WGANs for Semantic Image Inpainting

Hudson Cooper

Abstract

In this paper, I implement a Wasserstein GAN, a recent variation of the popular generative adversarial network framework for estimating generative models. The WGAN framework simultaneously trains two models, a generative model G and a 'critic' model f, which estimates the Wasserstein distance between the sampling distribution of G and the empirical distribution of observed data, providing G with a loss function with smooth and useful gradients almost everywhere. The output of f is easily interpretable as the semantic 'quality' of the generative model and can be reasonably applied to tasks other than simple image generation, as can be seen by my applying it to the task of inpainting.

1 Introduction

Generative models are an extremely active area of study in machine learning. Generative models aim to capture the often high dimensional probability distributions of observed data, either by explicitly estimating their probability density functions or by estimating models with similar sampling distributions. These models can be difficult to estimate in the case of high dimensional data, as the curse of dimensionality typically means that data is sparse in their embedding space and many assumptions must be made to decide how to handle the gaps. Naively using metrics which compare point to point distance, ℓ_1 or ℓ_2 distance, for example, often lead to extremely poor models.

GANs are a class of generative models which attempt to solve this problem by learning a similarity metric between the distributions of generated samples and observed samples. They attempt to capture *semantic* information about the observed data rather than distance in an arbitrary embedding space. For example, if we are attempting to learn a generative model to produce images of digits, we do not care about the exact values of the pixels in a generated "3"; we instead care whether the generated image is *qualitatively* a "3".

Inpainting is the task of recovering missing or corrupted portions of an image. A model that is good at inpainting should fill in the missing portions with content that is semantically appropriate and motivated by the regions around it. Thus GANs or other generative models that attempt to capture this semantic information are ripe for the task.

2 Method

In this section, I will review the generative adversarial network (GAN) of Goodfellow *et al.* [1] and its modification in Wasserstein GANs, introduced by Arjovsky *et al.* [2]. I will then introduce the method of inpainting described by Yeh *et al.* [3] and contrast it to the method I designed for the same process.

2.1 Generative Adversarial Networks

GANs were introduced in [1] as a novel method of estimating generative models via a minimax game played between a generative model G and a discriminative model D. G attempts to generate new samples similar to observed data, X, while D estimates the probability that a given sample comes from either X or from G. The training procedure is thus alternatingly training G in order to maximize the errors of D and training D to differentiate between samples from G and samples from X.

A useful analogy is to think of "the generative model... as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency" [1]. This method has several

advantages, in particular, the ability to train a generative model through backpropogation (so long as G and D are defined by continuous, differentiable functions) and do so in such a way that captures semantic understanding of the data distribution, rather than minimizing something decidedly non-semantic, for example the ℓ_1 error.
The GAN algorithm defines a known prior noise distribution $p_z(z)$ and a generator $G(z; g)$ which maps noise to the data space, defining a posterior distribution over generated samples, p_g. The discriminator, $D(x; d)$ maps from the data space to the range (0, 1), representing the probability that x comes from the observed data distribution p_{data} rather than from p_g. The training procedure, which alternatingly learns parameters g and d, thus maps to the following minimax game with value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{p_{data}(x)} \log(D(x)) + \mathbb{E}_{p_z(z)} \log(1 - D(G(z))) \tag{1}$$

2.2 Wasserstein Generative Adversarial Networks

The WGAN is a slight modification of the GAN described above. Arjovsky *et al.* prove in [4] that as the GAN discriminator described in [1] is trained to optimality, it saturates and suffers from vanishing gradients, proving problematic for updating the parameters for G. The WGAN solves this problem by simply replacing the discriminator $D(x, d)$ with a "critic," $f(x, w)$, which returns a real scalar value that not need look anything like a probability. Arjovsky, et al. show that as f is trained to maximize

$$W(G, f) = \mathbb{E}_{p_{data}(x)} D(x) - \mathbb{E}_{p_z(z)} f(G(z)), \tag{2}$$

W approximates the Wasserstein distance metric between p_data and p_z up to a factor of scale - as long as f is K-Lipschitz for some K, accomplished by explicitly clipping the parameters of f to some fixed range. The generator G is then tasked with minimizing W, so the WGAN algotithm is thus defined by an minimax game analogous to equation 1.
There are several arguments for using this formulation instead of the standard GAN. In [1], Goodfellow *et al.*, showed that the discriminator function D in the GAN formulation approximates the Jensen-Shannon (JS) divergence between p_{data} and p_g. Arjovsky, et al. note in [2] that minimizing the Wasserstein distance is weaker than minimizing JS divergence, but it has better guarantees with respect to continuity, differentiability, and the usefulness of its gradients on all parts of its domain. Under the WGAN framework, we can and should train our critic to optimality, while this is advised against in the standard GAN as it would lead to vanishing gradients. Additionally, reducing the estimated Wasserstein loss appears to correlate well with increased perceived generator quality in the domain of image generation, while the standard GAN loss may go up and down with increased image quality. This makes W a good candidate function to minimize in order to generate reasonable samples for inpainting.

2.3 Inpainting

Given an image x to be inpainted and a generative model G trained to produce images with semantic qualities similar to those of x, the next task is to generate images which match the specific missing portions. Thus a second loss function must be defined in order to specify how this task will be achieved.
In either inpainting method below, gradient descent is used to find the $\hat{z}$ which minimizes the specified loss $\mathbb{L}$ with respect to x and fill in the missing image portions with $G(\hat{z})$. Throughout the gradient descent process, $\hat{z}$ is constrained to the support of p_z in order to ensure all samples are being generated from the true generator distribution p_g.

2.3.1 The Method of Yeh *et al.*

[3] by Yeh *et al.* applies GANs to the task of inpainting and defines two useful loss metrics. The first loss they define is the contextual loss, the ℓ_1 distance between the known pixels of a corrupted image x and the corresponding pixels generated by a generative model G. Minimizing the contextual loss ensures that inpainting suggestions are faithful to pixels whose true values are known, such that the completed image is coherent:

$$\mathbb{L}_{contextual}(z) = ||M \odot G(z) - M \odot x|| \tag{3}$$

Here, M is a binary mask which takes the value 0 for every missing pixel and 1 everywhere else, and $\odot$ is the Hadamard elementwise product. The second loss defined in [3] is the perceptual loss, the portion of the GAN's discriminator loss dependent on the output of the generator. The perceptual loss ensures that the inpainting suggestions are drawn correctly from the generator distribution. Here, $G(z)$ is the image generated to satisfy the inpainting task:

$$\mathbb{L}_{perceptual}(z) = log(1 - D(G(z)) \tag{4}$$

Since my implementation uses a WGAN instead of a traditional GAN, I replace this perceptual loss given by D with the analogous WGAN loss given by f:

$$\mathbb{L}_{perceptual}(z) = -f(G(z)) \tag{5}$$

Finally, [Yeh]'s total inpainting loss is defined a linear combination of the contextual and perceptual losses combined with tradeoff parameter λ:

$$\mathbb{L}(z) = \mathbb{L}_{perceptual}(z) + \lambda\mathbb{L}_{perceptual}(z) \tag{6}$$

2.3.2 My Method

The idea behind my method is to remove the use of any with non-semantic driven losses like the ℓ_1 norm used in the contextual loss of [3]. Thus, I define a combined contextual and perceptual loss given solely by the critic f of a trained WGAN applied to the final inpainting product itself:

$$\mathbb{L}(z) = f(M \odot x + (1 - M) \odot G(z)) \tag{7}$$

3 Experiment

3.1 Data Set

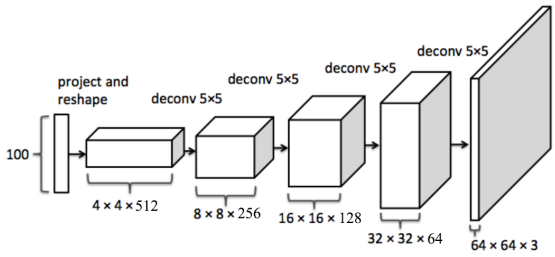
For all experiments, I used the CelebA face data set [5], an image set consisting of roughly 200,000 centered celebrity faces.

3.2 Architecture

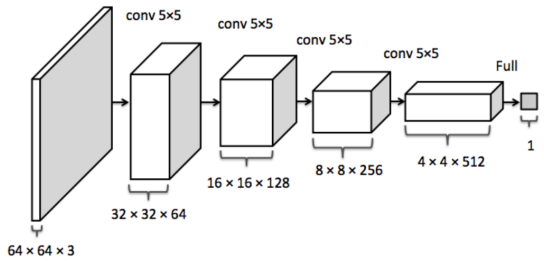
I used the DCGAN architecture from Radford *et al.* [6] trained with the Wasserstein loss instead of the standard GAN loss. The generator G is structured as follows: input z is drawn from a 100 dimensional vector, uniform on the unit [-1,1] hypercube. This input feeds into a 8192 unit fully connected layer reshaped into an array of 4x4 spatial dimensions and 512 channels. This is followed by four 5x5 transpose convolutions, each doubling the spatial dimensions and halving the number of channels, except for the output layer, which is 64x64x3. Each layer, including the fully connected layer after it is reshaped, is passed through a batch normalization and a relu activation. The output layer has no batch norm or relu, but it has a tanh activation, found by [6] to stabilize training. All weights are drawn from zero centered normal with variance .02, as suggested.
The critic f is structured essentially like G in reverse, taking as input 64x64x3 images with every value scaled to the range [-1,1] (the range of tanh). There are four convolutions, the first which outputs 32x32 spatial dimensions and 64 channels and has a leaky relu activation. Each subsequent convolution halves the number of spatial dimensions and doubles the number of channels and is followed by a batch norm and a leaky relu. All leaky relu functions have leak parameter $\alpha = .2$. Finally, there is a fully connected layer with a one dimensional output with no activation function. All weights must be clipped to the range [-c, c] during training to satisfy the K-Lipschitz property of f. I use c = .01, as suggested by [2].
Both architectures are shown in figure 1.

3.3 Training the Generative Model

The training process follows the procedure in Arjovsky *et al.* [2], using the RMSprop optimizer. I follow [2]'s suggestion of training the critic for 5 iterations for every 1 generator iteration. To keep f as optimal as possible without sacrificing training time, I also train f for 100 iterations for the first 25 generator iteration and 100 times every 500 generator iterations. I used the values of



(a)



(b)

Figure 1: (a) Structure of generator. (b) Structure of critic. Figure modified from [3]

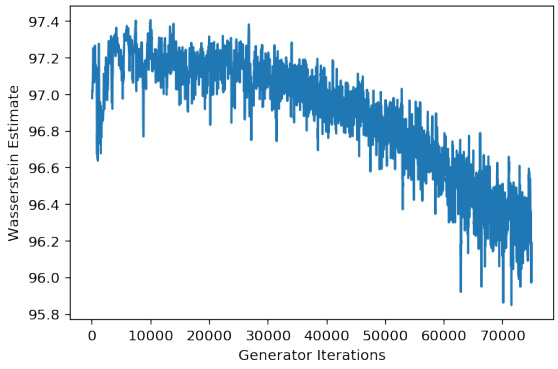


Figure 2: WGAN Loss with median smoothing (kernel size 101)

.00005 for learning rate and .99 for RMSprop's forgetting factor.
Due to time constraints (and a catastrophic Windows crash), the WGAN was not trained to optimality. It was trained for 75000 generator iterations, or roughly 24 epochs through the data. Figure 2: The estimated Wasserstein metric is plotted below against generator iterations. Training was extremely noisy, even with large bandwidth smoothing. This suggests to me that further hyperparameter tuning was necessary. Figure 3: resulting generated images. Quality suffers because network was not trained to optimality.

3.4 Application to Inpainting

The WGAN was trained on the entire CelebA data set, minus 200 hold out images. These hold out images were corrupted in the two following ways. The first corruption removed a 16x16 pixel square from the center of the face in each image. The second corruption removed 80% of the image's pixels at random. Below are images completed with my inpainting method and the method of [3], side by side for comparison. For Yeh *et al.*'s method, I used a λ of 1, weighting perceptual and contextual loss evenly. Figure 4: results for 10 randomly selected images.

Figure 3: 100 random samples from trained WGAN

Figure 4: (Left) my method. (Right) the method of [3]. (Both) Column one is the uncorrupted image. Columns two and three are the first corruption schema and its recovery. Columns four and five are the second corruption schema and its recovery.

4 Conclusion

I was not able to compare the results of my inpainting method directly to the results of [3] because they used a DCGAN with the 'vanilla' GAN discriminator D, while all experiments I conducted used the WGAN discriminator f. All images I generated using their loss function replaced D with f (equation 4 with equation 5).
Furthermore, due to time constraints, the WGAN could not be trained to optimality, and the quality of the generated images clearly suffer. Due to this, it is hard to compare the results of inpainting with my algorithm to those of [3]. My method appears to have generated reasonable inpainting suggestions under the first corruption schema, but in the second corruption schema, it appears that my method threw almost all contextual information away and simply generated something that looks like an average face.
My results are inconclusive, but it appears that minimizing the WGAN critic loss directly on the inpainted image produces reasonable (nonetheless poor) suggestions for the task. More experiments are clearly needed. I learned a lot in implementing these methods, and I plan to keep working on this experiment until I can reach satisfactory results.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[2] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[3] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with perceptual and contextual losses," *arXiv preprint arXiv:1607.07539*, 2016.

[4] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, vol. 2016, 2017.

[5] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

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