

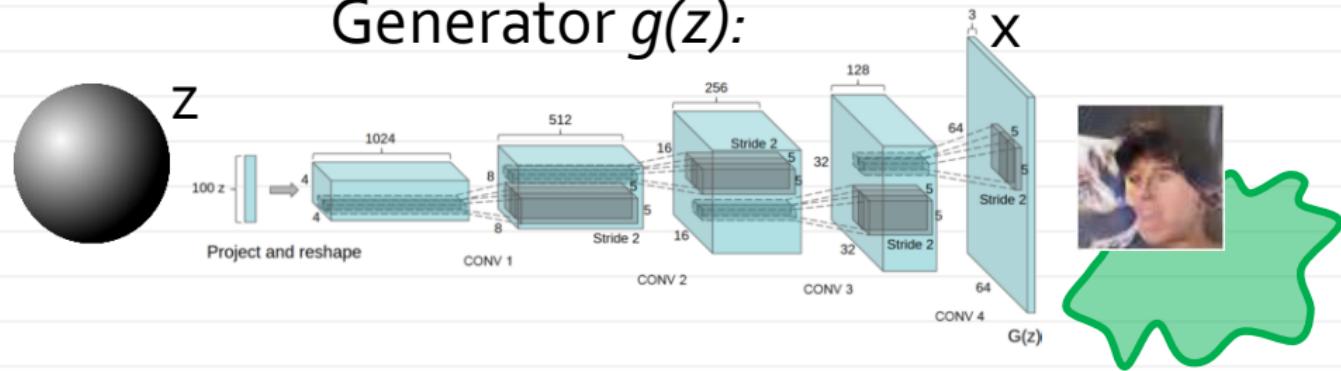
Lecture 8: Adversarial learning

Recap on Generative networks

- ConvNets are natural tools for image synthesis
- Both conditional synthesis (e.g. superresolution) and unconditional sampling (latent models) is possible
- Training loss matters, pixelwise loss often leads to blurry results
- Pretrained ConvNets can be used to extract good statistics/loss-functions
- Can we get even better statistics/losses?

Generative Adversarial Networks

Generator $g(z)$:



Discriminator $d(x)$:



[Goodfellow et al. 14]

GAN game: zero-sum variant

$$d : \mathcal{X} \rightarrow [0; 1] \quad g : \mathcal{Z} \rightarrow \mathcal{X}$$

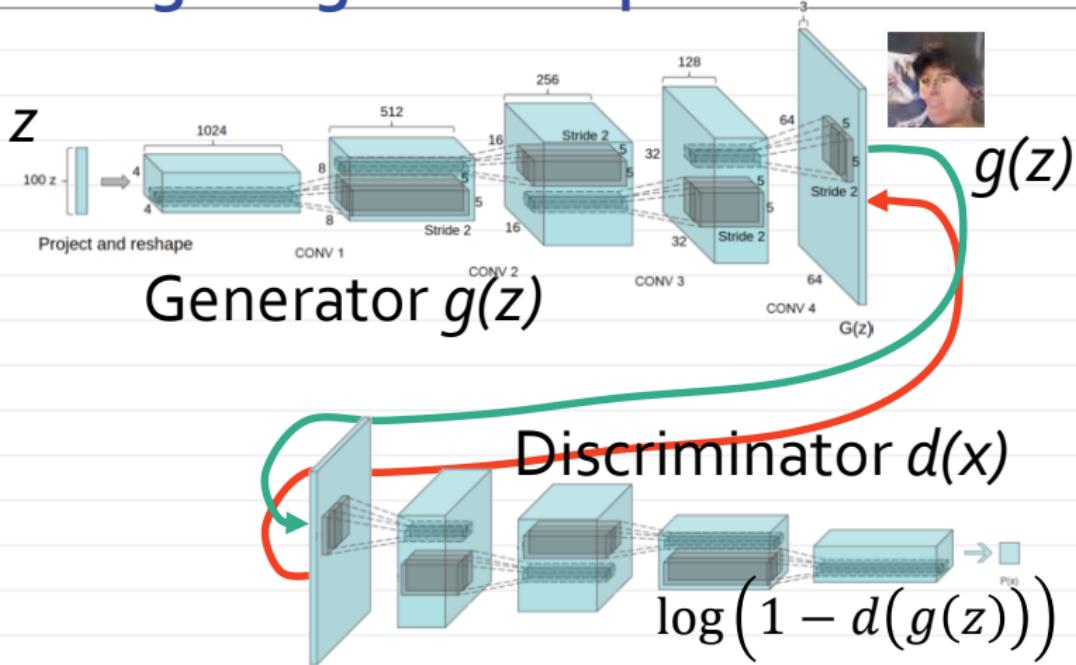
$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

- Discriminator tries to increase the value of $V(d, g)$ (*learns to tell p_{data} from $p(g(z))$*)
- Generator tries to decrease the value of $V(d, g)$ (*fool the discriminator*)

[Goodfellow et al. 14]

Updating the generator parameters



$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

Optimal discriminator

Let us derive optimal $d(x)$ given $p_{\text{data}}(x)$, $p_{\text{gen}}(x)$

$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

$$\begin{aligned}\frac{\delta V(d, g)}{\delta d(x)} &= \frac{\delta [p_{\text{data}}(x) \log d(x) + p_{\text{gen}}(x) \log(1 - d(x))]}{\delta d(x)} \\ &= \frac{p_{\text{data}}(x)}{d(x)} - \frac{p_{\text{gen}}(x)}{1 - d(x)} = 0\end{aligned}$$

$$(1 - d(x))p_{\text{data}}(x) - d(x)p_{\text{gen}}(x) = 0$$

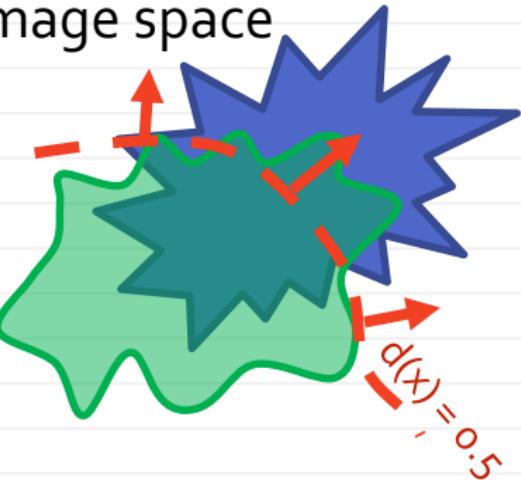
$$d(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}$$

Updating the discriminator parameters

“Fake”



Image space



“Real”

$$d(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}$$

Optimal discriminator works as *ratio estimator*

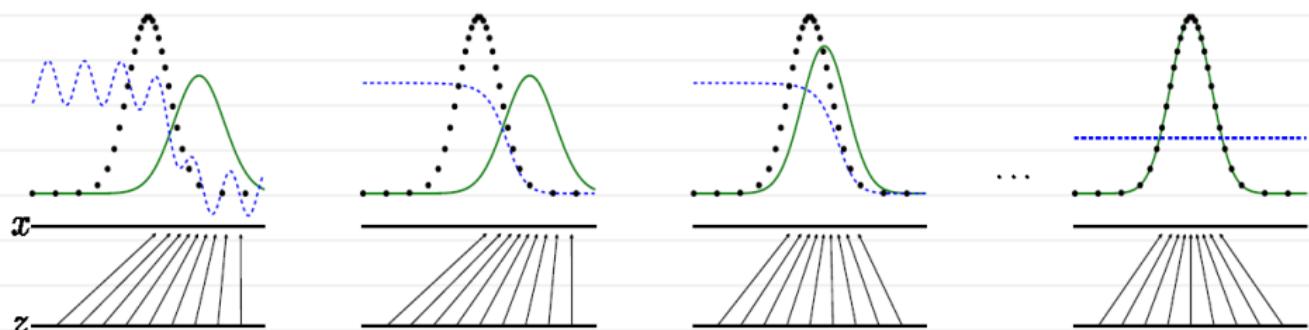
Adversarial learning turns high-D distribution alignment into classification

Adversarial learning

Looking for a *saddle point*:

$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$



[Goodfellow et al. 2014]

GAN game: practical variant

$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

Problem: generator gets *vanishing gradient* when the discriminator is much smarter.

Therefore, non-zero sum formulation is popular:

$$\max_d \mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

$$\max_g \mathbb{E}_{z \sim Z} [\log d(g(z))]$$

[Goodfellow et al. 14]

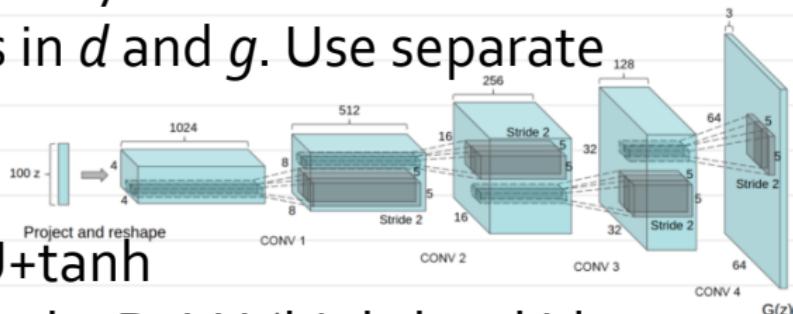
Tuning GANs

100 years ago: "The most important art is cinema"

Now: "The most important art is tuning GANs"

DCGAN ("deep conv. GAN") [Radford et al. 2015]:

- No max-pooling, only strides
- Use batch norms in d and g . Use separate batches.
- Use ADAM
- Generator: ReLU+tanh
- Discriminator: Leaky ReLU (high-level idea:
"dense gradients")



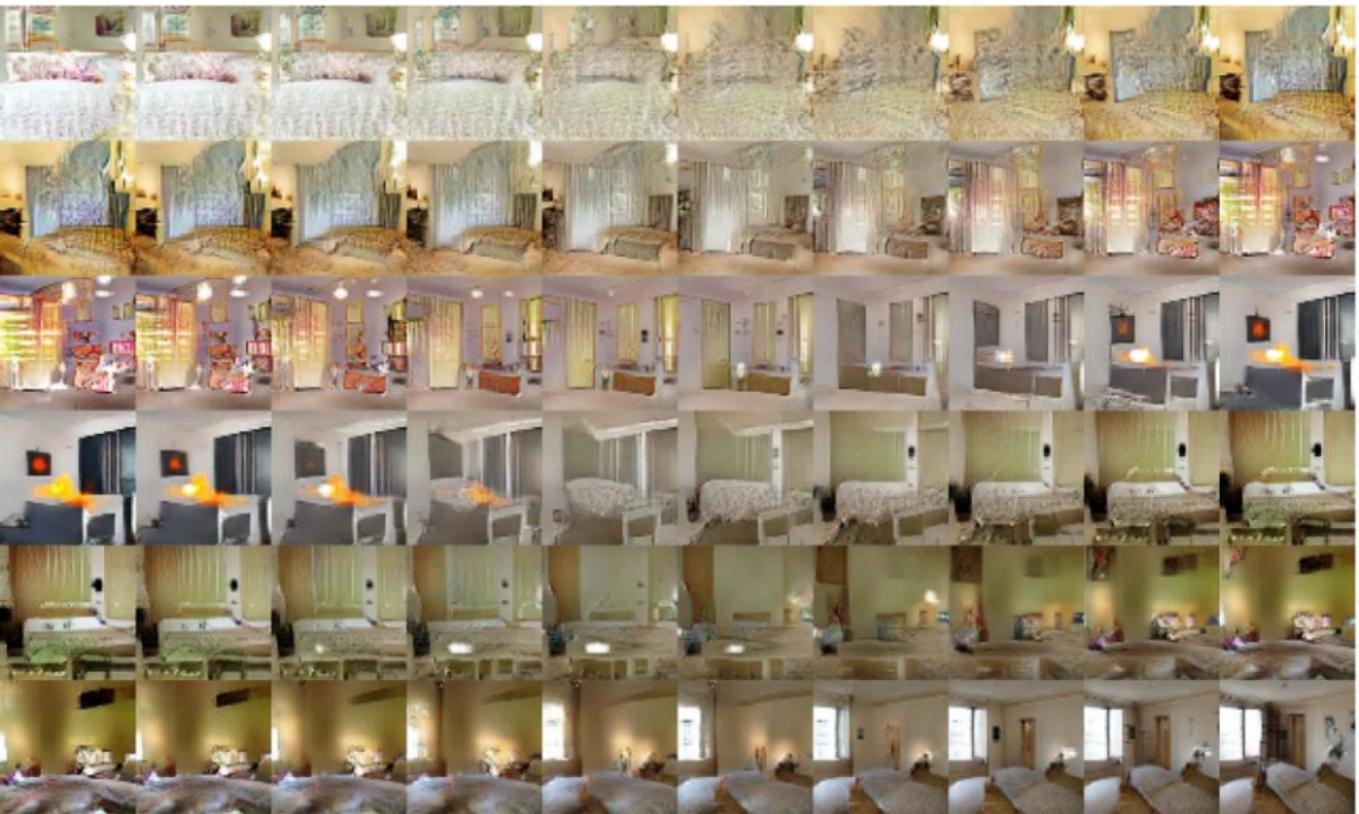
<https://github.com/soumith/ganhacks>

DCGAN results



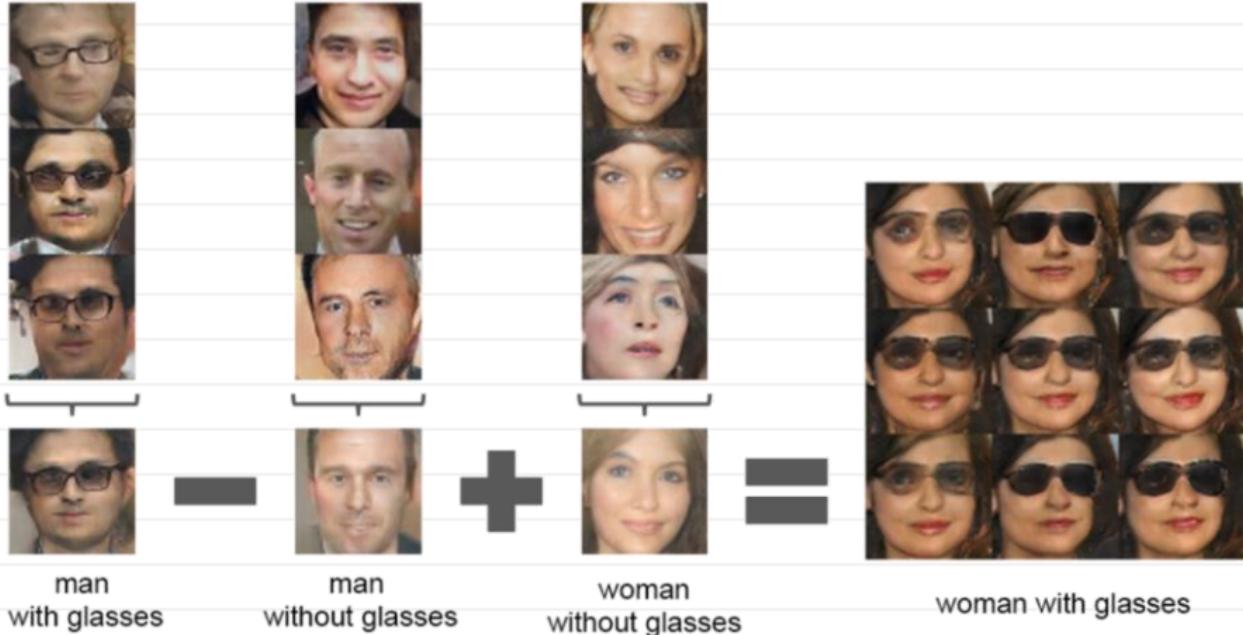
[Radford et al. 2015]

DCGAN



[Radford et al. 2015]

DCGAN



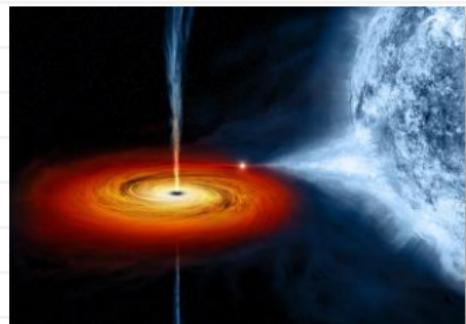
[Radford et al. 2015]

The problem of perfect generator

$$\min_g \mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

$$\max_g \mathbb{E}_{z \sim Z} [\log d(g(z))]$$

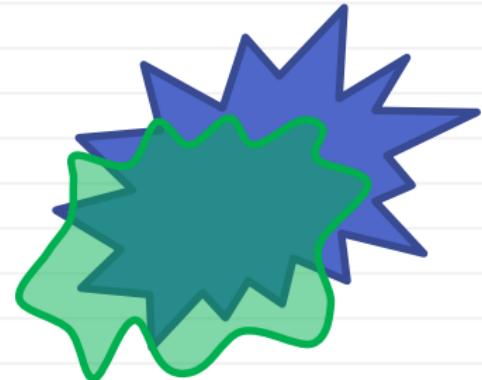
What is the optimal generator for a fixed discriminator?



Answer: $g(z) = \arg \max_x d(x)$

Main source of instability ("mode collapse").

Mode collapse in practice



DCGAN samples
[Radford et al. 2015]

Some hard modes missed (e.g. people with glasses)

Measuring diversity: Inception score

[Salimans et al. 2016]:

Inception v3-network prediction

$$IS(g) = \exp\left(\mathbb{E}_{z \sim Z} [\text{KL}(p(y|g(z)) \| p(y))]\right)$$

- Looks at prediction of 1000 ILSVRC classes
- Used to score generative models for all kinds of datasets
- Has lots of problems [Baratt & Sharma 2018]

No single good measure exists ☹

Further problems

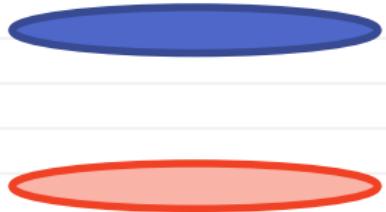
$$\min_g \max_d V(d, g) =$$

$$\mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

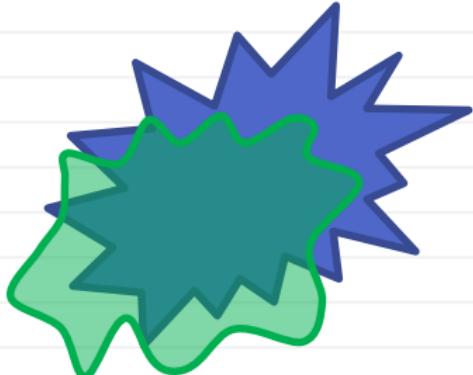


$$d(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} \rightarrow \min_g \text{JS}(X \| g(Z))$$

- Discriminator is defined in a “point-wise” fashion
- Equally bad for distributions with measure zero intersection



Alignment through EMD



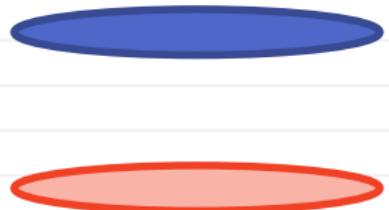
$$\min_g W(X \| g(Z))$$

Earth mover distance (aka Wasserstein-1 distance):



$$W(X \| Y) = \inf_{\gamma \in \Pi(X, Y)} \mathbb{E}_{(x, y) \sim \gamma} [\|x - y\|]$$

All random variables over
with marginal distributions X and Y



[Arjovsky et al 2017]

Wasserstein GAN

$$\min_g W(X \| g(Z))$$

$$W(X \| Y) = \inf_{\gamma \in \Pi(X, Y)} \mathbb{E}_{(x, y) \sim \gamma} [\|x - y\|]$$

Kantorovich-Rubinstein duality:

$$W(X \| Y) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X} [f(x)] - \mathbb{E}_{x \sim Y} [f(x)])$$

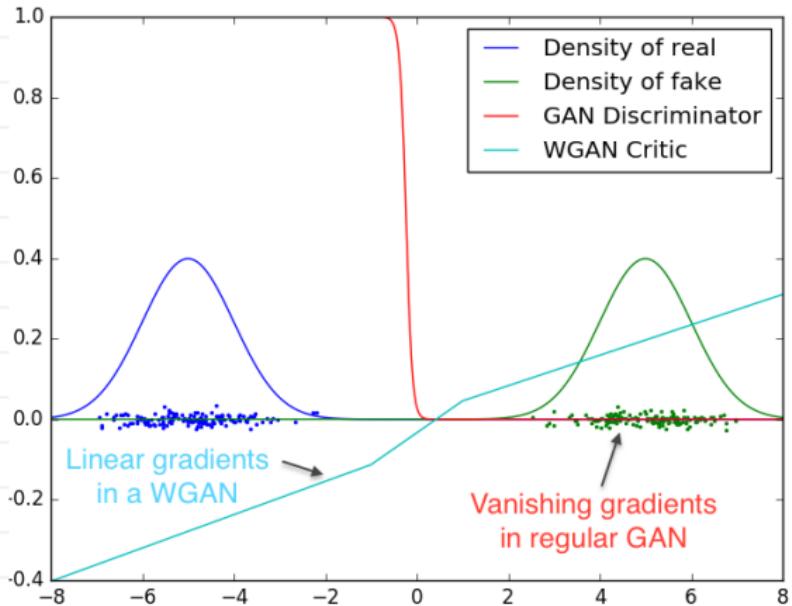
Wasserstein GAN [Arjovsky et al 2017]:

$$\min_g \max_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X} [f(x)] - \mathbb{E}_{z \sim Z} [f(g(z))])$$

(compare with the original GAN):

$$\min_g \max_d \mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

WGAN vs GAN



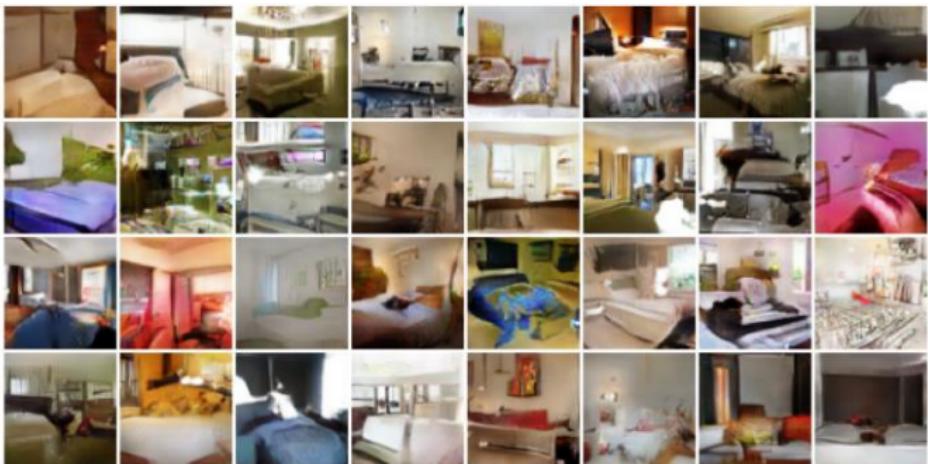
$$\max_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X} [f(x)] - \mathbb{E}_{z \sim Z} [f(g(z))])$$

$$\max_d \mathbb{E}_{x \sim X} [\log d(x)] + \mathbb{E}_{z \sim Z} [\log(1 - d(g(z)))]$$

Enforcing Lipschitz constant

$$\min_g \max_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X}[f(x)] - \mathbb{E}_{z \sim Z}[f(g(z))])$$

[Arjovsky et al. 2017]: implements $f(x)$ as a deep network with the weights bounded to $[-0.01; 0.01]$

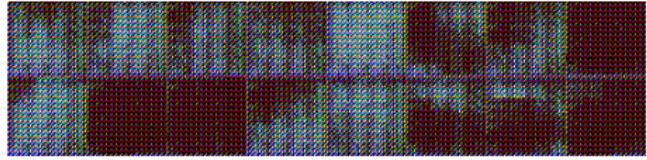


WGAN vs GAN

DCGAN generator:



“Bad” generator (no batchnorms, not enough filters):



[Arjovsky et al. 2017]

Enforcing Lipschitz constant: WGAN-GP

$$\min_g \max_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim X}[f(x)] - \mathbb{E}_{z \sim Z}[f(g(z))])$$

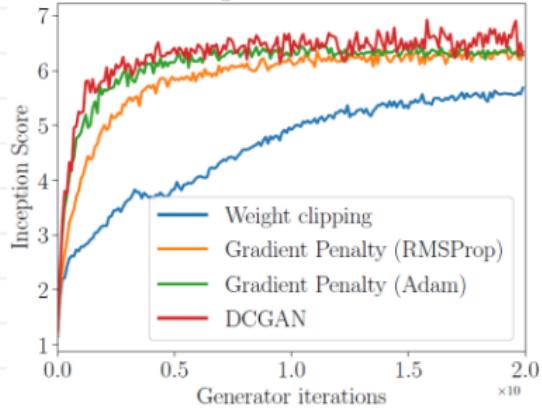
[Gulrajani et al. 2017]: do not clip weights, instead add the term to the objective:

$$+ \lambda \mathbb{E}_{x \sim X^*} [(\|\nabla_x f(x)\|_2 - 1)^2]$$

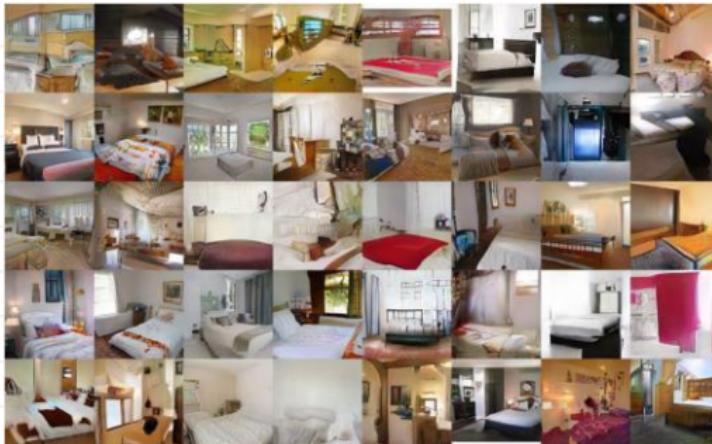
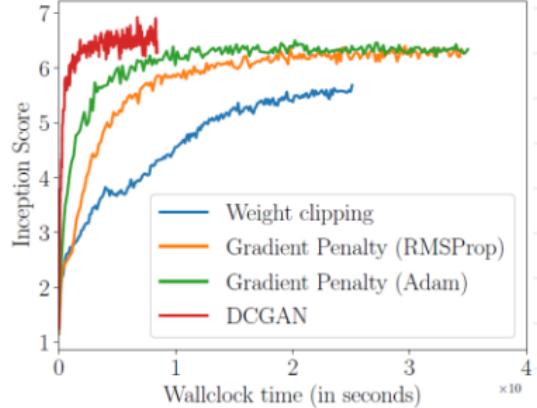
- X^* is an interpolation between X and $g(Z)$.
- Need functionality that implements gradient computation as a feedforward network (PyTorch has it)

WGAN-GP

Convergence on CIFAR-10



Convergence on CIFAR-10



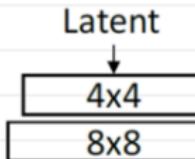
Progressive growing of GANs

G



Latent

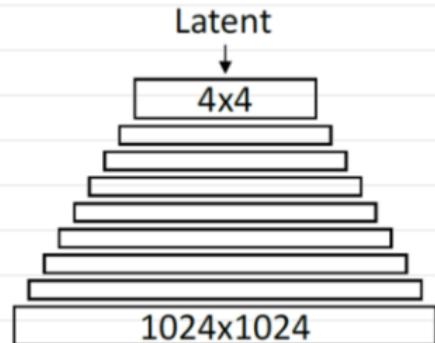
4×4



4×4

8×8

...



1024×1024



Reals

1024×1024

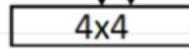
D



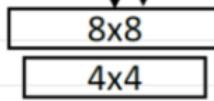
Reals



Reals



4×4



8×8

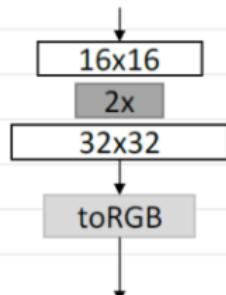
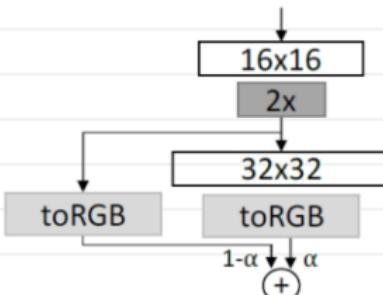
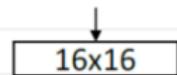
4×4

Training progresses

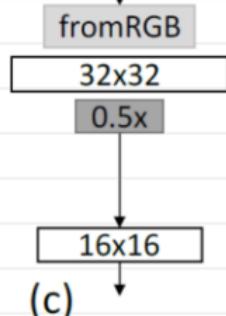
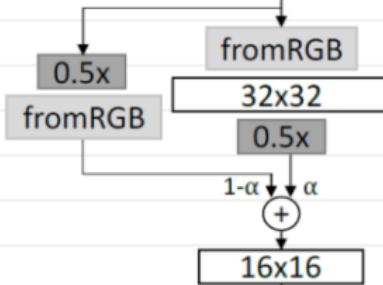
[Karras et al. 2017]

Progressive growing of GANs

G



D



(a)

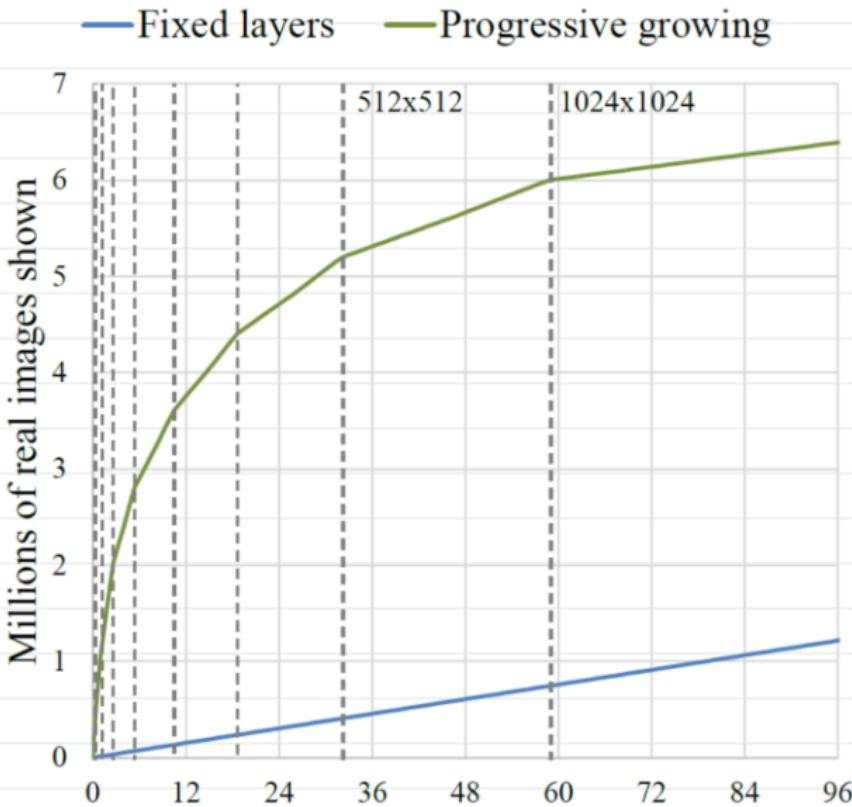
(b)

(c)

- Uses WGAN-GP variant

[Karras et al. 2017]

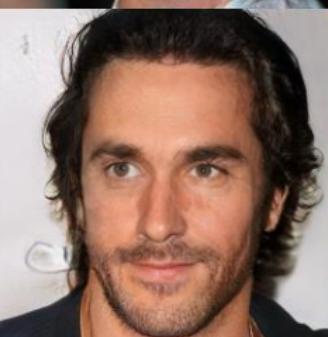
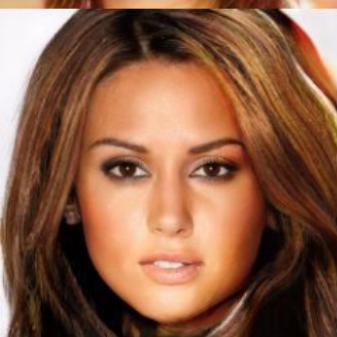
Progressive growing of GANs



[Karras et al. 2017]

Progressive growing of GANs

Amazing samples at 1024 x 1024:



[Karras et al. 2017]

Are GANs good latent models?

Real



Sample



$$g(\arg \min_z \|g(z) - x\|^2)$$

[thanks to
ShahRukh Athar]

Are GANs good latent models?

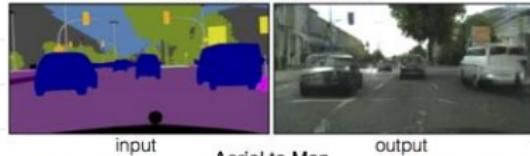


[Zhu et al. 2016]

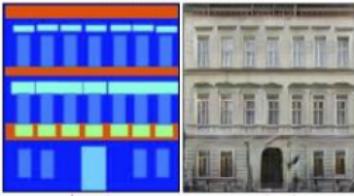
Pix2pix: a conditional GAN



Labels to Street Scene



Labels to Facade



BW to Color



Aerial to Map



Day to Night

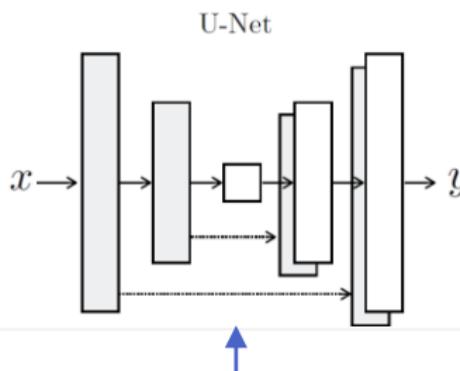


Edges to Photo

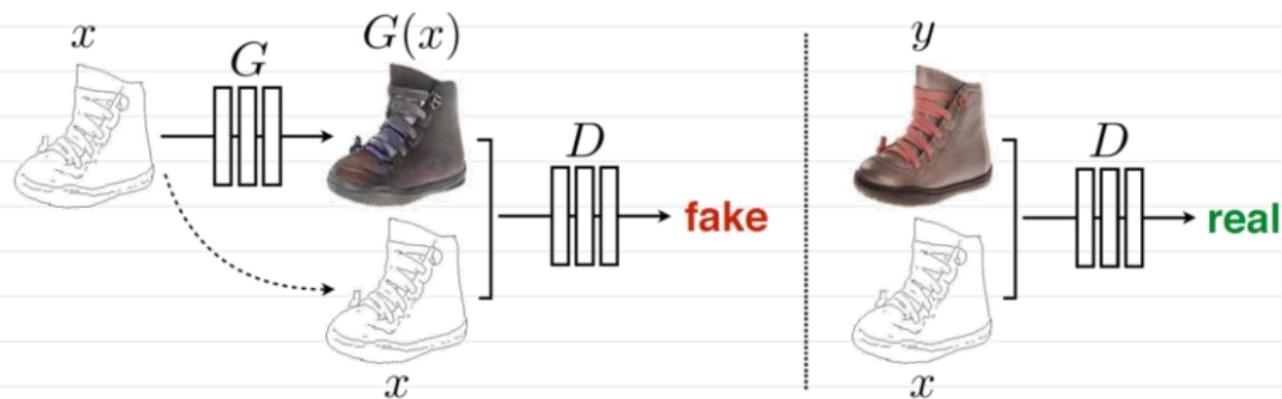


[Isola et al. 2017]

Pix2pix: a conditional GAN



- Use a combination of L1-loss and adversarial loss
- Adversarial loss is computed for patches ("PatchGAN")



[Isola et al. 2017]

GAN still fails at multi-modal tasks

Impressive generalizations:

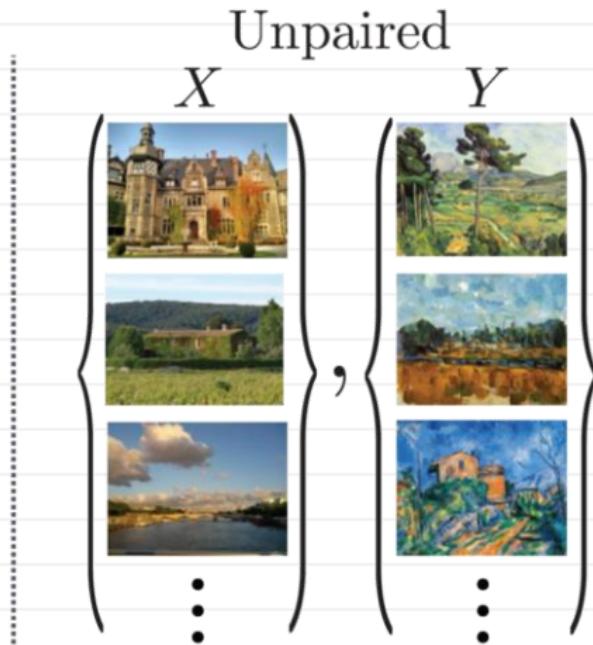


Colorization examples
(still model collapse):



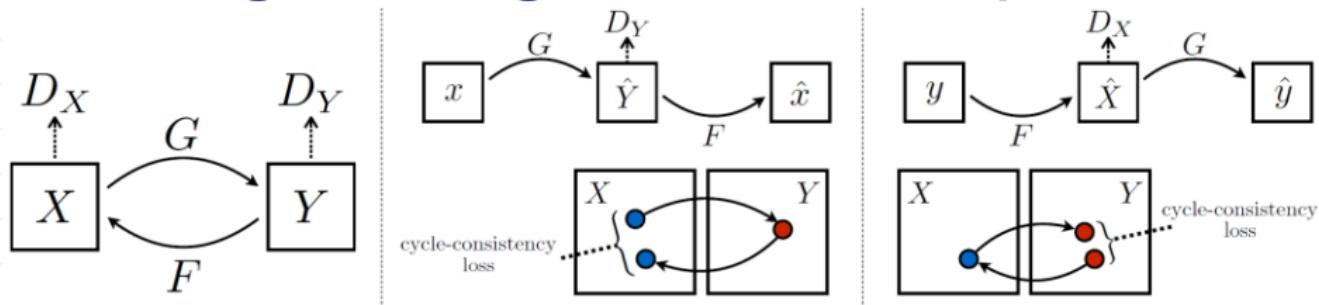
[Isola et al. 2017]

Unaligned image translation: CycleGAN



[Zhu et al. ICCV 2017]

Unaligned image translation: CycleGAN



$$L_{adv} = \mathbb{E}_{x \sim X} \left[\sum_p -\log D_y(G(x)[p]) \right]$$

$$L_{cycle} = \mathbb{E}_{x \sim X} [\|F(G(x)) - x\|_1]$$

$$L_{identity} = \mathbb{E}_{y \sim Y} [\|G(y) - y\|^2]$$

[Zhu et al. ICCV 2017]

Unaligned image translation: CycleGAN

Input x



Generated image $G(x)$



Reconstruction $F(G(x))$



[Zhu et al. ICCV 2017]

Identity loss: useful component

$$L_{\text{identity}} = \mathbb{E}_{y \sim Y} [\|G(y) - y\|^2]$$

Input



CycleGAN



CycleGAN+ L_{identity}



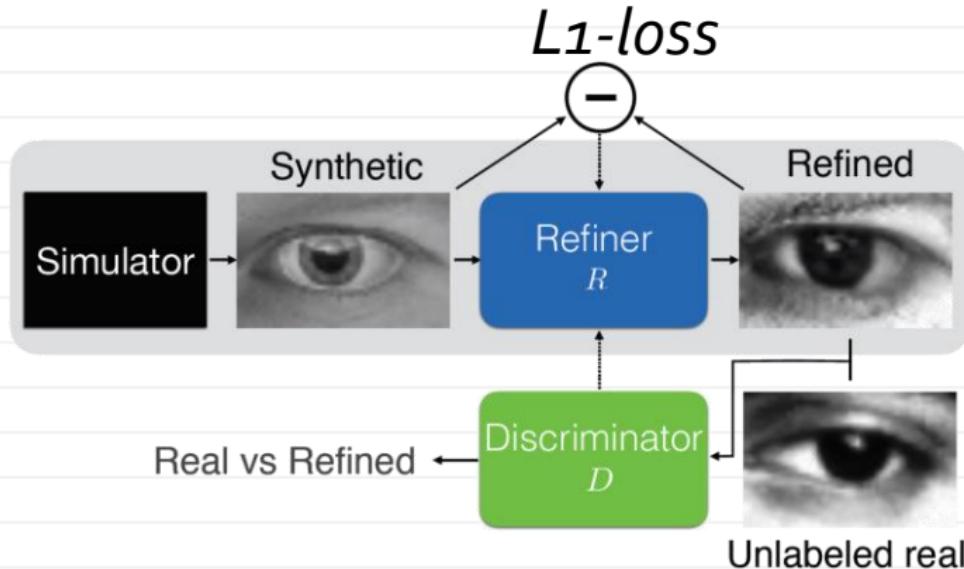
[Zhu et al. ICCV 2017]

Superresolution with GANs



[Ledig et al. 2017]

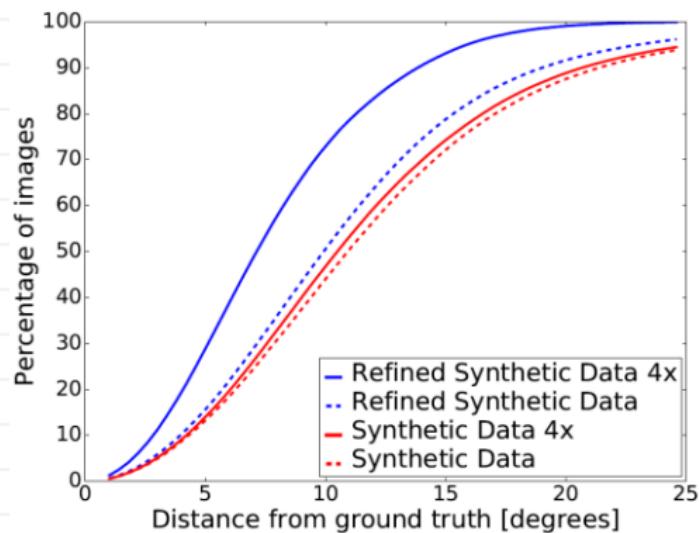
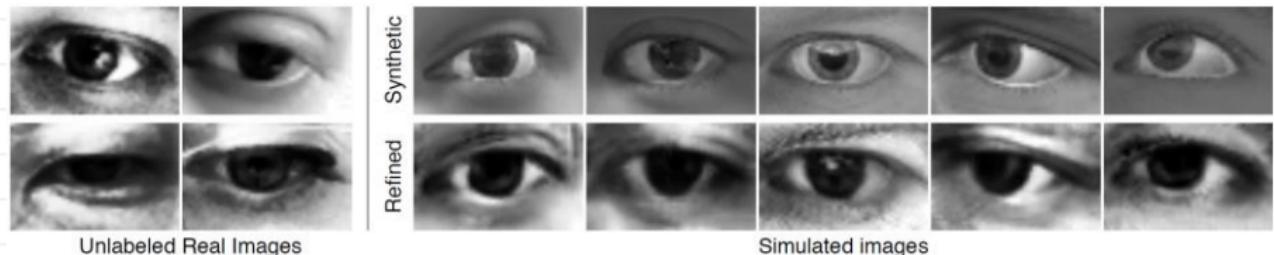
GAN for generating synthetic training data



- Using local (PatchGAN-type) discriminators
- Using history-based regularization of discriminators

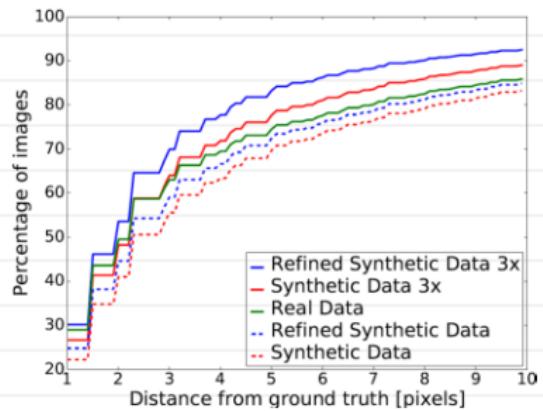
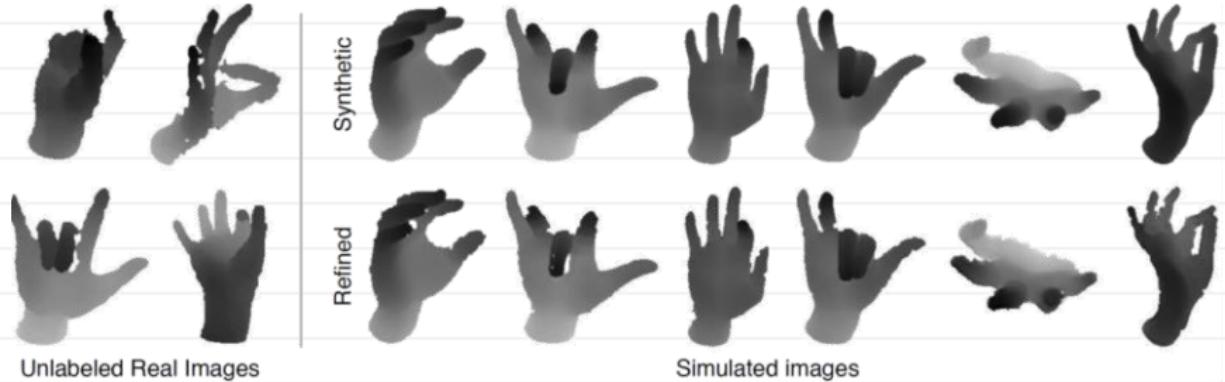
[Srivastava et al. CVPR17]

GAN for generating synthetic training data



[Srivastava et al. CVPR17]

GAN for generating synthetic training data

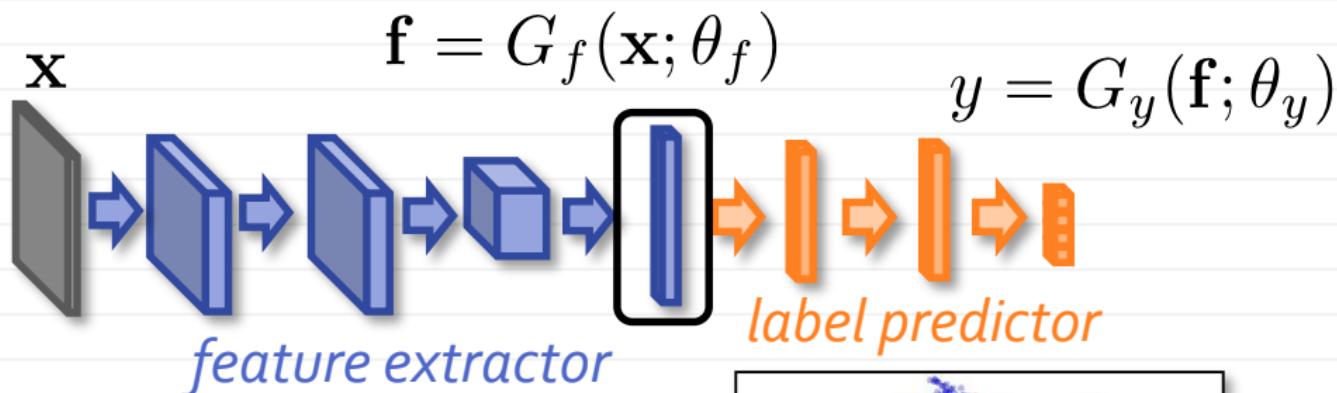


[Srivastava et al. CVPR17]

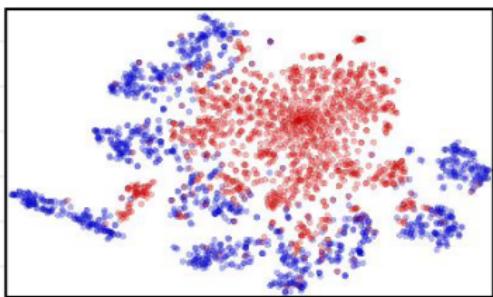
More examples: (semi-)synthetic to real



Domain shift inside a deep architecture



When trained on source only, feature distributions do not match:



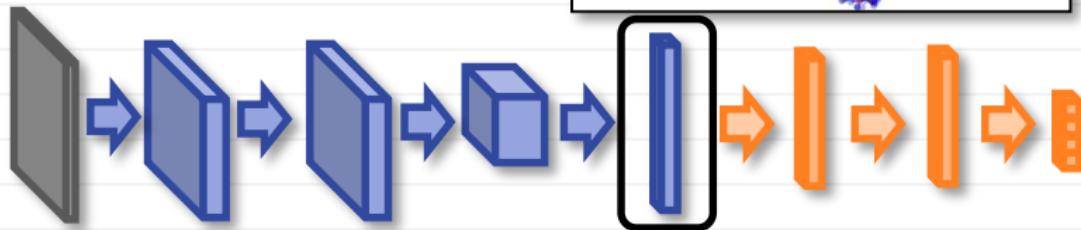
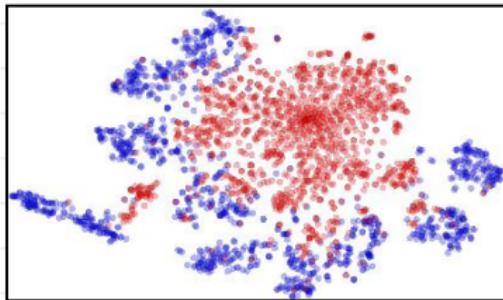
$$S(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \mid \mathbf{x} \sim S(\mathbf{x})\}$$

$$T(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \mid \mathbf{x} \sim T(\mathbf{x})\}$$

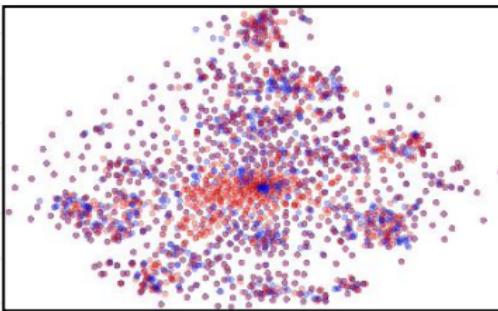
[Ganin et al. 2015]

Idea 1: domain-invariant features wanted

Feature distribution without adaptation:

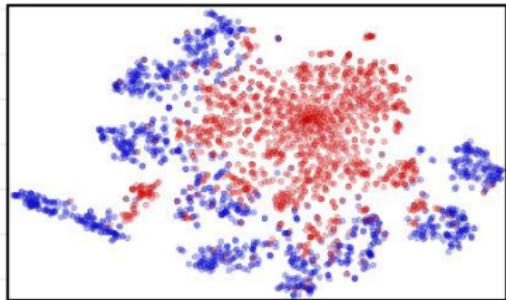
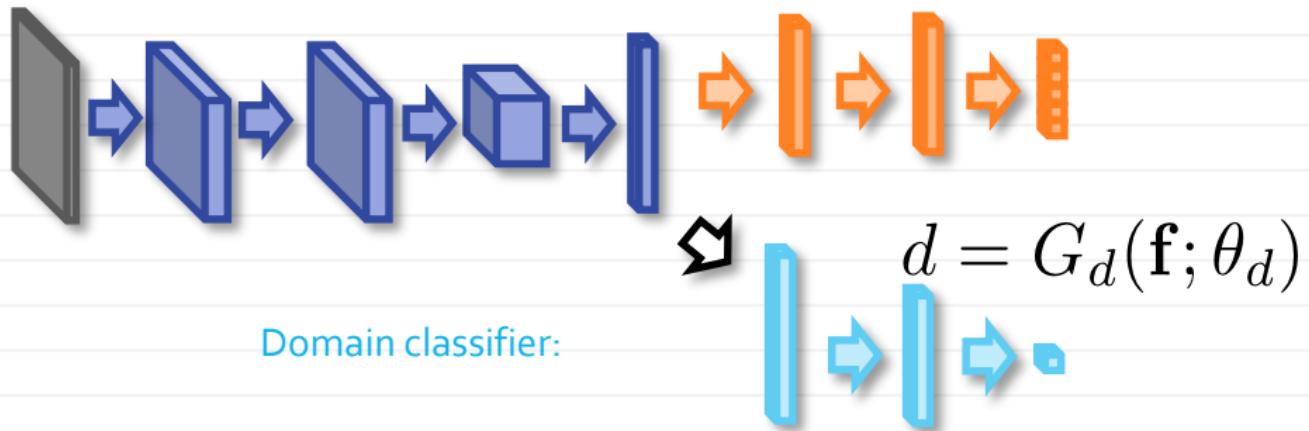


Our goal (after adaptation):

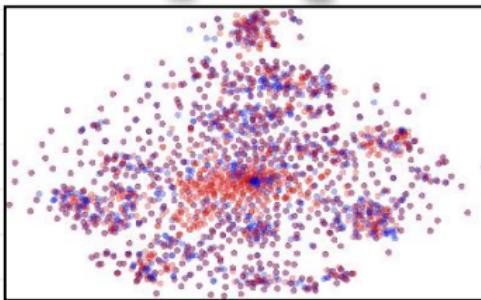


[Ganin et al. 2015]

Idea 2: measuring domain shift



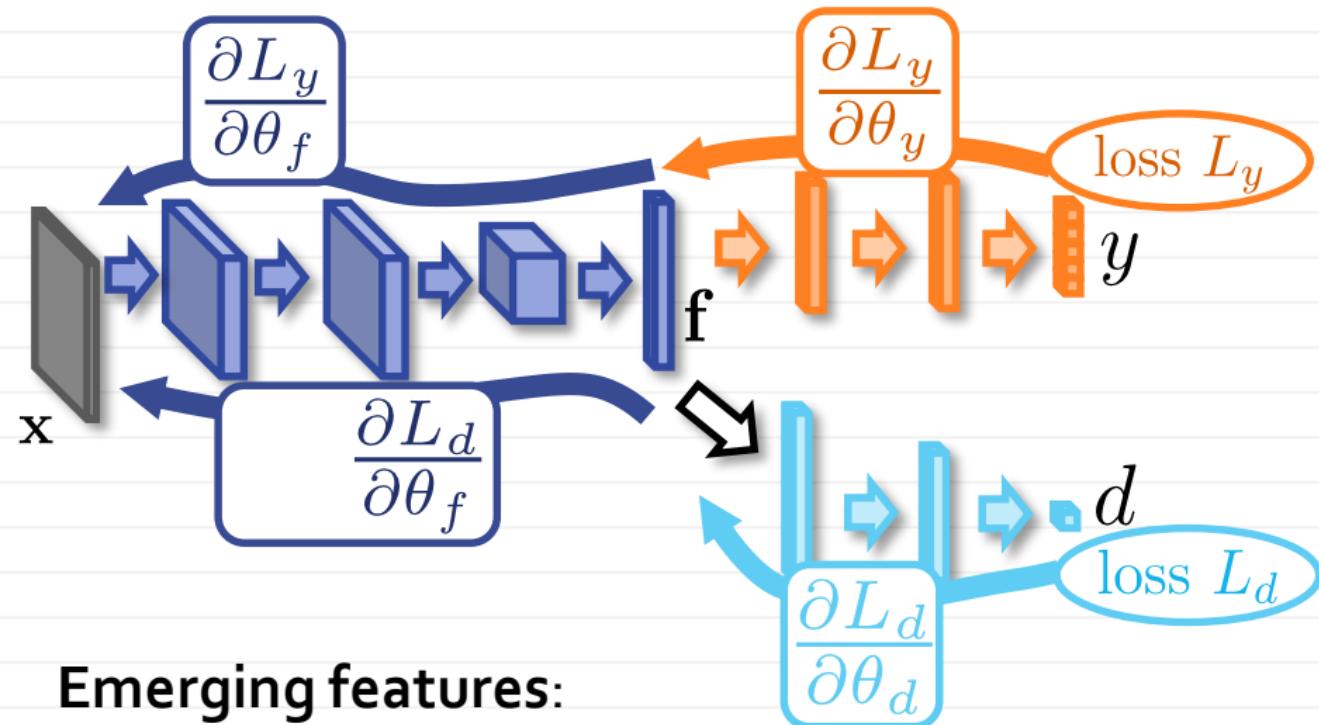
Domain loss low



Domain loss high

[Ganin et al. 2015]

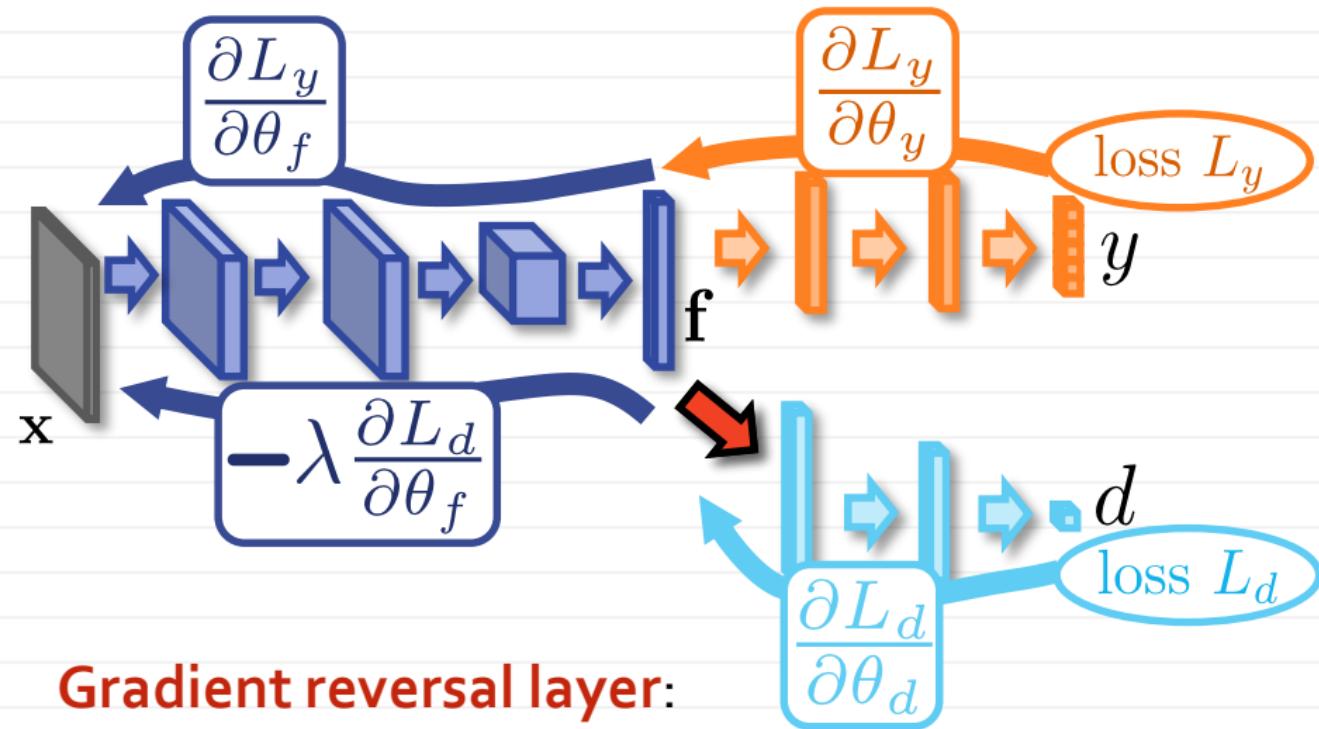
Idea 3: minimizing domain shift



Emerging features:

- Discriminative (good for predicting y)
- Domain-discriminative (good for predicting d)

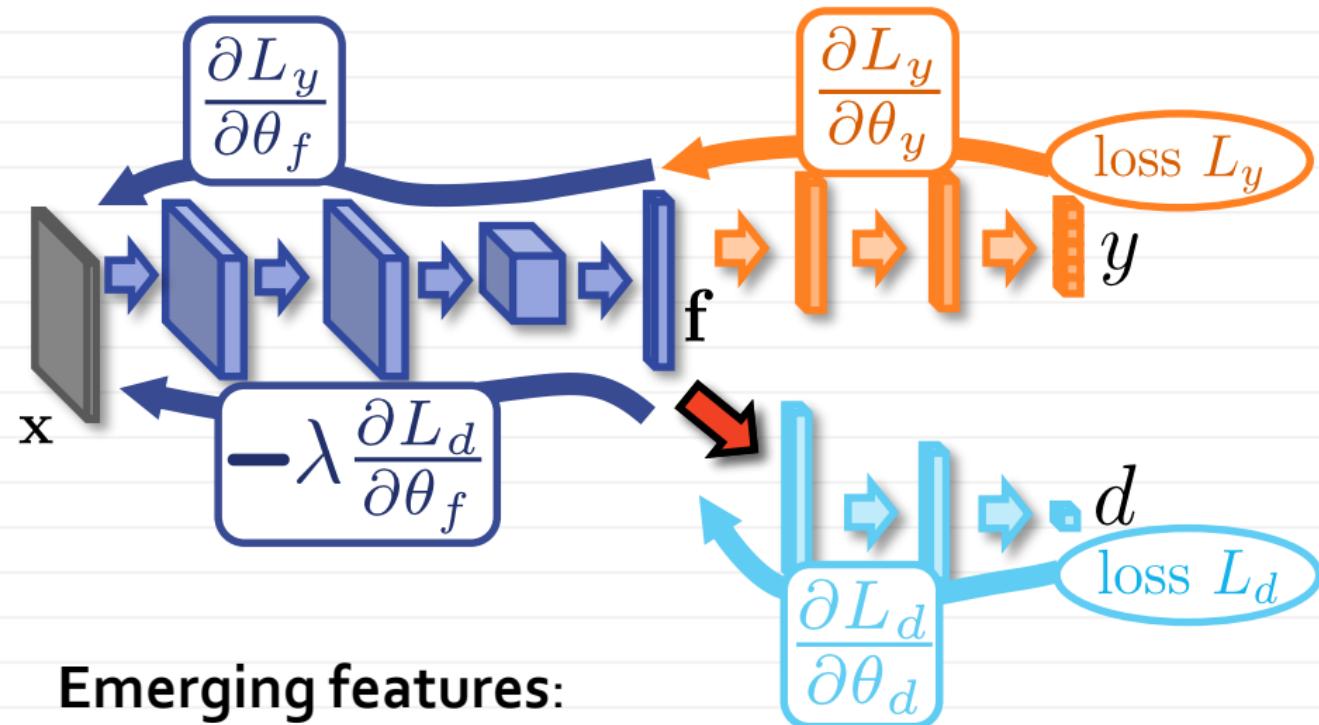
Idea 3: minimizing domain shift



Gradient reversal layer:

- Copies data without change at forwardprop
- Multiplies the gradient by $-\lambda$ at backprop

Idea 3: minimizing domain shift



Emerging features:

- Discriminative (good for predicting y)
- Domain-invariant (not good for predicting d)

Gradient reversal layer

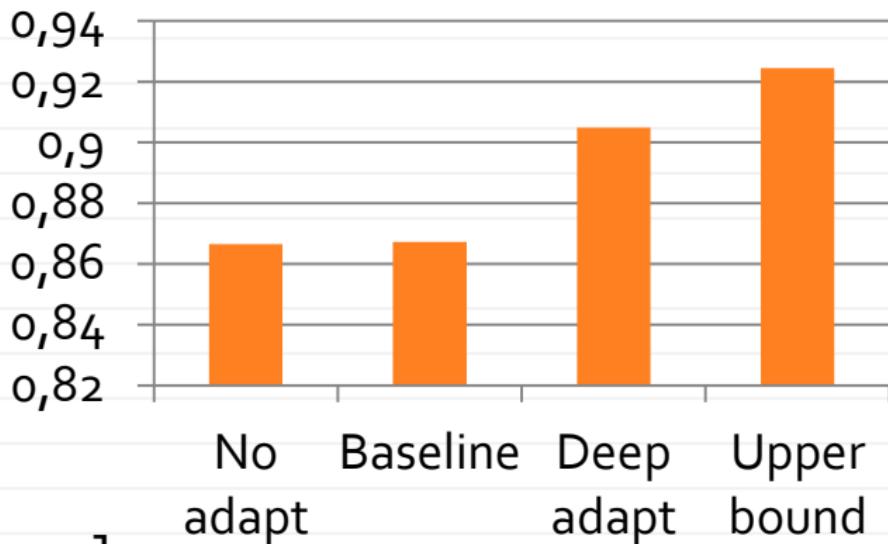
```
class GradReversalLayer : Layer {  
  
    float lambda;  
  
    blob forward (blob x)      {  
        return x  
    }  
  
    blob backward(blob dzdy) {  
        return multiply(dzdy, -lambda)  
    }  
}
```

Example: from synthetic to real



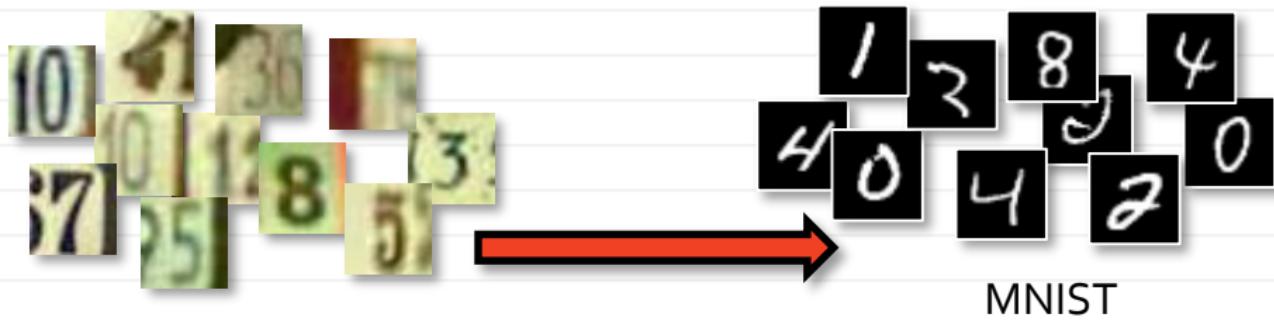
"Windows digits"

"House numbers"

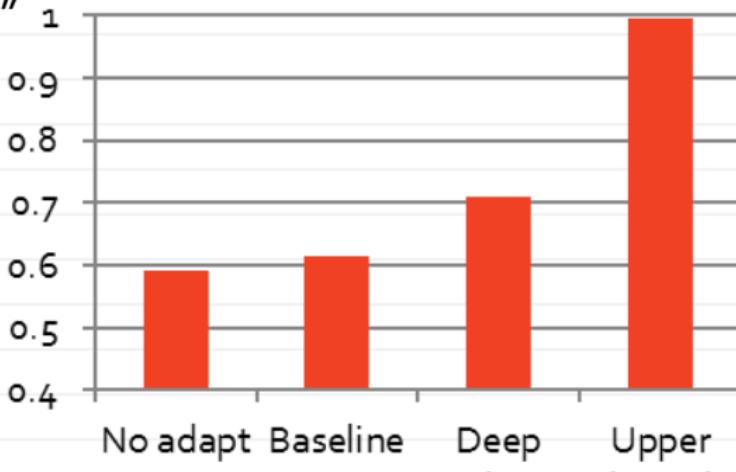


[Ganin et al. 2015]

Example: large gap



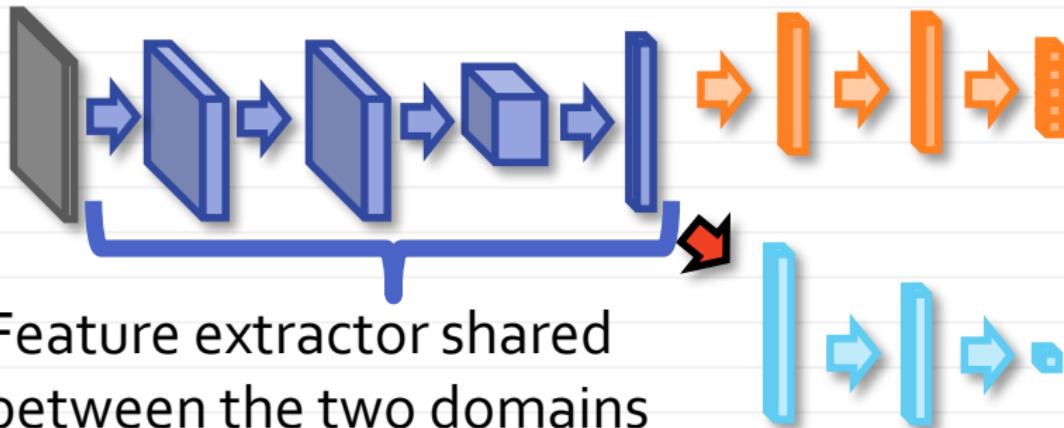
"House numbers"



Reverse
direction
does not
work ☹

[Ganin et al. 2015]

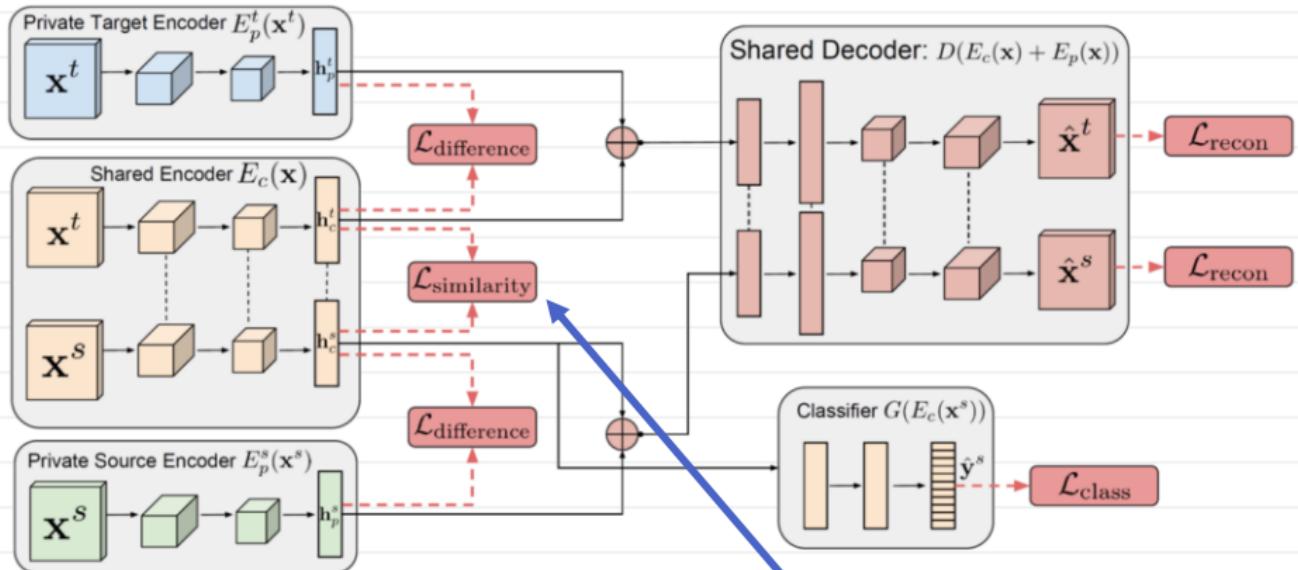
To share or not to share?



Feature extractor shared
between the two domains



Domain-separation networks

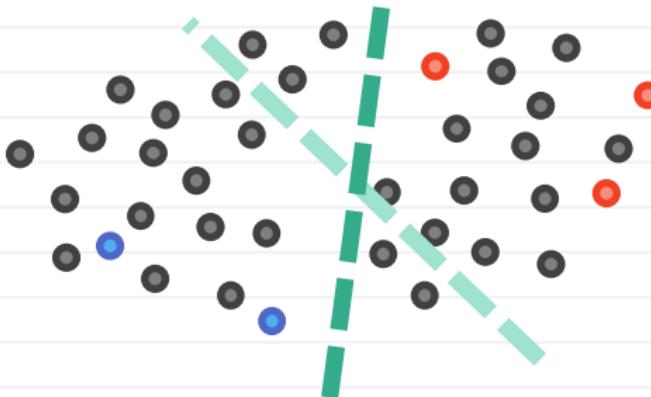


$$\mathcal{L}_{\text{difference}} = \left\| \mathbf{H}_c^{s\top} \mathbf{H}_p^s \right\|_F^2 + \left\| \mathbf{H}_c^{t\top} \mathbf{H}_p^t \right\|_F^2 \quad (\text{adversarial}) \text{ domain classifier}$$

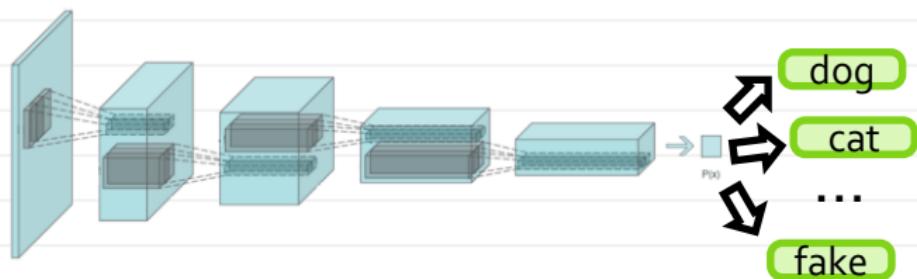
The classifier uses shared component only

[Bousmalis, Trigeorgis, et al. NIPS16]

Semi-supervised learning



New discriminator/classifier:



[Salimans et al. 2016]

Semi-supervised learning

$$\max_f = \mathbb{E}_{x,y \sim (X,Y)} \log \frac{f(x)[y]}{1 - f(x)[K+1]}$$

Standard log-loss

$$+ \mathbb{E}_{x \sim X} [\log(1 - f(x)[K+1])]$$
$$+ \mathbb{E}_{z \sim Z} [\log f(g(z))[K+1]]$$

Unsupervised loss

Model Perturbed MNIST results:	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [21]			333 \pm 14	
Virtual Adversarial [22]			212	
CatGAN [14]			191 \pm 10	
Skip Deep Generative Model [23]			132 \pm 7	
Ladder network [24]			106 \pm 37	
Auxiliary Deep Generative Model [23]			96 \pm 2	
Our model	1677 \pm 452	221 \pm 136	93 \pm 6.5	90 \pm 4.2
Ensemble of 10 of our models	1134 \pm 445	142 \pm 96	86 \pm 5.6	81 \pm 4.3

[Salimans et al. 2016]

Bibliography

Ian J. Goodfellow:

NIPS 2016 Tutorial: Generative Adversarial Networks. CoRR abs/1701.00160 (2017)

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, Yoshua Bengio:
Generative Adversarial Nets. NIPS 2014: 2672-2680

Alec Radford, Luke Metz, Soumith Chintala:

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR abs/1511.06434 (2015)

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen:

Improved Techniques for Training GANs. NIPS 2016: 2226-2234

Shane Barratt, Rishi Sharma A Note on the Inception Score. ArXiV 2018

Martín Arjovsky, Soumith Chintala, Léon Bottou:

Wasserstein Generative Adversarial Networks. ICML 2017: 214-223

Bibliography

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, Aaron C. Courville:

Improved Training of Wasserstein GANs. NIPS 2017: 5769-5779

Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen:

Progressive Growing of GANs for Improved Quality, Stability, and Variation. CoRR abs/1710.10196 (2017)

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, Alexei A. Efros:

Generative Visual Manipulation on the Natural Image Manifold. ECCV (5) 2016:
597-613

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros:

Image-to-Image Translation with Conditional Adversarial Networks. CVPR 2017:
5967-5976

Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros:

Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. ICCV 2017: 2242-2251

Bibliography

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. CVPR 2017: 105-114

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, Russell Webb:

Learning from Simulated and Unsupervised Images through Adversarial Training.
CVPR 2017: 2242-2251

Yaroslav Ganin, Victor S. Lempitsky:

Unsupervised Domain Adaptation by Backpropagation. ICML 2015: 1180-1189

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, Dumitru Erhan:

Domain Separation Networks. NIPS 2016: 343-351