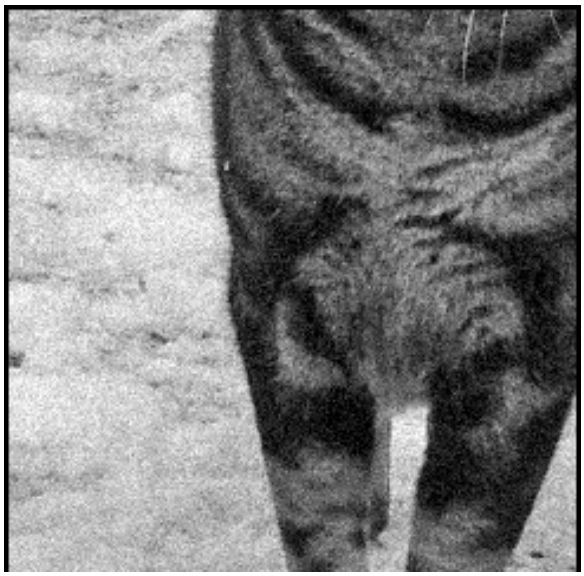
3.1 Noise Models

In our experiments, we trained the model for 20 epochs on three different types of noise: pure gaussian noise, gaussian noise + image dependent noise, and impulse noise. These noise models correspond to random noise, shot noise + random noise, and salt and pepper noise found in commercial DLSRs, respectively. Random noise is the easiest to model since it simply follows the standard gaussian distribution. A mixture of random noise and shot noise is a bit more involved since these types of noise depend on statistics from each image.

1. Gaussian Noise:
   $Y_i = X_i + N(0,\sigma^2)$

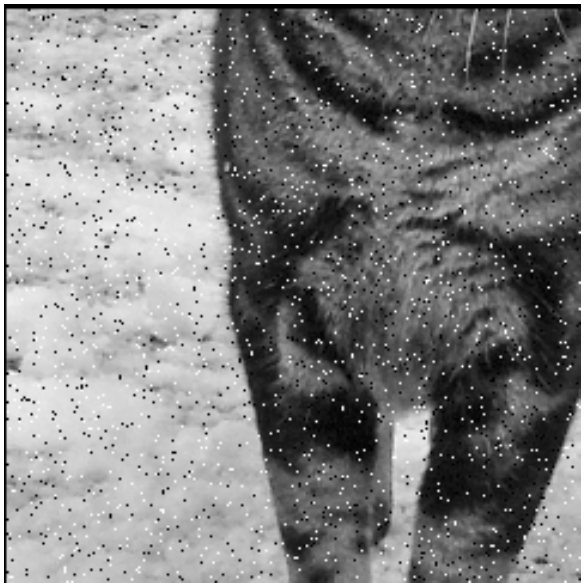


2. Gaussian Noise + Image Dependent Noise:
   $Y_i = X_i + N(0,\sigma^2(X_i)_j)$



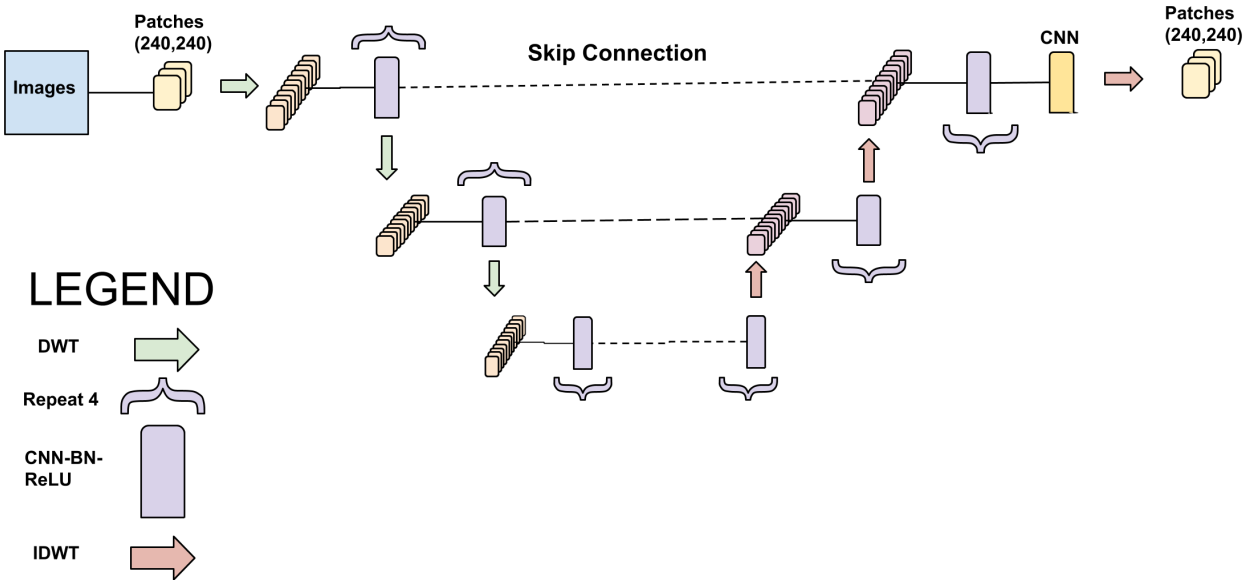
3. Impulse Noise:

   $Y_i = X_i[(X_i)_j > 0.5]_{j \in P}$



Where X_i is the i^{th} clean image in the dataset, Y_i is the output image and $N(\mu, \sigma^2)$ is the gaussian with mean μ and variance σ^2. $(X_i)_j$ is the j^{th} pixel in X_i. P is the set of pixels to be set to 0 or 1. 4% of the pixels per image were randomly chosen to be part of set P. The value of σ chosen for the first two noise models was ~6%.

3.2 Multi-Level Wavelett CNN



The Multi-level Wavelet-CNN, implemented using Liu et al [3] as a guide, is essentially broken into 4 main components, the Discrete Wavelet Transform, the Convolution-BatchNorm-ReLU meta-layer, the skip connection, and then Inverse Discrete Wavelett Transform.[1]

The discrete Wavelet Transform uses the Haar Wavelets convolved with the input image I to produce 4 subband images: I_1, I_2, I_3, I_4.

$$I_1 = (f_{LL} * I) \downarrow_2, I_2 = (f_{LH} * I) \downarrow_2, I_3 = (f_{HL} * I) \downarrow_2, I_4 = (f_{HH} * I) \downarrow_2$$

Where $f_{x,y}$ are the convolutional kernels, * is the convolution operation, $\downarrow_2$ is the downsampling operation.

$$f_{LL} = \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix} \quad f_{HL} = \begin{matrix} -1 & 1 \\ -1 & 1 \end{matrix}$$

$$f_{LH} = \begin{matrix} -1 & -1 \\ 1 & 1 \end{matrix} \quad f_{HH} = \begin{matrix} 1 & -1 \\ -1 & 1 \end{matrix}$$

While the DWT inherently uses a downsampling operation in its construction of the subband images, due to the inherent bi-orthogonality of the transform, the IDWT can reconstruct the original image almost perfectly with no degradation of signal quality

The Inverse Discrete Wavelett Transform takes 4 of these downsampled images and reconstructs the original image:

$$I = IDWT(I_1, I_2, I_3, I_4)$$

The Convolution-BatchNorm-ReLU layer can be understood as a cascaded series of operations, where the Convolution layer (Conv) is a standard 2-dimensional convolution with a 3 x 3 kernel, with stride of 1 and a padding size of 1.

$$Output = bias(C_{out_j}) + \sum_{k=0}^{C_in-1} weight(C_{out_j}, k) \star input(I, k)$$

Where $\star$ is the 2D cross correlation operation. The BatchNorm (BN) layer is implemented in the standard method of

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

The ReLU layer is also implemented in the standard method. The skip connection simply appends the feature maps across from each contracting layer to its corresponding expanding layer.

[1] Both the DWT and IDWT were implemented using a third party repo by fbcotter: https://github.com/fbcotter/pytorch_wavelets

After the initial DWT, the output subband images are passed through four convolutions, Batchnorms, and ReLU layers each before being passed through another DWT layer. This pattern is repeated two more times to create an overall model depth of 3. The outputs of this last depth are then passed along to the expanding network. The skip connections also concatenate the feature maps of the lower depth and the corresponding contracting network to create a feature stack that is twice as large as the contracting networks'. For our model, the number of channels for the convolutions on the first depth was set to 16 and 32 for the other two depths. We then apply a final CNN with a single output channel to produce our finalized output patches.

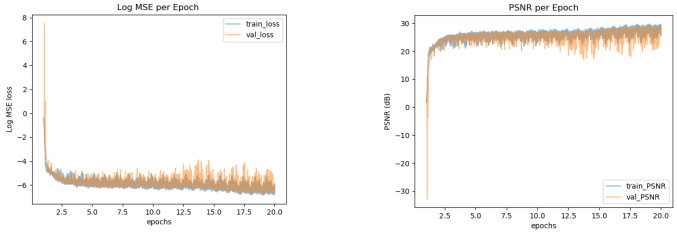
We used a supervised loss metric in Mean Squared Error Loss to train our model:

$$L(X_i, I_i) = ((X_i)_j - (I_i)_j)^2 \text{ for } j = 1,..,n$$

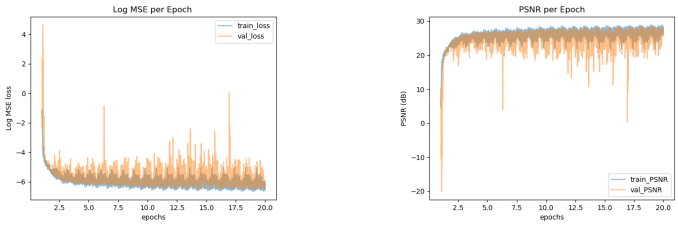
Where X_i is the original image and I_i is the output of our model. The optimizer for our model was ADAM with betas = (0.99,0.9999) and lr = 0.001.

The model was then evaluated using both PSNR and SSIM.

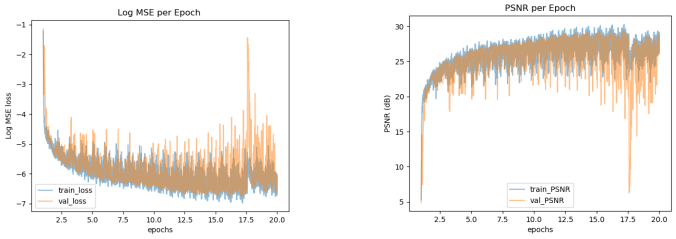
4.Analysis and Results



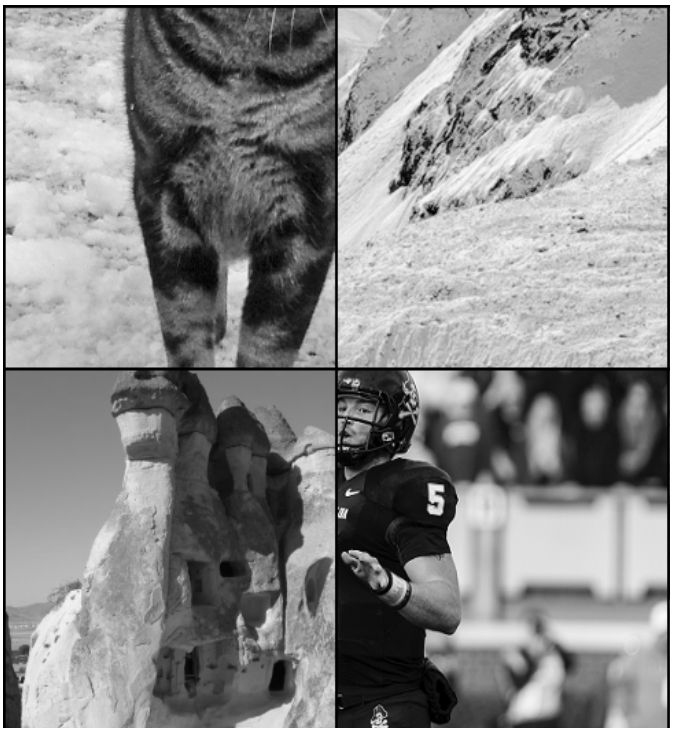
Train Loss and PSNR for Gaussian Noise Model



Train Loss and PSNR for Gaussian + Dependent Noise Model



Train Loss and PSNR for Impulse Noise Model



Original Clean Images

Gaussian Noise Model Outputs

Gaussian + Dependent Noise Model Outputs

Impulse Noise Model Outputs

Noise	PSNR	SSIM
Gaussian Noise	28.08	0.855
Gaussian + Dependent Noise	29.20	0.858
Impulse Noise	28.72	0.893

Liu et al [3] results for noise level of ~6% for pure gaussian noise:

PSNR = 33.15, SSIM = 0.9088

The results procured from our tests achieve ~85% as good results across all the three noise models. Though, it should be stated that our model was only trained for 6 hours compared to Liu et al's 24 hours, and utilized a training set ~1/6th the size.

6. Discussion

Overall, the results from our model are quite satisfactory. Though our PSNR and SSIM numbers are not as high as Liu et al predicted, they are well within a satisfactory range given the limitations of our hardware. Given that our channel dimensions were 1/6th the size of the original paper's, we still produced meaningful results. From the training results available we can see that the models trained in a fairly standard pattern, which consisted of quick drops in both our loss as well as our evaluation metric within the first couple of epochs. Afterwards, the training slowed logarithmically in the final 10 or so epochs. However, as we can see from the loss and PSNR curves of both our training and validation set, the model continued to optimize even during the last epoch. This gives us some reason to suspect that the models could potentially do even better if given a longer training time.

Before running our experiments, we believed that the dependent noise model would produce the best results in terms of PSNR and SSIM compared to both the pure gaussian noise and impulse noise models. This is because there is an inherent correlation between noise and pixel values in the image with the dependent noise model. The deviation of the gaussian in the dependent noise model is partially determined by the pixel intensities. In contrast, there is no such correlation with the pure gaussian noise or impulse noise. These models independently produce noise without regard to the intensities of the pixels.

This allows the dependent noise model to learn a more reliable mapping from noisy image to clean image. Pure gaussian noise and impulse noise contain no information about the image and must be simply subtracted from the image using a bias term, which may be harder for the model to learn than performing a transformation on the dataset using a matrix or activation.

In the end, the dependent noise model had the highest PSNR, though the results for the other two models were closer to the dependent noise model than what we expected. The most surprising result was that the impulse noise model actually produced a better PSNR result than the gaussian noise model and achieved 98% of the results achieved by the dependent noise model. Given that the impulse noise model simply turns pixels white or black, it seems like doing so would lose whatever information that pixel contained. But, perhaps since the model was trained on grayscale images, these white or black pixels were not as much of a challenge as the distortions caused by the independently distributed gaussian.

The SSIM results were even more surprising as the SSIM for the impulse noise model was 5% higher than either the gaussian or dependent noise models. This may be due to the fact that the artifacts produced by the impulse noise model are more spread out/smaller than the artifacts produced by either of the other two models.

7. Future Work

If we were to work on this model moving forward, one of the first steps we would take would be to ensure that the model has adequate hardware so that we can load our full sized model onto GPU memory without dealing with complaints of memory shortage. We would also ensure we have enough RAM to deal with the amount of images we need to load off and onto the GPU without crashing, which was a significant hurdle and forced us to use the limited training set time. Of course having enough time to train each model for the full day to accurately compare to the original Liu et al paper would be invaluable. It would also be interesting if we could test the other noise values Liu et al tested for their model and be able to report the results of nine different hyper parameters rather than the current three.

Moreover, given the utility of adversarial learning in the denoising process, as explained by Qinsong Yang et al [6], it may be useful to train the model with adversarial learning alongside the standard supervised approach. Another possible GAN denoising method may be to incorporate Cycle consistency into our network. [7] Instead of simply learning a mapping

$$F(X) \rightarrow Y$$

we enforce the network to be cycle consistent, i.e.

$$G(F(X)) \approx X$$

This forces the model to learn not only the forward mapping (from noisy image to clean image) but also the inverse mapping (from clean image to noisy image). This inverse mapping is actually the goal of the convex optimization solutions we used in class.

This may in fact help with interpretability of the model, since when we train this model we can not only determine the methods the model uses to produce a clean image, but also how the "noisy" kernel is reproduced. In a way, this gives us an idea of what the "deconvolutional" function may look like.

8. References

1. Baldi, Pierre. "Autoencoders, unsupervised learning, and deep architectures." *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012.

2. Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." *Neural networks* 2.5 (1989): 359-366.

3. Heinrich, Mattias P., Maik Stille, and Thorsten M. Buzug. "Residual U-Net Convolutional Neural Network Architecture for Low-Dose CT Denoising." *Current Directions in Biomedical Engineering* 4.1 (2018): 297-300.

4. Liu, Pengju, et al. "Multi-level wavelet-CNN for image restoration." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.

5. Zhang, Kai, et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising." *IEEE Transactions on Image Processing* 26.7 (2017): 3142-3155.

6. Yang, Qingsong, et al. "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss." *IEEE transactions on medical imaging* 37.6 (2018): 1348-1357.

7. Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

Gaussian Noise Model Outputs