Imperial College London

Generative Adversarial Networks (GANs)

Introduction to the theory of GANs

Jiashun Yao

Outline

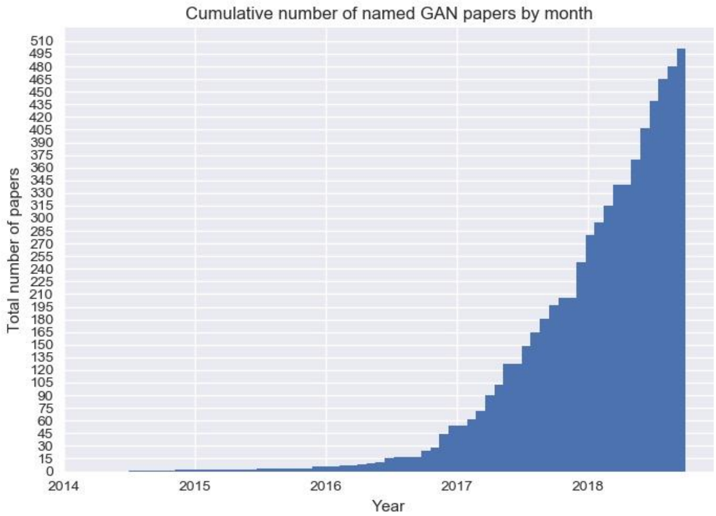
- Generative model
- Adversarial loss
- Theory of GANs
- How to train GANs
- Flavours of GANs

Why GANs

Quora Session with Yann LeCun
Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Answered July 28, 2016 · Upvoted by Joaquin Quiñonero Candela, studied Machine Learning and Gokul Krishnan, M.Sc Computer Science & Machine Learning, ETH Zurich (2018)

There are many interesting recent development in deep learning, probably too many for me to describe them all here. But there are a few ideas that caught my attention enough for me to get personally involved in research projects.

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

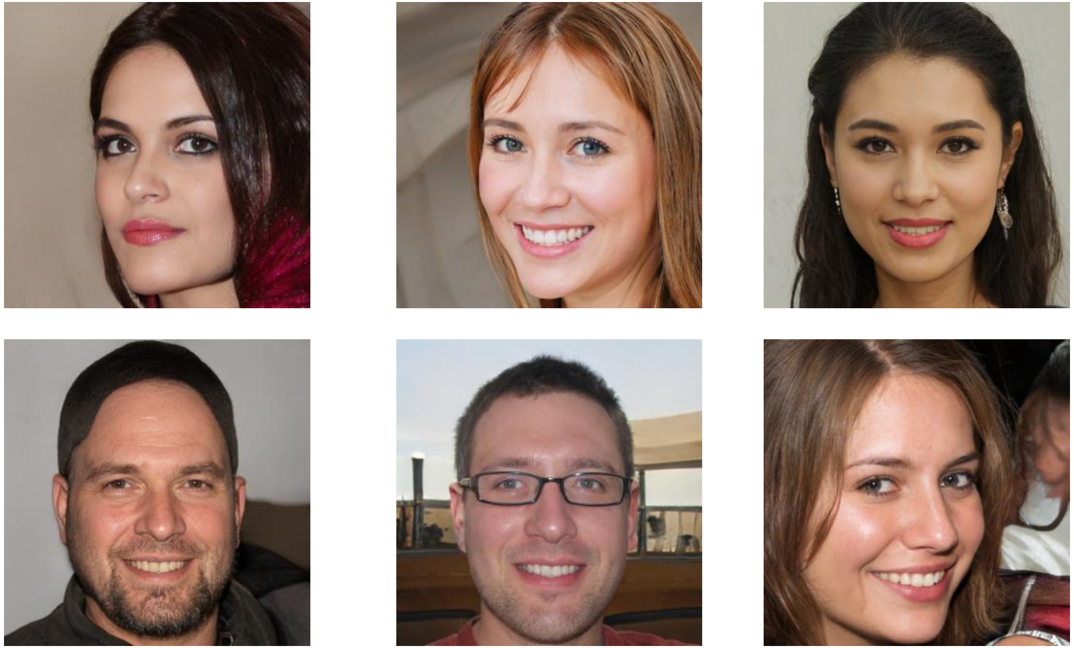


https://github.com/hindupuravinash/the-gan-zoo

- GANs has shown tremendous success in the generation of realistic data;
- Can be used to address many kinds of problems (generation, clustering, representation learning, translation …) for many types of data (image, audio, video, text …)

Generative adversarial networks
IJ Goodfellow, J Pouget-Abadie, M Mirza, B Xu... - arXiv preprint arXiv ..., 2014 - arxiv.org
We propose a new framework for estimating **generative** models via an **adversarial** process, in which we simultaneously train two models: a **generative** model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came ...
Cited by 30182 Related articles All 55 versions

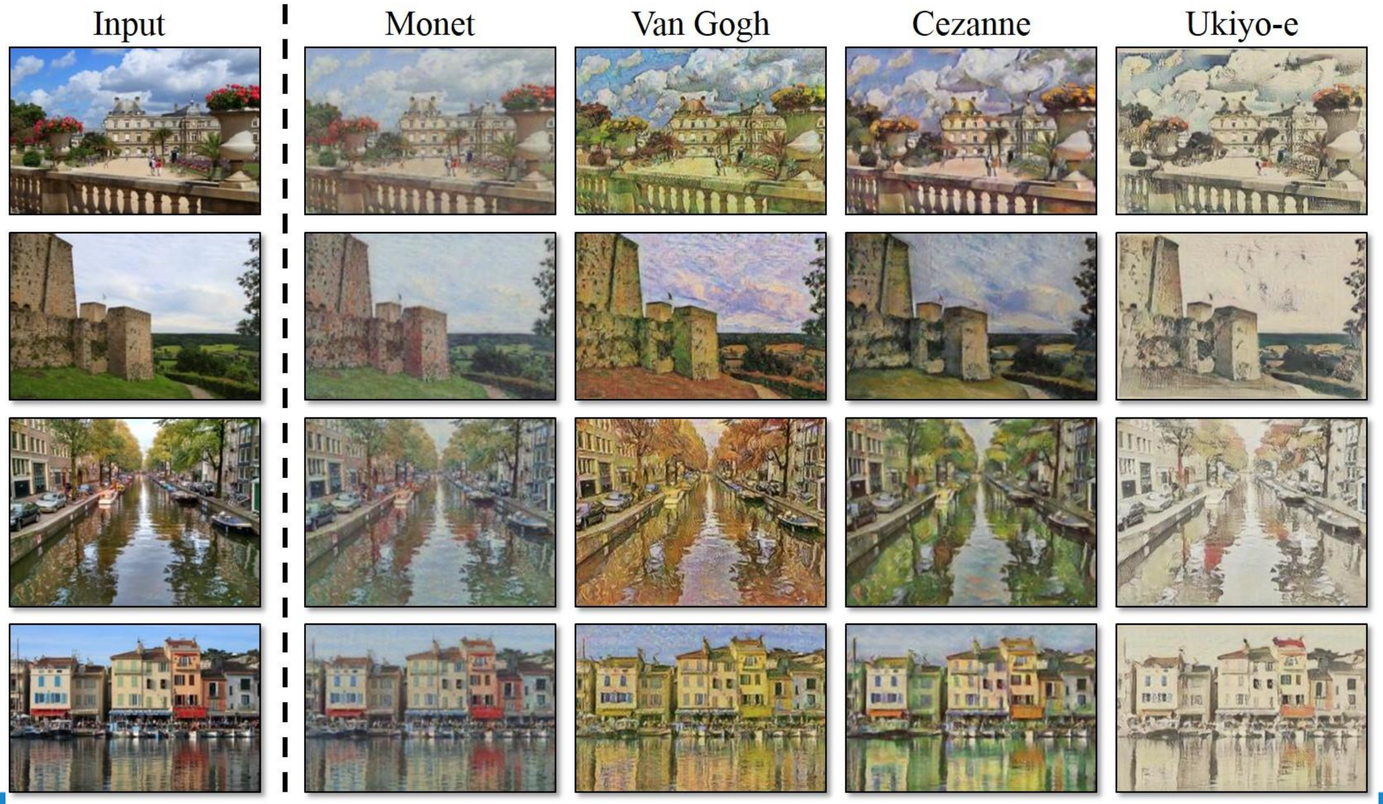
Example – 1 – generation of realistic human faces



https://arxiv.org/abs/1710.10196

Example – 2 – style mix of faces

Example – 3 – photos to arts



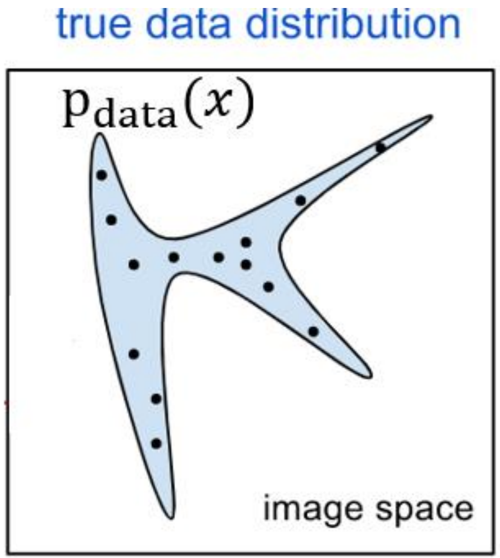
https://arxiv.org/abs/1703.10593

Generative model

Given training real data

→ find the distribution $\mathrm{p_{data}}(x)$ that explains where the samples are from

→ Use $\mathrm{p_{data}}(x)$ to generate realistic samples.



x – image or other data
high dimensional vector

Cannot explicitly know the probability density function that describes of $\mathrm{p_{data}}(x)$.

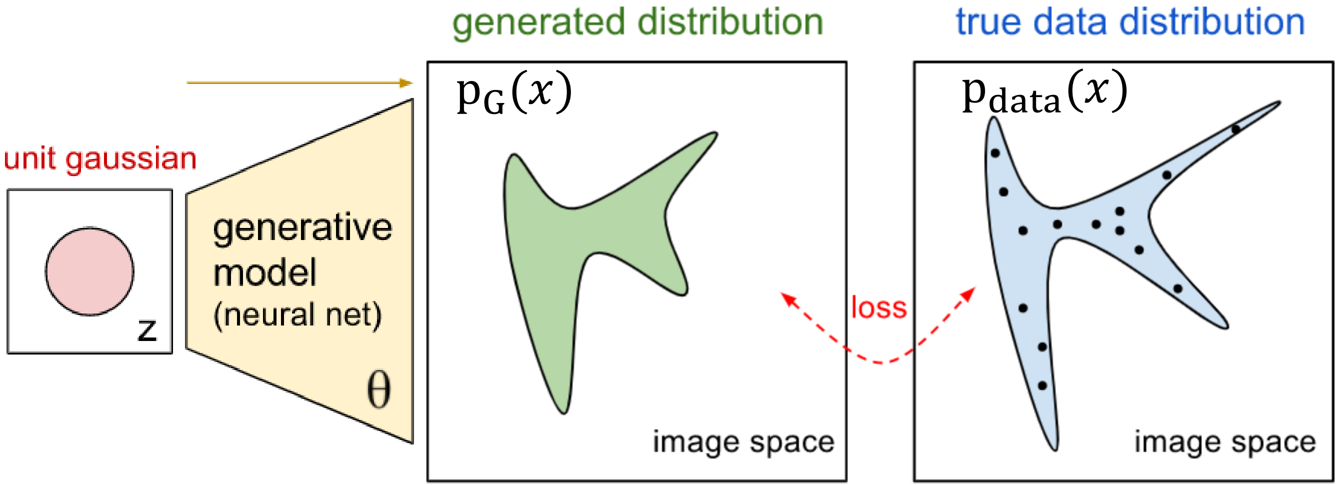
Use neural network G to learning $\mathrm{p_{data}}(x)$.

Image: https://openai.com/blog/generative-models/

Generative model

Given training real data, for a generative network (G) :

- G is a mapping function from z to data space
- True data distribution $\mathrm{p}_{\mathrm{data}}(x)$
- Generated data distribution $\mathrm{p}_{\mathrm{G}}(x)$



We want to find G^* that

$\mathrm{p}_{\mathrm{G}^*} \approx \mathrm{p}_{\mathrm{data}}$

x: high dimensional vector

Image: https://openai.com/blog/generative-models/

Adversarial loss for the generative model

To find optimal G^*, update network parameters, to minimise the divergence between $\mathrm{p_G}$ and $\mathrm{p_{data}}$:

$$\mathrm{G}^* = \arg\min_{\mathrm{G}} Div\,(P_{data}(x)||P_G(x))$$

No simple loss function available to measure this divergence.

VAE trains on evidence lower bound (ELBO), a surrogate of $Div\,(P_{data}(x)||P_G(x))$.

Adversarial loss for the generative model

VAE:

1. Sample x_i from training data
2. Train $Encoder(x_i) \rightarrow z_i, Decoder(z_i) \rightarrow \hat{x}_i$
   - Explicit loss between x_i and $\hat{x}_i$ (for images, pixel-wise comparison)
3. Once trained, sample z_m from z, to generate x_m

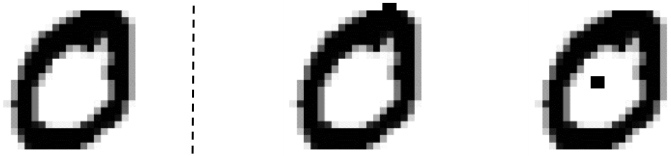
Issues of VAE:

- Pixel-loss not intelligent enough
- Blurry output

Adversarial loss for the generative model

VAE:
1. Sample x_i from training data
2. Train $Encoder(x_i) \rightarrow z_i, Decoder(z_i) \rightarrow \hat{x}_i$
   - Explicit loss between x_i and $\hat{x}_i$ (for images, pixel-wise comparison)
3. Once trained, sample z_m from z, to generate x_m

Issues of VAE:
- Pixel-loss not intelligent enough
- Blurry output



True *Sample 1* *Sample 2*

Which is better or equally good?

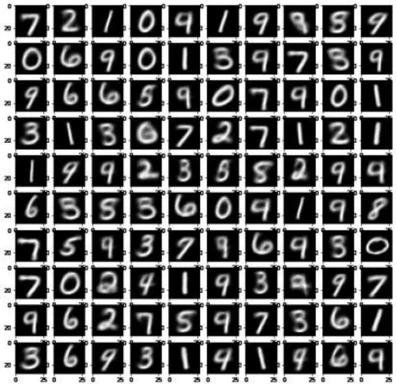
Adversarial loss for the generative model

VAE:

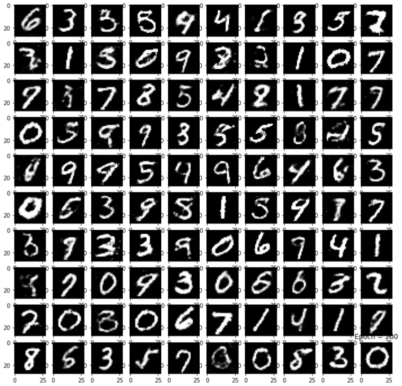
1. Sample x_i from training data
2. Train $Encoder(x_i) \rightarrow z_i, Decoder(z_i) \rightarrow \hat{x}_i$
    - Explicit loss between x_i and $\hat{x}_i$ (for images, pixel-wise comparison)
3. Once trained, sample z_m from z, to generate x_m

Issues of VAE:

- Pixel-loss not intelligent enough
- Blurry output



VAE



GANs

Distance between samples good enough?

Distance between complex distributions?

Adversarial loss for the generative model

GAN provides a method to optimise $G^* = \arg\min_G Div\,(p_{data}||p_G)$ by using another NN (Discriminator) for the loss function – the "adversarial loss".

Training GAN → Minimising J-S divergence
(other divergences/distances possible too)

- Intelligent & flexible
- "Perceptually better" than sample-wise losses

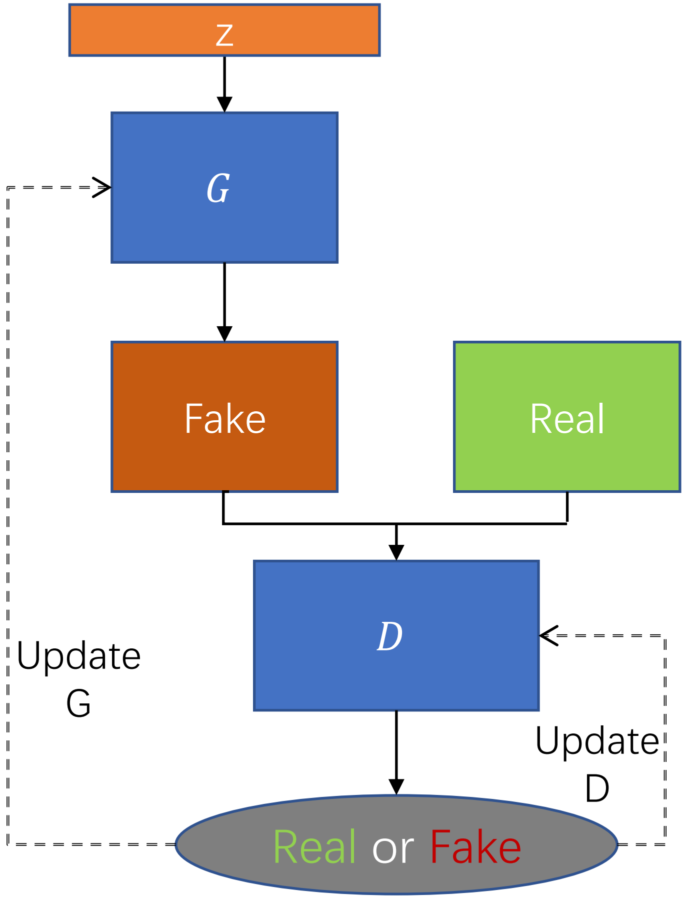
Theory of GANs

GANs: Generator NN (G) + Discriminator NN (D)

Generator G:
- Input: a vector z from $p_{prior}(z)$
- Output: **Fake** data sample $G(z)$

Discriminator D:
- Input: a **Real** data sample x or a **Fake** sample $G(z)$
- Output: a scalar (possibility of input to be **Real**)
  - D is a binary classifier

Real: training set data
Fake: generator output



Theory of GANs

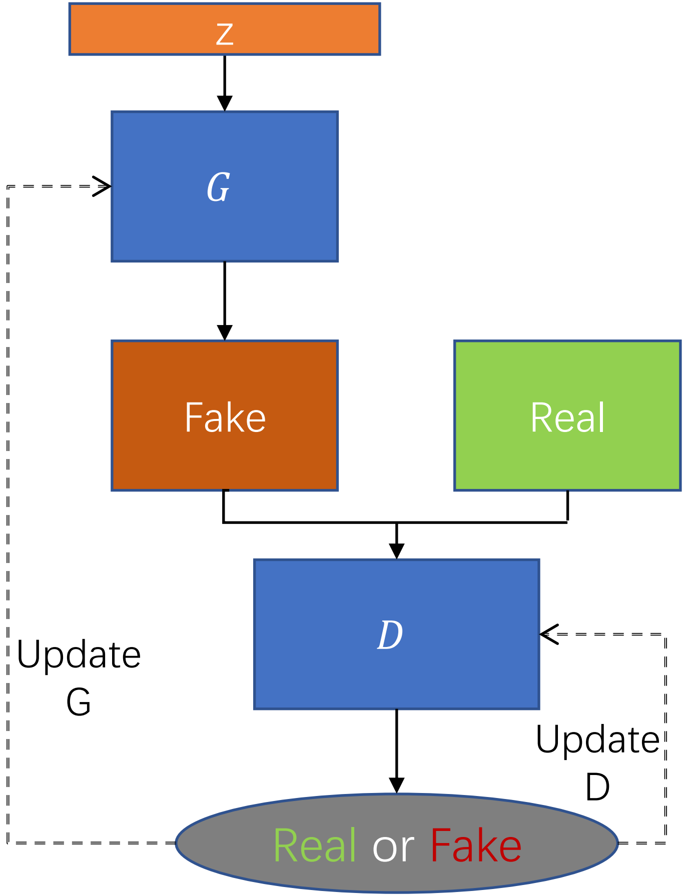
To make $G(z)$ → Realistic, we play a zero-sum competition game:

- train D to correctly label its input to be **Real** or **Fake**;
- train G to “fool” D to label $G(z)$ to be **Real**.

So that both D and G improve as during training:

- G can generate more and more realistic data
- D can tell more and more detailed differences between Fake and Real data

Optimal G: $\mathrm{p}_G = \mathrm{p}_{data}$



Theory of GANs

To play a zero-sum game, we use a min-max loss function:

$$\min_G \max_D V_{GAN}(G, D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{z\sim p_z}[\log\left(1 - D\left(G(z)\right)\right)]$$

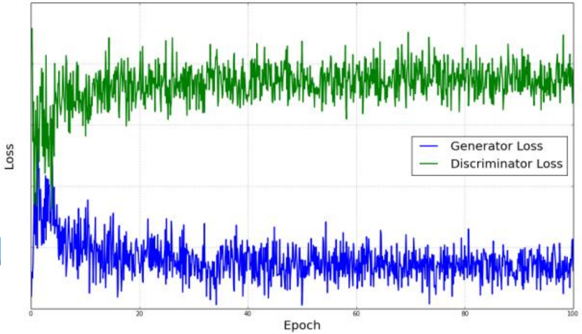
Train D to maximise the loss, so that:

- Input is **Real** → $D(x) \to 1$
- Input is **Fake** → $D(G(z)) \to 0$

Train G to minimise the loss:

- Fools D to give high score when input is **Fake**: $D(G(z)) \to 1$

Loss curve oscillates rather than decreases:



Theory of GANs

$$\min_G \max_D V_{GAN}(G,D) = \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{z\sim p_z}[\log\left(1 - D\left(G(z)\right)\right)]$$

Given x:

$$D^* = \arg\max_D \left[p_{data}(x)logD(x) + p_G(x)\log\left(1 - D(x)\right)\right]$$

It gives:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

Substitute into the objective function:

$$V_{GAN}(G,D^*) = \mathbb{E}_{x\sim p_{data}}\left[log\frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + \mathbb{E}_{z\sim p_z}\left[log\frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right]$$

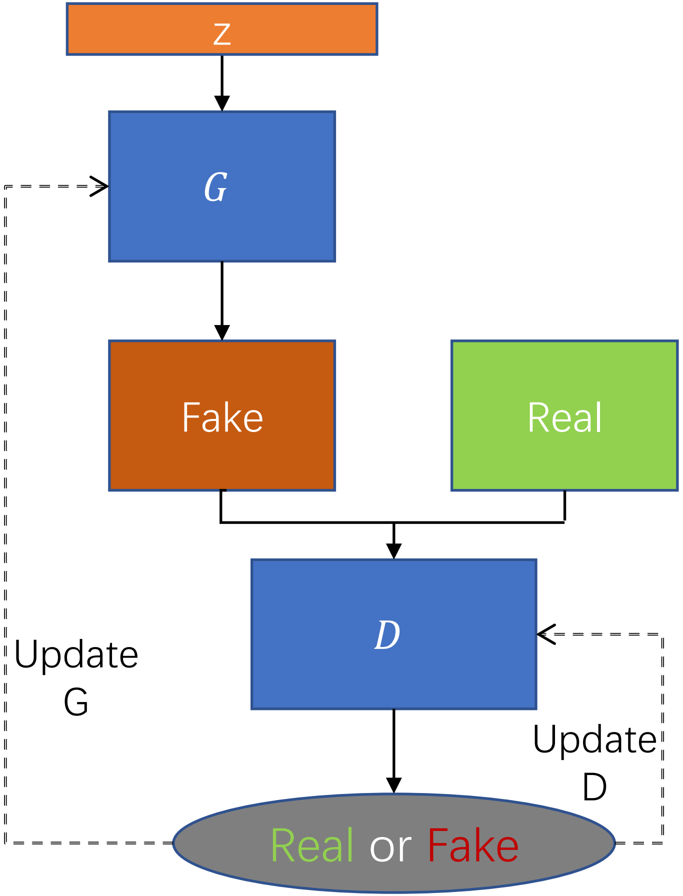
$$= -log4 + 2\,\mathrm{JSD}(p_{data}||p_G)$$

Jensen-Shannon divergence

Train GANs – will implement tomorrow

In each iteration:

1. Fix G, train D to distinguish Real or Fake data:
   - $D(x) \to 1$
     1. Sample real samples x
     2. Input x into D
     3. Train $D: D(x) \to 1$
   - $D(G(z)) \to 0$
     1. Sample vectors of z from $p_{prior}(z)$
     2. Input z into G to generate fake samples $G(z)$
     3. Train D: $D(G(z)) \to 0$
2. Fix D, train G to “fool” D:
   1. Sample vectors z from $p_{prior}(z)$
   2. Input z into G to generate fake samples $G(z)$
   3. Train G: $D(G(z)) \to 1$

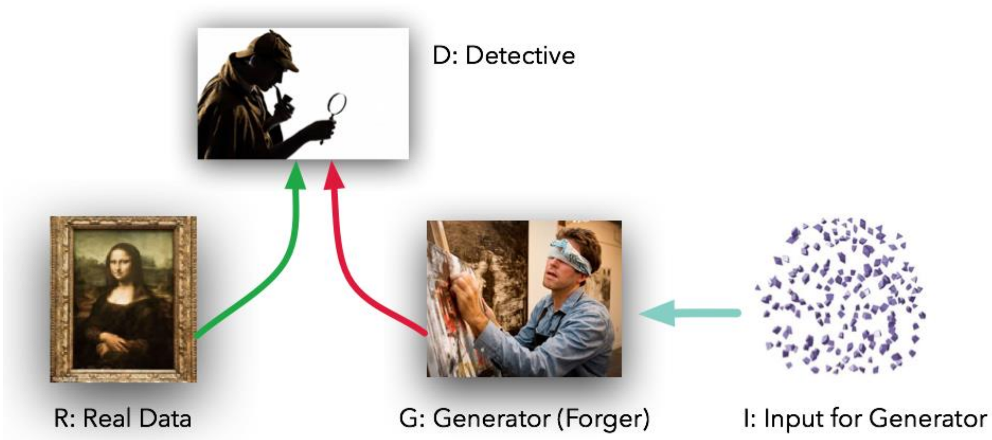


GAN illustration demo: https://poloclub.github.io/ganlab/

Theory of GANs

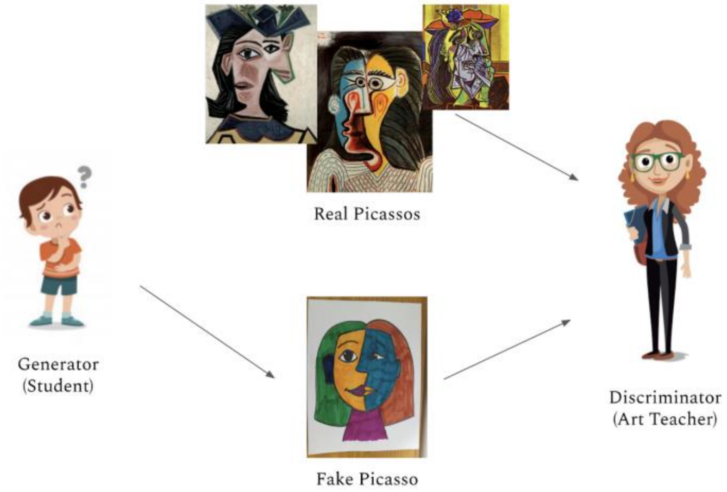
Relationship between G and D?

From the objective function



https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

From the process of interaction



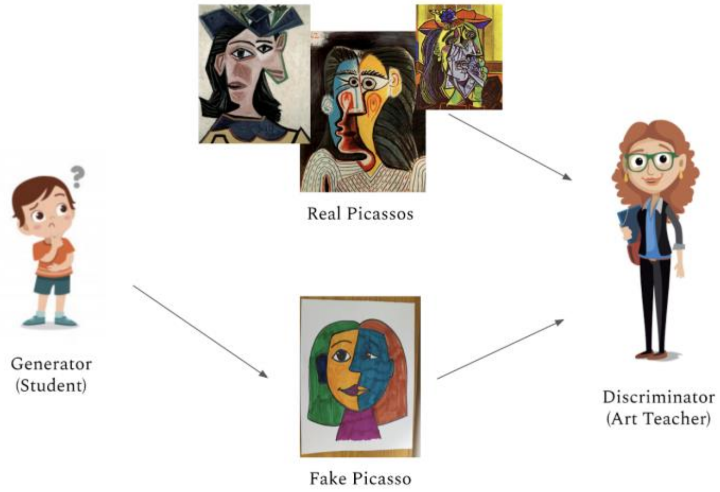
http://robotic.media.mit.edu/wp-content/uploads/sites/7/2021/03/EAAI-What-are-GANs_.pdf

Theory of GANs

Interaction between G and D :

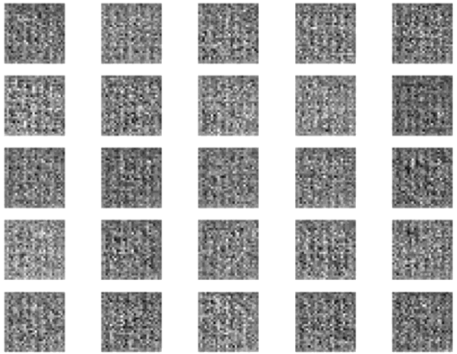
- D is leading the G
  - D is trained first
  - D provide "knowledge" to update G

- D needs to "teach" according to the current level of G
  - Measure the distance between current p_G and p_{data}

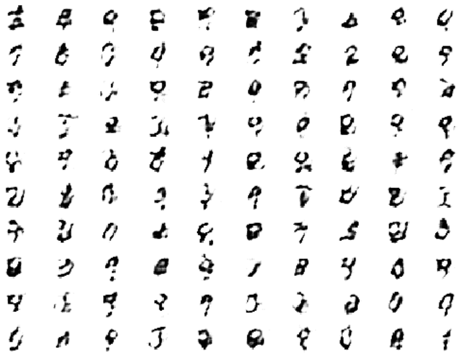


http://robotic.media.mit.edu/wp-content/uploads/sites/7/2021/03/EAAI-What-are-GANs_.pdf

Theory of GANs



0th Epoch



10th Epoch

100th Epoch

https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/

Pros and Cons

- GANs trained to minimise the JS divergence between p_G and p_{data}

- GANs produce "sharper" and more "perceptually realistic" results

- VAEs are stable in training, and converge faster
- GANs are hard to train, and have unclear stopping criteria

- VAEs provide generative model and inference model
  - Learn an encoder decoder pair
- GANs only has generative model

- VAEs are more theoretically justified

Many different flavours of GANs:

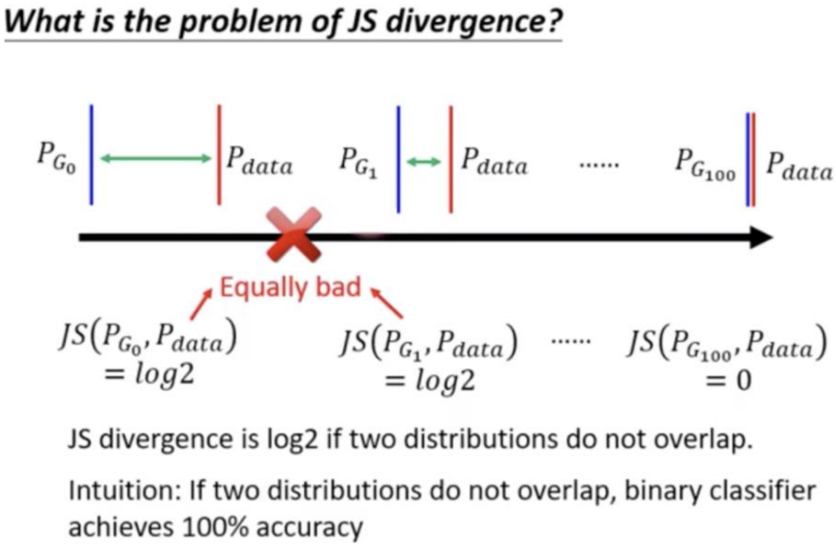
Different objective functions
GAN (JS divergence)
Wasserstein GAN (WGAN)
WGAN GP (Gradient Penalty)
f-GAN
LSGAN
Energy-based GAN (EBGAN)
BEGAN
Fisher GAN (Chi-square)
…

Different architectures
Conditional GAN (cGAN)
VAE-GAN
Cycle-GAN
BiGAN
BicycleGAN
Style-GAN
Self-attention GAN
BigGAN
Disco-GAN
Countless more …

Many different flavours of GANs:

Different objective functions
GAN (JS divergence)
Wasserstein GAN (WGAN)
WGAN GP (Gradient Penalty)
f-GAN
LSGAN
Energy-based GAN (EBGAN)
BEGAN
Fisher GAN (Chi-square)
...

Works for initial p_G and p_{data} are not overlapped (most of the cases).



Many different flavours of GANs:

Different objective functions
GAN (JS divergence)
Wasserstein GAN (WGAN)
WGAN GP (Gradient Penalty)
f-GAN
LSGAN
Energy-based GAN (EBGAN)
BEGAN
Fisher GAN (Chi-square)
…

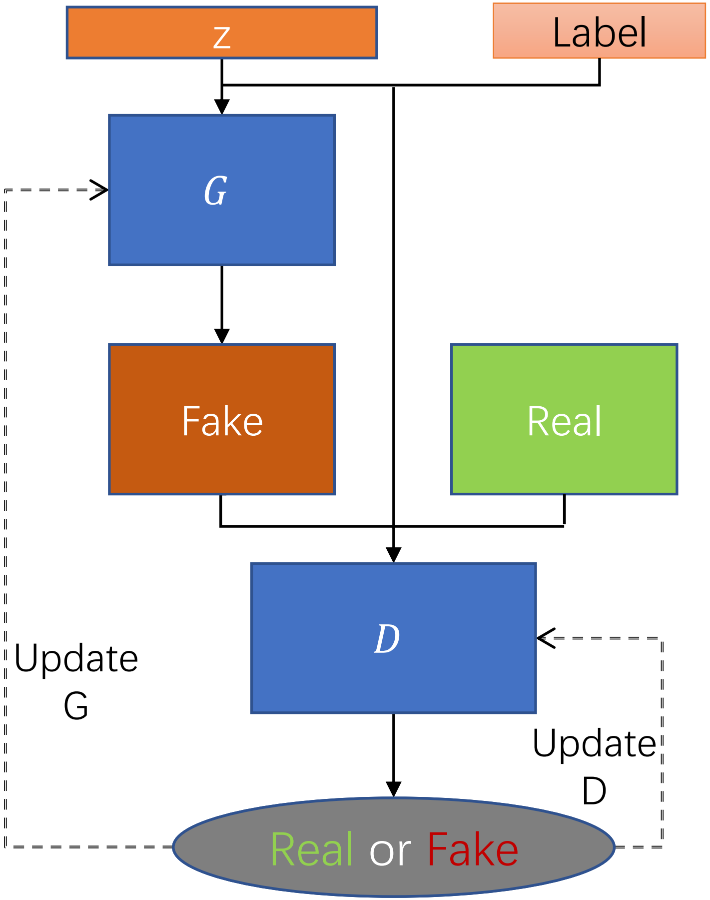
Different architectures
Conditional GAN (cGAN)
VAE-GAN
Cycle-GAN
BiGAN
BicycleGAN
Style-GAN
Self-attention GAN
BigGAN
Disco-GAN
Countless more …

Conditional GAN (cGAN)

Vanilla GAN has limited implementations – unsupervised generation of unlabelled data.

cGAN solves this by providing the label, so that the output of G should be:

- Realistic
- Matching to the given label

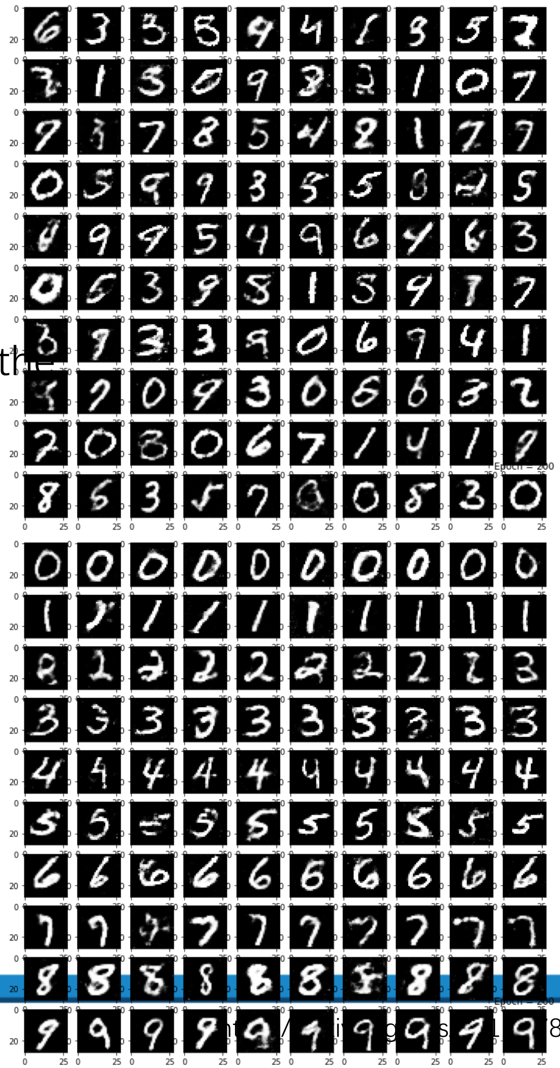


Conditional GAN (cGAN)

Vanilla GAN has limited implementations – unsupervised generation of unlabelled data.

cGAN solves this by providing the label, so that the output of G should be:

- Realistic
- Matching to the given label

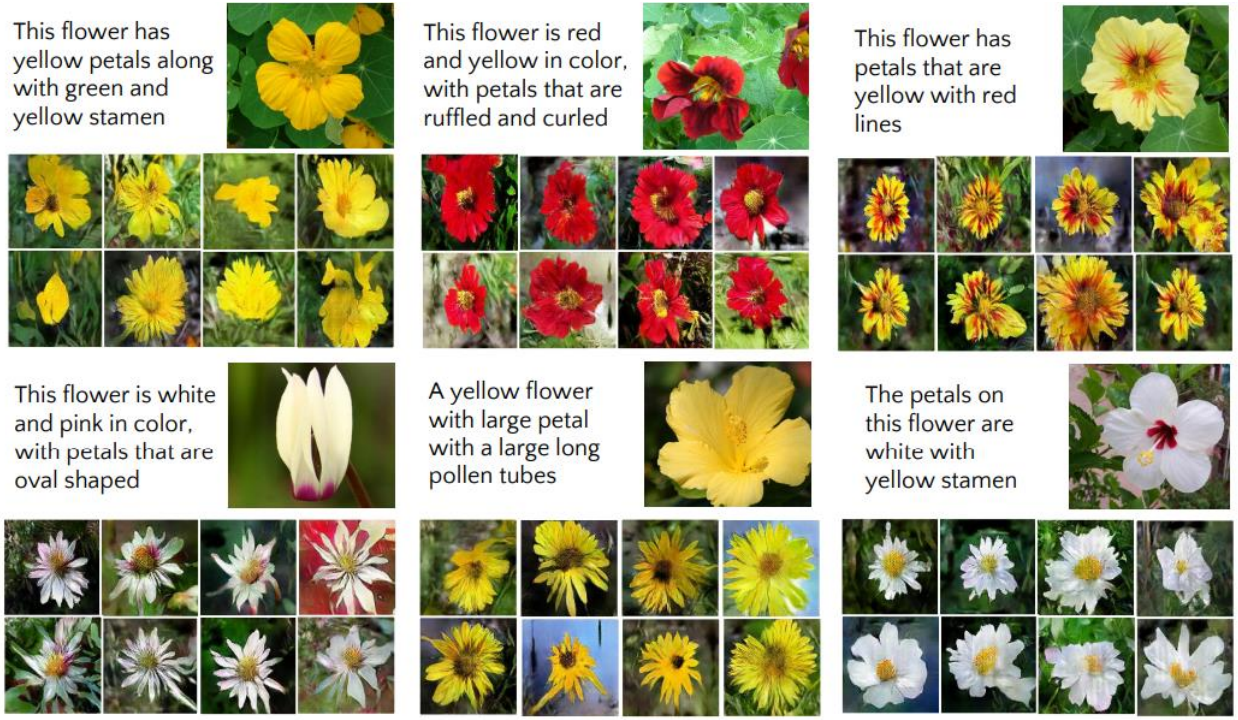


Vanilla GAN

cGAN

84.pdf

Text Conditioned Auxiliary Classifier GAN



https://arxiv.org/abs/1703.06412

Figure 3. Images synthesized from text descriptions using different noise vectors. In each block, the images at the bottom are generated from the text embeddings of the image description and a noise vector. The image on the top of each block are real images corresponding to the text description.

Conditional GAN + L1 in paired image translation – pix2pix

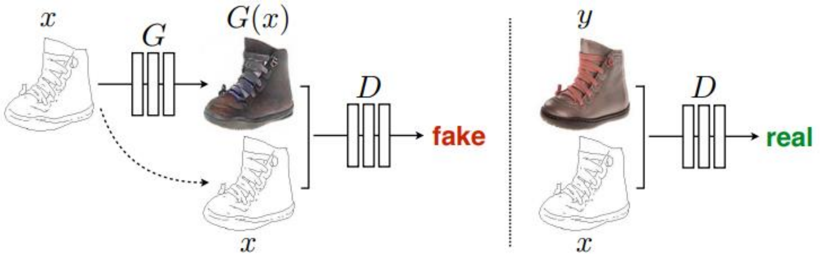
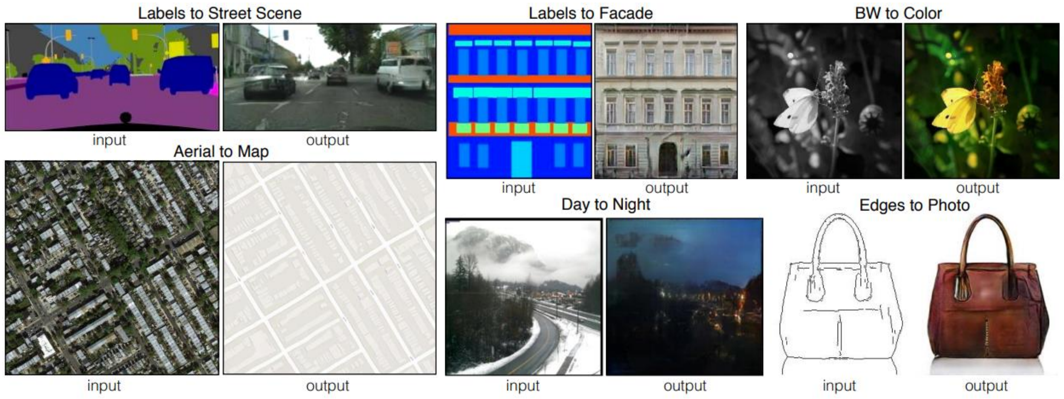
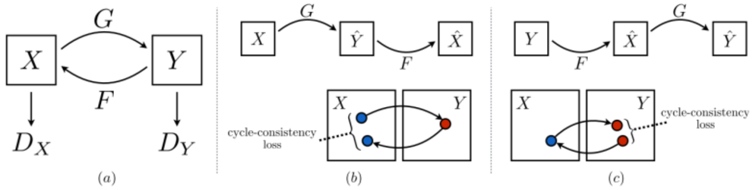


Image as G input, and as condition label.

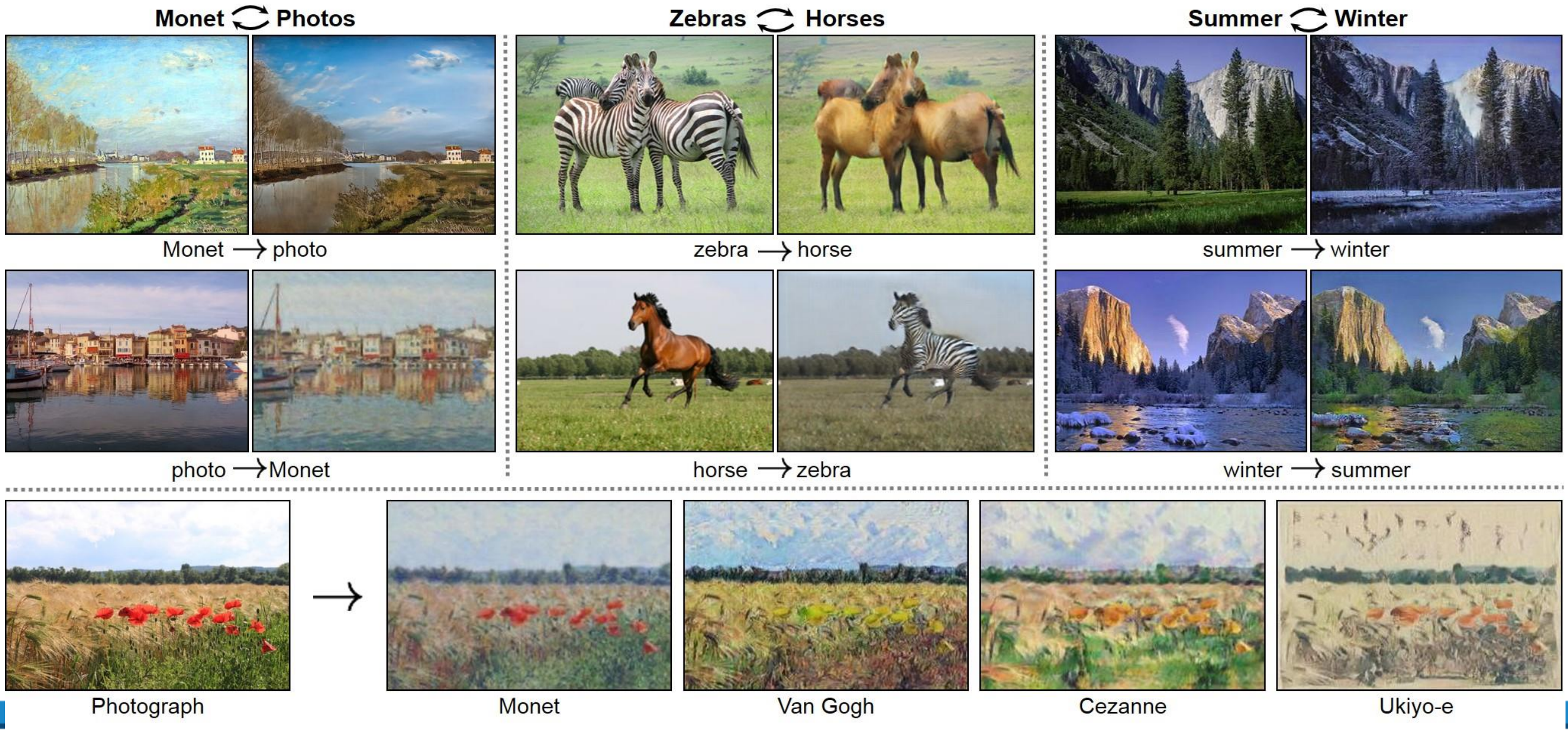


Interactive Demo: https://affinelayer.com/pixsrv/

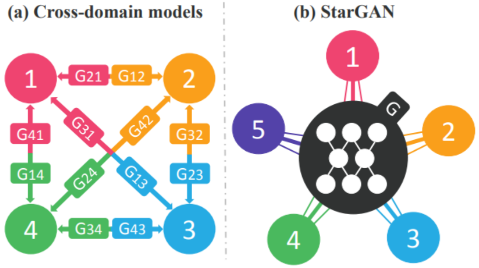
Cycle-GAN unpaired image translation



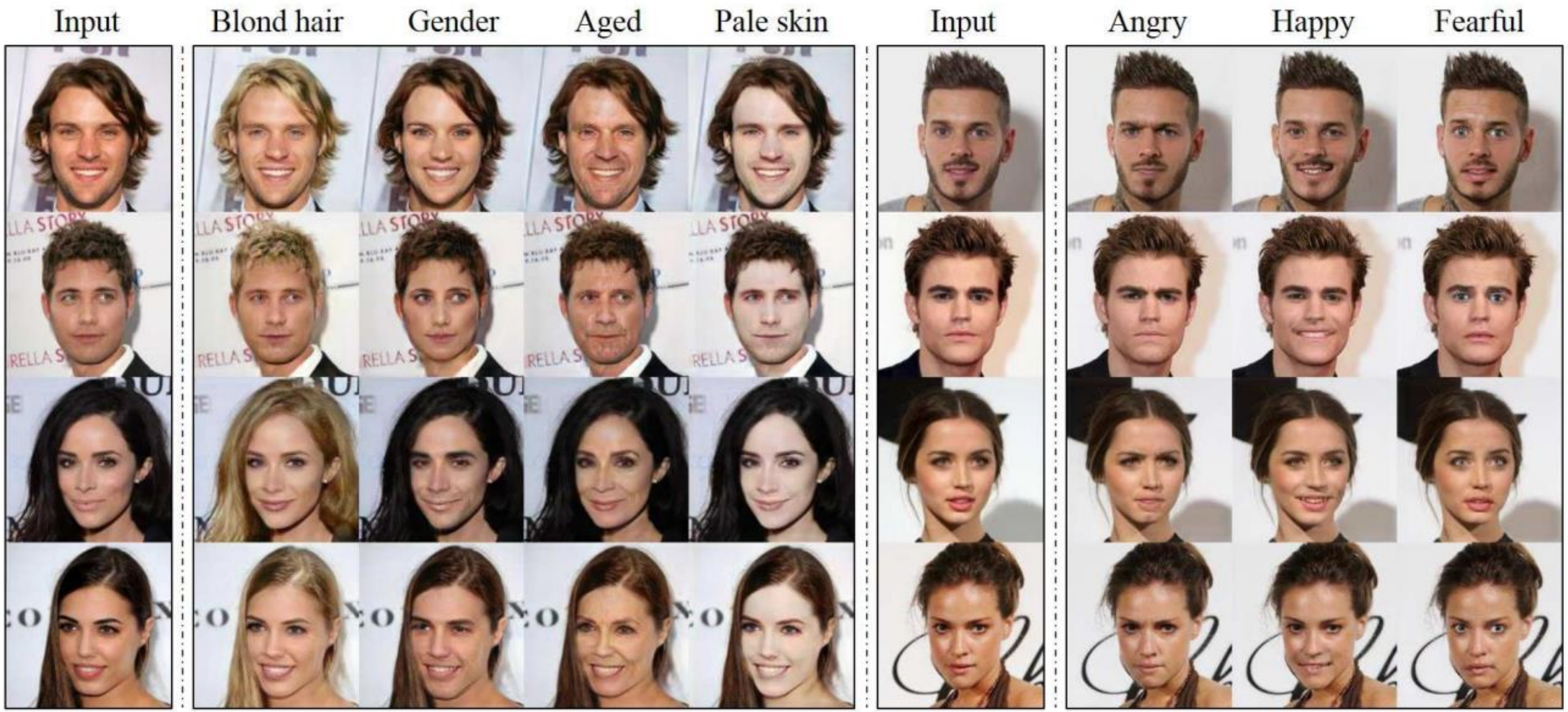
https://junyanz.github.io/CycleGAN/



StarGAN



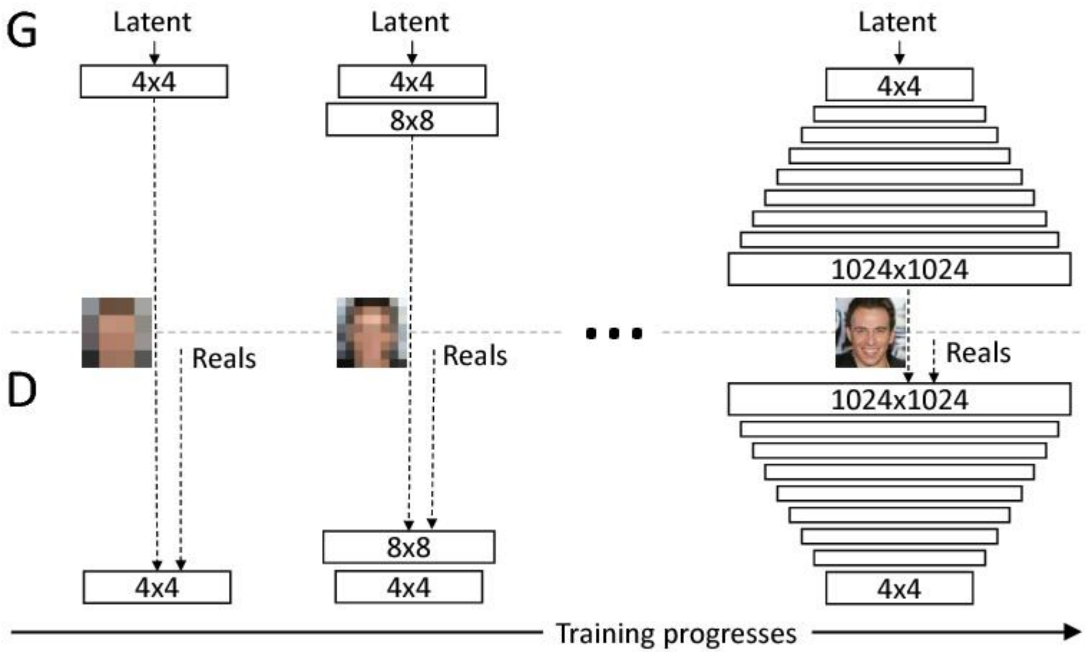
https://openaccess.thecvf.com/content_cvpr_2018/papers/Choi_StarGAN_Unified_Generative_CVPR_2018_paper.pdf



Progressively growing of GANs

https://arxiv.org/abs/1710.10196

Logic of multi-scale optimisations



Summary

- Generative model
- Adversarial loss
- Theory of GANs
- How to train GANs
- Flavours of GANs

Thanks for your attention.