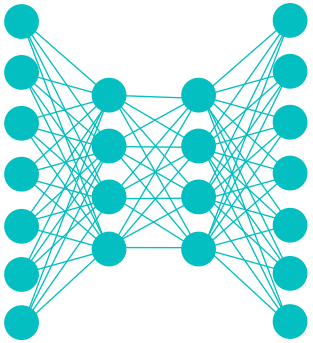
Lecture Notes for

Neural Networks and Machine Learning

Wasserstein GANs



Logistics and Agenda

- Logistics
  - **Student Presentation**: None
- Agenda
  - Wasserstein GAN
  - WGAN-GP
  - Demo
  - Town Hall

Last Time:

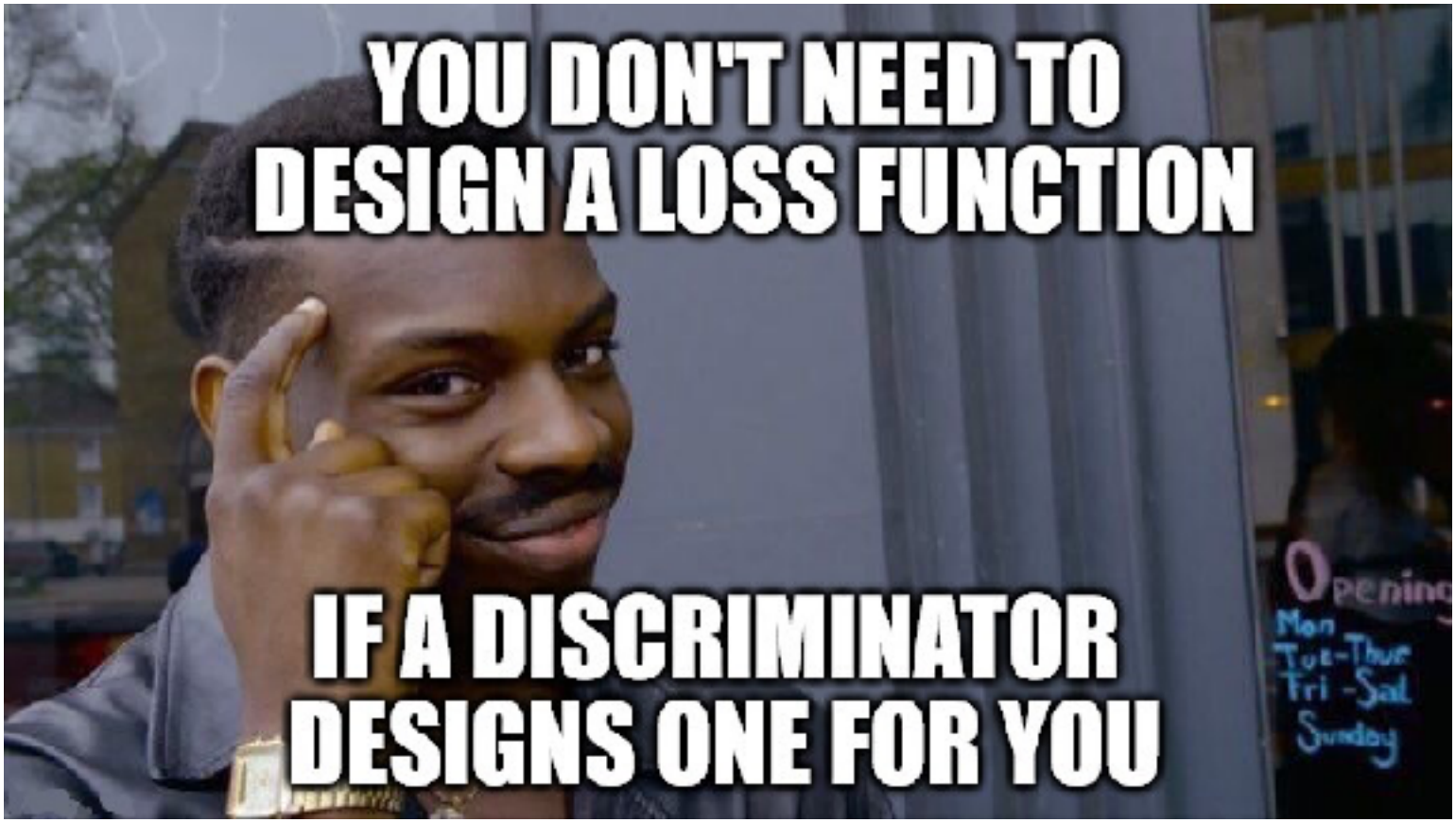
- It can be shown that the generator objective function (according to the Goodfellow loss) is optimal when it maximizes this:

$$C(G) = -\log(4) + KL\left(p_{data} \parallel \frac{p_{data} + p_g}{2}\right) + KL\left(p_g \parallel \frac{p_{data} + p_g}{2}\right)$$

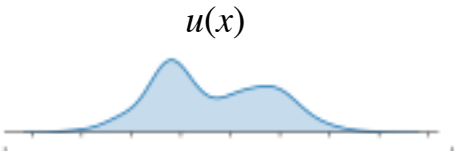
$$C(G) = -\log(4) + 2.JSD(p_{data} \parallel p_g)$$ Jenson-Shannon Divergence

- Which helps explain the **Vanishing gradients**: if $p_{data} \| p_g \approx 0$ then JSD saturates and the gradient is basically zero. Optimization has no idea what direction to move in…
  - **Indirect Solution:** Perhaps we can just use tips/tricks to get around the vanishing gradient problem?
    - Not covering these methods this semester …
  - **Direct Solution**: get rid of the loss function from Goodfellow
    - **Use a better objective: Wasserstein Distance**

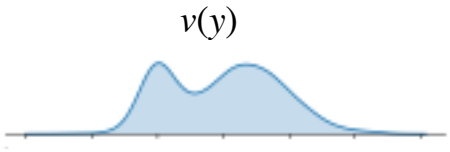
Wasserstein GANs

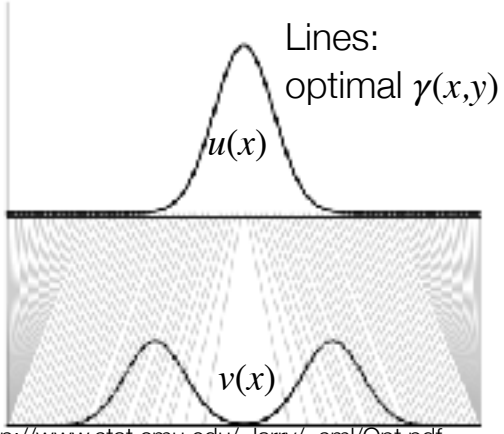


Wasserstein Distance (Earth Mover)



how to move dirt from u to v ?

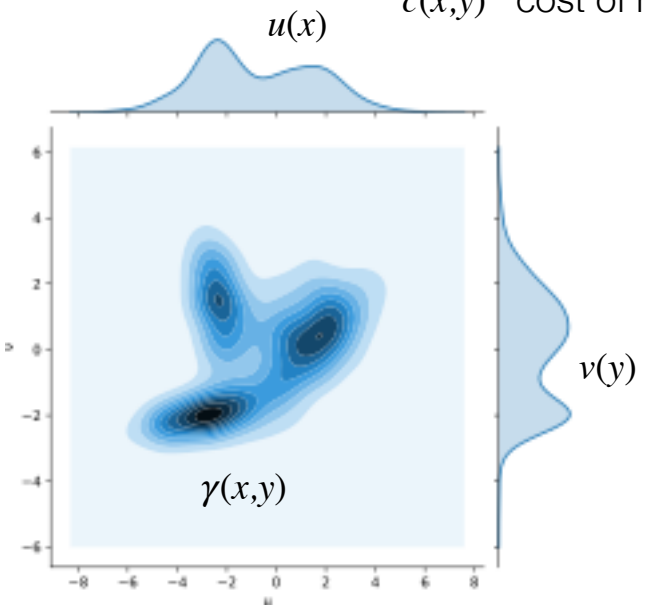




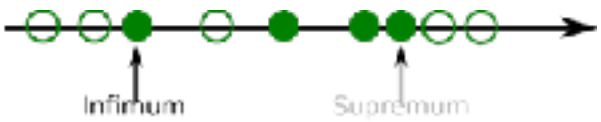
http://www.stat.cmu.edu/~larry/=sml/Opt.pdf

$\gamma(x,y)$ one plan to move u to v

$c(x,y)$ cost of moving u to v



$$\int\int c(x,y)\cdot\gamma(x,y)\,\partial x\partial y$$ Total cost of plan γ

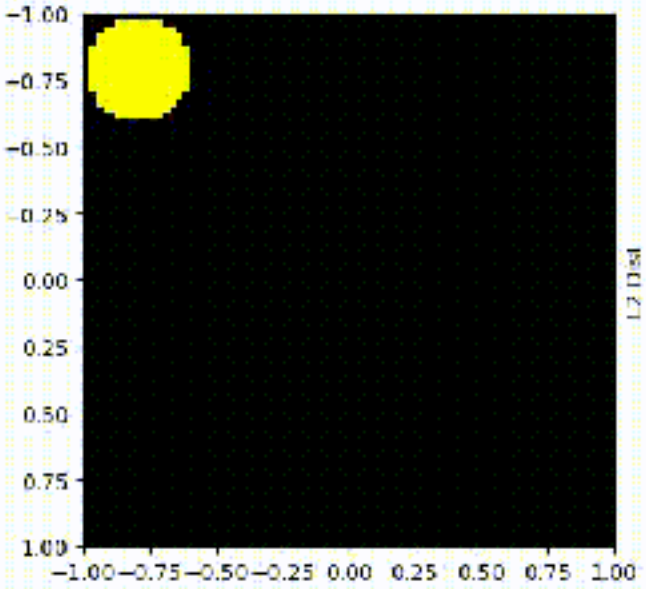


$$\inf_{\gamma\in\Pi}\int\int c(x,y)\cdot\gamma(x,y)\,\partial x\partial y$$ Optimal plan has greatest lower bound (minimum cost in set Π)

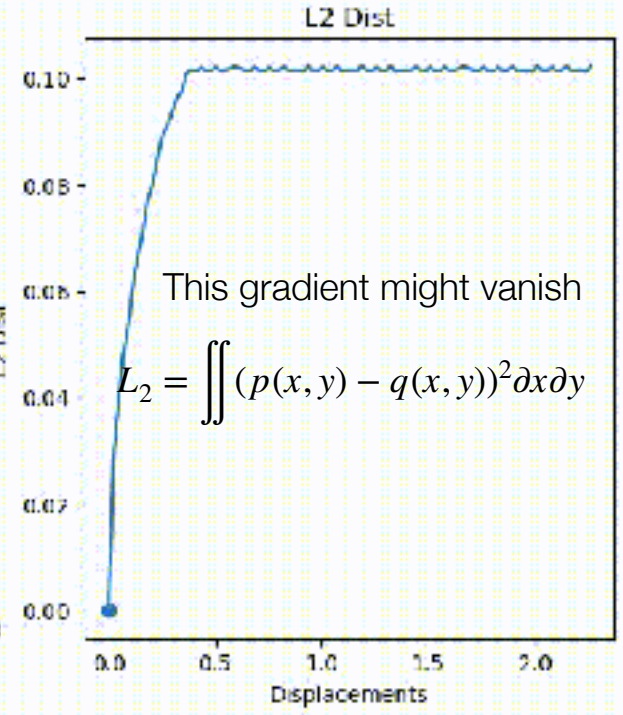
$$\inf \mathbf{E}_{x,y\leftarrow\gamma}\left[|x-y|\right] = \inf_{\gamma\in\Pi}\int\int |x-y|\cdot\gamma(x,y)\,\partial x\partial y$$ If cost is difference to move from x to y

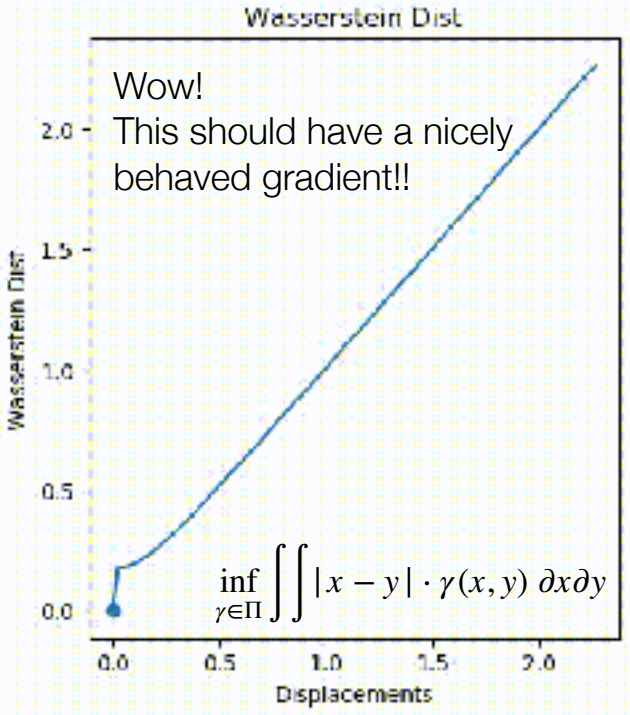
image: wikipedia

Wasserstein Distance



Two 2D Normal Distributions





- *Wasserstein GAN adds few tricks to allow the Discriminator to approximate Wasserstein (aka Earth Mover's) distance between real and model distributions. Wasserstein distance roughly tells "how much work is needed to be done for one distribution to be adjusted to match another" and is remarkable in a way that it is defined even for non-overlapping distributions.*

https://gist.github.com/ctralie/66352ae6ab06c009f02c705385a446f3

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Wasserstein Distance

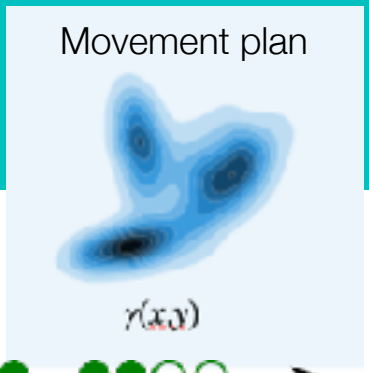
- How much do I need to change one distribution to get it to match another?
- Also, we will only have **samples** from the distributions (not the actual equations)
- **Wasserstein Distance** for continuous probabilities:

$$W(p_{data}, p_g) = \inf_{\gamma \in \prod(p_{data}, p_g)} \mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| \right]$$

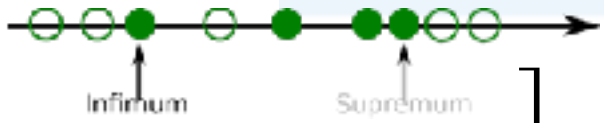
- $\inf$ is greatest lower bound (min of a set)
- γ is completely and utterly

intractable to **compute**

Wasserstein Duality, Critic



$$W(p_{data}, p_g) = \inf_{\gamma \in \prod(p_{data}, p_g)} \mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| \right]$$



$$= \inf_{\gamma} \mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| + \underbrace{\sup_{f} \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)] - \big(f(x) - f(y)\big)}_{\text{enforce constraint = finite, if } \gamma \in \Pi, \text{ else } +\infty} \right]$$

$$= \inf_{\gamma} \sup_{f} \mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| + \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)] - \big(f(x) - f(y)\big) \right]$$

$$= \sup_{f} \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)] - \inf_{\gamma} \underbrace{\mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| + \big(f(x) - f(y)\big) \right]}_{\text{new constraint = finite, if } |f| \leq 1, \text{ else } -\infty}$$

$$= \sup_{|f| \leq 1} \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)]$$

what have we done here????

read this: https://vincentherrmann.github.io/blog/wasserstein/

Wasserstein (Kantorovich-Rubinstein) Duality

- 1-Lipschitz constraint, critic: $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$
- Wasserstein Duality Formula (approx. for samples):

$$\mathscr{L}_{wass} = \sup_{|f| \leq 1} \mathbf{E}[f(x_{real})] - \mathbf{E}[f(x_{fake})] \approx \frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) - \frac{1}{n}\sum_{j=1}^{n} f(g(z^{(j)}))$$

- where sup is least upper bound (max within set $|f|<1$, same as valid movement plans set)
- f will be the critic, learned as a neural network, $m=n$

$$\mathscr{L}_{wass}^{D} = \frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) - f(g(z^{(i)}))$$

Max with f (called critic), freeze generator weights

$$\mathscr{L}_{wass}^{G} = \frac{1}{m}\sum_{i=1}^{m} f(g(z^{(i)}))$$

Max with g, freeze weights of critic

Sometimes we use this, minimizes hinge loss:

$$\frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) \cdot f(g(z^{(i)}))$$

when *values* are -1 to 1

f( )	*f*(*g*( ))	*f*-*f*(*g*)	*f* x *f*(*g*)
-1	-1	0	1
-1	1	-2	-1
1	-1	2	-1
1	1	0	1
		max	min

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

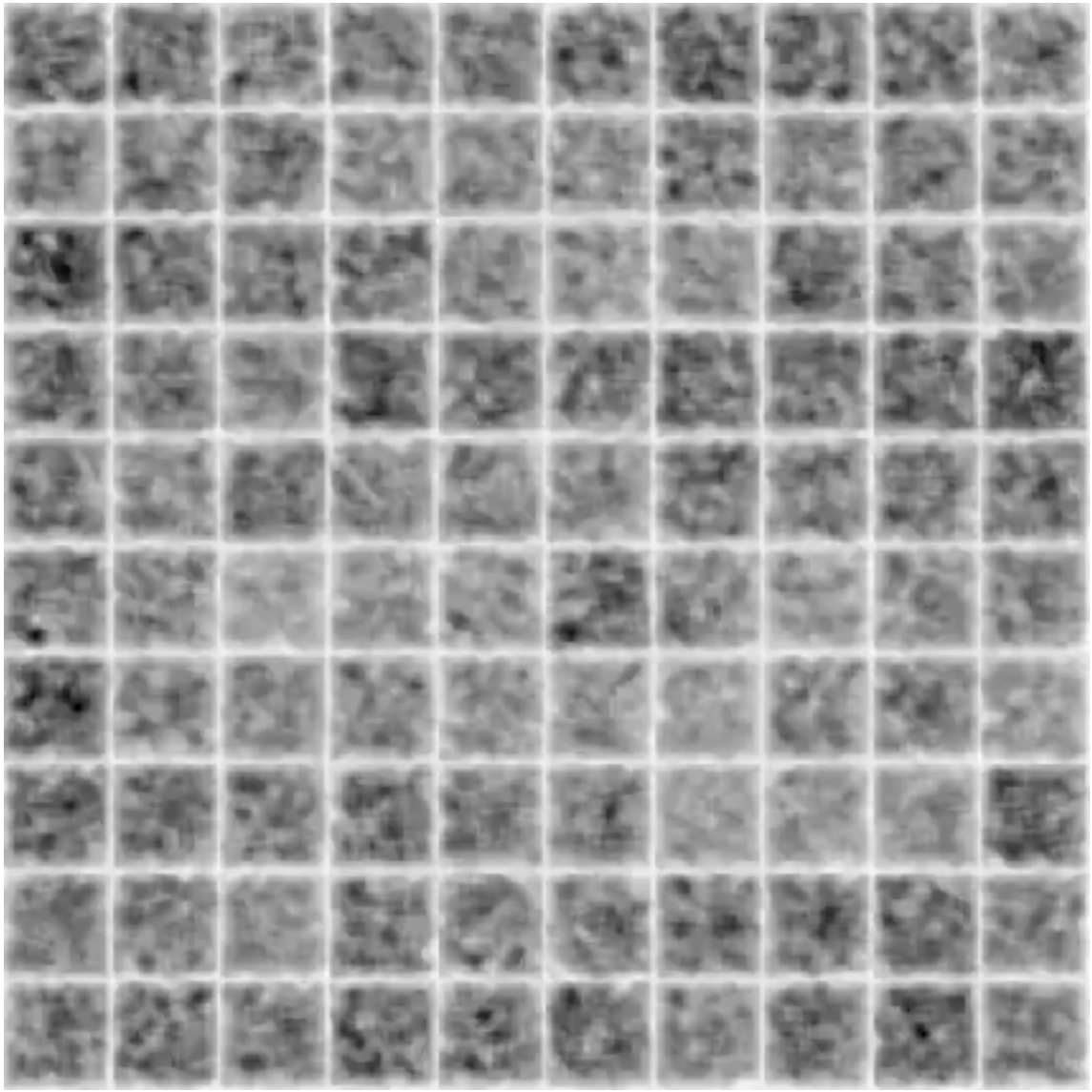
Practical Wasserstein Loss

- Discriminator (now critic) becomes **f**, and its output can be Lipschitz if all weights are small (squashes inputs)
  - so we clip them to (-c to c) (where $c \sim 0.01$)
  - critic output should be linear

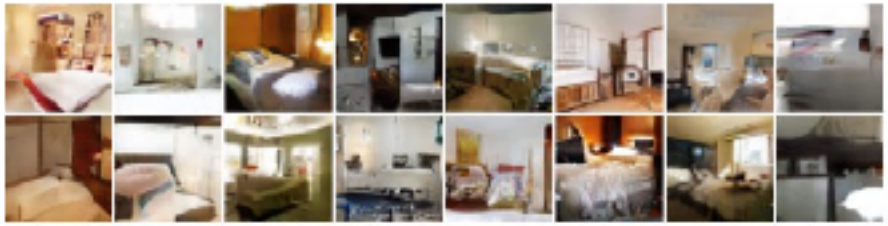
```
# define the constraint
const = ClipConstraint(0.01)
...
# use the constraint in a layer
model.add(Conv2D(..., kernel_constraint=const))
```

https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/

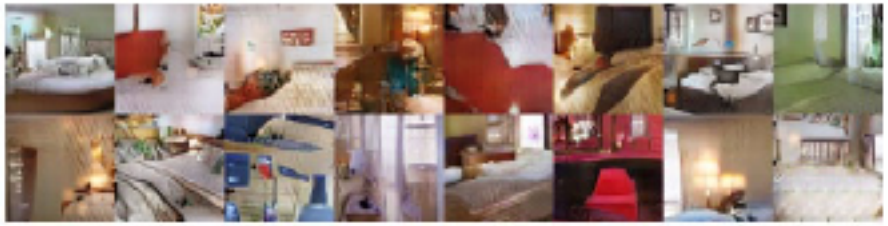
WGAN Example from Keras (MNIST)



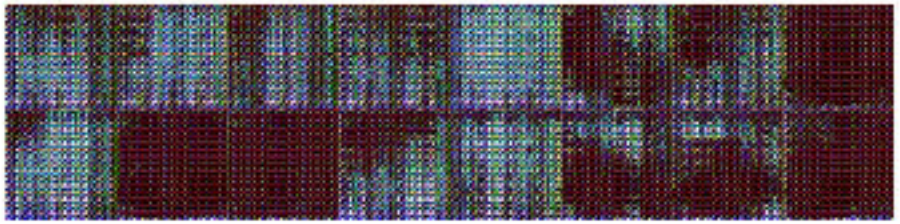
- In their empirical findings, no mode collapse problems
- And training longer resulted in good quality images (motivates that distributions overlap)
- Still some of the most competitive GAN results



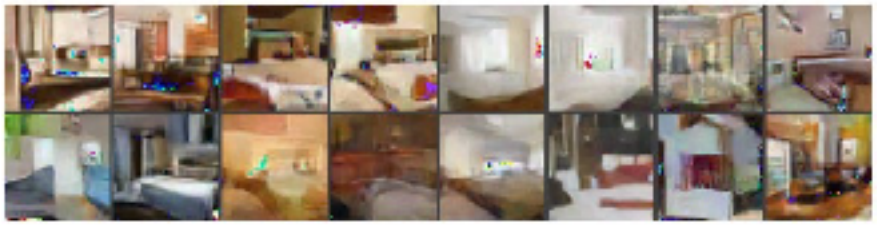
WGAN with DCGAN generator

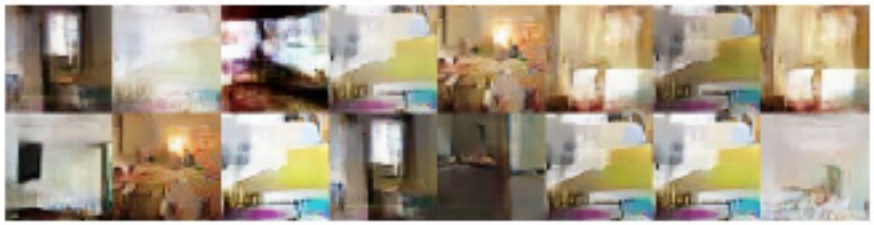


GAN with DCGAN generator



Without batch normalization &
constant number of filters at each layer





Using a MLP as the generator

So we should always use Wasserstein!

- **Actually not really**
- Its really, really,…. really slow
- Clipping (from WGAN paper):

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).

- Others have made improvements by incorporating gradient constraints
- New critic: WGAN with gradient penalty

WGAN-GP

WGAN-GP (Gradient Penalty)

$$\frac{1}{m}\sum_{i=1}^{m} f(g(z^{(i)}))$$

Maximize g,
freeze weights of critic
No change

- Theorem: a function is 1-Lipschitz if and only if its gradient norm is $\|\nabla f(\hat{x})\|_2 \leq 1$, but that is very constraining

Maximize f
freeze generator weights,

incentivize critic gradient norm to be one
Choose large lambda (e.g., 10)

$$\frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) - f(g(z^{(i)})) - \lambda \frac{1}{W}\sum_{i=1}^{W} \left(\|\nabla f(\hat{x}^{(i)})\|_2 - 1\right)^2$$

where for ϵ in U[0,1] $\hat{x}^{(i)} = \epsilon \cdot x_{real}^{(i)} + (1-\epsilon)\cdot g(z^{(i)})$

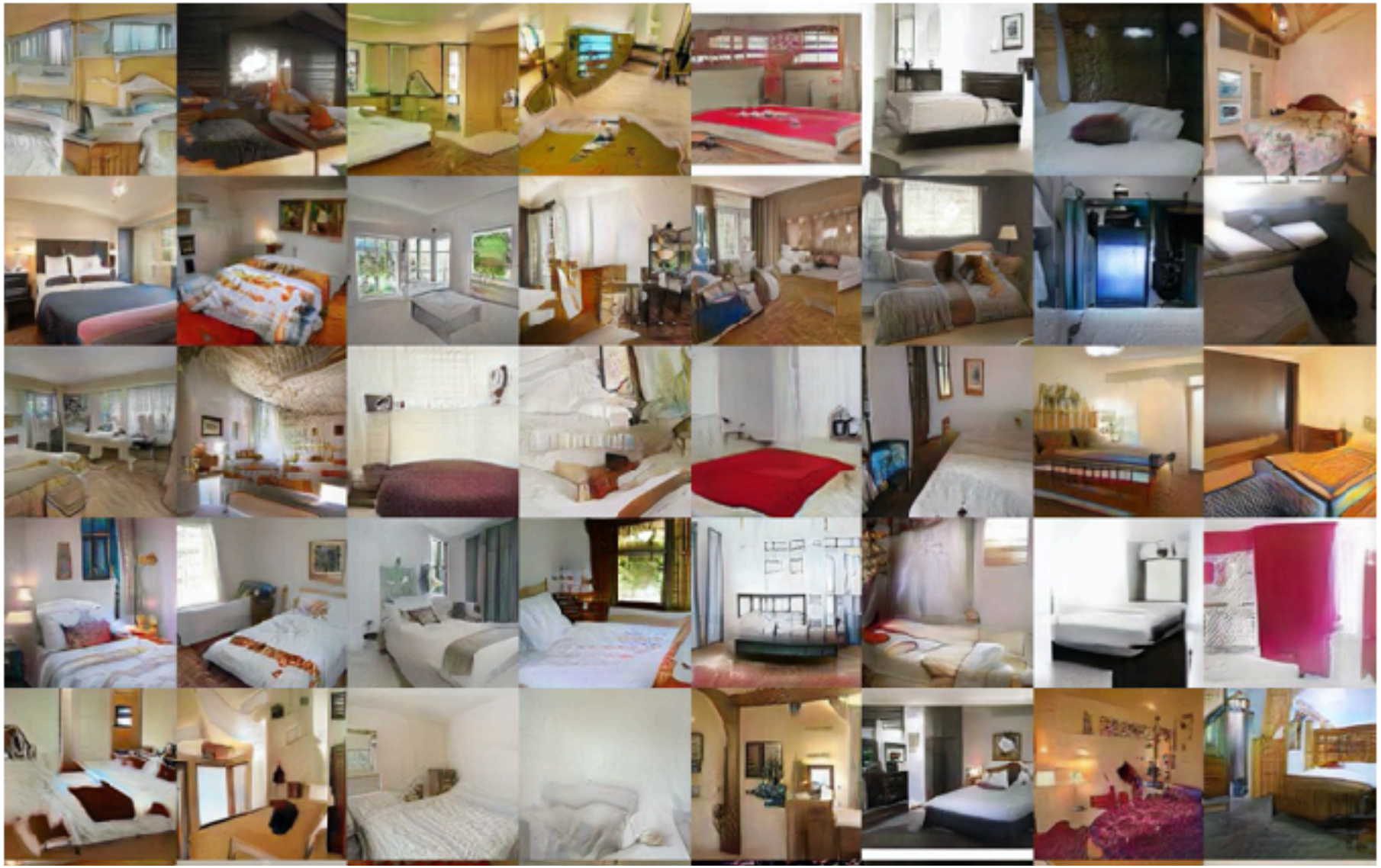
randomly mix together real and fake images, for gradient estimation

WGAN-GP Heuristics

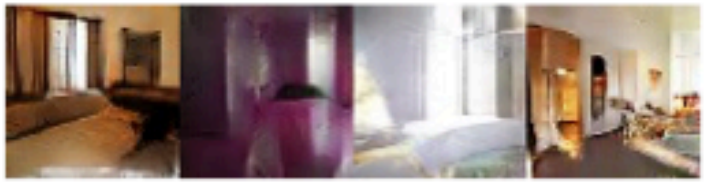
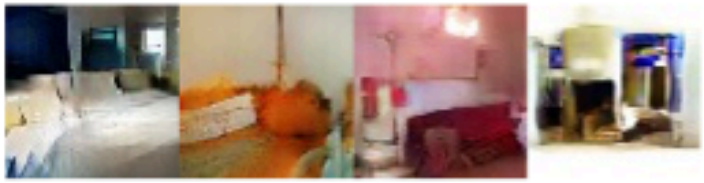
- Choose large penalty coefficient
- Don't use batch normalization (in critic)
  - it forces the gradient norm to be dependent on batches
  - but layer norm seems to be okay
- Enforce gradient norm to be close to one, rather than less than one

Two-sided penalty We encourage the norm of the gradient to go towards 1 (two-sided penalty) instead of just staying below 1 (one-sided penalty). Empirically this seems not to constrain the critic too much, likely because the optimal WGAN critic anyway has gradients with norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$ and in large portions of the region in between (see subsection 2.3). In our early observations we found this to perform slightly better, but we don't investigate this fully. We describe experiments on the one-sided penalty in the appendix.

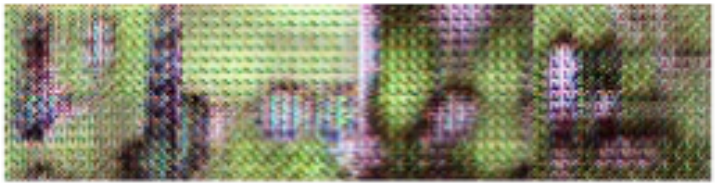
WGAN-GP Results



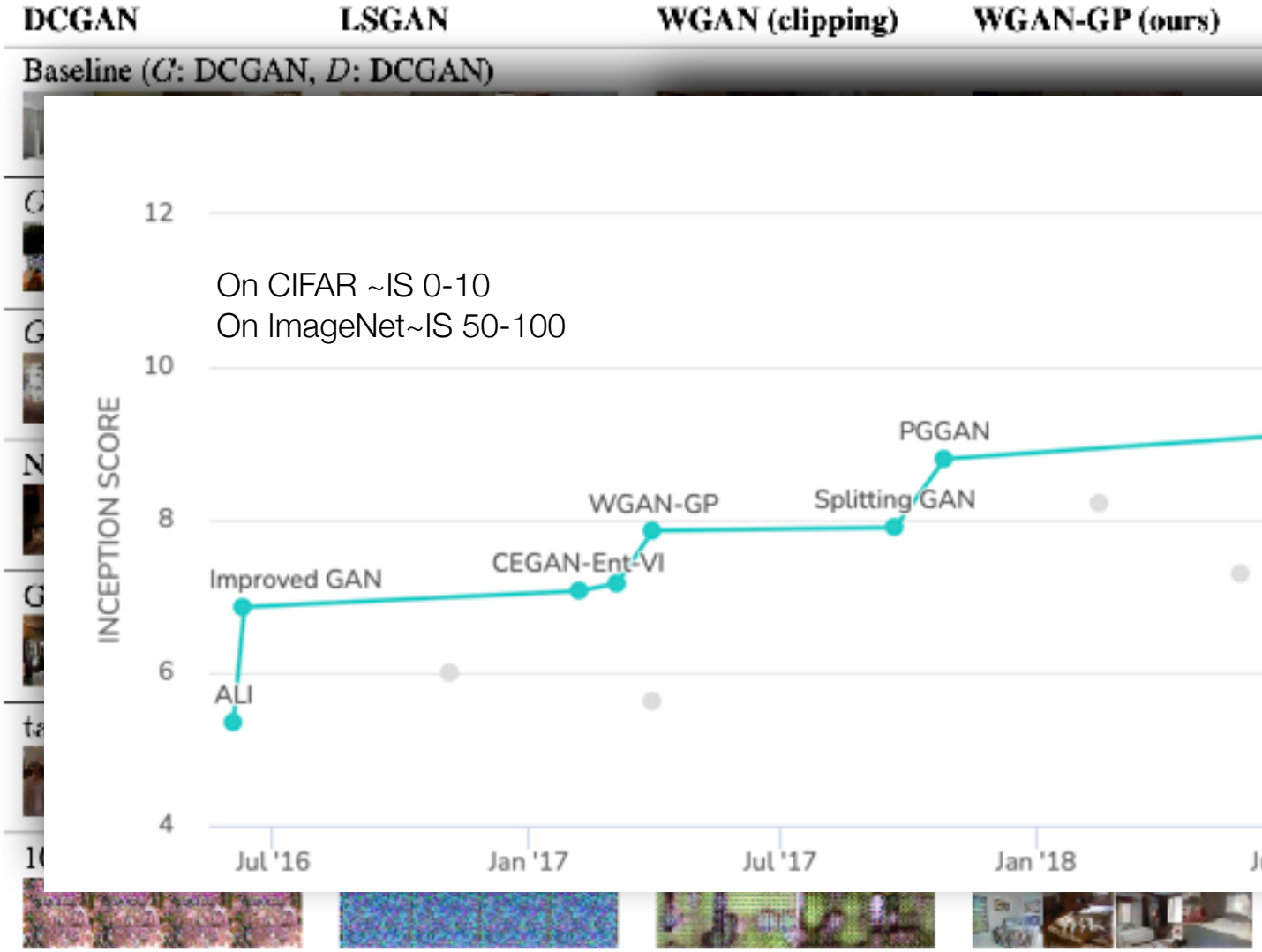
WGAN-GP Comparative Results

WGAN (clipping)	WGAN-GP (ours)
Baseline: G: DCGAN	Crit: DCGAN
	
G: DCGAN no BN	Crit: DCGAN
	
G: MLP	Crit: DCGAN
	

WGAN-GP Comparative Results

WGAN (clipping)	WGAN-GP (ours)
G/Crit: Tanh Non-linearities	
	
G/Crit: 101 Layer ResNet	
	

WGAN-GP Comparative Results



WGAN-GP

Main Repository:
07c GANsWithKeras.ipynb

Other Examples

Keras Example: https://github.com/keras-team/keras-contrib/blob/master/examples/improved_wgan.py DCGAN

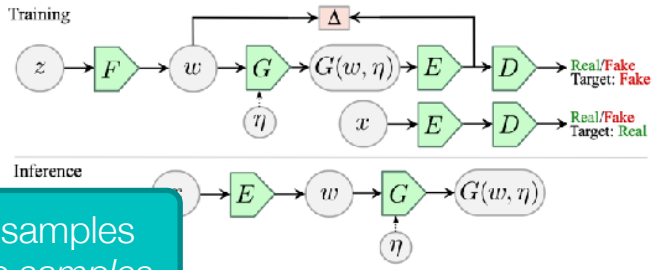
Demo by Keras Contrib Community

Other Example: https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/wgan_gp/wgan_gp.py MLP-GAN

Demo by Erik Linder-Norén

Aside: Back to ALAEs

Adversarial Latent Auto-Encoders, ALAE



Algorithm 1 ALAE Training

1: $\theta_F, \theta_G, \theta_E, \theta_D \leftarrow$ Initialize network parameters
2: **while** not converged **do**
3: Step I. Update E, and D
4: $x \leftarrow$ Random mini-batch from dataset
5: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
6: $L_{adv}^{E,D} \leftarrow \text{softplus}(D \circ E \circ G \circ F(z))) + \text{softplus}(-D \circ E(x)) +$
$\frac{\gamma}{2} \mathrm{E}_{p_{\mathcal{D}}(x)} \left[\|\nabla D \circ E(x)\|^2 \right]$
7: $\theta_E, \theta_D \leftarrow \text{ADAM}(\nabla_{\theta_D, \theta_E} L_{adv}^{E,D}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8: Step II. Update F, and G
9: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10: $L_{adv}^{F,G} \leftarrow \text{softplus}(-D \circ E \circ G \circ F(z)))$
11: $\theta_F, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_F, \theta_G} L_{adv}^{F,G}, \theta_F, \theta_G, \alpha, \beta_1, \beta_2)$
12: Step III. Update E, and G
13: $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
14: $L_{error}^{E,G} \leftarrow \|F(z) - E \circ G \circ F(z)\|_2^2$
15: $\theta_E, \theta_G \leftarrow \text{ADAM}(\nabla_{\theta_E, \theta_G} L_{error}^{E,G}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**

E,D: Detect fake samples
minimize D for fake samples

E,D: Detect real samples
minimize -D for real samples

E,D: Gradient Penalty
Keep Gradient Magnitude Small
Now we know why!!

F,G: Try to fool discriminator,
minimize -D for fake samples

E,G: Keep latent spaces similar

Pidhorskyi, Stanislav, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial latent autoencoders." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14104-14113. 2020.

GAN Town Hall

Lecture Notes for

Neural Networks and Machine Learning

Wasserstein GANs

Next Time:
BigGAN and StyleGAN

Reading: Those Papers!