Wassertein Generative Adversarial Networks (WGAN)

Machine Learning Project

ENS de Lyon, Spring 2018

louis.bethune@ens-lyon.fr
guillaume.coiffier@ens-lyon.fr

Introduction

The theoretical work of this report is supported by the articles [1], [2] and [3].

In [1], we found the original concept and definition of a generative adversarial network, and some basic results about the training.

In [2] can be found the definition of Wasserstein GAN, along some nice properties of these networks.

Finally, [3] describes an improved method for training Wasserstein GAN.

We gave no rigourous proofs of the theorems we exposed. The proofs can be found in their respective articles.

Our experimental work is given with this report. It is an implemention of the WGAN training algorithm using Python and the Keras library.

Contents

1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) are a class of unsupervised machine learning algorithm. They often consist of two multi-layer perceptrons contesting with each other in a two-player zero-sum game. They were introduced in 2014 by *Ian Goodfellow et al.* in [1].

1.1 Principle of GANs

One neural network is the generator $\mathcal{G}$. Its goal is to learn a probability distribution $\mathbb{P}_r$.

The other network is the discriminator $\mathcal{D}$, deciding if its input comes from $\mathcal{G}$'s output or from training examples following $\mathbb{P}_r$: $\mathcal{D}_(x)$ will be the probability that x comes from the data and not from $\mathcal{G}$. We train $\mathcal{D}$ in order to maximize the probability of correctly labelling both training examples and outputs of $\mathcal{G}$.

Parameters of $\mathcal{G}$ will be denoted as θ. Parameters of $\mathcal{D}$ will be denoted as ω. The probability density output by $\mathcal{G}$ will then be called $\mathbb{P}_\theta$.

When given a set of m training examples $x^{(i)}$ distributed according to $\mathbb{P}_r$, we want $\mathcal{G}$ to follow the distribution $\mathbb{P}_\theta$ that maximizes the likelihood of our data, that is to say :

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log(\mathbb{P}_\theta(x^{(i)}) \tag{1.1}$$

If both distributions admit a probability density, then this amount will assymptotically minimize the KL divergence $KL(\mathbb{P}_\theta || \mathbb{P}_r)$ (defined below in 2.2). However, it is unlikely in general that $\mathbb{P}_\theta$ admits a density, which leads to an undefined or infinite KL divergence. The typical solution to paliate this problem is to define a noise distribution $\mathbb{P}_z$. Rather than estimating $\mathbb{P}_r$ by a probability distribution $\mathbb{P}_\theta$ which does not have a density in general, we define a parametrized function $g_\theta : \mathcal{Z} \to \mathcal{X}$ fed with $z \sim \mathbb{P}_z$. This function g_θ is what the generator network $\mathcal{G}$ will compute, so that $\mathbb{P}_\theta = g_\theta(\mathbb{P}_z)$.

Simultaneously, we want $\mathcal{D}$ to be a "good" discriminator. $\mathcal{D}$ will represent a function f_ω that must minimize the quantity $\log(1 - f_\omega(g_\theta(z))$ for $z \sim \mathbb{P}_z$.

In other words, $\mathcal{G}$ and $\mathcal{D}$ can be seen as two players playing the minmax of value function $V(\mathcal{G}, \mathcal{D})$:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x \sim \mathbb{P}_r}[\log(f_\omega(x))] + \mathbb{E}_{z \sim \mathbb{P}_z}[\log(1 - f_\omega(g_\theta(z)))] \tag{1.2}$$

1.2 GAN's training

Training GAN is however considered as a hard task and is quite unstable. The main problem is that updates of the generator are getting worse and worse, although the result has clearly not converged to its optimal. The explanation lays within the gradient that is used to update $\mathcal{G}$. In practice, since the discriminator is trained faster than the generator, the generator's gradient tends to have very small values very quickly, which makes the update harder and longer.

This phenomenom is well known since the work of [4]. the following theorem illustrates what happens :

Theorem 1. *Let $g_\theta : \mathcal{Z} \to \mathcal{X}$ be a differentiable function that induces a distribution $\mathbb{P}_\theta$. Let $\mathbb{P}_r$ be the real data distribution. Let $\mathcal{D}$ be a differentiable discriminator. If the conditions of Theorems 2.1 or 2.2 are satisfied, $||\mathcal{D} - \mathcal{D}^*|| < \varepsilon$ and $\mathbb{E}_{z \sim \mathbb{P}_z}[||J_\theta g_\theta(z))||_2^2] \leqslant M^2$, then :*

$$||\nabla_\theta \mathbb{E}_{z \sim \mathbb{P}_z}[log(1 - f_w(g_\theta(z)))]||_2 < M \frac{\varepsilon}{1 - \varepsilon}$$

In other words, under some assumptions described in [2], if $\mathcal{D}^*$ is the optimal discriminator, the closer we get from $\mathcal{D}^*$, the smaller the gradient of $\mathcal{G}$, so the smaller the improvement.

2 Wassertein GAN

Wasserstein GAN (WGAN) are an improvement of the original GAN method. It especially makes the training of the networks easier. The key idea of WGAN is to change the way we measure distances between probability distributions. Instead of the KL divergence, we instead use the Earth-mover distance, which is defined below (in 2.4).

2.1 How to measure distances between probability distributions

Selectionning a distance ρ between probability distributions is the heart of the GAN framework. Different distances may indeed induce different behavior for convergence of distribution sequences.

In the WGAN framework, we want to define $\mathbb{P}_\theta$ such that the mapping $\theta \to \mathbb{P}_\theta$ is continuous. The weaker the distance, the easier it will be. This is important, because we want to have a loss function of form $\theta \to \rho(\mathbb{P}_\theta, \mathbb{P}_r)$, which will then be continuous.

Let $\mathcal{X}$ be a **compact** metric set. Let Σ be the set of all Borel subsets of $\mathcal{X}$ and $Prob(\mathcal{X})$ be the set of posible probability measures over $\mathcal{X}$.

Let $\mathbb{P}_1, \mathbb{P}_2 \in Prob(\mathcal{X})$. We define the following distances :

The Total variation (TV) distance

$$\delta(\mathbb{P}_1, \mathbb{P}_2) = \sup_{A \in \Sigma} |\mathbb{P}_1(A) - \mathbb{P}_2(A)| \tag{2.1}$$

The Kullback-Leibler (KL) divergence

$$KL(\mathbb{P}_1 || \mathbb{P}_2) = \int \log\left(\frac{\mathbb{P}_1(x)}{\mathbb{P}_2(x)}\right) \mathbb{P}_1(x) d\mu x \tag{2.2}$$

Where both probabilities are assumed to admit densities with respect to the same measure μ over $\mathcal{X}$.

The Jensen-Shannon (JS) divergence

$$JS(\mathbb{P}_1, \mathbb{P}_2) = KL(\mathbb{P}_1 || \mathbb{P}_m) + KL(\mathbb{P}_2 || \mathbb{P}_m) \tag{2.3}$$

where $\mathbb{P}_m = 1/2(\mathbb{P}_1 + \mathbb{P}_2)$.

The Earth-Mover (EM) distance, or Wassertein-1 distance

$$W(\mathbb{P}_1, \mathbb{P}_2) = \inf_{\gamma \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{2.4}$$

where $\Pi(\mathbb{P}_1, \mathbb{P}_2)$ is the set of all joint distributions γ whose marginal distributions are $\mathbb{P}_1$ and $\mathbb{P}_2$. Informly, the EM distance indicates how much "mass" should be moved in order to change $\mathbb{P}_1$ in $\mathbb{P}_2$ (hence the name). It is the "cost" of the optimal transport.

2.2 The Earth-Mover distance and its properties

WGAN uses a distance which is an approximation of the EM distance (see 2.4). It is interresting to see that some apparently simple sequences of probability distributions converge under the EM distance, but not using the other distances defined above. Examples are given in [2].

Moreover, udner reasonable assumptions, the loss function $\theta \to W(\mathbb{P}_\theta, \mathbb{P}_r)$ is continuous and differentiable almost everywhere.

Definition 1. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a locally Lipschitz function, denoted by $g_\theta(z)$ with z the first coordinate and θ the second. We say that g satisfies assumption (*) for a certain distribution $\mathbb{P}$ over $\mathcal{Z}$ if there are local Lipschitz constant $L(\theta, z)$ such that :

$$\mathbb{E}_{z\sim\mathbb{P}}[L(\theta, z)] < \infty$$

Theorem 2. *Continuity of the Wassertein distance (from [2]) Let $\mathbb{P}_r$ be a fixed probability distribution over $\mathcal{X}$. Let Z be a random variable over space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ a function. Let $\mathbb{P}_\theta = g_\theta(Z)$. Then :*

1. *If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.*
2. *If g is locally Lipschitz and satisfies assumption (*), then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere.*
3. *Statements 1 and 2 are false for the JS divergence and the KL divergence.*

This theorem also implies the following corollary :

Corollary 1. *Let g_θ be any feedforward neural network with parameters θ, and let $\mathbb{P}_z$ be a prior over $\mathcal{Z}$ such that $\mathbb{E}_{z\sim\mathbb{P}_z}[||z||] < \infty$. Then g_θ satisfies assumption (*) and therefore $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere and differentiable almost everywhere.*

In other words, the EM distance has good properties for our framework, which other distances don't have. The EM distance can be used as a sensible loss function in the context of GAN training.

2.3 Training of WGAN

We saw that WGAN use the EM distance as a loss function. However, even though this distance has good mathematical properties, its minimum is still hard to find. Thanks to the optimal transport theory and the Kantorovitch-Rubinstein duality (see [5] for further information), we have :

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{f:||f||_L \leqslant 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{2.5}$$

where the supremum is over all the 1-Lipschitz functions from $\mathcal{X}$ to $\mathbb{R}$. Rewriting $x \sim \mathbb{P}_\theta$ as the output of the generator network $\mathcal{G}$ and parametrizing the function f as f_ω, the training consists in solving the following problem :

$$\min_{\mathcal{G}:\theta} \max_{\mathcal{D}:\omega} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[f_w(g_\theta(z))] \tag{2.6}$$

This problem looks very similar to the initial GAN problem of equation 1.1. However, the function f_w has to be found among the 1-Lipschitz functions. Morever, the logarithms, which were intrinsic to the KL divergence, have dissapeared.

It turns out that this problem admits a solution :

Theorem 3. *Let $\mathbb{P}_r$ be any distribution, and $\mathbb{P}_\theta$ be the distribution of $g_\theta(Z)$ with Z some noise variable. Assume that g_θ satisfies assumption (*). Then :*

- *There is a solution f to the problem 2.5 (ie, the supremum is actually a maximum)*
- $\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z\sim\mathbb{P}_z}[\nabla f(g_\theta(z))]$

From there comes the idea of a discriminator network $\mathcal{D}$ learning the parametrized function f_w approximating f, while the generator networks learns function g_θ. We assume that the ω are in a compact space $\mathcal{W}$. To optimize f_ω, we do a back propagation over $\mathbb{E}_{z\sim\mathbb{P}_z}[\nabla f(g_\theta(z))]$, as in a classical GAN. However, this does not ensure that the weights stay in a compact space. To cope with this, we clip the parameters ω so they stay in a fixed box of $\mathbb{R}^d$ (for example, $\mathcal{W} = [-0.1, 0.1]^d$). The training algorithm is then the following :

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 prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  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

The training algorithm proposed in [2]

Notice that the discriminator $\mathcal{D}$ (here denoted by *critic*) makes n_{critic} loops of training while the generator $\mathcal{G}$ makes only one. This is made so that the discriminator stays at its optimal solution while the generator slowly improves itself.

The update algorithm used in RMSProp (Root Mean Square Propagation). It is an improved version of the stochastic gradient descent algorithm. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. For further information, see [6].

In practice, one has a different stopping condition than "θ has converged". In our own experiment (see chapter 4), we trained the network on a fixed number of epochs. In each epochs were given a fixed number of batchs of examples to the WGAN.

Note that the EM distance does not suffer from the vanishing gradient problem (as defined above in 1.2). The theorem, proved in [3], is the following :

Theorem 4. *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions over a compact metric space $\mathcal{X}$. Let f^* be the optimal solution of equation 2.5 (existence of f^* is ensured by theorem 3). Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$, that is to say*

$$\pi = \arg\min W(\mathbb{P}_r, \mathbb{P}_g) \;=\; \underset{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)}{\arg\min}\; \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

Then, if f^* is differentiable and $\pi(x = y) = 0$, let $x_t = tx + (1-t)y$ with $0 \leqslant t \leqslant 1$. It holds that :

$$\mathbb{P}_{(x,y)\sim\pi}\left(\nabla f^*(x_t) = \frac{y - x_t}{||y - x_t||}\right) = 1$$

Corollary 2. *$f*$ has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$*

This ensures that the gradient of the discriminator function does not tend to zero as we get closer from the optimal solution (by a continuity argument).

Thanks to this result, we know that WGAN can be trained more easily than classical GANs. Moreover, the model is less sensible to the network's architecture, and it is not required anymore to maintain a very precise balance between $\mathcal{D}$ and $\mathcal{G}$'s results.

3 Improved Wassertein GAN

Although the clipping idea and the whole WGAN framework improved the training of generative adversarial networks, it still causes some issues. In [3] are described some problems caused by the weight clipping operation of the discriminator $\mathcal{D}$. Another solution, called gradient penalty, is also proposed.

3.1 Instability due to clipping

Clipping can lead to optimization issues too. In the following pictures is compared the norm of the gradients for several values of the hyper parameter c, which is the clipping threshold (values are kepts in $[-c, c]^d$).

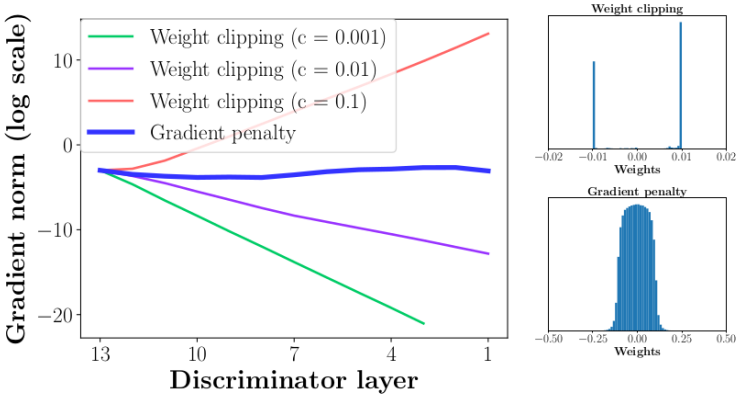

FIGURE 1 : *Some comparative results between clipping and gradient penalty (from [3])*

We can see that depending of the value of c, the gradient norm can go to zero, or diverge. Since the clipping forces the value to stay in a certain interval, we have a saturation of the norm of the gradient around c. As we can see on the top-right figure, the gradient norm tends to have only two values, 0 and c.

Such problems are not encountered with the gradient penalty algorithm. Gradient norm have a nice distribution, and we do not suffer from saturation.

Figure 2 show results of training WGAN on toy examples, using (top) weight clipping and (bottom) gradient penalty. Critics trained with weight clipping fail to capture higher moments of the data distribution. In fact, clipping the weights to ensure that the function is Lipschitz tends bias the discriminator to simpler functions. We only get simplified approximation of the optimal solution.

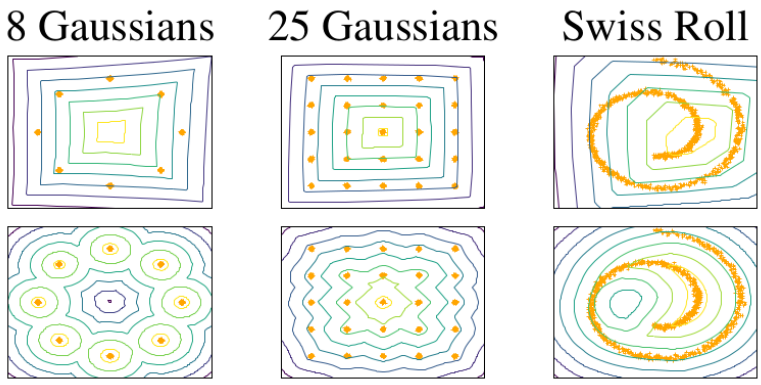

FIGURE 2 : *Comparison of results of clipping and gradient penalty algorithms on the quality of outcome (from [3])*

3.2 The Gradient Penalty WGAN

Gradient penalty is an alternative way of enforcing the Lipschitz constraint. We get rid of the hyperparameter c, which does not have to be carefully tuned anymore. A differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere, so we consider directly constraining the gradient norm of the critic's output with respect to its input. For a random sample $z \sim \mathbb{P}_z$, the new objective function is :

$$\mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[f_w(g_\theta(z))] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} f_w(\hat{x})||_2 - 1)^2] \tag{3.1}$$

where the term $\lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} f_w(\hat{x})||_2 - 1)^2]$ is the gradient penalty, which is added to the original objective function (2.6)

The coefficient λ is a new hyperparameter, called the *penalty coefficient*.

The law $\mathbb{P}_{\hat{x}}$ is a convex combination of the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_\theta$. This is motivated by the fact that the optimal critic f^* contains straight lines with gradient norm 1 connecting coupled points from $\mathbb{P}_r$ and $\mathbb{P}_\theta$ (see 4)

This new objective function leads to the following algorithm :

Algorithm 2 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.
while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon\boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

The improved training algorithm proposed in [3]

Notice that this algorithm has a very similar structure than the algorithm given in 2.3. The two main differences are the sampling of a mixed data from $\varepsilon\mathbb{P}_r + (1 - \varepsilon)\mathbb{P}_\theta$ with a uniformly random ε, and the regression algorithm.

Unlike algorithm 2.3 which uses RMSProp, this algorithm uses the Adam (Adaptive Moment Estimation) algorithm to backpropagate.

4 Experimental results

4.1 Global presentation

Due to our lack of computation power, we were not able to train a WGAN to generate large pictures. Instead, we tried to learn gaussian laws in 1D. More precisely :

- $\mathbb{P}_r = \mathcal{N}(2, 1)$
- $\mathbb{P}_z = \mathcal{U}([0, 5])$

The program was written in python, using Keras. Both our generator $\mathcal{G}$ and discriminator $\mathcal{D}$ have two hidden layers of 128 neurons.

We did not implement the GP-WGAN algorithm, as it required more time to setup.

4.2 Results and encountered problems

Conclusion

WGAN are an improvement of the GAN framework. The modified objective function used in the WGAN is based on the Earth-Mover distance, and not on the Jensen-Shannon divergence. This allow more stability in the training process. However, this distance require the discriminating function to be k-Lipschitz. The method used to enforce this condition is weight clipping, and it has been show in [3] that this method lead to simplified functions and results that are not satisfying in practice. The same article proposed the WGAN-GP algorithm, which use the gradient penalty method instead of weight clipping. The WGAN-GP is currently the state of the art in the GAN framework.

Bibliography

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[5] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[6] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[7] An implementation of a wgan using keras. https://github.com/tdeboissiere/DeepLearningImplementations/tree/master/WassersteinGAN. Accessed: 2018-04-18.

[8] An implementation of a gp-wgan using keras. https://github.com/keras-team/keras-contrib/blob/master/examples/improved_wgan.py. Accessed: 2018-04-30.