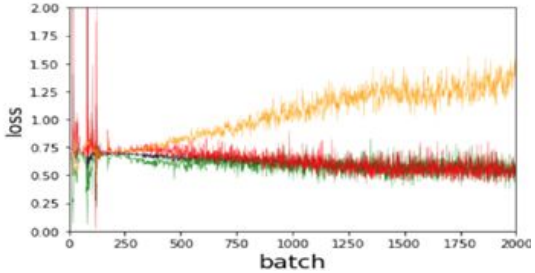
GAN Training Pathologies

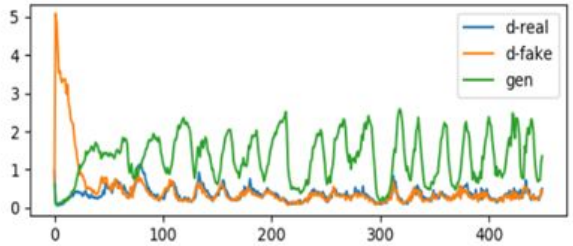
Summary

Diminished gradient: the discriminator gets too successful that the generator gradient vanishes and learns nothing.



Mode collapse: the generator collapses which produces limited varieties of samples.

Non-convergence: the model parameters oscillate, destabilize, and never converge.



Summary

Most of this lecture is motivated by:

- **From GAN to WGAN**
  - https://arxiv.org/abs/1904.08994
- **Improved Techniques for Training GANs**
  - https://arxiv.org/abs/1606.03498

Generative models training

Many generative models create a model $\boldsymbol{\theta}$ that maximizes the Maximum Likelihood Estimation **MLE**. i.e. finding the best model parameters that fit the training data the most.

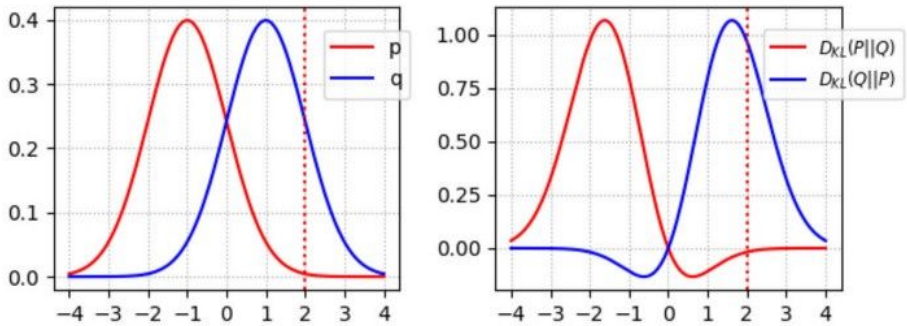
$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i|\theta)$$

This is the same as minimizing the KL-divergence ***KL(p,q)*** which measures how the probability distribution $\boldsymbol{q}$ (estimated distribution) diverges from the expected probability distribution $\boldsymbol{p}$ (the real-life distribution).

$$D_{KL}(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

Kullback-Leibler divergence. Main issues

KL(x) drops to 0 for area where ***p(x)*** → ***0***. For example, in the figure on the right below, the red curve corresponds to *D(p, q)*. It drops to zero when ***x>2*** where ***p*** approaches 0.



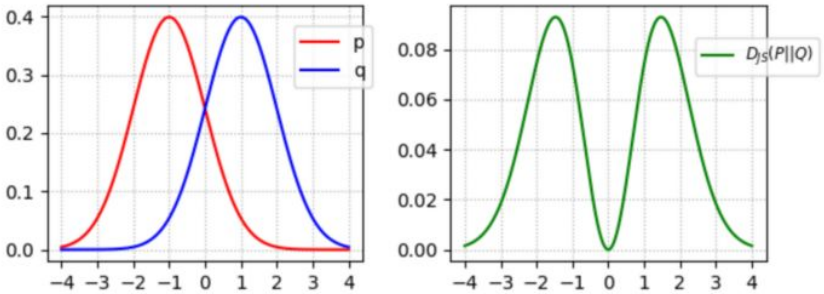
The KL-divergence ***DL(p, q)*** penalizes the generator if it misses some modes of images: the penalty is high where *p(x) > 0* but *q(x)* → *0*. Nevertheless, it is acceptable that some images do not look real. The penalty is low when *p(x)* → *0* but *q(x)>0*. (Poorer quality but more diverse samples)

The reverse KL-divergence ***DL(q, p)*** penalizes the generator if the images do not look real: high penalty if *p(x)* → *0* but *q(x) > 0*. But it explores less variety: low penalty if *q(x)* → *0* but *p(x) > 0*. (Better quality but less diverse samples)

Jensen-Shannon divergence

Training GANs has treated as optimizing the generator model is treated as minimizing the JS-divergence.

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$$



JS-divergence is symmetrical. It will penalize poor images badly. (when $p(x) \rightarrow 0$ and $q(x) > 0$)

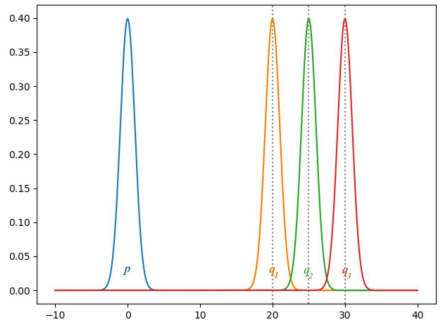
If the discriminator is optimal (performing well in distinguishing images), the generator's objective function becomes

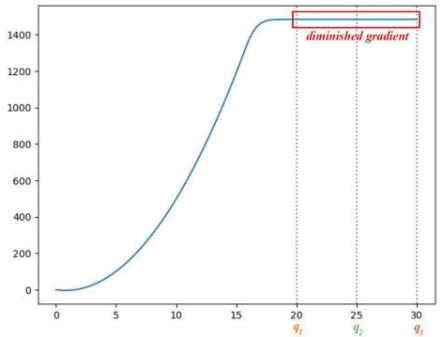
$$\min_G V(D^*, G) = 2D_{JS}(p_r \| p_g) - 2\log 2$$

Vanishing gradient

What happens to the JS-divergence gradient when the data distribution ***q*** of the generator's images does not match with the ground truth ***p*** for the real images?

Let's consider an example in which ***p*** and ***q*** are Gaussian distributed and the mean of ***p*** is zero. Let's consider ***q*** with different means to study the gradient of *JS(p, q)*.





JS-divergence *JS(p, q)* between *p* and *q* with means of *q* ranging from 0 to 30.

The gradient for the JS-divergence vanishes from *q1* to *q3*. The GAN generator will learn extremely slow to nothing when the cost is saturated in those regions. In particular, in early training, *p* and *q* are very different and the generator learns very slow.

Vanishing gradient

Attempts to remedy:

- **Wasserstein loss:** The Wasserstein loss is designed to prevent vanishing gradients even when you train the discriminator to optimality.

- **Modified minimax loss:** The original GAN paper proposed a modification to minimax loss to deal with vanishing gradients.

Vanishing gradient

Name	Discriminator Loss	Generator Loss
Minimax GAN	$\mathcal{L}_D^{GAN} = -\mathbb{E}_{\mathbf{x}} \log D(\mathbf{x}) - \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$	$\mathcal{L}_G^{GAN} = \mathbb{E}_{\mathbf{z}} \log(1 - D(G(\mathbf{z})))$
Non-Saturating GAN	$\mathcal{L}_D^{NSGAN} = \mathcal{L}_D^{GAN}$	$\mathcal{L}_G^{NSGAN} = -\mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$
Least-Squares GAN	$\mathcal{L}_D^{LSGAN} = \mathbb{E}_{\mathbf{x}} (D(\mathbf{x}) - 1)^2 + \mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))^2$	$\mathcal{L}_G^{LSGAN} = \mathbb{E}_{\mathbf{z}} (D(G(\mathbf{z})) - 1)^2$
Wasserstein GAN	$\mathcal{L}_D^{WGAN} = -\mathbb{E}_{\mathbf{x}} D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))$	$\mathcal{L}_G^{WGAN} = -\mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))$
WGAN-GP	$\mathcal{L}_D^{WGANGP} = \mathcal{L}_D^{WGAN} + \lambda \mathbb{E}_{\mathbf{x},\mathbf{z}} (\lVert \nabla D(\alpha \mathbf{x} + (1-\alpha) G(\mathbf{z})) \rVert_2 - 1)^2$	$\mathcal{L}_G^{WGANGP} = \mathcal{L}_G^{WGAN}$
DRAGAN	$\mathcal{L}_D^{DRAGAN} = \mathcal{L}_D^{GAN} + \lambda \mathbb{E}_{\mathbf{x} \sim p_{data} + \mathcal{N}(0,c)} (\lVert \nabla D(\mathbf{x}) \rVert_2 - 1)^2$	$\mathcal{L}_G^{DRAGAN} = \mathcal{L}_G^{GAN}$
BEGAN	$\mathcal{L}_D^{BEGAN} = \mathbb{E}_{\mathbf{x}} \lVert \mathbf{x} - AE(\mathbf{x}) \rVert_1 - k_t \mathbb{E}_{\mathbf{z}} \lVert G(\mathbf{z}) - AE(G(\mathbf{z})) \rVert_1$	$\mathcal{L}_G^{BEGAN} = \mathbb{E}_{\mathbf{z}} \lVert G(\mathbf{z}) - AE(G(\mathbf{z})) \rVert_1$

Mode collapse

The objective of the GAN generator is to create images that can fool the discriminator ***D*** the most.

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

Extreme case: ***G*** is trained extensively without updates to ***D***. The generated images will converge to find the optimal image $\boldsymbol{x}^*$ that deceives ***D*** the most, the most realistic image from the discriminator perspective. In this extreme, $\boldsymbol{x}^*$ will be independent of $\boldsymbol{z}$.

$$x^* = argmax_x D(x)$$

Mode collapse

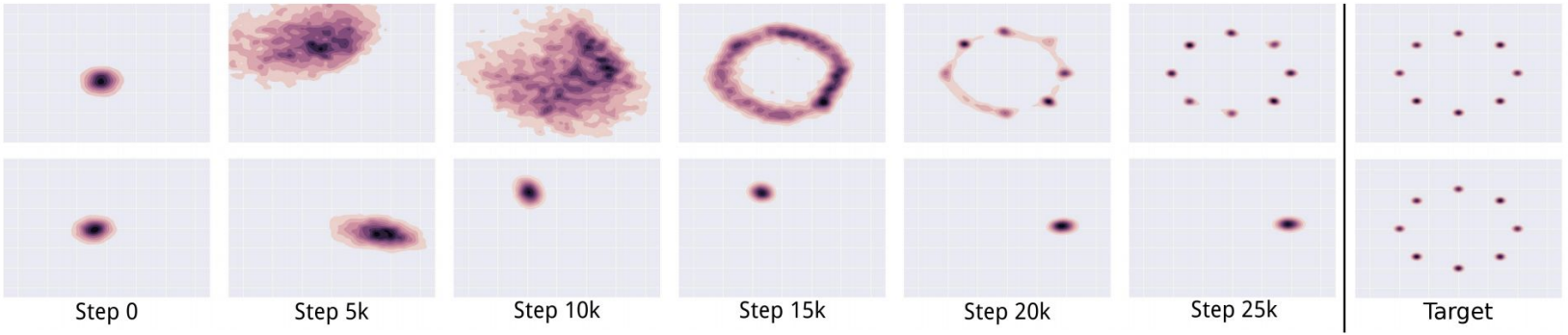
This is bad news. The mode collapses to a **single point**. The gradient associated with $\boldsymbol{z}$ approaches zero.

$$\frac{\partial J}{\partial z} \approx 0$$

Restart the training in the discriminator: the most effective way to detect generated images is to detect this single mode. Since the generator has minimized the impact of $\boldsymbol{z}$ already in the generated samples, the gradient from the discriminator will likely push the single point around for the next most vulnerable mode. This is not hard to find. The generator produces such an imbalance of modes in training that it deteriorates its capability to detect others.

Now, both networks are overfitted to exploit short-term opponent weakness.

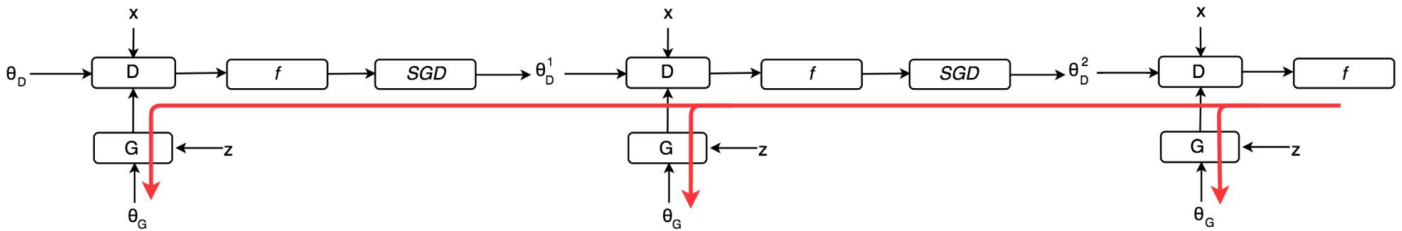
Mode collapse



Mode collapse

Attempts to remedy:

- **Wasserstein loss:** The Wasserstein loss alleviates mode collapse by letting you train the discriminator to optimality without worrying about vanishing gradients. If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on. So the generator has to try something new.
- **Unrolled GANs:** Unrolled GANs use a generator loss function that incorporates not only the current discriminator's classifications, but also the outputs of future discriminator versions. So the generator can't over-optimize for a single discriminator.



Oscillation

GAN is based on the zero-sum non-cooperative game (if one wins the other loses) also called minimax. Your opponent wants to maximize its actions and your actions are to minimize them.

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

In game theory, the GAN model converges when the discriminator and the generator reach a Nash equilibrium. Nash equilibrium happens when one player will not change its action regardless of what the opponent may do.

Consider two player A and B which control the value of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. Player A wants to maximize the value $\boldsymbol{xy}$ while $\boldsymbol{B}$ wants to minimize it.

$$\min_{\mathbf{B}} \max_{\mathbf{A}} V(D, G) = \boldsymbol{xy}$$

The Nash equilibrium is $x=y=0$.

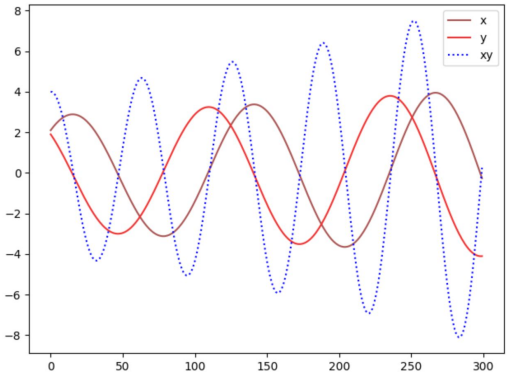
Oscillation

Let's see whether we can find the Nash equilibrium easily using the gradient descent. We update the parameter $\boldsymbol{x}$ and $\boldsymbol{y}$ based on the gradient of the value function V.

$$\Delta x = \alpha \frac{\partial(xy)}{\partial(x)}$$
$$\Delta y = - \alpha \frac{\partial(xy)}{\partial(y)}$$

where $\boldsymbol{\alpha}$ is the learning rate

When we plot $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{xy}$ against the training iterations, we realize our solution does not converge.



It is an excellent showcase that some cost functions will not converge with gradient descent, in particular for a non-convex game.

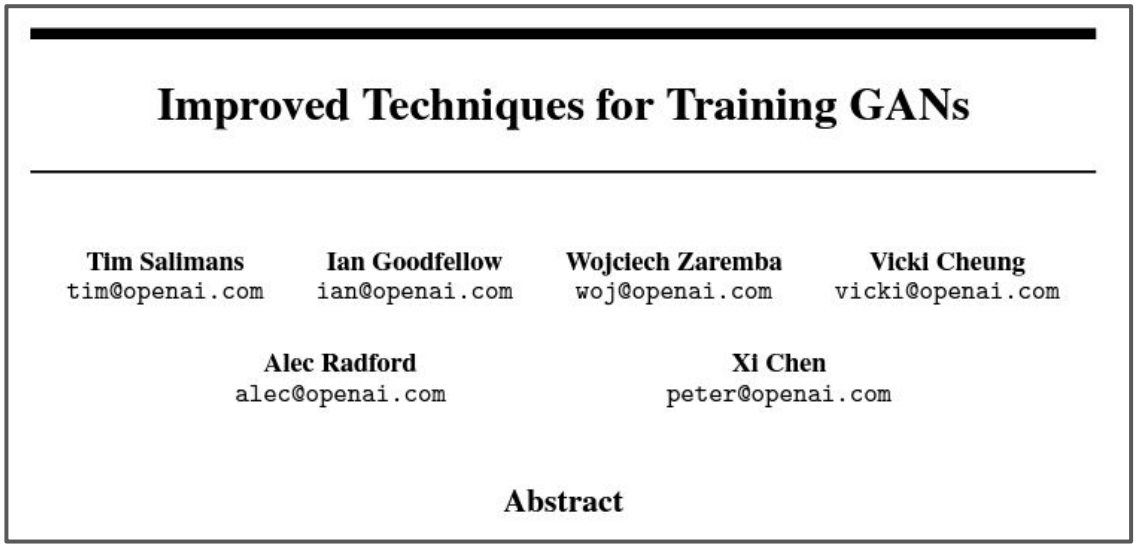
Oscillation

Attempts to remedy:

Researchers have tried to use various forms of regularization to improve GAN convergence, including:

- **Adding noise to discriminator inputs**
- **Penalizing discriminator weights**

Improving GAN training



Improved Techniques for Training GANs

Tim Salimans
tim@openai.com

Ian Goodfellow
ian@openai.com

Wojciech Zaremba
woj@openai.com

Vicki Cheung
vicki@openai.com

Alec Radford
alec@openai.com

Xi Chen
peter@openai.com

Abstract

https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b

https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html