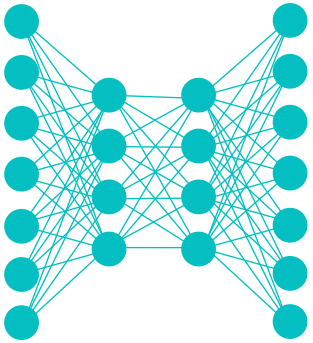
Lecture Notes for

Neural Networks and Machine Learning

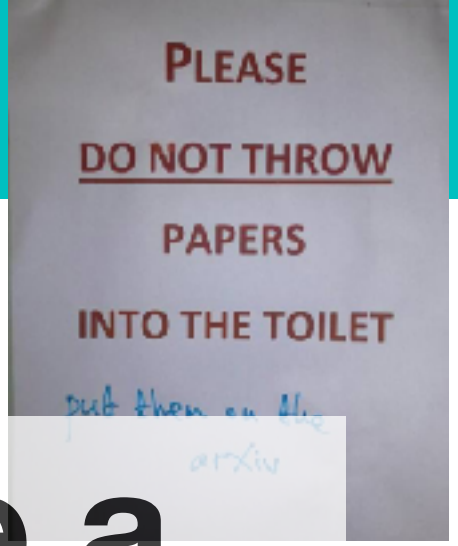
Wasserstein GANs



Logistics and Agenda

- Logistics
  - **Student Presentation**: Next Time, GLOW
- Agenda
  - Wasserstein GAN
  - WGAN-GP
  - Demo
  - BigGAN

Change the loss function



Name	Value Function
GAN	$L_D^{GAN} = E[\log(D(x))] + E[\log(1 - D(G(z)))]$ $L_G^{GAN} = E[\log(D(G(z)))]$
LSGAN	$L_D^{LSGAN} = E[(D(x) - 1)^2] + E[D(G(z))^2]$ $L_G^{LSGAN} = E[(D(G(z)) - 1)^2]$
WGAN	$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$ $L_G^{WGAN} = E[D(G(z))]$ $W_D \leftarrow clip\_by\_value(W_D, -0.01, 0.01)$
WGAN_GP	$L_D^{WGAN\_GP} = L_D^{WGAN} + \lambda E[(\lVert \nabla D(\alpha x - (1-\alpha)G(z)) \rVert - 1)^2]$ $L_G^{WGAN\_GP} = L_G^{WGAN}$
DRAGAN	$L_D^{DRAGAN} = L_D^{GAN} + \lambda E[(\lVert \nabla D(\alpha x - (1-\alpha)(x + \delta)) \rVert - 1)^2]$ $L_G^{DRAGAN} = L_G^{GAN}$
CGAN	$L_D^{CGAN} = E[\log(D(x,c))] + E[\log(1 - D(G(z),c))]$ $L_G^{CGAN} = E[\log(D(G(z),c))]$
InfoGAN	$L_{D,Q}^{InfoGAN} = L_D^{GAN} - \lambda L_I(x,c)$ $L_{G,Q}^{InfoGAN} = L_G^{GAN} - \lambda L_I(G(z),c)$
ACGAN	$L_D^{ACGAN} = L_D^{GAN} + E[P(class = c \mid x)] + E[P(class = c \mid G(z))]$ $L_G^{ACGAN} = L_G^{GAN} + E[P(class = c \mid G(z))]$
EBGAN	$L_D^{EBGAN} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$ $L_G^{EBGAN} = D_{AE}(G(z)) + \lambda \cdot PT$
BEGAN	$L_D^{BEGAN} = D_{AE}(x) - k_t D_{AE}(G(z))$ $L_G^{BEGAN} = D_{AE}(G(z))$ $k_{t+1} = k_t + \lambda(\gamma D_{AE}(x) - D_{AE}(G(z)))$

There has to be a more principled way of doing this!!!!

The best loss function to use:

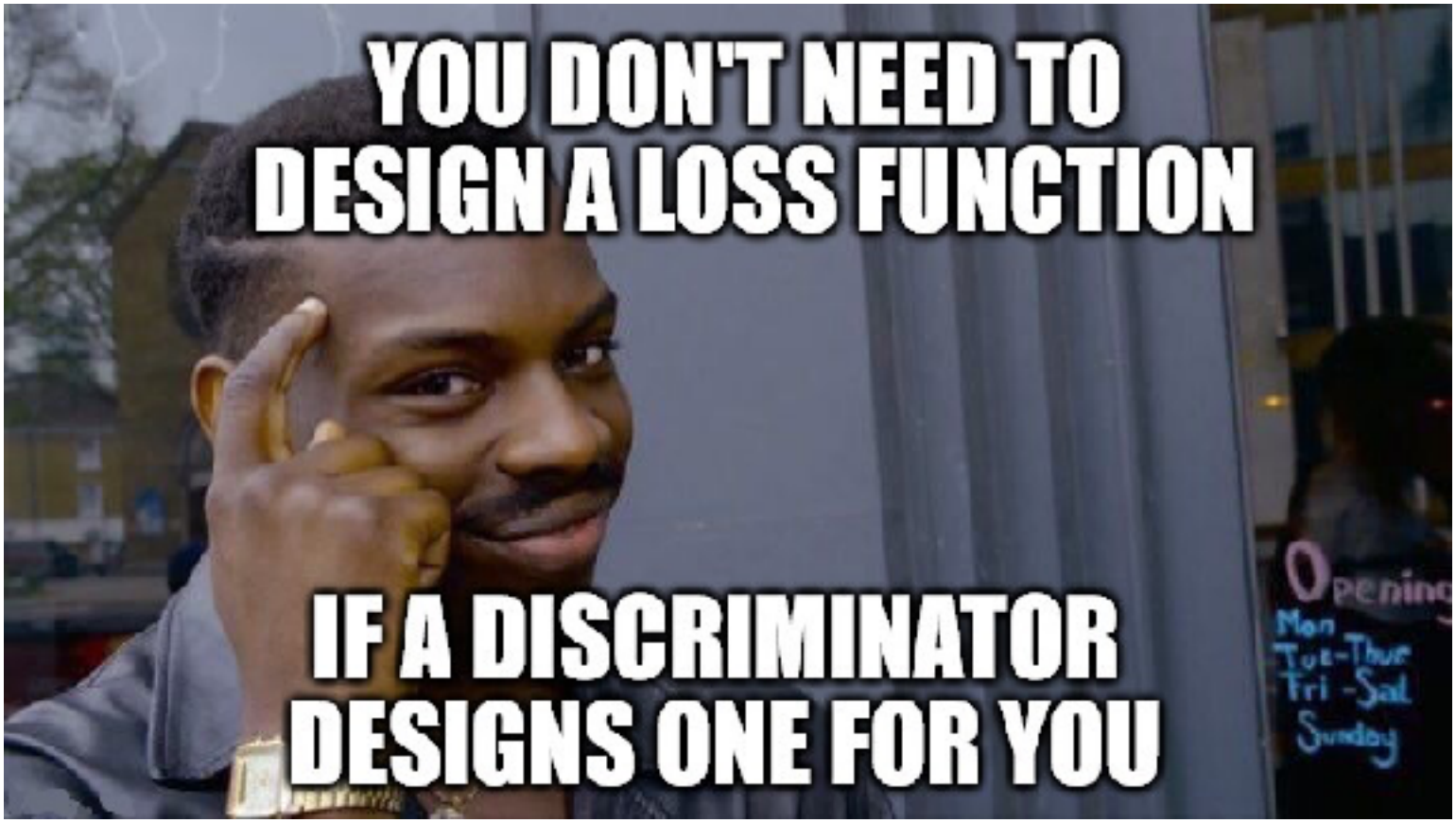
$$E_{(x,y)\sim p_{data}}[-\log D(x,y)] - E_{z\sim p_z}[\log(1 - D(G(z,y),y))]$$

	MNIST	FASHION	CIFAR	CELEBA
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8

Academic blasphemy:
Maybe it doesn't really matter

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

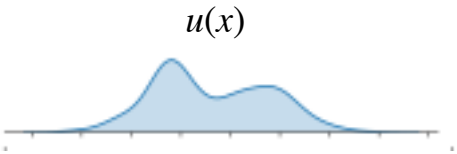
Wasserstein GANs



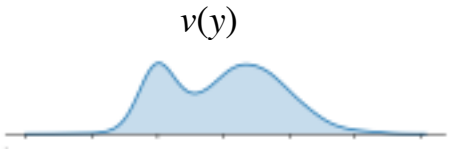
Caveat

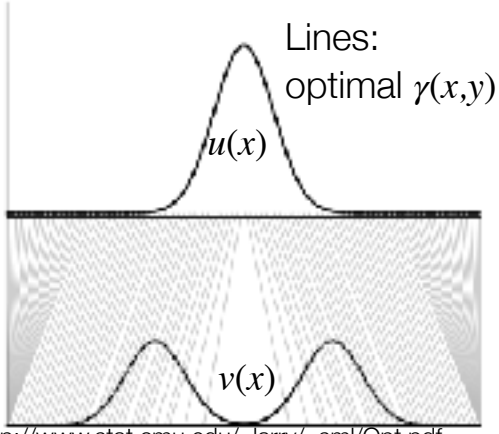
- Wasserstein Loss is actually very useful
  - But the mathematics pinning it down are based on approximations (like always)
  - So its **not really true** to say its actually working for the motivated reasons
  - But **it might be**
- So until we have a better theory, **lets assume it works** because it truly is a great distance measure

Wasserstein Distance (Earth Mover)



how to move dirt from u to v ?

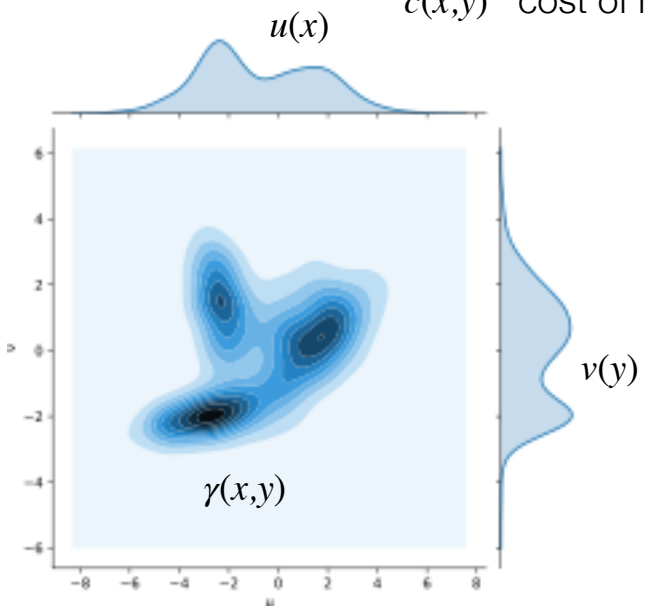




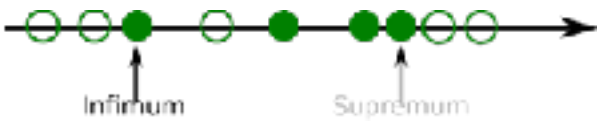
http://www.stat.cmu.edu/~larry/=sml/Opt.pdf

$\gamma(x,y)$ 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

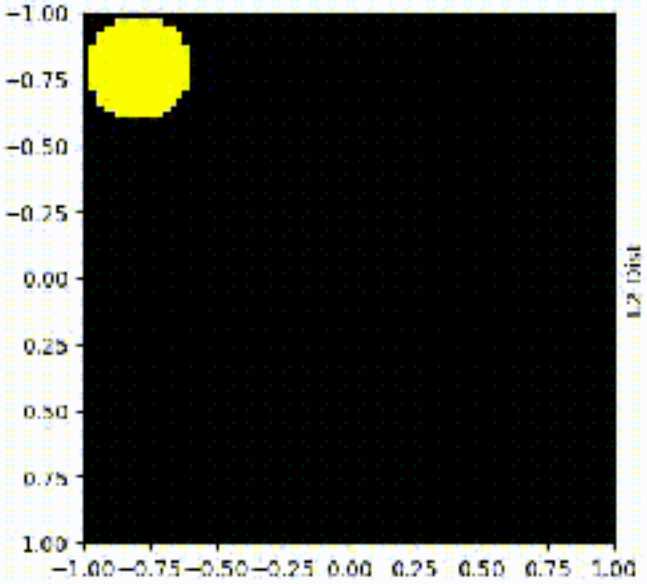


$$\inf_{\gamma\in\Pi}\int\int c(x,y)\cdot\gamma(x,y)\,\partial x\partial y$$ Optimal plan has greatest lower bound (minimum cost in set Π)

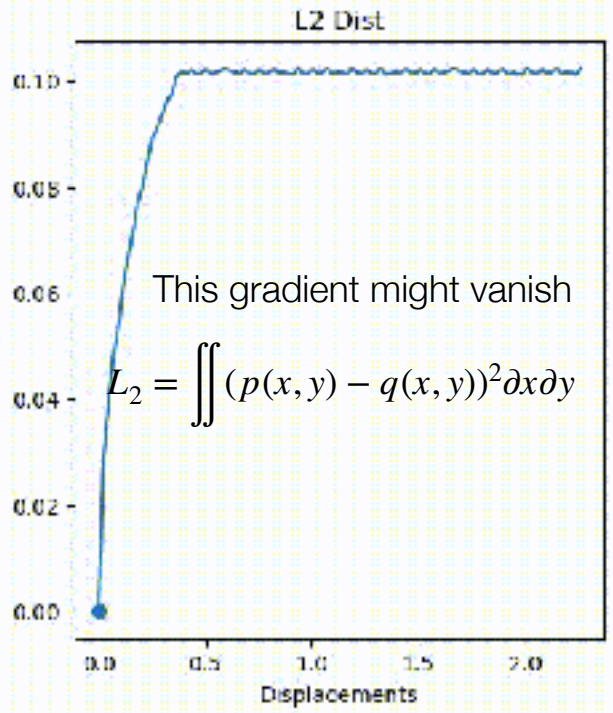
$$\inf \mathbf{E}_{x,y\leftarrow\gamma}\left[|x-y|\right] = \inf_{\gamma\in\Pi}\int\int |x-y|\cdot\gamma(x,y)\,\partial x\partial y$$ If cost is difference to move from x to y

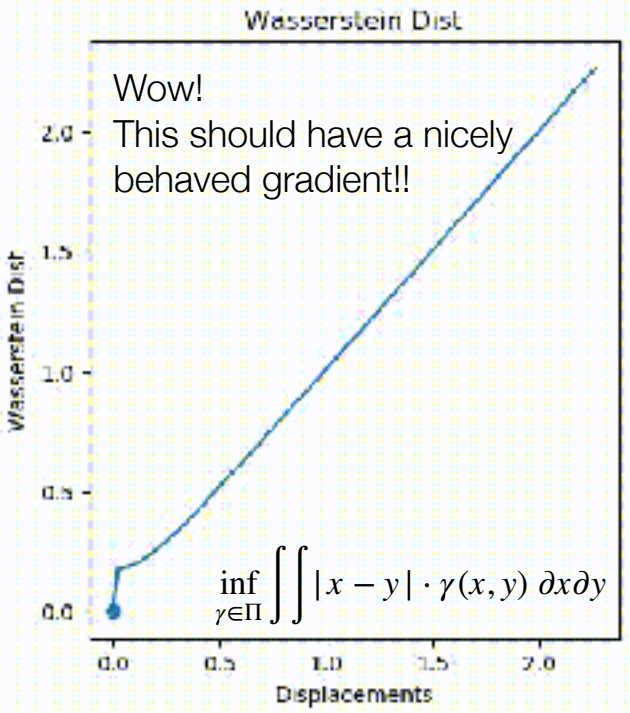
image: wikipedia

Wasserstein Distance



Two 2D Normal Distributions





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

Wasserstein

- *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.*
- That should help training even when the generator and discriminator are not matching
- …but Wasserstein Distance is not tractable to calculate for generative distributions so its time to start approximating (but only a little)!

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

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

$$= \inf_{\gamma} \mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| + \underbrace{\sup_{f} \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)] - \left(f(x) - f(y)\right)}_{\text{enforce constraint } =0, \text{ if } \gamma \in \Pi, \text{ else } +\infty} \right]$$

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

$$= \sup_{f} \mathbf{E}_{x \leftarrow p_{data}}[f(x)] - \mathbf{E}_{x \leftarrow p_g}[f(x)] - \inf_{\gamma} \underbrace{\mathbf{E}_{x,y \leftarrow \gamma} \left[\|x - y\| + \left(f(x) - f(y)\right) \right]}_{\text{new constraint } =0, \text{ if } |f| \leq 1, \text{ else } -\infty}$$

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

what have we done here????

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

Wasserstein 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 (maximization within set)
- f will be the critic, learned as a neural network, m=n

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

Maximize f (called critic), freeze generator weights

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

Maximize g, freeze weights of critic

Sometimes this for critic minimize hinge loss:

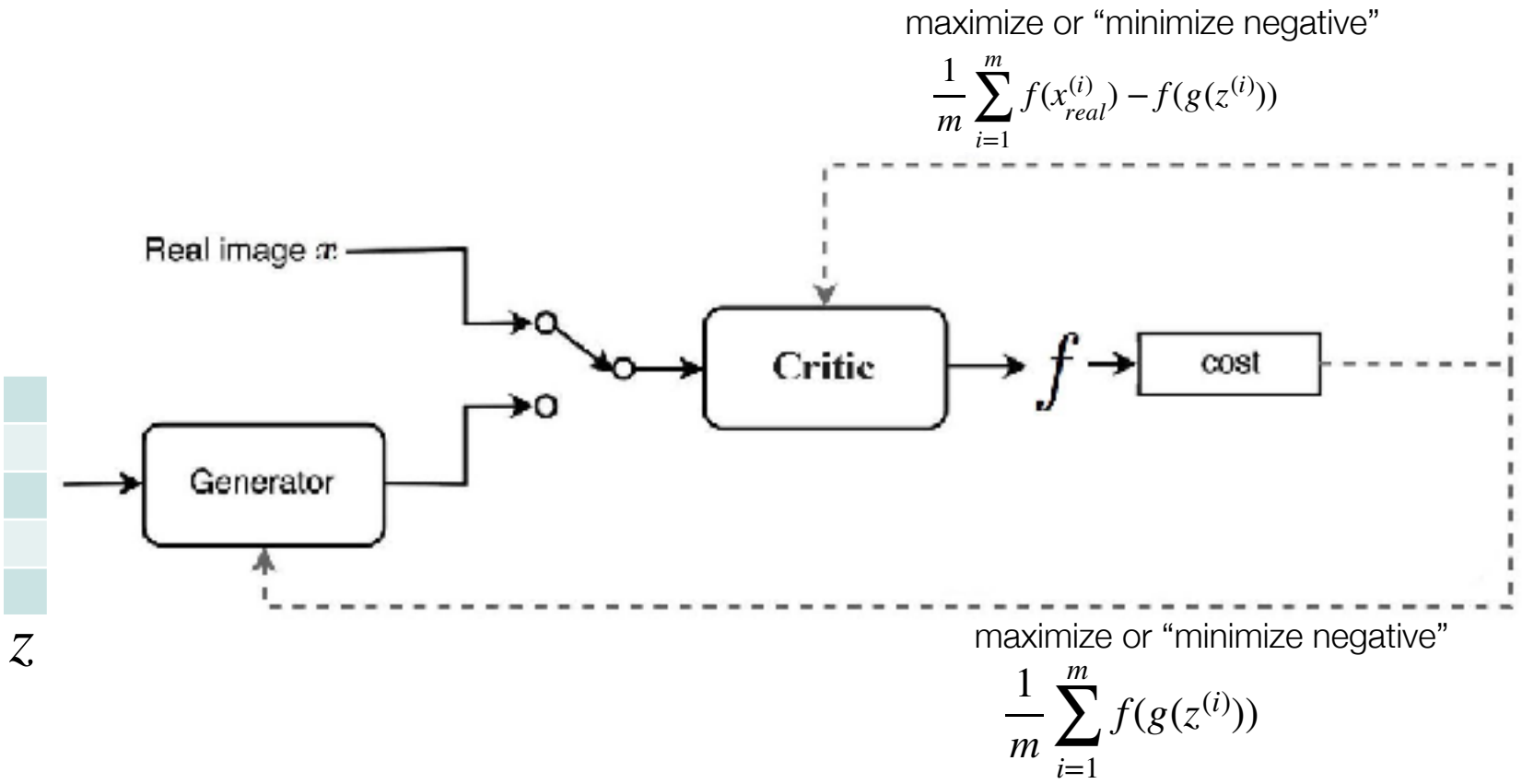
$$\frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) \cdot f(g(z^{(i)}))$$

when *label* is -1 and 1

real	fake	f(x)-g(x)	f(x)*g(x)
-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

Wasserstein Architecture



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

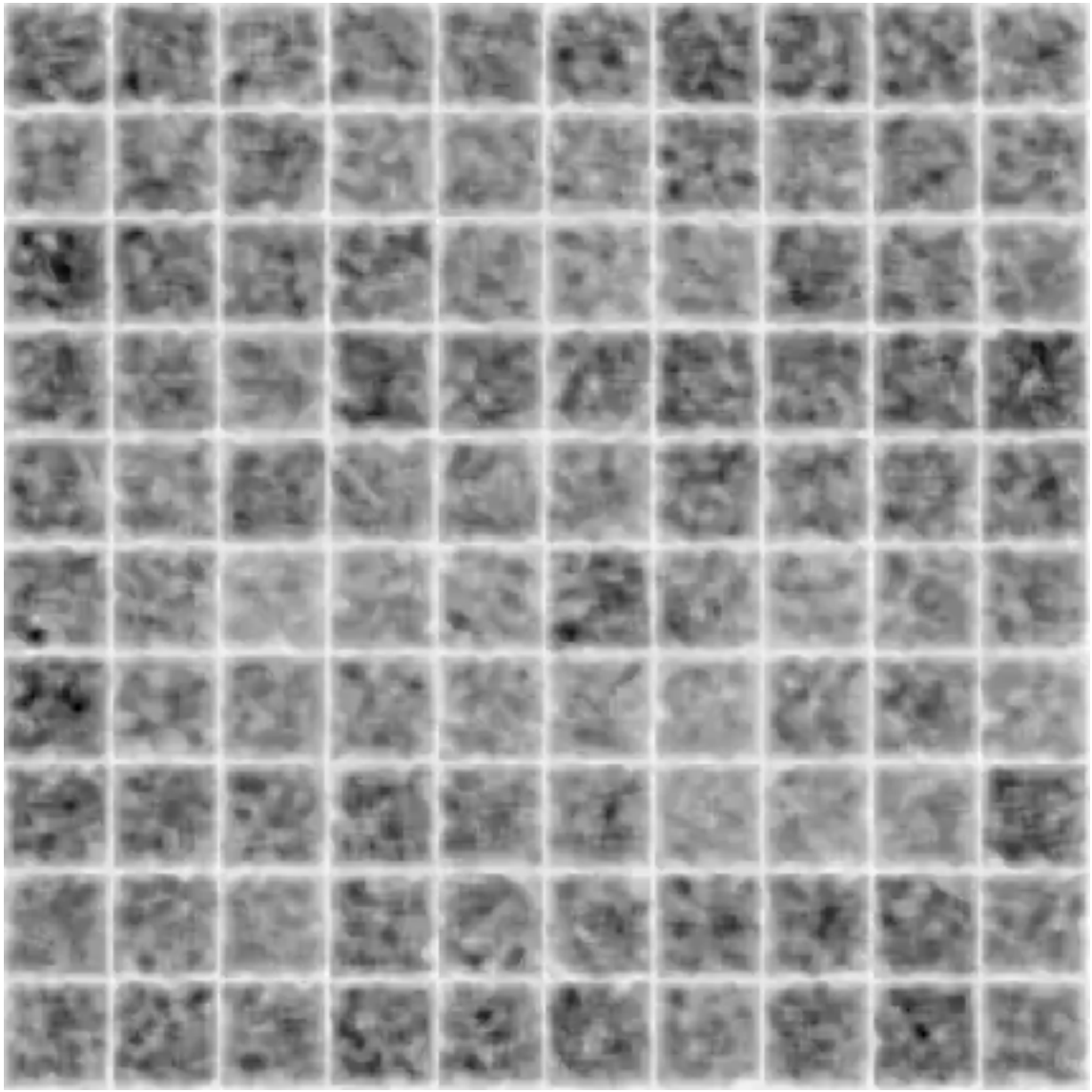
Practical Wasserstein Loss

- 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 standalone critic model
def define_critic(in_shape=(28,28,1)):
    # weight initialization
    init = RandomNormal(stddev=0.02)
    # weight constraint
    const = ClipConstraint(0.01)
    # define model
    model = Sequential()
    # downsample to 14x14
    model.add(Conv2D(64, (4,4), strides=(2,2), padding='same'
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))
    # downsample to 7x7
    model.add(Conv2D(64, (4,4), strides=(2,2), padding='same'
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=0.2))
    # scoring, linear activation
    model.add(Flatten())
    model.add(Dense(1))
```

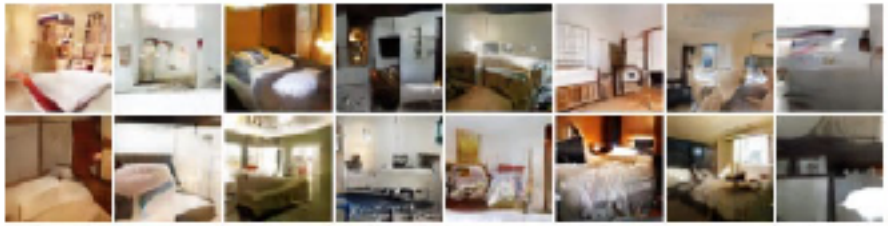
https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/

WGAN Example from Keras (MNIST)

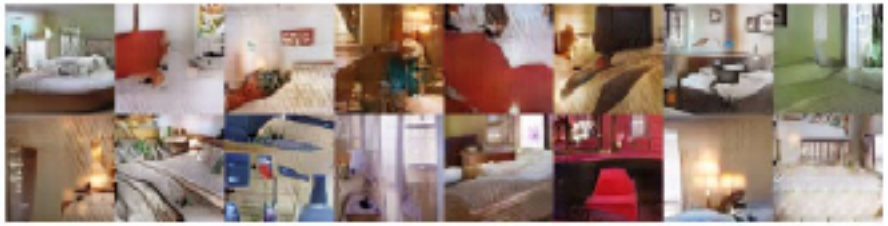


This is a conditional GAN, which like the LSGAN, adds some control over the class being generated.

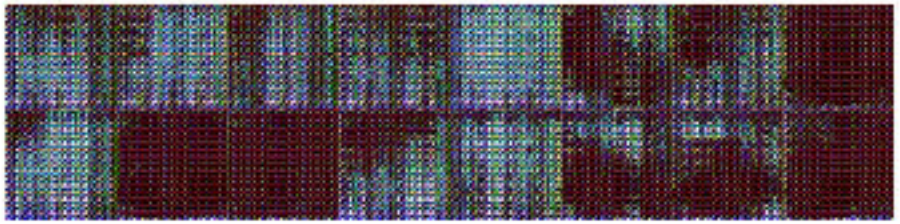
- In their empirical findings, no mode collapse problems
- And training longer resulted in good quality images (motivates that distributions overlap)
- Still some of the most competitive GAN results



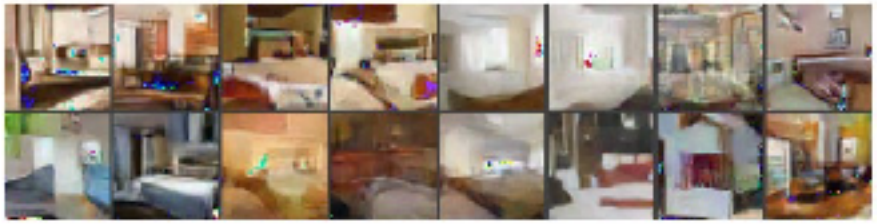
WGAN with DCGAN generator

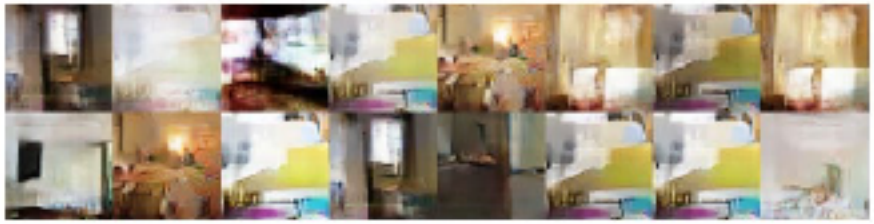


GAN with DCGAN generator



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 slow
- Clipping:

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
- But… its still really slow compared to others

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 has a norm less than or equal to one.

Maximize f
freeze generator weights,
incentivize critic gradient norm to be one

Choose large lambda
like lambda = 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 $\mathrm{U}[0,1]$ $\quad \hat{x}^{(i)} = \epsilon \cdot x_{real}^{(i)} + (1-\epsilon)\cdot g(z^{(i)})$

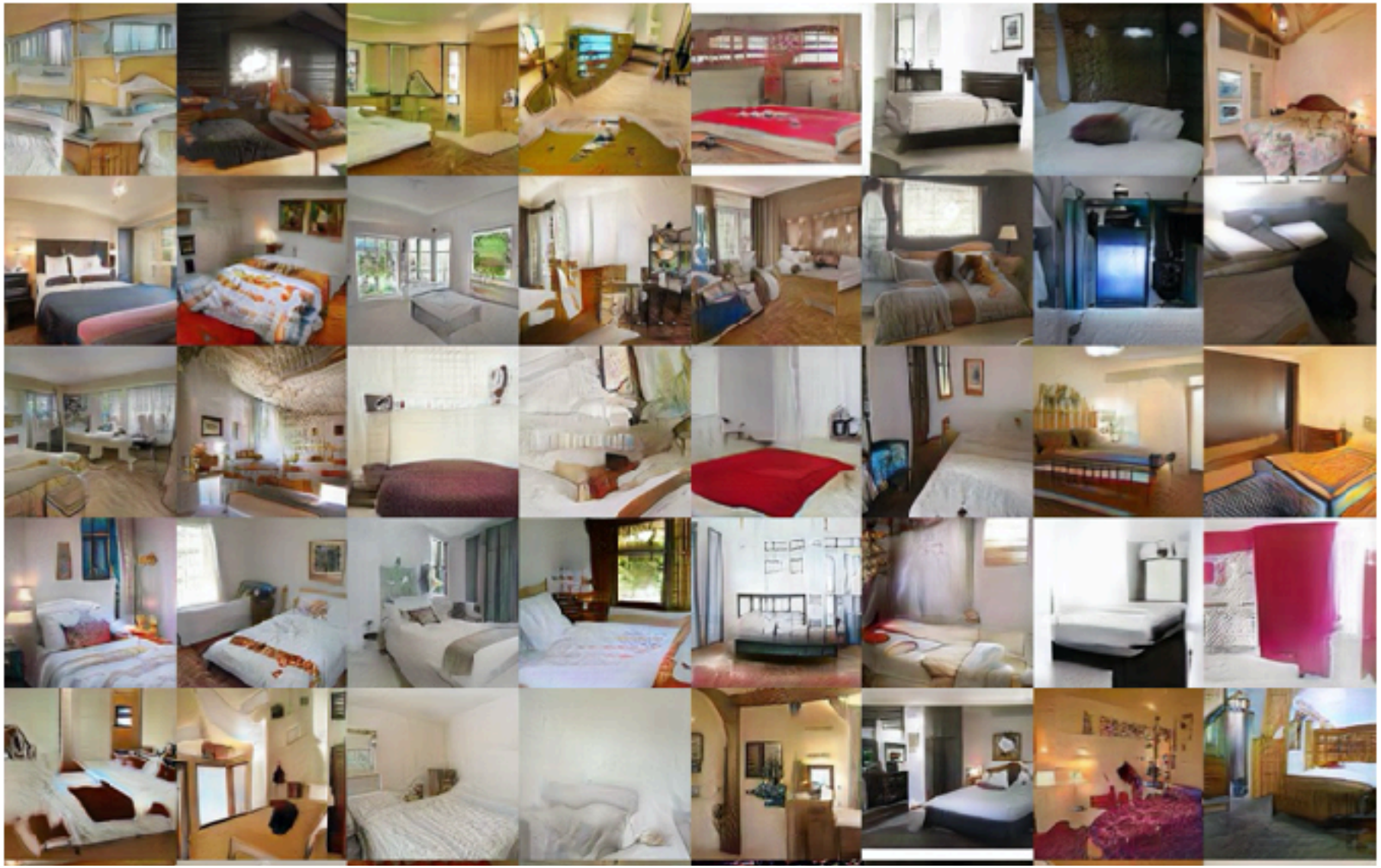
randomly mix together real and fake images

WGAN-GP Heuristics

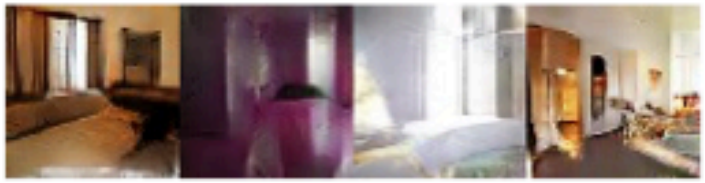
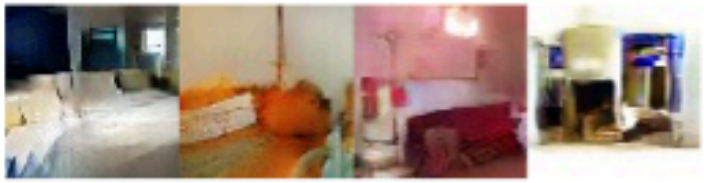
- Choose large penalty coefficient
- Don't use batch normalization (in critic)
- Enforce gradient norm to be close to one, rather than less than one

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.

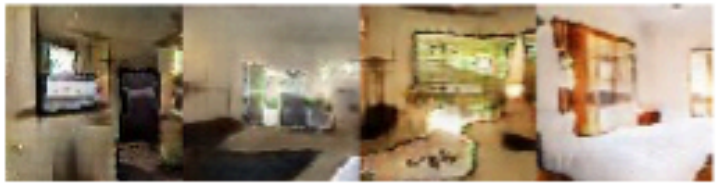
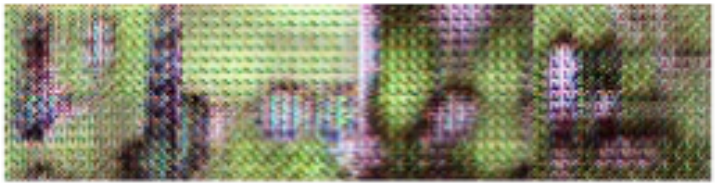
WGAN-GP Results



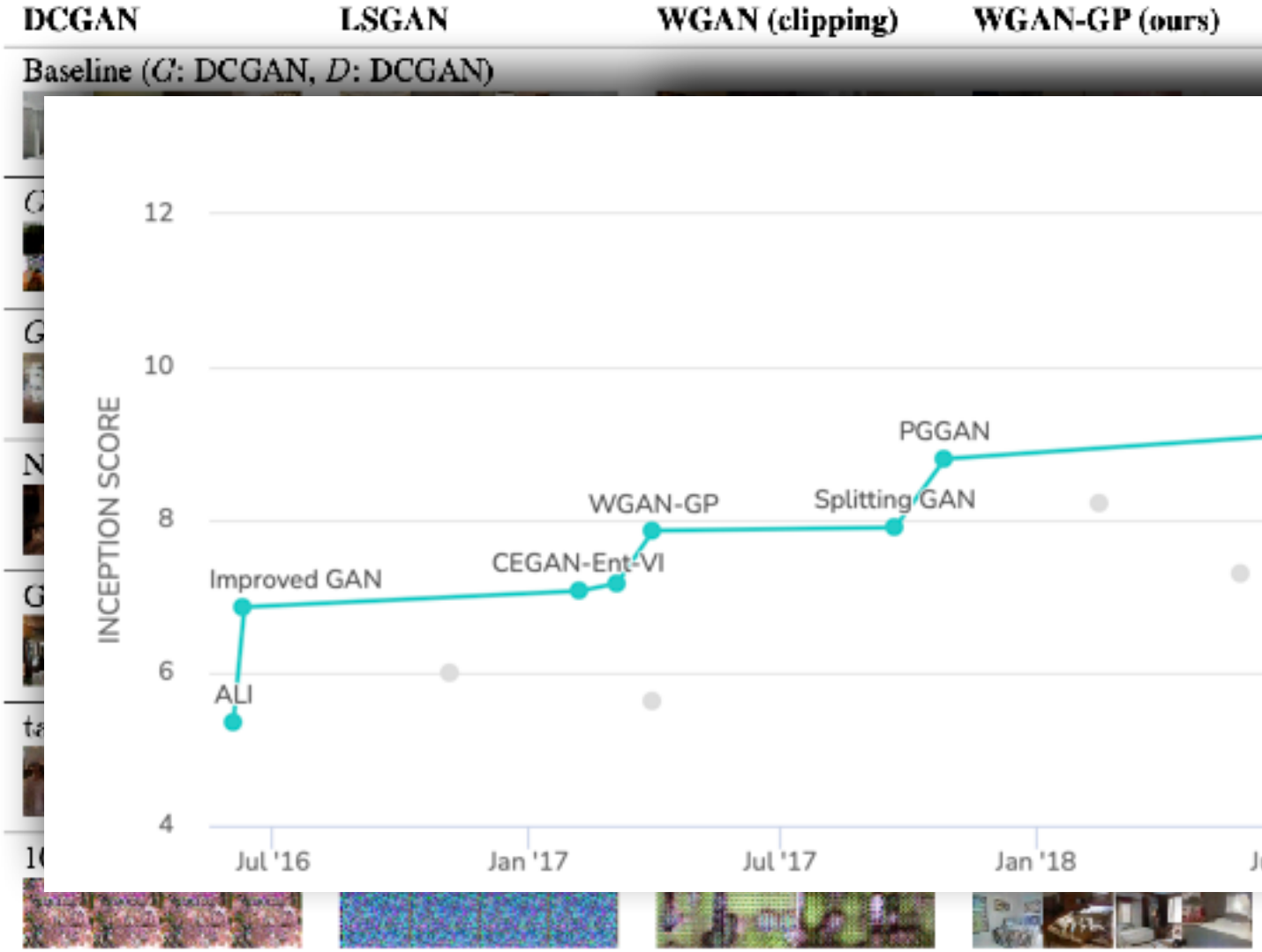
WGAN-GP Comparative Results

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

WGAN-GP Comparative Results

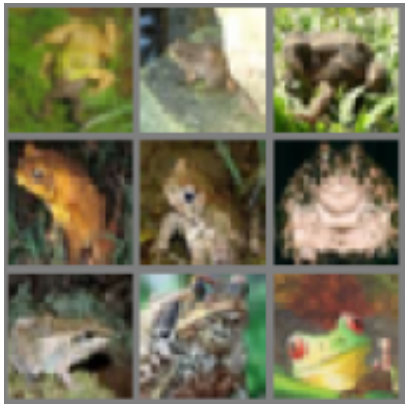
WGAN (clipping)	WGAN-GP (ours)
G/Crit: Tanh Non-linearities	
	
G/Crit: 101 Layer ResNet	
	

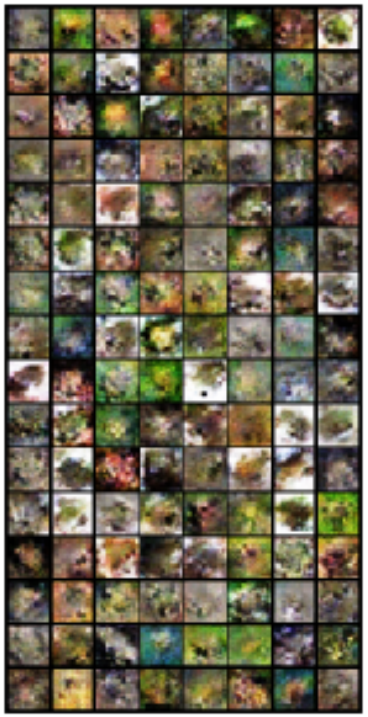
WGAN-GP Comparative Results



WGAN-GPs

Master Repository:
`07c GANsWithTorch.ipynb`





Other Examples

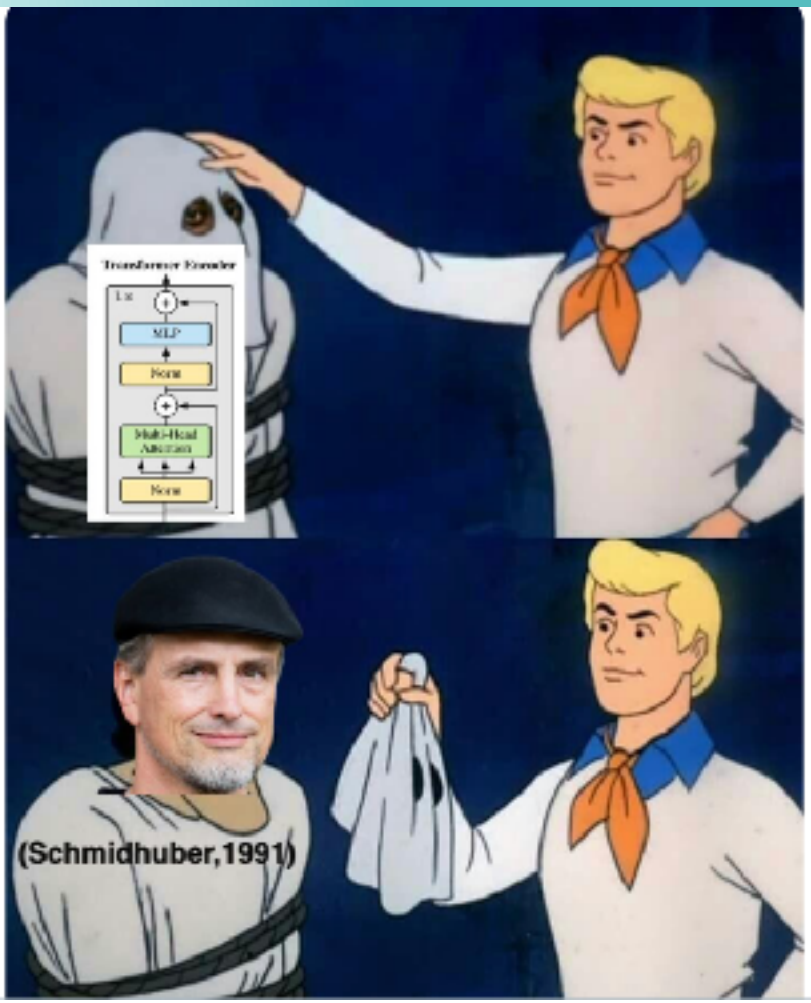
Keras Example: https://github.com/keras-team/keras-contrib/blob/master/examples/improved_wgan.py DCGAN

Demo by Keras Contrib Community

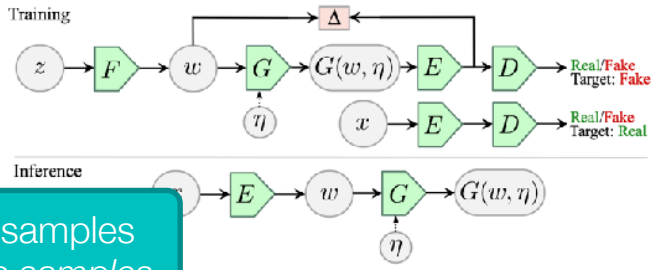
Other Example: https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/wgan_gp/wgan_gp.py MLP-GAN

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: St
4: x
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**

Based On Nash Equilibrium

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.

BigGAN

In a field with 1000's of competing papers, BigGAN is here to use the most meaningful Portions of each paper and put them into One BIG paper.

Andriy Burkov @burkov · 22h

People who try to learn machine learning (or another similar) science by themselves find it very hard to understand why those 1/N or 1/2 are used. It takes years before they realize that it doesn't serve any purpose other than aesthetically pleasing the scientist who wrote them.

ately solves the following optimization

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}); \quad J(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\boldsymbol{w})$$

function. In this work, we consid de

e sum of squared errors, $\mathcal{L}_i(\boldsymbol{w}) = \frac{1}{2} \|y$

n the average correct answer for a give

BigGAN Overview

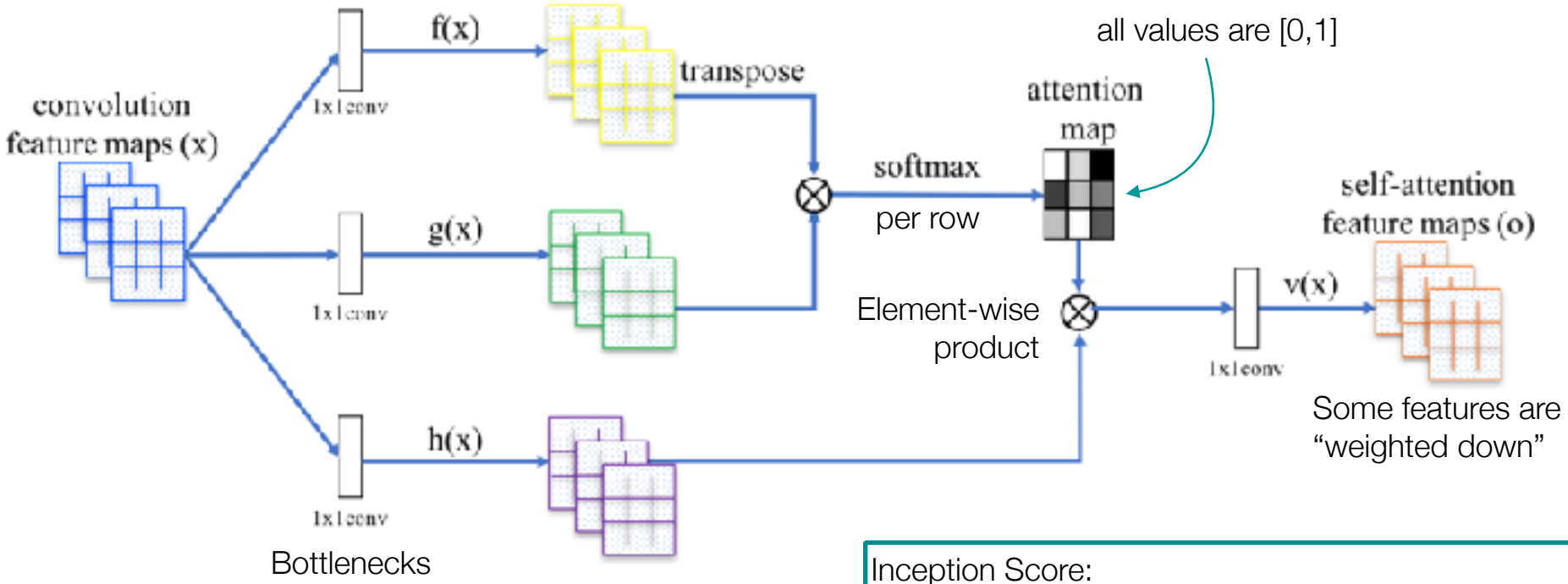
- This is an agglomeration of everything we already know about GANs from *2013-present day*
- Training is hard, so use heuristics
  - large batches, feature matching
  - use hinge loss (max margin)
- Use attention, conditional classes, spectral normalization, moving average of weights, orthogonal weight initialization, skip connections, orthogonal regularizers
- **Truncation trick:** sample a wide range during training, then truncate for generating results

- Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2018.
- Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019. 2019
- Self-Attention Generative Adversarial Networks, 2018.
- A Learned Representation For Artistic Style, 2016.
- Spectral Normalization for Generative Adversarial Networks, 2018.
- Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017.
- Exact Solutions To The Nonlinear Dynamics Of Learning In Deep Linear Neural Networks, 2013.
- Neural Photo Editing with Introspective Adversarial Networks, 2016.

https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/

BigGAN Part One: Self Attention

Model used in both generator and discriminator



Model	Inception Score
AC-GAN (Odena et al., 2017)	28.5
SNGAN-projection (Miyato & Koyama, 2018)	36.8
SAGAN	**52.52**

Inception Score:

$$\hat{p}(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|\mathbf{x}^{(i)}),$$

Expected class distribution through a trained CNN

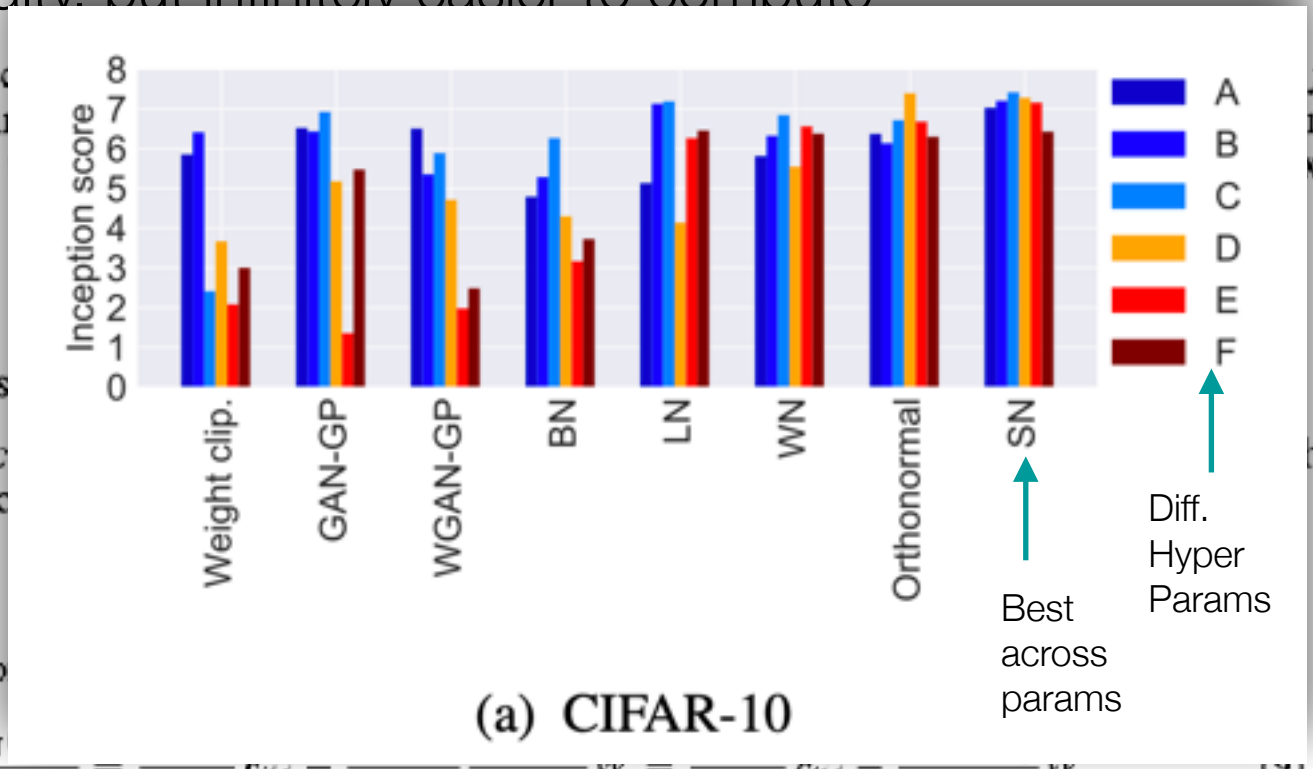
$$\text{IS}(G) \approx \exp\left(\frac{1}{N}\sum_{i=1}^{N} D_{KL}(p(y|\mathbf{x}^{(i)}) \parallel \hat{p}(y))\right).$$

average KL Div with generated images

- Zhang, Goodfellow, Metaxas, Odena. Self-Attention Generative Adversarial Networks, 2018.

BigGAN Part Two: Spectral Normalization

- Taking the spectral norm of each weight matrix satisfies the Lipschitz constraint $\sigma(W) \approx 1$ which functions like a Wasserstein penalty, but infinitely easier to compute



(a) CIFAR-10

Our spec … constrai … is equal … f by literally … norm $\|g\|_{\text{Lip}}$ … k norm of A)

(6)

which is …

Our *spec* … the Lipsc … hat it satisfies

(8)

The gradient[2] o …

$$\frac{\partial \bar{W}_{\text{SN}}}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \frac{W}{\sigma(W)} = \frac{E_{ij}}{\sigma(W)} - \frac{1}{\sigma(W)^2} \frac{\partial \sigma(W)}{\partial W_{ij}} W = \frac{E_{ij}}{\sigma(W)} - \frac{1}{\sigma(W)^2} W \quad (9)$$

$$= \frac{1}{\sigma(W)} \left(E_{ij} - [\boldsymbol{u}_1 \boldsymbol{v}_1^{\mathrm{T}}]_{ij} \bar{W}_{\text{SN}} \right), \quad (10)$$

And we can back propagate through the calculation!

https://arxiv.org/pdf/1802.05957.pdf Spectral Normalization for Generative Adversarial Networks, 2018.

BigGAN Part Three: Orthogonality

- Initialize with orthogonal weights per layer
- $W \cdot W^T = I$
- Add orthogonal regularization to loss:

$$\mathscr{L}_{orthogonal} = \alpha_{orth} \sum \| W \cdot W^T - \mathbf{I} \|$$

Usually too restrictive…

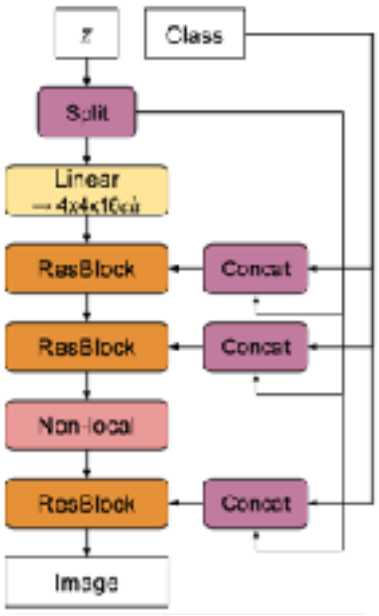
applied across channels of filters

$$\mathscr{L}_{orthogonal} = \alpha_{orth} \sum \| W \cdot W^T \odot (\mathbf{1} - \mathbf{I}) \|$$

penalize non zero diagonals

Brock, Lim, Ritchie, Wetson, Neural Photo Editing with Introspective Adversarial Networks. 2017 https://arxiv.org/abs/1609.07093

BigGAN Part Four: Conditional Class Info



(a) Generator

RGB image $x \in \mathbb{R}^{128\times128\times3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock down 1024
ResBlock 1024
ReLU
Global sum pooling
dense $\rightarrow$ 1

(b) Discriminator for unconditional GANs.

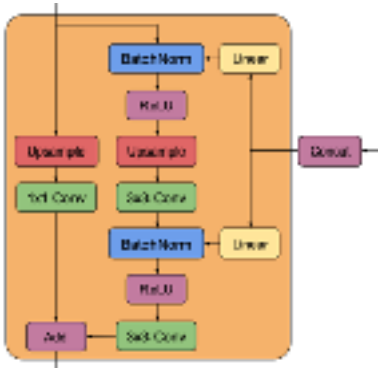
RGB image $x \in \mathbb{R}^{128\times128\times3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
Concat(Embed(y), h)
ResBlock down 512
ResBlock down 1024
ResBlock 1024
ReLU
Global sum pooling
dense $\rightarrow$ 1

(c) Discriminator for conditional GANs. For computational ease, we embedded the integer label $y \in \{0, \ldots, 1000\}$ into 128 dimension before concatenating the vector to the output of the intermediate layer.

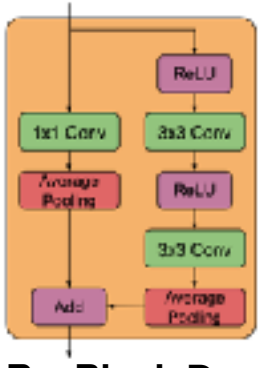
Categorical, One Hot Class

Embedding

Can help to learn **class specific** properties in generator and discriminator



ResBlock Up



ResBlock Down

https://arxiv.org/pdf/1802.05957.pdf Spectral Normalization for Generative Adversarial Networks, 2018.

BigGAN Part Five: Miscellaneous

- Update discriminator twice as often as generator
- Large batch size: 1024 or 2048
- Use skip connections in model architecture, starting from z
- Use LOTS of filters: 150% more filters than related work
- **Truncation trick**: during training, use wider sampling than during evaluation
- Use hinge loss: $\frac{1}{m}\sum_{i=1}^{m} f(x_{real}^{(i)}) \cdot f(g(z^{(i)}))$
- Use moving average of weights: $W_k = \sum_i \gamma^i \underbrace{W_{k-i}}_{\text{past}}$

BigGAN Results




Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	IS
256	64	81.5	SA-GAN Baseline			52.52
512	64	81.5	✗	✗	✗	58.77(±1.18)
1024	64	81.5	✗	✗	✗	63.03(±1.42)
2048	64	81.5	✗	✗	✗	76.85(±3.83)
2048	96	173.5	✗	✗	✗	92.98(±4.27)
2048	96	160.6	✓	✗	✗	94.94(±1.32)
2048	96	158.3	✓	✓	✗	98.76(±2.84)
2048	96	158.3	✓	✓	✓	99.31(±2.10)
2048	64	71.3	✓	✓	✓	86.90(±0.61)

BigGAN Results

256 x 256

BigGAN Results

512 x 512

BigGAN Results: Linear Interpolation

The GAN Takeaways

- Approximation reigns supremum, err maximum
  - But do not be too skeptical
  - … be skeptical of your skepticism
- Intractable mathematics encourages researchers to find their method works via poor practices
- Its unclear what loss actually works and how to incentivize different behaviors, but we are getting better

Lecture Notes for

Neural Networks and Machine Learning

Wasserstein GANs and BigGAN

Next Time:

Beyond Generation

Reading: Chollet CH8