Image Generation with Generative Adversarial Networks

2017 Spring

BRI623 Pattern Recognition

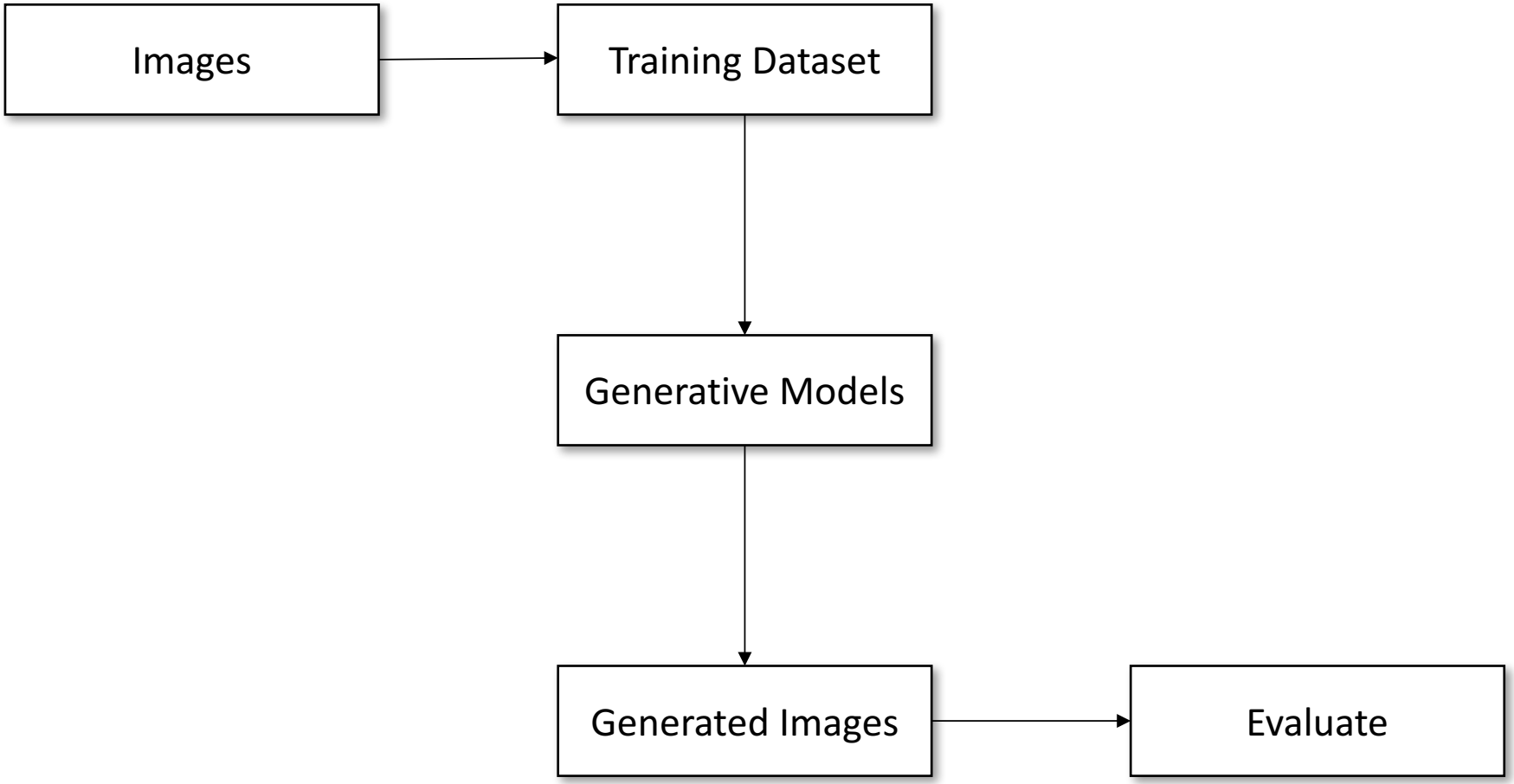
Term Project

Team 3

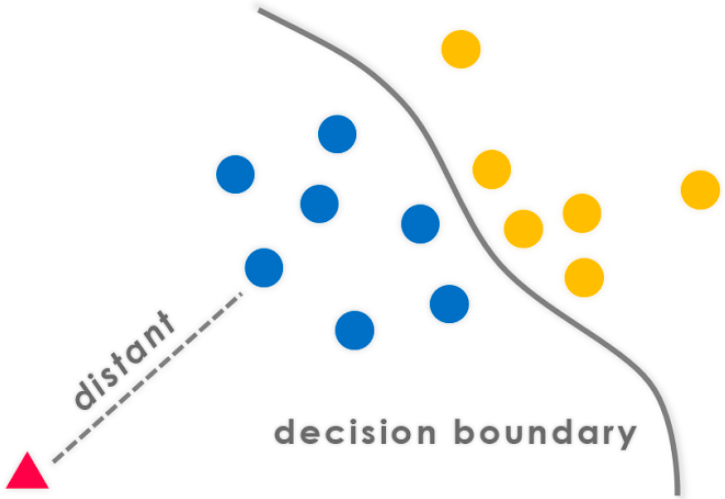
김호중, 도형록, 송채빈

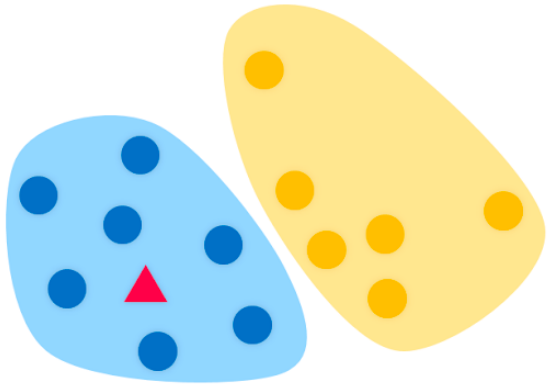
Outlines

Objective: Construct a pattern recognition system can generate images



Generative Models





Discriminative Models

$$P(y|x)$$

Generative Models

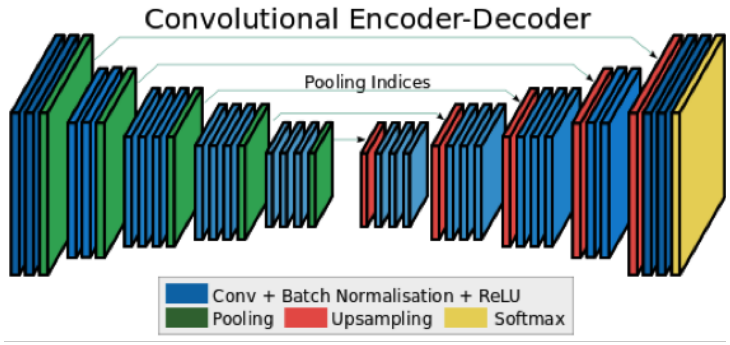
$$P(x,y)$$

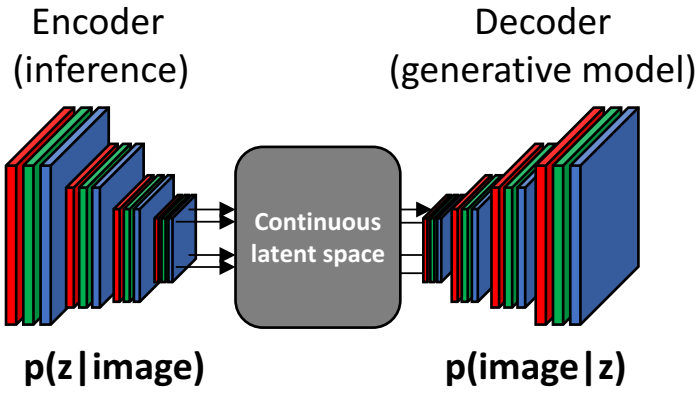
Usually generative models are more useful, but difficult to train

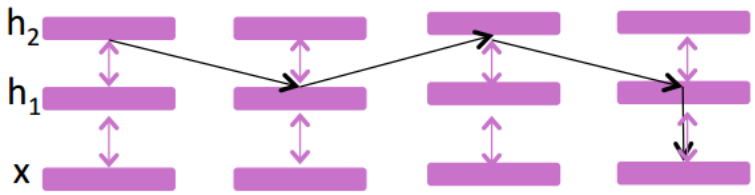
Moreover, in high-dimension a large training set is required

Generative Models for Images

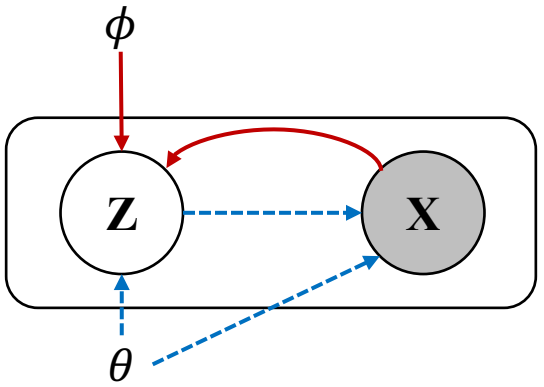
- Autoencoder + Markov chain Monte Carlo (Bengio et al, 2013)
- Variational Autoencoder (Kingma and Welling, 2014)





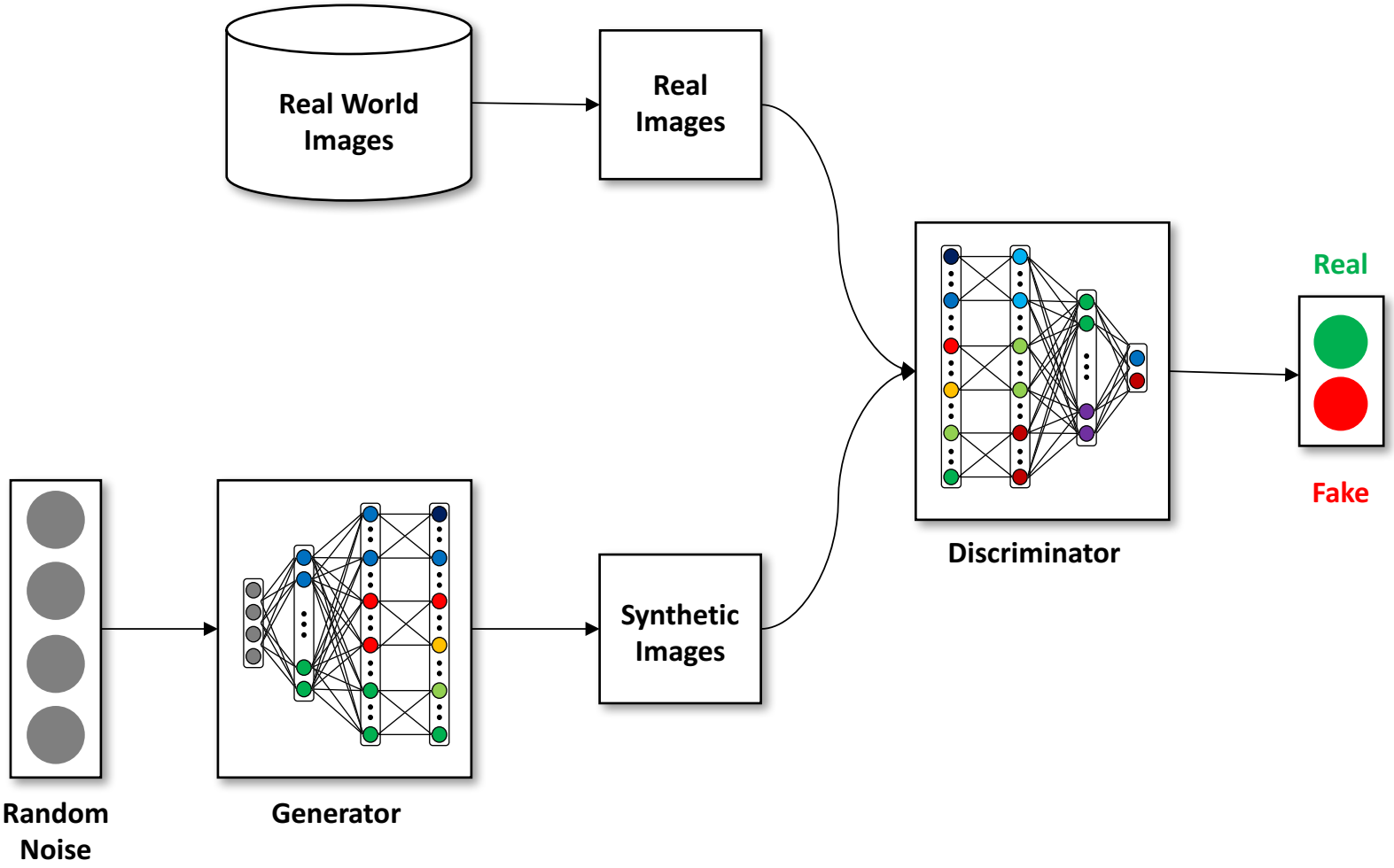


Markov chain in deep autoencoders



Generative Adversarial Networks

- Generative Adversarial Nets (Goodfellow et al, 2014)
  - Estimation of generative models via an adversarial process
  - Adversarial: generator (G), discriminator (D)



Generative Adversarial Networks

- Objective: make good counterfeit bills
- Generator(G): counterfeit maker
- Discriminator(D): counterfeit detector

Real

Fake

Generative Adversarial Networks

- **GAN: two-player minimax game**

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

D: discriminator (input: image, output: **real**/**fake**)

G: generator (input: random noise, output: generated data)

D(x): probability that assign to **real**

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

$$\min_G V(G) = \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Loss Functions for GANs

- Loss functions

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(-v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Squared Hellinger	$1-\exp(-v)$	$t<1$	$\frac{t}{1-t}$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$

GAN (Goodfellow et al, 2014)

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

Least Squares GAN (Mao et al, 2016)

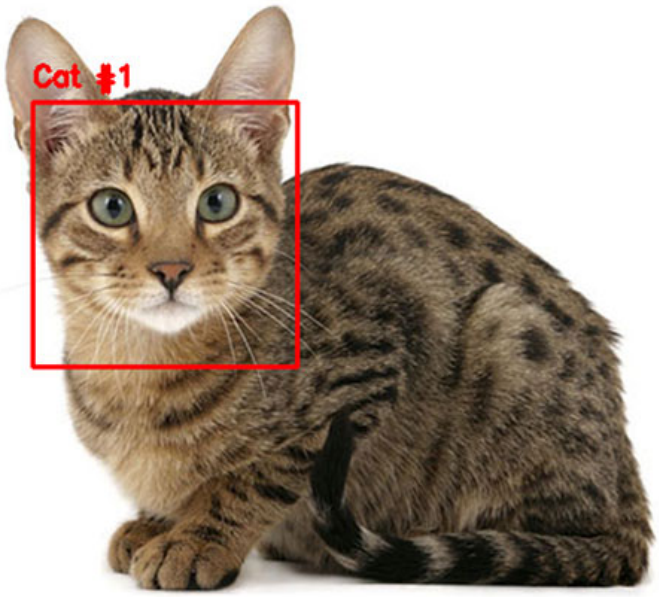
$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$

Data

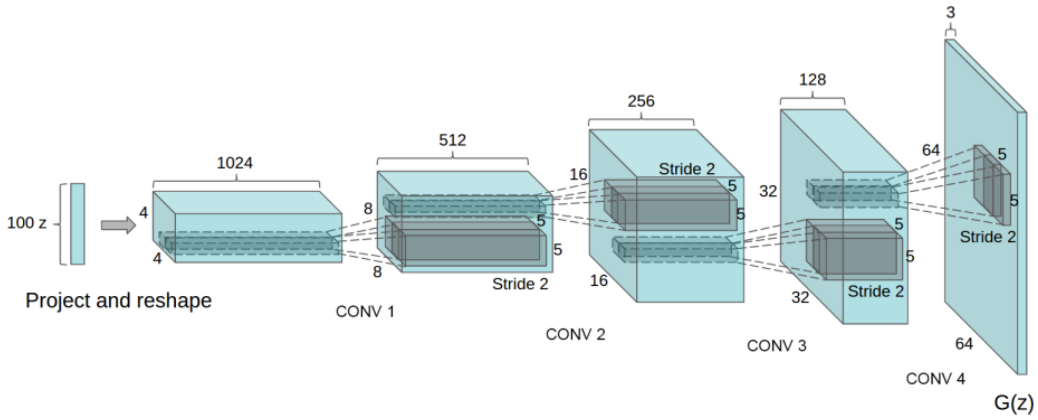
Cat Face Images

- Cat images from Stanford pet dataset + google image crawling
- Extract cat face from the collected images using Open CV cat face detector
- All the images resized into 64 x 64 resolution
- Total 1,193 images



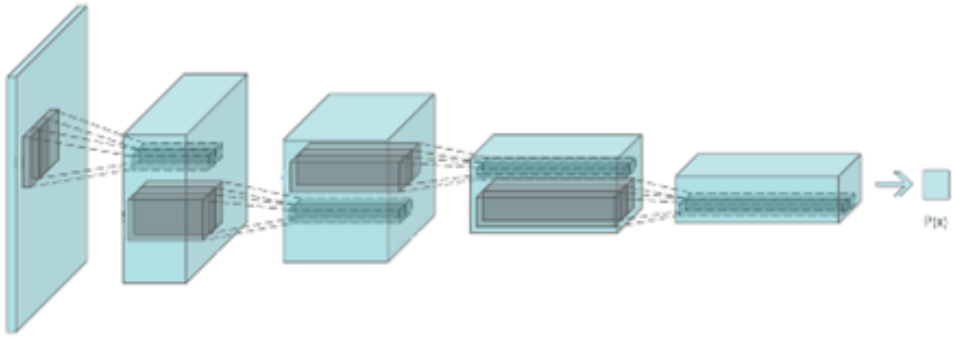
Model Structures

- Deep Convolutional Generative Adversarial Network (Radford et al, 2015)



Generator

- Dense layer at input
- Transpose convolutional layers
- No pooling layers
- Activation: Relu/tanh(output)
- Batch normalization



Discriminator

- Convolutional layers
- No pooling layers
- Activation: Leaky Relu (leak = 0.2)
- Batch normalization

Model Structures

- Objective functions
  - Logistic sigmoid / Wasserstein / least squares
- Add Gaussian noise (mean = 0, std = 0.05)
- Normalize RGB intensity for tanh output layer in D
- Epoch: 1,500
- Minibatch size: 64
- Optimizer
  - Adam
  - Learning rate: 0.0001
  - Beta1: 0.5
  - Gradient clipping

Results

Real Images

Logistic Sigmoid Loss GAN

Results

Least Squares GAN

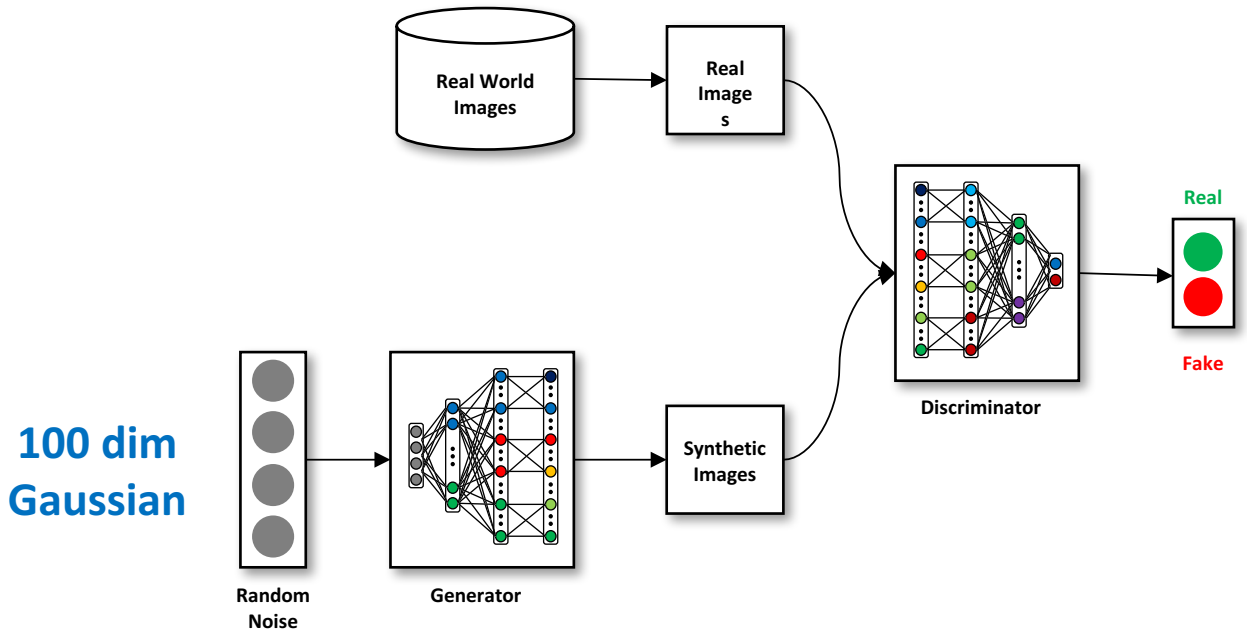
Wasserstein GAN

Results

- GANs learn the density, do not memorize training data
- Evidence that GAN did not memorize the training images: **Odd Eyes**

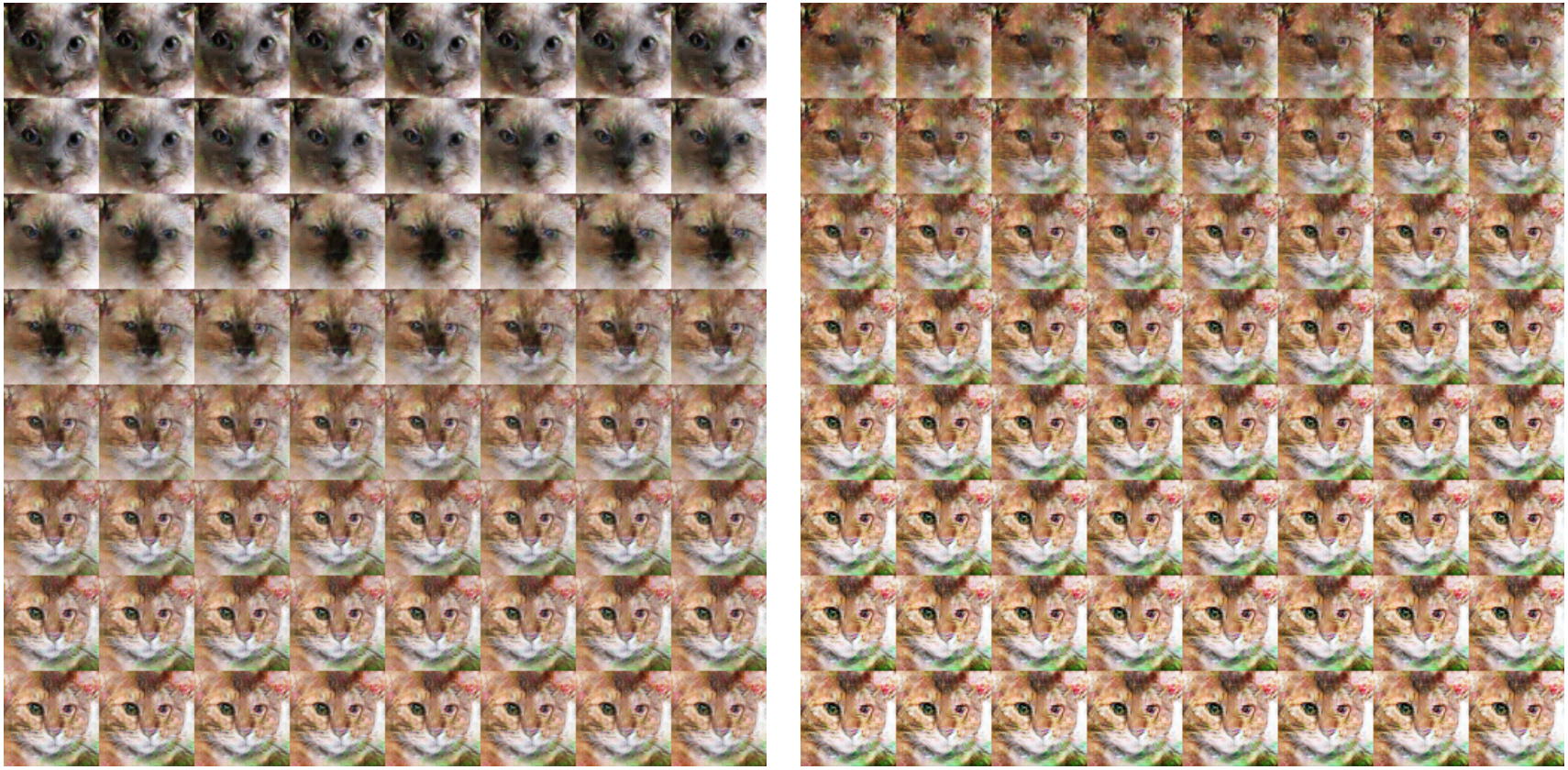
Results

- If the learning was successful: small changes in input ➔ small changes in output



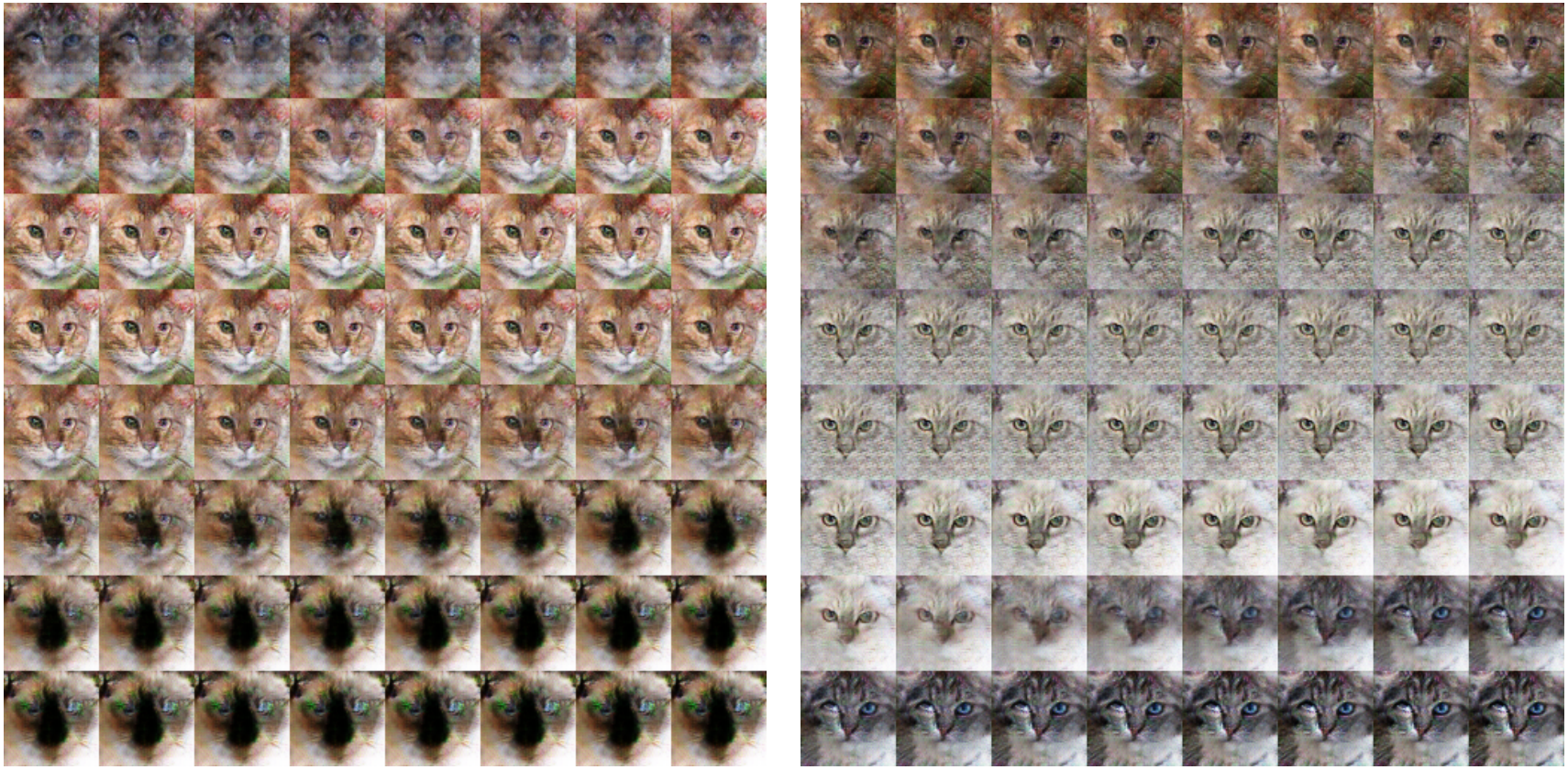
Generate samples fixing 99 dimensions and increase the remaining one continuously

Results



Generate samples fixing 99 dimensions and increase the remaining one continuously

Results



Generate samples fixing 99 dimensions and increase the remaining one continuously

Conclusions

- We constructed pattern recognition system that generates cat face images based on deep convolutional generative adversarial networks
- DCGAN successfully learned the training images and generated cat-like synthetic images
- DCGAN did not memorize the training example but approximated the distribution of cat face images
- Quality of generated images was not satisfactory, maybe the number of training images were not enough (or quality of training images were not good)