Effects of Gradient Penalty on Generative Adversarial Networks

Shalin Patel
Albert Nerken School of Engineering
The Cooper Union
patel10@cooper.edu

Abdullah Siddiki
Albert Nerken School of Engineering
The Cooper Union
siddiki@cooper.edu

Abstract

The training instability of Generative Adversarial Networks (GANs) has inhibited their success. The Wasserstein GAN (WGAN) made significant progress toward the stable training of GANs, but it still generated low-quality samples and failed to converge in some settings. A recently suggested alternative method of enforcing the Lipschitz contraint is to penalize the norm of the gradient of the critic with respect to its input. We attempted to adapt this gradient penalty to the traditional GAN objective function as opposed to the WGAN objective. Our results have evidence of increased training stability with the addition of the gradient penalty to the traditional GAN.

1 Background

1.1 Generative Adversarial Networks

The Generative Adversarial Network (GAN) is a framework proposed by Goodfellow et al that consists of two networks trained simultaneously, the generator G and the discriminator D. The network G is given some noise input and attempts to create data samples from the input space. The goal of D is to distinguish between generated samples and true data samples. The GAN is mathematically represented with the following minimax game:

$$\min_G \max_D \mathop{\mathbb{E}}_{x\sim\mathbb{P}_x} [\log D(x)] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z} [\log(1 - D(G(z))] \tag{1}$$

where $\mathbb{P}_x$ is the data distribution, $\mathbb{P}_z$ is the input noise distribution[4].

Training D optimally prior to an update to G makes our minimization of the minimax value function a minimization of the Jensen-Shannon divergence between the data and model distributions on x. This process is costly and can lead to vanishing gradients if the discriminator begins to overfit. Thus, its better to alternate between *k* steps of optimizing D and one update to G. However, this causes generator updates to minimize a stochastic lower bound to the Jensen-Shannon divergence [3]. Lowering the lower bound doesn't necessarily mean the loss is being minimized. As a result, there could be meaningless gradient updates. This leads to instability when training the GAN [2].

1.2 Wasserstein GANs

Arjovsky et al. claim the the Jensen-Shannon Divergence [2] may not be continuous, thus is an unfit gradient for G.

The Earth-Mover $W(g, p)$ distance is proposed as an alternate to the Jensen-Shannon Divergence. Essentially, it is the minimum cost of moving of transporting mass in order to transform the distri-

bution q onto the distribution p [5]. This Earth-Mover distance is desirable because continuous and differentiable virtually anywhere given some constraints.

The value function for the Wasserstein GAN is obtained from the Kantorovich-Rubinstein duality:

$$\min_{G} \max_{D \epsilon \mathcal{D}} \mathop{\mathbb{E}}_{x \sim \mathbb{P}_x} [D(x)] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [D(G(z))] \tag{2}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions. Minimizing the value function based on the generator's paramets minimizes the Earth-Mover distance between $\mathbb{P}_x$ and $\mathbb{P}_z$.

The WGAN's value functions allows for much more stable training of the WGAN compared to the GAN. Also, the WGAN's value function correlates with output quality while the GAN's value function does not.

1.3 Issues with weight clipping

Weight clipping in WGANs have shown to lead to optimization difficulties, and even when optimizing successfully, the resulting critic can have an unpredictable value surface. Since optimizing the critic with constricted weights is only an approximation that is equivalent to optimizing iver a small set of k-Lipschitz functions, for many problems, even when the optimization does converge, the solutions converged to provide very different gradients from the optimal one. This is because the optimal critic loss according to the WGAN has gradient with norm 1 almost everywhere. Under weight clipping constraint, most neural networks implement a k-Lipschitz constraint, which biases the critic towards simpler functions.

Other issues with weight clipping include exploding and vanishing gradients. Depending on the value of the clipping threshold c, interactions between the weight constraint and the cost function inevitably results in this issue. If the weights are constricted to be too small, earlier layers in the critic are prevented from receiving a useful training signal, with the gradient vanishing as we backpropagate, which can lead to slow learning. If the weights are not constrained enough, since the objective training encourages weights to lie at the ends of the allowed range, we see exploding gradients. Others like Arjovsky et al [1] have tried to use batch normalization in the critic to mitigate these issues, but WGAN critics can still fail to learn with these changes.

1.4 Gradient Penalty

As discussed in prior sections, while weight clipping is somewhat effective in stabilizing GAN training, it may result in some undesirable behaviors. An alternate method proposed to stabilize training is adding a gradient penalty, first implemented in Gulrajani et al [5]. This alternative method enforces the 1-Lipschitz constraint on the training objective function by directly constraining the gradient norm of the critic function with respect to its input. They enforce of a soft version of the constraint at only certain points sampled from a distribution over the input space $\widetilde{x} \sim \mathbb{P}_{\widetilde{x}}$. The gradient of the critic is evaluated and its squared distance from is penalized in the critic loss function. This penalty is added to the original critic loss and is formulated as

$$P = \lambda \times \mathop{\mathbb{E}}_{\widehat{x} \sim \mathbb{P}_{\widetilde{x}}} \left[\left(\left\| \nabla_{\widehat{x}} D(\widehat{x}) \right\|_2 - 1 \right)^2 \right] \tag{3}$$

The new cost function, given enough capacity on the critic helps to recover the true Wasserstein distance, which does not necessarily happen in WGANs due to clipping.

The points $\mathbb{P}_{\widetilde{x}}$ at which the gradient is penalized is defined by taking straight lines between points in the data distribution $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$:

$$\in \sim U[0,1], x \sim \mathbb{P}_r, \widetilde{x} \sim \mathbb{P}_g \tag{4}$$

$$\widehat{x} = \epsilon x + (1 - \epsilon)\widetilde{x} \tag{5}$$

Enforcing the Lipschitz constraint along these lines seemed sufficient and experimentally resulted in good performance, as enforcing the Lipshitz constraint everywhere is intractable. The only hyperparamater introduced by this penalty is λ, which controls trade-off between optimizing the

penalty term and original objective. Batch normalization is omitted in the critic because the penalized training objective using batch normalization in the discriminator changes the form of the problem: instead of specifying a function mapping a single input to a single output, a discriminator with batch normalization specifies a function mapping from an entire batch of inputs to a batch of outputs, invalidating the penalized training objective. Adam optimizer is used in this method because it allows for momentum based optimizers to be used since there is no weight clipping.

2 Experiments

In this section we show the effects of adding gradient penalty to a standard GAN critic loss function trained on the MNIST dataset. We use a slightly altered set of constraints on the architectural topology of Convolutional GANs known as Deep Convolutional GANs (DCGAN) as in Radford et al [6]. The guidelines outlined by their paper are as follows:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

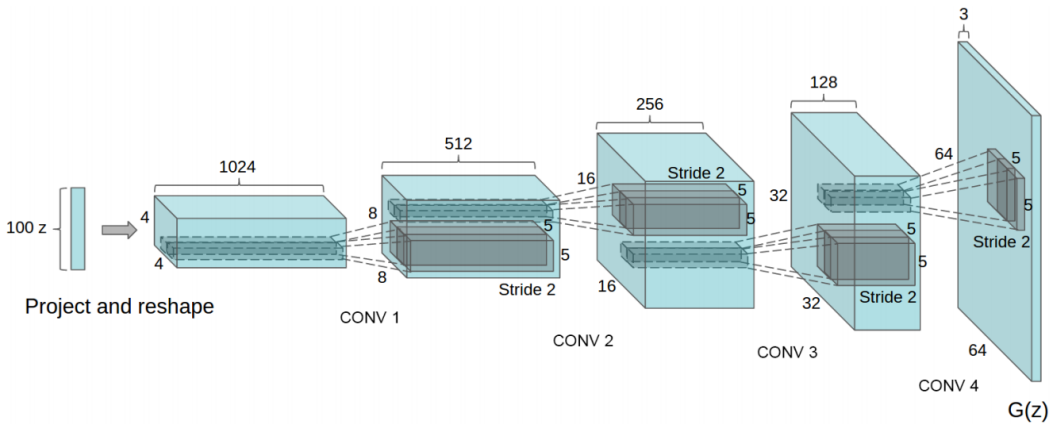

Figure 1: Diagram of the DCGAN generator from [6].

As discussed in section 1.4 of this paper, gradient penalty does not have the desired effect on architectures with batch normalization. For our experiments we remove the batch normalization in both the generator and discriminator, which is an architecture that is notably more difficult to train. Experiments are performed first without gradient penalty and then with gradient penalty of various λ. Critic loss and sample quality are compared. Code is adapted from the code associated with [6][1] and can be found in our git repository[2].

2.1 Results

Below are some of our samples from running the GAN, with and without gradient penalty.

[1] https://github.com/carpedm20/DCGAN-tensorflow
[2] https://github.com/sSsXzZz/DCGAN-tensorflow

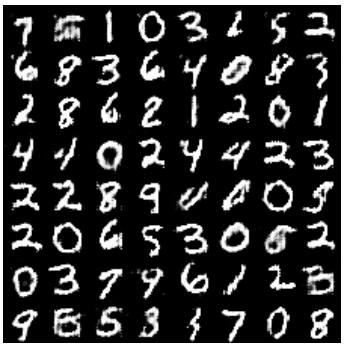

Figure 2: No Gradient Penalty

Figure 3: λ=1

Figure 4: λ=10

The samples with the gradient penalty generally look better than the ones without. For example, the 5's in Figure 2 are much more noisy than the ones in Figure 3 and Figure 4. Also, there is less variance in the samples without the gradient penalty. Many of the 2's in Figure 2 look alike.

3 Conclusion

We take the gradient penalty previously only applied to the Wasserstein GAN and apply it to the standard GAN objective function. We show that gradient penalty increases the quality of image samples generated from training on the MNIST dataset. Future work can be done to better grasp the extent of the effects of gradient penalty on the standard GAN objective. Finer tuning of the hyperparamater λ can be done to find the best trade off between the standard GAN objective and the gradient penalty term. Further work could also include adding the gradient penalty term to other difficult to train GANs such as architectures with constant number of filters in the generator, a 4-Layer 512-dim ReLU MLP generator, added nonlinearities, etc., and comparing image quality from the baseline without gradient penalty.

4 Acknowledgements

We would like to thank professor Chris Curro, adjunct professor at The Cooper Union, for guiding us in implementing our model.

References

[1] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: 1701.04862.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875.

[3] Ian J. Goodfellow. *On distinguishability criteria for estimating generative models*. 2014. arXiv: 1412.6515.

[4] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661.

[5] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028.

[6] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. arXiv: 1511.06434.

Appendix

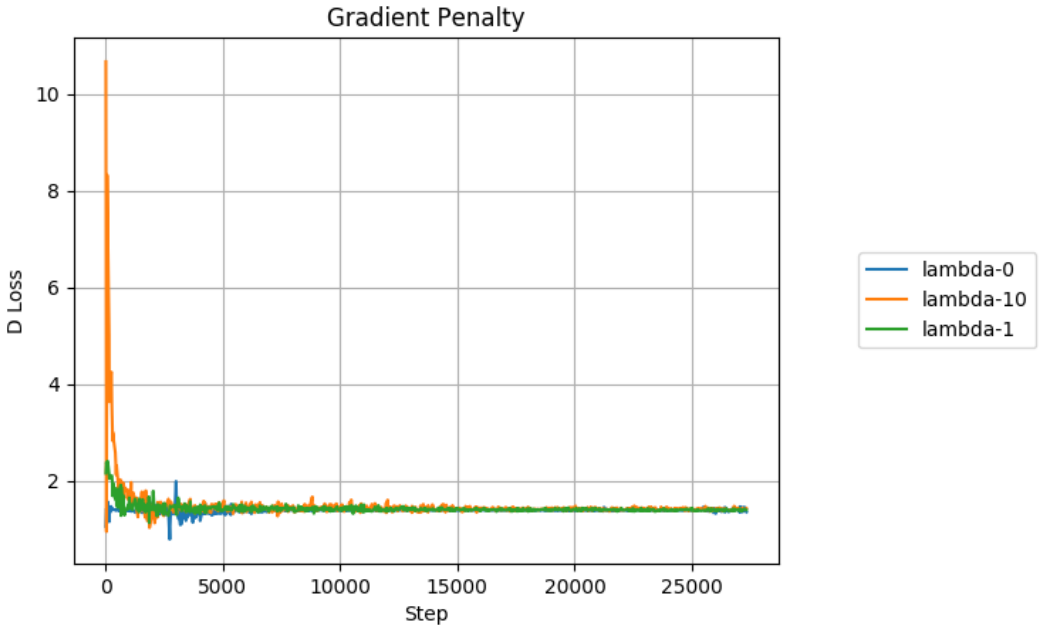

Figure 5: Discriminator Loss

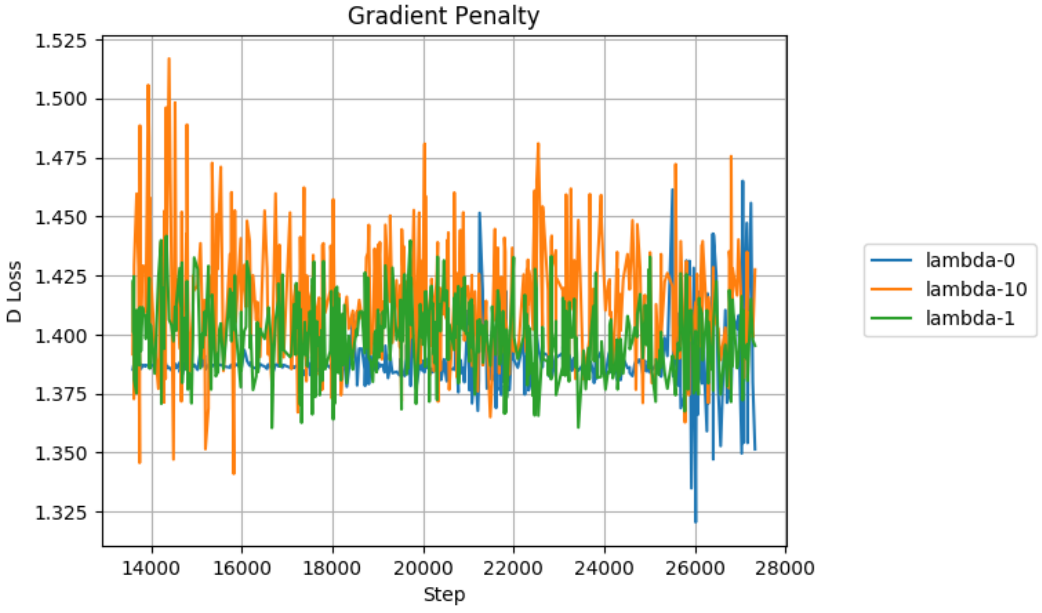

Figure 6: Discriminator Loss Zoomed