Wasserstein Generative Adversarial Networks and Their Applications

Kyle Astroth

I. Introduction

In 2014, Ian Goodfellow and his colleagues proposed the machine learning architecture of the Generative Adversarial Network (GAN), a generative model that can produce images, text, or music. [1] In 2016, Tim Salimans alongside Goodfellow and other OpenAI researchers responded to the initial GAN paper with a paper on improved techniques for training GANs that discussed methods to stabilize training and encourage convergence. [2] This research directly led to the development of the Wasserstein GAN (WGAN) and subsequent papers by Martin Arjovsky and his colleagues. [3][4] The Wasserstein GAN aimed to reduce mode collapse and create more stable training than the original GAN. Today, there are many interesting applications of Wasserstein GANs including data augmentation for emotion recognition [5] and singing voice synthesizers [6].

II. Problem Statement

While the initial Generative Adversarial Network architecture was groundbreaking in the machine learning community, it was challenging to train and faced several issues such as hard to achieve nash equilibrium, vanishing gradient, mode collapse, and lack of an evaluation metric. These are important terms when discussing GANs and can be defined as follows. Nash equilibrium is a game theory concept where the optimal outcome of a game is where there is no incentive to deviate from the initial strategy. The vanishing gradient problem occurs when the gradient gets smaller as we move backwards through the hidden layers of a neural network. This means that neurons in the earlier layers learn much slower than neurons in later layers. Mode collapse occurs in generative models when the generator can only produce a single type of output or a small set of outputs. GAN evaluation is still a discussed topic, and there is no one agreed upon performance metric for comparing models. The Wasserstein GAN attempted to address some of these problems with the Wasserstein Distance, and it is important to define the three relevant metrics for quantifying the similarity between two probability distributions: Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence, and Wasserstein distance. KL divergence measures how one probability distribution p diverges from a second expected probability distribution q, and achieves the minimum when $p(x) == q(x)$ everywhere. JS divergence measures the similarity between two probability distributions, but is bounded by [0, 1] and is symmetric rather than asymmetric. The Wasserstein, or Earth Mover's, distance is a measure of distance between two probability distributions. It describes the least distance, or

minimum cost, of moving one pile of "earth" – a certain distribution – to form a different pile of "earth" – another distribution. See Table 1 for the relevant equations, and Figure 1 for a visualization of Wasserstein distance. Finally, it is also important to understand what it means for a function to be K-Lipschitz continuous when discussing WGANs. A real-valued function $f: \mathbb{R} \rightarrow \mathbb{R}$ is considered K-Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in \mathbb{R}$:

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

III. Survey of Methods

Generative Adversarial Networks are based on Game Theory and mimic a minimax two-player game. The architecture consists of simultaneously training two models: a generative model, G, and a discriminative model, D, that estimates the probability that a sample came from the training data rather than G. [1] A popular metaphor is of an art forger, the generator, and an art critic, the discriminator. The generator and the discriminator can be defined by multilayer perceptrons, with G generating samples by passing a random noise vector through a multilayer perceptron, and the system can be trained with backpropagation. To learn the generator's distribution p_g over data x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(x; \theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(x; \theta_g)$ that outputs a single scalar. *D(x)* represents the probability that *x* came from the data rather than p_g. We train *D* to maximize the probability of assigning the correct label to both training examples and samples from *G*, while simultaneously training *G* to minimize *log(1 - D(G(z)))*. This gives us the following loss function to optimize [1]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

In practice, the objective function may not provide a sufficient gradient for *G* to learn well, so when *G* is poor, *D* can reject samples with high confidence. In this case, rather than training *G* to minimize *log(1 - D(G(z)))*, *G* can be trained to maximize *log D(G(z))* to provide much stronger gradients early in learning without altering the fixed point of the dynamics of *G* and *D*. In Theorem 1 of their paper, Goodfellow et al. prove that this minimax game has a global optimum for $p_g = p_{data}$, and in Proposition 2 they demonstrate that if *G* and *D* have enough capacity, and at each step of the loss function *D* is allowed to reach its optimum given *G*, and p_g is updated to improve this criterion:

$$\mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data} [1]. See Figure 2 for the proposed GAN algorithm.

In *Improved Techniques for Training GANs*, Salimans et. al use new techniques to improve the performance of the original GAN by encouraging convergence. These techniques include: feature matching, minibatch discrimination, historical averaging, one-sided label smoothing, and virtual

batch normalization. Feature matching addresses the instability of GANs by specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate data that matches the statistics of the real data. [2] In this scenario, the new loss function can be defined as:

$$||\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\mathbf{f}(G(\boldsymbol{z}))||_2^2.$$

Minibatch discrimination is a strategy to avoid the generator collapsing to a parameter setting where it always emits the same point by allowing the discriminator to look at multiple data examples in combination, rather than isolation. Each minibatch, the closeness between every pair of samples $c(x_i, x_j)$ is approximated to find the overall summary of one data point: $o(x_i) = \Sigma_j c(x_i, x_j)$. $o(x_i)$is then explicitly added to the input of the model. [2] When applying historical averaging, each player's cost is modified to include a term $||\theta - \frac{1}{t}\Sigma_{i=1}^{t}\theta[i]||^2$, where $\theta[i]$is the value of the parameters at past time i. This addition penalizes the training speed when θ is changing too dramatically in time. [2] One-sided label smoothing replaces the 0 and 1 targets fed to the discriminator with smoothed values such as 0.9 and 0.1 because it has been shown to reduce the vulnerability of neural networks to adversarial examples. [2] Finally, with virtual batch normalization each example x is normalized based on the statistics collected on a reference batch of examples chosen at the start of training and x itself. This solves the issue of batch normalization causing the output of a neural network for an input example x to be highly dependent on several other inputs x' in the same minibatch. By applying these techniques, the authors were able to achieve state-of-the-art results with a model that generates MNIST samples indistinguishable from real data by humans and CIFAR-10 samples with a human error rate of 21.3%. [2]

Following these two notable papers on GANs, Arjovsky et al. proposed Wasserstein Generative Adversarial Networks as an alternative to traditional GANs that improve the stability of learning, solve mode collapse, and provide meaningful learning curves that are useful for debugging and hyperparameter searches. [3] Their paper contains extensive theoretical work highlighting the connections to different distances between distributions, and while the previous GAN paper touched on semi-supervised learning, the WGAN paper focuses on unsupervised learning. In the loss function for the original GAN, it has been shown that the minimization of the Jensen-Shannon divergence:

$$JS(p^*, p^\theta) = \frac{1}{2}\left(KL(p^*||\frac{p^* + p^\theta}{2}) + KL(p^\theta||\frac{p^* + p^\theta}{2})\right)$$

is induced between the real distribution p^* and the generator distribution p^θ. In Theorems 1 and 2, it is proven that the JS divergence is not a "sensible" cost function when learning distributions supported by low dimensional manifolds, but the Wasserstein distance is "sensible." This is because even when two distributions are located in lower dimensional manifolds without overlaps, the Wasserstein distance can still provide a meaningful and smooth representation of the distance. [3] Thus, a transformation of the GAN loss function based on Kantorovich-Rubinstein duality is proposed:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

Where the function f of the new loss function must satisfy $||f||_L \leq K$, or that it should be K-Lipschitz continuous. This results in the discriminator being trained to learn a K-Lipschitz continuous function to compute the Wasserstein distance, rather than sorting real from fake samples. Thus, the discriminator is referred to as a 'critic' since it is no longer trained to classify. Further, we see that if we have a parameterized family of functions $\{f_w\}_{w \in W}$ that are all K-Lipschitz for some *K*, the modified GAN loss function becomes:

$$W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

as proved in Theorem 3. [3] The WGAN algorithm is shown in Figure 3. The remaining problem concerning WGANs is maintaining K-Lipschitz continuity of f_w during the training so all parameters w lie in a compact space. The authors' simple solution is to clamp the weights to a fixed box after each gradient update. [3]

However, as discussed in *Improved Training of Wasserstein GANs* by Gulrajani et al., there are issues that arise due to the use of weight clipping to enforce a Lipschitz constraint on the critic. Weight clipping in WGANs leads to optimization difficulties, so an alternative is proposed: penalizing the norm of the gradient of the critic with respect to its input. [4]

IV. Conclusion

As WGANs are still an open research topic, there have been many fascinating research papers and developments since their introduction in 2016. In 2018, research by Yun Luo and Bao-Liang Lu on data augmentation using a WGAN was published by IEEE. [5] Their paper introduced a Conditional WGAN framework for electroencephalography (EEG) data augmentation that significantly improved the accuracies of emotion recognition models – which typically suffer from low accuracy due to lack of EEG data. In 2019, IEEE published work by Chandna et al. that presented a generative model for singing voice synthesis that was trained using the WGAN framework. [6] Some interesting continued reading on WGANs can be found in *Wasserstein GANs Work Because They Fail (to Approximate the Wasserstein Distance)*, a 2021 paper by Cambridge researchers that examines how the theoretical foundations and the practical implementation of WGANs fundamentally differ. It argues that for good WGANs, the loss function typically does not approximate the Wasserstein distance well, and that good generators do not produce high quality images despite a poor approximation by the discriminator, but *because* of it. [7] GANs, and WGANs, are an exciting, impactful, and still relatively new field of Machine Learning research that will continue to be studied and improved upon in the years to come.

V. Figures

Table 1

Comparing Two Probability Distributions	
Jensen-Shannon (JS) Divergence	$D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})$
Kullback-Leibler (KL) Divergence	$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$
Wasserstein (Earth Mover's) Distance	$W(P_r, P_f) = \inf_{\gamma \sim \pi} \mathbb{E}_{(x,y)\sim\gamma}[\|\|x-y\|\|] = \frac{1}{K} \sup_{\|\|f\|\|_L \leq K} \mathbb{E}_{P_r}[f] - \mathbb{E}_{P_f}[f]$

Figure 1

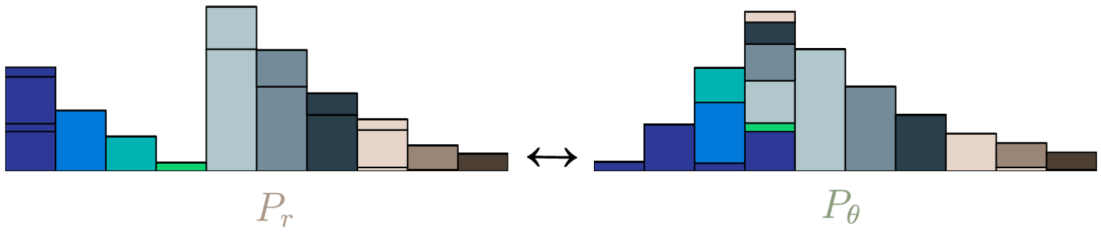

Figure 2

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 3

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of priors.
    $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

VI. References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in Neural Information Processing Systems (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[2] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen, *Improved techniques for training gans*, Advances in Neural Information Processing Systems (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou, *Wasserstein generative adversarial networks*, Proceedings of the 34th International Conference on Machine Learning (Doina Precup and Yee Whye Teh, eds.), Proceedings of Machine Learning Research, vol. 70, PMLR, 06–11 Aug 2017, pp. 214–223.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, *Improved training of wasserstein gans*, Advances in Neural Information Processing Systems (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] Yun Luo and Bao-Liang Lu, *Eeg data augmentation for emotion recognition using a conditional wasserstein gan*, 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018, pp. 2535–2538.

[6] Pritish Chandna, Merlijn Blaauw, Jordi Bonada, and Emilia Gmez, *Wgansing: A multi-voice singing voice synthesizer based on the wasserstein-gan*, 2019 27th European Signal Processing Conference (EUSIPCO), 2019, pp. 1–5.

[7] Jan Stanczuk, Christian Etmann, Lisa Maria Kreusser, and Carola-Bibiane Schnlieb, *Wasserstein gans work because they fail (to approximate the wasserstein distance)*, 2021. arXiv:2103.01678v4 [stat.ML]