Generative Adversarial Networks for Electron Miscroscope Image Denoising

Chuan Chen*
cc6580@nyu.edu
Center for Data Science
New York University
New York, NY 10012

Michael Stanley*
mhs592@nyu.edu
Center for Data Science
New York University
New York, NY 10012

1 Introduction

Image denoising has always been a problem of interest in experimental sciences. Due to hardware limitations, images obtained from equipment need to be cleaned and denoised before further scientific analysis can be conducted. This issue becomes even more of a problem when working on the microscopic scale. For our project, we aim to denoise electronic microscope images of atomic structures. Traditional denoising methods use mean-squared error as a loss function, but due to its averaging nature, the resulting images often have problems such as blurriness or exhibiting phantom artifacts [11]. To overcome these issues, we propose a generative adversarial network (GAN) based denoising method in which we first train a GAN to generate realistic microscope images, then optimize over the latent space to find a clean image that most resembles the noisy image we were given. Since well-trained GANs could only generate reasonable images, we hypothesize that our method can help avoid the problems brought by traditional denoising methods.

2 Related Work

2.1 Generative Adversarial Networks

Generative Adversarial Networks are a class of models that learn the training data's distribution to generate new samples from that same distribution. Since their introduction by Goodfellow in 2014, GANs have gained much attention in the field of machine learning [5]. GANs are comprised of two separate models, a generator and a discriminator. The generator model aims to generate fake images that look like the real training images. The discriminator model aims to distinguish the fake images that outputted by the generator from the real images of the training set. During training, the generator weights are updated to fool the discriminator and to generate more realistic-looking fake images, while the discriminator weights are updated in the aims of correctly classify the real and fake images. Ideally, training should reach an an equilibrium when the generator is creating images that look as if they came directly from the training data, and the discriminator always outputs a 50% confidence that the generator output is real. Although, true convergence is rarely achieved during actual training, and GANs have shown good results without perfect convergence.

In 2015, Radford et al. proposed an extension of GANs by combining them with deep convolutional networks, where they explicitly used convolutional layers in the generator and convolutional-transpose layers in the discriminator [14]. They termed their model Deep Convolutional Generative Adversarial Networks (DCGAN). This extension led to a surge in the application of GANs on natural image processing. The paper conducted extensive empirical trials with varying model designs, configurations, and training schemes, and their approach to training and designing the GAN has become the de-facto standard in the area for stable GANs and provided the inspiration for our GAN architecture.

*Authors are listed alphabetically and contribute equally to the paper.

For their implementation, they have trained a DCGAN which generates $3 \times 64 \times 64$ RGB natural images from a latent vector z that is randomly drawn from the standard Gaussian. The success of their model has led to its use in several different applications, with AI-generated faces being one of the most popular. The image below shows the training images for the DCGAN on the left and the generated images on the right.

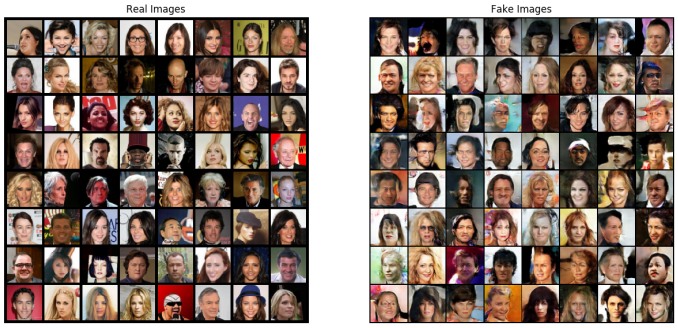

Figure 1: real and generated images of DCGAN

Although by the human eye, unnatural features in the generated images can still be noticed, the generator largely captured important features such as facial structures and hair placements.

The success of DCGAN largely motivated our hypothesis of GAN-based denoising. Although electronic microscope images have much larger dimensions in our study, 512×512 as opposed to 32×32 and 64×64 for traditional GAN implementations, they typically contain less complex features than natural images. Thus, training a GAN for our purposes seems possible with a few modifications.

2.2 Image Denoising

The classic approach to signal denoising is the Wiener filter [16], which assumes a stationary signal, stationary additive noise, and known spectral characteristics of both. In the 1990s, wavelets were introduced to map a noisy signal to a sparser representation which can then be thresholded to remove small coefficient "noise" components. Both approaches are still widely used in signal processing, but neither is learning based: they cannot benefit from training data.

Since 2015, the dominant approach to image denoising has been learning based models. Leveraging the deep convolutional neural network (CNN) architectures introduced in [9] for denoising, models such as DnCNN [17] achieved state of the art performance in denoising. Most CNN denoising methods, however, do not generalize outside of the noise levels that were present during training. [11] shows that bias terms in denoising CNN architectures can overfit to the noise level, and introduced a bias-free denoising model (BF-DnCNN) that generalizes to noise levels unseen during training.

2.3 Generative Models for Inverse Problems

Inverting generative models (from the image space to the latent space) remains an open research problem. [3] introduced a gradient based method that finds, for a target image, the latent vector that maps to an image very similar to the target image. Formally, the method seeks to find the z_{opt} that:

$$z_{opt} = \arg\min_{z} \|I - G(z)\|_2^2 \tag{1}$$

where I is the target in the image space, z is a randomly initialized vector in the latent space, and G is the generator mapping the latent space to the image space. Gradient descent methods can be utilized to solve this optimization problem. Eq(2) is not necessarily convex, so a global minimum is not guaranteed.

[10] first applied this GAN optimization method to image denoising, utilizing a trained DCGAN model on human face images. The approach is quite successful, though only tested on small images of 32x32 pixels. No literature exists (to our knowledge) of applying GAN latent space gradient

optimization images larger than 32x32, or outside of natural images. [1] applies a similar method to compressed sensing of 64x64 pixel natural images.

3 Problem Definition and Algorithm

3.1 Task

The problem addressed in this paper is electron microscope image denoising. Given a noisy image, I_{noisy}, as input and the corresponding clean image, I_{clean}, the desired output is an image $I_{denoised}$ that is as similar as possible to I_{clean}. For simulated data, I_{clean} is known and the noise model is controlled. For actual microscope images, I_{clean} is not available and the noise model is not known.

In order for this approach to work, it is vital that we have a generator that could actually be capable of generating a variety of different realistic images so that a close clean image can always be found no matter what noisy image was given. Without a good generator, we would be unable to find the optimal latent vector no matter how good the optimizer design is. Thus, our first task is to train a well-performing generator model that takes in standard Gaussian random vectors and returns a variety of realistically looking microscope images.

To utilize the trained generator for denoising, we perform gradient optimization in the latent space to minimize reconstruction error in the image space, as introduced in [3]. Specifically, the denoised output image $I_{denoised} = G(z_{opt})$ where G is the trained generator and z_{opt} is defined:

$$z_{opt} = \arg\min_{z} \|I_{noisy} - G(z)\|_2^2 \tag{2}$$

While the optimization step seeks to minimize reconstruction error with respect to I_{noisy}, the true task is to minimize the reconstruction error $\|I_{clean} - I_{denoised}\|_2^2$. However, I_{clean} is not available during training. A central hypothesis of the denoising approach presented here is that $G(z_{opt})$, the image closest to I_{noisy} in the range of the GAN, is very similar to I_{clean}. This hypothesis will be evaluated below.

3.2 Algorithm

3.2.1 GAN Architecture

The design of our GAN architecture was largely inspired by Radford's DCGAN paper [14]. Shown in figure 2, our generator model consists of a series of deconvolutional layers that takes an input random latent vector and outputs a 512×512 image.

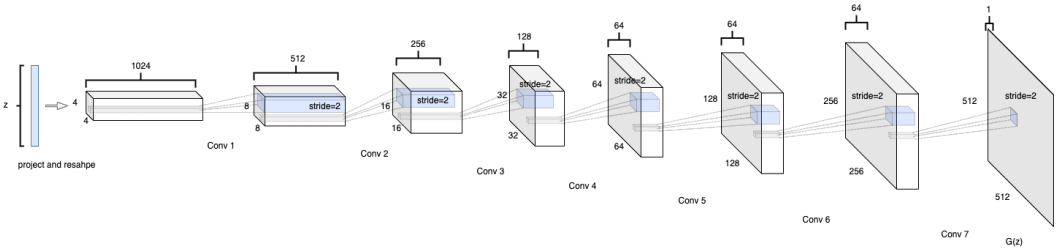

Figure 2: Generator Architecture

Our disciminator architecture uses a series of convolutional layers that takes in a 512×512 image and outputs the probability of the input image being real. As suggested by the trials from the DCGAN paper, pooling layers were not used to downsample or upsample the images. Instead, strided convolutions were used. This allows the network to learn its own spatial sampling. Batch normalization was applied at each layer in both models to help stabilize the training process, except for the output layer of the generator and the input layer of the discriminator [14]. Through the DCGAN paper findings, ReLu activation was used in the generator except for the output layer which uses the Tanh function to return it to the input data range of $[-1, 1]$. Leaky ReLu was used in the discriminator with the Sigmoid function for the output layer.

The 2 models work together during the GAN training. The Binary-Entropy loss function is used to update the model weights

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -\left[y_n \cdot \log x_n + (1 - y_n) \cdot \log 1 - x_n\right] \tag{3}$$

which provides the calculation of both log components in the objective function, $\log(D(x))$ and $\log(1 - D(G(z))$. We specify what part of the BCELoss equation to use with different y inputs during training. Two separate Adam optimizers were also set up, one for the generator G and one for the discriminator D. The entire GAN training progress is presented in algorithm 1.

Algorithm 1: GAN training

for *each epoch* **do**
  // update D: train with real batch
  output = nedD(real-images)
  calculate loss $\ell(output, 1)$
  // update D: train with real batch
  noise = a random latent vector
  fake-images = netG(noise)
  output = neD(fake-images)
  calculate loss $\ell(output, 0)$
  add the 2 errors together
  update netD weights
  // update G
  output = netD(fake-images)
  calculate loss $\ell(output, 1)$
  update netG weights
  for *every 50 epochs* **do**
    generate fake images on fixed random vectors
  end
end

Algorithm 2: GAN optimizer

Input: Noisy image I_{noisy}, Generator G() accepting as input $z \in N(0, 1)^d$

for *i = 1 to 10* **do**
  Initialize $z_i \in N(0, 1)^d$
  for *epoch = 1 to* m_{epochs} **do**
    Generate image $I_G = G(z_i)$
    Calculate Loss $\ell(I_{noisy}, I_G) = \|I_{noisy} - I_G\|_2^2$
    $z_i \leftarrow z_i - \alpha \nabla_z \ell$
  end
end
Select $z_{opt} = \arg\min_z \|Im - Im_{G,i}\|_2^2$
Reset epoch = 0
while $(epoch < n_{epochs})$ & $(\ell > \epsilon)$ **do**
  Generate image $I_G = G(z_{opt})$
  Calculate Loss: $\ell(I_{noisy}, I_G) = \|I_{noisy} - I_G\|_2^2$
  $z_{opt} \leftarrow z_{opt} - \alpha \nabla_z \ell$
  $epoch \leftarrow epoch + 1$
end

Output: $I_{opt} = I_G$

3.2.2 Optimizer Algorithm

The optimization problem seeks the minimum defined in Eq(2). Eq(2) is not a convex problem, so the optimization algorithm is susceptible to finding local minima. To mitigate this concern, the algorithm is composed of two optimization phases. In the first phase, multiple latent vectors are randomly initialized and a small number of optimization steps are performed on each. In the second phase, the latent vector with the lowest reconstruction error is optimized further. While not theoretically guaranteed, this approach was found empirically to arrive at similar denoised images over multiple trials with only moderate, incremental computational expense. Algorithm 2 describes the approach in detail. Figure 12(b) illustrates the sampling and selection aspects of Algorithm 2.

Notably, the GAN optimization approach presented here does not make any assumption about the noise mechanism in images. This is in contrast to supervised approaches that are trained to operate on a specified type and level of noise.

3.2.3 Baseline: Supervised Denoising Base

We compare the performance of the GAN denoiser with the supervised denoiser of [12]. This denoiser is trained on the same simulated images as the GAN (described below) with a Poisson(1) noise level. The denoiser architecture follows the UNet architecture [15]. The model was provided with pre-trained weights by the authors, although the provided model was trained to generalize to real microscope images (via training data distortions), which likely hindered performance on the simulated data.

4 Experimental Evaluation

4.1 Data

4.1.1 Microscope Data

Three thousand simulated grey-scale electronic microscope images of shape 876×927 were used to train the Generative Adversarial Network [12]. The pixel values of the input images are in the range of $[0, 1]$, and are re-scaled to $[-1, 1]$ since it has showed in related literature to help with training [14]. Since the Generator is designed to output 512×512 images, random 512×512 patches were cut from the input images. As opposed to resizing the images which might cause loss of information and skew the atomic structure of the atoms, cutting random patches avoids these issues and increase the robustness of the model. Figures below shows an example of the original input image and the random patches cut from the input image.

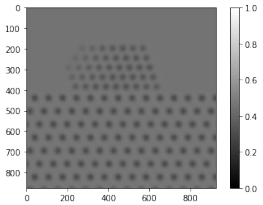

Figure 3: a random training sample

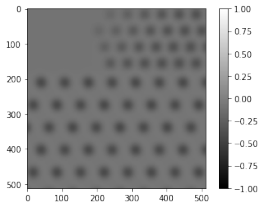

Figure 4: a randomly cut patch

The generated patches were scaled back to its original range of $[0, 1]$ for denoising optimization.

Simulated electronic microscope images that were not in the training set for the GAN were used for denoising purposes. A 512×512 patch was cut from each image, and noises from both the Poisson and Gaussian distribution were added with different noise levels. Figure 5 shows an example of a image patch with different noise levels.

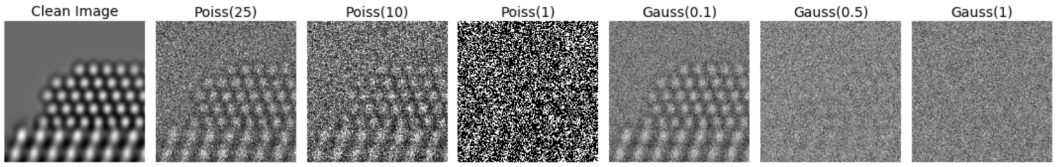

Figure 5: image with different noises added to be used for denoising

The real microscope data comes in video format with 40 frames of images with size 1215×1208 pixels. The images used for denoising herein are randomly cropped 512×512 pixel images of stills from that video with no additional scaling.

4.1.2 Natural Image Data

The optimizer algorithm was initially tested using the well-studied DCGAN generator on natural images before applying to microscope data. 50,000 natural images of size 32x32 pixels were randomly chosen from CIFAR100 [8] and used to train two DCGAN models: one model with rgb color channels and one model in grayscale. Both models used all 50,000 images for training.

4.2 Methodology

4.2.1 GAN Training

Generators were trained over 3 different latent dimensions of 50, 100, and 150 to generate the 512×512 microscope image. Both the generator model weights and the discriminator model weights were initialized from a Normal distribution with mean=0, stdev=0.02 [14]. The Adam optimizer [7] with a learning rate of 0.0002 and a momentun of 0.5 was used for both models for weight updates

An initial training of the GAN did not yield satisfactory results. GANs are notorious for being hard to train since both the generator model and the discriminator model are trained concurrently in a zero-sum game. This means that improvements to one model come at the cost of the other. Since there is no objective loss function, there is no way to objectively assess the progress of the training and the quality of the model purely by the value of the losses. Thus, the training of the entire network involves finding and arriving at an equilibrium between the two competing goals, which we can see from figure 6 did not happen with our training since our losses failed to converge.

Figure 6: losses from the initial GAN training

The lack of an objective loss function also means that the training progress must be inspected using the quality of the generated images at each step. The objective evaluation of the generator remains an open problem in the field, and although many methods such as kernel estimations [14] have been explored, many of them showed little to no relevance with model performance [2]. Thus, many fall back to manual inspection of images through out the training process, which remains one of the most common and intuitive ways to evaluate GANs. Shown in figure 7, the initial trianing of the GAN did not learn to generate any atomic structures, only the vacuum background, which confirms the bad model performances we expect from observing the loss curves.

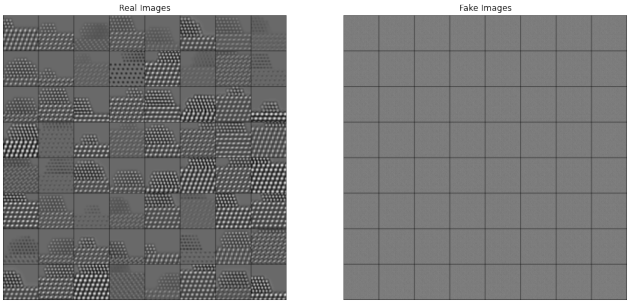

Figure 7: generated images from the initial GAN on the right

When the 2 losses of the GAN failed to converge, the model enters what's called the failure mode. Failure mode typically happens when the discriminator gets too good too fast. So it will always be able to distinguish between generated and real samples, giving it a loss of 0, and which will mean that there's is no more loss gradients flowing, and the weights stop getting updated for the entire network. To promote a stable training process, we have imposed the following training scheme modifications.

4.2.2 GAN training modifications

The first and most intuitive modification we made was re-designing the discriminator architecture to be symmetric to the generators, as a similar number of parameters in the model should provide a even ground for the competition. Figure 8 shows an updated discriminator structure.

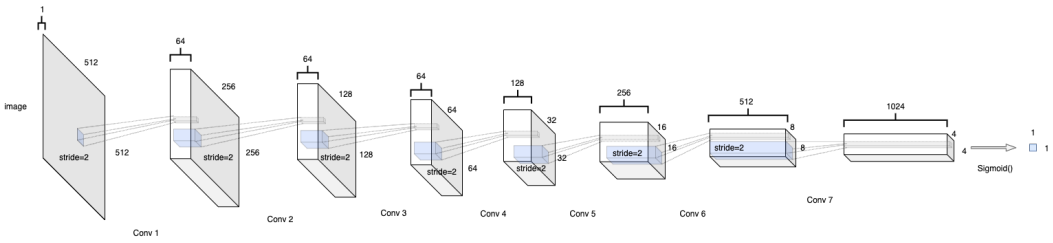

Figure 8: updated discriminator architecture

Then, we removed the last `Sigmoid()` activation function from the discriminator, and used `BCEWithLogitsLoss()` instead of `BCELoss()` as our criterion to take advantage of the log-sum-exp trick for numerical stability [13]. To discourage the discriminator from getting too good at the start of the training, we incorporated label smoothing where we set the label to the real images to 0.9 as opposed to 1. Label smoothing is a technique to discourage the discriminator from being over confident about its classification [4]. After these modifications, we noticed that the GAN losses have started converging, and the images generated look realistic compared to true microscope images.

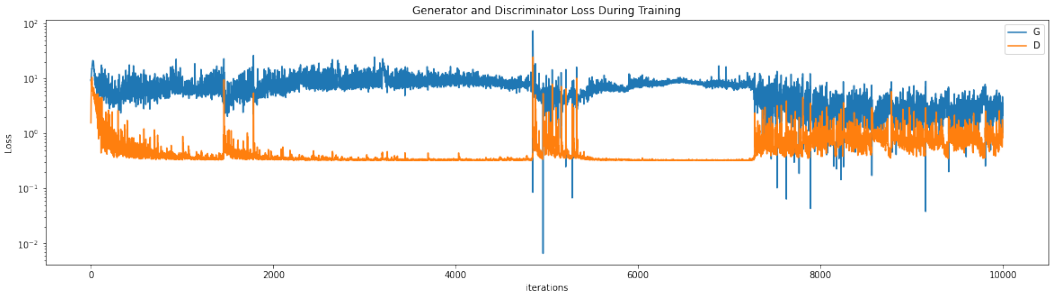

Figure 9: training losses after modifications

Upon closer observation, however, we noticed that the same images (enclosed by red boxes) was repeatedly generated from different random vectors shown in Figure 10.

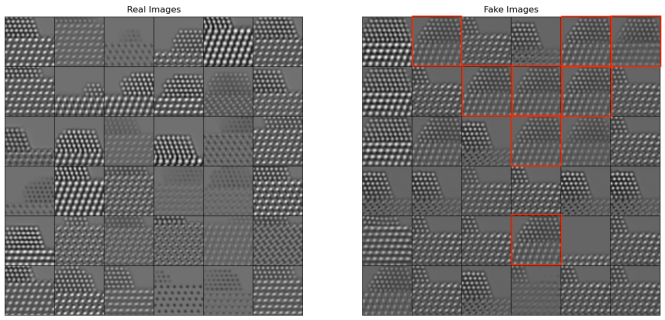

Figure 10: real Vs. fake images generated by the GAN after modifications

This is another common issue encountered in GAN training called model collapse. To overcome this issue, we employed the Two Time-Scale Update Rule (TTUR) proposed by Heusel et al. in 2017 [6]. We increased the learning rate for the discriminator to 0.0004 and decreased the learning rate for the generator to 0.0001. By forcing the generator to take smaller steps, we prevent it from choosing convenient yet imprecise solutions to win the adversarial game.

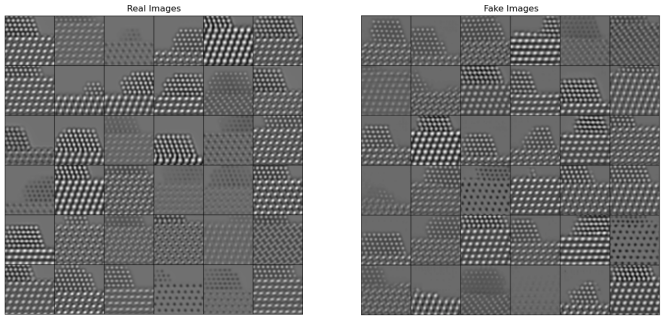

Figure 11: real Vs. fake images generated by the GAN after TTUR

The best performing generator model for each of the three different latent vector dimensions was selected for denoising applications. Now that we have obtained a fairly good generator, we can move on to the next objective of our project.

4.2.3 Optimizer Training

Previous work on GAN optimization only considered small images: [10] did not test on images larger than 32x32. To ensure that the approach was viable for the much larger microscope images, the optimization algorithm was applied to images with incrementally greater dimensionality:

1. Clean, grayscale, 32x32 pixel CIFAR100 images
2. Clean, color, 32x32 pixel CIFAR100 images
3. Clean, grayscale, 512x512 microscope images
4. Noisy, grayscale, 512x512 microscope images

A DCGAN architecture [14] trained for 500 epochs was used as the generator for the CIFAR100 images. The GAN described in 3.2.1 and 4.2.1 was used as the generator for microscope images.

Two tasks were performed for each image category. First, the optimizer was given a target image generated by the GAN. This task removes any dependency on the diversity of the GAN image space and tests only the optimizer, as the optimal image is known to be present. Second, the optimizer was given a target image not generated by the GAN and not present in the training data. The second task reflects the true application of the optimization algorithm. The optimizer successfully completed the first task to arbitrary precision for all image categories. For brevity, the Results will focus on the second task for noisy microscope images, as this is the aim of the overall project.

A variety of gradient-based optimizers and learning rates were considered. The Adam [7] optimizer with a learning rate of 0.01 was found to be most effective for all tasks and image categories. Figure 12(a) compares PSNR learning curves for multiple optimizers. Algorithm 2 was run with $m_{epochs} = 1,500$ and $n_{epochs} = 10,000$. Figure 12(b) shows the learning curves on the denoising microscope image task for latent dimension 50.

4.3 Results

The GAN optimization approach was applied to denoising both simulated and real electron microscope data. The GAN denoiser is quantitatively compared with the supervised denoiser baseline on simulated data at a variety of noise levels. No clean images exist for the real electron microscope data, so performance cannot be quantified or usefully evaluated.

Peak signal to noise ratio (PSNR) is commonly used to quantify the performance of denoising algorithms and is defined:

$$PSNR = 10 * \log_{10}(\frac{\max(I_p)^2}{MSE}) \tag{4}$$

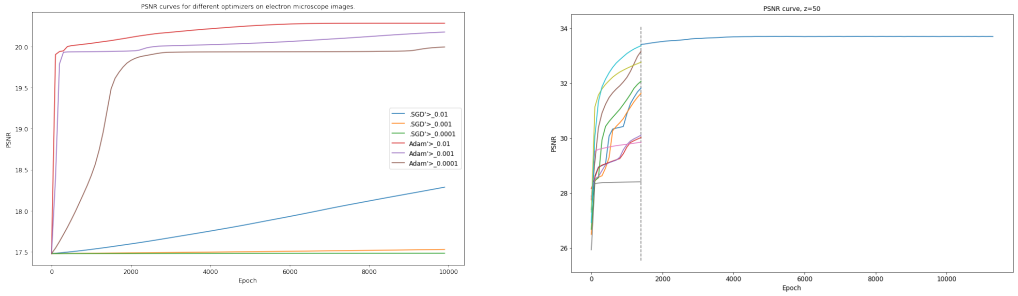

Figure 12: PSNR learning curves for a) different optimizers and b) twn samples of Algorithm 2. Both applied to GAN with latent dimension 50 over 10,000 epochs.

where I_p is the maximum pixel intensity of the true image and MSE is the pixel-wise mean squared error between the denoised image and the true image. Section 4.3.1 will use the PSNR to compare the performance on various denoising tasks.

4.3.1 Simulated Noise

Figure 13 compares the performance of the supervised and GAN denoisers on simulated noisy microscope images. Poisson and Gaussian noise at various levels is considered. The GAN denoiser with latent dimension 50 ("GAN50") achieves a higher PSNR than the other GAN denoisers for all noise levels and outperforms the supervised denoiser for all noise levels except Poiss(1). The supervised denoiser was trained for Poiss(1) noise and is not intended to be applied at different or unknown noise levels, so this result agrees with expectation. GAN50 generalizes exceptionally well, achieving PSNR levels 27.6-27.7 for every noise level except Poiss(1). Qualitatively, images generated by GAN50 are less blurry than those from the supervised denoiser, validating the motivating hypothesis that adversarial loss denoisers can capture finer detail than MSE based denoisers. However, the GAN50 images do not match the atomic structure of the original, with more rows of slightly smaller atoms. Since the optimizer found similar images for each noise level, this likely indicates that this is in fact the closest image in the range of the GAN (i.e. the optimizer was successful) but that the GAN does not have sufficient diversity to generate an image with the correct atomic structure.

As an additional baseline, [12] states that the supervised denoiser can achieve a PSNR of 38.05 for Poiss(1) noise. We were unable to replicate that result, likely because the pre-trained model applied in this paper was optimized for real images.

4.3.2 Real Electron Microscope Images

Figure 14 shows denoised images corresponding to real microscope images. As no clean image exists, PSNR cannot be calculated here. Qualitatively, the results agree with the simulated results above. The supervised denoiser appears to more closely match the atomic structure, but the GAN denoiser (for latent dimension 50) is far less blurry. Microscope images are often modeled with Poisson noise, so it is possible that the noise in the microscope image is close to the noise range in which the supervised model was trained. Because the real noise level is not fully understood, we cannot test how the denoisers generalize to different noise levels in real images.

4.4 Discussion

The mismatch of atomic structures outputted by the GAN optimizer shows the challenges of GAN based denoising. GANs are hard to train and evaluate as is, when used for denoising purposes, it becomes ever so important that it be able to not just generate realistic images, but be able to do so across every possible input in the latent domain. Intuitively, longer training of the GAN should train it to cover more latent space. However, in actual implementation, we noticed signs of model collapse for large epochs as opposed for smaller epochs, even with our modification efforts to prevent it from doing so. When exhibiting signs of model collapse, even if the GAN is capable of generating images across the entire domain, the limited variety of the images would still hinder denoising efforts.

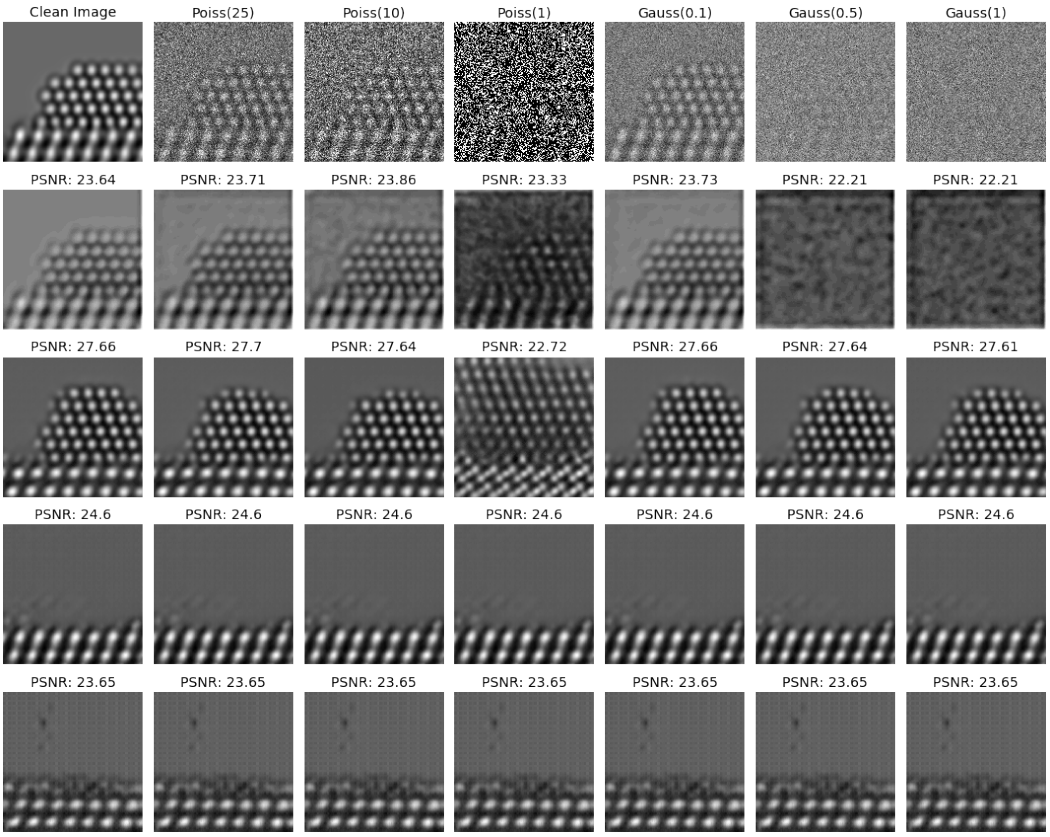

Figure 13: Denoising results across various simulated noise levels (top) for supervised (second row) and GAN denoiser with latent dimension 50 (third row), 100 (fourth row), and 150 (fifth row).

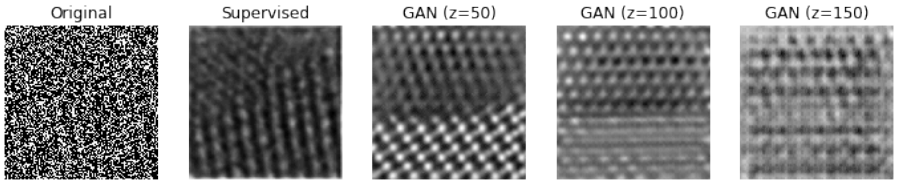

Figure 14: Multiple denoisers applied to real microscope image.

We also take a note to recognize that the generator with a latent dimension of 50 out-performed latent dimensions of 100 and 150. This is a very interesting observation. On a high level, we suspect that since the 50 dimensional latent space is much smaller than 100 or 150, our generator covered the smaller space and mapped them to images much better than the larger spaces. Therefore, when we performed our optimization, most points on the S50 maps to good images but most points on S150 may not. However, since high-dimensional latent spaces are hard to understand, the exact reason for this phenomenon will require more extensive analysis.

Overall, when compared to the MSE denoiser results, GAN generated clean images from the 50 latent dimension (shown on the third row in figure 13) resulted in a better PSNR across most noise levels, and visually the images look much clearer as well, with the features of the atoms being clearly represented. GAN generated clean images also preserved the contrast between atomic structure and background vacuum of the underlying clean image, which is something the MSE based denoiser could not recover as distinctly. This confirms our hypothesis that GAN based denoising would yield clean images that look more realistic. The bad performance of the GAN on denoising Poisson(1) noise can be understood, as it is a much larger noise than the others. It is also the only noise level where the MSE result won over the GAN results, although by a small margin. This is also expected

since the MSE denoiser specifically trained for this noise level. But the consistent winning of GAN results for all other noise levels shows the robustness of the GAN based denoiser in its ability to generalize. In reality, this robustness will serve to be much more valuable, as noise levels in actual images will vary due to different machines and the inconsistency in human operations.

5 Conclusions

The results described above indicate that latent space optimization of GAN models has potential value for image denoising if the GAN model can generate sufficiently diverse images. Specifically, the image $I_{denoised}$ that minimizes reconstruction error with I_{noisy} is also similar to I_{clean}. Further, the approach generalizes across noise types and levels better than supervised denoisers. The GAN denoised images are less blurry than the MSE-basd supervised denoiser. Lastly, previous literature on GAN denoising focused on small, natural images. The results here indicate that the approach can viably extend to larger, application-specific images (e.g., electron microscope images). Overall, our hypothesis is supported by the results, and our GAN based denoiser showed promising potential.

The results also indicate a number of directions worthy of exploration. Quantifying the diversity of GAN output is a non-trivial problem. An objective evaluation metric of GAN models remains an unsolved problem in the field since there is no consensus as to which measure best captures strengths and limitations of models and should be used for fair model comparison [2]. Manual inspection is an intuitive and commonly used metric, but it is subjective and time-consuming. Although many current metrics, such as nearest neighbor and average log-likelihood, have been explored, nearly all lack the capability to objectively assess the overall performance of the model [2].

Recent state of the art methods such as the Frechet Inception Distance (FID) show promising qualitative results consistent with human judgement. Calculating an FID score requires a trained Inception V3 model, the parameters of which must be summarized as a multivariate Gaussian for both the real and generated image. The distance between the two distributions is then computed using the Frechet distance to indicate model performance [6]. Generating an FID score for microscope images is a promising future direction for improving GAN performance, and together with manual inspection, should help us in developing a more capable GAN that can lead to closer optimization results.

Another possible approach to denoising microscope images would be to condition the GAN on the microscope parameters (e.g., focal length, pose) to provide greater control (and potentially generate greater diversity) over the generated images. Approaches to consider would be conditional GANs or embedding the parameters in the latent space.

Currently, we do not have access to a bias-free version of the MSE denoiser that is trained on the microscope data. However, it will be an interesting future exploration for comparing its results to the GAN based denoising results to see if the advantage of the generalizability of the GAN still remains over the MSE results. With this result, further analysis into the performance, robustness, training difficulty, and etc. between the two methods can help evaluate the future potential of GAN based denoising.

6 Lessons Learned

While building the optimizer, the biggest difficulty was knowing whether or not it was finding the best image. In many cases, the optimized image was not very similar to the target, but the cause of the discrepancy was not obvious. This led us to develop the suite of optimizer tests described in Section 4.2.3 so that we could be confident that the optimizer was performing well in the unexplored context of microscope images. Diagnosing the reason that an approach is failing is a much harder task in practice than in the classroom setting, and this was a prime example of that.

We also learned (as many others have) that GANs are very difficult to train effectively. We tried a number of heuristics that are suggested in the literature and by practitioners. Besides the successful modifications detailed in the Methodology, many trials failed to improve the GAN such as increasing the capacity for both models, selecting different loss functions, changing the weight update schedules, and etc. Through experimentation we finally arrived at a set of hyperparameters and a training scheme that worked, though later attempts to retrain did not always succeed. While very powerful, GANs

remain poorly understood in both theory and practice. Many recently proposed and state of the art models in data science pose the same problem of a lack of understanding of its precise mechanism behind its intuition. As a result, it is often impossible for researchers to know exactly what to do to improve their training and results. As we learned from this project, it requires patience and an extensive search of literature to achieve desirable results through trial and error.

Acknowledgements

The authors would like to acknowledge our advisors, Carlos Fernandez-Granda and Sreyas Mohan, for their guidance, helpful discussions, and assistance with the data and code infrastructure.

References

[1] A. Bora, A. Jalal, E. Price, and A. Dimakis. Compressed sensing using generative models. *International Conference on Machine Learning*, pages (pp. 537–546), 2017.

[2] A. Borji. Pros and Cons of GAN Evaluation Measures. *arXiv e-prints*, art. arXiv:1802.03446, Feb. 2018.

[3] A. Creswell and A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 30(7):1967–1974, 2018.

[4] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1701.00160, Dec. 2016.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1406.2661, June 2014.

[6] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv e-prints*, art. arXiv:1706.08500, June 2017.

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Arxiv preprint*, 2014.

[8] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images (cifar100 dataset), 2009.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2020.

[10] Z. Lipton and S. Tripathi. Precise recovery of latent vectors from generative adversarial networks. *International Conference on Learning Representations*, 2017.

[11] S. Mohan, Z. Kadkhodaie, E. Simoncelli, and C. Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[12] S. Mohan, R. Manzorro, J. Vincent, B. Tang, D. Sheth, E. Simoncelli, P. Crozier, and C. Fernandez-Granda. Deep denoising for scientific discovery: A case study in electron microscopy. *Preprint*, 2020.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[14] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1511.06434, Nov. 2015.

[15] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2020.

[16] N. Wiener. Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications. *Technology Press*, 1950.

[17] 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*, 26(7):3142–3155, 2017.

Contributions

Chuan processed the microscope training data and built and trained the GAN. Michael built the optimizer, validated it on natural images, and ran the benchmark, MSE denoiser. Both contributed equally to the paper. The code for methodologies described in the paper can be found on our Github repository

A Appendix: DCGAN optimization for CIFAR100 images

A.1 Grayscale Images

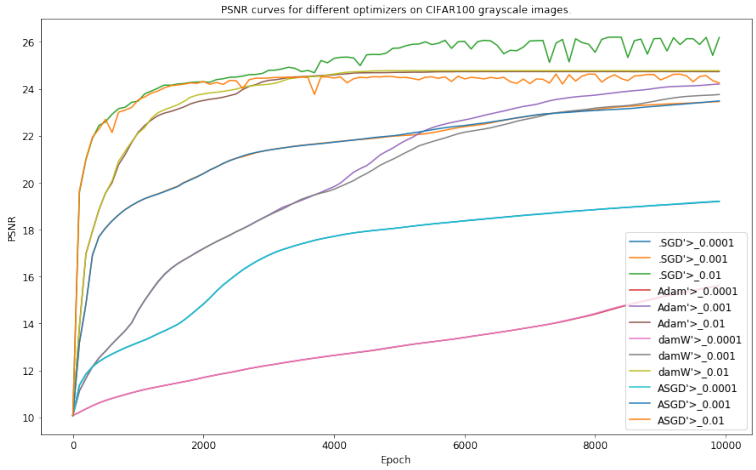

Figure 15: PSNR Curves for different optimizers and learning rates on grayscale CIFAR100 images

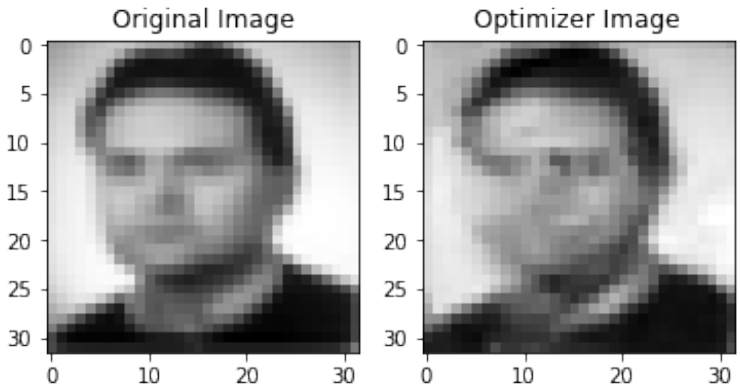

Figure 16: Image reconstruction via GAN optimization

Figure 16 shows the output of the optimizer with a target image not generated by the GAN. Figure 15 shows the PSNR learning curves for various optimizers.

A.2 RGB Color Images

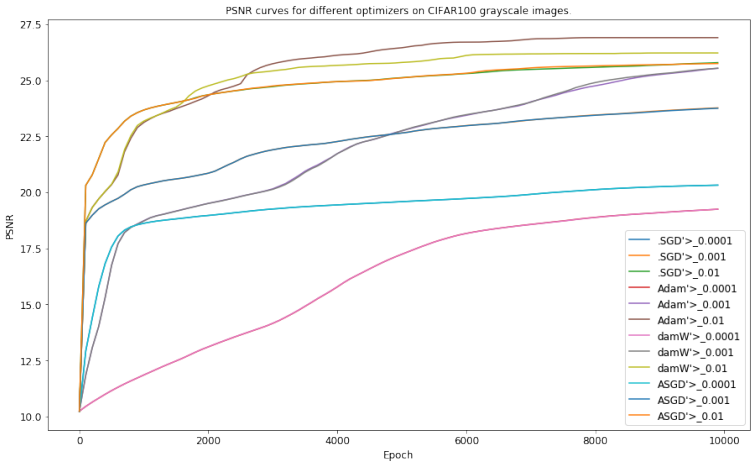

Figure 17: PSNR Curves for different optimizers and learning rates on grayscale CIFAR100 images

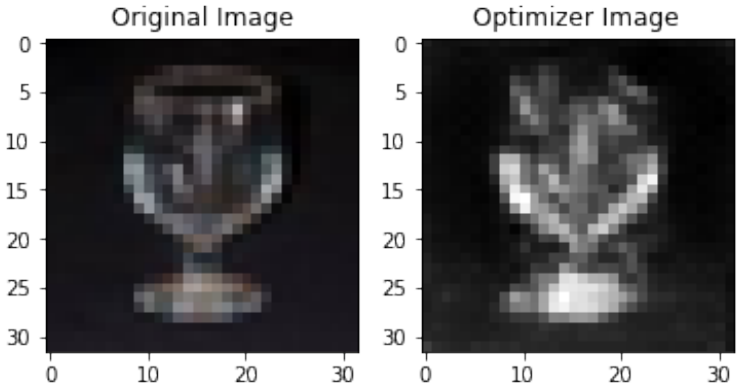

Figure 18: Image reconstruction via GAN optimization

Figure 16 shows the output of the optimizer with a target image not generated by the GAN. Figure 15 shows the PSNR learning curves for various optimizers.