Factors Influencing Success of Conditional Generative Adversarial Networks

Kelly Marshall
NYU Shanghai
Shanghai, China
km3888@nyu.edu

Jeff Soules
NYU Courant
New York, NY
jds504@nyu.edu

Yuntian Ye
NYU Shanghai
Shanghai, China
yy1947@nyu.edu

Abstract

Conditional GANs can produce near-magical results in the field of image manipulation. However, most research into improved performance focuses on clever design, with results also driven by compute power and extensive data sets. Less attention is paid to comparative study of factors determining the success of both new and core methods.

To remedy this situation we present a tool which enables easy methodical experimentation on both hyperparameters and loss functions using the Pix2Pix conditional GAN architecture. We review the theoretical foundations of how loss functions influence GAN training, with particular attention to the issue of premature termination of training progress. Finally, we present the results of a small investigation into several loss function and hyperparameter settings in the edgemap-to-image conditional-GAN task.

1. Introduction

1.1. Generative Adversarial Networks

Generative Adversarial Networks are a family of algorithms which have gained popularity within the past several years as a means of learning a distribution from observed data and generating novel data based on that distribution. This is accomplished through a game-theoretic approach: two artificial neural networks are trained against each other, one which learns to imitate the true distribution and another which learns to discriminate between the original data and that created by the generator. GANs differ in the loss functions used to update their networks' parameters, with different functions providing different learning guarantees, and performance over different datasets. In the original implementation, the loss function for the discriminator is the Jensen-Shannon (JS) divergence between the true and generated distributions. [2]

1.2. Drawbacks to Original GANs

Despite representing a major step forward for generative models, the original GAN model proposed by Goodfellow *et al*. was hindered by the shortcomings of the JS divergence, which is based on the Kullback-Leibler divergence. KL divergence cannot finitely quantify the distance between two probability distributions with non-overlapping support; the resulting loss function does not reliably provide gradients once the discriminator has been thoroughly trained. As a result, the training process of the original GAN is unstable and is not guaranteed to converge, nor to train to maximum effectiveness. Moreover, continuing to make acceptable progress for the generator may require the discriminator to be artificially hampered with noise: resulting in a complete network which favors blurry, unrealistic images.

Many alternative loss functions have been proposed to overcome the shortcomings of the JS-divergence loss function in original GANs, with varying degrees of theoretical support. Moreover, due to a lack of systematic exploration of the factors influencing GAN performance, there is disagreement about the relative importance of hyperparameters and loss function choice. Our current work reviews some of the better-grounded proposals and compares the performance of the original GAN with that of the Wasserstein GAN in conditional GAN applications. We also lay out a software framework for more systematic exploration of the influence of different parameter settings on cGAN performance.

1.3. Conditional GANs

Building on the generative functionality of the unconditional GANs proposed by [2], conditional GANs are GANs trained to condition their output on both the domain and a specific input–in effect learning a conditional distribution. By feeding class information to the generator network, the generated output takes on characteristics of the input[10]. This results in output that is shaped by conditioning while still resembling real data. For instance, a generator which generates images from the MNIST dataset could be conditioned on a one-hot vector encoding of the desired digit's

class; after training, it would output realistic depictions of whichever specific digit is presented as stimulus.

Within image processing, conditional GANs find application in areas of style transfer (applying a different "filter" to an underlying image) and image completion/infilling, as well as *de novo* image or video creation based on a classification, semantic mapping, or text description.

2. GANs and Loss Function

The most prominent factor limiting GAN performance is the choice of a loss function to train the network. The loss function determines the discriminator's view of the system's effectiveness and provides the only source of feedback for the generator. Moreover, loss function choice is a tempting target for research because it is highly visible and presents interesting theoretical challenges, which we review below.

2.1. WGANs

Proposed in 2017, the Wasserstein GAN (WGAN) [1] addresses the shortcomings of the original JS-divergence-based loss function by instead utilizing the Wasserstein, or Earth-Mover (EM), distance (1).

$$W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[f(x)\right] \quad (1)$$

Unlike the JS divergence, the EM distance can measure the difference between any two distributions and give an interpretable gradient, and even reflect the relative distance between two completely distinct probability distributions. Consequently, the WGAN can theoretically guarantee convergence regardless of the initial probability distributions used, and results in a more stable training process with greatly reduced incidence of mode collapse, as supported by empirical results. In addition, the loss outputted by the EM distance can be used to measure the GAN's performance.

One drawback of using the Wasserstein distance is that we must ensure the discriminator is a Lipschitz function in order to compute its gradient. To enforce this constraint, the Wasserstein GAN clips the weights of the discriminator within a predetermined (hyperparametric) range. While this is effective, the WGAN paper notes that it is "a clearly terrible way to enforce a Lipschitz constraint" [1] and leaves open the possibility for improvements in this regard.

2.2. WGAN Gradient Penalty

One such improvement intended to solve the problem of enforcing the Lipschitz continuity of the discriminator function without clipping its weights is the so-called 'Improved WGAN', or WGAN Gradient-Penalty. [3] Instead of strictly clipping the weights of the discriminator, which severely limits its ability to represent more complex functions, WGAN-GP instead adds a penalty to its loss function, which encourages the weights to conform to the Lipschitz constraint. It does this by calculating the norm of the discriminator's gradient and penalizing values greater than 1. Doing so causes the discriminator's gradient updates to favor weight configurations which maintain the Lipschitz constraint, though it does not rigorously ensure that the discriminator will be a Lipschitz funciton.

2.3. Relaxed WGAN

Seeking to strike an acceptable balance between the benefits of WGANs and WGAN-GPs, the Relaxed Wasserstein GAN modifies the Wasserstein distance function by incorporating elements of the Bregman divergence. The R-WGAN algorithm continues to clamp the discriminator's weights, but unlike the original WGAN, it does so asymmetrically to control the volatility of the gradient. This allows the weights greater flexibility to capture more complex patterns, without violating the theoretical constraints of Lipschitz continuity. Because of this improved flexibility, the R-WGAN is better suited for the use of more sophisticated discriminator models, such as ResNet and DenseNet, thereby providing the potential for better performance[4].

3. Pix2Pix and State-of-the-Art cGANs

Our present work investigates use of conditional GANs for image transfer. Because of the complexity of the problem space and the conditioning constraint on the generator, image transfer is currently a much more significant target for optimization than handwriting recognition.

The Pix2Pix project [7] (and its derivatives such as Vid2Vid) currently represent a high-water mark in the task of image-to-image transfer. Pix2Pix's conditional GAN architecture, when properly trained, can address a wide variety of problems in mapping images in one form to another, such as a grayscale to colorized, incomplete to infilled, or edge-mapped to synthesized photograph.

Three key innovations have determined the success of this project. First, the generator is designed with a "U-Net" architecture (inclusive of skips between layers). This design augments a traditional bottleneck encoder-decoder with direct links between layers of equivalent size (so for instance the second convolutional layer of the encoder would link directly to the second-to-last deconvolutional layer of the decoder). The links preserve structural features from the input domain, gently nudging the generator toward images which match the input.

The second major innovation is in the loss function (2):

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (2)$$

The Pix2Pix design incorporates an L1 loss term (measuring the difference between the generated output and the conditional input) as part of the loss to be minimized. This loss

trains for overall similarity of the generated image to the target image, or "low-frequency" similarity. Previous work in cGANs incorporated an L2 loss; however the L1 loss term promotes long-range similarity without incentivizing localized blurriness, as an L2 term would do.

Finally, the Pix2Pix architecture incorporates what its authors term a "Markovian discriminator," which assumes the independence of pixels distant from one another in the generated images. This creates a windowing effect: the discriminator focuses on the authenticity of localized segments of the image (realized as style or texture differences) and relies on the L1 loss component to ensure the overall similarity of the two images. The resulting image is both locally and globally similar to the input domain.

These innovations produce very good results on a broad selection of cGAN image transfer tasks. The Pix2Pix framework centers several hyperparameters, as well as choice of loss function, which form the target of the present ablation study.

4. Present Work

Optimally configuring a GAN solution is a topic of ongoing research–especially for advanced applications like conditional GANs. Theoretical properties aside, a recent large-scale study from Google Brain [9] found choice of loss function to be less significant than hyperparameter tuning, given limited computing resources; other commentators [6] suggest this claim is overstated and merely shows that less favorable hyperparameters were used to start with for some architectures. Clearly more study is needed, along with tools that facilitate easy specification of hyperparameter-tuning experiments and classification of results.

4.1. Goals

Our present work represents three contributions:

- Developing a tool within TensorFlow which can be used to easily assess the impact of different loss functions, training times, and hyperparameters;
- Providing methods to use this tool with the advanced features offered by cloud environments like Google Colab; and
- Demonstrating results of some experiments modifying different training parameters.

To this end we present a tool which structures experiments over the hyperparameters (and loss function) used to train a Pix2Pix-based conditional GAN. Our submission also includes a notebook skeleton which sets up an environment in which this code (or other experimental code) can be run in Google Colab.

Within this framework we have tested four GAN loss functions: the Jensen-Shannon-based loss function in Goodfellow's original paper [2] (`minimax`), along with that paper's recommended adaptation (`modified`); the WGAN [1] loss function (`wgan`), and the loss function used in Pix2Pix (`pix2pix`). We have experimented with using this framework against the edgemap-to-image task using a 500-image subset of the edges2handbags dataset [13][14], and briefly with the semantic-map-to-image task using the facades dataset [11] (which is supported in our pipeline but will not be discussed in detail).

Parameter	**Default**
Loss function	Pix2pix
Minibatch size	1
L1 term weight	100.0
GAN loss weight	1.0
Optimizer learning rate	0.0002
Optimizer momentum	0.5
Image scaling size (for jitter)	10/

Table 1. Hyperparameter list

4.2. Methods

Our architecture adapts Christopher Hesse's [5] TensorFlow implementation of the Pix2Pix architecture. Our original experiments with the TFGAN library proved unworkable; however, we have augmented the Pix2Pix code with implementations of the above loss functions derived from Joel Shor's TFGAN implementations [12].

Unless otherwise specified, we augment every generator loss function with an L1 loss term as per Pix2Pix. (Weight of this loss term is a hyperparameter.) Thus, loss function selection is the loss function for the GAN generator and discriminator.

We have evaluated our various experiments quantitatively (by looking at trends in loss functions) and qualitatively, by comparing the images output by the cGAN at different stages of training. To facilitate this comparison, models were run with the same initial random seed (though this does not help with comparing across minibatch sizes). We acknowledge the considerable interest in quantitative evaluation metrics for the quality of generated images (such as FCN-score [8]), however pursuing these has been out of scope for the present study.

4.3. Hyperparameters

Table 1 shows the hyperparameters recognized by our architecture, along with their default values. Of these, we experimented with modifying the loss function and L1 weight, and briefly explored the effects of increased minibatch size. (We perform stochastic gradient descent by default.)

Experiments run are detailed in table 2. These obviously represent focused exploration rather than an exhaustive search of the hyperparameter space.

Run	Loss	L1 Weight	LR	Batch
A	Pix	100	0.002	1
B	Mini	100	0.002	1
C	Mod	100	0.002	1
D	Wgan	100	0.002	1
E	Pix	50	0.002	1
F	Mod	50	0.002	1
G	Wgan	50	0.002	1
H	Pix	0	0.002	1
I	Mod	0	0.002	1
J	Wgan	0	0.002	1
K	Pix	100	0.001	1
L	Wgan	100	0.001	1
M	Pix	100	0.01	1
N	Pix	100	0.002	4
O	Pix	100	0.002	8

Table 2. List of Experiments detailing loss function (Pix2Pix, Minimax/JS, Modified JS, Wasserstein), weight on the L1 long-range-loss term, Adam-optimizer learning rate, and minibatch size (1 for stochastic gradient descent).

For all datasets, input images were cropped to 128x128 pixels (after first downscaling to 140x140 in order to apply jitter, as per the Pix2Pix methods). Due to computational resource constraints, we trained on a small subset of the edges2handbags dataset (which contains approximately 160,000 images in its entirety); our training runs were 5000 steps (10 epochs for SGD); by contrast [7] trained on the full dataset for 15 epochs with minibatch size 4.

5. Results

We evaluate the various GAN models qualitatively based on the individual images, and quantitatively based on the loss functions.

5.1. General Trends

We note that nearly all of the models tested showed fairly good performance, fairly quickly (typical training time was on the order of 40 minutes). While we have not directly confirmed the results of the Pix2Pix paper (given the reduced data set with which we have worked), we have confirmed that results of the published quality are highly plausible even without prohibitively-large compute resources.

5.2. Loss Function and L1 Term

Figure 1 provides an overview of the GAN loss functions with various weights of L1 loss: **A-D** are Pix2Pix, Minimax, Modified, and Wasserstein loss at L1 $\lambda = 100$; **E-G** show Pix2Pix, Modified and Wasserstein loss at L1 $\lambda = 50$; and **H-J** are Pix2Pix, Modified, and Wasserstein loss with L1 term omitted. Images from throughout training are shown, on roughly 250, 500, 1000, 2000, 3000, 4000, 5000 steps (modified slightly to favor interesting comparisons over instances where the target image was particularly featureless). As Minimax results (**B**) were not substantially different from Modified, Minimax is excluded from later trials.

In contrast to the prediction of Arjovsky *et al*. [1] and our results in the milestone, the Wasserstein GAN model shows no clear advantage over the other models. The results in Fig. 1 (in which columns **D**, **G**, and **J** are WGAN models) show no particular advantage over any other loss function in early or late training; Minimax, Modified, and WGAN are all quite similar, and if anything, **D** shows more smudging, internal blurriness, and incomplete texture than **A-C**. The Pix2Pix loss function (columns **A**, **E**, and **H**) is notable for being somewhat less careful about edges and filling than the other models; however it generates a greater diversity of color and texture. Other models converge to a uniform brown, while the Pix2Pix-loss model shows more red and blue tones.

The most obvious result from this comparison is the importance of the L1 loss term in the Pix2Pix design. Quality suffers somewhat when the L1 term is given half weight; and declines dramatically when it is removed entirely (columns **H**, **I**, and **J**). While the Pix2Pix loss (**H**) performs better than the Modified or Wasserstein losses, performance is still dramatically worse than earlier models, with considerable background noise, until the end of the training process (10 epochs).

The loss function graphs bear out the importance of the L1 term for this application. As Fig. 2 shows, while the GAN losses across the board are fairly flat for all loss functions, most of the improvement in generator loss comes from the L1 term.

5.3. Learning Rate

Learning rate did not emerge as a particularly significant factor. Three trials were conducted considering learning rate: **K**, **L**, and **M**. The first two used a halved learning rate of 0.0001, with Pix2Pix and WGAN losses respectively; the latter used Pix2Pix loss with a learning rate of 0.001 (5x increased from baseline). Unsurprisingly, both models with a slower learning rate took slightly more steps before reaching plausibility, but the ultimate results were indeterminate by midway through the, training process, as Fig. 3 shows (compare to row 6 of Fig. 1). The Pix2Pix loss function would seem to be most sensitive to this change; trial **K** (Pix2Pix, halved rate, left) shows similar results to halving the L1 coefficient in that the palette has flipped to blue and black, but interestingly both doubling and halving the learning rate results in a less saturated image with more gaps. By this point in training, the Wasserstein loss model appears largely unaffected.

Figure 1. Image series for Loss Function, L1 experiments throughout training. Column labels correspond to trials listed in Table 2.

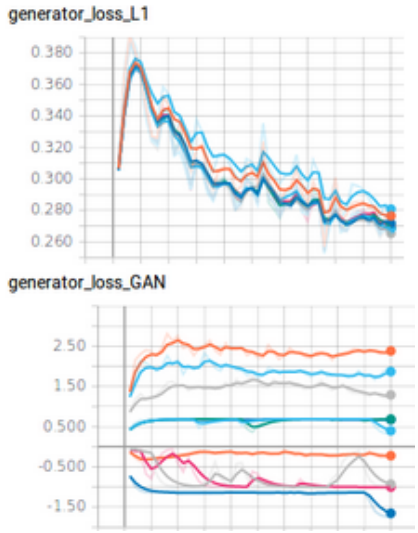

Figure 2. Generator losses for trials A-J, by component.

Figure 3. Step 3990, Trials K, L, M.

5.4. Batch Size

Finally, we examined varying minibatch sizes with the Pix2Pix loss function vs. the stochastic gradient descent approach used in other trials. Trials **N** and **O** represent this approach with minibatch sizes 4 and 8, respectively (larger batches were not tried due to time constraints). Results were generally similar to other trials for like total number of training examples viewed; unsurprisingly, larger minibatches led to more flexibility in recovering unusual colors and patterns. While the differences to the eye are subtle, increased minibatch size clearly improves loss: Fig. 4 shows that losses drop faster and lower for **O** (pink, minibatch size

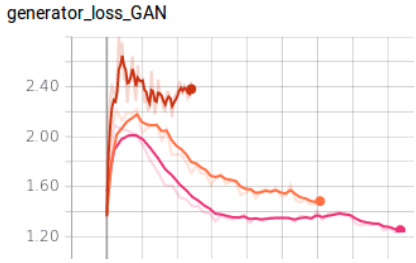

Figure 4. GAN Generator loss, trials A, N, O (from top to bottom), differing by minibatch size (1, 4, 8, respectively). Relative training scale.

8) and **N** (middle, orange, minibatch size 4) compared to default **A** (top, red), even at the same relative point in training. For a qualitative comparison, the first and fourth rows of Fig. 5 show typical results from trials **N** and **O** after these models have trained on roughly the same total number of images as at the end of training for the other trials.

Trials **N** and **O** also demonstrate that the architecture presented here will continue to improve with more training time (and most likely, more data); while we have not approached the 15 epochs of 16k images of the Pix2Pix paper, we note fairly good results after a mere hour and a half of training time on a desktop computer, as Fig. 5 shows. Using the full dataset and training longer is expected to produce results similar to the published Pix2Pix results.

Figure 5. First row per model is relative equivalent of end-of-training for other models. Results below show effectiveness of further training.

6. Conclusion

The results of this investigation are more mixed than our milestone paper and tend to support [9]'s assertion that hyperparameter settings matter, perhaps more than loss function. While our earlier investigations found that choice of loss function was decisive in how rapidly the model converged on human-like performance, there was no clear standout loss function in the present investigation. The most important loss for the cGAN architecture we studied was the L1 loss and its relative weight: much more than the localized loss. This is understandable, considering that in the Pix2Pix design, the GAN loss is meant to focus on local textures [7]. The image data set we studied has many common features (plain white backgrounds, low variation of texture, most fashion bags are brown or black), so this data set deemphasizes the importance of the choice of GAN loss function. It is possible that a different data set would have produced different results.

But this, too, supports [9]'s claim: for dataset is problem-specific, and if different datasets favor different loss functions, then there is no single loss function with an obvious universal advantage.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv:1701.07875v3*, 2017.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *arXiv:1406.2661*, 2014.

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[4] X. Guo, J. Hong, T. Lin, and N. Yang. Relaxed wasserstein with applications to gans. *arXiv:1705.07164v3*, 2018.

[5] C. Hesse. Image-to-image translation in tensorflow, 2017. https://affinelayer.com/pix2pix/.

[6] J. Hui. Gan, wasserstein gan, & wgan-gp, 2018. https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490.

[7] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[8] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[9] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly. Are gans created equal? a large-scale study. *arXiv:1711.10337v4*, 2018.

[10] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[11] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR*, Saarbrucken, Germany, 2013.

[12] J. Shor. Tfgan library, 2017. https://github.com/tensorflow/models/tree/master/research/gan.

[13] S. "Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015.

[14] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.