LEARNING GENERATIVE MODELS FOR DEMIXING OF STRUCTURED SIGNALS FROM THEIR SUPERPOSITION USING GANS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, Generative Adversarial Networks (GANs) have emerged as popular alternative for modeling complex high dimensional distributions. Most of the existing works implicitly assume that the clean samples from the target distribution are easily available. However, in many applications this assumption is violated. In this paper, we consider the problem of learning GANs under the observation setting when the samples from target distribution are given by superposition of two structured components. We propose two novel frameworks: denoising-GAN and demixing-GAN. The denoising-GAN assumes access to clean samples from the second component and try to learn the other distribution, whereas demixing-GAN learns the distribution of the components in the same time. Through of comprehensive numerical experiments, we demonstrate that proposed frameworks can generate clean samples from unknown distributions, and provide competitive performance in tasks such as denoising, demixing, and compressive sensing.

1 INTRODUCTION

In this paper, we consider the classical problem of separating two structured signals observed under the following superposition model:

$$Y = X + N, \tag{1}$$

where $X \in \mathcal{X}$ and $N \in \mathcal{N}$ are the constituent signals, and $\mathcal{X}, \mathcal{N} \subseteq \mathbb{R}^n$ denote the two structured sets. In general the separation problem is inherently ill-posed; however, with enough structural assumption on $\mathcal{X}$ and $\mathcal{N}$, it has been established that separation is possible. Depending on the application one might be interested in estimating only X, which is referred to as *denoising*, or in recovering both X and N which is referred to as *demixing*. Both demixing and denoising arise in a variety of important practical applications in the areas of signal/image processing, computer vision, and statistics Chen et al. (2001); Elad et al. (2005); Bobin et al. (2007); Candès et al. (2011).

Most of the existing techniques assume some prior knowledge on the structures of $\mathcal{X}$ and $\mathcal{N}$ in order to recover the desired component signal(s). Prior knowledge about the structure of $\mathcal{X}$ and $\mathcal{N}$ can only obtained if one has access to the generative mechanism of the signals, or has access to clean samples from the probability distribution defined over sets $\mathcal{X}$ and $\mathcal{N}$. In many practical settings neither of these may be feasible. In this paper, we consider the problem of separating constituent signals from superposed observations when clean access to samples from the distribution is not available. In particular, we are given superposed observations $\{Y_i = X_i + N_i\}_{i=1}^K$ where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{N}$ are i.i.d samples from their respective (unknowns) distributions. In this setup, we explore two questions: First, *How can one learn prior knowledge about the individual components from superposition samples?* Second, *Can we leverage the implicitly learned constituent distributions for tasks such as denoising and demixing?*

1.1 SETUP AND OUR TECHNIQUE

Motivated by the recent success on generative models in high dimensional statistical inference tasks such as compressed sensing in Bora et al. (2017; 2018), in this paper we focus on Generative Adversarial Network (GAN) based generative models to implicitly learn the distributions, i.e., generate

samples from their distributions. Most of the existing works on GANs typically assume access to clean samples from the underlying signal distribution. However, this assumption clearly breaks down in the superposition model considered in our setup, where the structured superposition makes training generative models very challenging.

In this context, we investigate the first question with varying degrees of assumption about the access to clean samples from the two signal sources. We first focus on the setting when we have access to samples only from the constituent signal class $\mathcal{N}$ and observations, Y_i's. In this regard, we propose the *denoising*-GAN framework. However, assuming access to samples from one of the constituent signal class can be restrictive and is often not feasible in real world applications. Hence, we further relax this assumption and consider the more challenging demixing problem, where samples from the second constituent component are not available and solve it using what we call the *demixing*-GAN framework.

Finally, to answer the second question, we use our trained generator(s) from the proposed GAN frameworks for denoising and demixing tasks on unseen test samples (i.e., samples not used in the training process) by discovering the best hidden representation of the constituent components from the generative models. In addition to the denoising and demixing problems, we also consider a compressive sensing setting to test the trained generator(s). Below we explicitly list the contribution made in this paper:

1. Under the assumption that one has access to the samples from one of the constituent component, we extend the canonical GAN framework and propose *denoising*-GAN framework. This learns the prior from the training data that is heavily corrupted by additive structured component. We demonstrate its utility in denoising task via numerical experiments.
2. We extend the above denoising-GAN and propose *demixing*-GAN framework. This learns the prior for both the constituent components from their superposed observations, without access to separate samples from any of the individual components. We demonstrate its utility in demixing task via numerical experiments.

The rest of this paper is organized as follows: In section 2, we discuss relevant previous works, and compare our novelty over these existing methods. In section 3, we formally introduce our proposed approach, and in section 4, we provide some experimental results to validate our framework. Finally, we conclude this paper by providing a summary and discussion of future research directions in section 5.

2 Application and Prior Art

To overcome the inherent ambiguity issue in problem 1, many existing methods have assumed that the structures of sets (i.e., the structures can be low-rank matrices, or have sparse representation in some domain (McCoy & Tropp, 2014)) $\mathcal{X}$ and $\mathcal{N}$ are a prior known and also that the signals from $\mathcal{X}$ and $\mathcal{N}$ are "distinguishable" (Elad & Aharon, 2006; Soltani & Hegde, 2016; 2017; Druce et al., 2016). This assumption is a big restriction in many real world applications. Recently, there have been some attempts to automate this *hard-coding* approach. Among them, structured sparsity Hegde et al. (2015), dictionary learning Elad & Aharon (2006), and in general manifold learning are the prominent ones. While these approaches have been successful to some extent, they still cannot fully address the need for prior structure. Over the last decade, deep neural networks have been demonstrated to learn useful representations of real world signals such as natural images, and thus have helped us understand the structure of these high dimensional signals, for e.g. using deep generative models (Ulyanov et al., 2017).

In this paper, we focus on Generative adversarial networks GANs) Goodfellow et al. (2014) as the generative models for implicitly learning the distribution of constituent components. GANs have been established as a very successful tool for generating structured high-dimensional signals (Berthelot et al., 2017; Vondrick et al., 2016) as they do not directly learn a probability distribution; instead they generate samples from the target distribution(s) (Goodfellow, 2016). Hence, they do not need to parametrize the distributions. If we assume that the structured signals are drawn from a distribution lying on a low-dimensional manifold, GANs generate points in the high-dimensional space that resemble those coming from the true underlying distribution.

Since their inception by Goodfellow et al. (2014), there has been a flurry of works on GANs (Zhu et al., 2017; Yeh et al., 2016; Subakan & Smaragdis, 2018). In most of the existing works on GANs with few notable exceptions Wu et al. (2016); Bora et al. (2018); Kabkab et al. (2018); Hand et al. (2018); Zhu et al. (2016), it is implicitly assumed that one has access to clean samples of the desired signal. However, in many practical scenarios the desired signal is often accompanied with unnecessary components. Recently, GANs have also been used for capturing of the structure of high-dimensional signals specifically for solving inverse problems such as sparse recovery, compressive sensing, and phase retrieval (Bora et al., 2017; Kabkab et al., 2018; Hand et al., 2018). Specifically, Bora et al. (2017) have shown that generative models provide a good prior for structured signals, for e.g., natural images, under compressive sensing settings over sparsity based recovery methods. They rigorously analyze the statistical properties of a generative model based compressed sensing and provide theoretical guarantees and experimental evidence to support their claims. However, they don't explicitly propose an optimization procedure to solve the recovery problem. They simply suggest using stochastic gradient based methods in the low-dimensional latent space to recover the signal of interest. This has been addressed in Shah & Hegde (2018), where the authors propose using a projected gradient descent algorithm for solving the recovery problem directly in the ambient space (space of the desired signal). They provide theoretical guarantees for the convergence of their algorithm and also demonstrate improved empirical results over Bora et al. (2017).

While GANs have found many applications, most of them need direct access to the clean samples from the unknown distribuiton, which is not the case in many real applications such as medical imaging. AmbientGAN framework Bora et al. (2018) partially addresses this problem. In particular, they studied various measurement models and showed that their GAN can find the samples of clean signals from corrupted observations. Although similar to our effort, there are several key differences between ours and AmbientGAN. Firstly, AmbientGAN assumes that the measurement model and parameters are known, which is a very strong and limiting assumption in real applications. One of our main contributions is addressing this limitation by studying the demixing problem. Second, for the noisy measurement settings, AmbientGAN assumes an arbitrary measurement noise and no corruption in the underlying component. We consider the corruption models in signal domain rather than as a measurement one. This allows us to study the denoising problem from a highly structured corruption.

3 Background and the proposed idea

3.1 Background

Generative Adversarial Networks (GANs) are one of the successful generative models in practice was first introduced by Goodfellow et al. (2014) for generating samples from an unknown target distribution. As opposed to the other approaches for density estimation such as *Variational Auto-Encoders (VAEs)* Kingma & Welling (2013), which try to learn the distribution itself, GANs are designed to generate samples from the target probability density function. This is done through a zero-sum game between two players, *generator*, G and *discriminator*, D in which the generator G plays the role of producing the fake samples and discriminator D plays the role of a cop to find the fake and genuine samples. Mathematically, this is accomplished through the following *min-max* optimization problem:

$$\min_{\theta_g} \max_{\theta_d} \quad \mathbb{E}_{x\sim\mathcal{D}_x}[log(D_{\theta_d}(x))] + \mathbb{E}_{z\sim\mathcal{D}_z}[log(1 - D_{\theta_d}(G_{\theta_g}(z)))], \tag{2}$$

where θ_g and θ_d are the parameters of generator networks and discriminator network respectively, and $\mathcal{D}_x$ denotes the target probability distribution , and $\mathcal{D}_z$ represents the probability distribution of the hidden variables $z \in \mathbb{R}^h$, which is assumed either a uniform distribution in $[-1, 1]^h$, or standard normal. it has been shown that if G and D have enough capacity, then solving optimization problem 2 by alternative stochastic gradient descent algorithm guarantees the distribution $\mathcal{D}_g$ at the output of the generator converges to $\mathcal{D}_x$. Having discussed the basic setup of GANs next we discuss the proposed modifications this to basic GAN setup that allow for usage of GANs as a generative model for denoising and demixing structured signals.

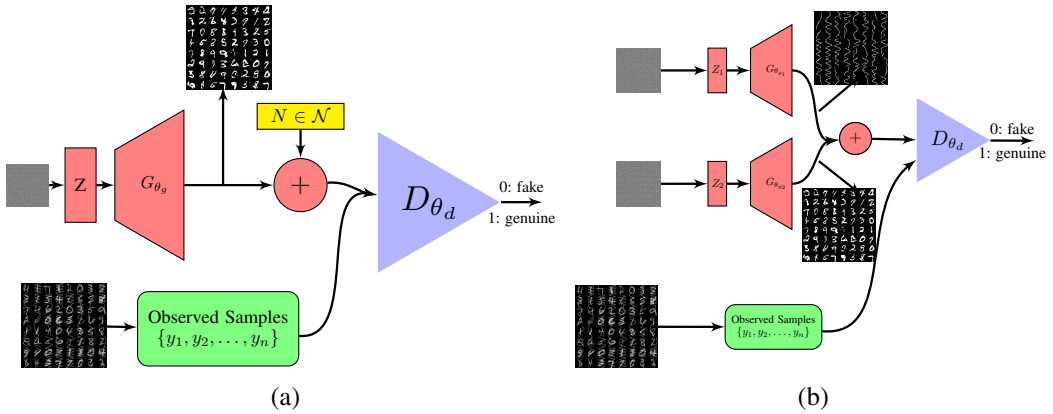

Figure 1: The architecture of proposed GANs. (a) denoising-GAN. (b) demixing-GAN.

3.2 DENOISING-GAN

Our idea is inspired by AmbientGAN due to Bora et al. (2018) in which they used a regular GAN architecture to solve some inverse problems such as inpainting, denoising from unstructured noise, etc. In particular, assume that instead of clean samples, one has access to a corrupted version of the samples where the corruption model is captured by a known random function $f_N(n)$ where N is a random variable. For instance, the corrupted samples can be generated via just adding noise, i.e., $Y = X + N$. The idea is to feed Discriminator with observed samples, y_i's (distributed as Y) together with the output of generator G which is corrupted by model $f_N(n)$. Our denoising-GAN framework is illustrated in Figure 1(a). This framework is similar to one proposed in Ambient GAN paper however, the authors did not consider denoising task from structured superposition based corruption. In the experiment section, we show that denoising-GAN framework can generate clean samples even from structured corruption.

Now we use the trained denoising-GAN framework for denoising of a new test corrupted image which has not been used in the training process. To do that we use our assumption that our components have some structure and the representation of this structure is represented by the last layer of the trained generator, i.e., $X \in G_{\widehat{\theta}_g}$[1]. This observation together with this fact that in GANs, the low-dimension random vector z is representing the hidden variables, leads us to this point: in order to denoise a new test image we have to find a hidden representation, giving the smallest distance to the corrupted image in the space of $G_{\widehat{\theta}_g}$ (Shah & Hegde, 2018; Bora et al., 2017). In other words, we have to solve the following optimization problem:

$$\widehat{z} = \arg\min_{z} \|u - G_{\widehat{\theta}_g}(z)\|_2^2 + \lambda \|z\|_2^2, \tag{3}$$

where u denotes the corrupted test image. The solution of this optimization problem provides the (best) hidden representation for unseen image. Thus, the clean image can be reconstructed by evaluating $G_{\widehat{\theta}_g}(\widehat{z})$. While optimization problem 3 is non-convex, we can still solve it by running gradient descent algorithm in order to get a stationary point[2].

3.3 DEMIXING-GAN

Now, we go through our main contribution, demixing. Figure 1(b) shows the GAN architecture, we are using for the purpose of separating or demixing of two structured signals form their superposition. As illustrated, we have used two generators and have fed each of them with a random noise vector $z \in \mathbb{R}^{100}$ uniformly distributed in $[-1, 1]^{100}$. We also assume that they are independent from each other. We have used the same architecture for both of $G_{\theta_{g_1}}$ and $G_{\theta_{g_2}}$ where they are implemented as the generator of DCGAN (Radford et al., 2015). Next, the output of generators are

[1] $G_{\widehat{\theta}_g}(.)$ denotes the trained generator network with parameter $\widehat{\theta}_g$.

[2] While we cannot guarantee the stationary point is a local minimum, but the empirical experiments show that gradient descent (implemented by backpropagation) can provide a good quality result.

summed up and the result is fed to the discriminator along with the superposition samples, $y_i's$. The architecture of the discriminator is also chosen according to the discriminator of DCGAN. In Figure 1(b), we just show the output of each generator after training for an experiment case in which the mixed image consists of 64 MNIST binary image LeCun & Cortes (2010) (for X part) and a second component constructed by random sinusoidals (for N part) (please see the experiment section for the details of all experiments). Somewhat surprisingly, this architecture based on two generators can produce samples from the distribution of each component after enough number of training iterations. We note that this approach is fully unsupervised as we only have access to the mixed samples and nothing from the samples of constituent components is known. As mentioned above, this is in sharp contrast with AmbientGAN and our previous structured denoising approach. As a result, demixing-GAN framework can generate samples from the second components (for example sinusoidal, which further can be used in the task of denoising where the corruption components are generated from highly structured sinusoidal waves).

Now similar to the denoising-GAN framework, we can use the trained generators in Figure 1(b), for demixing of the constituent components for a given test mixed image which has not been used in training. Similarly, we have to solve the following optimization problem:

$$\widehat{z_1}, \widehat{z_2} = \arg\min_{z} \|y - G_{\widehat{\theta}_{g_1}}(z_1) - G_{\widehat{\theta}_{g_2}}(z_2)\|_2^2 + \lambda_1 \|z_1\|_2^2 + +\lambda_2 \|z_2\|_2^2, \tag{4}$$

where u denotes the test mixed image and each component can be estimated by evaluating $G_{\widehat{\theta}_{g_1}}(\widehat{z_1})$ and $G_{\widehat{\theta}_{g_2}}(\widehat{z_2})$[3]. Similar to the previous case, while the optimization problem in 4 is non-convex, we can still solve it through block coordinate gradient descent algorithm, or in a alternative minimization fashion. We note that in both optimization problems 3 and 4, we did not project on the box set $[-1, 1]^{100}$. Instead we have used regularizer terms in the objective functions (which are not meant as projection step). We empirically have observed that imposing these regularizers can help to obtain good quality images in our experiment; plus, they may help that the gradient flow to be close in the region of interest by generators. This is also used in Bora et al. (2017).

4 Numerical Experiments

In this section, we present various experiments showing the efficacy of the proposed frameworks (depicted in Figure 1(a) and Figure 1(b)) in three different setups. First, we will focus on the denoising from structured corruption both in training and testing scenarios. Next, we focus on demixing signals from structured distributions. Finally, we explore usage of generative models from the proposed GAN frameworks compressive sensing setup. In all the following experiments, we did our best for choosing all the hyperparameters.

The network architectures for discriminator and generator in these experiments are similar to ones proposed in DCGAN Radford et al. (2015). DCGAN is a CNN based GAN, in which there are some convolution layers used in both generator and discriminator followed by batch normalization (except last layer of generator and first layer of discriminator). All the nonlinear activation layers are Relu except the last layer of discriminator.

4.1 Structured corruption models

For all the experiments, we have used binary MNIST dataset. For the corruption part, we have used two structured noise models similar to Chen & Srihari (2014). In the first one, we generate random vertical and horizontal lines, and add them to the dataset. The second structured noise is constructed based on random sinusoidal waves in which the amplitude, frequency, and phase are random numbers. We define the level of corruption as the number of sinusoidal or lines added to the original image. We note that both of these corruption models are highly structured. These two corruption model along with the clean binary MNIST image have been shown in the figure 2.

[3] $G_{\widehat{\theta}_{g_1}}(.)$ and $G_{\widehat{\theta}_{g_2}}(.)$ denote the first and second trained generator with parameter $\widehat{\theta}_{g_1}$ and $\widehat{\theta}_{g_2}$, respectively.

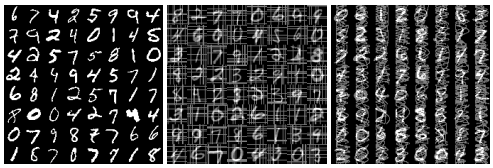

Figure 2: Left: Clean binary MNIST image. Middle: Corrupted image with random horizontal and vertical lines. Right: Corrupted image with random sinusoidal waves.

4.2 DENOSING FROM STRUCTURED CORRUPTION – TRAINING

We use GAN architecture illustrated in Figure 1(a) for removing of the structured noise. The setup of the experiment is as follows: we use 55000 images with size 28×28 corrupted by either of the above corruption models[4]. The resulting images, y_i's are fed to the discriminator. We also use hidden random vector $z \in \mathbb{R}^{100}$ drawn from uniform distribution in $[-1, 1]^{100}$ for the input of generator. During the training, we use 64 mini-batches along with the regular loss function in the GAN literature, stated in problem 2. The optimization algorithm is set to Adam optimizer, and we train discriminator and generator one time in each iteration. We set the number of epochs to 64. To show the evolution of the quality of output samples by generator, we fix an input vector z, and save the output of the generator in different times during the training process. Figure 3 shows the denoising process for both corruption models. As we can see, the GAN used in estimating the

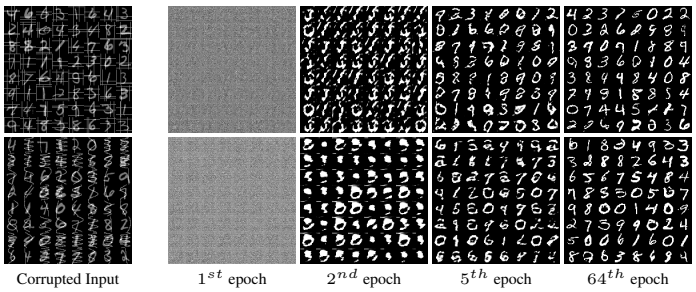

Figure 3: Evolution of outputted samples by generator for fixed z. Top rows is for random horizontal and vertical corruption. Bottom row is for random sinusoidal corruption.

clean images from structured noises is able to generate clean images after training of generator and discriminator. In the next section, we use our trained generator for denoising of new images which have not been used during the training.

4.3 DENOSING FROM STRUCTURED CORRUPTION – TESTING

Now we test our framework with test corrupted images for both models of corruption introduced above. For reconstructing, we solve the optimization problem in 3 to obtain solution $\widehat{z}$. Then we find the reconstructed clean images by evaluating $G_{\widehat{\theta}_g}(\widehat{z})$. In the figures 4, we have used the different level of corruptions for both of the corruption models. In top right, we varies the level of corruption from 1 to 5 with random sines. The result of $G_{\widehat{\theta}_g}(\widehat{z})$ has been shown below of each level of corruption. In the top left, we have the similar experiment with various level of vertical and horizontal corruptions. Also, we show the denoised images in the below of the corrupted ones. As we can see, even with heavily corrupted images (level corruption equals to 5), GAN is able to remove the corruption from unseen images and reconstruct the clean images.

In the bottom row of Figure 4, we evaluate the quality of reconstructed images compared to the corrupted ones through a classification test. That is, we use a pre-trained model for MNIST clas-

[4]We set level of corruption 1 for sinusoidal and 2 for the vertical and horizontal lines.

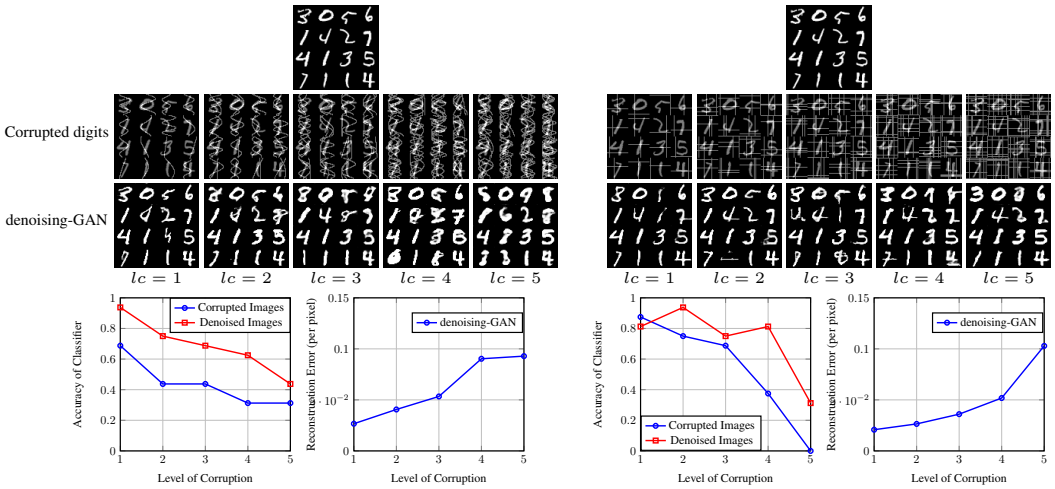

Figure 4: The performance of trained generator in different level of corruption (lc) for denoising of unseen corrupted digits. Top row: Ground truth digits together with corrupted digits with random sinusoids, and vertical and horizontal lines with level of corruption from 1 to 5. Bottom row: Classification accuracy of pre-trained MNIST classifier for both corrupted and denoised digits along with the reconstruction error per pixel.

sifier which has test accuracy $\%98$[5]. We feed the MNIST classifier with both denoised (output of denoising-GAN) and corrupted images with different level of corruptions. For the ground truth labels, we use the labels corresponding to the images before corruption. One interesting point is that when the level corruption increases the denosied digits are sometimes tweaked compared to the ground truth. That is, in pixel-level they might not close to the ground truth; however, semantically they are the same. We also plot the reconstruction error per pixel (normalized by 16 images).

4.4 Demixing of structured signals – Training

In this section, we present the results of our experiments for demixing of the structured components. To do this, we use the proposed architecture in Figure 1(b). For this experiment, we consider four sets of constituent components. In the first two, similar to the denoising case, we use both random sinusoidal waves and random vertical and horizontal lines for the second structured constituent component. The difference here is that we are interest in generating of the samples from the second component as well. In figure 5, we show the training evolution of two fixed random vectors, z_1 and z_2 in $\mathbb{R}^{100}$ in the output of two generators. In the left panel, we have used one random sinusoidal waves for the second component. As we can see, our proposed GAN architecture can learn two distributions and generate samples from each of them. In the right panel, we repeat the same experiment with random vertical and horizontal lines as the second component. While there are some notion of mode collapse, still two generator can produce the samples from the distribution of the constituent components.

In the second scenario, our mixed images comprise of two MNIST digits from 0 to 9. In this case, we are interested in learning the distribution from which each of the digits are drawn. Left panel in Figure 6 show the evolution of two fixed random vectors, z_1 and z_2. As we can see, after 32 epoch, the output of the generators would be the samples of MNIST digits. Finally, in the last scenario, we generate the mixed images as the superposition of digits 1 and 2. In the training set of MNIST dataset, there are around 6000 digits 1 and 2. We have used these digits to form the set of superposition images. The left panel of Figure 6 shows the output of two generators, which can learn the distribution of the two digits. The interesting point is that these experiment show that each GAN can learn the existing digit variety in MNIST training data set, and we typically do not see

[5]The architecture comprises of two initial convolutional layers along with max-pooling followed by a fully connected layer and a dropout on top of it. Relu is used for all the activation functions and a soft-max function is used in the last layer.

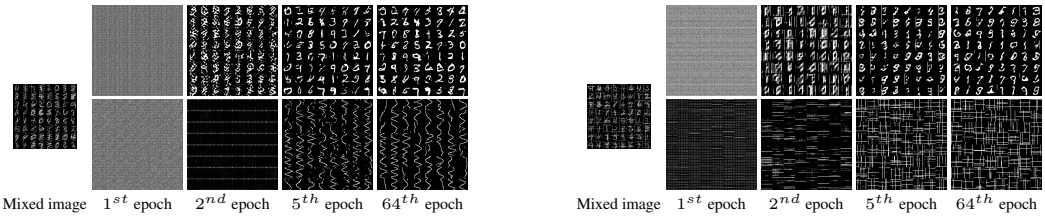

Figure 5: Evolution of output samples by two generators for fixed z_1 and z_2. Right panel shows the evolution of the two generators in different epochs where the mixed images comprise of digits and sinusoidal. The first generator is learning the distribution of MNIST digits, while the the second one is learning the random sinusoidal waves. Left Panel shows the same experiment with random horizontal and vertical lines as the second components in the mixed images.

mode collapse, which is a major problem in the training of GANS (Goodfellow, 2016). Due to lack of space, we present our results for the compressive sensing setup in the appendix.

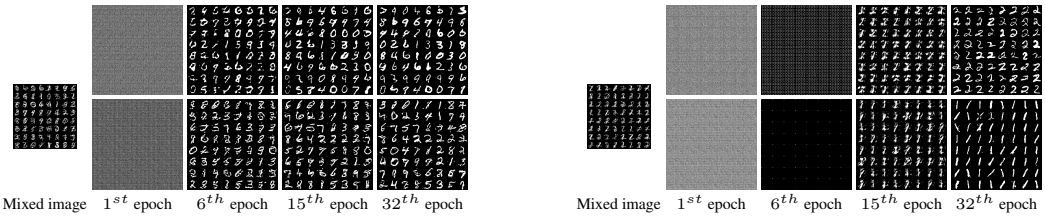

Figure 6: Evolution of output samples by two generators for fixed z_1 and z_2. Right panel shows that each generator is learning the distribution of one digit out of all 10 possible digits. The mixed images comprise two arbitrary digits between 0 to 9. Left Panel shows the similar experiment where the mixed images comprise only digits 1 and 2.

4.5 DEMIXING OF STRUCTURED SIGNALS – TESTING

Similar to the test part of denoising, in this section, we test the performance of two trained generators in demixing scenario for the mixed images, have not been seen in the training time. In Figure 7, we have illustrated demixing on three different input mixed images. In the left and middle panel, we consider the mixed images generated by adding a digit (drawn from MNIST test dataset) and a random sinusoidal. Then the goal is to separate (demix) these two from the given superimposed image. To do this, we use GAN trained for learning the distribution of digits and sinusoidal waves (the left panel of Figure 5) and solve the optimization problem in 4 through an alternative minimization fashion. As a result, we obtain $\widehat{z_1}$ and $\widehat{z_2}$. The corresponding constituent components in each panel is then obtained by evaluating $G_{\widehat{\theta}_{g_1}}(\widehat{z_1})$ and $G_{\widehat{\theta}_{g_2}}(\widehat{z_2})$. In the right panel of Figure 7, we have shown the third experiment in which the input is the superposition of two digits (1 and 2) drawn from the MNIST test dataset. In this experiment, we have used the GAN trained for learning the distribution of digits 1 and 2 (right panel in figure 6). As we can see, our proposed GAN can separate two digits; however, similar to the denoising case, the outputs of it are semantically correct but not exactly same as the ground truth constituent components.

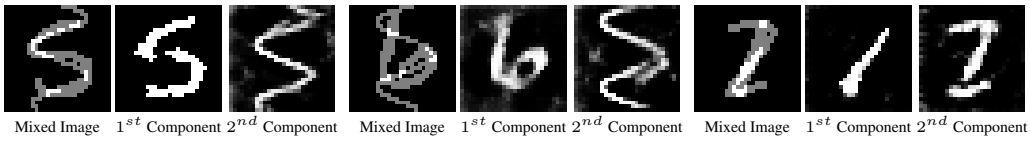

Figure 7: The performance of trained generators for demixing of two constituent components. Left and middle: The mixed image is the superposition of an unseen digit (drawn from MNIST test dataset) and a random sinusoidal. Right: The mixed image is the superposition of digits 1 and 2 drawn from MNIST test dataset.

REFERENCES

D. Berthelot, T. Schumm, and L. Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

J. Bobin, J. Starck, J. Fadili, Y. Moudden, and D. Donoho. Morphological component analysis: An adaptive thresholding strategy. *IEEE Trans. Image Proc.*, 16(11):2675–2681, 2007.

A. Bora, A. Jalal, E. Price, and A. Dimakis. Compressed sensing using generative models. *Proc. Int. Conf. Machine Learning*, 2017.

A. Bora, E. Price, and A. Dimakis. Ambientgan: Generative models from lossy measurements. In *Int. Conf. on Learning Rep. (ICLR)*, 2018.

E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

G. Chen and S. Srihari. Removing structural noise in handwriting images using deep learning. In *Proc. Indian Conf.e on Comp. Vision Graph. Image Proc.* ACM, 2014.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43 (1):129–159, 2001.

J. Druce, S. Gonella, M. Kadkhodaie, S. Jain, and J. Haupt. Defect triangulation via demixing algorithms based on dictionaries with different morphological complexity. In *Proc. European Workshop on Struc. Health Monitoring (IWSHM)*, 2016.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Proc.*, 15(12):3736–3745, 2006.

M. Elad, J. Starck, P. Querre, and D. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Appl. Comput. Harmonic Analysis*, 19(3):340–358, 2005.

I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pp. 2672–2680, 2014.

P. Hand, O. Leong, and V. Voroninski. Phase retrieval under a generative prior. *arXiv preprint arXiv:1807.04261*, 2018.

C. Hegde, P. Indyk, and L. Schmidt. Approximation algorithms for model-based compressive sensing. *IEEE Trans. Inform. Theory*, 61(9):5129–5147, 2015.

M. Kabkab, P. Samangouei, and R. Chellappa. Task-aware compressed sensing with generative adversarial networks. *Proc. Assoc. Adv. Artificial Intelligence (AAAI)*, 2018.

D. P Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

M. McCoy and J. Tropp. Sharp recovery bounds for convex demixing, with applications. *Foundations of Comp. Math.*, 14(3):503–567, 2014.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

V. Shah and C. Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2018.

M. Soltani and C. Hegde. Iterative thresholding for demixing structured superpositions in high dimensions. *NIPS Workshop on Learning. High Dimen. Structure (LHDS)*, 2016.

M. Soltani and C. Hegde. Fast algorithms for demixing sparse signals from nonlinear observations. *IEEE Trans. Sig. Proc.*, 65(16):4209–4222, 2017. ISSN 1053-587X.

Y . Subakan and P. Smaragdis. Generative adversarial source separation. In *Proc. Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, pp. 26–30, 2018.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *arXiv preprint arXiv:1711.10925*, 2017.

C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pp. 613–621, 2016.

J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Adv. Neural Inf. Proc. Sys. (NIPS)*, pp. 82–90, 2016.

R. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. Do. Semantic image inpainting with perceptual and contextual losses. arxiv preprint. *arXiv preprint arXiv:1607.07539*, 2, 2016.

J. Zhu, P. Krähenbühl, E. Shechtman, and A. Efros. Generative visual manipulation on the natural image manifold. In *Proc. European Conf. Comp. Vision (ECCV)*, pp. 597–613, 2016.

J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proc. Int. Conf. Comp. Vision (ICCV)*, 2017.

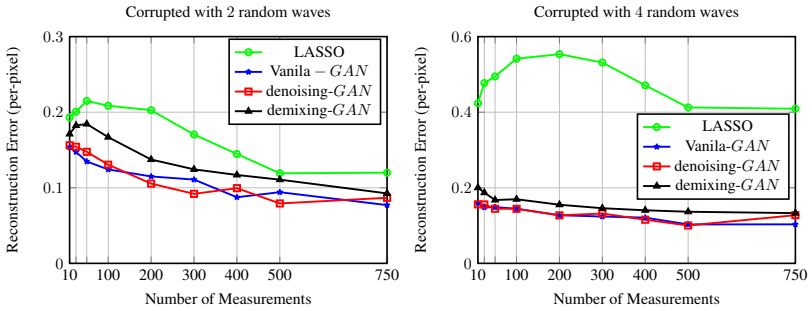

Figure 8: Performance of Different GANs in Compressive Sensing Experiments.

5 APPENDIX

5.1 COMPRESSIVE SENSING

In this section, we present results in the compressed sensing setting. In particular, we assume the following observation model by a random sensing matrix $A \in \mathbb{R}^{m \times n}$ under structured corruption by n as follows

$$Y = A(X + N), \tag{5}$$

where the entries of A are i.i.d Gaussian with zero mean and variable $\frac{1}{m}$. For a given generator model $G_{\widehat{\theta}}$ inference we solve the following inference problem

$$\min_{z} \|Y - AG_{\widehat{\theta}}(z)\| + \lambda \|z\|_2^2, \tag{6}$$

where the generator $G_{\widehat{\theta}}$ is taken from various generator networks. For this experiment for the signal X we choose random MNIST images from the test set, and N is taken random sinusoidal corruption varying number of waves. We compare generator models learned from clean MNIST images, generator model obtained from *denoising*-GAN and *demixing*-GAN framework. Both the generator models learned from denoising-GAN and demixing-GAN were trained under wave corruption model. We choose the appropriate generator from demixing-GAN by manually looking at the output of two generators. Then we choose the one that gives MNIST like images as the output. We also compare our approach to LASSO as MNIST images are naturally sparse. The regularization parameter for LASSO was fixed to 0.1. For solving the inference problem in equation 6, we have used ADAM optimizer with step size set to 0.01. Since the problem is non-convex, we have used 10 random initialization with 10000 iterations. We choose the one gives the best measurement error. We set the parameter $\lambda = 0.1$ for all the generative models as it leads to the best results on the validation set.

The results of this experiment are presented in Figure 8 where we report per-pixel reconstruction error on 25 images chosen randomly from the test dataset for two different corruption levels. The plot on the left is obtained when the signal is corrupted with 2 random waves, whereas the right plot corresponds to the corruption with 4 random waves. In both the figures, we observe that with corruption the performance of LASSO significantly degrades. The generator from demixing-GAN improves the performance over LASSO. Generator obtained from denoising-GAN performs comparable to the vanilla GAN. This experiment establishes the quality of the generative models learned from GANs as the prior. The comparable performance of denoising GAN with vanilla GAN shows that meaningful prior can be learned even from heavily corrupted samples.