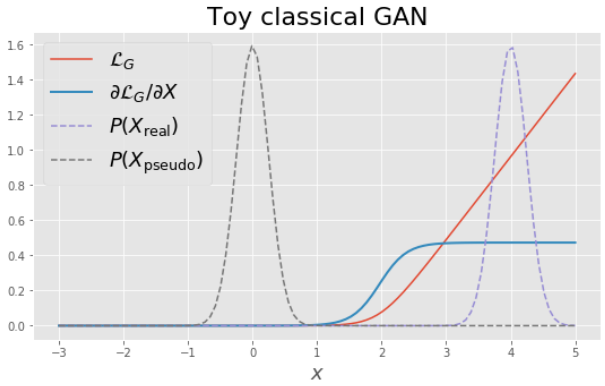
Andrey Ustyuzhanin

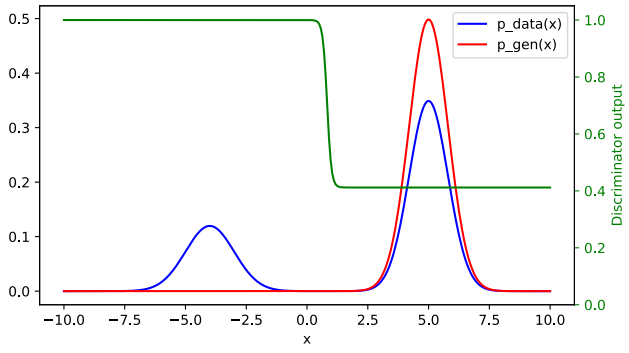
Advanced GANs

2021

Recap: problems with GANs

- Delicate generator and discriminator balancing
  - Generator wins → mode collapse
  - Discriminator wins → vanishing gradients
- Disjoint supports for p_g and p_d leads to quick win of D via vanishing gradients
- Mode collapse



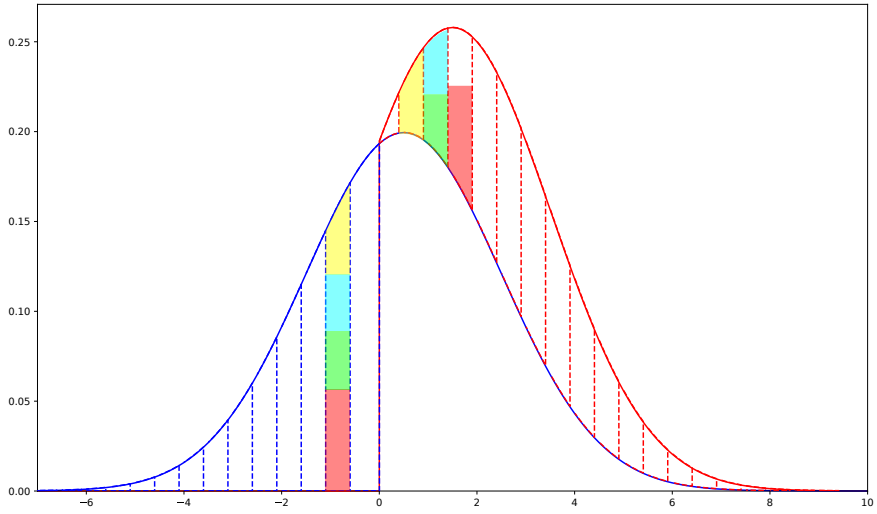


Wasserstein GAN Motivation

- How can we solve vanishing gradients problem?
- **IDEA**: Use a measure that would be smooth regardless of support differences

Wasserstein distance

Also called
"Earth mover's distance" (EMD)



- Distributions $P(x)$ and $Q(x)$ are viewed as describing the **amounts of "dirt" at point x**
- We want to convert one distribution into the other by **moving around** some amounts of dirt

- The cost of moving an amount $\boldsymbol{m}$ from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$ is $\boldsymbol{m}\times\|\boldsymbol{x_2}-\boldsymbol{x_1}\|$
- $\mathrm{EMD}(P,Q)$ = **minimum total cost** of converting P into Q

Idea of definition

- Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:
  $\gamma(x_1, x_2)dx_1dx_2$ – how much dirt we're moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$
- Then, the cost of moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$ is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2)dx_1dx_2$$

- and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2)dx_1dx_2 = \mathbb{E}_{x_1, x_2 \sim \gamma(x_1, x_2)} \|x_2 - x_1\|$$

- Since we want to convert P to Q, the plan has to satisfy:

$$\int_{x_1} \gamma(x_1, x_2)dx_1 = Q(x_2), \qquad \int_{x_2} \gamma(x_1, x_2)dx_2 = P(x_1)$$

Idea of Definition

- Let π be the set of all plans that convert P to Q, i.e.:

$$\pi = \left\{ \gamma: \quad \gamma \geq 0, \qquad \int_{x_1} \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int_{x_2} \gamma(x_1, x_2) dx_2 = P(x_1) \right\}$$

- Then, the Wasserstein distance between P and Q is:

$$\mathrm{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} \|x_2 - x_1\|$$

Optimization over all transport plans – not too friendly

Wasserstein Distance

For continuous case, there are a set of p-Wasserstein distances, with $W_p(p_x, q_y)$ defined with $x \in M, y \in M$ and a distance D on x, y:

$$W_p(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x,y)^p d\gamma(x,y),$$

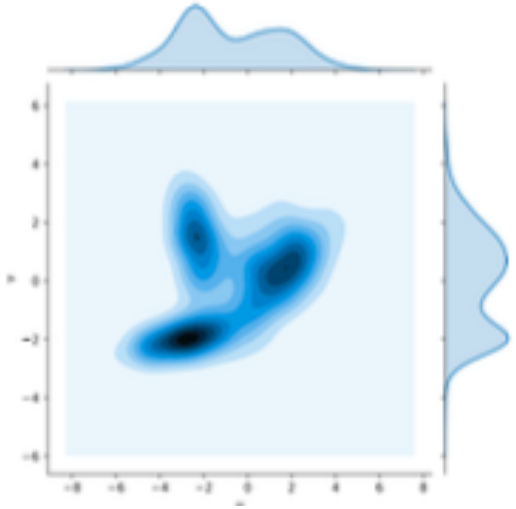
where $\Pi(x,y)$ is a set of all joint distributions having p_x, q_y as their marginals.

W_1 distance

In particular, W_1 distance with Euclidean norm is:

$$W(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x,y) d\gamma(x,y) = \inf_{\gamma \in \Pi(x,y)} \mathbb{E}(||x - y||)$$

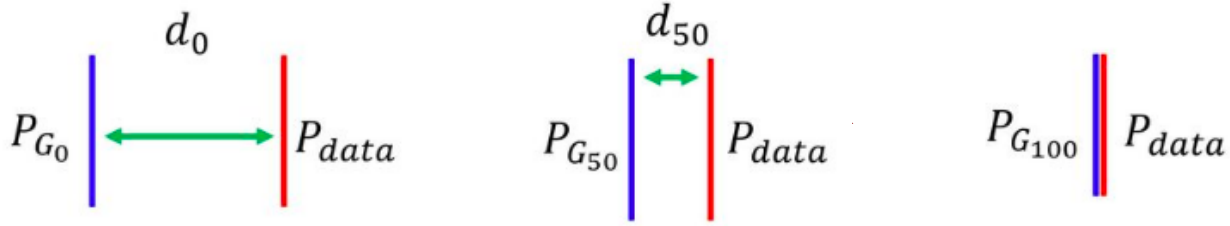
Which brings an evident connection to EMD.



Wikimedia

Two dimensional representation of the transport plan between horizontal (μ) and vertical ν pdfs. Note, that this is not unique plan. The inf must be taken over all possible plans.

Convergence Example

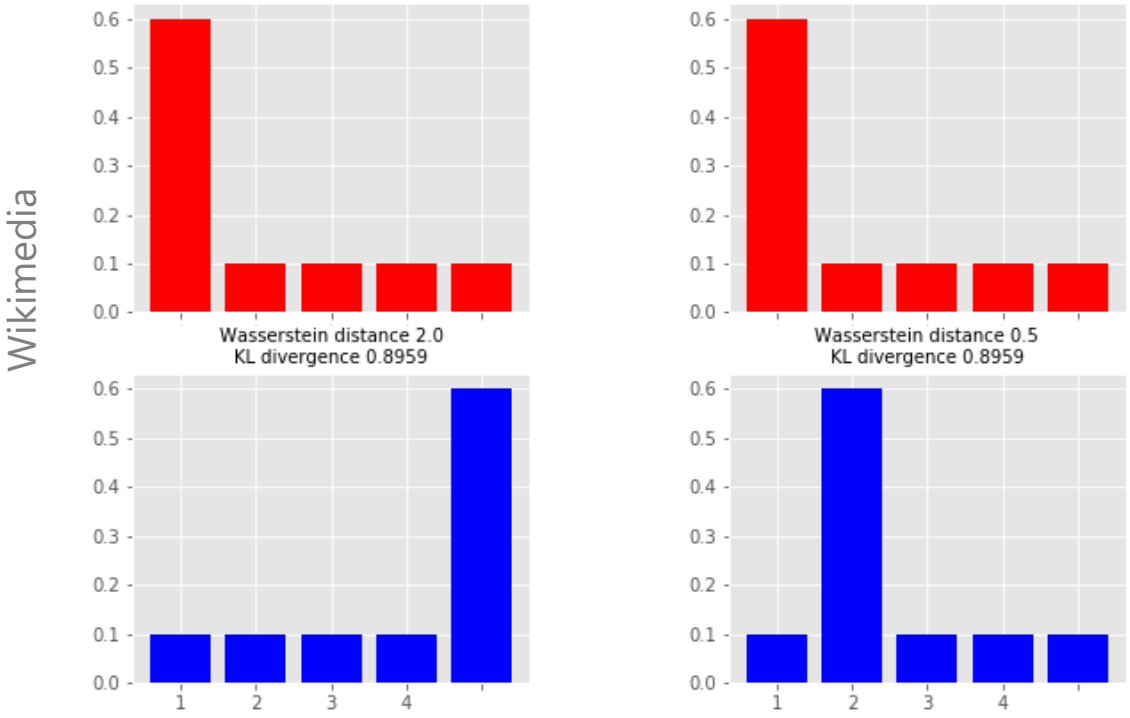


TV:	1	1	0
KL:	∞	∞	0
JS:	$\log 2$	$\log 2$	0
W:	d_0	d_{50}	0

Mass Attention

W takes into account the distance at which the differences in the distributions is located.

This is exactly what we need to deal with vanishing gradients!



W distance properties

P – true PDF, Q – generated PDF.

- For a sequence of generated distributions Q_n:

  $KL(P||Q_n) \to 0$ → $JS(P; Q_n) \to 0$ → $W(P; Q_n) \to 0$, $Q_n \xrightarrow{\text{D}} P$

- For $Q_\theta \sim g_\theta(z)$, $g_\theta(z)$ continuos

  $W(Q_\theta; Q)$ is continuous and can be restricted to be differentiable almost everywhere.

- W-distance doesn't suffer from zero gradients unlike KL or JS.
- More robust in multidimensional cases than KL.
- Uneasy to optimize.

Wasserstein distance computation

$$\mathrm{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} ||x_1 - x_2||$$

How do we compute it?

Those “transport plans” don’t look friendly…

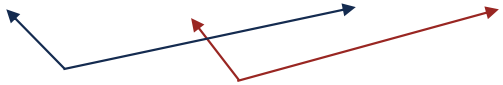
Kantorovich-Rubinstein duality

Dark magic time!

$$\mathrm{EMD}(P,Q) = \inf_{\gamma\in\pi} \mathbb{E}_{x_1,x_2\sim\gamma}||x_1 - x_2||$$

Step 1. Add this:

$$+\inf_{\gamma}\sup_{f} \mathbb{E}_{x_1,x_2\sim\gamma}\left[\mathbb{E}_{s\sim P} f(s) - \mathbb{E}_{t\sim Q} f(t) - (f(x_1) - f(x_2))\right]$$



- $f(x)$ - any function $X \to \mathbb{R}$
- the term equals to zero for $\gamma \in \pi$
- the term equals to $+\infty$ for $\gamma \notin \pi$

Those terms cancel each other when $\gamma \in \pi$

$$\pi = \left\{\gamma : \int \gamma(x_1, x_2)dx_1 = Q(x_2), \int \gamma(x_1, x_2)dx_2 = P(x_1)\right\}$$

Step 2: Remove the $\gamma \in \pi$ condition from the whole expression:

$$= \inf_{\gamma}\sup_{f} \mathbb{E}_{x_1,x_2\sim\gamma}\left[||x_1 - x_2|| + \mathbb{E}_{s\sim P} f(s) - \mathbb{E}_{t\sim Q} f(t) - (f(x_1) - f(x_2))\right]$$

Kantorovich-Rubinstein duality

Step 3 (dark magic): Infimum and supremum operations can be swapped under certain conditions

$$= \sup_f \inf_\gamma \left[\mathbb{E}_{s\sim P} f(s) - \mathbb{E}_{t\sim Q} f(t) + \underbrace{\mathbb{E}_{x_1,x_2\sim\gamma} \left[||x_1 - x_2|| - (f(x_1) - f(x_2)) \right]}_{\text{This becomes zero}} \right]$$

Step 4: impose a restriction $\|f(x)\|_L \leq 1$:
$\forall a, b \in R: |f(a) - f(b)| \leq \|a - b\|$

$$\mathrm{EMD}(P, Q) = \sup_{||f||_L \leq 1} \left[\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x) \right]$$

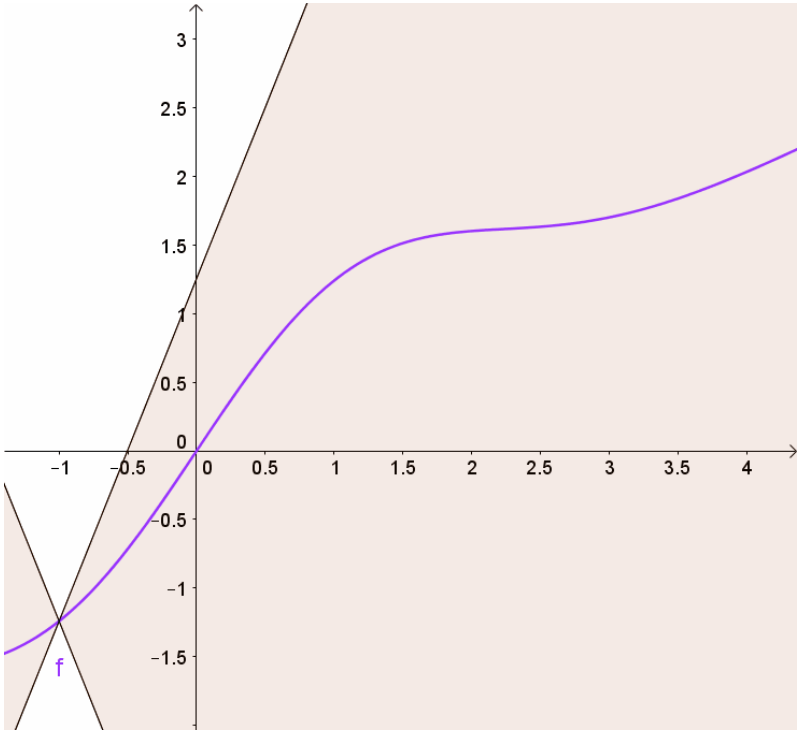
https://vincentherrmann.github.io/blog/wasserstein/

Lipschitz continuity

- f is Lipschitz-k continuous if there exists a constant $k \geq 0$, such that for all x_1 and x_2:

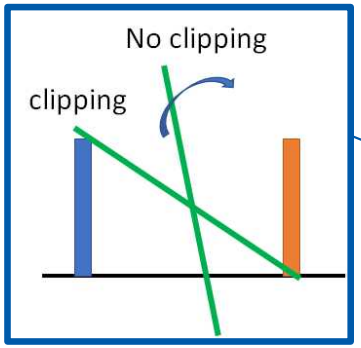
$$|f(x_1) - f(x_2)| \leq \mathrm{k} \cdot \|x_1 - x_2\|$$

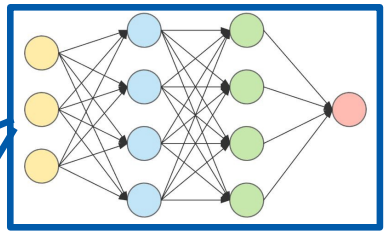
- Thus, f in EMD could be approximated by a neural network, we just must make sure it meets the Lipschitz continuity constraint.



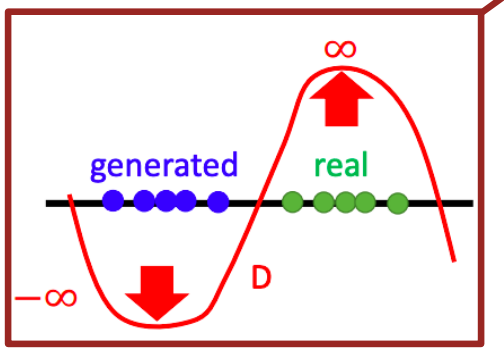
img from https://en.wikipedia.org/wiki/Lipschitz_continuity

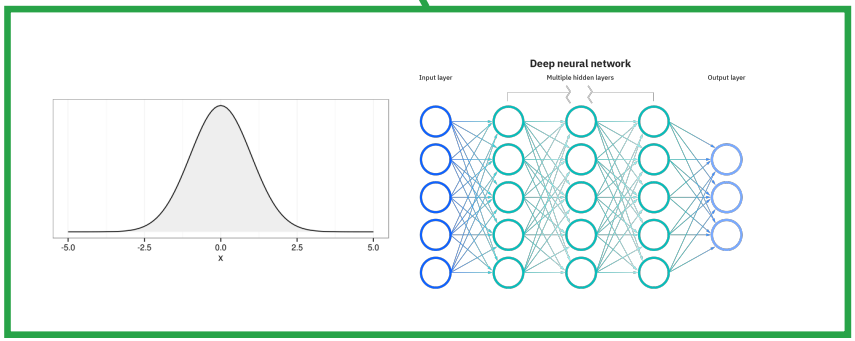
Lipschitz-1 Condition and Neural Networks





$$W(P;Q) = \sup_{f \in Lip_1} \left(\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x)\right),$$





Lipschitz-1 Condition and Neural Networks

$$W(P;Q) = \sup_{f \in Lip_1} \left(\mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x) \right),$$

- f is a neural net – **discriminator** ('**critic**' in the original paper).
- The expectations is estimated from samples.
- Lipschitz-1 continuity can be replaced with Lipschitz-k continuity
  - estimate $k \times \mathrm{W}(P, Q)$
  - achieved **by clipping the weights** of the critic:
    $w \to \mathrm{clip}(w, -c, c)$ with some constant c.

M. Arjovsky et al. **Wasserstein GAN**

Wasserstein GAN

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

$f_w(x)$ – discriminator(critic)
$g_\theta(z)$ – generator

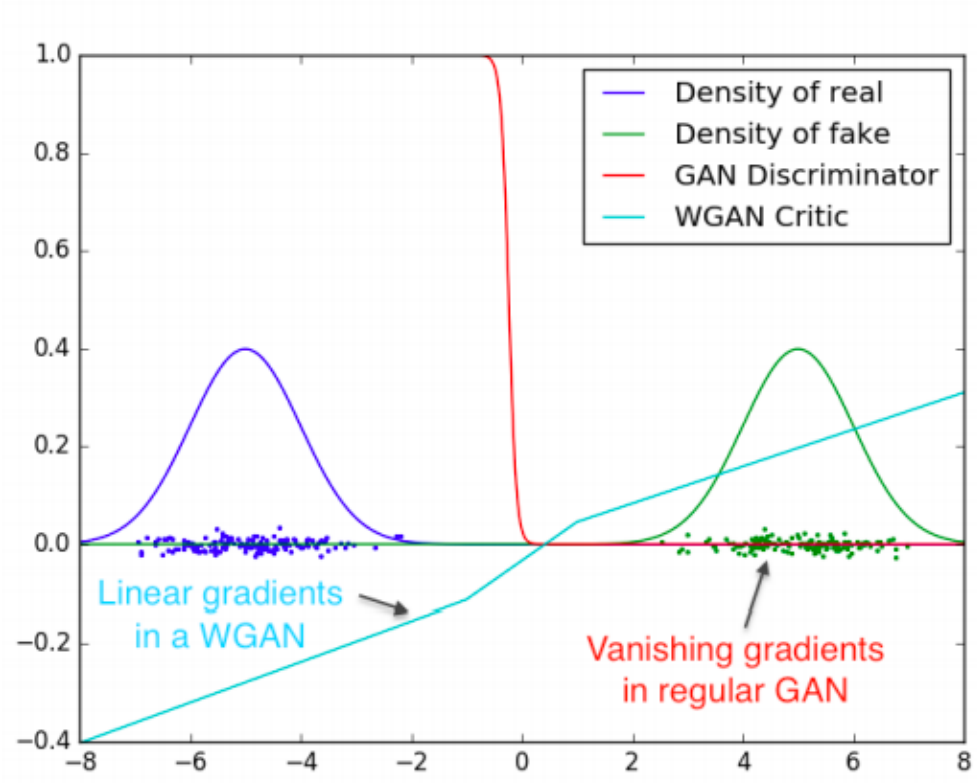
EMD()

Lipschitz constraint

https://arxiv.org/abs/1701.07875

WGAN: problems solved

- the vanishing gradient problem is **solved**;
- mode collapse problem is **addressed;**
- Quote from the paper: "Weight clipping is a *clearly terrible* way to enforce a Lipschitz constraint".



https://arxiv.org/abs/1701.07875

WGAN: results

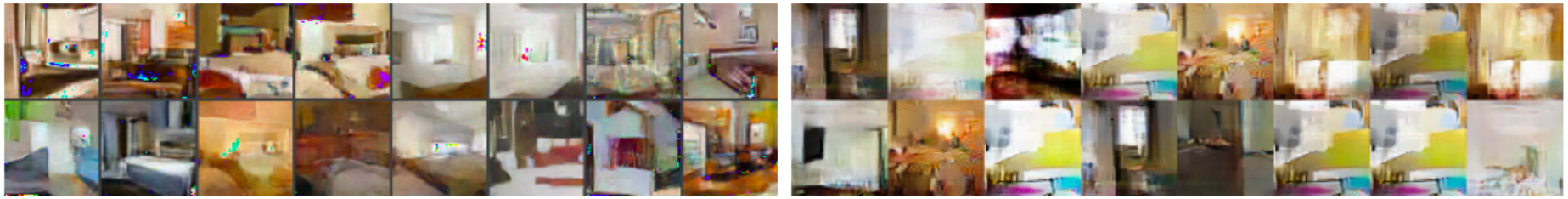


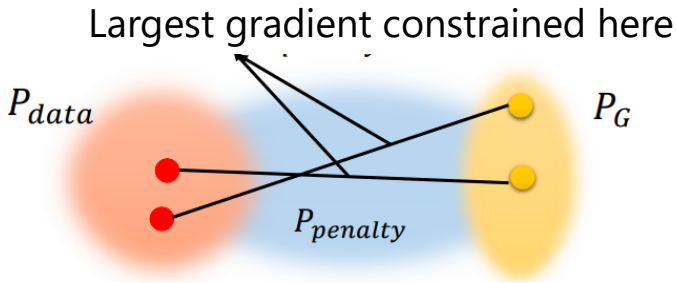
Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

WGAN + Gradient Penalty (WGAN-GP)

- Weight clipping makes the critic less expressive and the training harder to converge
- Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q
- Also: $\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$
- Can replace weight clipping with a gradient penalty term:

$$\text{GP} = \lambda \int \max[(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]\, dx$$

$$\text{GP} = \lambda \mathbb{E}_{\tilde{x} \sim \mathbb{P}_{\tilde{x}}}[(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]$$

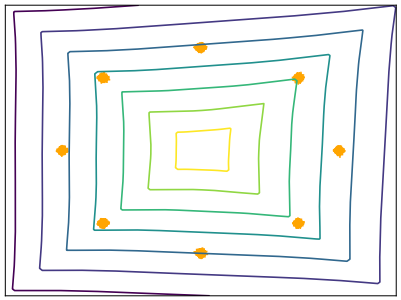


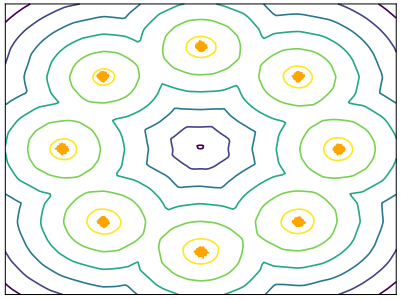
$$\mathbb{P}_{\tilde{x}}: \begin{bmatrix} \tilde{x} = \alpha x_1 + (1-\alpha) x_2 \\ \alpha \sim \text{Uniform}(0,1) \\ x_1 \sim P \\ x_2 \sim Q \end{bmatrix}$$

Ishaan Gulrajani **Improved Training of Wasserstein GANs**

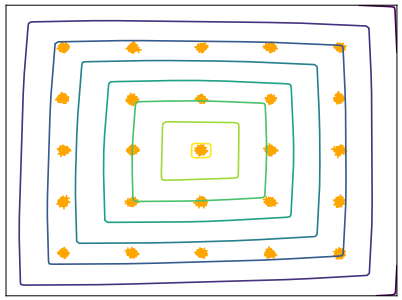
WGAN + GP illustration

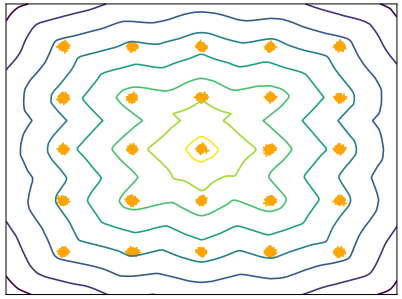
8 Gaussians



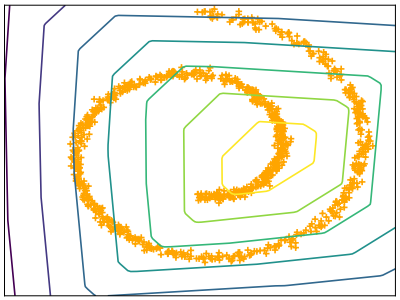


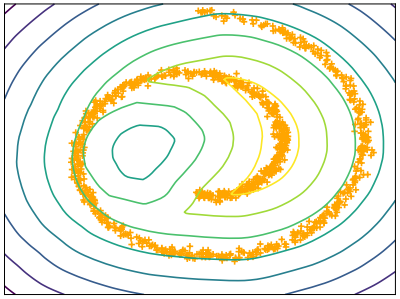
25 Gaussians





Swiss Roll





WGAN-GP training

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

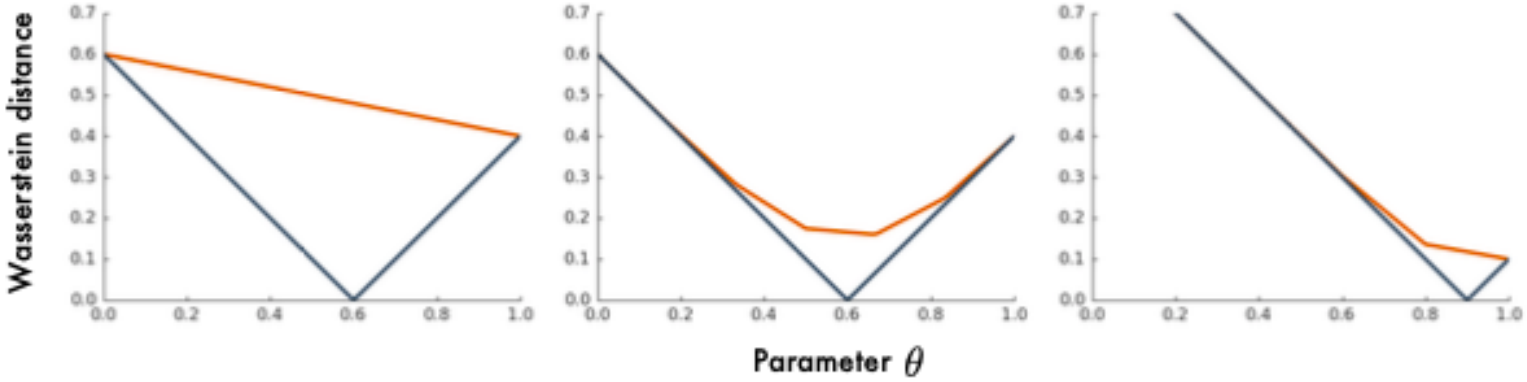
Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

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, Ishaan, et al. "Improved training of Wasserstein GANs." Advances in neural information processing systems. 2017.

WGAN: problems

- The expected EMD gradients can differ from the true gradients.
- This leads to problems even for Bernoulli distribution.



Red for sample gradient expectation, blue is for real gradients solution.
Left to right $\theta^* = 0.6; 0.6; 0.9$.

M. Bellemare et al. The Cramer Distance as a Solution to Biased Wasserstein Gradients

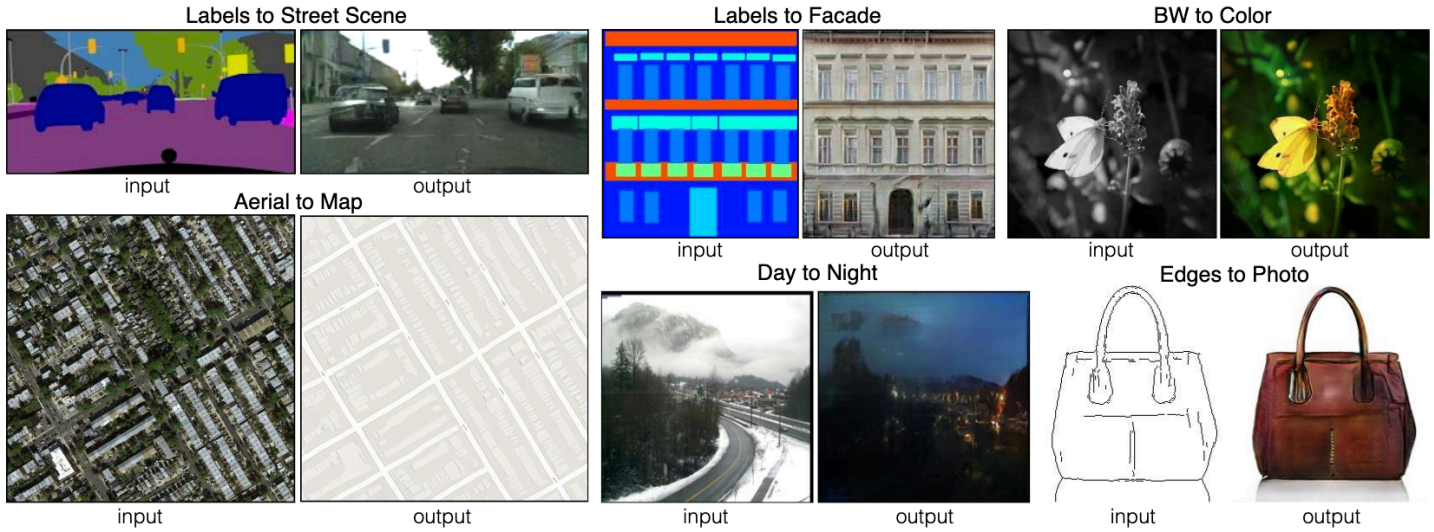
Recap

WGAN and WGAN-GP

- use interpretable loss function based on EMD;
- simpler and more robust training but needs to control discriminator's Lipschitz continuity;
- one of the most popular architectures;
- avoids vanishing gradients;
- partially addresses mode collapse problem;
- Issues: gradients can be biased.

Motivation for image-to-image translation

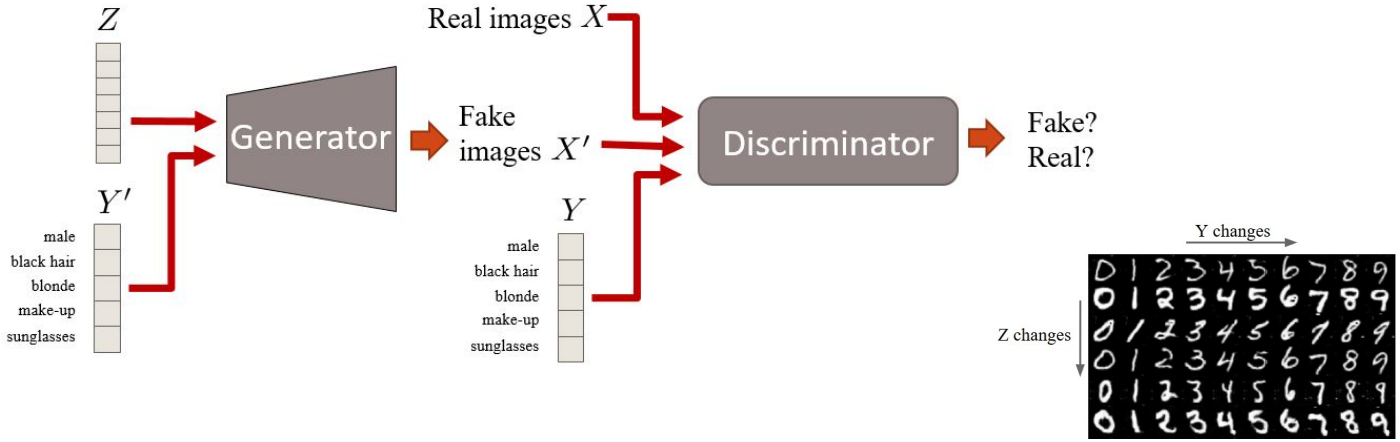
- How can we translate from one kind of image to the same kind but with a bit different representation (domain)?



- Turn night into day, winter -> summer, photo -> painting, etc.

https://arxiv.org/abs/1611.07004

Conditional GAN (cGAN) reminder

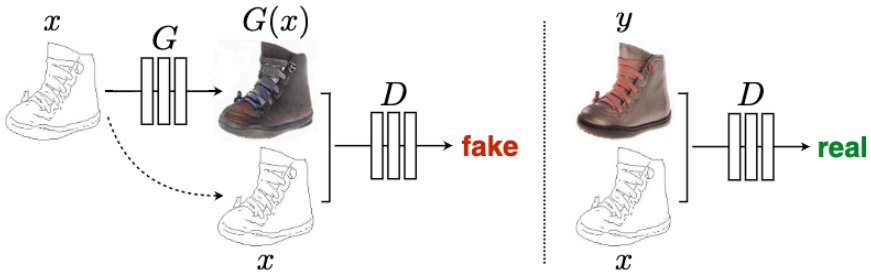


- We can concatenate labels with random noise vector, both for G and D;
- Giving more information to the generator leads to better samples and faster convergence than unconditional GANs;

https://arxiv.org/abs/1411.1784
https://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

Pix2Pix model

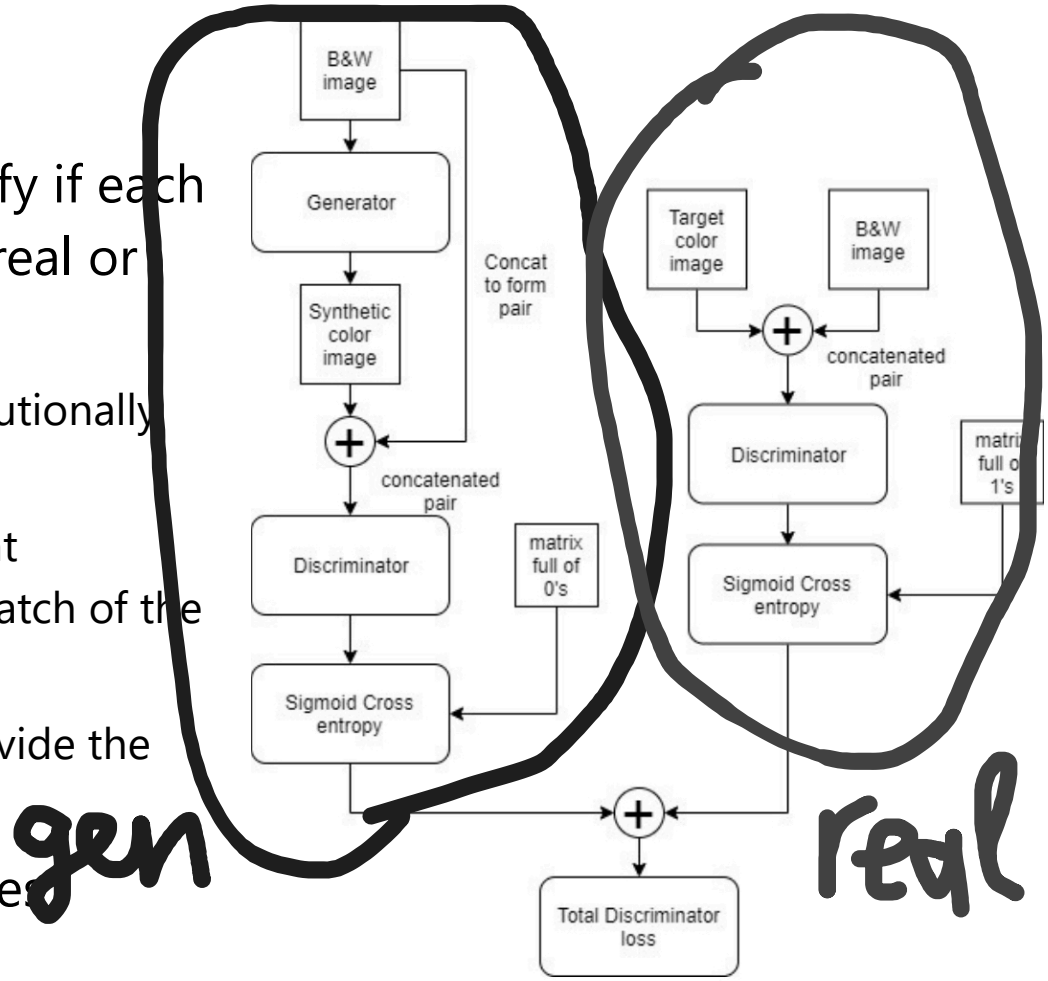
- Translates images from source domain to target domain:
  - Map / aerial image
  - Sketch / photorealistic object
  - Winter / summer
  - Night / day
- Adds target domain image as a condition to G along with noise z, and to D like cGAN does



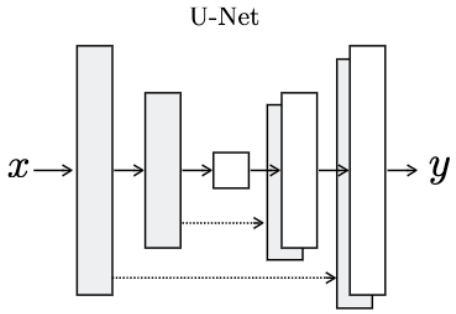
https://arxiv.org/pdf/1611.07004.pdf

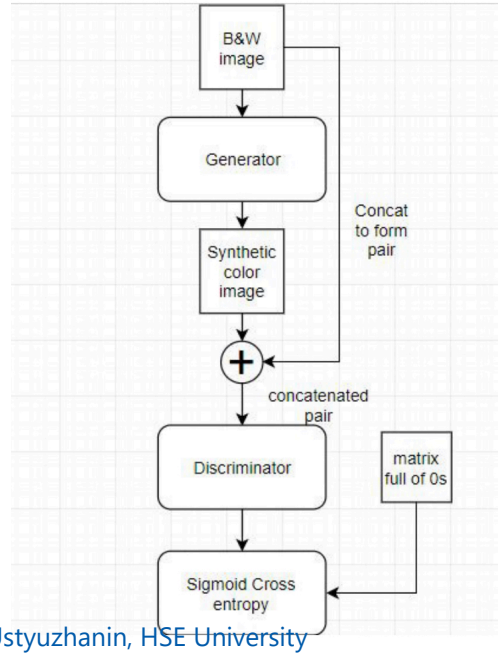
Pix2Pix architecture. Discriminator

- Discriminator: **PatchGAN**, discriminator tries to classify if each N×N patch in an image is real or fake.
  - Run this discriminator convolutionally across the image,
  - Each tensor element represent classification result for NxN patch of the original image,
  - averaging all elements to provide the ultimate output of D.
- Loss: generated + real losses



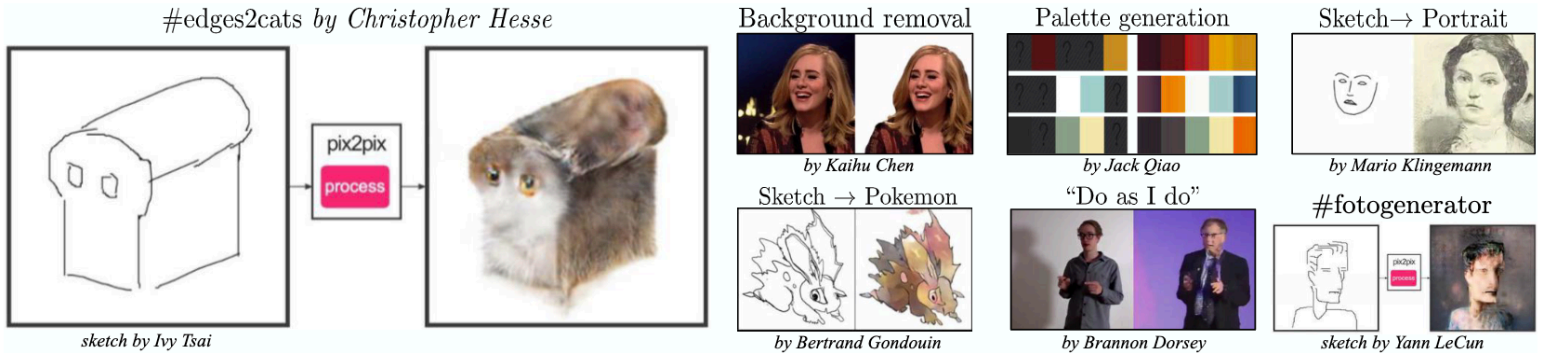
Pix2Pix architecture. Generator

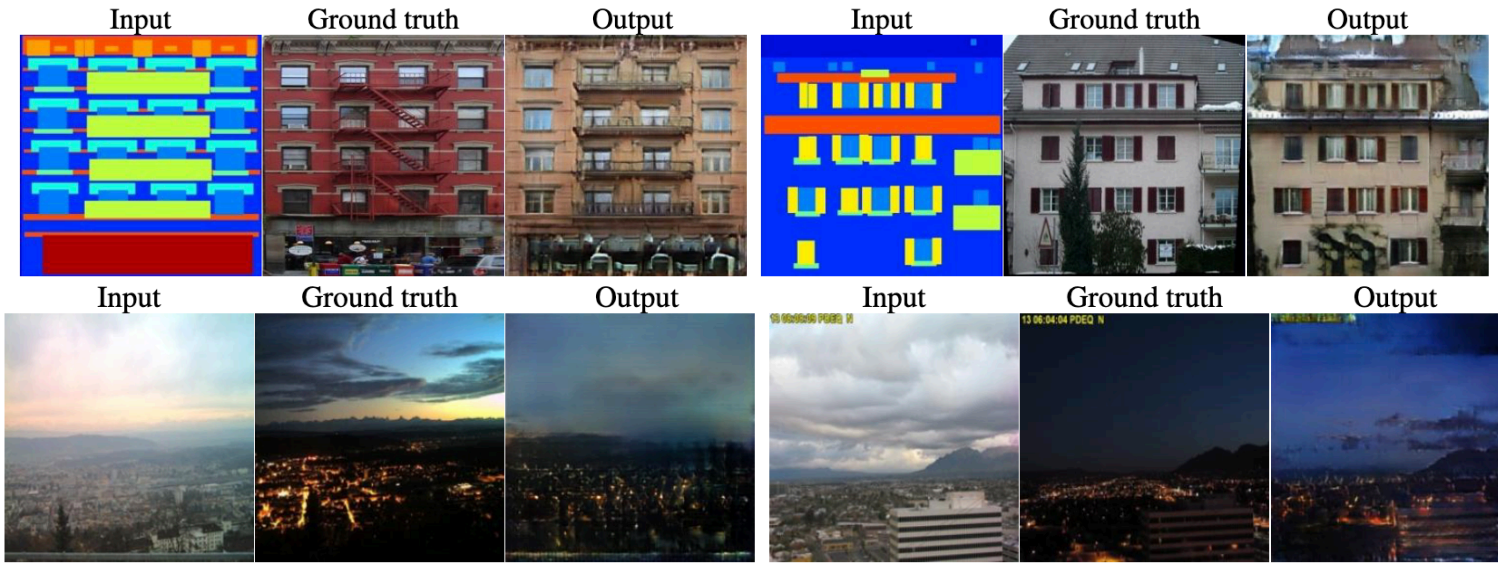




- Generator
  - U-Net architecture with skip-connections
  - Takes B&W image as input
  - Randomness is added in terms of dropout
  - Loss function: measures how real the synthetic images look
  - Add L1 between target and G(x)
- Training:
  - Take ImageNet
    - convert the color images to B&W
    - the color image itself is the target.
  - Pix2Pix networks can be trained by iterating through the data set

Pix2Pix examples

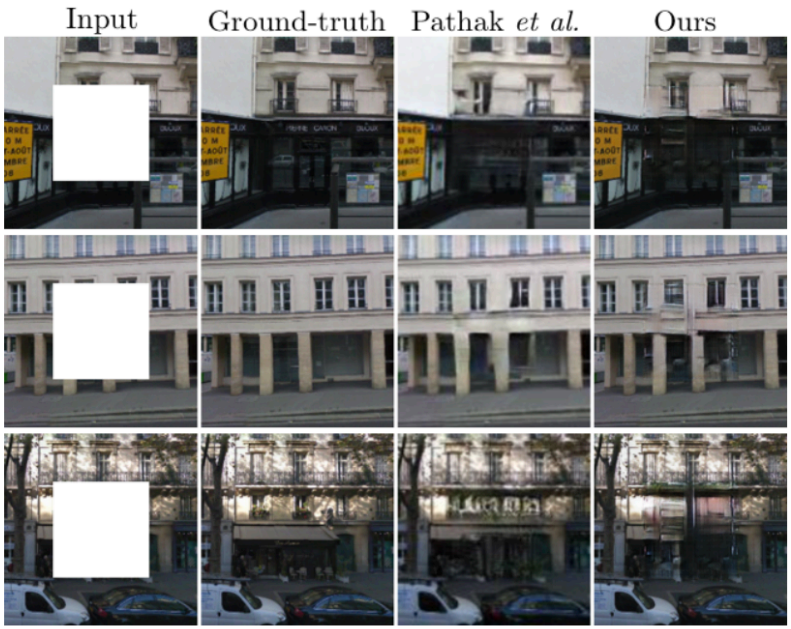




https://arxiv.org/pdf/1611.07004.pdf

Pix2Pix takeaway

- Conditional GAN by image. Requires paired image dataset;
- Suitable for many image to image:
  - Sketch -> X
  - Segmentation
  - Pose transfer
  - Day -> night, winter -> summer
  - Map -> aerial image
  - Colorization
  - Inpainting
- PatchGAN for discriminator
- U-Net as a generator can fill in the central hole by information from the periphery of the photos.
  - Combines regular loss with L1
- Prone to a mode collapse issues

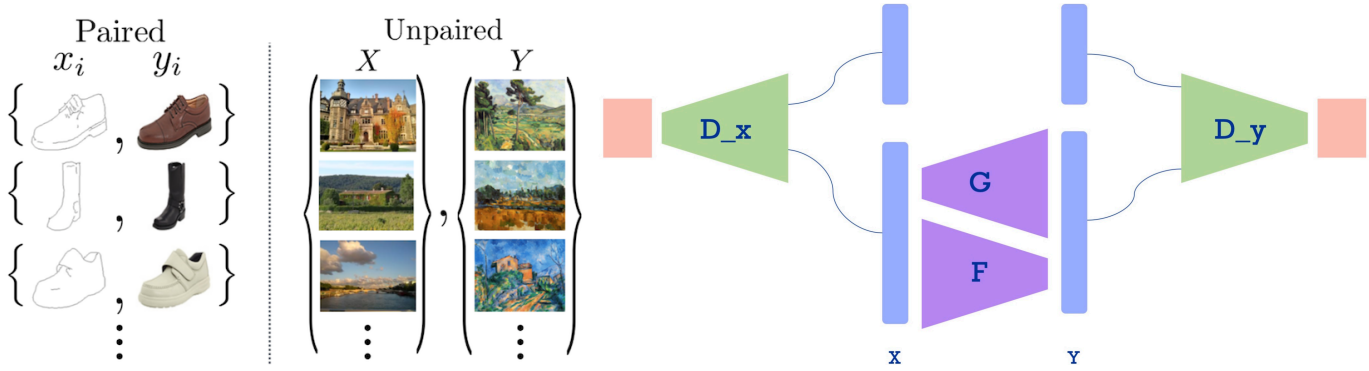


https://arxiv.org/pdf/1611.07004.pdf

CycleGAN – unpaired image to image map

- Pix2Pix requires paired sample of images to work
  - Expensive in terms of handwork
  - Requires an algorithm for filtering unlabeled images, not always well-defined
- CycleGAN solves the task on the set level:
  - takes 2 sets of *unpaired* data from domain X and domain Y, no white noise needed
  - aims to learn the underlying relation between these two domains, in both directions.

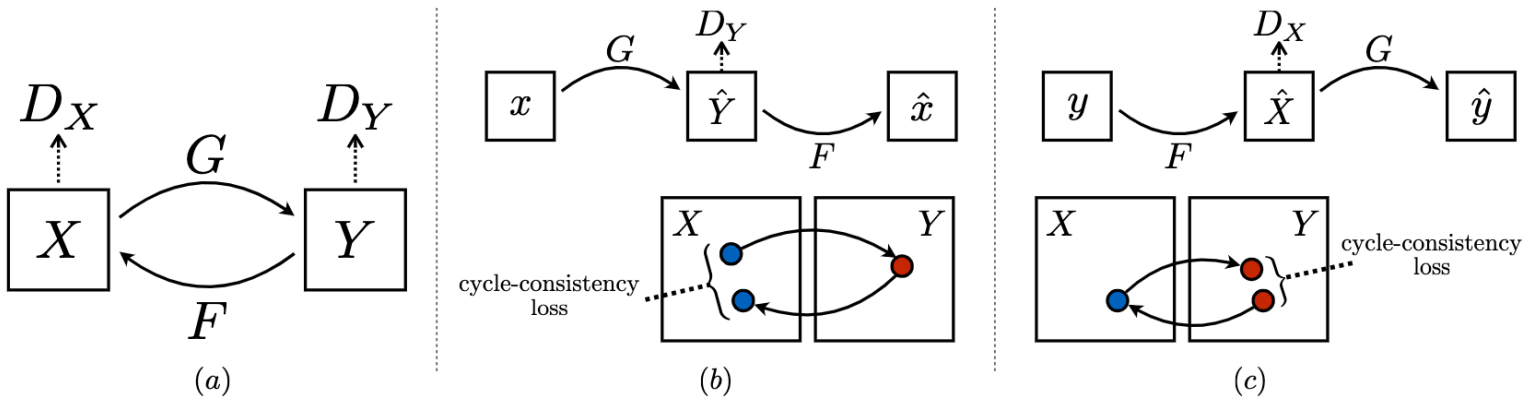
    1) x ∈ X→ y ∈ Y is done by the generator G, 2) y → x is done by F. No prior



https://bit.ly/3bT1urM

CycleGAN details

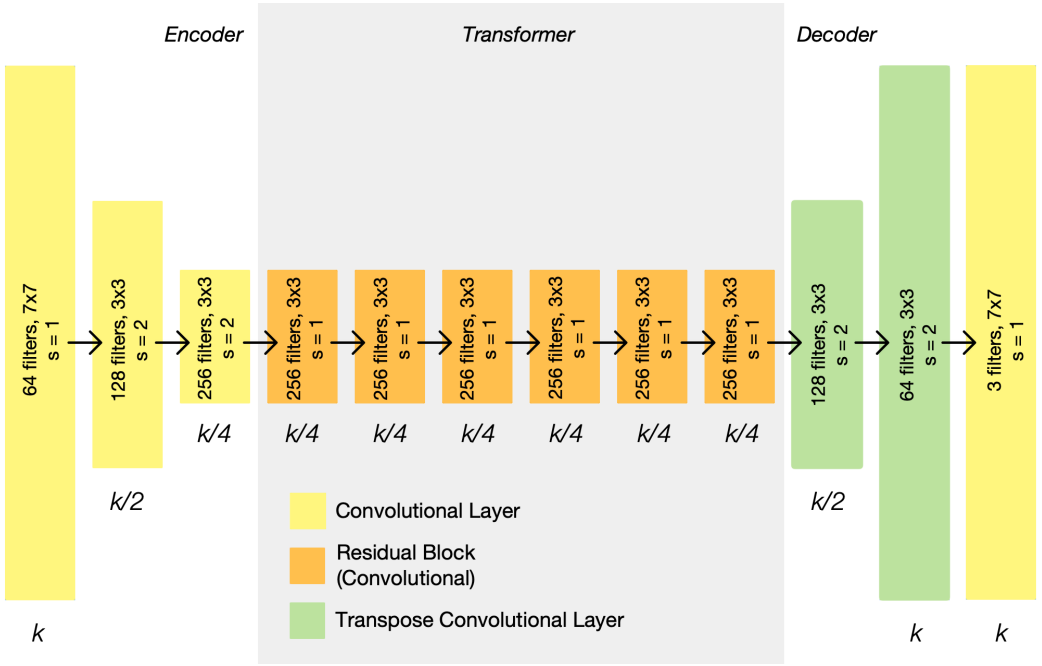
- Adversarial loss: Least Square GAN's loss for both G-D_Y and F-D_X pairs
- Cycle consistency loss:



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

https://arxiv.org/pdf/1703.10593.pdf

CycleGAN Generators

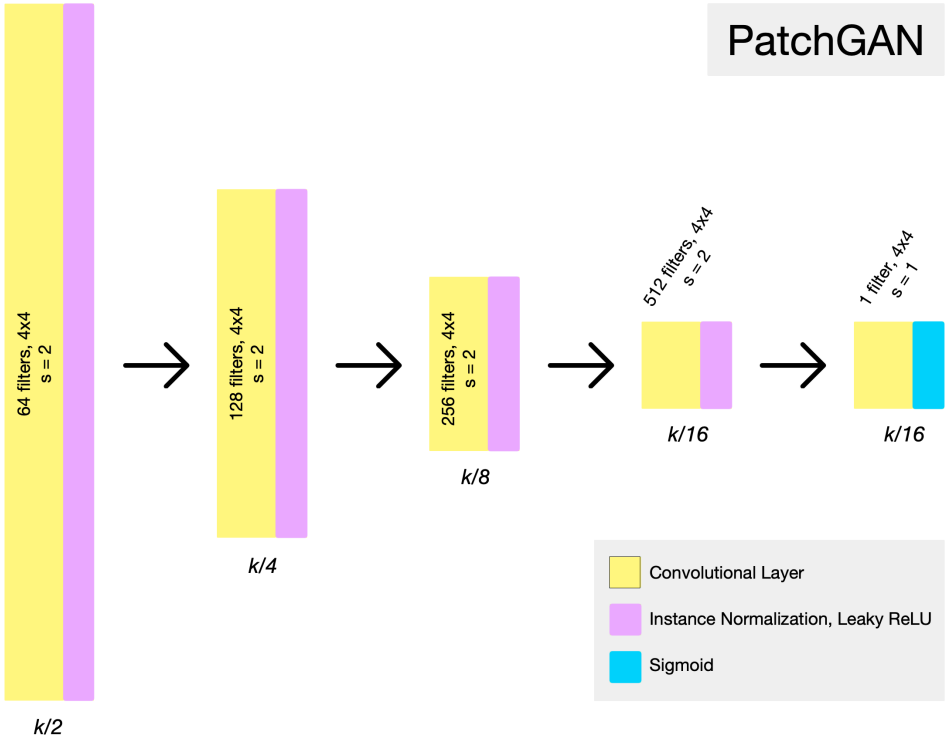


- Both G and F have same architecture
  - take 256x256 image, downscale it to 64x64 (256 channels) and upscale it back
  - Since the generators' architecture is fully convolutional, they can handle arbitrarily large input once trained.

https://bit.ly/2SqyLmZ

CycleGAN Discriminator

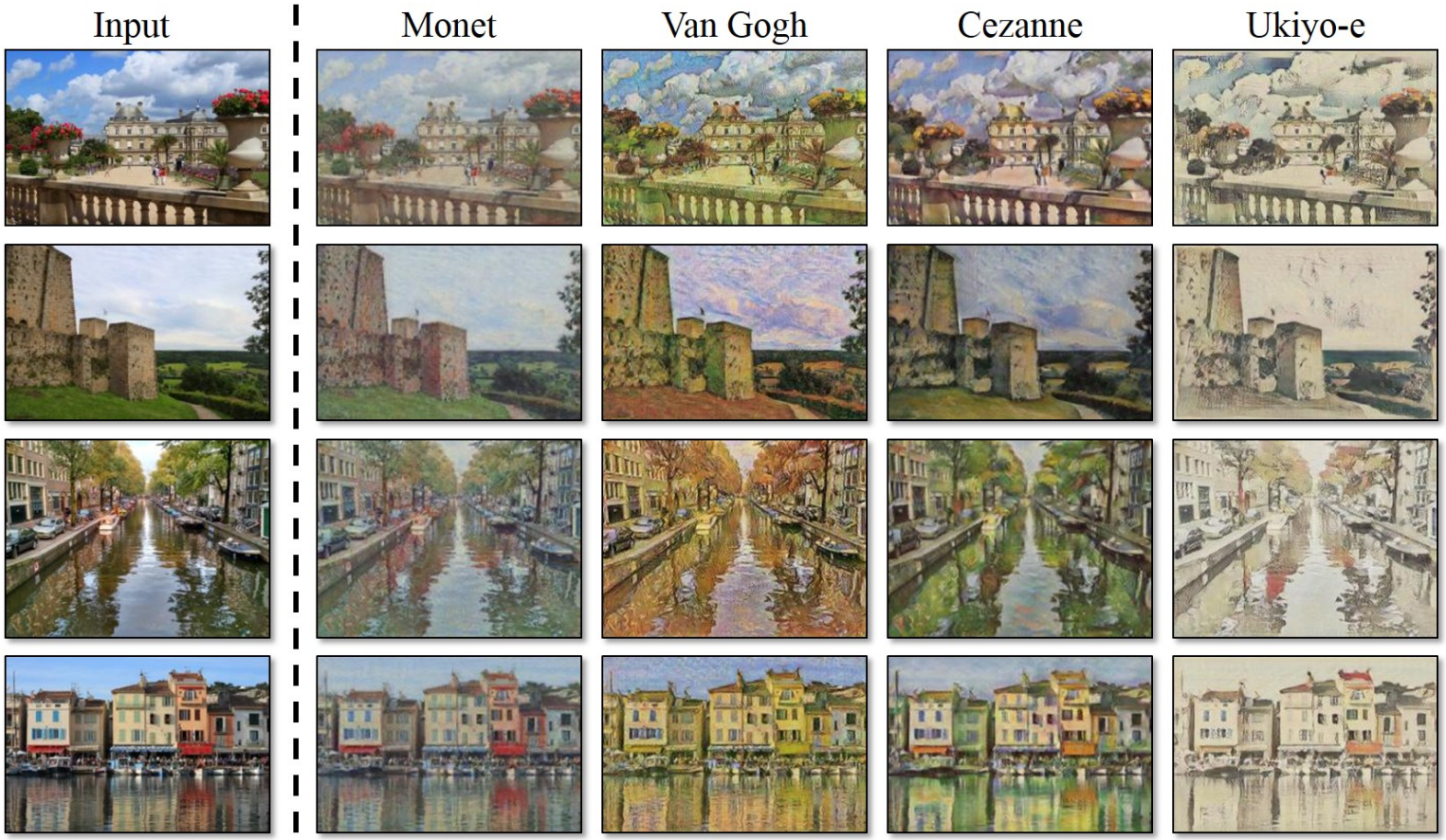
- PatchGAN: a fully convolutional network, that takes in an image, and produces a matrix of probabilities, each referring to the probability of the corresponding “patch” of the image being “real”.



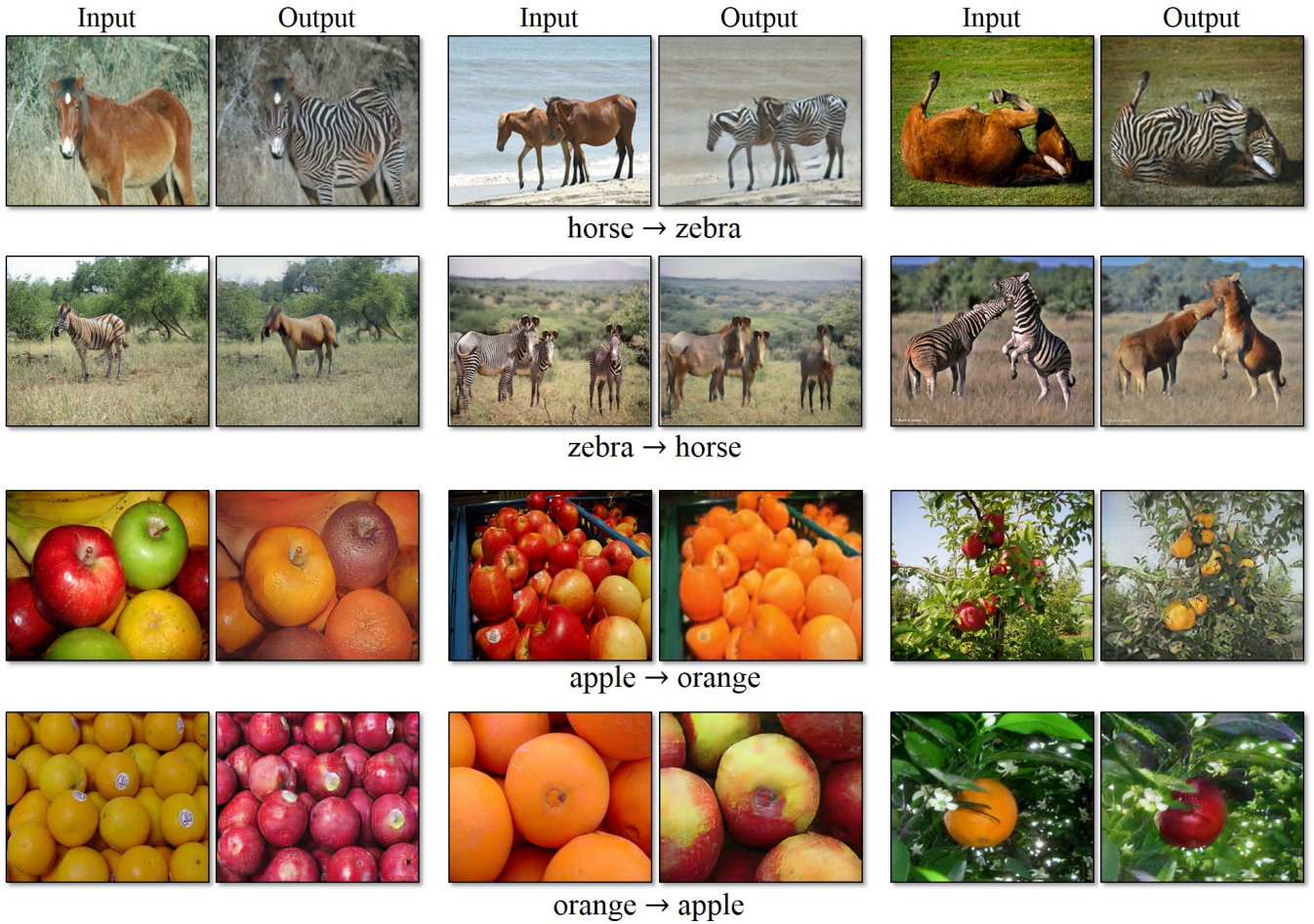
CycleGAN training

- Data: random image pair from domain X and Y
- Oscillation reduction: to prevent the model from changing drastically from iteration to iteration, the discriminators were fed a history of the 50 most recent generated images.

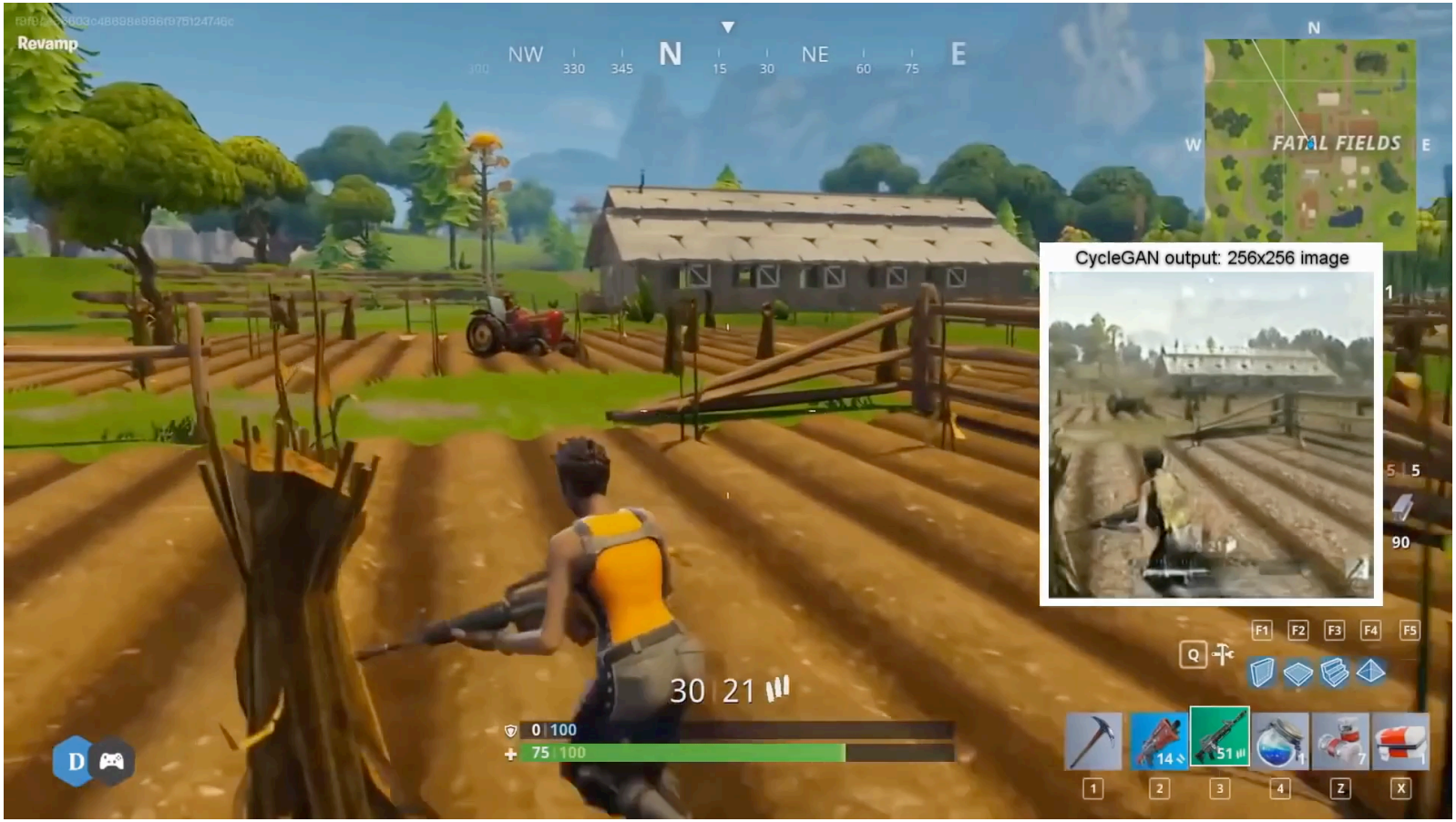
CycleGAN results. Style transfer



CycleGAN results. Transfiguration



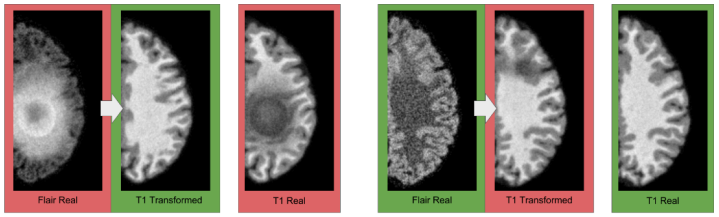
CycleGAN results. Fortnite to PUBG



https://www.youtube.com/watch?v=xkLtgwWxrec

CycleGAN takeaways

- State of the art of unpaired image-to-image translation for a couple of years
- Cycle loss to mitigate mode collapse
- Descendants & Similar approaches:
  - BiCycleGAN: Multimodal Image-to-Image Translation, https://arxiv.org/pdf/1711.11586
  - CyCADA: Cycle-Consistent Adversarial Domain Adaptation, https://arxiv.org/pdf/1711.03213
  - DiscoGAN: Discover Cross-Domain Relations with Generative Adversarial Networks
- Work nicely on tasks with color or texture changes (e.g., day-to-night photo translations, or photo-to-painting). However, tasks that require substantial geometric changes, such as cat-to-human, usually fail.
- Word of caution: as any GAN-based method, it fundamentally hallucinates on the content it should be thoroughly studied before deployment.



(a) A translation removing tumors (b) A translation adding tumors

https://arxiv.org/pdf/1805.08841.pdf

Super resolution

Super Resolution GAN (SRGAN)

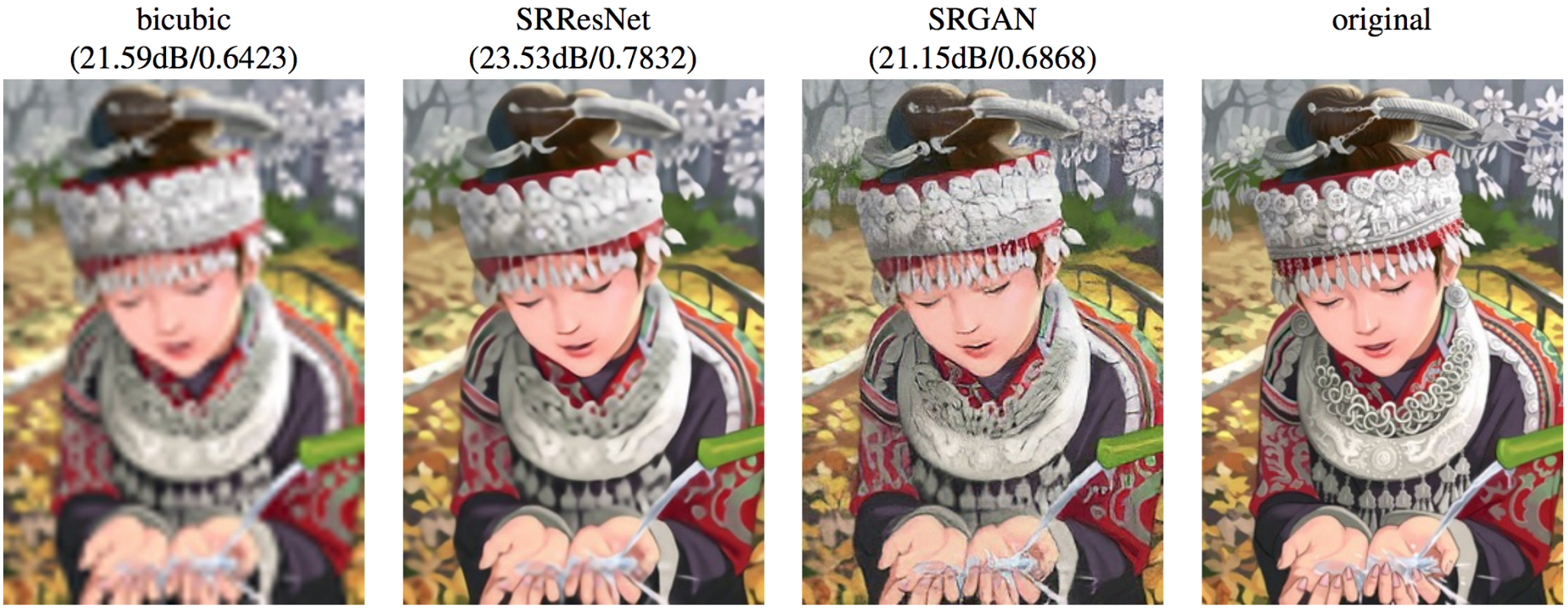
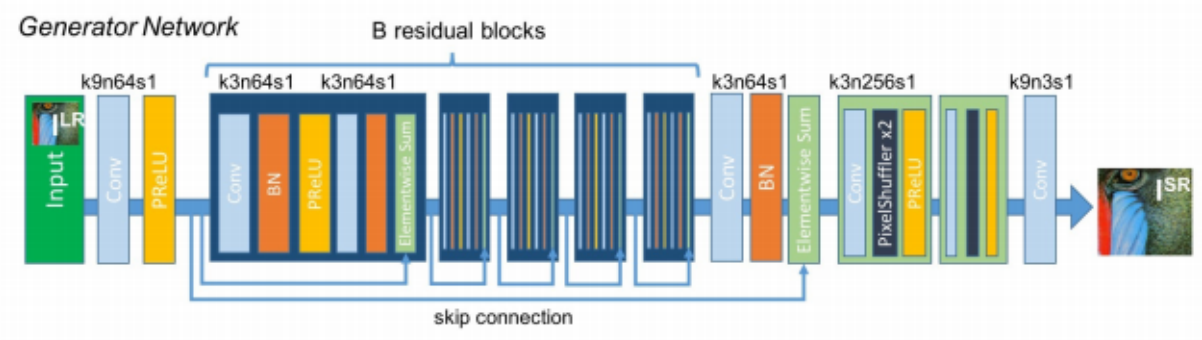


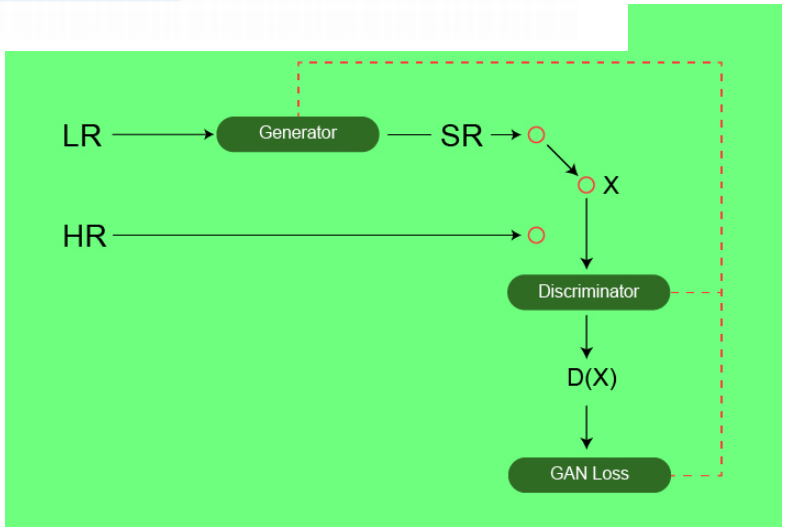
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

https://arxiv.org/pdf/1609.04802.pdf

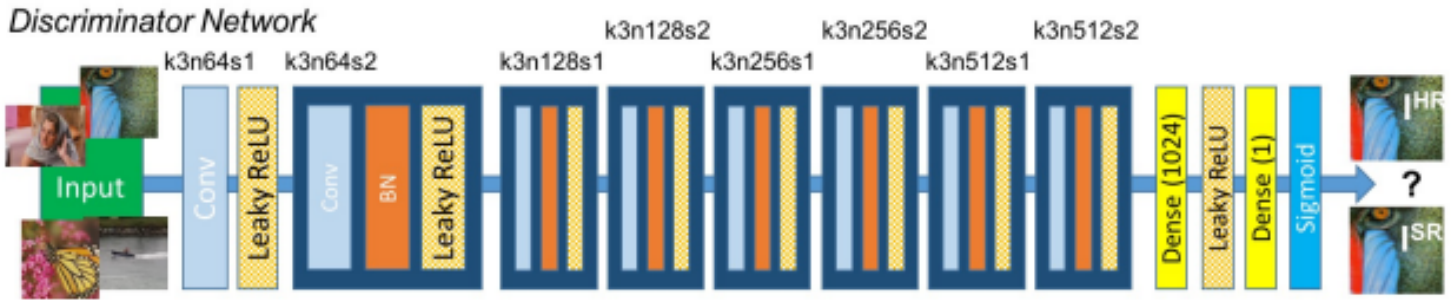
SRGAN. Generator



- Parametric ReLU as an activation function



SRGAN. Discriminator



- Discriminator architecture used in this paper is like DCGAN (deep convolution) architecture with LeakyReLU as activation:
  - eight convolutional layers with of 3×3 filter kernels,
  - increasing by a factor of 2 from 64 to 512 kernels.
  - strided convolutions are used to reduce the image resolution each time the number of features is doubled.

SRGAN. Loss function

- Perceptual loss:
  - Content loss

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

  - Adversarial loss

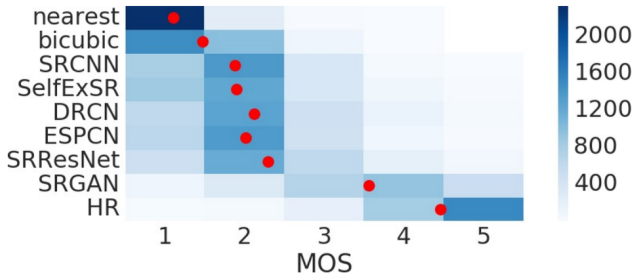
$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

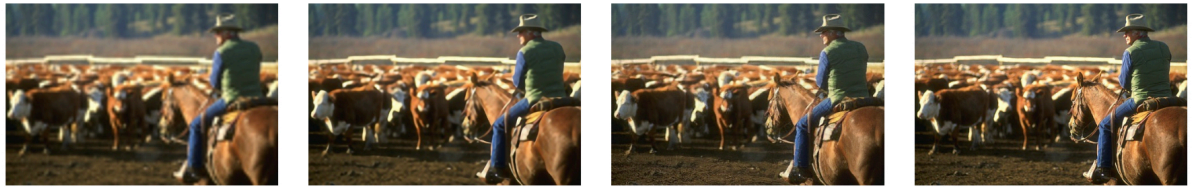
  - Overall:

$$l^{SR} = \underbrace{\underbrace{l^{SR}_{\text{X}}}_{\text{content loss}} + \underbrace{10^{-3} l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

SRGAN results

	SRResNet-		SRGAN-		
Set5	MSE	VGG22	MSE	VGG22	VGG54
PSNR	32.05	30.51	30.64	29.84	29.40
SSIM	0.9019	0.8803	0.8701	0.8468	0.8472
MOS	3.37	3.46	3.77	3.78	3.58
Set14					
PSNR	28.49	27.19	26.92	26.44	26.02
SSIM	0.8184	0.7807	0.7611	0.7518	0.7397
MOS	2.98	3.15*	3.43	3.57	3.72*



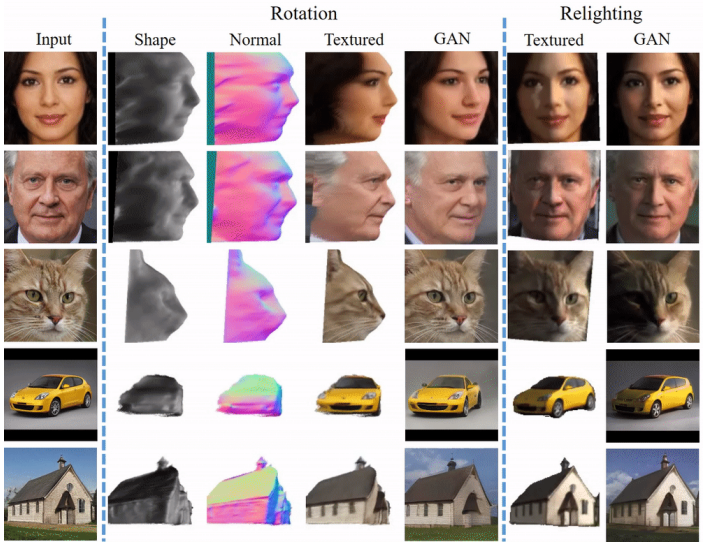


- State of the art 4x upsampling
- Perceptual loss
- SRGAN was able to generate state-of-the-art results which the author validated with extensive Mean Opinion Score (MOS) test on three public benchmark datasets

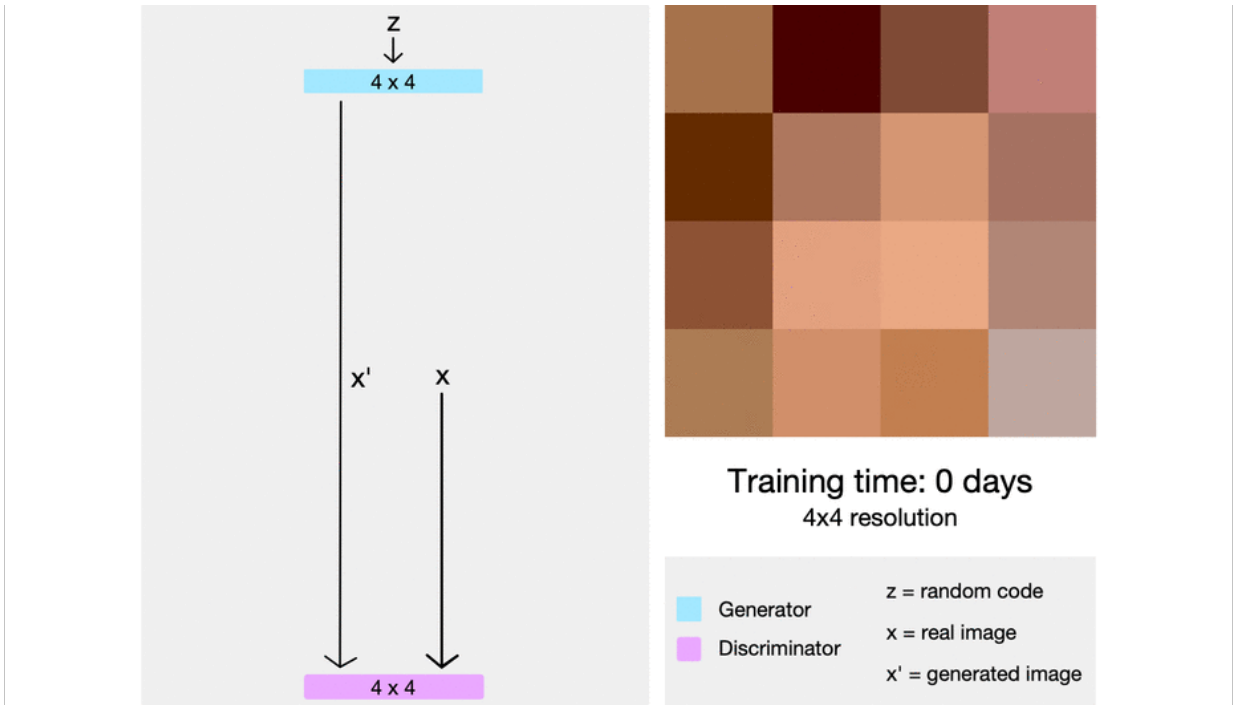
Latent space exploration

GAN latent space exploration

- Motivation: since there is a space that joins all the images, are there any interesting properties that we can explore or exploit?
  - Transition from one image to another image
  - Change visual condition of a single image
  - Learn 3D representation by 2D projection



Step 1. Progressive learning



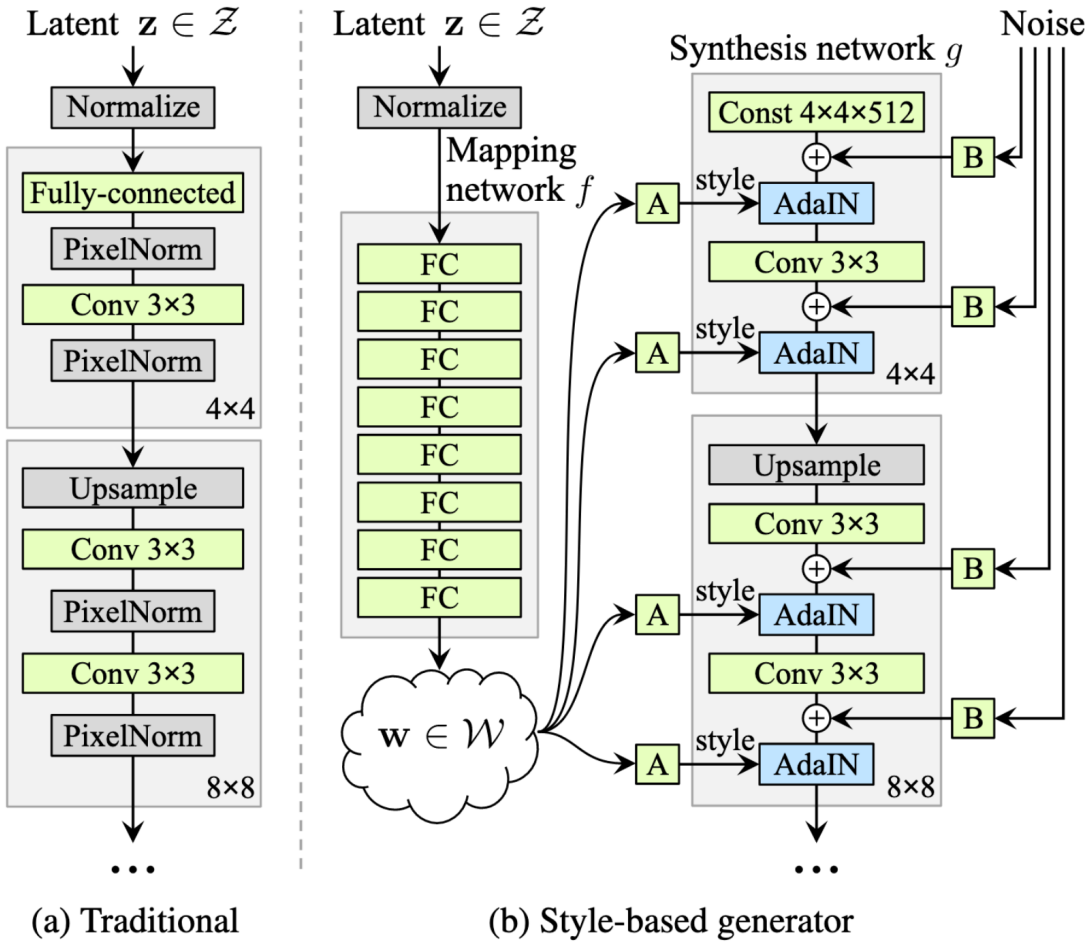
- ProGAN – based on WassersteinGAN

https://bit.ly/2Sphs5M

Step 2. Style disentanglement

- Latent space dis-entanglement – factorized feature representation by different dimensions in the latent space
- StyelGAN builds on the idea of WGAN, ProGAN, progressively grows its images but addresses these problems by first disentangling the latent space and then then using that improved latent space to inform the growing of the image step-by-step

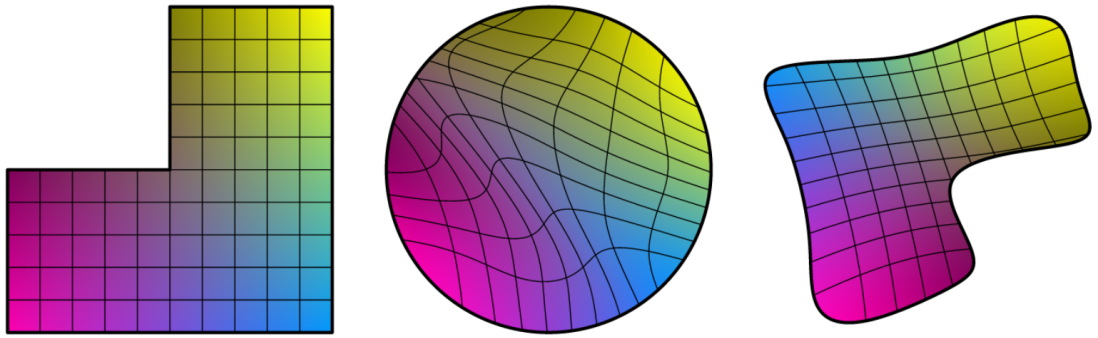
ProGAN vs StyleGAN



(a) Traditional (b) Style-based generator

https://arxiv.org/pdf/1812.04948.pdf

Mapping space intuition



(a) Distribution of features in training set

(b) Mapping from $\mathcal{Z}$ to features

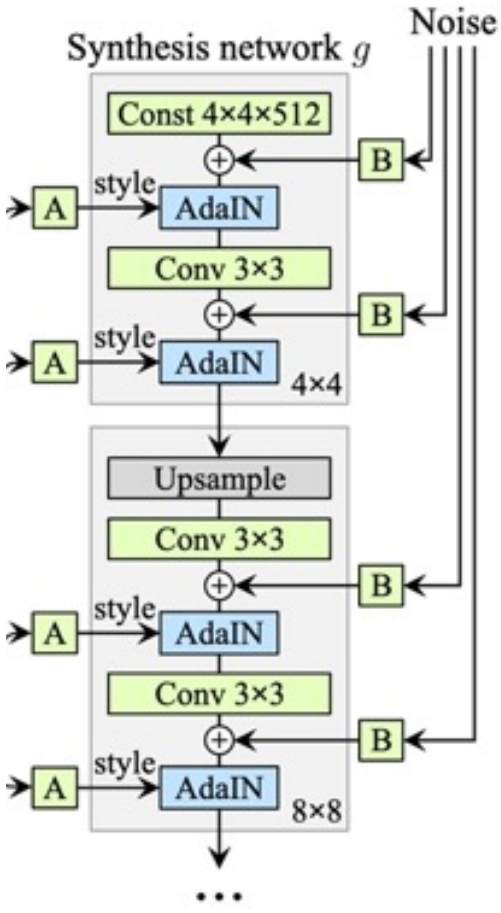
(c) Mapping from $\mathcal{W}$ to features

- For example, the dataset may be lacking men with long hair. So, at this point in the data distribution, we have a gap. Without W, we would now have a warped latent space, as the network learns that men apparently don't have long hair. Going from Z to W allows you to 'unwarp' space such that it can still allow generating men with long hair, even when they are absent from the dataset.

https://arxiv.org/pdf/1812.04948.pdf

StyleGAN Synthesis network

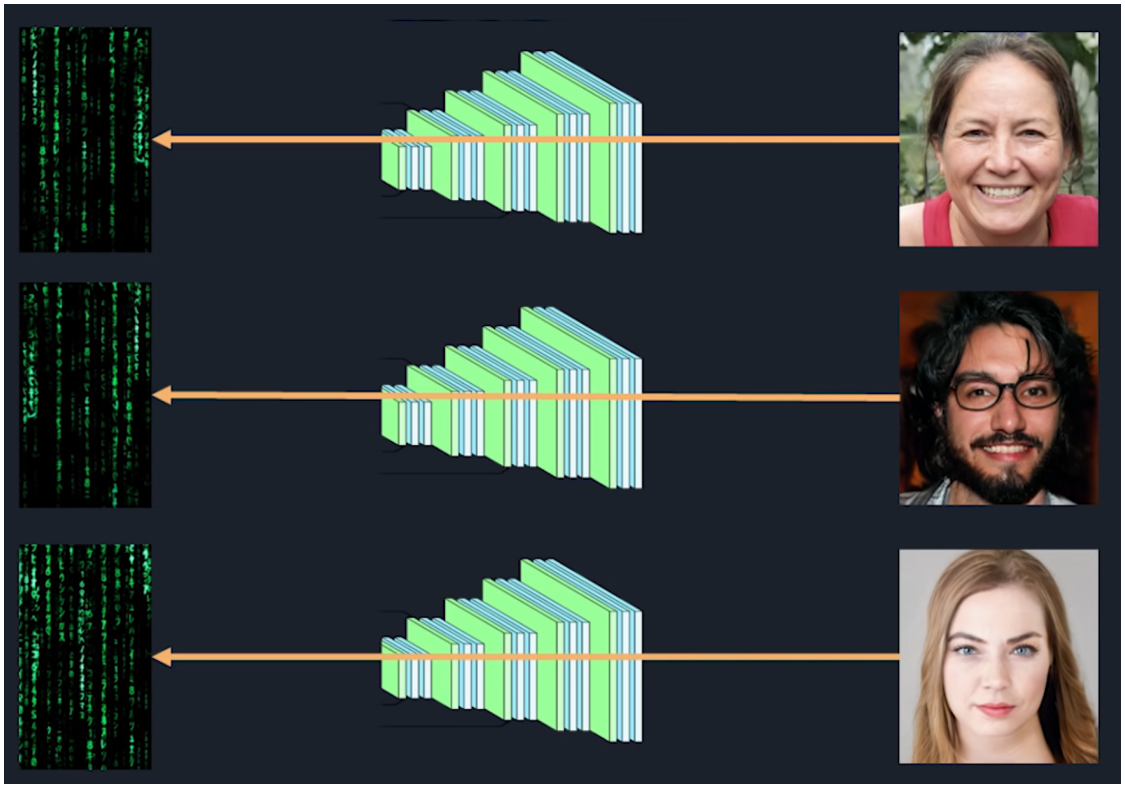
- Blending layers (AdaIN)
  - Plug in *w* into the corresponding image resolution
  - using style modules (AdaIN)
  - affects the features of that resolution of the image generation process one level at a time
- Level grouping:
  - coarse, up to 8×8, affects general pose, hair style: the base features;
  - middle, resolution of 16×16 to 32×32, affects finer facial features, eyes open/closed, etc; and
  - fine, resolution of 64×64+, affects colors and fine features.
- Noise add stochastic features (freckles, hair, small marks, etc.)



Style levels

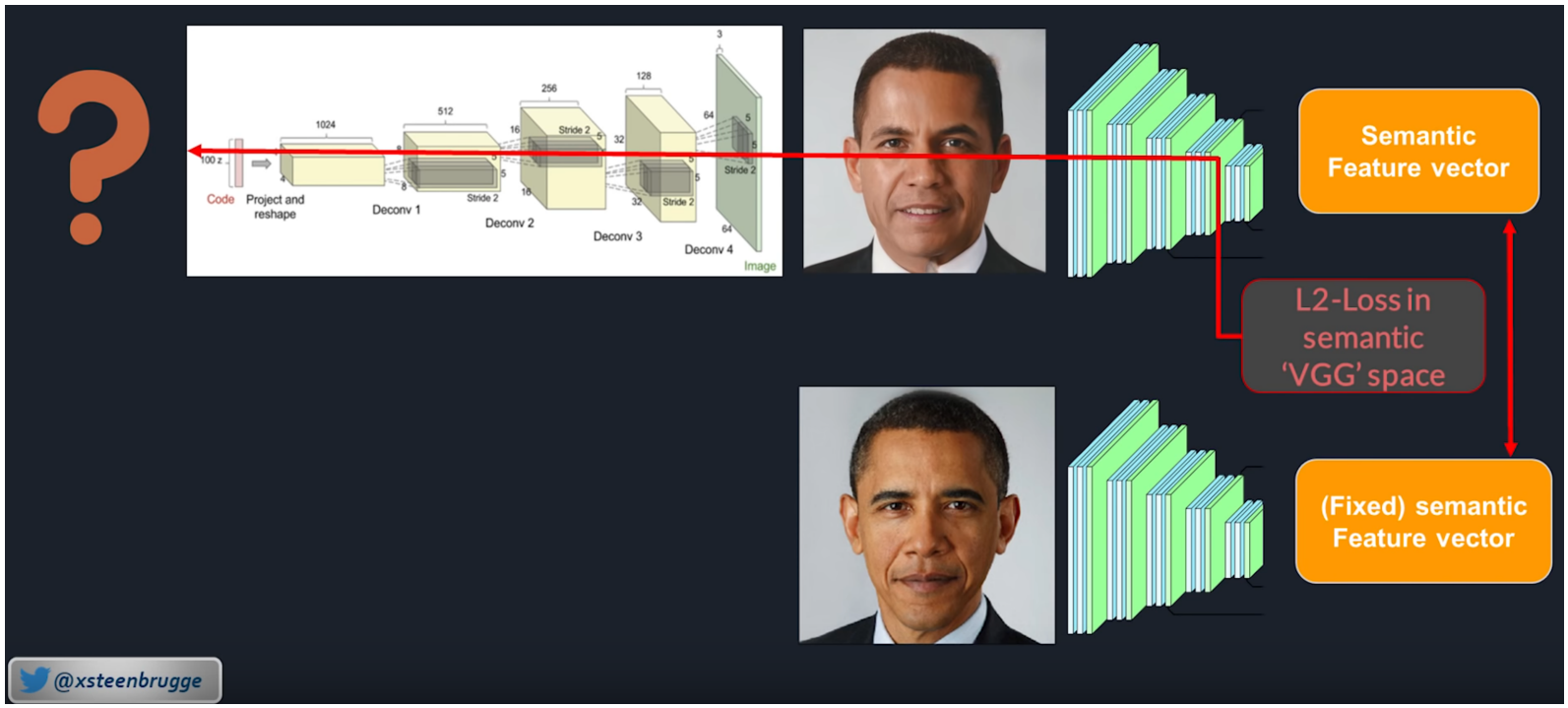
https://arxiv.org/abs/1812.04948

Face space



https://bit.ly/3hUeEIG

Face space + perceptual loss



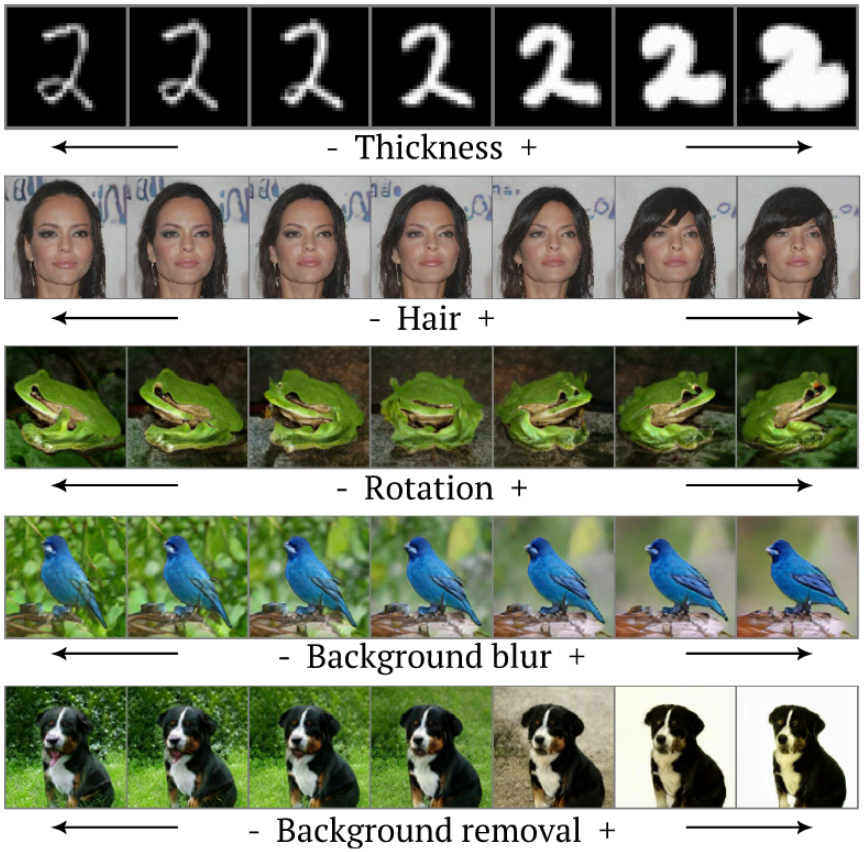
Face space walk

https://bit.ly/3ufjBhL

https://bit.ly/3bTHntw

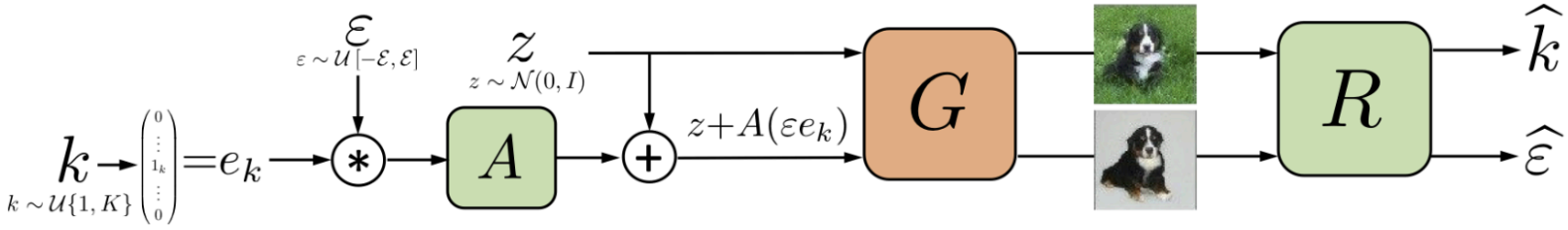
Discovery of meaningful directions

- Generation of new samples (data augmentation)
- Exploration of missing pieces for the dataset
- Saliency detection
- Not limited to computer vision!



https://arxiv.org/pdf/2002.03754.pdf

Direction search protocol



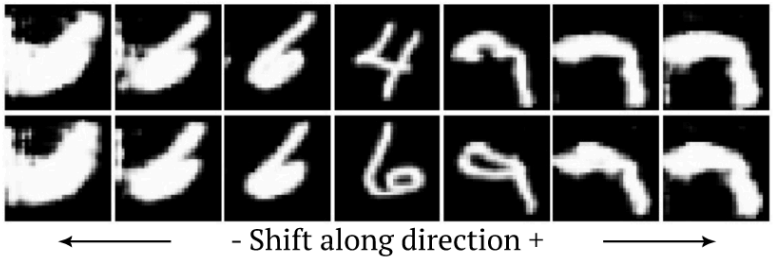
- Given: pre-trained generator G, that maps from Z to R^d, K – number of directions in the latent space we want to discover
- Matrix A ∈ $R^d \times K$
- Sample a random vector z and
  - z_1 which is z randomly shifted along a random direction from A by ε , i.e. z + A(εe_k)
- Reconstructor R, maps an image pair (G(z), G(z_1)), to $(\hat{k}, \hat{\varepsilon})$:
  - $\hat{k}$- direction index,
  - $\hat{\varepsilon}$ - prediction of the shift magnitude.

Search optimization

- Learning is performed via minimizing the following loss function:

$$\min_{A,R} \mathop{\mathbb{E}}_{z,k,\varepsilon} L(A,R) = \min_{A,R} \mathop{\mathbb{E}}_{z,k,\varepsilon} \left[L_{cl}(k,\widehat{k}) + \lambda L_r(\varepsilon,\widehat{\varepsilon}) \right]$$

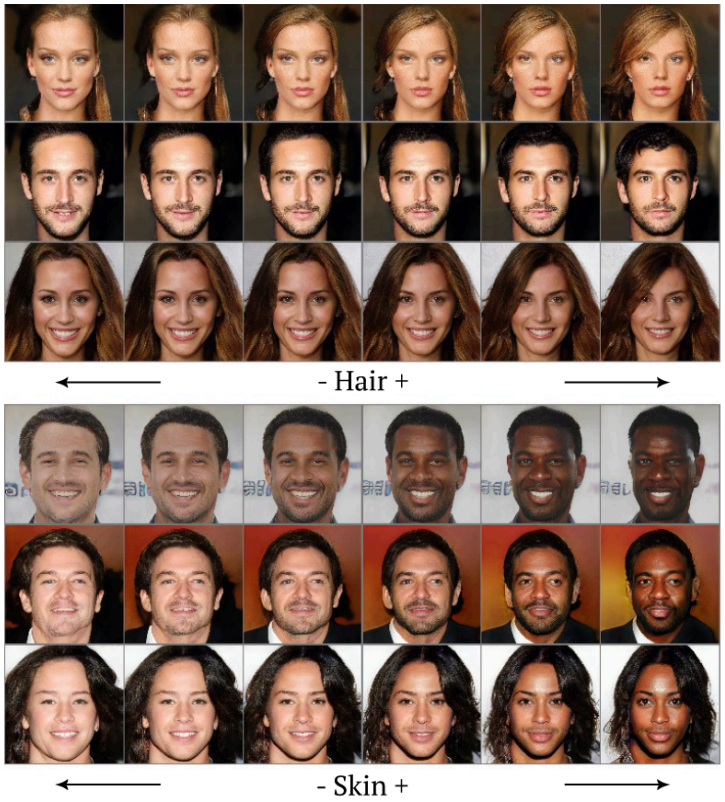
- $L_{cl}(.,.)$ – is a cross-entropy classification term
- L_r – is a mean absolute error regression term



- Regression term mitigates mode collapse like on the figure above

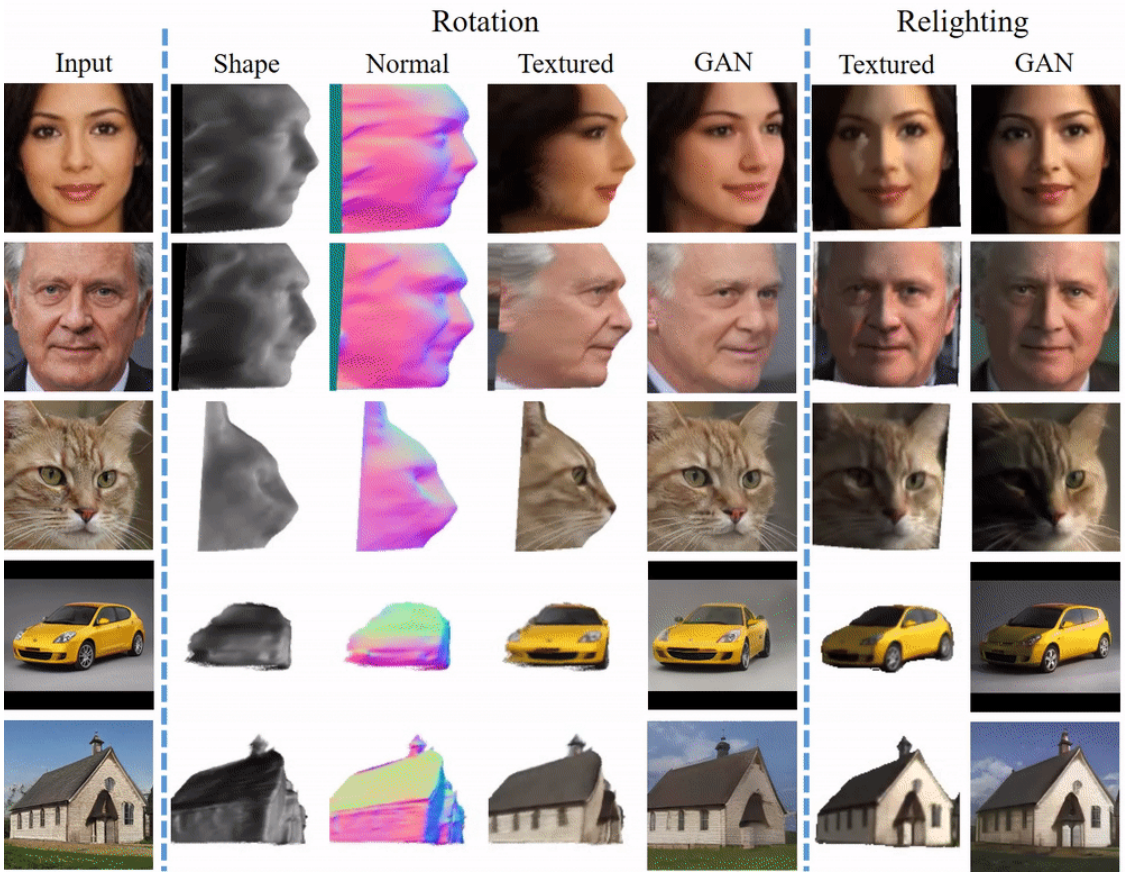
Results

- Examples of directions discovered for ProGAN and CelebA dataset



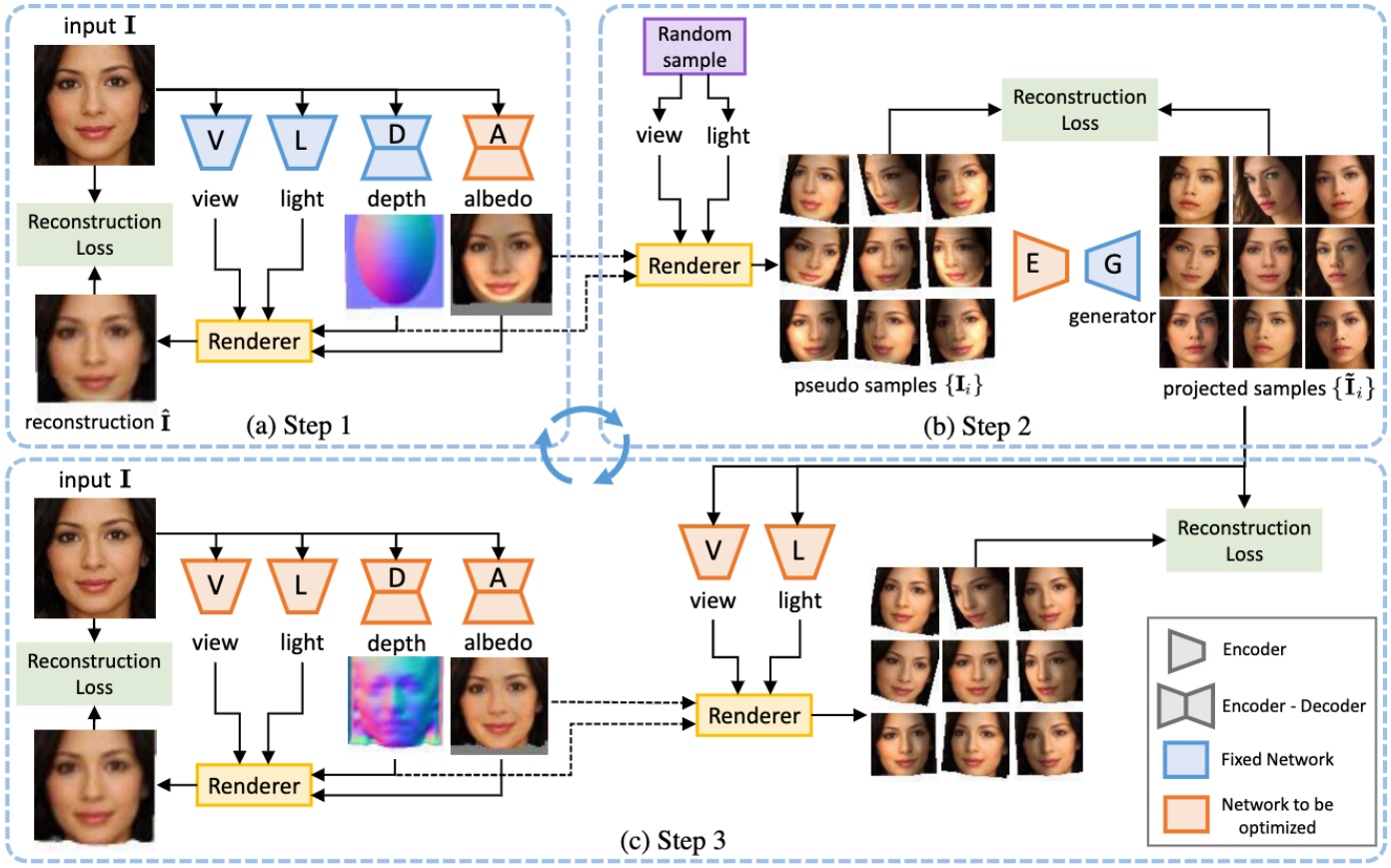
- Segmentation masks for BigGAN

From 2D to 3D



https://github.com/XingangPan/GAN2Shape

GAN2Shape method overview



https://openreview.net/pdf?id=FGqiDsBUKL0

In this video

- Wasserstein GANs to mitigate vanishing gradients and mode collapse issues
- Generative adversarial networks architectures for image-to-image translation:
  - Pix2Pix, CycleGAN, SRGAN, ProGAN, StyleGAN, GAN2Shape
- Ideas:
  - PatchGAN, cycle consistency loss, perceptual loss, latent space dimension disentanglement, style modules, latent space exploration
- Tasks:
  - Image to image translation
    - Labeling, Style transfer, In-painting, Super-resolution
  - Data augmentation
  - 3D model reconstruction

https://github.com/eriklindernoren/PyTorch-GAN

Thank you!


austyuzhanin@hse.ru



anaderiRu


hse_lambda

Andrey Ustyuzhanin

WGAN: spectral normalization

› Spectral normalisation proposes to use normalised weights:

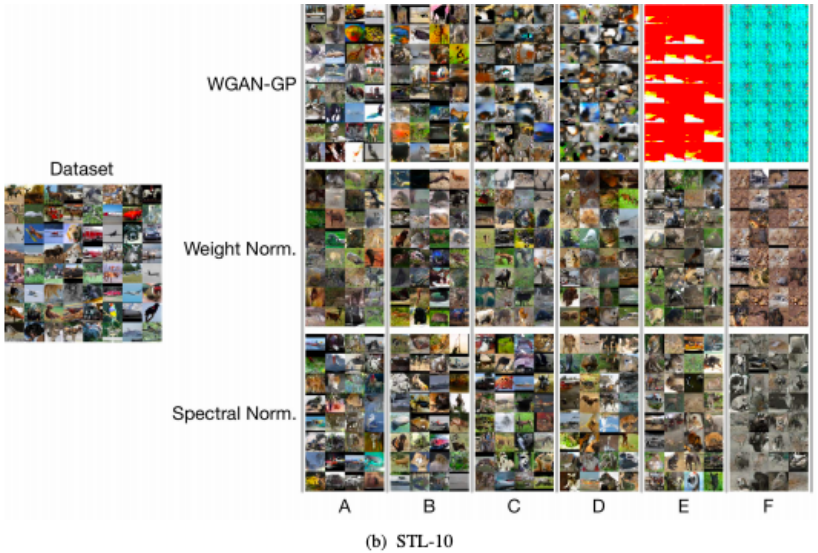
$$W_{SN} = \frac{W}{\sigma(W)}$$

where:

$$\sigma(W) = \max_{h:h\neq 0} \frac{||Wh||_2}{||h||_2}$$

› this gives constraints on gradient:

$$||f||_{Lip} \leq \prod_{i=1}^{l} \sigma(W_l).$$



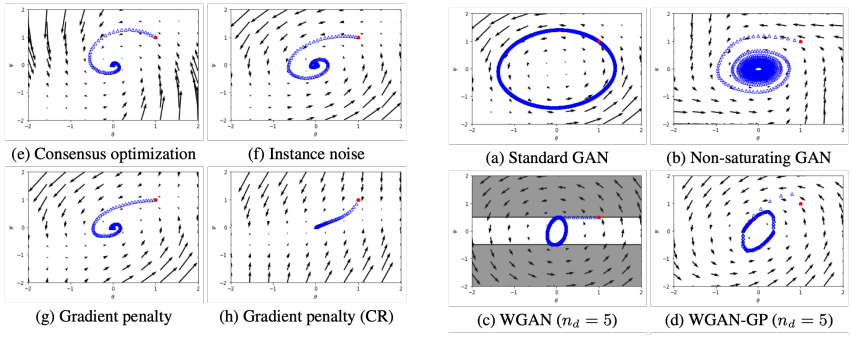
(b) STL-10

Figure 6: Generated images on different methods: WGAN-GP, weight normalization, and spectral normalization on CIFAR-10 and STL-10.

Miyato et al. Spectral Normalization for Generative Adversarial Networks, ICLR 2(

Zero-centered gradient penalty

target distribution: $p_{\text{data}} = \delta(x)$
generator distribution: $p_{gen} = \delta(x - \theta)$
discriminator: $D_\psi(x) = \psi \cdot x$



Proposals:

- Venerable instance noise
- Zero-based gradient penalty without interpolation:

$$\frac{\gamma}{2}\mathbb{E}_{x\sim p_{\text{data}}}\|\nabla D(x)\|^2$$

https://arxiv.org/abs/1801.04406 analyzes GAN training behavior on a very simple system: