Deep Neural Networks (Głębokie Sieci Neuronowe)

Module 2: Generative Adversarial Networks

Krzysztof Krawiec

Wydział Informatyki i Telekomunikacji
Politechnika Poznańska
2019/2020
http://www.cs.put.poznan.pl/kkrawiec/

Outline

1. Generative Adversarial Networks
2. Pix2pix (PatchGAN)
3. CycleGAN
4. Deep Convolutional GAN (DCGAN)
5. Wasserstein GAN

Generative Adversarial Networks

Generative Adversarial Nets

Image-to-Image Translation with Conditional Adversarial Networks

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

http://arxiv.org/abs/1406.2661

Discrimination vs. generation

- Discriminative model (e.g., classification, regression): learns a mapping from an input space X to some output space Y.

- Generative model: learns how to generate samples from a (possibly complex) probability distribution.
  Examples:
  - Deep Belief Networks,
  - Boltzmann machines,
  - Deep Boltzmann machines,

- Key question: How to evaluate the adequacy of the samples generated by a generative model?

Adversarial setting

The goal: learn the generator's distribution pg over some data x.

Approach:

- Define a prior on input noise variables p(z),
- Represent a mapping to data space as G(z), where G is a differentiable function (parameterized with some parameters).
- Define D(x) that outputs a single scalar that represents the probability that x came from the data rather than from G.
- Train simultaneously:
  - D to maximize the probability of assigning the correct label to both training examples and samples from G.
  - G to minimize log(1 − D(G(z))):

Adversarial setting

Loss function defines a two-player minimax game with value function:

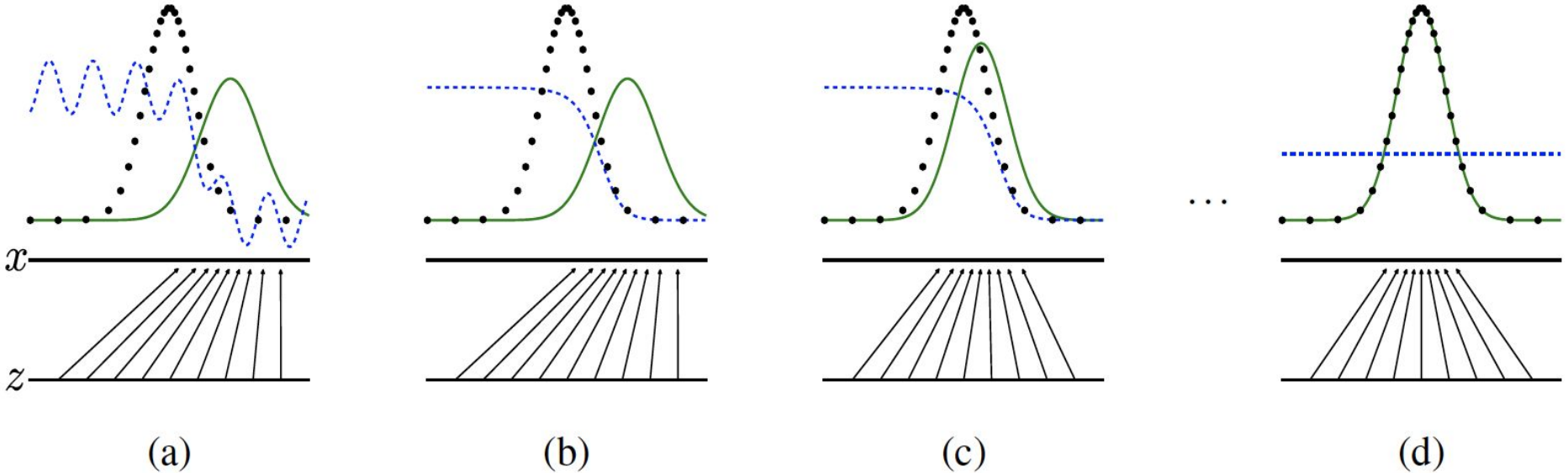
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Simultaneously rewarding:

- D for maximizing its output for true data,
- G for causing D to fail to discriminate between the true data and the ‘fake’ examples it generates.

Illustration (1D)

- blue, dashed line: discriminative distribution
- black, dotted line: data generating distribution
- green, solid line: generative distribution



Technical realization

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

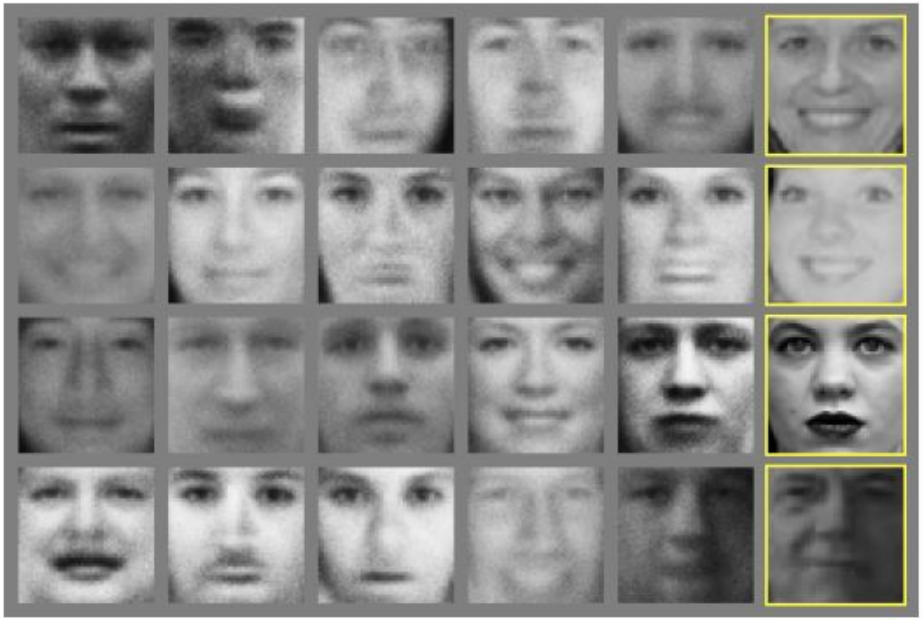
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

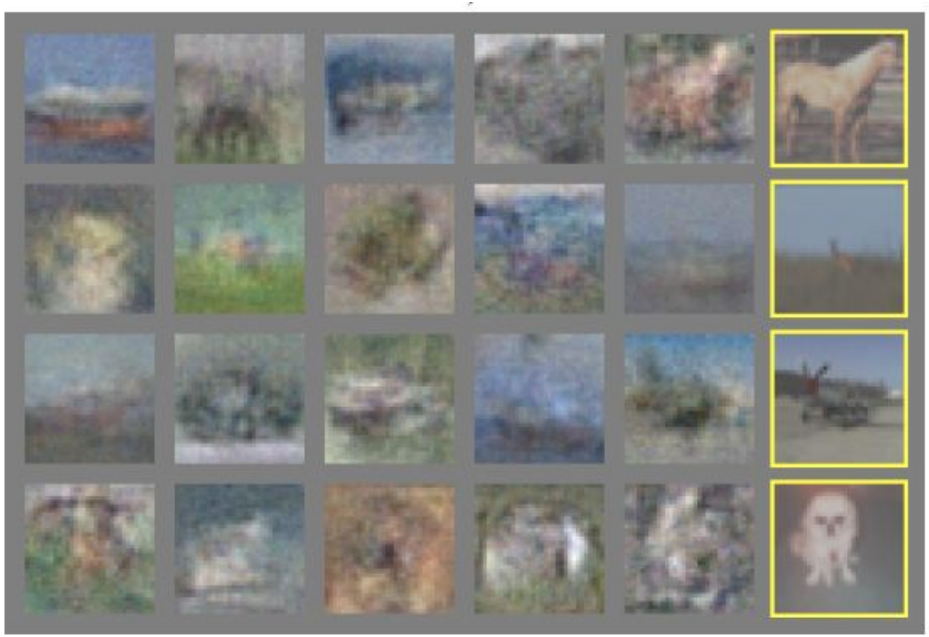
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Visualization (MNIST, TFD)

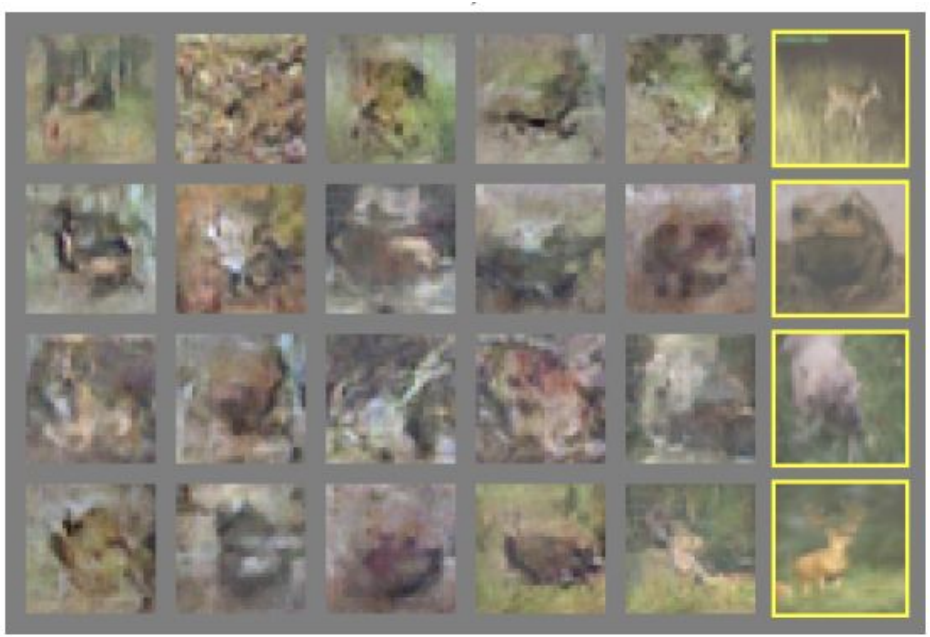


- Truly random sample, not cherry-picked.
- Rightmost column shows the nearest training example of the neighboring sample

Visualizations (CIFAR-10)



c)



d)

Fully connected (left) vs. convolutional (right)

Comments

- Notice the relationship to autoencoders.
- The paper contains also theoretical part - proofs of convergence.

- The space of z values may have meaningful interpretation:
  E.g.: Digits obtained by linearly interpolating between coordinates in z space of the full model:

Conceptual confrontation with other approaches

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Nearly all models incur extreme difficulty	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted

Pix2pix (PatchGAN)

Image-to-Image Translation with Conditional Adversarial Networks

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

https://arxiv.org/abs/1611.07004

The role of loss

CNNs learn to minimize a loss function – an objective that scores the quality of results – and although the learning process is automatic, a lot of manual effort still goes into designing effective losses. In other words, we still have to tell the CNN what we wish it to minimize. But, just like King Midas, we must be careful what we wish for!

- If we take a naive approach and ask the CNN to minimize the Euclidean distance between predicted and ground truth pixels, it will tend to produce blurry results. This is because Euclidean distance is minimized by averaging all plausible outputs, which causes blurring.
- Q: How to make the output indistinguishable from reality?
- A: Generative Adversarial Networks

Conditional GANs

- GANs learn a generative model of data
- Conditional GANs (cGANs) learn a conditional generative model.

GAN:

$$G : z \rightarrow y$$

where: y - prediction (image), z - random noise vector

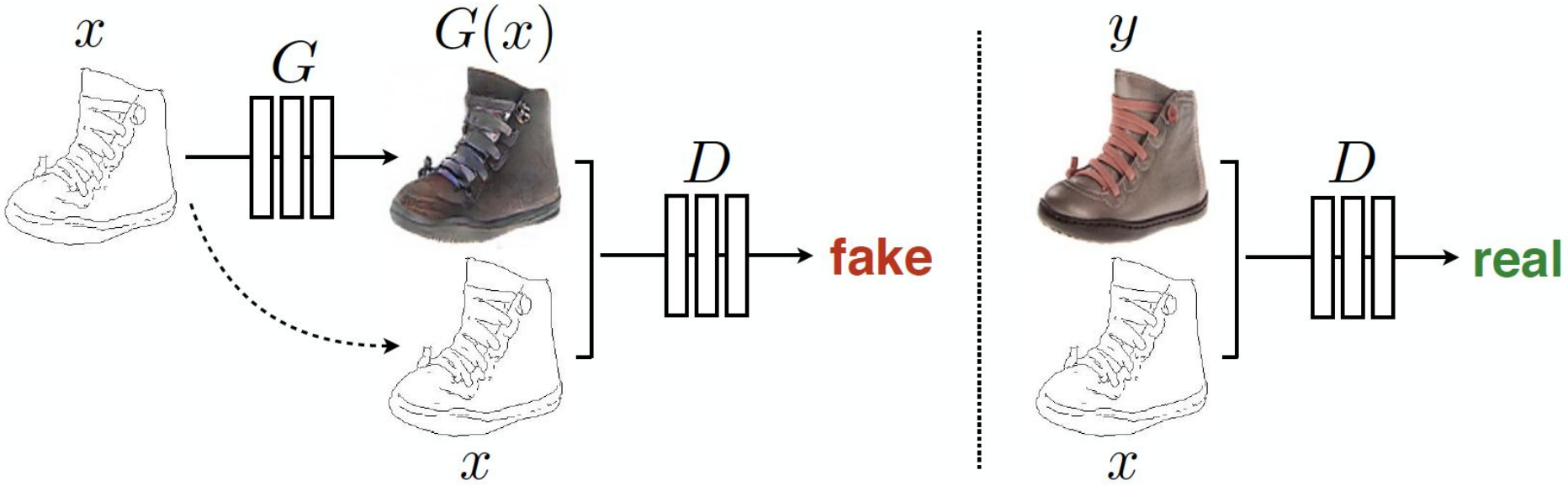
Conditional GAN:

$$G : \{x, z\} \rightarrow y$$

where x - input (e.g., input image)

Discriminator: D: (x,y) -> {0,1} (fake vs. real)

Training protocol



Definition of loss

Loss:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

In practice, combining with the distance from the true image helps:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

(L2 caused more blurring)

The overall loss/objective:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Regular GAN

Control baseline (regular GAN):

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z))]$$

Why GANs? Why not X -> Y?

Without z, the net could still learn a mapping from x to y, but would produce deterministic outputs, and therefore fail to match any distribution other than a delta function.

- A bit stretched, but conveys the essence of the generative approach: noise forces the learning process to form robust intermediate representations.

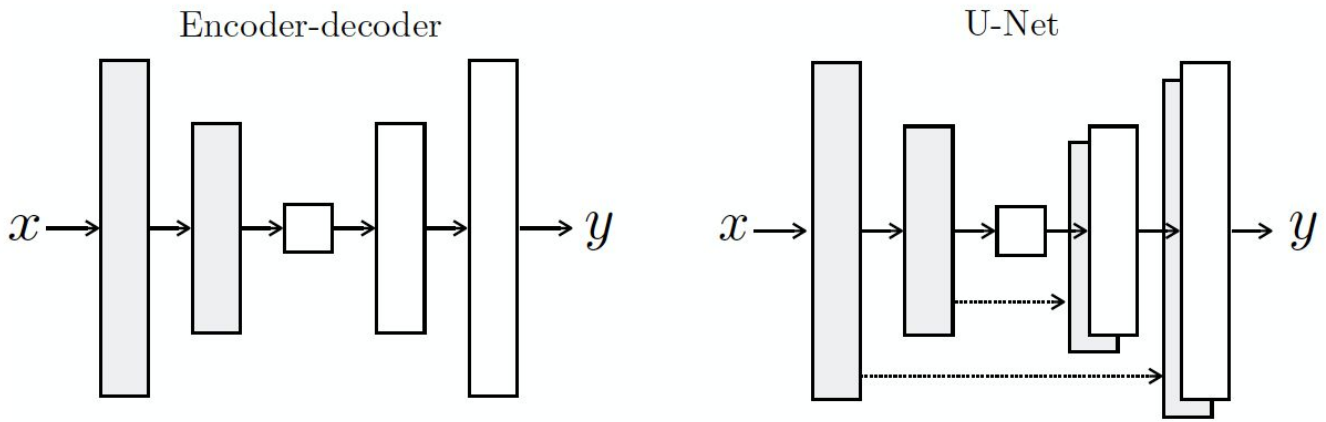
- Original GANs used Gaussian noise.
  - This did not prove effective.
- Pix2pix adds noise via dropout applied to G.
  - Also at test time!

The architecture

Basic building block for both G and D:

- convolution
- batch normalization
- ReLU

Two considered architectures for G (U-Net used in the experiments):



Motivation: desired spatial correspondence between x and y.

Discriminator

- Because L1 causes blurring, it already takes care of the low frequencies (overall ‘rough’ structure of the image).
- Therefore, the design of D focuses on detail (high frequencies).
- Thus, it’s enough for D to work with small image patches.
- D takes an NxN patch of y and decides whether it’s real or fake.
  - More specifically: whether it comes from a real image or fake image.
- Observations:
  - D is thus a ‘textural discriminator’ (and the associated loss can be considered texture loss).
  - D perceives the image as a Markov random field, with pixels more distant than patch’s diameter being considered as independent random variables.

Training

Training setup follows the original GAN paper and other earlier recommendations:

- One step gradient descent on D interwoven with one step of GD on G.
- D trained slower than G, by using G*/2 loss
- Rather than minimize
- $$\log(1 - D(x, G(x, z))$$
- maximize

$$\log D(x, G(x, z))$$

Experimental domains

Unidirectional (citation numbers from the original paper):

- Architectural labels→photo, trained on CMP Facades [45].
- BW→color photos, trained on [51].
- Edges→photo, trained on data from [65] and [60]; binary edges generated using the HED edge detector [58] plus postprocessing.
- Sketch→photo: tests edges→photo models on human-drawn sketches from [19].
- Day→night, trained on [33].
- Thermal→color photos, trained on data from [27].
- Photo with missing pixels→inpainted photo, trained on Paris StreetView from [14].

Bidirectional:

- Semantic labels↔photo, trained on the Cityscapes dataset [12].
- Map↔aerial photo, trained on data scraped from Google Maps.

Evaluation

Pixel-by-pixel distance (e.g., L1 or L2) too primitive to capture the quality of the produced output.

Therefore, other evaluation methods:

1. Amazon Mechanical Turk (AMT): human subjects score the plausibility
    a. The subject (Turker) is shown the pair of x and y and should point to the fake one
2. Testing whether an off-shelf recognition system can recognize objects in the synthesized images (‘inception score’)
    a. Used FCN-8s

Ablation experiments

Ablation experiments

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	**0.66**	**0.23**	**0.17**
Ground truth	0.80	0.26	0.21

Table 1: FCN-scores for different losses, evaluated on Cityscapes labels↔photos.

Ordinary GAN performs very bad because it ‘collapses’ into producing almost the same output for all inputs.

Observations

- Learns effectively even from small samples (e.g. 400 images in facade dataset) - in part because of patch-based perspective.
- Computationally efficient (training time in the order of hours on Titan X)
- Querying under 1 second
- Many more experiments covered in the paper:
  - impact of patch size: “From PixelGANs (1x1) to PatchGANs to ImageGANs (286x286)”
  - colorizing ‘hallucinations’, etc.
- G is a fully convolutional network, so it can be applied to images of arbitrary dimensions.

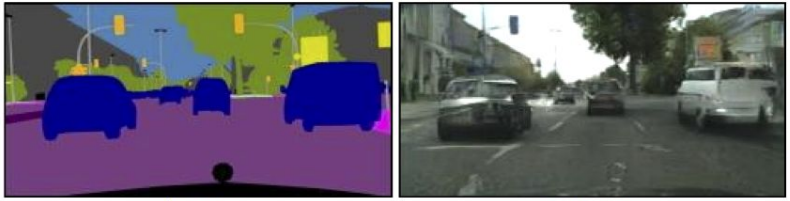
More general thoughts:

- In backpropagation, D ‘translates’ binary classification signal (fake/real) into a loss function.
- Adversarial setting = competitive coevolution.

Exploiting the FCN nature of G

G trained on 256x256, when tested on 512x512:

Labels to Street Scene

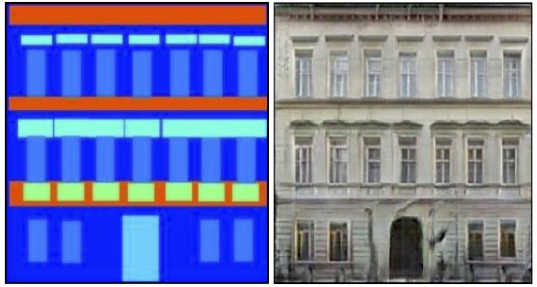


input output

Aerial to Map

input output

Labels to Facade



input output

Day to Night

input output

BW to Color

input output

Edges to Photo

input output

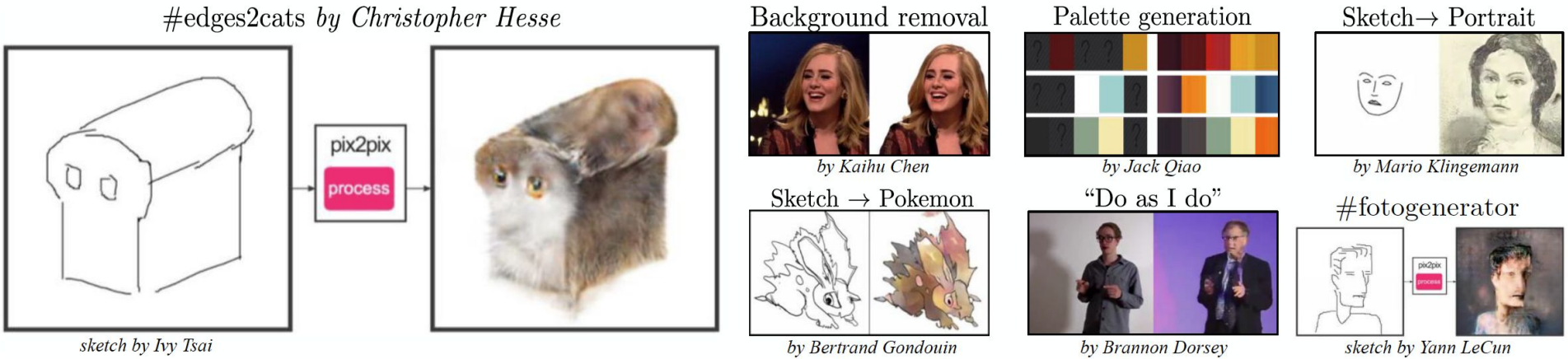


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin, *"Do As I Do" pose transfer* [2] by Brannon Dorsey, and *#fotogenerator* by Bosman et al. [4].

Human-drawn sketches to color images

Thermal images to RGB

Figure 12: *Learning to see: Gloomy Sunday*: An interactive artistic demo developed by Memo Akten [8] based on our `pix2pix` codebase. Please click the image to play the video in a browser.

https://vimeo.com/260612034

CycleGAN

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

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

http://arxiv.org/abs/1703.10593

The task of image-to-image translation

Train a mapping:

$$G : X \rightarrow Y$$

such that the output

$$\hat{y} = G(x)$$

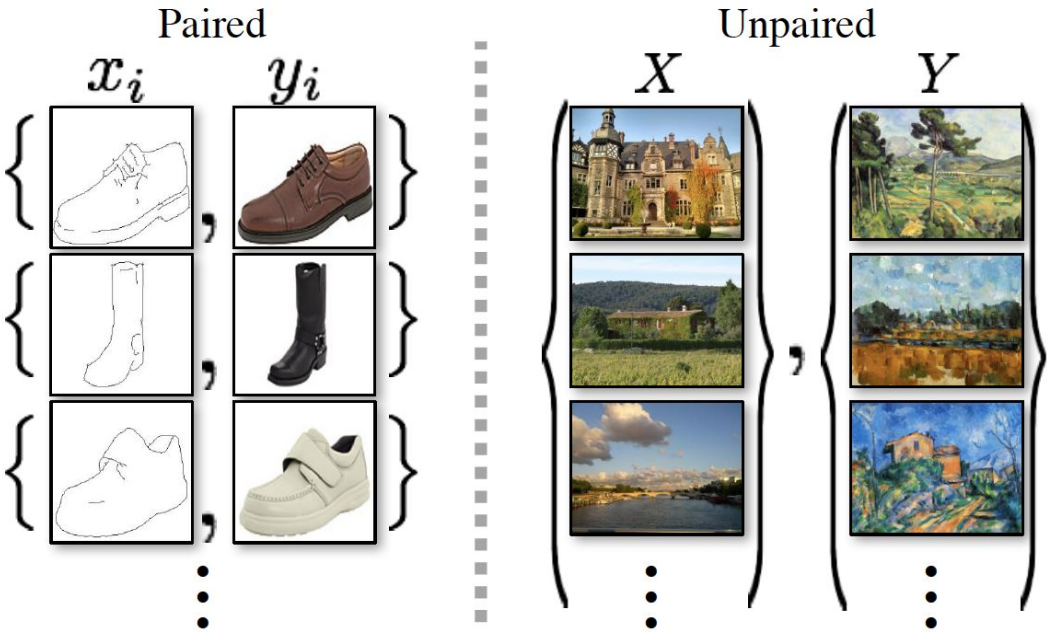
is indistinguishable from images y ∈ Y by an adversary D trained to classify y_hat apart from y.

Note:

- typical adversarial setting,
- in general applicable beyond images.

Motivation

- Paired examples for image-to-image translation tasks are hard to come by.
- However, it still make sense to talk about translation *between entire domains/sets*.



The challenges

Switching to sets is not enough:

1. No guarantee that an individual input x and output y are paired up in a meaningful way.
    - There are infinitely many mappings G that will induce the same distribution over y^.
    - In other words: a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution.
2. In practice, it is difficult to optimize the adversarial objective in isolation:
    - Problem of *mode collapse*: all input images map to the same output image; optimization fails to make progress.

Solution: Cycle consistency

Motivation: Translating a sentence from English to French, and then back from French to English, should result in the original sentence.

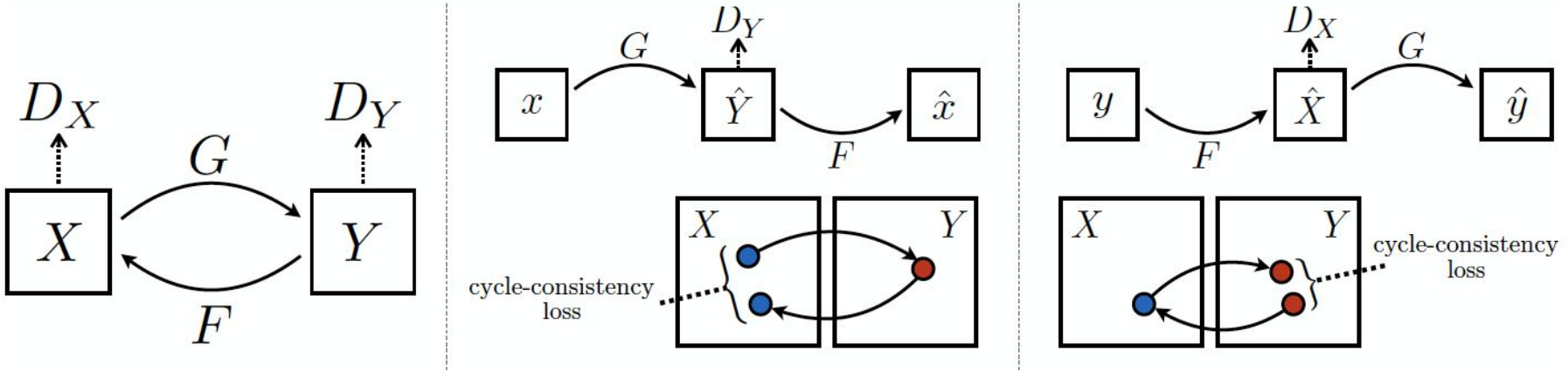
- Therefore: train two translators simultaneously:

$$G : X \rightarrow Y$$
$$F : Y \rightarrow X$$

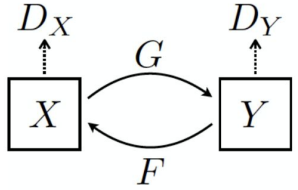
- and apply simultaneously *cycle consistency loss* that encourages:

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$$
$$y \rightarrow F(y) \rightarrow G(F(y)) \approx y$$

Cycle consistency: full model



Loss definition



Adversarial component for G (and analogously for F):

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

Cycle consistency loss:

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

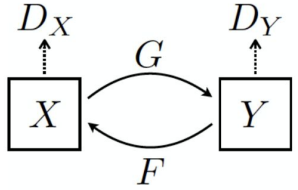
Total loss (λ=10):

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

The task: solve

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

Comments



Advantages:

- No pairing required.
- No need for a task-specific similarity function between input and output.
- The input and the output (x,y) do not have to lie in the same space.

Notice:

- This can be viewed as training two autoencoders in parallel:

$$F \circ G : X \to X$$

$$G \circ F : Y \to Y$$

- However, the intermediate representations are ‘enforced’ here.
- Also related to: adversarial autoencoders.

Architecture

Generators:

- stride-2 convolutions,
- residual blocks
- fractionally-strided convolutions
- 6 blocks for 128x128 images, 9 blocks for 256x256 images.

Discriminators:

- Patch-level
- 70 x 70 PatchGANs (i.e., Pix2pix, Image-to-Image Translation with Conditional Adversarial Networks, Isola et al.)

Training

1. Rather than using the original log-likelihood, uses least-square loss:
   - More stable in training.
   - Produces better results.

Technically: training G to minimize

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$$

and D to minimize

$$\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$$

2. Addressing “model oscillation”: updating discriminators using a history of generated images, rather than the ones produced by the latest generators.

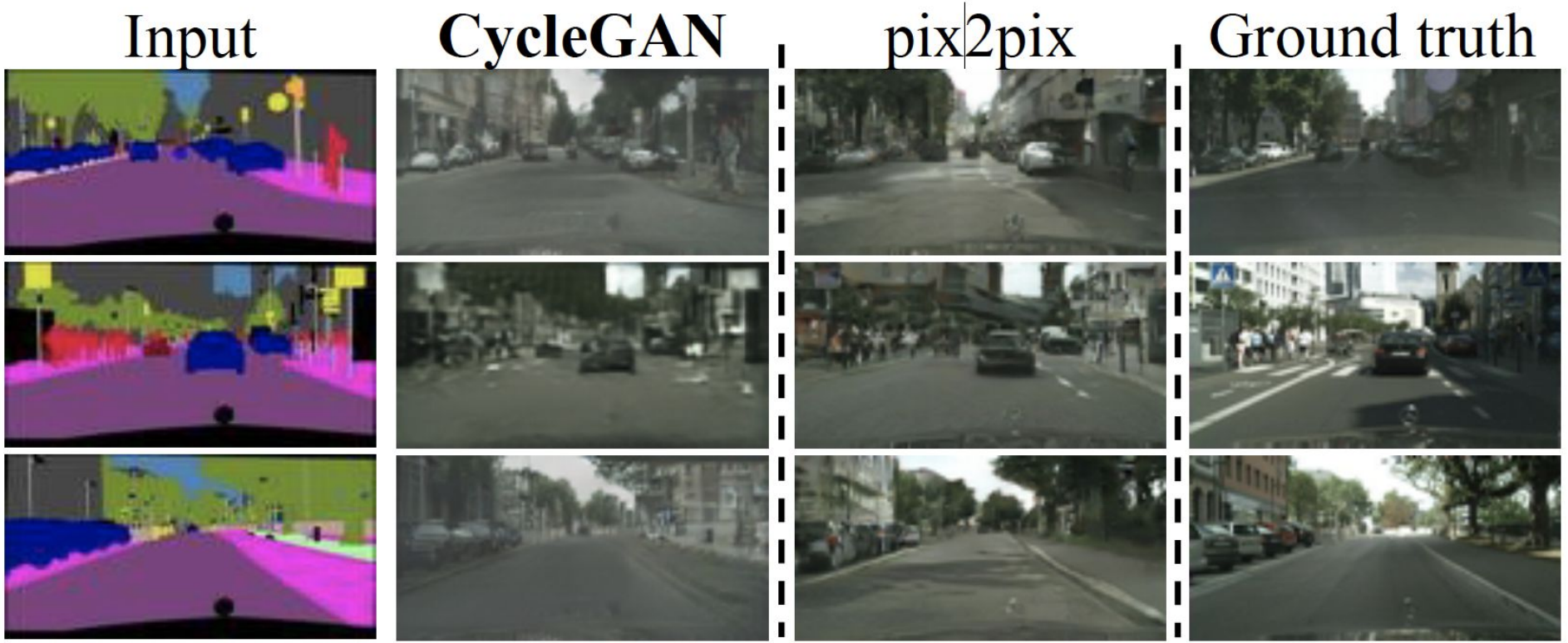
3. Adam algorithm, batch size = 1 (!)

Input x Output $G(x)$ Reconstruction $\mathrm{F}(G(x))$






Results



Notice: pix2pix uses paired examples!

Metrics

- Human assessment: Amazon Mechanical Turk (AMT) perceptual study:
  - Subject shown pairs of images (real, fake) an asked to click the image they thought was real.
- FCN score:
  - Applies an FCN (CNN) network to perform semantic segmentation of the scene.
  - Comparing the returned labels to ground truth labels (IoU?)

Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [32]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [9, 7]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [46]	0.7% ± 0.5%	2.6% ± 1.1%
Feature loss + GAN	1.2% ± 0.6%	0.3% ± 0.2%
CycleGAN (ours)	**26.8% ± 2.8%**	**23.2% ± 3.4%**

Table 1: AMT "real vs fake" test on maps↔aerial photos at 256×256 resolution.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.40	0.10	0.06
BiGAN/ALI [9, 7]	0.19	0.06	0.02
SimGAN [46]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [22]	0.71	0.25	0.18

Table 2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

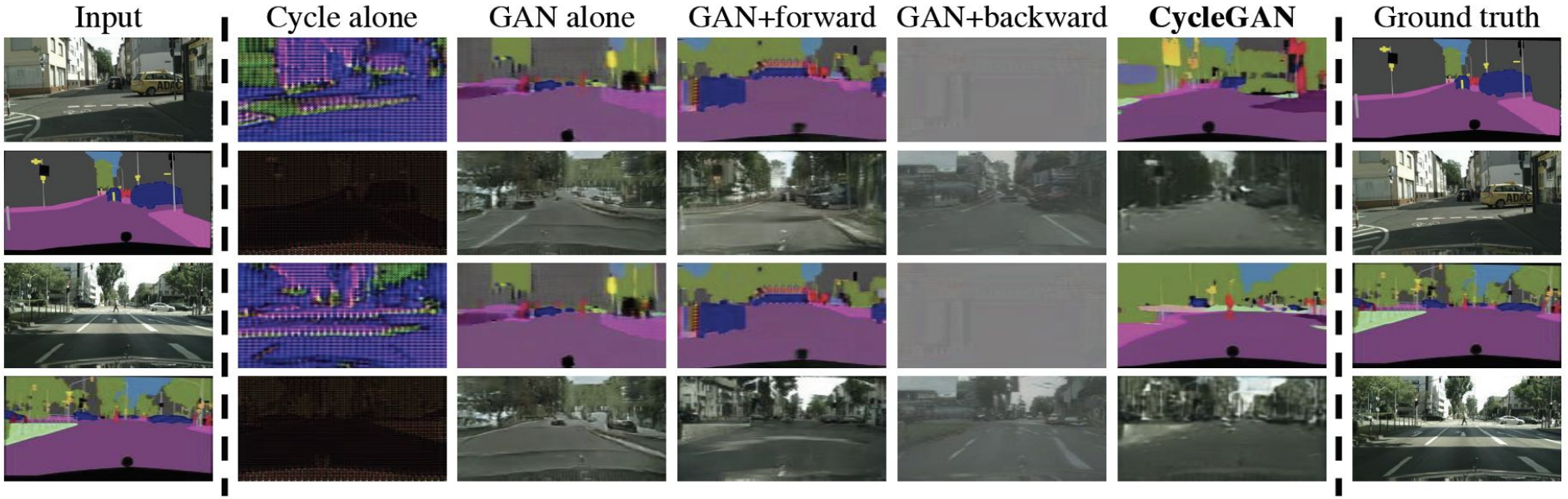


Figure 7: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (our full method), and ground truth. Both *Cycle alone* and *GAN + backward* fail to produce images similar to the target domain. *GAN alone* and *GAN + forward* suffer from mode collapse, producing identical label maps regardless of the input photo.


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

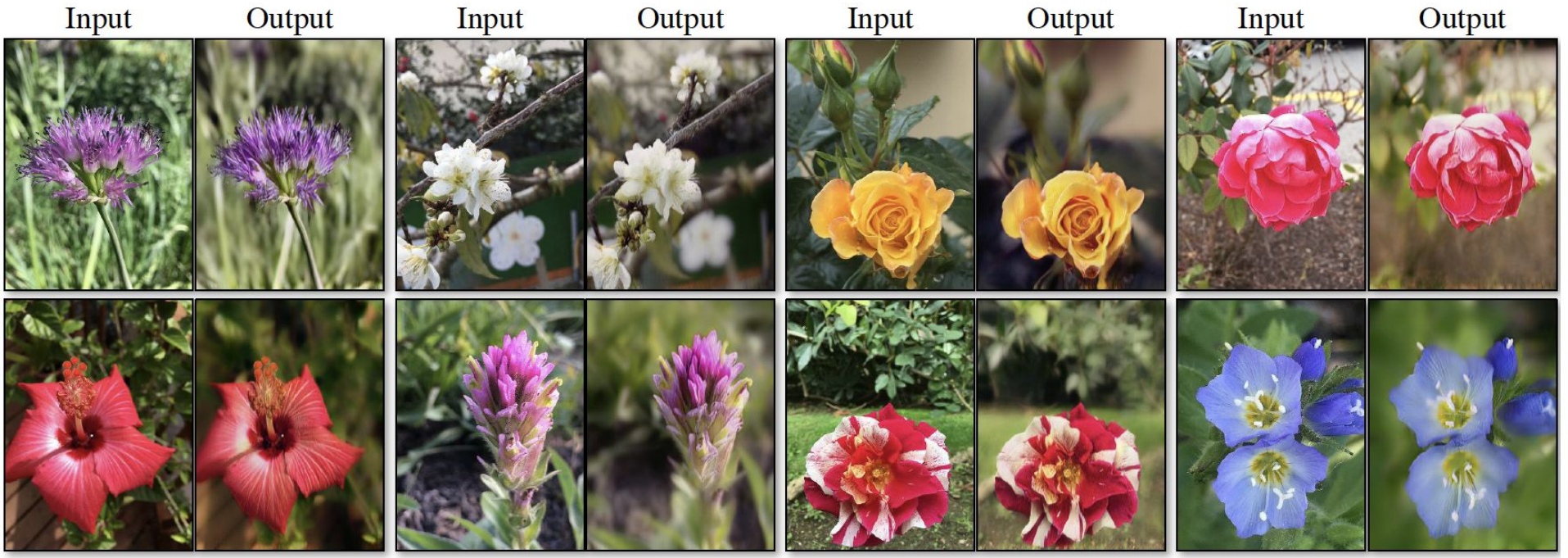


Figure 14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive and random examples.

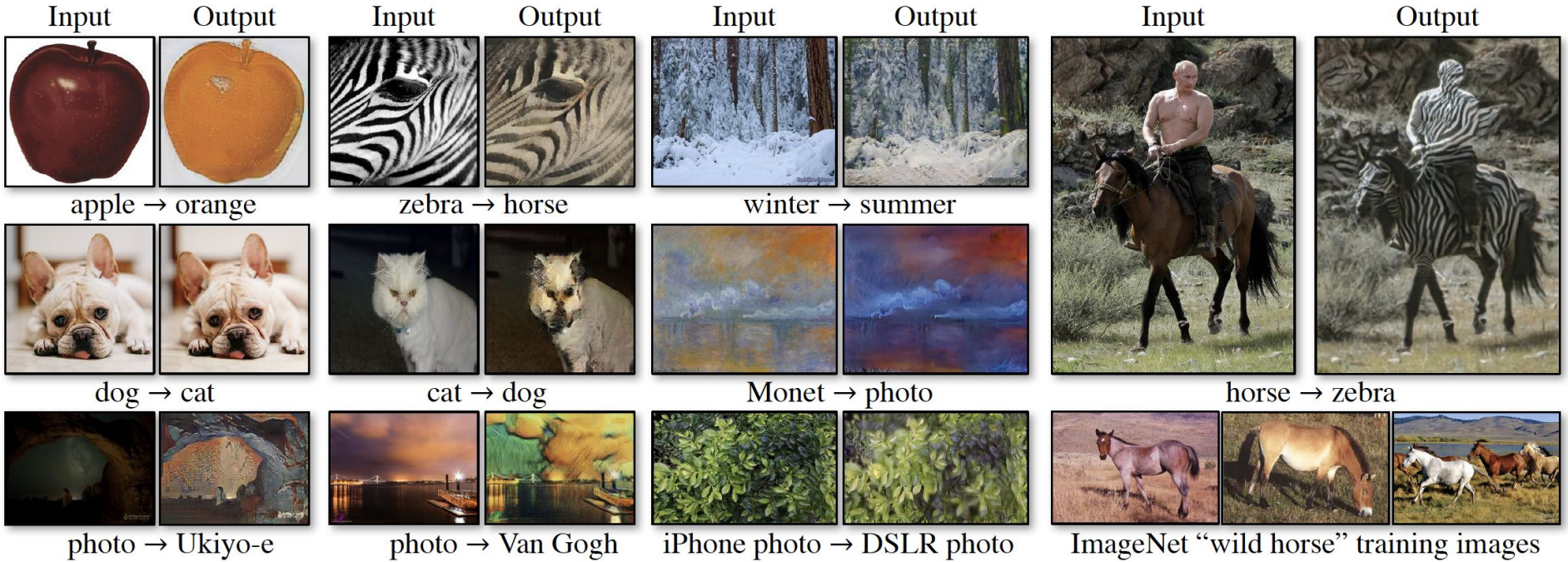


Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for more comprehensive results.

Comments

- More baseline algorithms used in the paper (CoGAN, SimGAN, BiGAN/ALI)

Observation (more general):

- Notice how different the 'fake-real' discrimination task is from the typical classification tasks.
- It must almost certainly rely on completely different level of abstraction.

Deep Convolutional GAN (DCGAN)

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Alec Radford, Luke Metz, Soumith Chintala

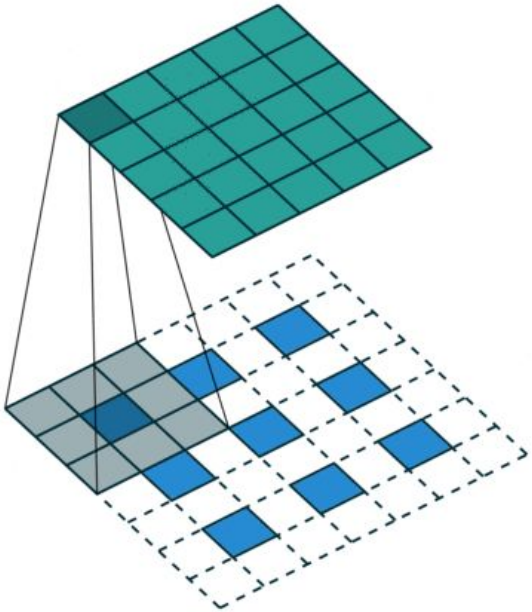
https://arxiv.org/abs/1511.06434

Motivations

Come up with guidelines for designing GANs that are stable to train: “constraints on the architectural topology of Convolutional GANs”

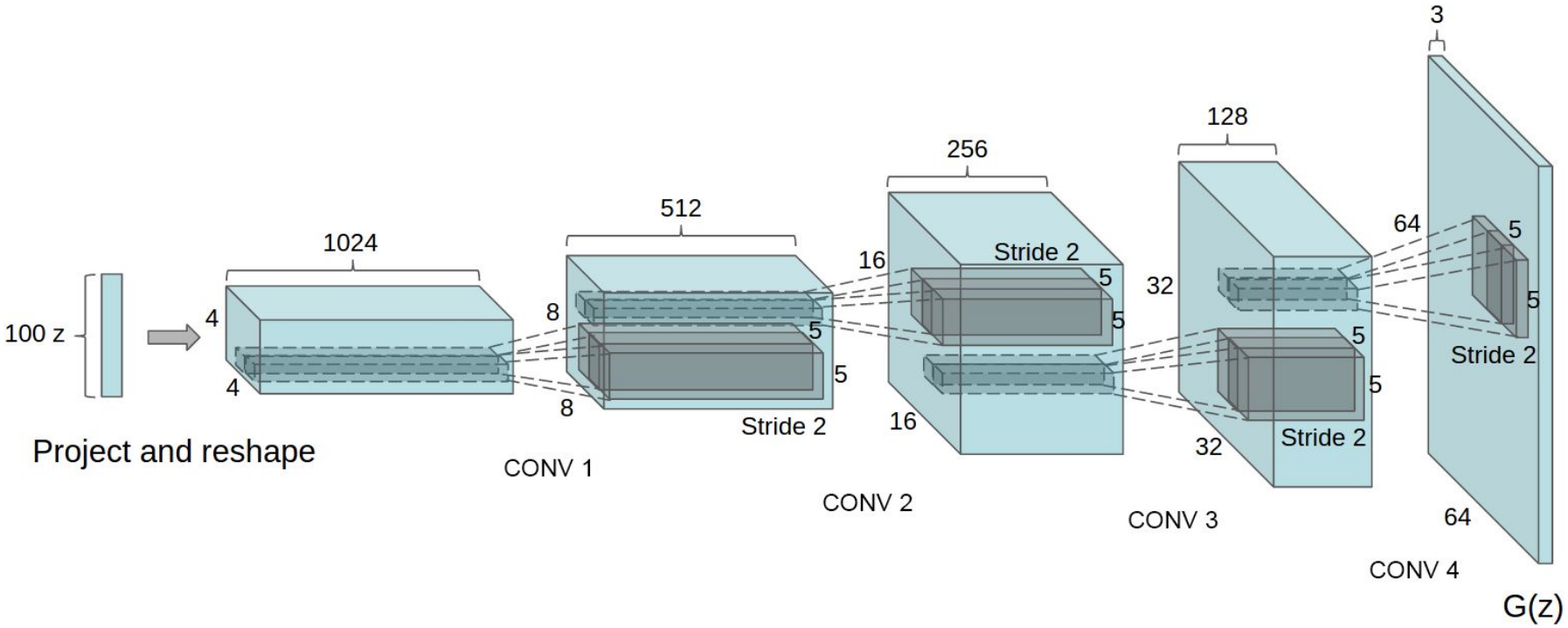
- Replace pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
  - Allows the network to learn its own spatial downsampling.
- Use batchnorm in both the generator and the discriminator.
  - Exception: Applying batchnorm to all layers resulted in sample oscillation and model instability.
  - Therefore: no batchnorm in generator’s output layer and discriminator’s input layer.
- Remove fully connected hidden layers for deeper architectures.
  - Except for the first layer of G, which multiplies z by a matrix and reshapes it for convolution.
- Use ReLU activation in generator for all layers
  - Except for the output, which uses Tanh: bounded activation allowed the to faster saturate and cover the color space of the training distribution.
- Use LeakyReLU activation in the discriminator for all layers.

Fractionally strided transposed convolution



Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning, https://github.com/vdumoulin/conv_arithmetic

Generator architecture



Results

Massive experimental investigation; among others:

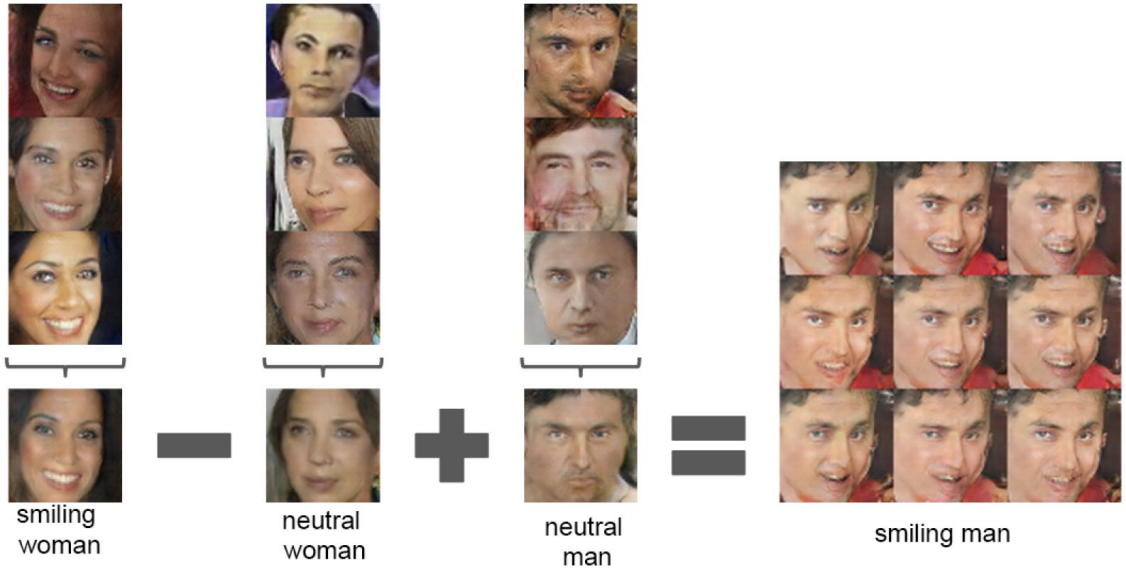
- Generating bedrooms: Large-scale Scene Understanding (LSUN) dataset
- Classifying CIFAR-10 and numbers in StreetView House Numbers dataset (SVHN) using GANs as a feature extraction.
- Human faces (scrapped from the internet, 10k people, 3M images)
- Visualization of features

Interesting preprocessing for bedrooms: autoencoder for dataset deduplication.

- Trained de-noising (?) dropout regularized ReLU autoencoder on 32x32 downsampled center-crops of training examples.
- Code layer binarized via thresholding, obtaining so a form of semantic hash code.
- FP rate < 0.01. Helped removing nearly 275,000 [near] duplicates.

Results

1. Z vectors of samples are averaged over columns.
2. Arithmetic performed on the mean vectors, creating a new vector Z'.
3. The center sample on the right produced by feeding Z' to G.
4. Uniform noise +-0.25 added to Z' to produce the 8 other samples (to show interpolation capabilities).



Visualization of features

Random filters Trained filters

- Right: guided backpropagation (Springenberg et al., 2014) visualizations of maximal axis-aligned responses for the first 6 learned convolutional features from the last convolution layer in the discriminator.
- Notice how localized the features are (recall: last convolutional layer!)

A representative of other variants of GANs: Coulomb GANs

Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields

Thomas Unterthiner, Bernhard Nessler, Calvin Seward, Günter Klambauer, Martin Heusel, Hubert Ramsauer, Sepp Hochreiter

http://arxiv.org/abs/1708.08819

Motivation

- It has been proven that GAN learning does converge when discriminator and generator are learned using a two time-scale learning rule (Heusel et al., 2017).
  - Convergence means that the expected SGD-gradient of both the discriminator objective and the generator objective are zero.
  - Thus, neither the generator nor the discriminator can locally improve, i.e., learning has reached a local Nash equilibrium.
- However, convergence alone does not guarantee good generative performance.
  - It is possible to converge to sub-optimal solutions which are local Nash equilibria.

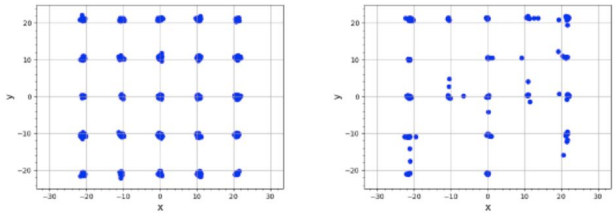
Authors' proposal:

- Coulomb GAN, which has only one Nash equilibrium.
- Show that this Nash equilibrium is optimal, i.e., the model distribution matches the target distribution.

Side-note on mode collapse

Mode collapse is a special case of a local Nash equilibrium associated with suboptimal generative performance.

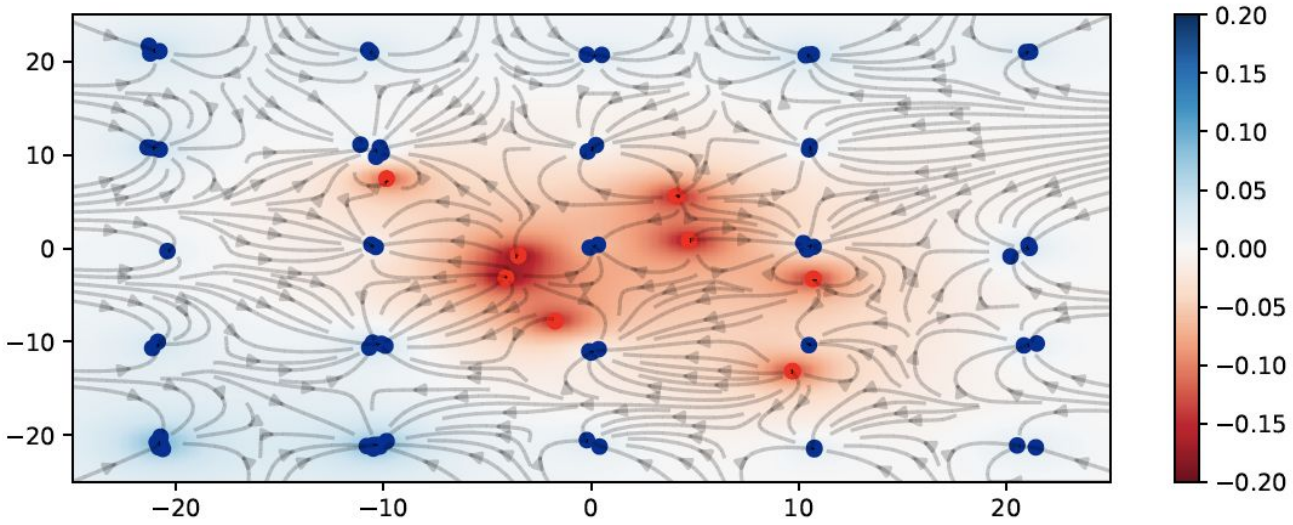
- For example:
  - Assume a two-mode real world distribution where one mode contains too few and the other mode too many generator samples.
  - If no real world samples are between these two distinct modes, then the discriminator penalizes to move generated samples outside the modes.
  - Therefore the generated samples cannot be correctly distributed over the modes.
- Thus, standard GANs cannot capture the global sample density such that the resulting generators are prone to neglect large parts of the real world distribution.



(a) True data (b) GAN

Coulomb GAN: The intuition

True samples (blue) and generated samples (red) create a potential field (scalar field). Blue samples act as sinks that attract the red samples, which repel each other. The superimposed vector field shows the forces acting on the generator samples to equalize potential differences, and the background color shows the potential at each position.



Realization

Rather than minimizing the difference* between densities ρ(a) = $p_y(a)$ - $p_x(a)$, where $p_x(a)$ is the model density and $p_y(a)$ is the target density, minimize a potential function defined via kernel functions k(a,b) that define the influence of a point at b onto a point at a:

$$\Phi(\boldsymbol{a}) \;=\; \int \rho(\boldsymbol{b})\, k\,(\boldsymbol{a},\boldsymbol{b})\; \mathrm{d}\boldsymbol{b}\,, \quad k(\boldsymbol{a},\boldsymbol{b}) \;=\; \frac{1}{(\sqrt{\|\boldsymbol{a}-\boldsymbol{b}\|^2 \;+\; \epsilon^2})^d}$$

Minimizing this potential function causes minimization of the difference of densities ρ(a).

* In conventional GANs, D(a) is optimized to approximate the probability of seeing a target sample, or ρ(a).

Technical realization

For N_x fakes (transformed by the generator) and N_y real samples, an unbiased estimate of Φ is:

$$\hat{\Phi}(\boldsymbol{a}) = \frac{1}{N_y}\sum_{i=1}^{N_y} k\left(\boldsymbol{a}, \boldsymbol{y}_i\right) - \frac{1}{N_x}\sum_{i=1}^{N_x} k\left(\boldsymbol{a}, \boldsymbol{x}_i\right)$$

Update rule for the discriminator:

$$d\boldsymbol{w} \leftarrow \nabla_{\boldsymbol{w}}\left[\frac{1}{2}\sum_{i=1}^{N_x}\left(D(\boldsymbol{x}_i) - \hat{\Phi}(\boldsymbol{x}_i)\right)^2 + \frac{1}{2}\sum_{i=1}^{N_y}\left(D(\boldsymbol{y}_i) - \hat{\Phi}(\boldsymbol{y}_i)\right)^2\right]$$

Update rule for the generator (note: G is implicit in this formula):

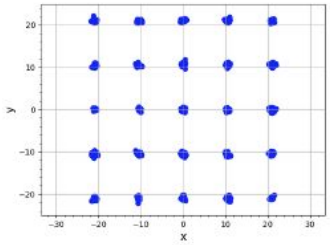
$$d\boldsymbol{\theta} \leftarrow \nabla_{\boldsymbol{\theta}}\left[-\frac{1}{2}\frac{1}{N_x}\sum_{i=1}^{N_x} D\left(\boldsymbol{x}_i\right)\right]$$

(more technical details in the paper).

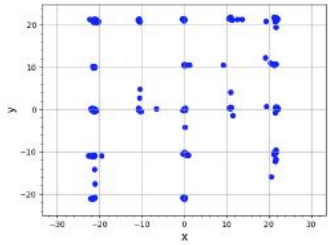
Results

- Tested on images CelebA, LSUN bedrooms, and CIFAR-10.
- Coulomb GANs tend to outperform standard GAN approaches like BEGAN and DCGAN, but are outperformed by the Improved Wasserstein GAN.
  - However Improved Wasserstein GAN used a more advanced network architecture based on ResNet blocks (Gulrajani et al., 2017).
- Results in terms of FID, Frechet Inception Distance (Frechet = Wasserstein-2):
  - Like inception score, but takes into account the distributions.

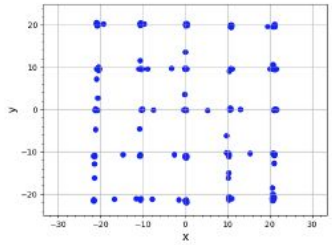
data set	BEGAN	DCGAN	WGAN-GP	MMD	ours
CelebA	29.2 / 28.5	21.4 / 12.5	4.8 / 4.2	63.2	9.3
LSUN	113 / 112	70.4 / 57.5	20.5 / 9.5	94.9	31.2
CIFAR10	-	-	29.3 / 24.8	38.2	27.3



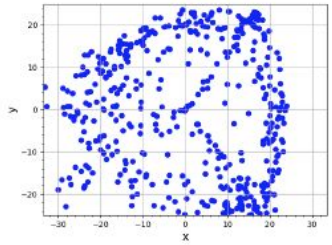
(a) True data



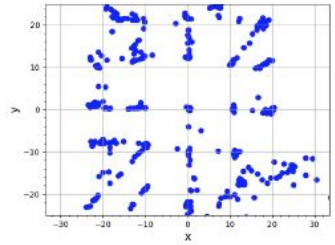
(b) GAN



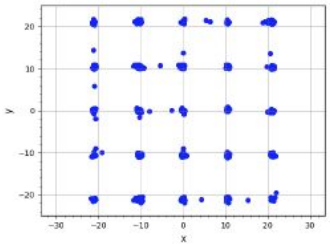
(c) Geometric GAN



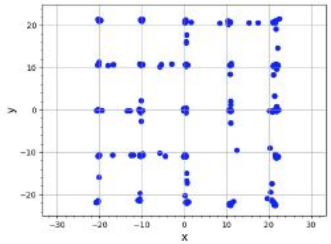
(d) WGAN



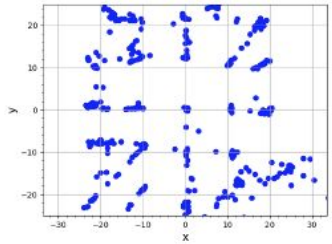
(e) meanGAN + proj.



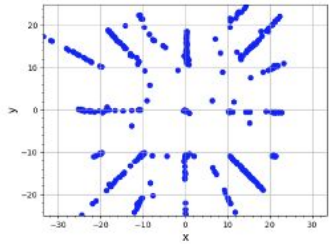
(f) Coulomb GAN



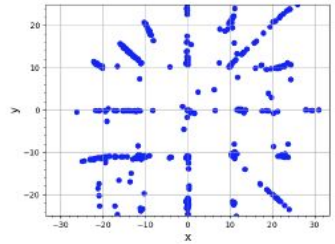
(g) GAN + WD



(h) Geo. GAN + WD



(i) WGAN + WD



(j) meanGAN + WD

Wasserstein GAN

Wasserstein GAN

Martin Arjovsky, Soumith Chintala, Léon Bottou

https://arxiv.org/abs/1701.07875

Motivation

- Learning a generator in GAN is learning a probability distribution using a parameterized family of densities P_θ:

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

The authors:

- Point to the fact that training a traditional GAN aims at minimizing the KL divergence between P_θ and the real distribution P_r.
- Discuss the pros and cons of various measure of the similarity of P_θ to P_r.
- Provide a comprehensive theoretical analysis of how the Earth Mover (EM) distance behaves in comparison to popular probability distances and divergences used in the context of learning distributions.
- Propose Wasserstein GAN that minimizes a reasonable and efficient approximation of the EM distance.

Distances/divergence studied in the paper

Total Variation (TV) distance:

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

Kullback-Leibler (KL) divergence:

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x)$$

Jensen-Shannon (JS) divergence:

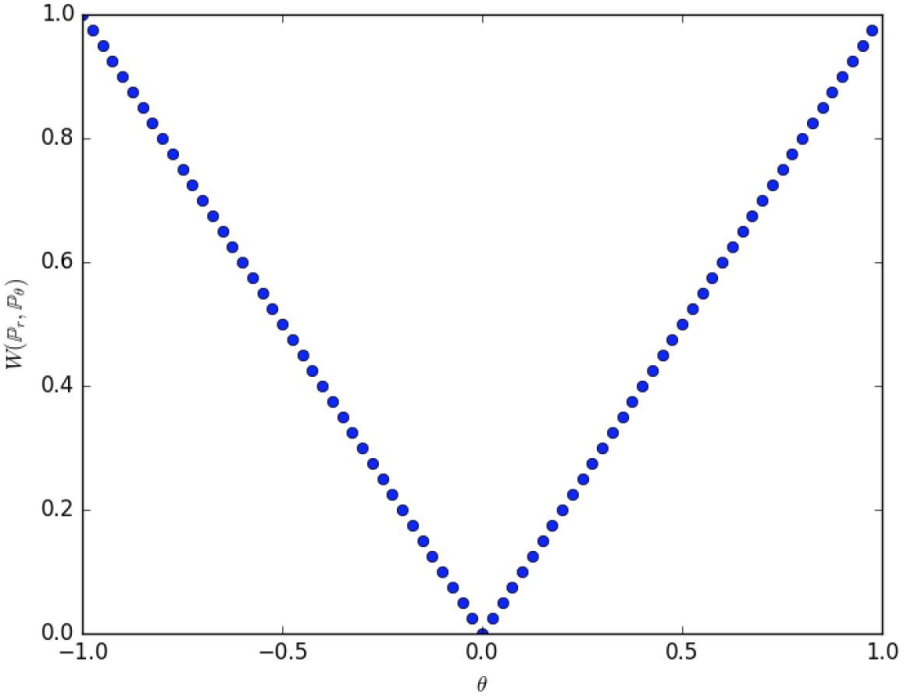
$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m)$$

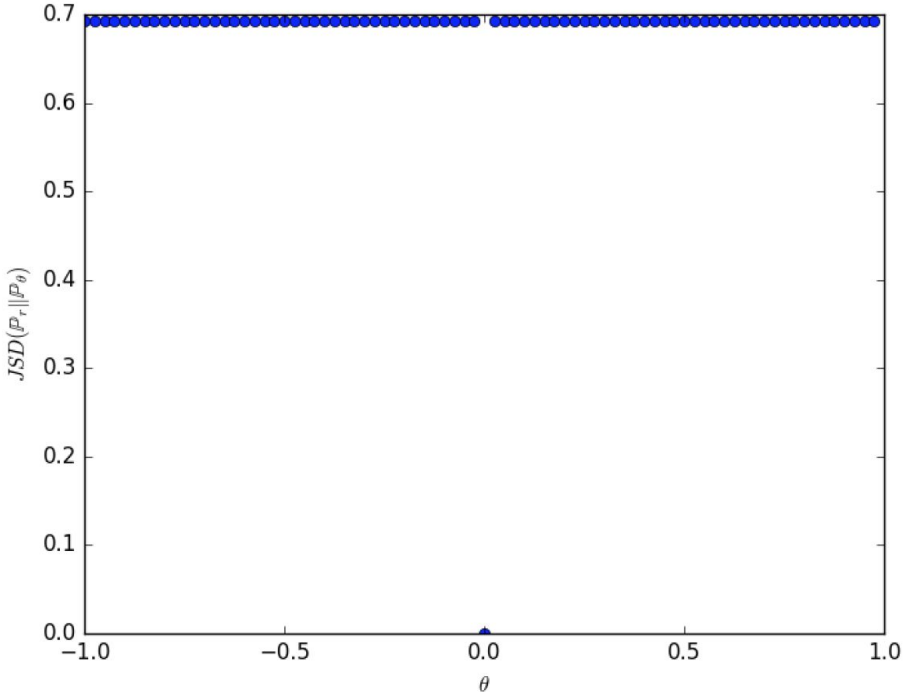
Earth-Mover (EM) distance (Wasserstein-1):

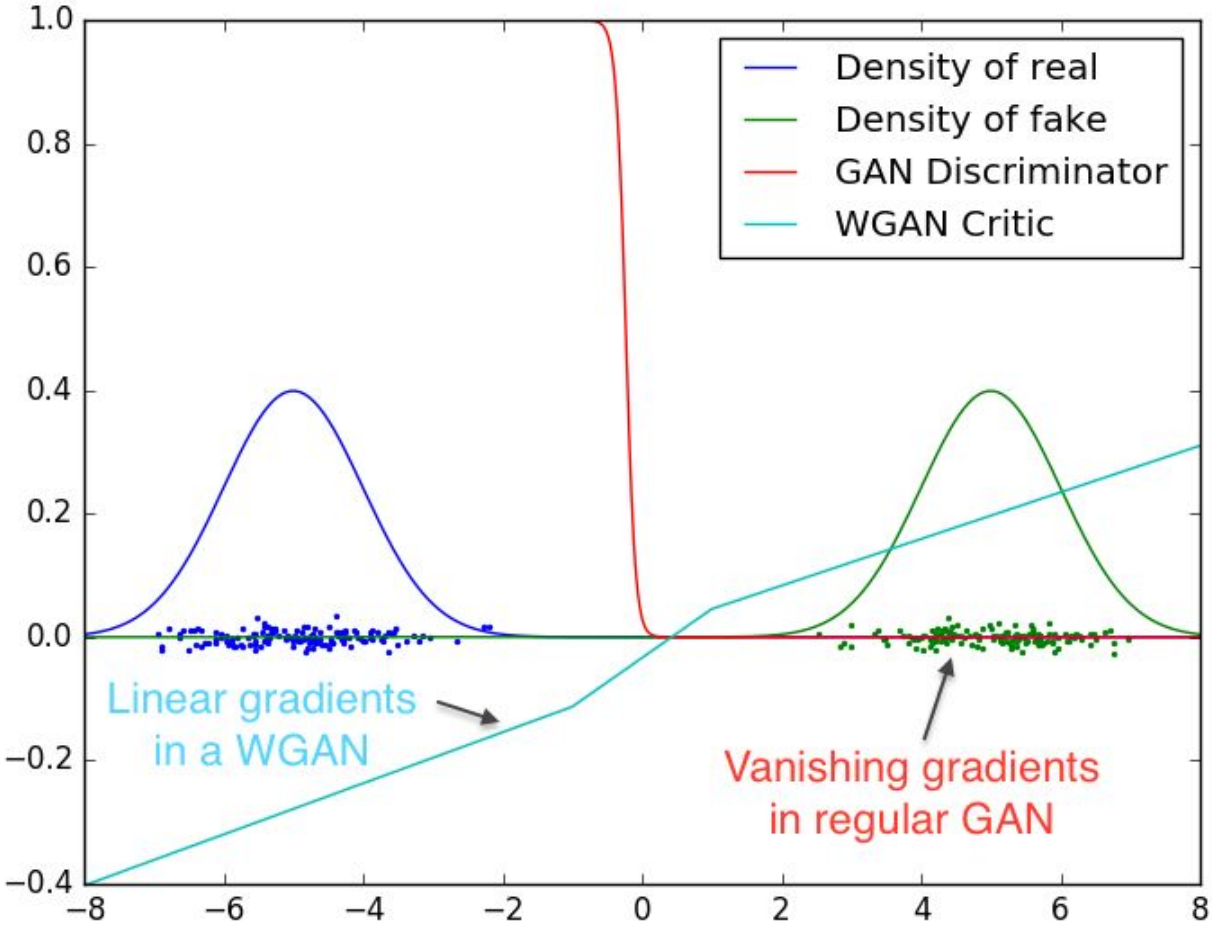
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \left[\, \|x - y\| \, \right]$$

Distance/divergence characteristics

as a function of distribution parameter θ:






Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients
in a WGAN
Vanishing gradients
in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

Wasserstein GAN

Key element: weight clipping.

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

RMSProp: T. Tieleman and G. Hinton. Lecture 6.5, RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Implications

- a meaningful loss metric that correlates with the generator's convergence and sample quality
- improved stability of the optimization process

Shown empirically.