CENG 796
Deep Generative Models

Generative Adversarial Networks - Part II
(Important developments in GANs)

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Recap

- Last lecture:
  - GAN formulation, fundamentals
  - Bayes Optimal D, G
  - Vanishing gradient problems during GAN training
- This lecture:
  - Important developments in GANs in terms of training formulation, architectures, regularization and scaling up.

Deep Convolutional GAN (DCGAN)

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

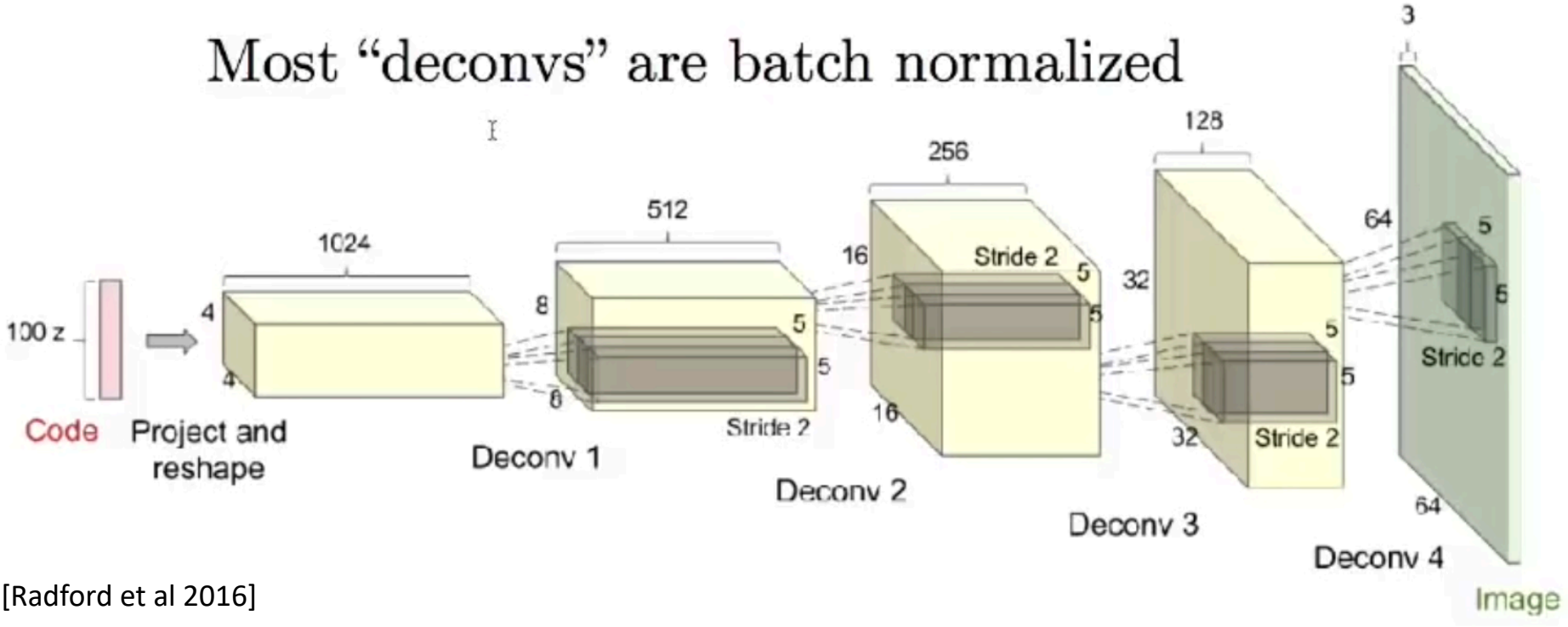
Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

ABSTRACT

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Deep Convolutional GAN (DCGAN)



[Radford et al 2016]

On transposed convolution: https://distill.pub/2016/deconv-checkerboard/

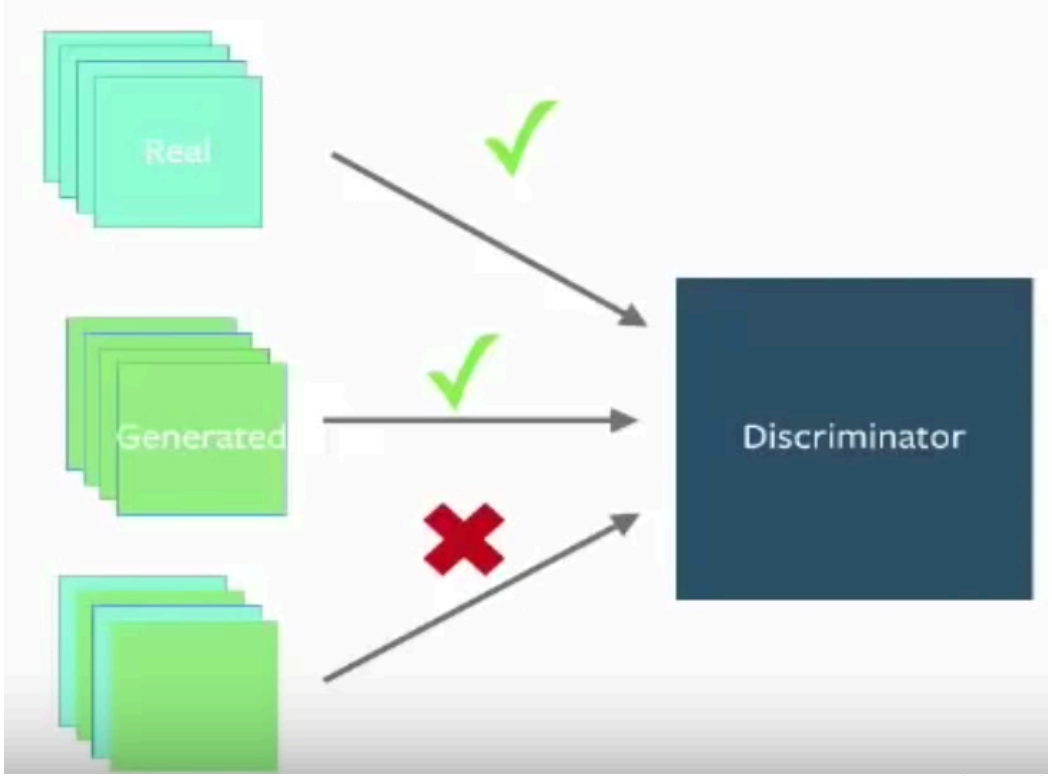
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN's prescription

Architecture guidelines for stable Deep Convolutional GANs

- 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.

DCGAN Batch Norm



Chintala 2016

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

- Good samples on datasets with 3M images (Faces, Bedrooms) for the first time

[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

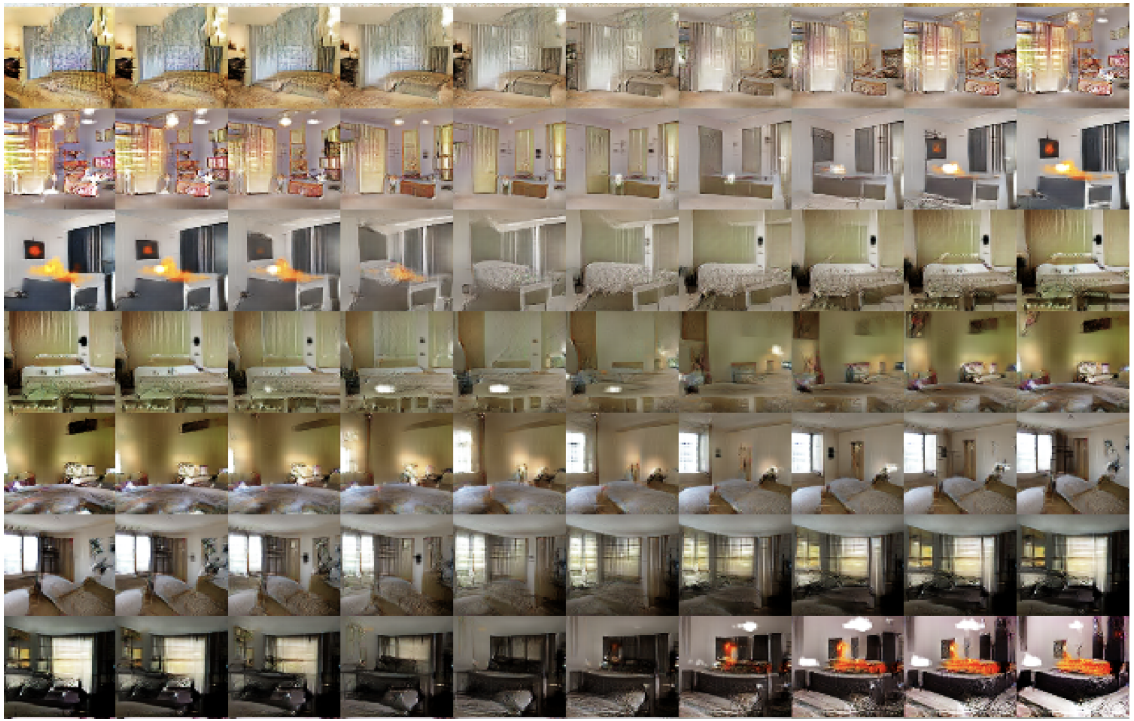
DCGAN - Key Results

[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

- Smooth interpolations in high dimensions

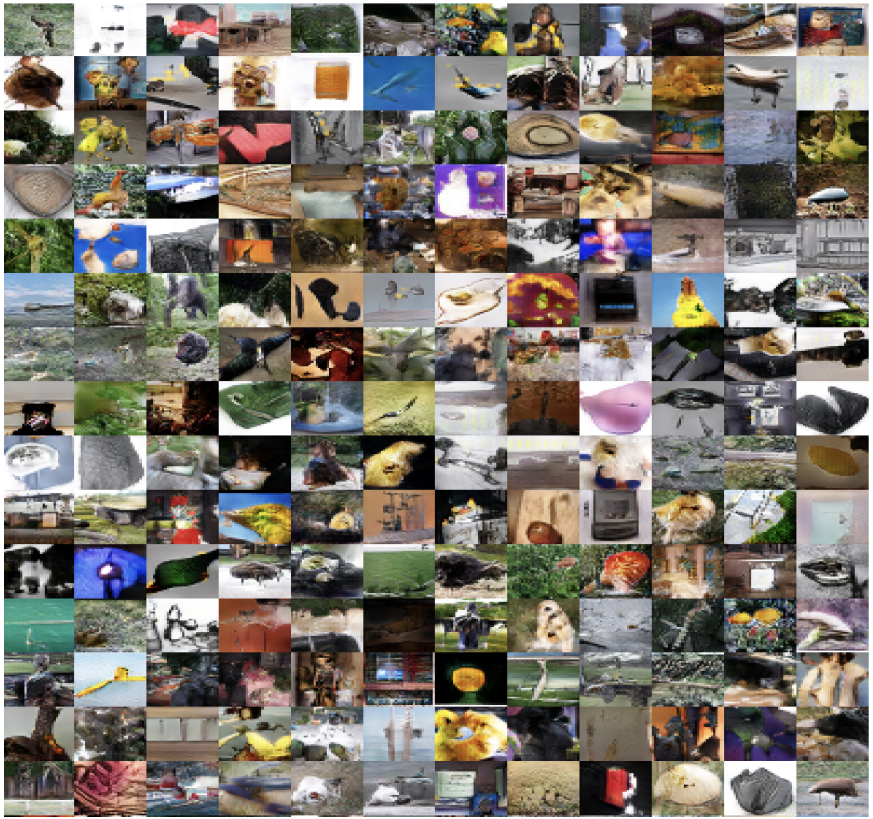


[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

- Imagenet samples

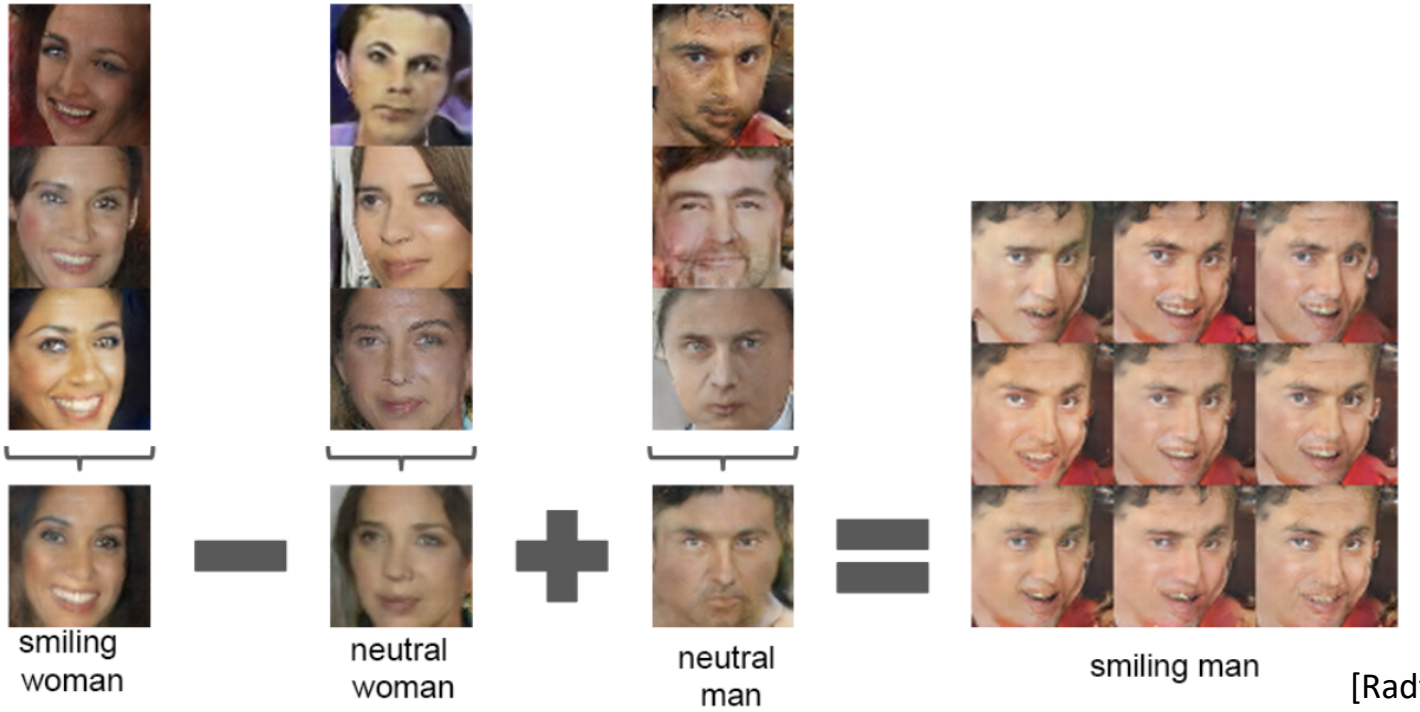


[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

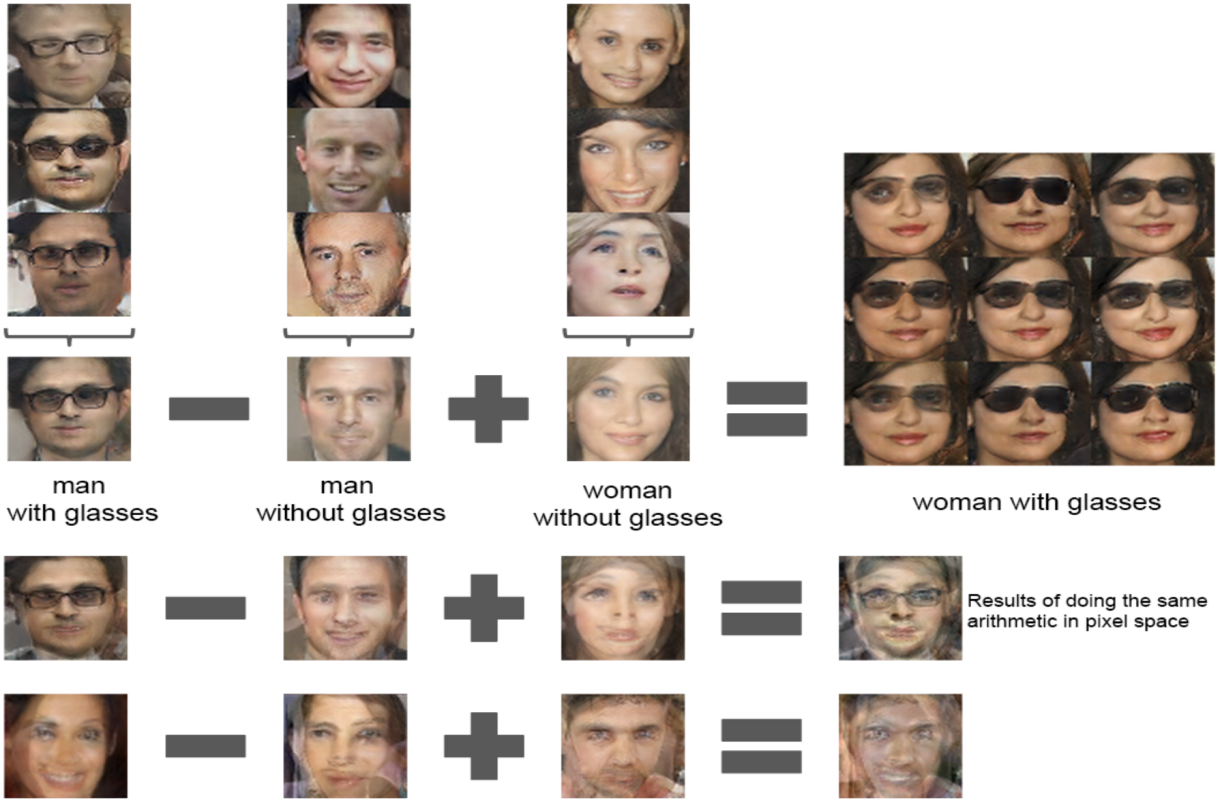
- Vector Arithmetic



[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results



[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

[Radford et al 2016]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

DCGAN - Key Results

Representation Learning results on CIFAR-10

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

[Radford et al 2016]

DCGAN - Conclusions

- Incredible samples for any generative model
- GANs could be made to work well with architecture details
- Perceptually good samples and interpolations
- Representation Learning
- **Problems to address:**
  - Unstable training
  - Brittle architecture / hyperparameters

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Improved training of GANs

- Feature Matching
- Minibatch discrimination
- Historical Averaging
- Virtual batch normalization
- One-sided label smoothing

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.radford@gmail.com
Xi Chen peter@openai.com

Salimans NeurIPS 2016

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Improved training of GANs

- Feature Matching

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

Generator objective

Salimans 2016

Improved training of GANs

- Minibatch discrimination

$$\mathbf{f}(\boldsymbol{x}_i) \in \mathbb{R}^A \qquad T \in \mathbb{R}^{A \times B \times C} \qquad M_i \in \mathbb{R}^{B \times C}$$

$$c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||M_{i,b} - M_{j,b}||_{L_1}) \in \mathbb{R}$$

$$o(\boldsymbol{x}_i)_b = \sum_{j=1}^{n} c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$$

$$o(\boldsymbol{x}_i) = \left[o(\boldsymbol{x}_i)_1, o(\boldsymbol{x}_i)_2, \ldots, o(\boldsymbol{x}_i)_B\right] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n \times B}$$

Salimans 2016

Helps addressing mode collapse by allowing discriminator to detect if the generated samples are too close to each other.

Improved training of GANs

- Historical Averaging

$$||\boldsymbol{\theta} - \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\theta}[i]||^2$$

$\boldsymbol{\theta}[i]$ is the value of the parameters at past time i

Salimans 2016

Improved training of GANs

- One-sided label smoothing

Default discriminator cost:

```
cross_entropy(1., discriminator(data))
+ cross_entropy(0., discriminator(samples))
```


One-sided label smoothed cost (Salimans et al 2016):

```
cross_entropy(.9, discriminator(data))
+ cross_entropy(0., discriminator(samples))
```

Figure source: NeurIPS tutorial Goodfellow 2016

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Improved training of GANs

- Virtual Batch Normalization

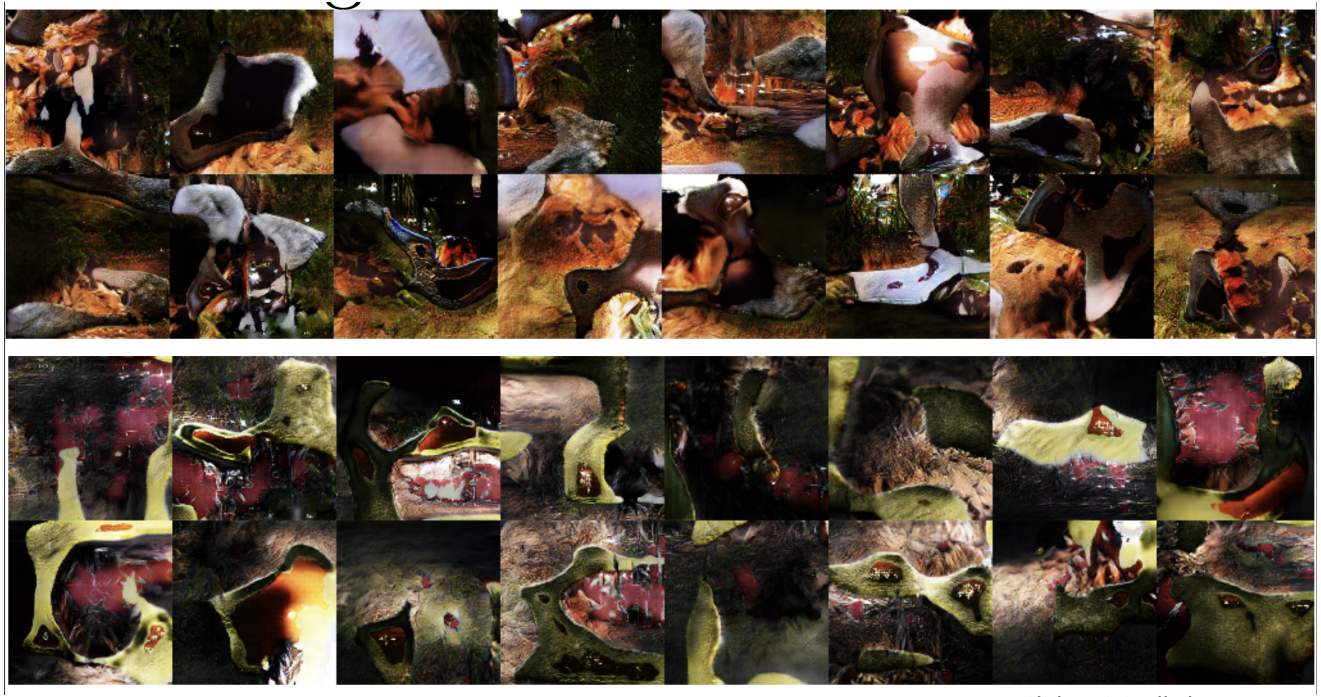


Figure source:
NeurIPS tutorial
Goodfellow 2016

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Improved training of GANs

- Virtual Batch Normalization
  - Use a reference batch (fixed) to compute normalization statistics
  - Construct a batch containing the sample and reference batch

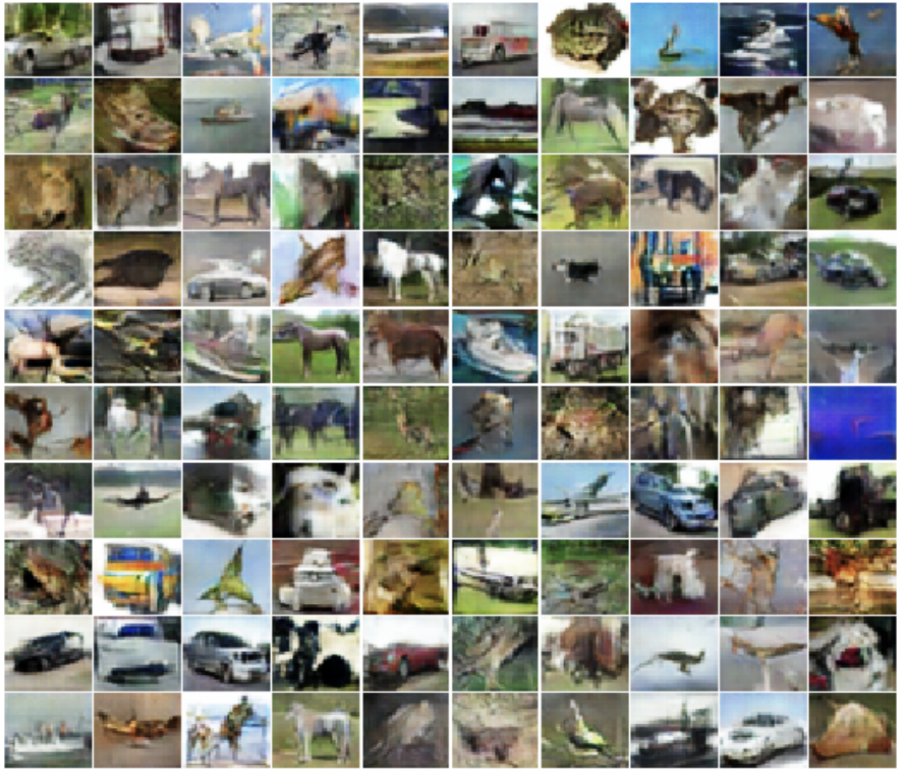
Improved training of GANs

- Semi-Supervised Learning
  - Predict labels in addition to fake/real in the discriminator
  - Approximate way of modeling p(x,y)
  - Generator doesn't have to be made conditional p(x|y)
  - Use a deeper architecture for the discriminator compared to generator

$$
\begin{aligned}
L &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)}[\log p_{\text{model}}(y|\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim G}[\log p_{\text{model}}(y=K+1|\boldsymbol{x})] \\
&= L_{\text{supervised}} + L_{\text{unsupervised}}, \text{ where} \\
L_{\text{supervised}} &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)} \log p_{\text{model}}(y|\boldsymbol{x}, y<K+1) \\
L_{\text{unsupervised}} &= -\{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \log[1-p_{\text{model}}(y=K+1|\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim G} \log[p_{\text{model}}(y=K+1|\boldsymbol{x})]\}
\end{aligned}
$$

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Improved training of GANs



Salimans 2016

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Wasserstein GAN

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Wasserstein Distance

- We have seen $KL(P_r \| P_g)$ and $JSD(P_r, P_g) = KL(P_r \| (\frac{P_r + P_g}{2}) + KL(P_g \| (\frac{P_r + P_g}{2}))$
- Another distance measure inspired from Optimal Transport is the Earth Mover (EM) distance
-
- $W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} [||x - y||]$ erator minimizes the Earth Mover / Wasserstein distance between data and generated distributions.

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Kantorovich Rubinstein Duality

- $$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y|| \right]$$
- Intractable to estimate

- $$\text{Kantorovich Rubinstein Duality: } W(P_r, P_g) = \sup_{\|f_L\| \leq 1} \mathbb{E}_{x\sim P_r} \left[f(x) \right] - \mathbb{E}_{x\sim P_g} \left[f(x) \right]$$
- Search over joint distributions is now a search over 1-Lipschitz functions

$f : X \to Y$ is K-Lipschitz if for distance functions d_X and d_Y on X and Y $d_Y(f(x_1), f(x_2)) \leq K d_x(x_1, x_2)$

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Wasserstein GAN - Pseudocode

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.

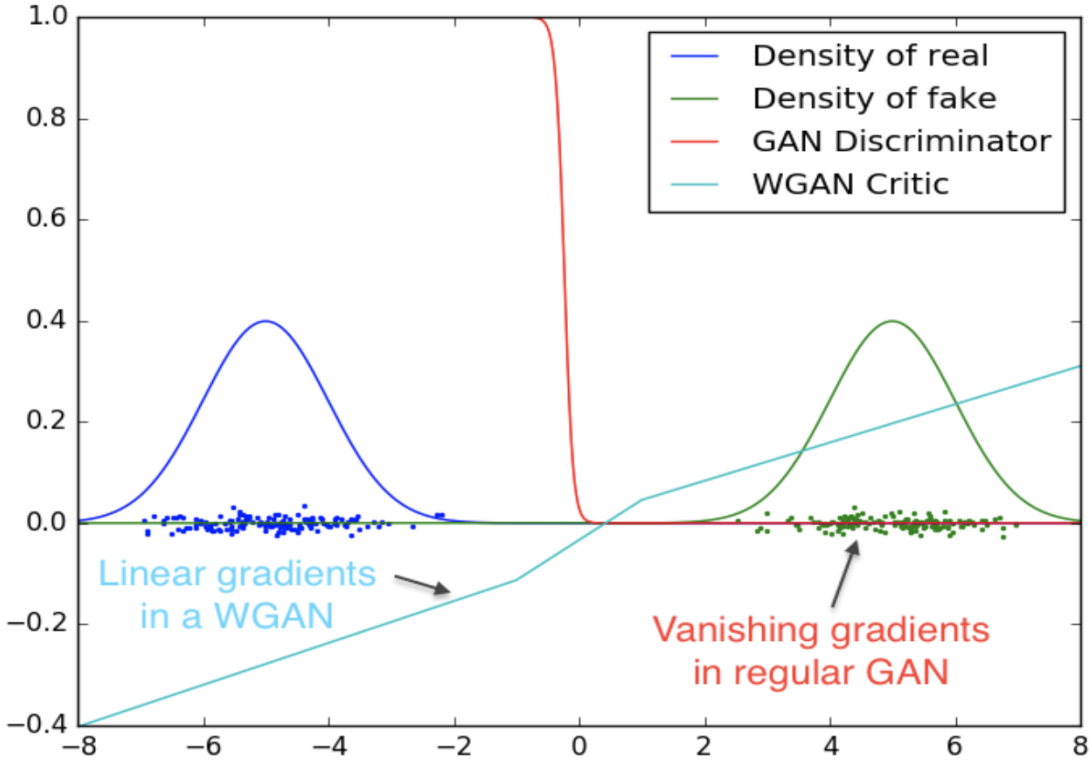
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

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

(Arjovsky et al 2017)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Wasserstein GAN - Training critic to converge

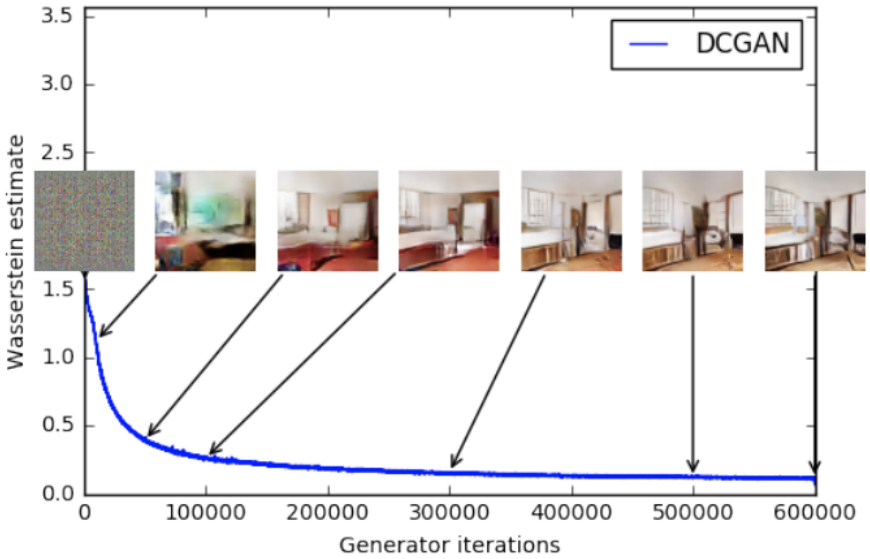


(Arjovsky et al 2017)

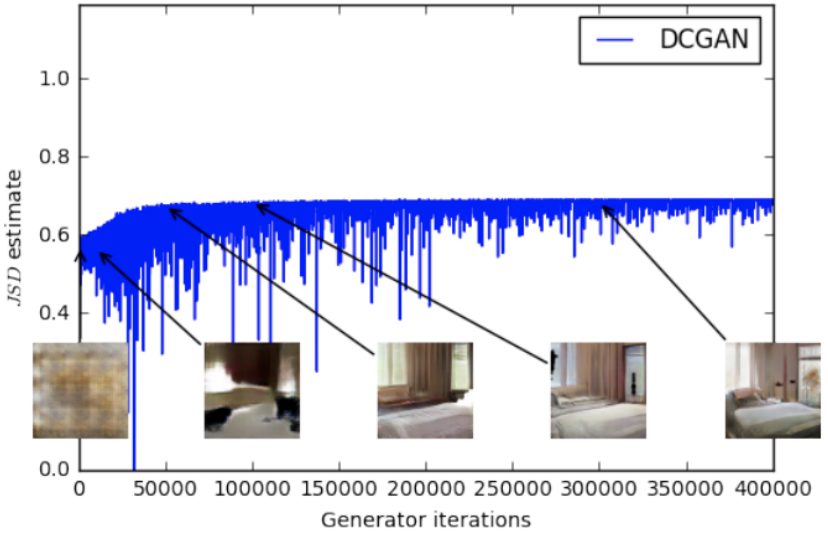
Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Wasserstein distance correlates with sample quality

Wasserstein Estimate

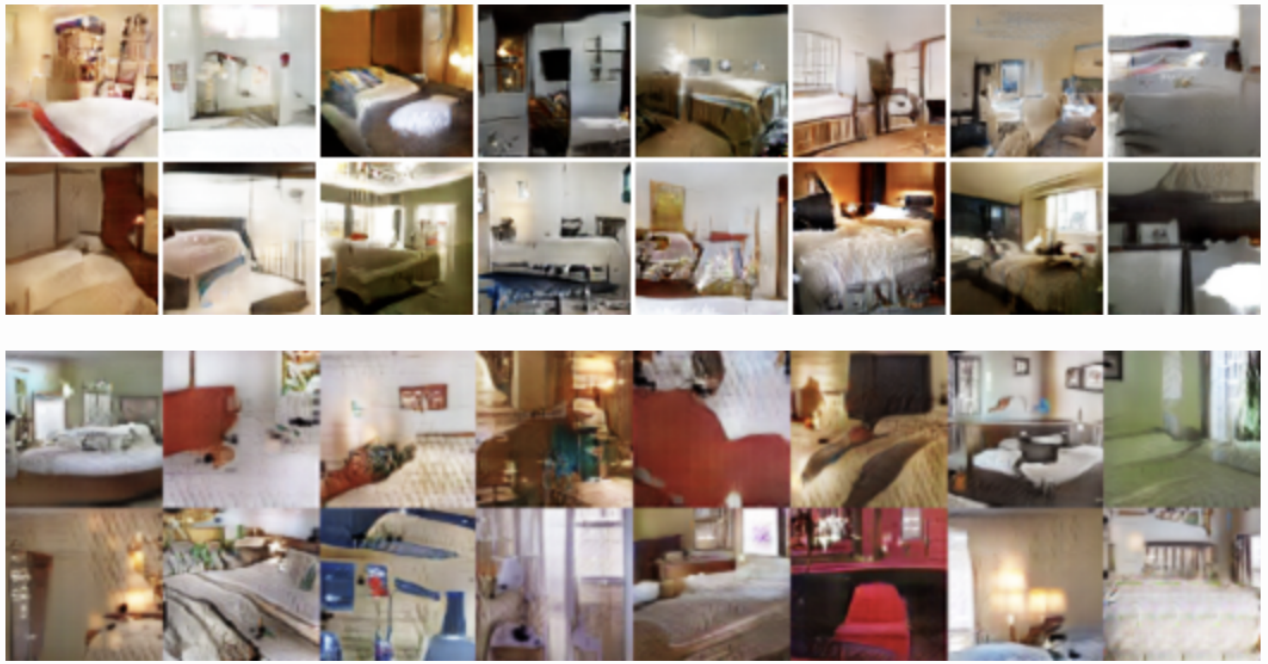


JSD Estimate



(Arjovsky et al 2017)

WGAN Samples on par with DCGAN

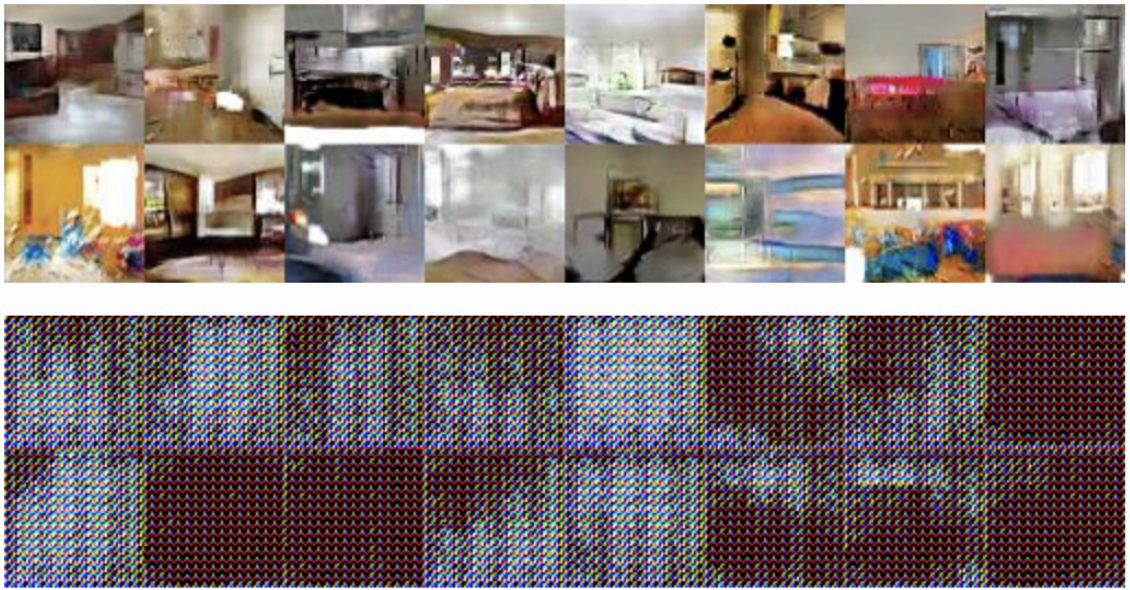


Top: WGAN with the same DCGAN architecture. *Bottom:* DCGAN

(Arjovsky et al 2017)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN robust to architecture choices



Top: WGAN with DCGAN architecture, no batch norm. *Bottom:* DCGAN, no batch norm.

(Arjovsky et al 2017)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN robust to architecture choices

Top: WGAN with MLP architecture. *Bottom:* Standard GAN, same architecture. (Arjovsky et al 2017)

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN Summary

Standard GAN

$$\min_G \max_D \mathbb{E}_{x \sim P_r} [\log D(x)] + \mathbb{E}_{\tilde{x} \sim P_g} [\log(1 - D(\tilde{x}))]$$

Wasserstein GAN

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim P_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})]$$

(Arjovsky et al 2017)

WGAN Summary

- New divergence measure for optimizing the generator
- Addresses instabilities with JSD version (sigmoid cross entropy)
- Robust to architectural choices
- Progress on mode collapse and stability of derivative wrt input
- Introduces the idea of using lipschitzness to stabilize GAN training
- Negative:

> Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs). We experimented with simple variants (such as projecting the weights to a sphere) with little difference, and we stuck with weight clipping due to its simplicity and already good performance. However, we do leave the topic of enforcing Lipschitz constraints in a neural network setting for further investigation, and we actively encourage interested researchers to improve on this method.

(Arjovsky et al 2017)

WGAN-GP: Gradient Penalty for Lipschitzness

Improved Training of Wasserstein GANs

Ishaan Gulrajani[1]*, **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

Abstract

Generative Adversarial Networks (GANs) are powerful generative models, but suffer from training instability. The recently proposed Wasserstein GAN (WGAN) makes progress toward stable training of GANs, but sometimes can still generate only poor samples or fail to converge. We find that these problems are often due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can lead to undesired behavior. We propose an alternative to clipping weights: penalize the norm of gradient of the critic with respect to its input. Our proposed method performs better than standard WGAN and enables stable training of a wide variety of GAN architectures with almost no hyperparameter tuning, including 101-layer ResNets and language models with continuous generators. We also achieve high quality generations on CIFAR-10 and LSUN bedrooms. †

Gulrajani et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN-GP: Gradient Penalty for Lipschitzness

Proposition 1. *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions in $\mathcal{X}$, a compact metric space. Then, there is a 1-Lipschitz function f^* which is the optimal solution of* $\max_{\|f\|_L \leq 1} \mathbb{E}_{y \sim \mathbb{P}_r}[f(y)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)]$. *Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$, defined as the minimizer of:* $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\pi}[\|x - y\|]$ *where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of joint distributions $\pi(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. Then, if f^* is differentiable[‡], $\pi(x = y) = 0$[§], and $x_t = tx + (1-t)y$ with $0 \leq t \leq 1$, it holds that* $\mathbb{P}_{(x,y)\sim\pi}\left[\nabla f^*(x_t) = \frac{y - x_t}{\|y - x_t\|}\right] = 1$.

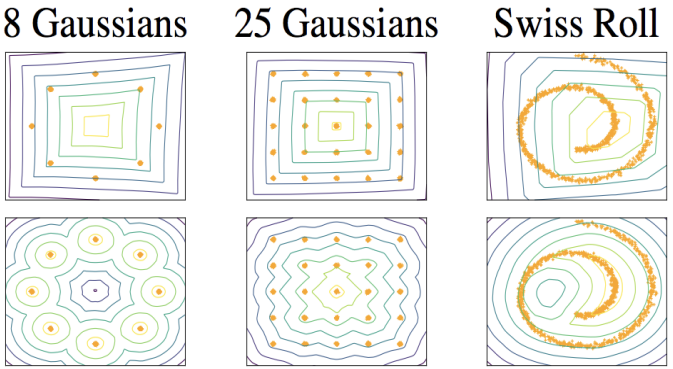
Corollary 1. *f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$.*

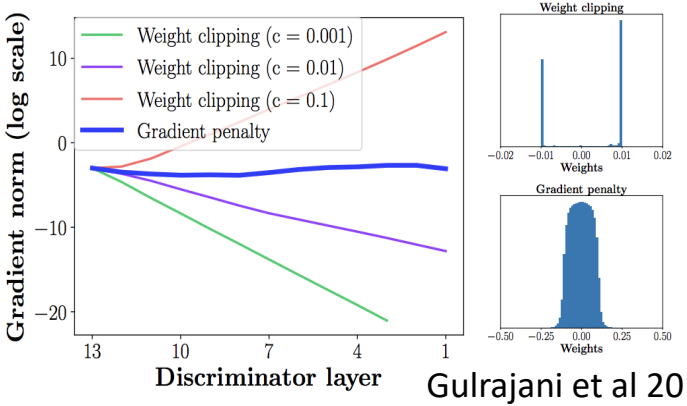
Gulrajani et al 2017

WGAN-GP: Gradient Penalty for Lipschitzness

$$\max_D \underbrace{\mathbb{E}_{x\sim P_r}\left[D(x)\right] - \mathbb{E}_{\tilde{x}\sim P_g}\left[D(\tilde{x})\right]}_{\text{Wasserstein critic objective}} + \lambda \underbrace{\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right]}_{\text{Gradient Penalty for Lipschitzness}}$$

$$\hat{x} \leftarrow \epsilon x + (1-\epsilon)\tilde{x}$$





Gulrajani et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN-GP: Pseudocode

Algorithm 1 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

Gulrajani et al 2017

WGAN-GP: BatchNorm

No critic batch normalization Most prior GAN implementations [22, 23, 2] use batch normalization in both the generator and the discriminator to help stabilize training, but batch normalization changes the form of the discriminator's problem from mapping a single input to a single output to mapping from an entire batch of inputs to a batch of outputs [23]. Our penalized training objective is no longer valid in this setting, since we penalize the norm of the critic's gradient with respect to each input independently, and not the entire batch. To resolve this, we simply omit batch normalization in the critic in our models, finding that they perform well without it. Our method works with normalization schemes which don't introduce correlations between examples. In particular, we recommend layer normalization [3] as a drop-in replacement for batch normalization.

Gulrajani et al 2017

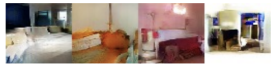
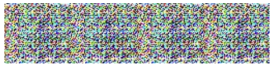
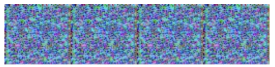
WGAN-GP: Robustness to architectures

Nonlinearity (G)	[ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]
Nonlinearity (D)	[ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]
Depth (G)	[4, 8, 12, 20]
Depth (D)	[4, 8, 12, 20]
Batch norm (G)	[True, False]
Batch norm (D; layer norm for WGAN-GP)	[True, False]
Base filter count (G)	[32, 64, 128]
Base filter count (D)	[32, 64, 128]

Min. score	Only GAN	Only WGAN-GP	Both succeeded	Both failed
1.0	0	8	192	0
3.0	1	88	110	1
5.0	0	147	42	11
7.0	1	104	5	90
9.0	0	0	0	200

Gulrajani et al 2017

WGAN-GP: Robustness to architectures

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)				
G: No BN and a constant number of filters, D: DCGAN				
G: 4-layer 512-dim ReLU MLP, D: DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

Gulrajani et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN-GP: High quality samples

Gulrajani et al 2017

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

WGAN-GP: High quality samples

Table 3: Inception scores on CIFAR-10. Our unsupervised model achieves state-of-the-art performance, and our conditional model outperforms all others except SGAN.

Unsupervised		Supervised	
Method	Score	Method	Score
ALI [8] (in [27])	$5.34 \pm .05$	SteinGAN [26]	6.35
BEGAN [4]	5.62	DCGAN (with labels, in [26])	6.58
DCGAN [22] (in [11])	$6.16 \pm .07$	Improved GAN [23]	$8.09 \pm .07$
Improved GAN (-L+HA) [23]	$6.86 \pm .06$	AC-GAN [20]	$8.25 \pm .07$
EGAN-Ent-VI [7]	$7.07 \pm .10$	SGAN-no-joint [11]	$8.37 \pm .08$
DFM [27]	$7.72 \pm .13$	WGAN-GP ResNet (ours)	$8.42 \pm .10$
WGAN-GP ResNet (ours)	$7.86 \pm .07$	**SGAN** [11]	$8.59 \pm .12$

Gulrajani et al 2017