Chapter 11

Generative Adversarial Networks (GAN)

Neural networks and deep learning

Supervised learning

Data: $(\boldsymbol{x}, \boldsymbol{y})$
$\boldsymbol{x}$ is data, $\boldsymbol{y}$ is label

Goal: Learn a *function* to map $\boldsymbol{x} \rightarrow \boldsymbol{y}$

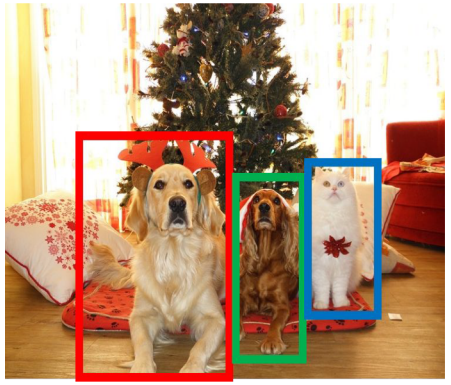
Examples:

Classification,
regression
object detection
semantic segmentation,
image captioning, etc.

→ **Cat**

Classification

Applications of supervised learning



DOG, DOG, CAT

Object Detection

GRASS, CAT,
TREE, SKY

Semantic Segmentation

A cat sitting on a suitcase on the floor

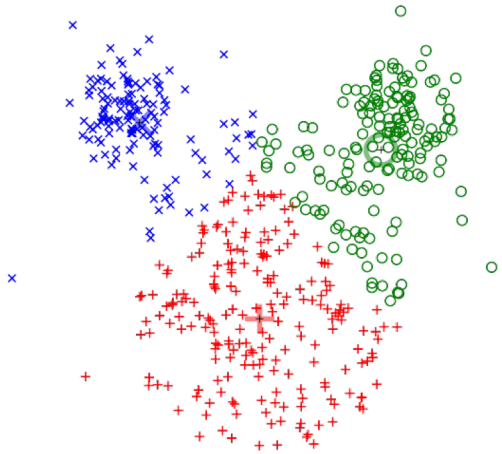
Image captioning

Unsupervised learning

Data: x
Just data, no labels!

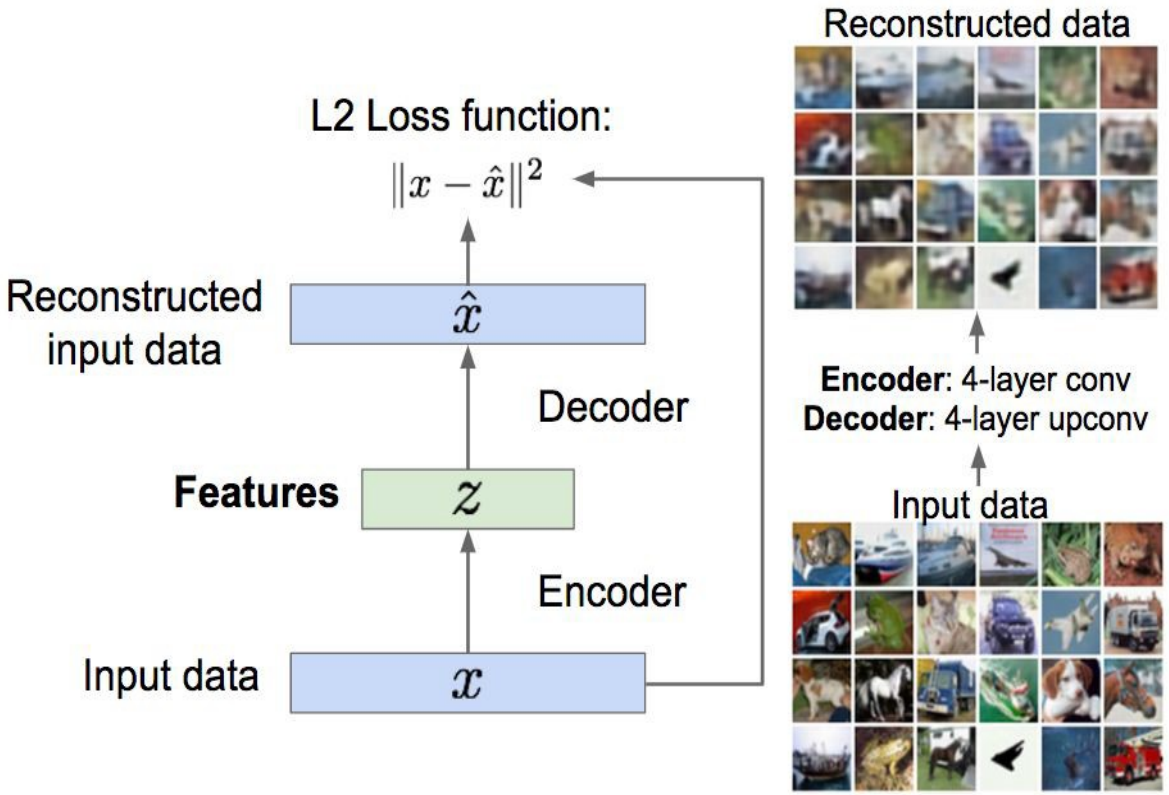
Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, latent feature learning, density estimation, etc.



K-means clustering

Unsupervised learning: Autoencoders



Autoencoders
(Feature learning)

Why Generative Models?

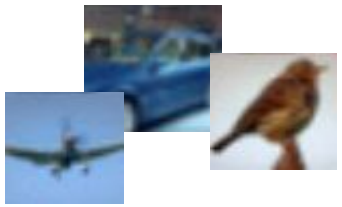
- **We've only seen *discriminative mod*els so far**
  - Given an data $\boldsymbol{x}$, predict a label $\boldsymbol{y}$
  - Estimates $p(\boldsymbol{y}|\boldsymbol{x})$

- **Discriminative models have several key limitations**
  - Can't model $p(\boldsymbol{x})$, i.e., the probability of seeing a certain data
  - Thus, can't sample from $p(\boldsymbol{x})$, i.e., **can't generate new data**

- ***Generative models* (in general) cope with all of above**
  - Can model $p(\boldsymbol{x})$
  - Can generate new data or images

Generative models

Given training data, generate new samples from same distribution



Training data $\sim p_{data}(\boldsymbol{x})$

Generated samples $\sim p_{model}(\boldsymbol{x})$

Want to learn $p_{model}(\boldsymbol{x})$ similar to $p_{data}(\boldsymbol{x})$

Addresses density estimation, a core problem in unsupervised learning

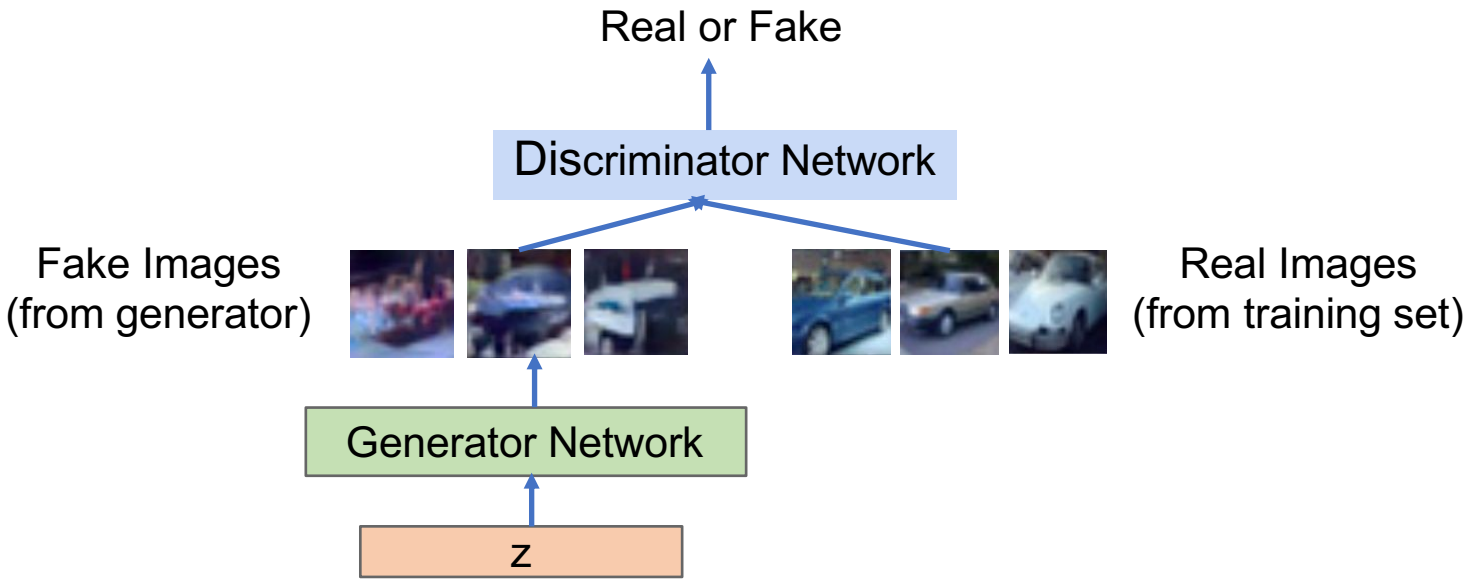
Generative Adversarial Networks (GAN)

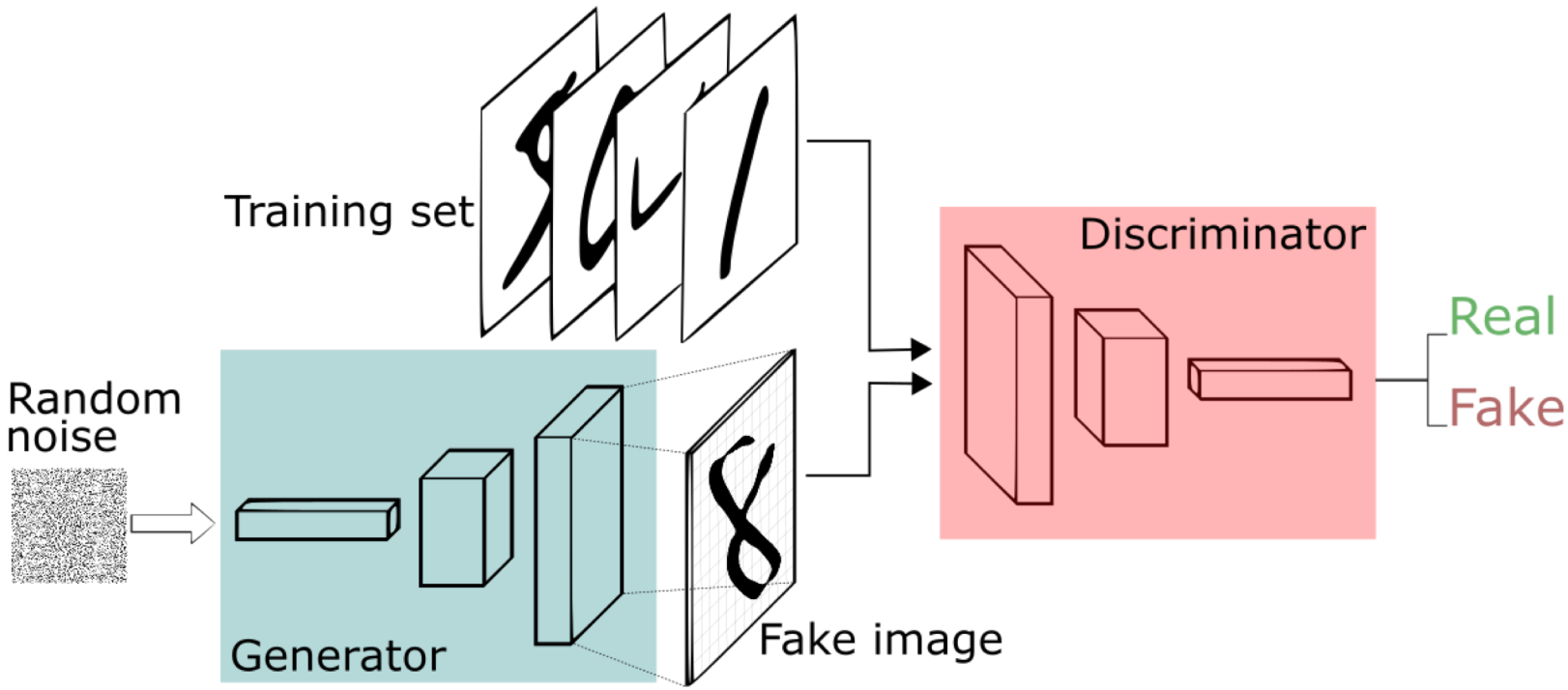
- **Generative**
  - Learn a generative model - that generates samples mimicking a source.
- **Adversarial**
  - Trained in an adversarial setting – two networks compete with each other
  - Two networks: Generator (G) and Discriminator (D)
- **Networks**
  - Use Deep Neural Networks

GANs: Two player game

Generator network: generates new data and try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake (generated) images




Training set
Discriminator
Real
Fake
Random
noise
Generator
Fake image

GAN's min-max game

Given the reward V, the *min-max game* between D and G is formulated as

$$\min_G \max_D V(D,G)$$

In the **minimax game** where:

- The Discriminator is trying to maximize its reward $V(D,G)$
- The Generator is trying to minimize Discriminator's reward

Generator: this net generates data to be as good as source data

Discriminator: this net tries to tell the difference between the real data and fake data; this is a two-class classifier.

GAN's adversarial process

$$\min_G \max_D V(D,G)$$

where the reward is defined as

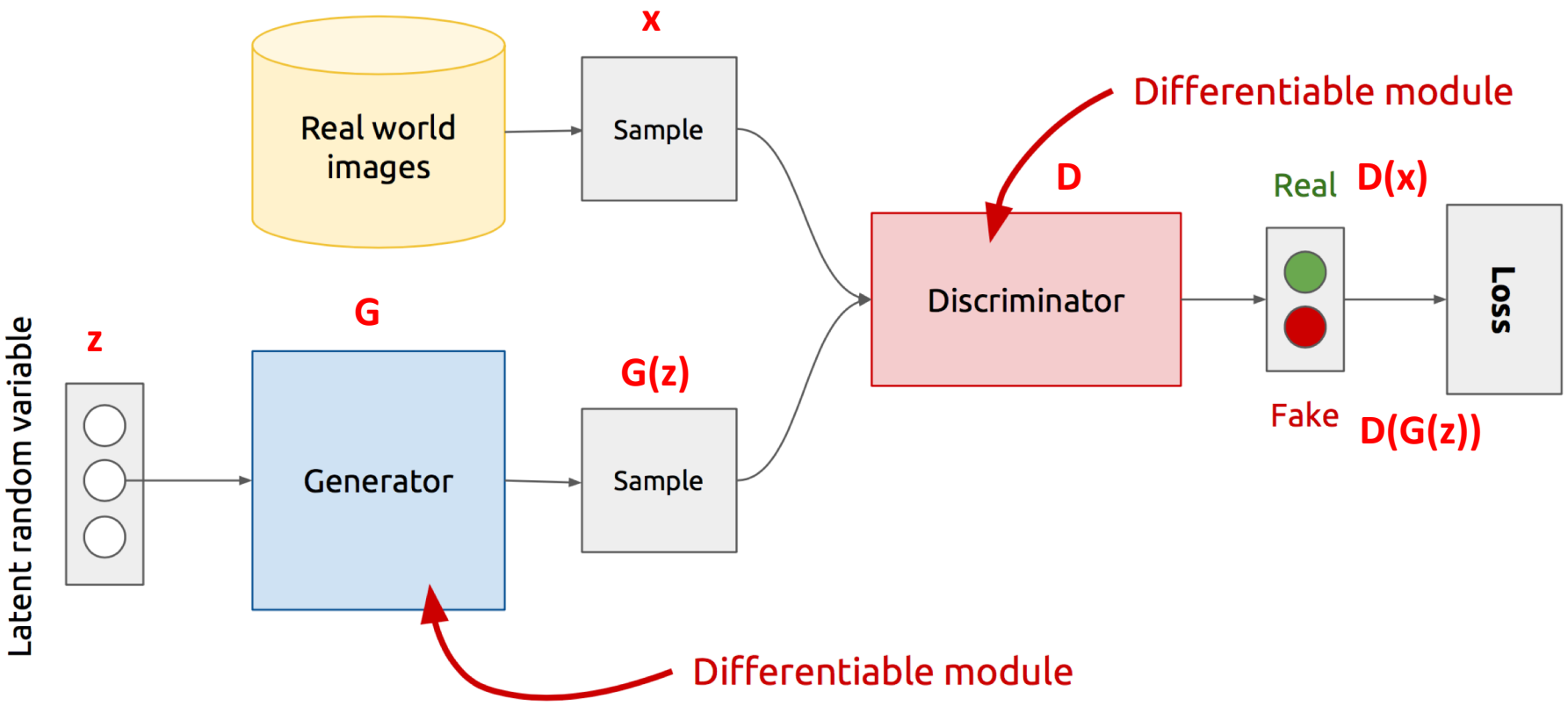
$$V(D,G) = E_{x\sim p_{data}(x)}\{\log D(x)\} + E_{z\sim p_g(z)}\{\log(1 - D(G(z)))\}$$

Compare with:

$$J = -\frac{1}{P}\sum_{p=1}^{P} d_p \, log\left(f(u_p)\right) + (1 - d_p) log\left(1 - f(u_p)\right)$$

Note that $E_{x\sim p(x)}\{f(x)\}$ denotes the *expectation* of function $f(x)$ over $x \sim p(x)$

GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the data.

https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training GAN

The cost function J_D of the discriminator is given by

$$J_D = -E_x\{log(D(x))\} - E_z\left\{log\left(1 - D(G(z))\right)\right\}$$

where x and z denote source samples and noise, respectively.
The first term on RHS is the cost of D seeing x as good samples.
The second term on RHS is the cost of D seeing samples $G(z)$ generated by noise z as fake samples.

The cost J_G of the generator is given by

$$J_G = -E_z\left\{log\left(D(G(z))\right)\right\}$$

This is the cost of D seeing generated samples as good samples.

Training GAN

In training, the minmax game between G and D are implemented by minimizing both J_D and J_G in each epoch.

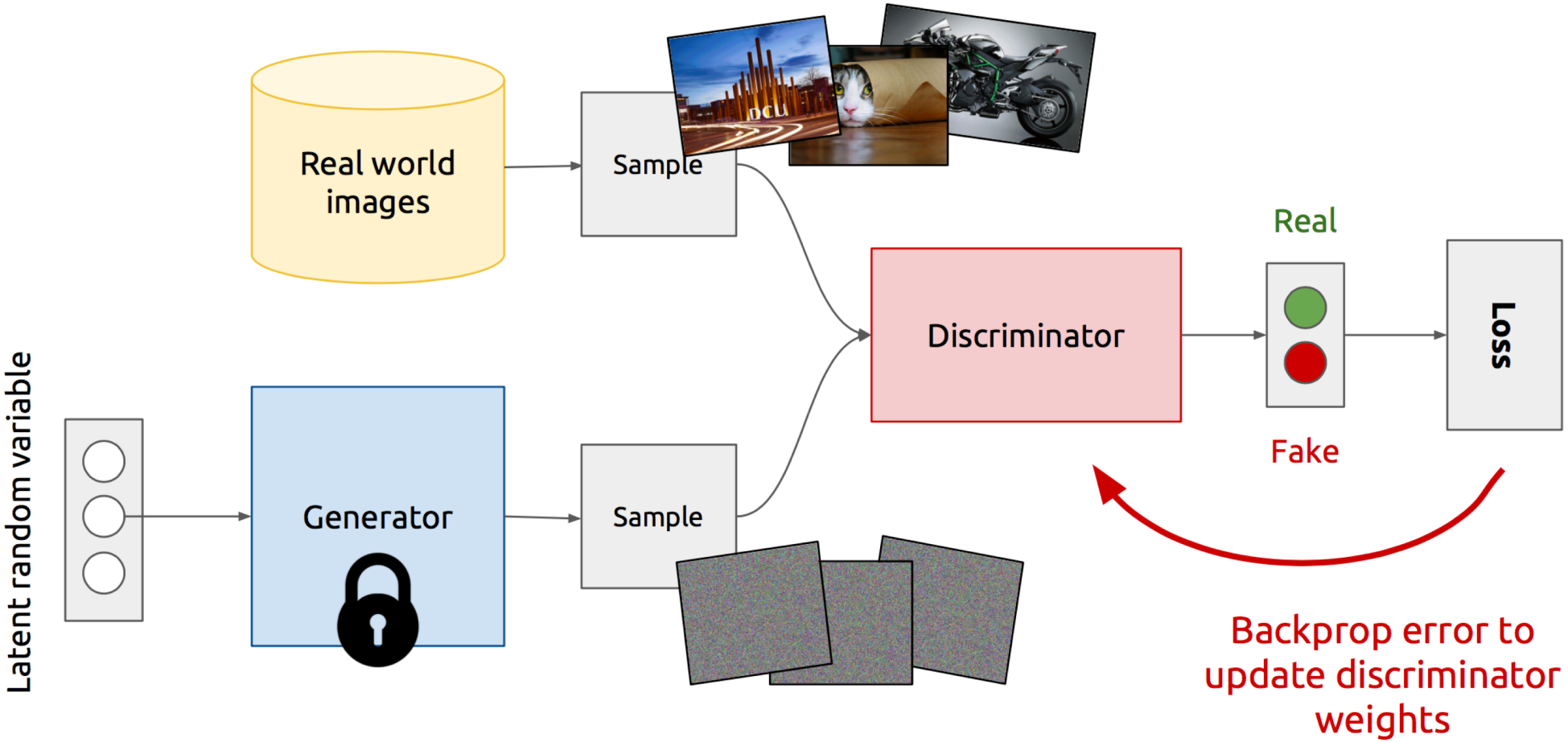
Iterate until convergence:

minimize J_D (on discriminator weights)

minimize J_G (on generator weights)

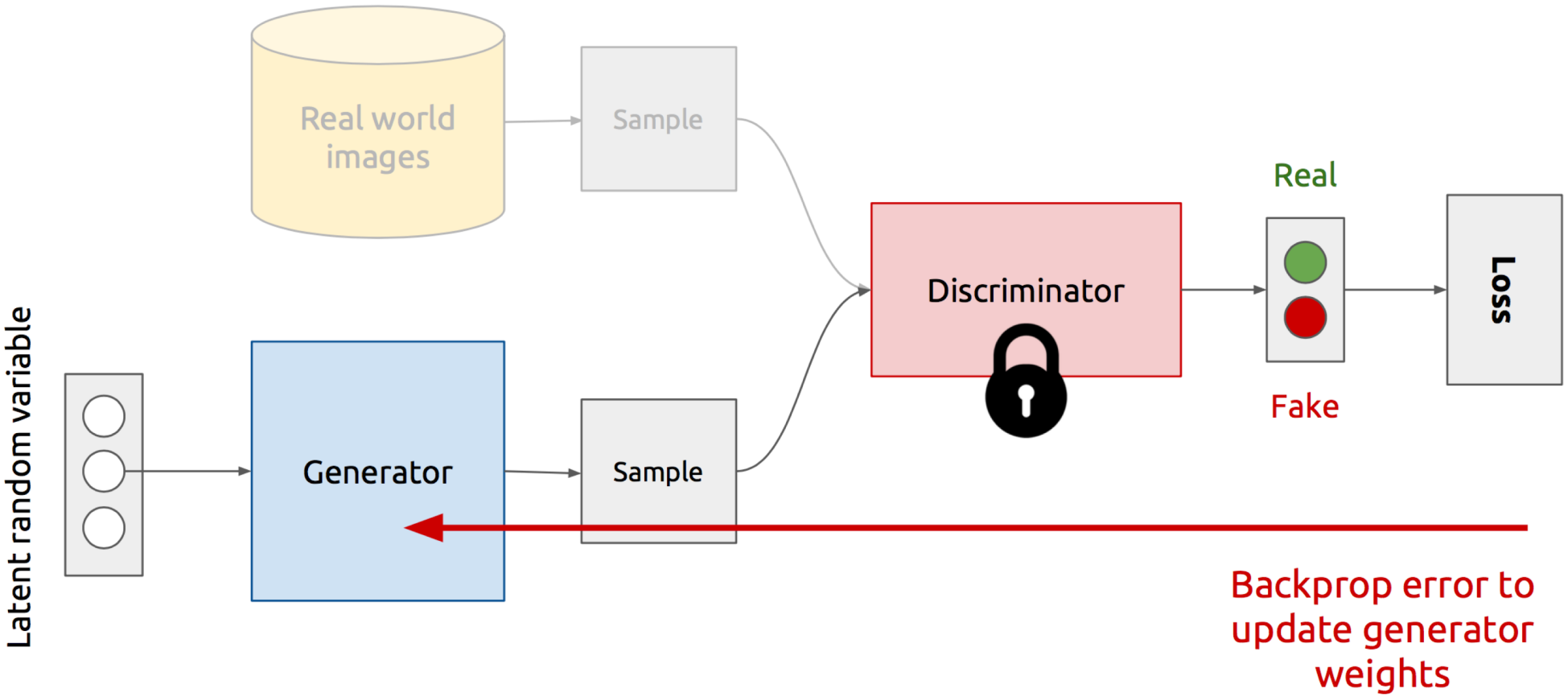
That is, the generator (G) attempts to generate samples as good as source samples and the discriminator (D) attempts classify source samples as true samples and generator samples as false samples.

Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Optimality of GAN cost

$$J_D = -E_x\{log(D(x))\} - E_z\left\{log\left(1 - D(G(z))\right)\right\}$$
$$= -\sum_x p_{data}(x)\, log(D(x)) - \sum_z p_z(z)\, log\left(1 - D(G(z))\right)$$
$$= -\sum_x p_{data}(x)\, log(D(x)) + p_g(x) log(1 - D(x))$$

When J_D is optimum:

$$\frac{\partial J}{\partial D(x)} = -\frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0$$
$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$
$$J_D{}^* = -\sum_x p_{data}(x)\, log\left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right) + p_g(x) log\left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right)$$

Optimality of GAN cost

For $p_g = p_{data}$:

$$D^*(x) = \frac{1}{2}$$

$$J_D{}^* = -\sum_x p_{data}(x)\, log\left(\frac{1}{2}\right) + p_g(x) log\left(\frac{1}{2}\right) = -log(4)$$

This optimum is known as the *Nash equilibrium*.

Optimality of GAN cost

KL divergence:

$$KL(p||q) = \sum_x p(x)\, log\left(\frac{p(x)}{q(x)}\right)$$

$$KL\left(p_{data}||\frac{p_g + p_{data}}{2}\right) = \sum_x p_{data}(x)\, log\left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right) + log(2)$$

$$J_D{}^* = -log4 + KL\left(p_{data}||\frac{p_g + p_{data}}{2}\right) + KL\left(p_g||\frac{p_g + p_{data}}{2}\right)$$
$$= -log4 + JS(p_{data}||p_g)$$

Optimality of GAN cost

$$J_D{}^* = -log4 + JS(p_{data}||p_g)$$

Where $JS(p_{data}||p_g)$ is the **Jensen-Shannon divergence** between the data generation process and the model's distribution:

$$JS(p_{data}||p_g) = KL\left(p_{data}||\frac{p_g + p_{data}}{2}\right) + KL\left(p_g||\frac{p_g + p_{data}}{2}\right)$$

JS divergence between two distributions is always non-negative and zero only when they are equal.

The minimum value of $J_D{}^* = -\log(4)$ when the generative model perfectly replicating the data generating process. The GAN learning minimizes JS divergence between the source data and the data generated by the generator.

Example 1

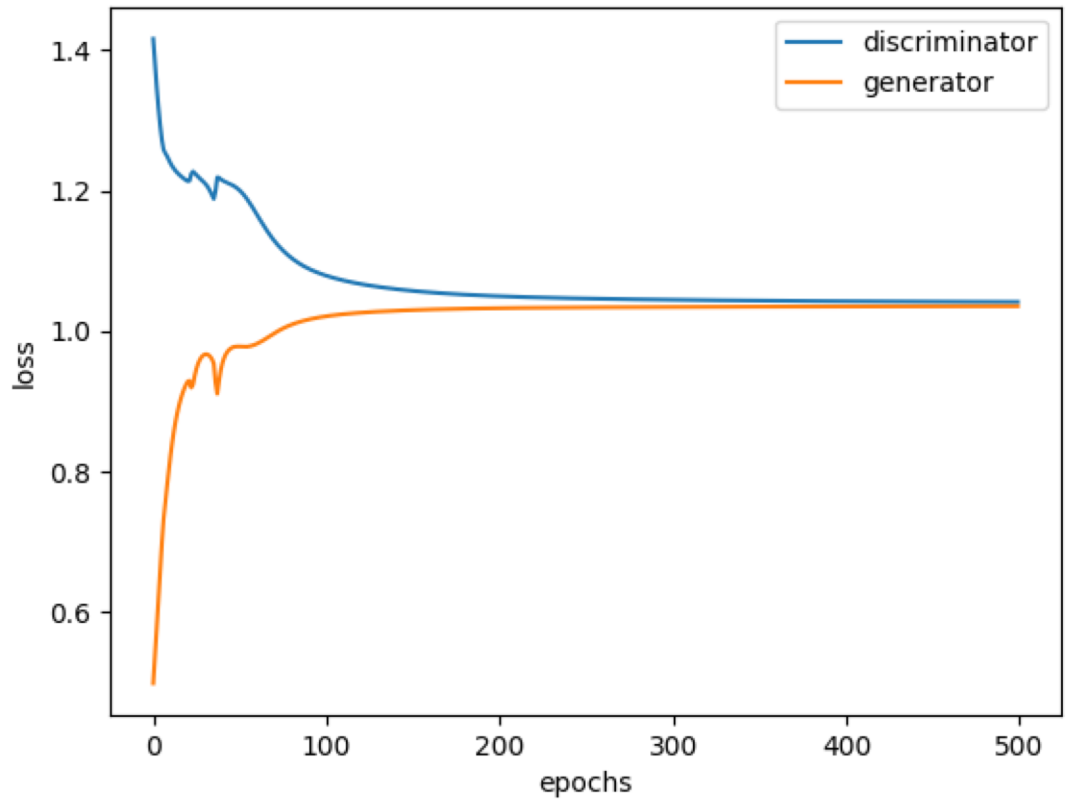
Design a GAN to learn Gaussian distributed data with mean = 0 and standard deviation = 1.0.
The generator receives uniformly distributed 1-dimensional inputs in the range [-1, +1].

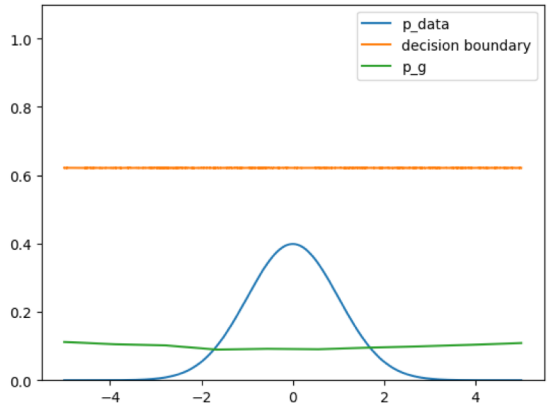
Discriminator is a 4-layer DNN:

- No of inputs = 1
- No of neurons in hidden-layer 1 = 5 (sigmoid)
- No of neurons in hidden-layer 2 = 5 (sigmoid)
- No of outputs = 1 (logistic neuron for two-class classification)

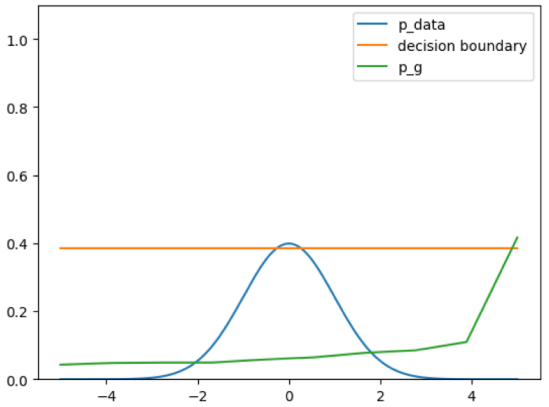
The generator is a 4-layer DNN:

- No of inputs = 1
- No of neurons in hidden-layer 1 = 5 (sigmoid)
- No of neurons in hidden-layer 2 = 5 (sigmoid)
- No of outputs = 1 (tanh unit for regression)

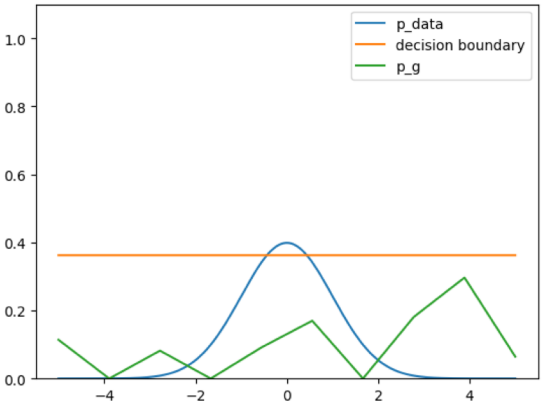

discriminator
generator
1.4
1.2
1.0
0.8
0.6
loss
0
100
200
300
400
500
epochs



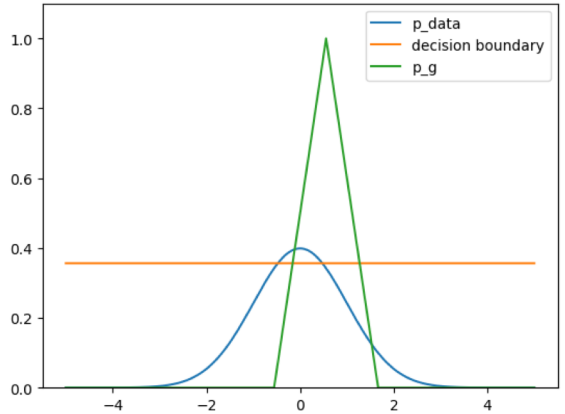
1st Iteration



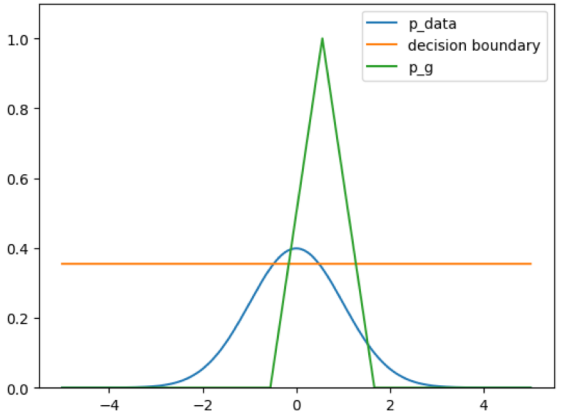
40th Iteration



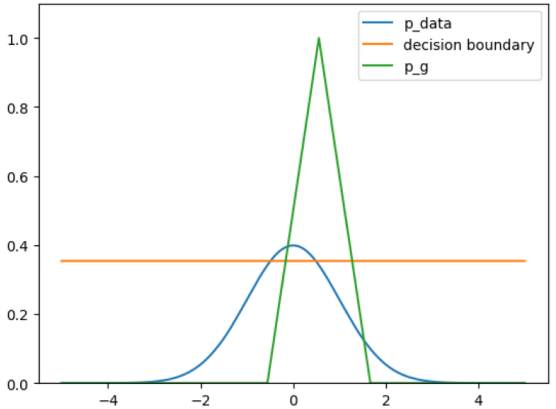
80th Iteration



120th Iteration



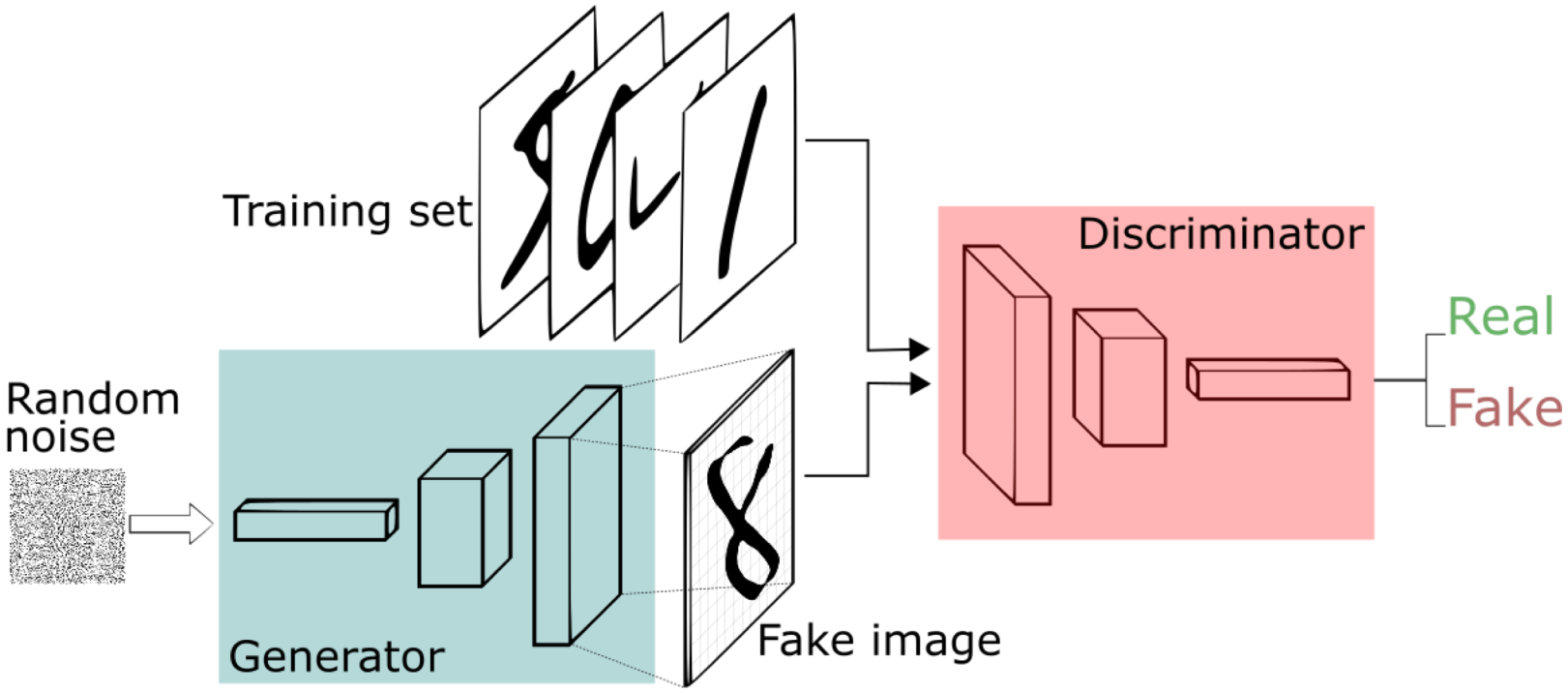
160th Iteration



200th Iteration

Example 2

Design a GAN to generate MNIST images from a uniformly distributed noise vector of 100 dimensions.



Generator:

Input 100 nodes, random noise drawn from a uniform distribution

Hidden layer: 100 ReLU units

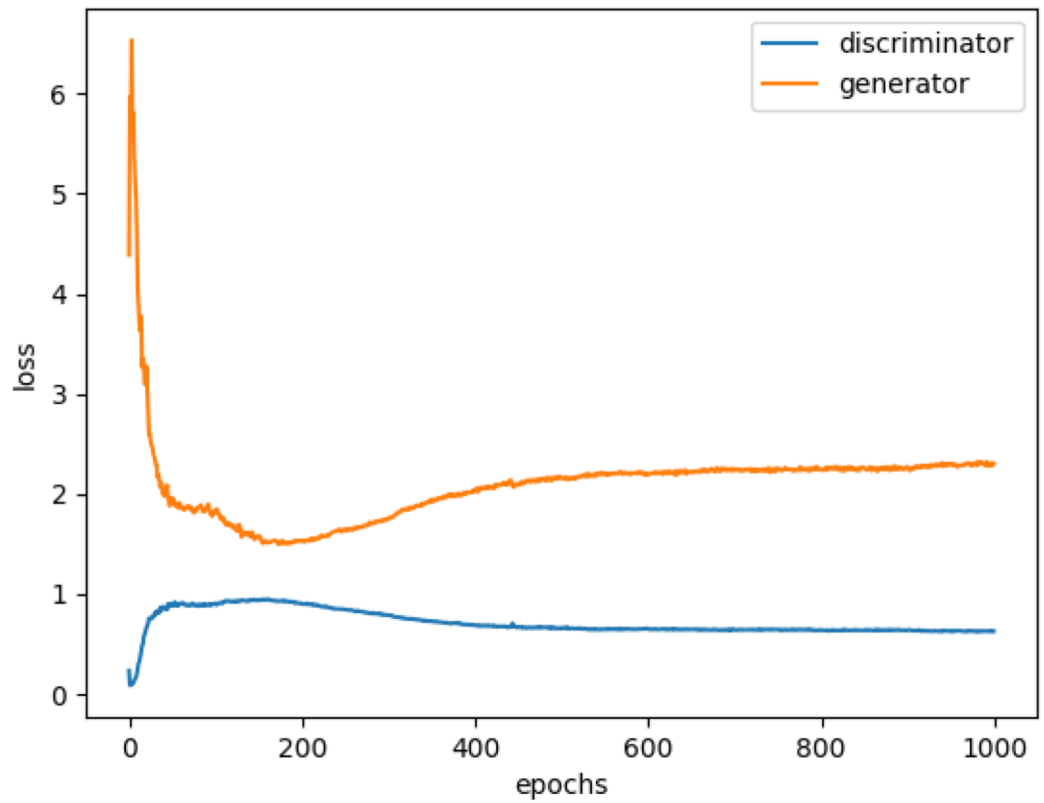
Output layer: 28x28 sigmoid units

Discriminator:

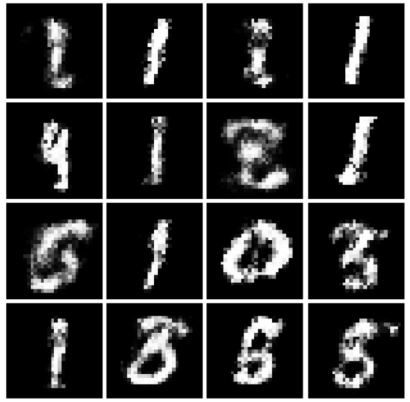
Input 28x28 units

Hidden layer: 128 ReLU units

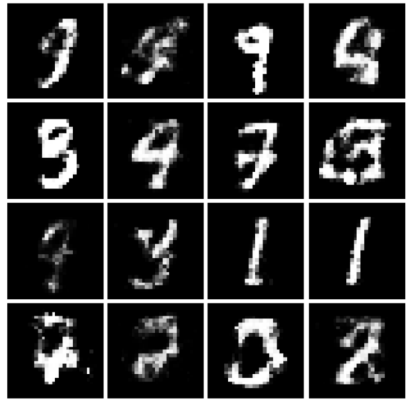
Output layer: one logistic neurons


discriminator
generator
loss
epochs
0
1
2
3
4
5
6
0
200
400
600
800
1000

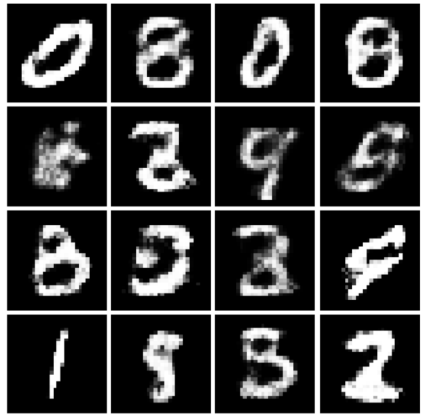
1st Iteration



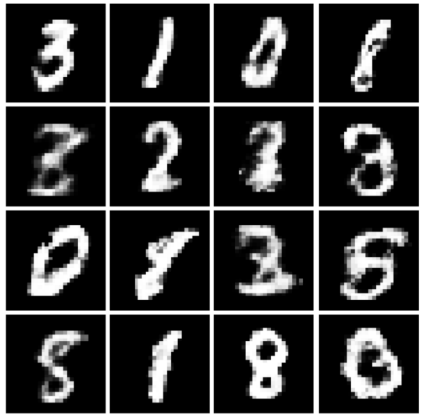
40th Iteration



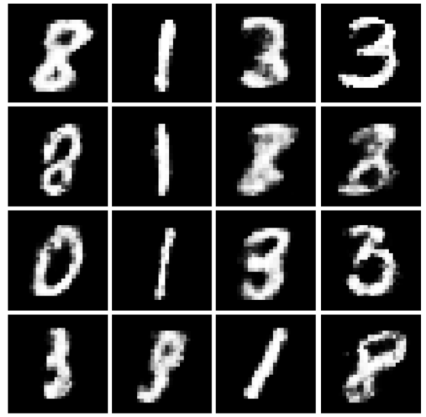
80th Iteration



160th Iteration



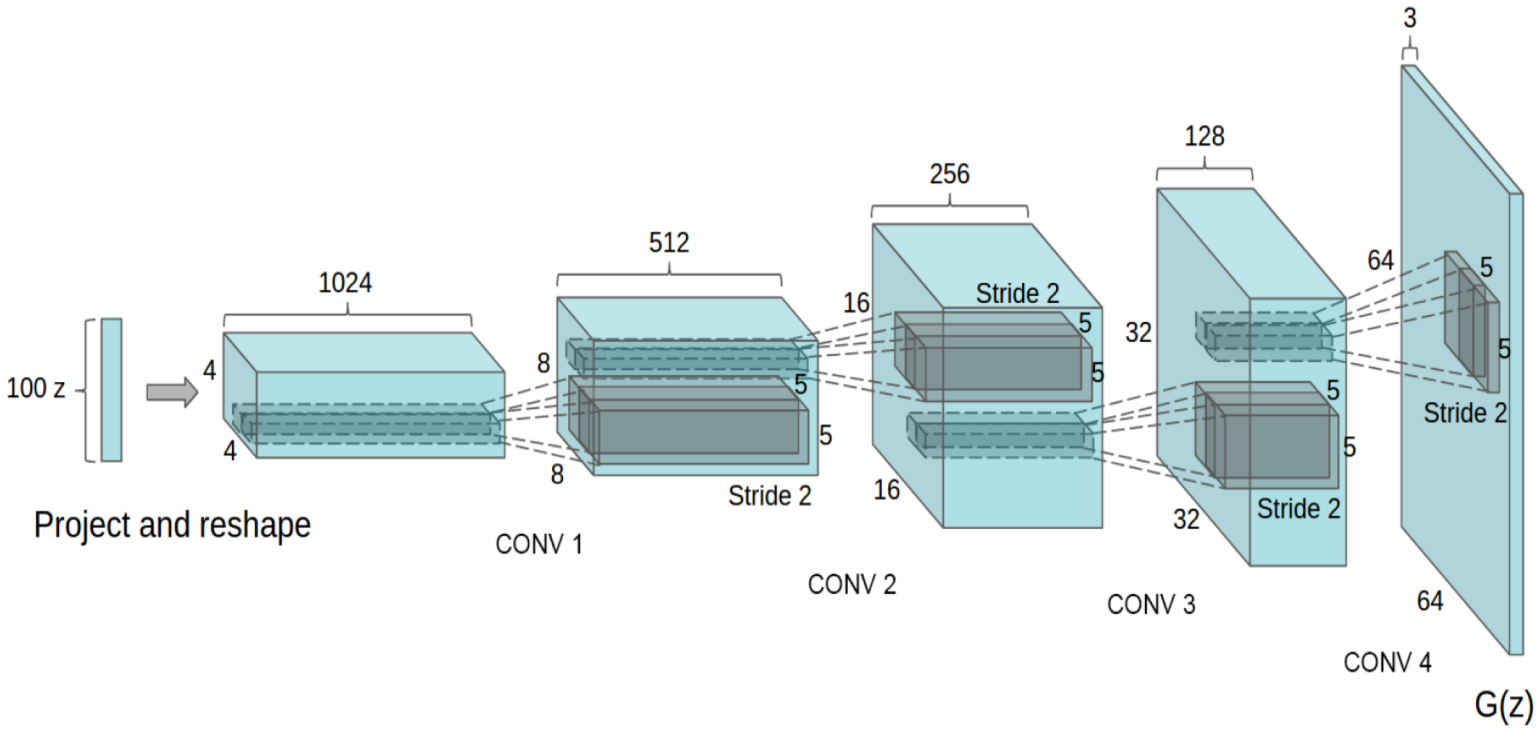
320th Iteration



1000th Iteration

Deep Convolutional GANs (DCGANs)

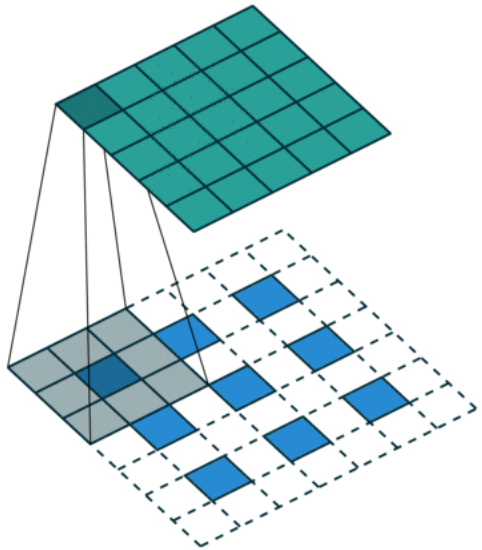
Generator Architecture



Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).



Up-sampling by fractional striding
Input 3x3, output = 5x5
stride =1convolution window = 3 x 3 (with respect to output)

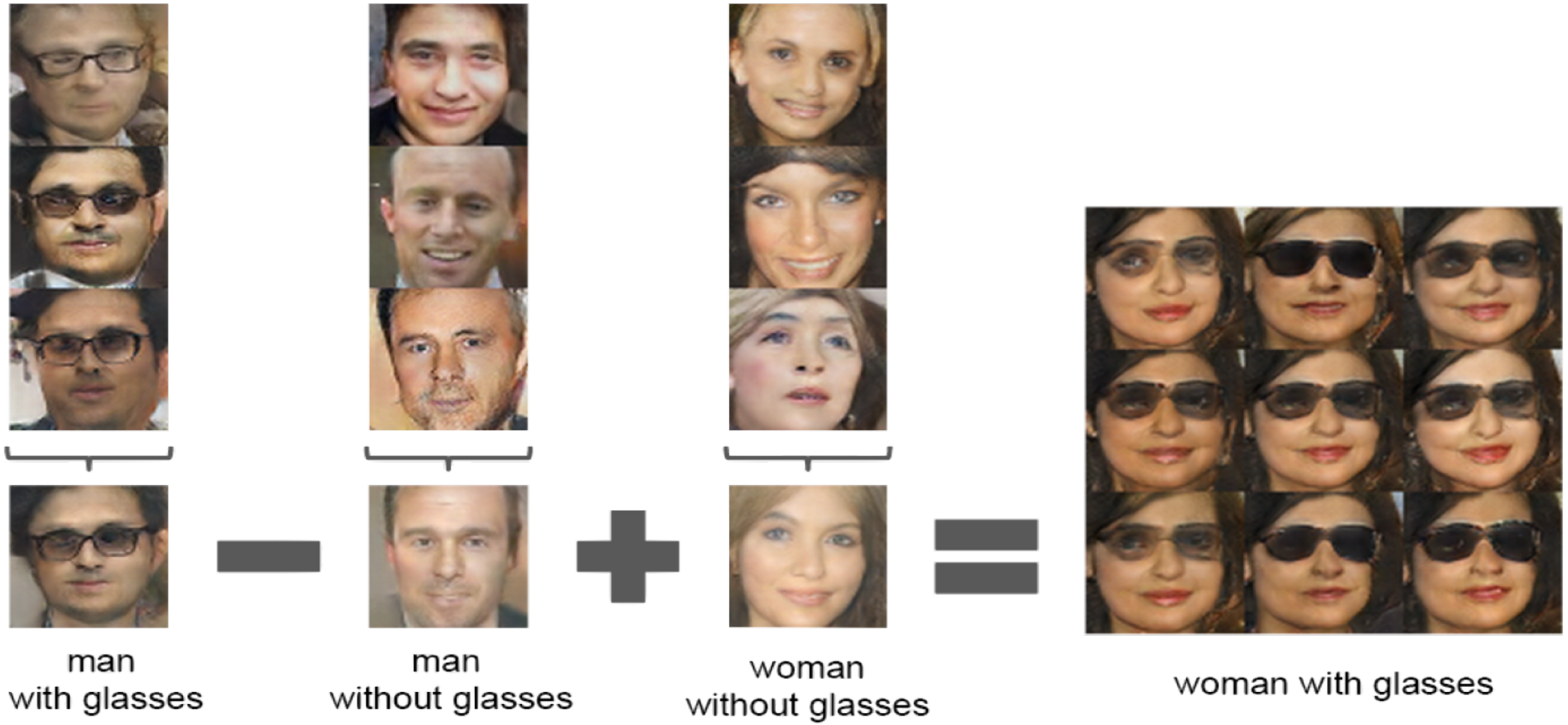
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

DCGAN: Bedroom images

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

Latent vectors capture interesting patterns…



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

Advantages of GANs

- **Plenty of existing work on Deep Generative Models**
  - Boltzmann Machine
  - Deep Belief Nets
  - Variational AutoEncoders (VAE)

- **Why GANs?**
  - Sampling (or generation) is straightforward.
  - Training doesn't involve Maximum Likelihood estimation.
  - Robust to Overfitting since Generator never sees the training data.
  - Empirically, GANs are good at capturing the modes of the distribution.

Problems with GANs

- **Probability Distribution is Implicit**
  - Not straightforward to compute $p(\boldsymbol{x})$
  - Thus **Vanilla GANs** are only good for Sampling/Generation.

- **Training is Hard**
  - Non-Convergence
  - Mode-Collapse

Training Problems

- **Non-Convergence**
- Mode-Collapse

Non-convergence

- **Deep Learning models (in general) involve a single player**
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - SGD has convergence guarantees (under certain conditions).
  - **Problem:** With non-convexity, we might converge to local optima.

$$\min_G L(G)$$

- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

  - SGD was not designed to find the Nash equilibrium of a game.
  - **Problem:** We might not converge to the Nash equilibrium at all.

Non-Convergence Example

$$\min_{x} \max_{y} V(x, y)$$

Let $V(x, y) = xy$

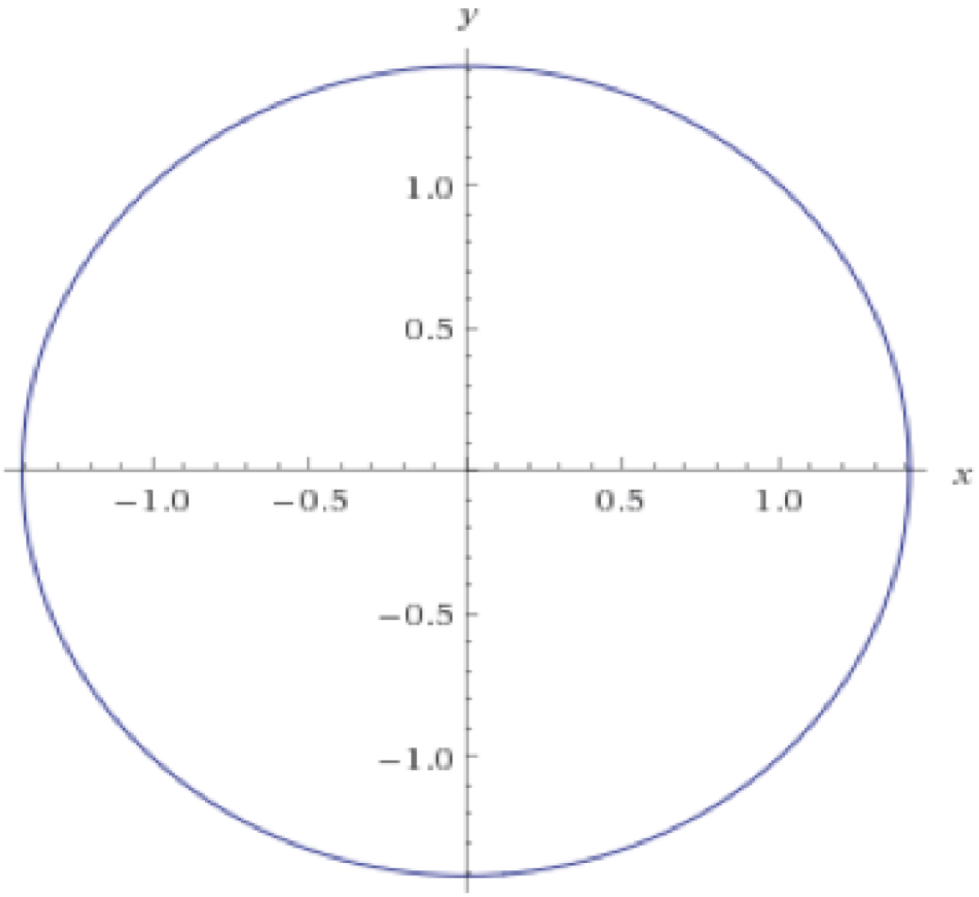
• State 1:	x > 0	y > 0	V > 0	Increase y	Decrease x
• State 2:	x < 0	y > 0	V < 0	Decrease y	Decrease x
• State 3:	x < 0	y < 0	V > 0	Decrease y	Increase x
• State 4 :	x > 0	y < 0	V < 0	Increase y	Increase x
• State 5:	x > 0	y > 0	V > 0	Increase y	Decrease x

== State 1

Non-Convergence Example

$$\min_{x} \max_{y} \; xy$$

- $\frac{\partial}{\partial x} = -\mathrm{y} \quad \ldots \quad \frac{\partial}{\partial y} = \mathrm{x}$
- $\frac{\partial^2}{\partial y^2} = \frac{\partial}{\partial x} = -y$
- Differential equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge



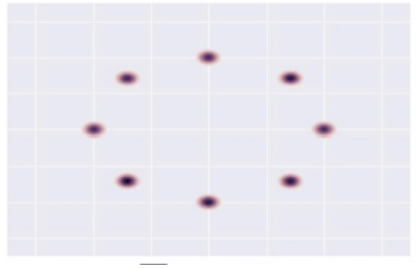
Problems with GANs

- Non-Convergence
- **Mode-Collapse**

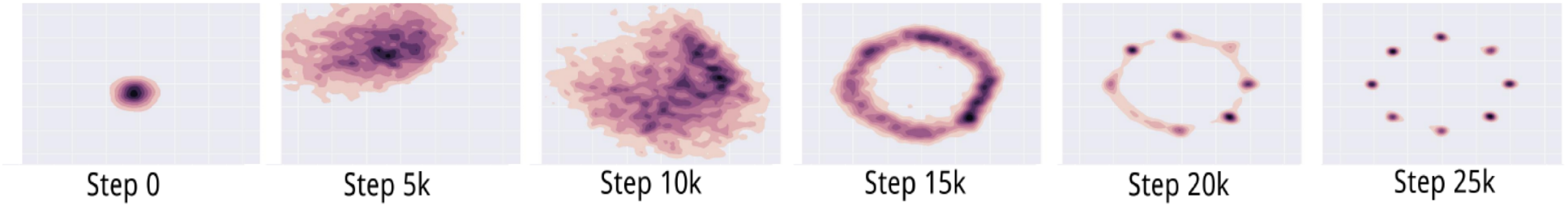
Mode-Collapse

- **Generator fails to output diverse samples**

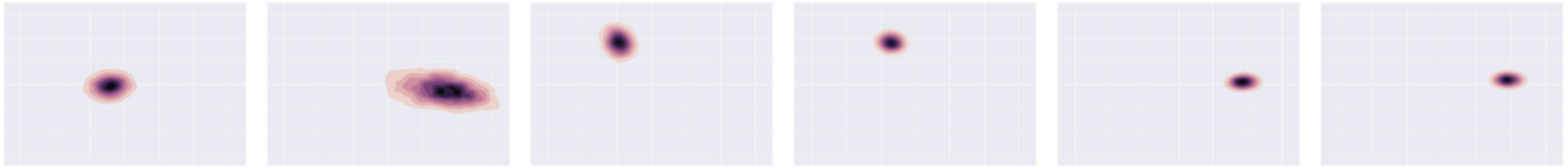
Target



Expected



Output



Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

Some Solutions

- **Mini-Batch GANs**
- **Supervision with labels**

- **Some recent attempts :-**
  - Unrolled GANs
  - W-GANs

Basic (Heuristic) Solutions

- **Mini-Batch GANs**
- Supervision with labels

How to reward sample diversity?

- **At Mode Collapse,**
  - Generator produces good samples, but a very few of them.
  - Thus, Discriminator can’t tag them as fake.

- **To address this problem,**
  - Let the Discriminator know about this edge-case.

- **More formally,**
  - Let the Discriminator look at the entire batch instead of single examples
  - If there is lack of diversity, it will mark the examples as fake

- **Thus,**
  - Generator will be forced to produce diverse samples.

Mini-Batch GANs

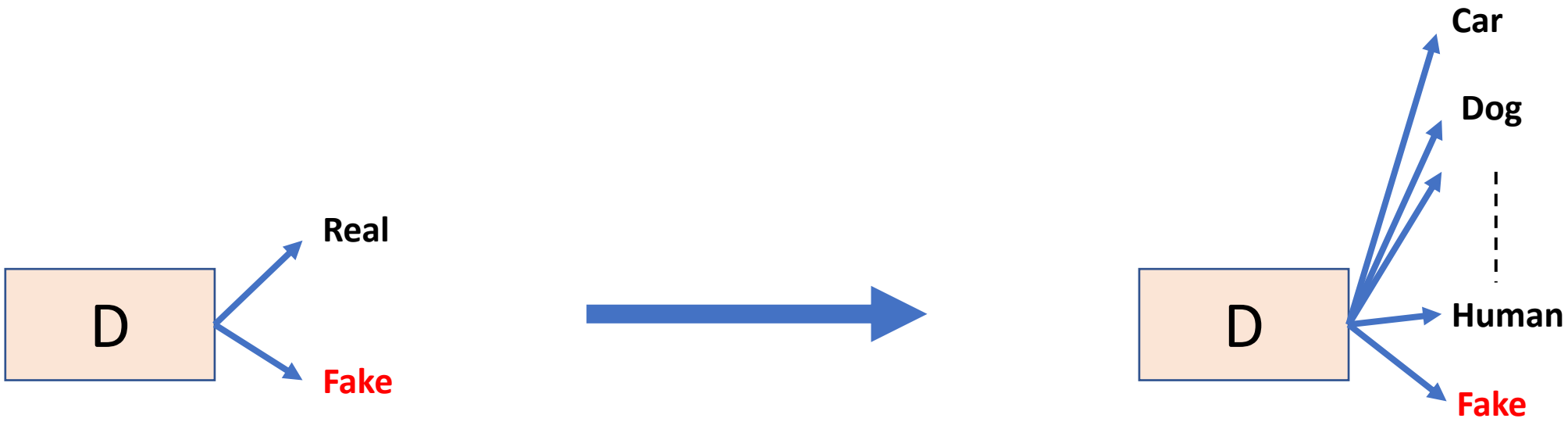
- **Extract features that capture diversity in the mini-batch**
  - For e.g. L2 norm of the difference between all pairs from the batch
  - That is, the errors at an intermediate layer for real and fake samples are minimized.
- **Feed those features to the discriminator along with the image**
- **Feature values will differ b/w diverse and non-diverse batches**
  - Thus, Discriminator will rely on those features for classification
- **This in turn,**
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches

Basic (Heuristic) Solutions

- Mini-Batch GANs
- **Supervision with labels**

Supervision with Labels

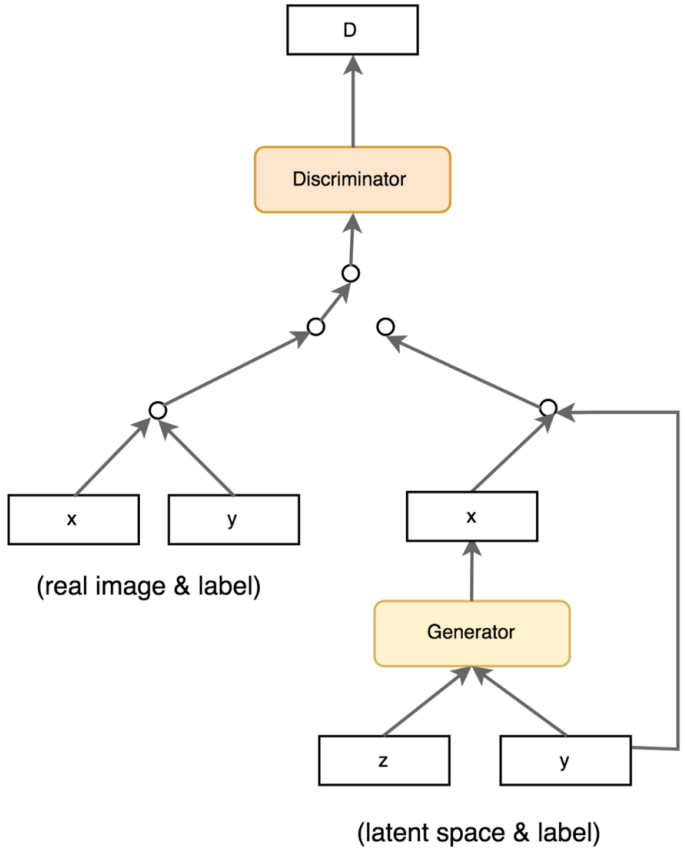
- Label information of the real data might help



- Empirically generates much better samples

Conditional GANs

Conditions the inputs to the generator and discriminator with the labels.



Conditional GANs

MNIST digits generated conditioned on their class label.

$[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$ →

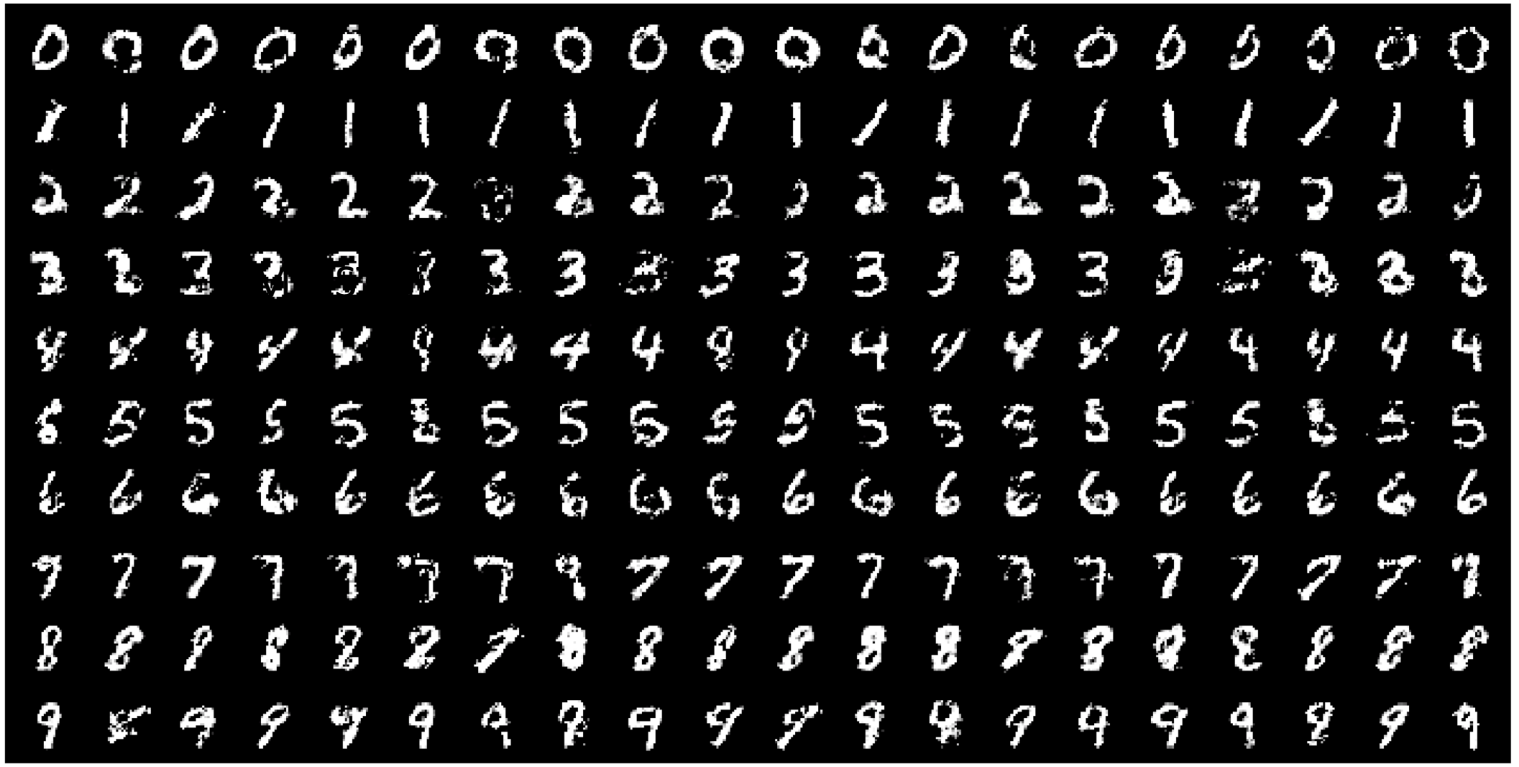


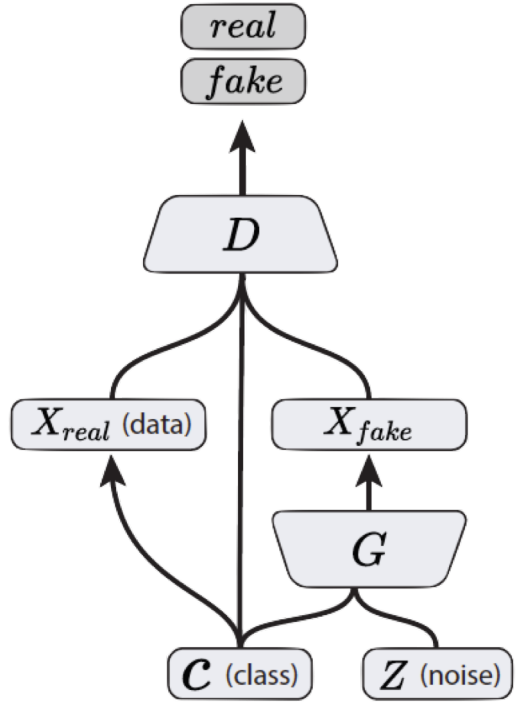
Figure 2 in the original paper.

Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).

Conditional GANs

- Simple modification to the original GAN framework that conditions the model on *additional information* for better multi-modal learning.

- Lends to many practical applications of GANs when we have explicit *supervision* available.



Conditional GAN
(Mirza & Osindero, 2014)

Image Credit: Figure 2 in Odena, A., Olah, C. and Shlens, J., 2016. Conditional image synthesis with auxiliary classifier GANs. *arXiv preprint arXiv:1610.09585*.

Mirza, Mehdi, and Simon Osindero. "**Conditional generative adversarial nets**". arXiv preprint arXiv:1411.1784 (2014).

Image-to-Image Translation

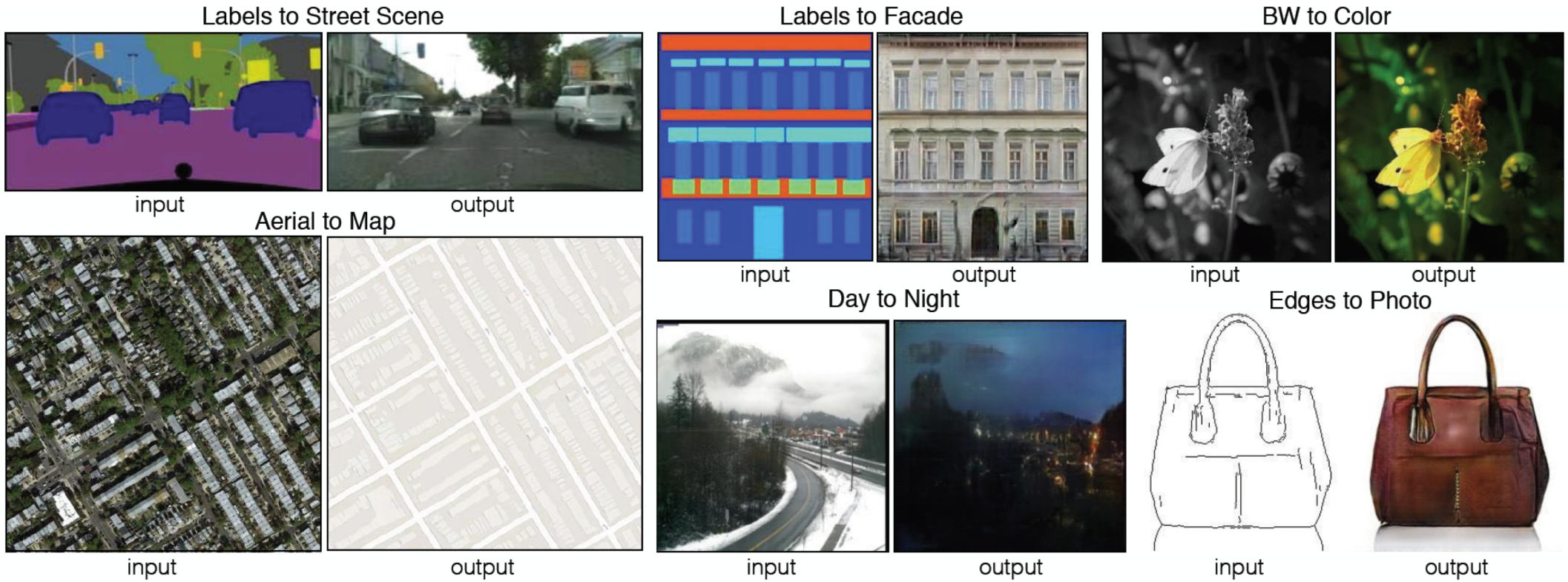


Figure 1 in the original paper.

[Link to an interactive demo of this paper]

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "**Image-to-image translation with conditional adversarial networks**". arXiv preprint arXiv:1611.07004. (2016).

Image-to-Image Translation

- Architecture: *DCGAN*-based architecture
- Training is conditioned on the images from the source domain.
- Conditional GANs provide an effective way to handle many complex domains without worrying about designing *structured loss* functions explicitly.

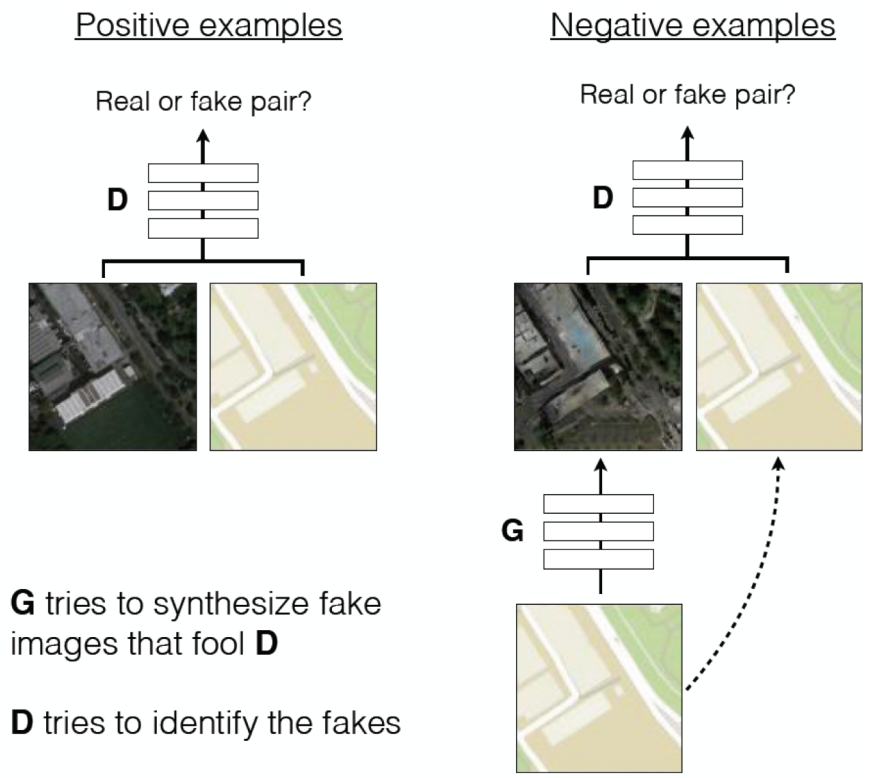


Figure 2 in the original paper.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "**Image-to-image translation with conditional adversarial networks**". arXiv preprint arXiv:1611.07004. (2016).

Text-to-Image Synthesis

Motivation

Given a text description, generate images closely associated.

Uses a conditional GAN with the generator and discriminator being condition on “dense” text embedding.

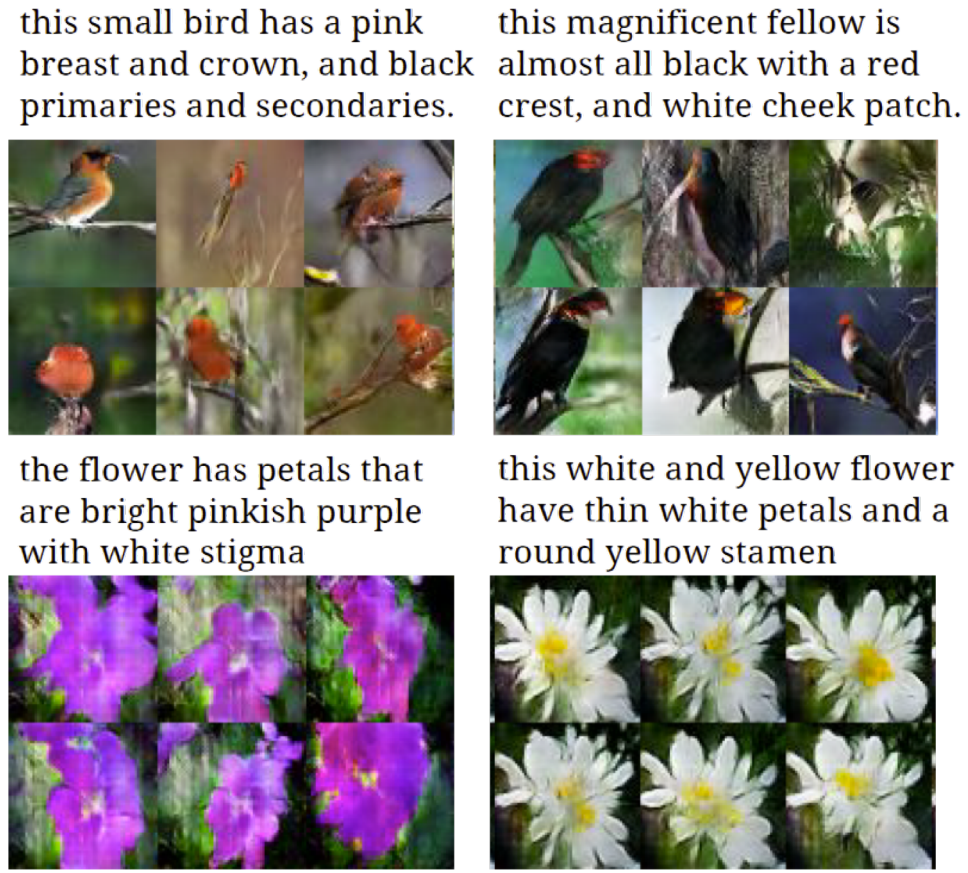


Figure 1 in the original paper.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. “**Generative adversarial text to image synthesis**”. ICML (2016).

Text-to-Image Synthesis

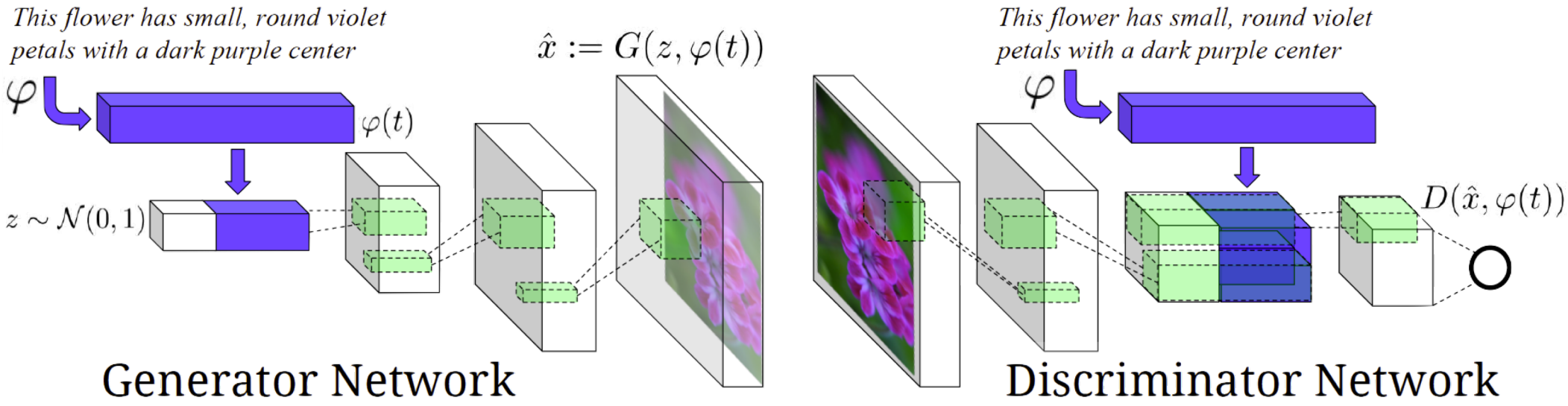


Figure 2 in the original paper.

Positive Example:

Real Image, Right Text

Negative Examples:

Real Image, Wrong Text

Fake Image, Right Text

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. "**Generative adversarial text to image synthesis**". ICML (2016).

Face Aging with Conditional GANs

- Differentiating Feature: Uses an *Identity Preservation Optimization* using an auxiliary network to get a better approximation of the latent code (z*) for an input image.
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories.

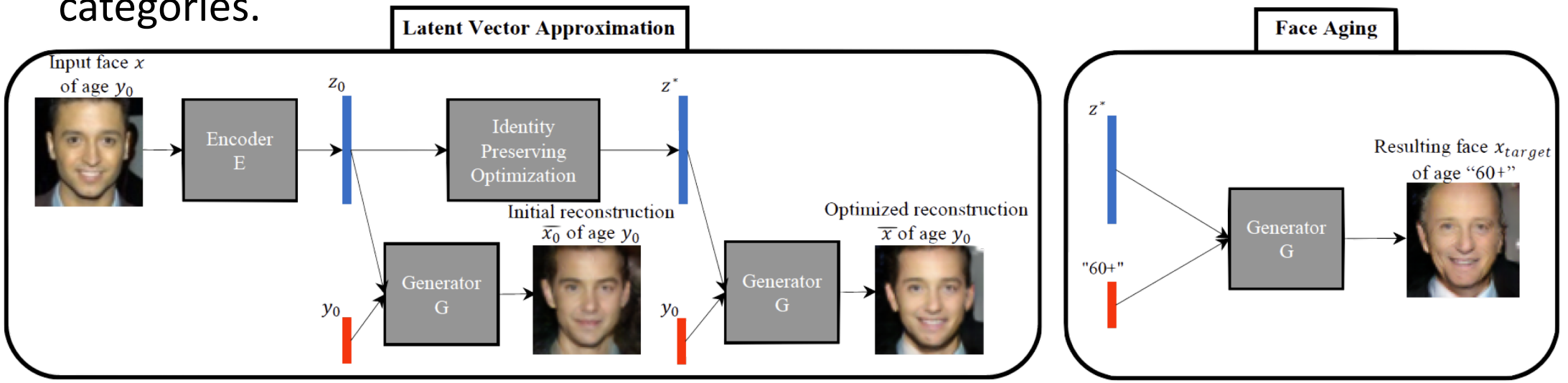


Figure 1 in the original paper.

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.

Face Aging with Conditional GANs

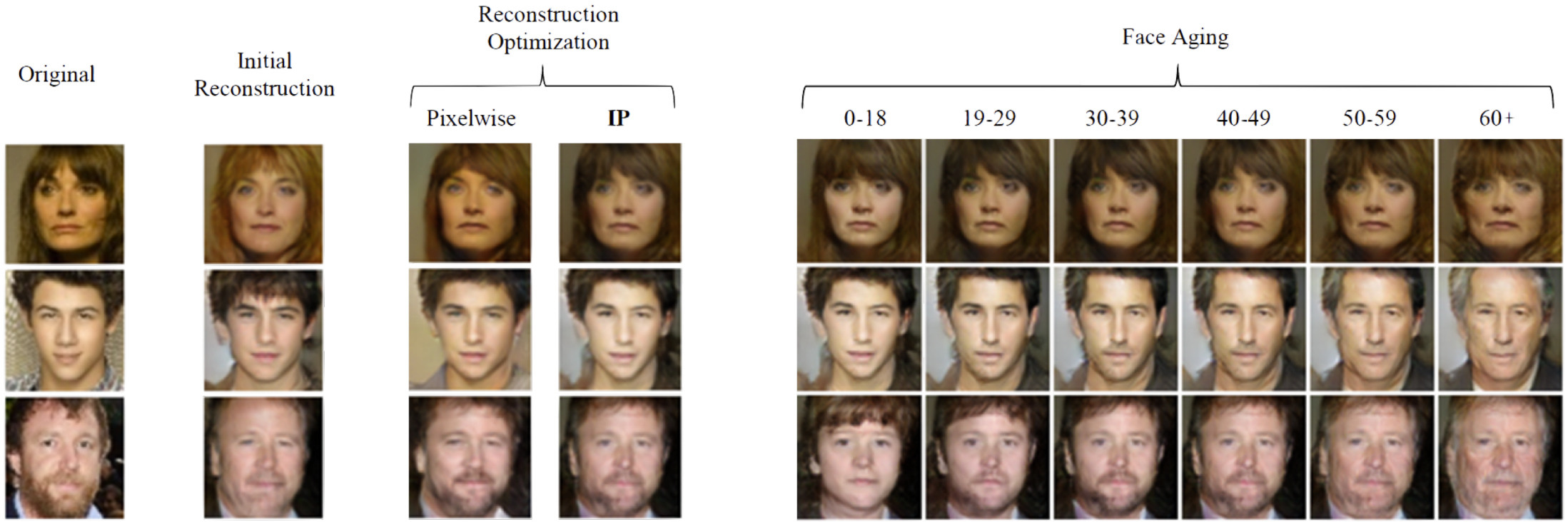


Figure 3 in the original paper.

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.