Recent Advances in GANs

Team members: Quinn Frank and Cole Juracek

Abstract

Generative Adversarial Networks (GANs) are a powerful class of neural network architectures designed to generate images comparable to those in its training set. While initially only competitive with other top algorithms at best, GANs have since evolved the ability to create photo-realistic images that have successfully crossed the threshold of human discrimination in a variety of domains. Training GANs is often difficult for several reasons, however, with little indication of what is wrong upon failure. Various regularization techniques have been proposed to assist in training, and we investigate one known as the Wasserstein GAN (WGAN). In this paper, we implement and compare results between a vanilla GAN and the WGAN on the CIFAR-10 dataset.

1 Introduction

The original work on GANs started with Goodfellow et al. in 2014. In this paper, Goodfellow proposed a new method for estimating generative models via an adversarial process involving 2 networks: the *generator* G and the *discriminator* D. The generator's job is to estimate the underlying data distribution p_{data} and fool the discriminator into thinking its images are legitimate. The discriminator's job is to distinguish real images from generated ones, which it expresses via a probability. Specifically, $P(x)$ denotes the probability the discriminator thinks an image is real. Mathematically, we can imagine D and G playing the following minimax game over the following value function $V(D, G)$:

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)}[log(D(\mathbf{x}))] + E_{z \sim p_z(z)}[log(1 - D(G(\mathbf{z}))))]$$

First, D optimizes the value function. If D is performing well, then it should assign high probability to actual training data (large $D(\mathbf{x})$) and small probability to generated images (small $D(G(\mathbf{z}))$). After updating the discriminator, the generator then tries to *minimize* the same function. The discriminator is not involved in the first term - it can only make the second term as small as possible, meaning it wants to fool the discriminator into thinking its generated images are legitimate (large $D(G(\mathbf{z}))$). Given enough time and computation resources, G should theoretically be able to completely approximate the data's distribution ($p_g = p_{data}$), and the discriminator will be unable to distinguish real images from fake ones beyond random guessing ($D(\mathbf{x}) = 0.5$).

1.1 Motivation and importance of the problem

Unfortunately, GANs and their variations are harder to train than traditional neural networks. The traditional method of investigating loss curves does not work as there is no "test set", and so it is hard to tell when to stop training. Furthermore, because the losses of D and G directly play off each other, the training losses will be much less stable and may fluctuate more than normal.

Inspecting the images visually may work, but this approach does not help when the network isn't training correctly and the output looks like noise. Additionally, qualitative assessment leaves room for interpretation and doesn't allow us to concretely say if a generated image is "better" or "worse".

A novel problem known as *mode collapse* is also unique to GAN training. We feed in random noise to the generator, which then forms an image that is likely to be distinct from all the other generator images. Mode collapse occurs when we get very similar images regardless of the random noise passed in. The generator has essentially found a weak point in the discriminator. The discriminator can of course learn to reject these, but then the generator can shift to a different set of vulnerable points and the process repeats.

There is also the issue of handling 2 networks training against each other. Typically we train and update the discriminator over a fixed number of batches, and then update the generator once. But this is another hyperparameter to choose in advance - if we don't train the discriminator enough before a generator update, we risk instability and mode collapse. But if we update the discriminator too much before the generator, then the generator will never fool the discriminator and will obtain vanishing gradients.

Various methods have been proposed to mitigate the difficulty and instability of GAN training. In section 2 we will investigate the Wasserstein GAN, a lightweight method of modifying GAN training that leaves the original architecture almost completely intact.

2 Related works

The original GAN was applied to the MNIST dataset. However, in our problem we are working with the CIFAR-10 dataset, containing images with 3 color channels. Convolutional neural networks (CNNs) have had great success with images in this format, and GANs are no different. The *Deep Convolutional Generative Adversarial Network* (DCGAN), introduced by Radford et al. in 2016, has adapted the successful approach of using convolutions in neural networks and applied it to this problem. In addition to the convolutional architecture described by the paper, it also proposed a set of useful guidelines for future GAN implementations. Due to the success of the DCGAN over the original GAN, the term "vanilla GAN" will refer to the DCGAN implementation for image generation.

We now introduce the WGAN. First proposed by Arjovsky et. al in 2017, the WGAN utilizes a new loss function with better theoretical properties to be used for training the networks. Specifically, they measure the *Earth-Mover* or Wasserstein-1 distance between the data's distribution P_r and the generator's learned distribution P_g, given by:

$$W(P_r, P_g) := \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}[||x-y||]$$

with $\Pi(P_D, P_g)$ the set of all possible joint distributions induced by the 2 arguments given as marginals. So essentially this cost is under the "optimal transport plan" (smallest γ).

Unfortunately, this infimum is intractable. However, we can invoke the Kantorovich-Rubinstein duality, which states that we may rewrite the EM-distance as:

$$W(P_r, P_\theta) = \sup_{||f||_L \leq 1} E_{x\sim P_r}[f(x)] - E_{x\sim P_\theta}[f(x)]$$

With $||f||_L \leq 1$ indicating all 1-Lipschitz continuous functions. To use this loss function, we utilize both real data ($x \sim P_r$) and generated data ($x \sim P_\theta$), feed into the generator ($f(x)$), and take the sample mean (in place of expectation). To enforce the Lipschitz constraint, the author's propose "weight-clipping": simply clamping all of the discriminator's weights to be in $[-c, c]$ after each gradient update. However, the authors also claim that "weight clipping is a clearly terrible way to enforce a Lipschitz constraint" (Arjovsky et al., 2017). Thus this leaves the door open for other possibilities.

This work was built upon by Gulrajani1 et al. in the 2017 "Improved Training of Wasserstein GANs". The authors take an alternative approach to enforcing the Lipschitz constraint. Instead of clipping the weights after gradient updates, a penalty term on the gradient of the discriminator with respect to its input is added to the loss (*gradient penalty*), producing better results with minimal hyperparameter tuning. This penalty is given by:

$$L = E_{\tilde{x}\sim P_g}[D(\tilde{x})] - E_{x\sim P_r}[D(\mathbf{x})] + \lambda E_{\hat{x}\sim P_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$$

The loss is identical in all but notation to the one above, except with the addition of the gradient penalization in place of weight clipping. Note that $\hat{x}$ is a mixture between the data distribution and the generator distribution.

3 Details of the project

Note: For exact specifics of the architecture, see the code on GitHub `https://github.com/quinnfrank/GAN-regularization`

We now discuss the architecture of the GANs that we have implemented. For the DCGAN discriminator the goal is to translate random noise sampled from a standard multivariate normal into images of size (3x32x32). We accomplish this with a series of convolutional transpose layers (no pooling) to upsample coupled with batch normalization and ReLU activations on all but the last layer. A final Tanh activation is used before these images are ready to be judged by the discriminator.

The discriminator is tasked with taking images of size (3x32x32) and outputting a probability that the image is real. A series of convolutions is applied to an image to reduce the size (without pooling), then batch normalization and Leaky ReLU activations are used on all but the first and last layers. Finally, a sigmoid activation is used to squash the values between 0 and 1 giving the image probabilities.

The first WGAN model is implemented practically the same way - we simply remove the sigmoid output layer from the discriminator as the Wasserstein loss is not constrained to be in $[0, 1]$. Aside from that, there are 2 notable changes. First is the previously mentioned use of the Wasserstein loss over the traditional minimax problem for vanilla GANs (see section 2 for theory). Next is enforcing the Lipschitz constraint via weight clipping, crudely accomplished by clipping all of the discriminator's weights to be bounded in $[-0.01, 0.01]$ after a gradient update step. The generator functions exactly as before.

The improved WGAN model has exactly the same architecture as the above WGAN, only with a slightly modified loss. To enforce the Lipschitz constraint for the improved WGAN, we sidestep weight clipping and instead add a penalization term to the Wasserstein loss based on the gradient of the discriminator with respect to its inputs.

Training for all models, as is typical in deep learning, is done through batch gradient descent/ascent on a specified loss function. Generated images with noise from the standard normal along with real data are used to update the discriminator, potentially several times, and then the generator is updated to improve against the new discriminator.

We have implemented the *Frechet Inception Distance* (FID) as a way to quantitatively measure image quality. To calculate this, both real and generated images are passed through a pre-trained Inception v3 network used for object detection. Using these results, we then take statistics from both groups and calculate the *Frechet* distance between them. Thus, a lower FID score indicates a better generated image, with 0 being a perfect score.

Finally, we investigate the structure of the latent space via interpolation. By linearly interpolating between 2 points in the latent space fed to the generator, we can theoretically perform vector arithmetic or morph between images. We will show how smoothly transitioning between 2 vectors in the latent space produces a smooth transition between 2 generated images.

3.1 Contribution of each member of the team

Cole: I worked on implementing the vanilla GAN, although we ultimately ended up using Quinn's implementation due to better performance. I also worked on both versions of the WGAN and writing/compiling the report.

Quinn: I wrote most of the code which ultimately ended up building our models, including the DCGAN architecture, the FID calculation, the vanilla GAN training procedure, and the model implementing WGAN with gradient norm penalization. I contributed to the WGAN with weight clipping, though Cole had worked on that previously. I also created the slides for our presentation.

4 Experimental results

4.1 CIFAR-10 Data

For the project, we opted to work with the CIFAR-10 dataset. CIFAR-10 consists of 70000 (3x32x32) color images (with the last 10000 used for testing). The images span 10 evenly balanced categories, hence the name: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

4.2 Hyperparameters

Various hyperparameters across our models can be found below. "DG Ratio" describes the number of discriminator updates per generator update. We tried to reproduce the parameters described in the original papers as closely as possible, but also experimented with other choices as well:

	Number of Epochs	Batch Size	Learning Rate	Optimizer	DG Ratio
Vanilla GAN (DCGAN)	40	128	0.0002	Adam	1
WGAN	100	64	5e-5	RMSprop	5
Improved WGAN	100	64	0.0001	Adam	5

(We also use a value of $\lambda = 10$ for the improved WGAN gradient penalty as described in the paper).

We now move to discuss the results of our training. For each model, we show the loss curves and a sample of generated images.

4.3 Vanilla GAN

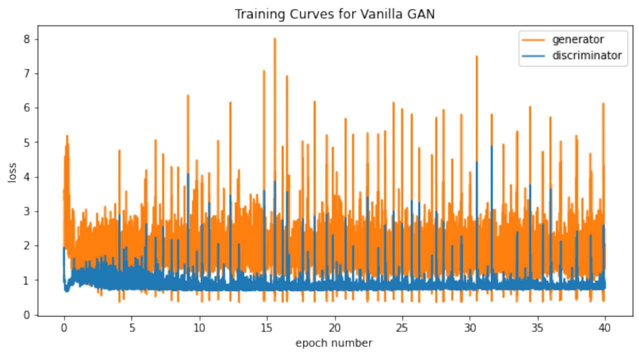

Figure 1: Vanilla GAN Loss Development

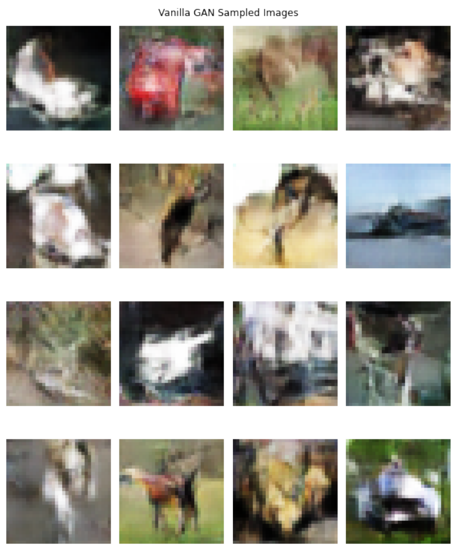

Figure 2: Sampled Images from Vanilla GAN

4.4 WGAN

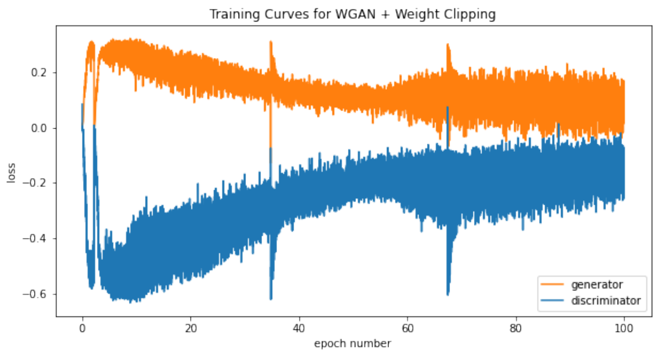

Figure 3: WGAN Loss Development

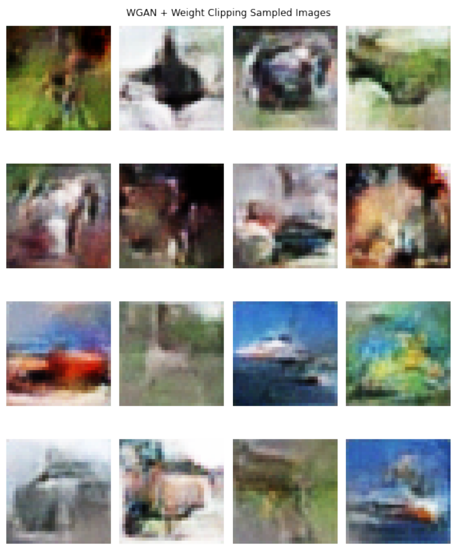

Figure 4: Sampled Images from WGAN

4.5 Improved WGAN

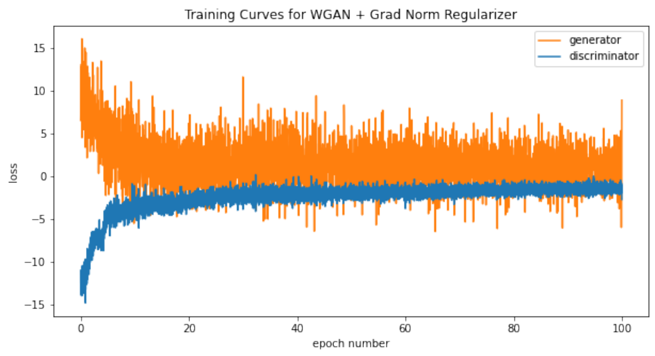

Figure 5: Improved WGAN Loss Development

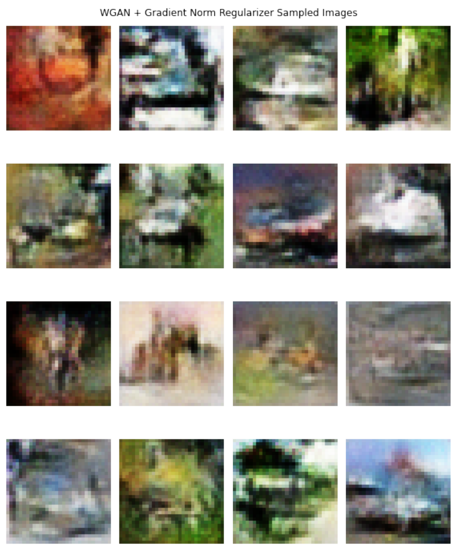

Figure 6: Sampled Images from Improved WGAN

While we don't see a drastic improvment in image quality on the 2 WGAN's compared to the vanilla GAN, the images look to be at least competitive. Furthermore, we see much more stable training via investigation of the loss curves compared to the vanilla GAN training, which behaves quite erratically. Note that we cannot compare losses directly, as they are all different.

Across all models, images are on the border of being recognizable. Clearly some are "animal looking" and the GANs also appear to capture realistic environments (grass, water, sky, etc.). With more computing power, I believe we could have iterated more quickly and obtained images more comparable to the various papers, although it is impressive to see images move past "just noise".

Aside from comparing the generated images visually, it is better to have a metric for evaluating performance. As previously discussed, we have implemented the FID metric for comparing the 3 models:

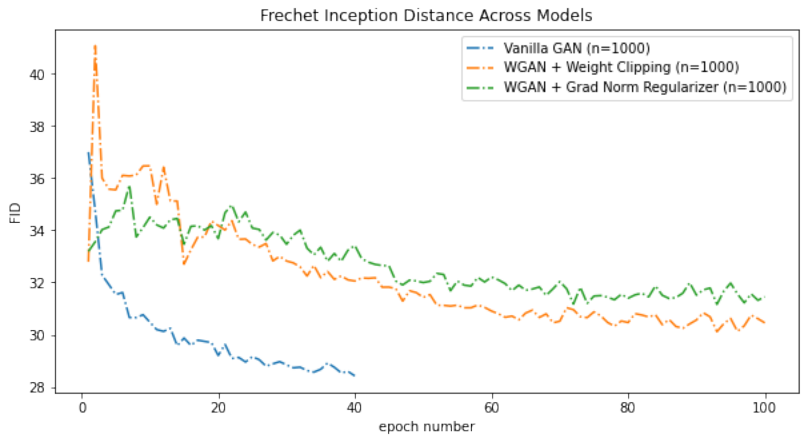

Figure 7: FID between Models

Finally, we show the results of our interpolation in the latent space:

The results are promising. While it's hard to tell exactly what labels the 2 original images could be (rightmost is likely a frog), we very clearly see a smooth transition from one to the other.

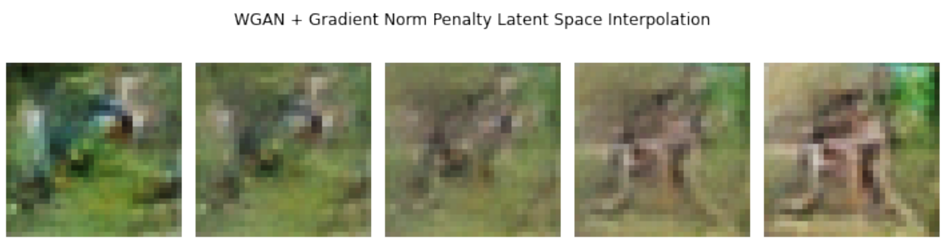

Figure 8: Caption

5 Concluding remarks

5.1 Limitations / Future Work

While the WGANs did appear to stabilize the training and avoid mode collapse as promised, there seems to be a slight decrease in image quality compared to the vanilla GAN. Furthermore, the FID graph suggests that the vanilla GAN is able to converge faster (40 epochs vs. 100 epochs). The papers suggest that comparable / better image quality can be obtained with these techniques. If we had more computation power, I believe we could have iterated more quickly on training different architectures to achieve better performance across all models.

There are also a large number of other proposed methods for stabilizing GAN training. *Spectral normalization* follows from the same idea of the WGAN but instead enforces the Lipschitz constraint by normalizing the spectral norm of the generator's weights. *Stable rank normalization* is another related, more general technique that normalizes not only the spectral norm, but also the stable rank of a matrix.

Different architectures can be used as well. One such popular idea is the *conditional GAN*, which allows for conditioning on a class label to generate class-specific images. This type of GAN is also thought to stabilize the training.

5.2 Conclusion

GANs have surpassed novel image generation and are now able to generate images indistinguishable from real ones across a variety of domains. Their impressive performance is often hindered by a variety of training difficulties, however. Poor image generation can be caused by several factors, and its hard to pinpoint exactly where the training process is going wrong.

In this paper we explored variations on the Wasserstein GAN, which seeks to stabilize traditional GAN training through utilization of a different loss function based on the Earth Mover's distance. WGANs promise easier training and avoidance of common problems in GAN, namely mode collapse.

3 models were trained and compared on the CIFAR-10 dataset. Results are promising - all 3 models are clearly capable of producing images that are far from noise, and most vaguely resemble one of the 10 labels in the dataset as well. The hypothesis that WGANs help stabilize training is seen visually via inspecting the loss curves. We also explored other results such as comparison through the Frechet Inception Distance and linear interpolation in the latent space.

Acknowledgments

Thank you to PyTorch for the fantastic documentation and tutorials, as well as to the curators of the CIFAR-10 dataset. A special thanks to the entire teaching staff of ECE 685 as well for a great semester despite the circumstances.

References

[1] Goodfellow, Ian J., et al. "Generative Adversarial Networks." ArXiv:1406.2661 [Cs, Stat], June 2014. arXiv.org, http://arxiv.org/abs/1406.2661.

[2] Radford, Alec, et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” ArXiv:1511.06434 [Cs], Jan. 2016. arXiv.org, http://arxiv.org/abs/1511.06434.

[3] Arjovsky, Martin, et al. “Wasserstein GAN.” ArXiv:1701.07875 [Cs, Stat], Dec. 2017. arXiv.org, http://arxiv.org/abs/1701.07875.

[4] Gulrajani, Ishaan, et al. “Improved Training of Wasserstein GANs.” ArXiv:1704.00028 [Cs, Stat], Dec. 2017. arXiv.org, http://arxiv.org/abs/1704.00028.