ITMO UNIVERSITY

Saint Petersburg, Russia

Специализированные технологии машинного обучения / Advanced Machine learning Technologies

Lecture 4 – Generative Models

Outline

ITMO UNIVERSITY

1. Autoencoders (AEs): “encode -> code -> decode”
2. Latent variables and latent space;
3. Variational autoencoders (VAEs);
4. Generative Adversarial Networks (GANs).
5. Conditional Adversarial Networks (CGANs).

ITMOre than a UNIVERSITY

Unsupervised Learning

ITMO UNIVERSITY

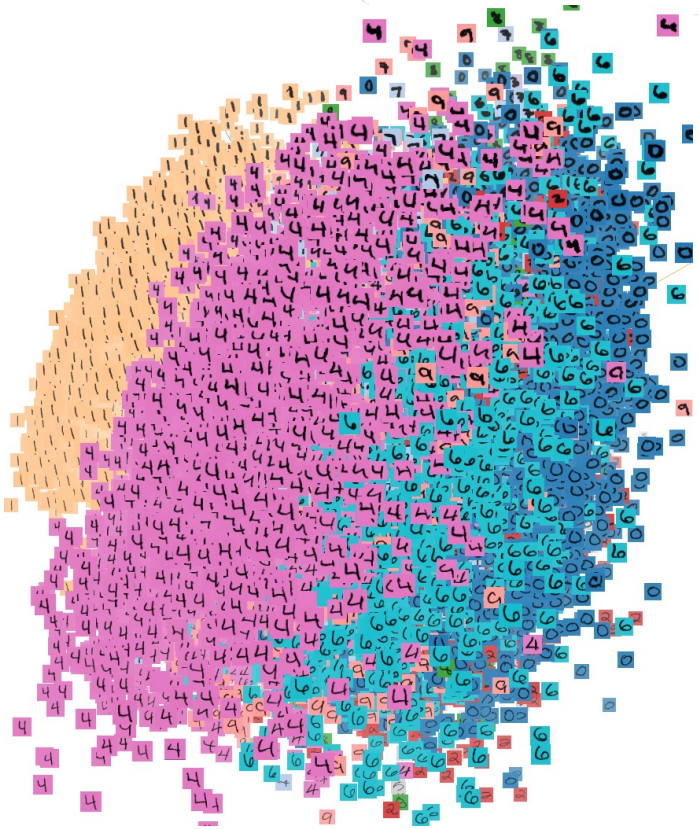
Unsupervised Learning

Data: X (no labels!)

Goal: Learn the structure of the data (density estimation)

- Clustering/grouping,
- Density estimation,
- Dimensionality reduction,

? Autoencoders (AE),
? Variational AE (VAE)
? Generative Adversarial Networks (GAN)



ITMOre than a UNIVERSITY

Why Generative Models?

ITMO UNIVERSITY

- **Supervised learning models are discriminative models**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**
- **Discriminative models have several key limitations**
  - Can't model **P(X)**, i.e. the probability of seeing a certain image
  - Thus, can't sample from **P(X)**, i.e. **can't generate new objects**
- **Unsupervised models**
  - Estimate **P(x)** as density distribution but cannot sample new objects with desired properties
- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new objects

Autoencoders

ITMO UNIVERSITY

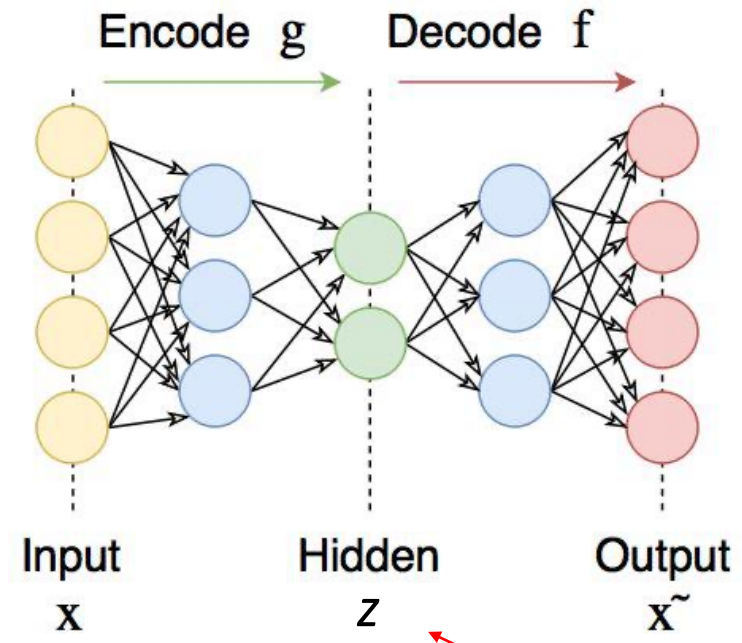
- The autoencoder idea was a part of NN history for decades *(LeCun et al, 1987).*

- Traditionally an autoencoder was used for dimensionality reduction and feature learning.

- Recently, the connection between autoencoders and latent space modeling has brought variational autoencoders (VAEs) to the front of generative modeling.

ITMOre than a UNIVERSITY

Autoencoders



Autoencoders try to learn **Identity function**, so that:

$$\tilde{x} = f(g(x)) = f(z) \cong x$$

“dense” autoencoder

“latent variables”, “latent space”

Autoencoders

ITMO UNIVERSITY

- Given data x *(no labels)* we would like to learn the functions *g (encoder) and f (decoder)* where:

$$\mathrm{g}(x) = s(wx + b) = z$$

$$\mathrm{f}(z) = s(w'\mathrm{z} + b') = \hat{x}$$

so that $h(x) = f\big(g(x)\big) = \hat{x}$

where h is an **approximation** of the identity function.

z is some **latent representation** or **code** and s is a non-linearity such as the sigmoid

($\hat{x}$ is x's reconstruction)

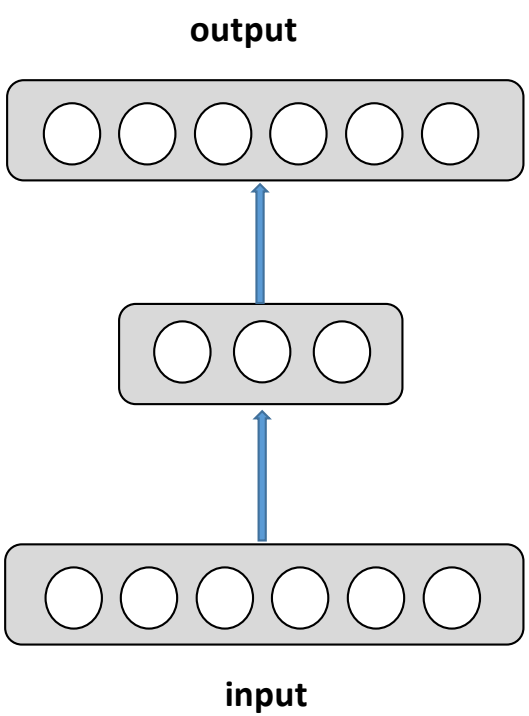
- Add some regularization not to learn trivial reconstruction

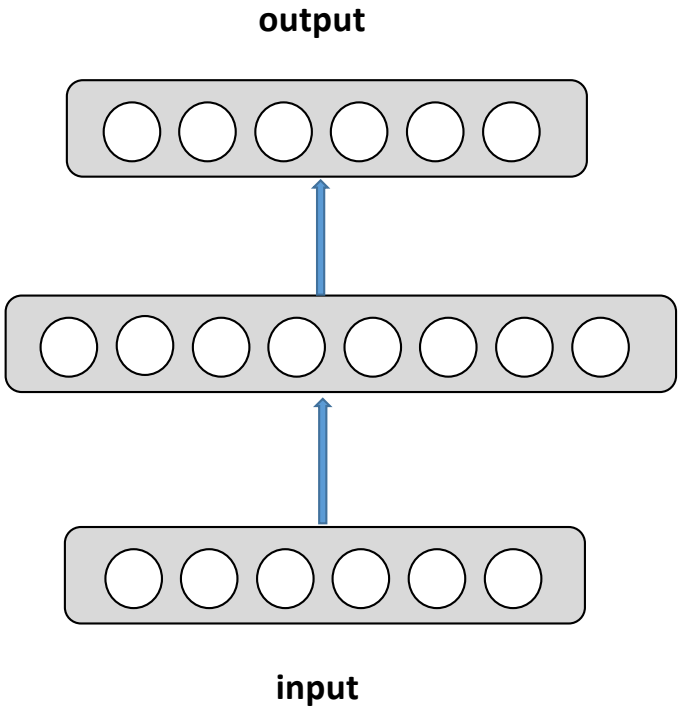
ITMO More than a UNIVERSITY

Undercomplete AE vs overcomplete AE

ITMO UNIVERSITY

Two basic types of AE structures:





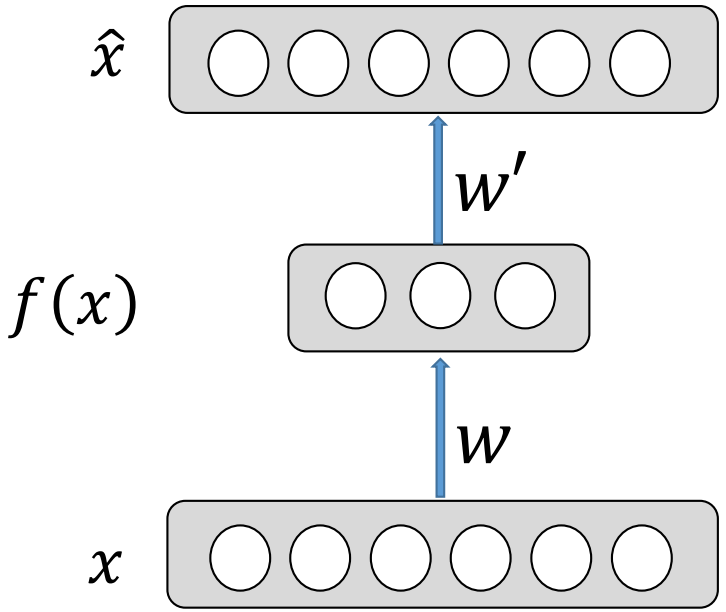
ITMO More than a UNIVERSITY

Undercomplete AE

ITMO UNIVERSITY

Hidden layer is **undercomplete** if smaller than the input layer

- Compresses the input => dimension reduction
- Good features for representation of meaningful characteristics of the input data.



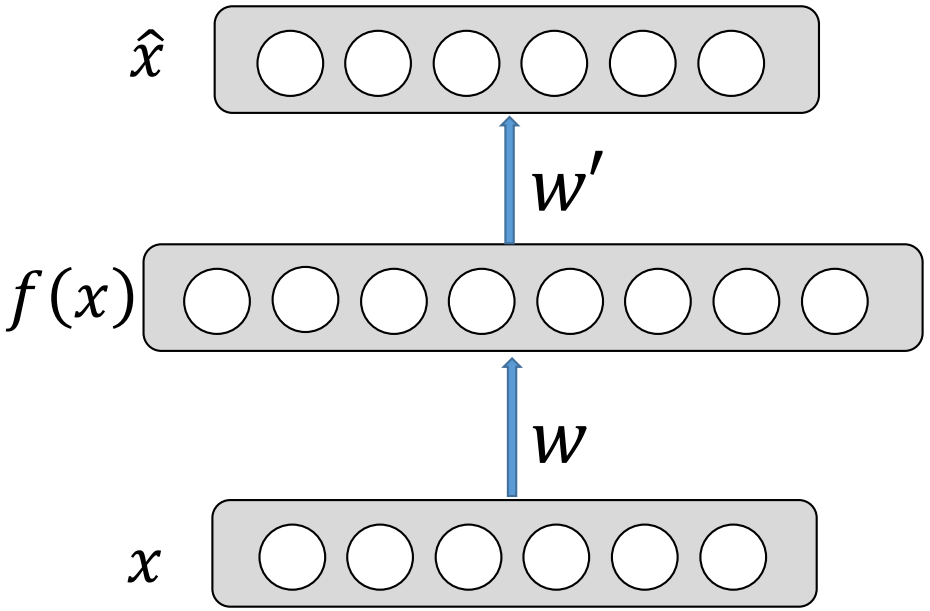
ITMO more than a UNIVERSITY

Overcomplete AE

ITMO UNIVERSITY

Hidden layer is **overcomplete** if its dimension is greater than the input layer

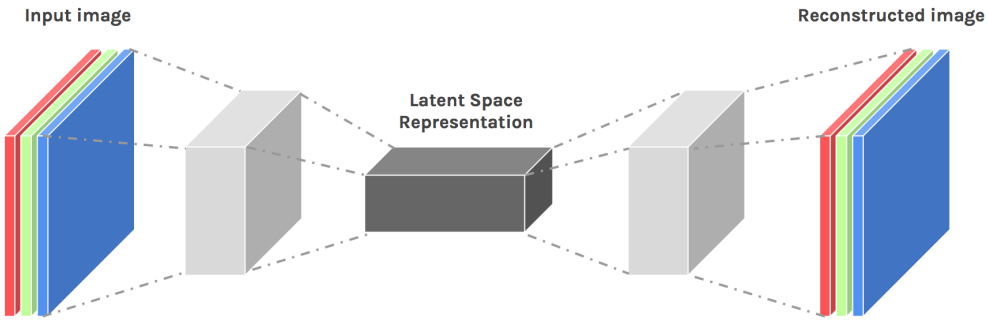
- No compression in hidden layer.
- Each hidden unit could copy a different input component.
- No guarantee that the hidden units will extract meaningful structure.
- A higher dimension code helps to model a more complex distribution.
- Overfitting?

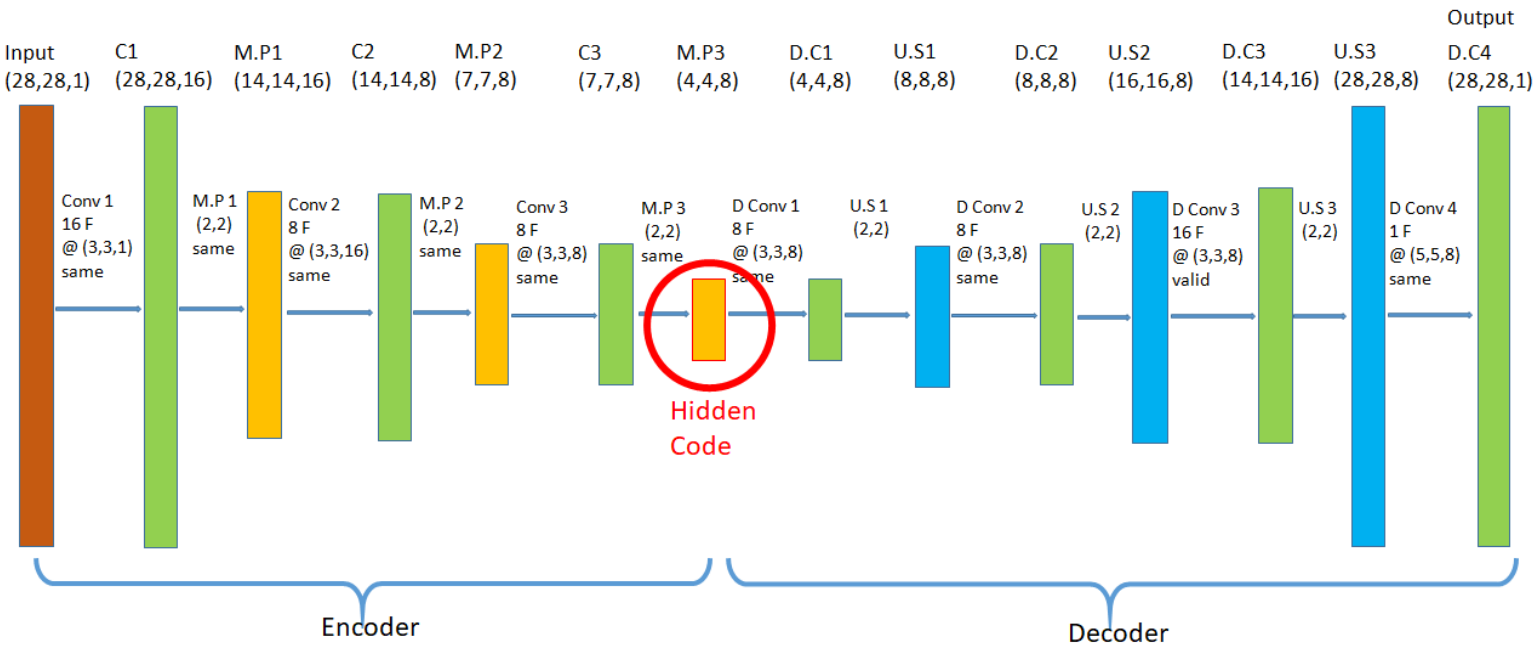


ITMO More than a UNIVERSITY

Convolutional AE

ITMO UNIVERSITY

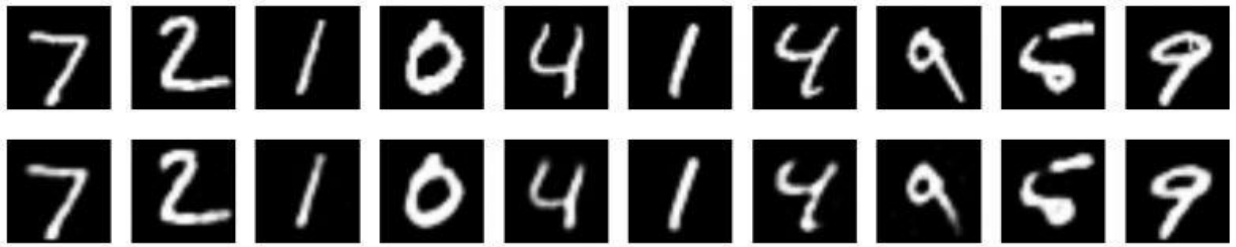




Keras example results

ITMO UNIVERSITY

Dense AE:



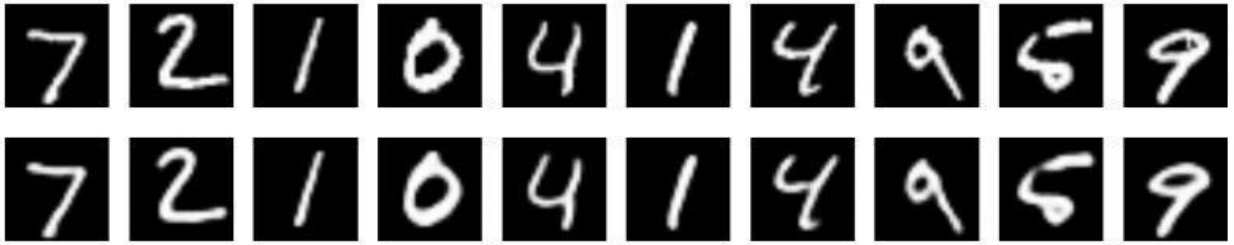
ITMOre than a UNIVERSITY

Keras example results

ITMO UNIVERSITY

Dense AE:

Conv AE:



ITMOre than a UNIVERSITY

Regularization

ITMO UNIVERSITY

Motivation:

- We would like to learn meaningful features **without** varying the code's dimensions (Overcomplete or Undercomplete).

Solution: apply other constraints to the training process.

→ Add regularization term to the loss function:

$$L(x, f(g(x))) + \Omega(h),$$

For example, $\Omega(h) = \lambda * |h|$

- prevent overfitting;
- more robust and meaningful codes (latent variables);
- prevent learning trivial identity function;

ITMOre than a UNIVERSITY

Denoising Autoencoders

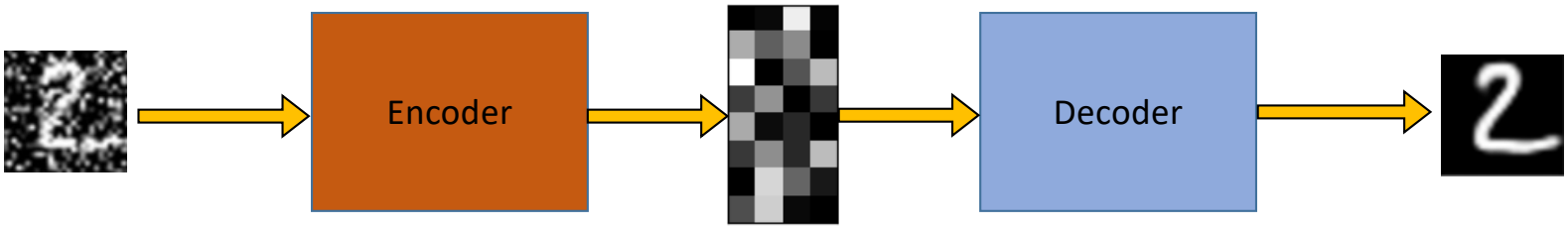
ITMO UNIVERSITY

Intuition:

- Add some noise to the input and compare with clean data (train to reconstruct initial image)

$$L(x, f(g(\hat{x}))).$$

A more robust model



Noisy Input

Latent space representation

Denoised Input

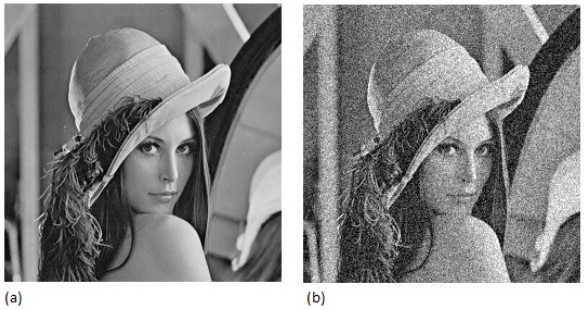
ITMOre than a UNIVERSITY

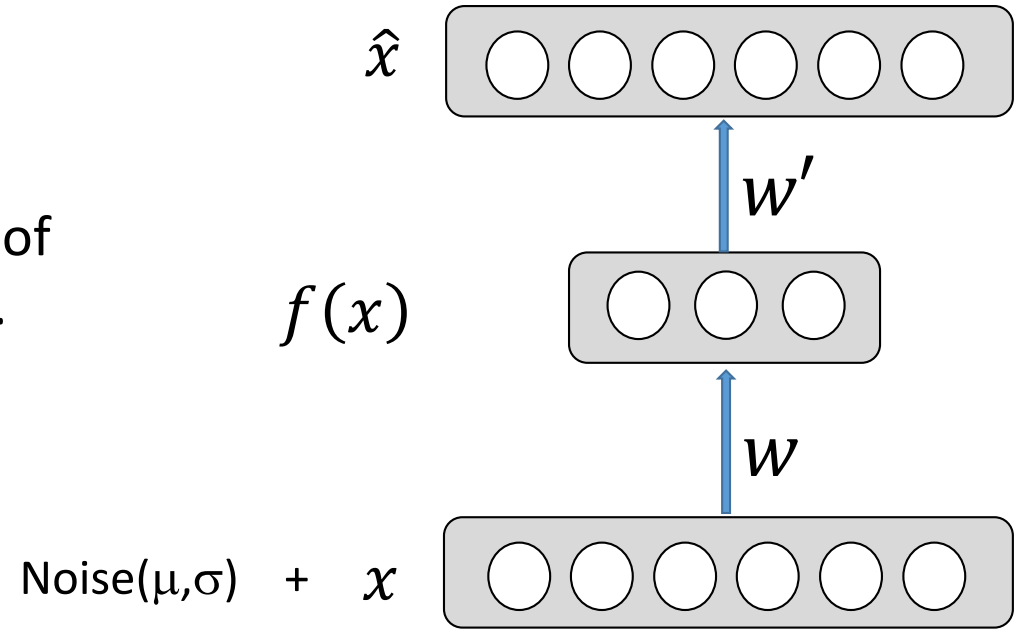
Denoising Autoencoders

ITMO UNIVERSITY

Idea: A robust representation against noise:

- Random assignment of subset of inputs to 0, with probability v.
- Gaussian additive noise.

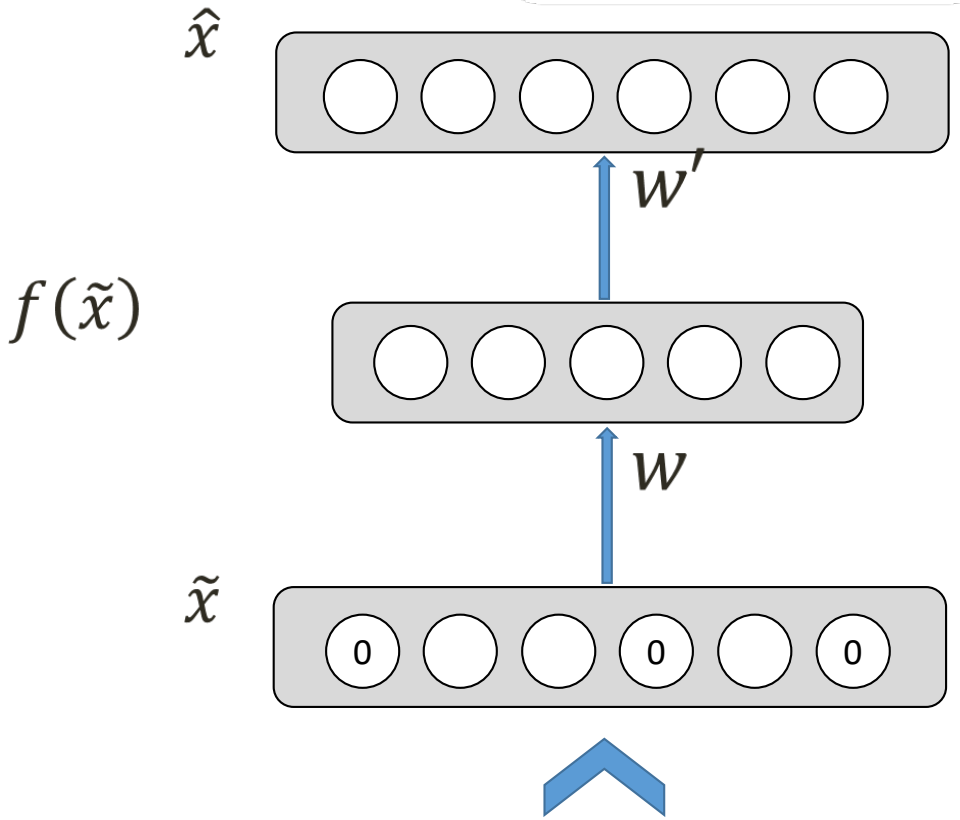




Denoising Autoencoders

ITMO UNIVERSITY

- Reconstruction is computed from noisy image;
- Loss function compares prediction with the noiseless image;
- We are forcing the hidden layer to learn a generalized structure of the data!



Denoising Autoencoders - process

ITMO UNIVERSITY

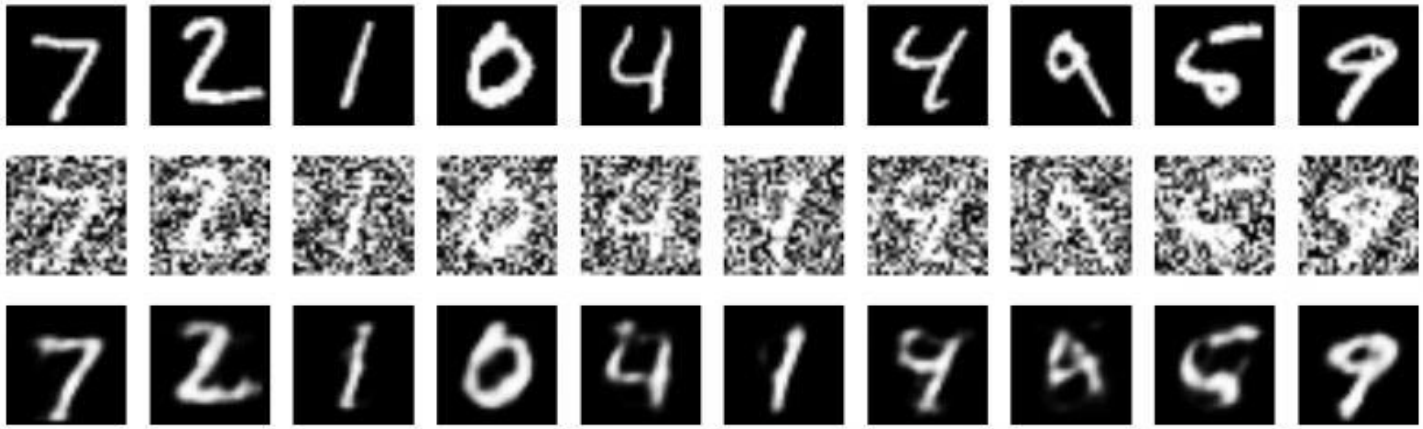
Taken some input x Apply Noise $\tilde{x}$

ITMO more than a UNIVERSITY

Denoising convolutional AE - example

ITMO UNIVERSITY

Keras implementation result:



ITMOre than a UNIVERSITY

Latent space: motivation

ITMO UNIVERSITY

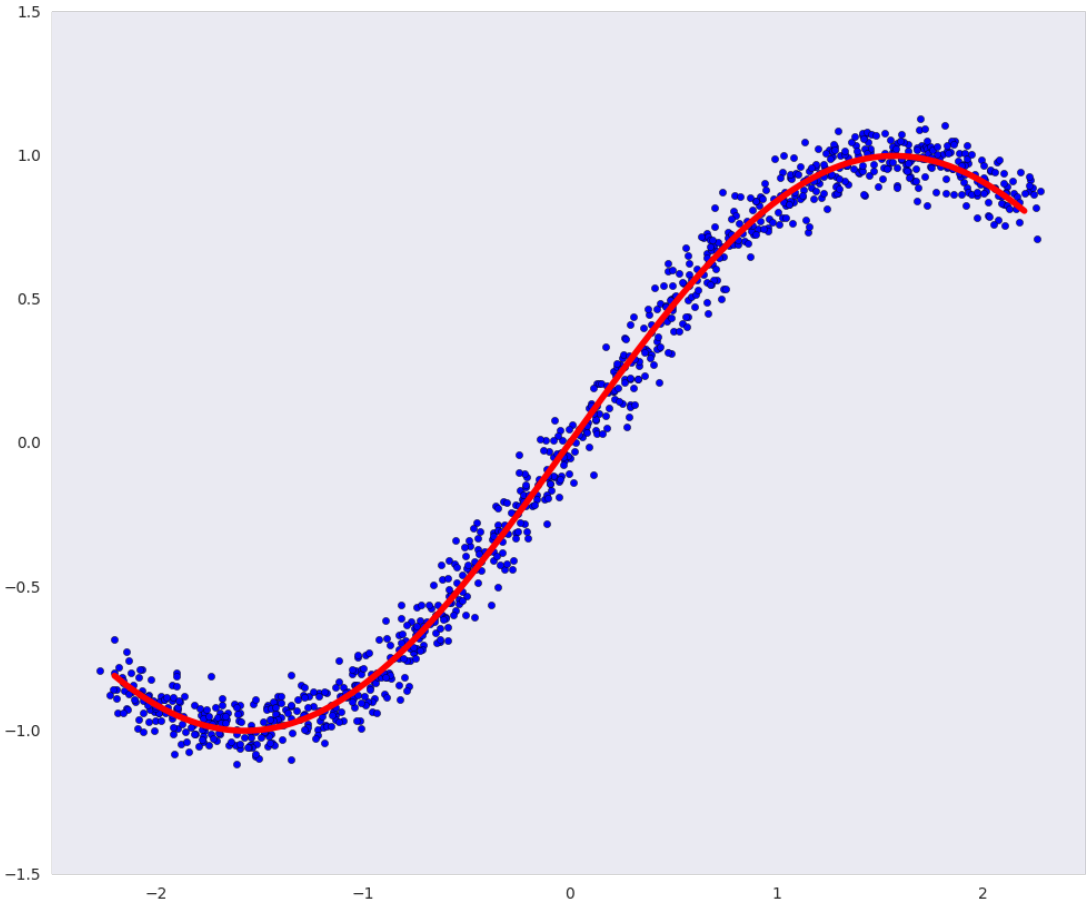
Learning the identity function seems trivial, but with added constraints on the network (such as limiting the number of hidden neurons or/and regularization) we can learn information about the structure of the data

→ Trying to capture the **distribution of the data**

Let's take a look at the codes….

ITMOre than a UNIVERSITY

Latent space



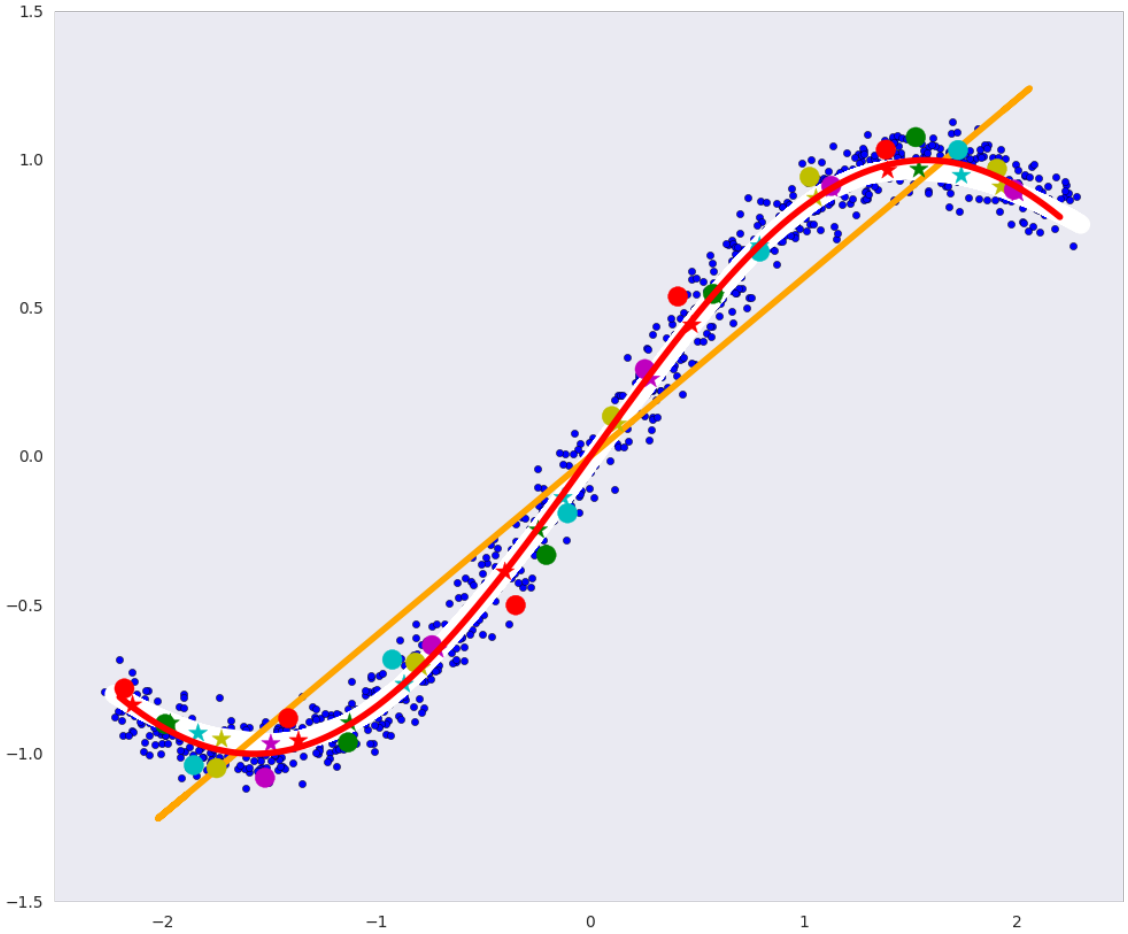
- Dots – initial distribution of the samples;
- Red line – desirable manifold

(objects near the line are ~digit images)

https://habr.com/ru/post/331500/

Latent space

ITMO UNIVERSITY



- the white line is the manifold to which the blue data points after the auto-encoder go, that is, the auto-encoder's attempt to construct the manifold that determines the most variations in the data;
- The orange line is the manifold to which the blue data points go after the PCA;
- colored circles - points that turn into asterisks of the corresponding color after encoding;
- colored asterisks - respectively, the images of the circles after the auto-encoder.

White line – is “encoded manifold” or “latent space”

https://habr.com/ru/post/331500/

Latent variables

ITMO UNIVERSITY

➢ We can consider the general population as a process of generating data, which depends on a number of hidden variables (random variables).

➢ The dimension of the data can be much higher than the dimension of the hidden random variables that these data define. Consider the process of generating the next digit: how the digit will look can depend on many factors:

- **desired numbers;**
- **thickness of the stroke;**
- **tilt**
- **neatness;**

➢ Each of these factors has its own prior distribution.

➢ For example, the probability that a figure of **8** will be drawn is the Bernoulli distribution with a probability of 1/10,

➢ The **thickness of the stroke** also has some distribution and may depend on both accuracy and its hidden variables, such as the **thickness of the handle** or the **temperament of a person** (again with his distributions).

➢ The auto-encoder itself in the learning process will (somehow) learn the hidden factors, for example, such as those listed above, some of their complex combinations.

Various types of “2”

However, the joint distribution that auto-encoder learns does not have to be simple, it can be some kind of complex curved area.

Code manifold and real world manifold:

ITMO UNIVERSITY

If we take two objects and look at objects lying on an arbitrary curve between them, then most likely intermediate objects will not belong to the general population, since the variety on which the general population lies can be very curved and small.

moving in a straight line in the space of numbers from one “8” to another

ITMOre than a UNIVERSITY

Code manifold and real world manifold:

ITMO UNIVERSITY

If we take two objects and look at objects lying on an arbitrary curve between them, then most likely intermediate objects will not belong to the general population, since the variety on which the general population lies can be very curved and small.

moving in a straight line in the space of numbers from one “8” to another

But if we move along a curve between codes (and if the code’s manifold is well parametrized), then the decoder will translate this curve from the code space to a curve that does not leave the defining manifold in the space of objects. That is, intermediate objects on the curve will belong to the entire population.

moving in a straight line in the space of codes from one “8-code” to another

ITMOre than a UNIVERSITY

Learning latent space

ITMO UNIVERSITY

- **X** — random sample (single image) from the dataset of 28x28 pixels images,
- **Z** - a random value of hidden factors that determine the number in the picture
- **P(X)** - the probability distribution of the images of the figures in the pictures, i.e. the probability of a particular image of a digit *generally* to be drawn (if the picture is not like a digit, then this probability is extremely small);
- **P(Z)** - the probability distribution of hidden factors, for example, the distribution of the thickness of the stroke;
- **P(Z|X)** - the probability distribution of hidden factors for a given picture (a different combination of hidden variables and noise can lead to the same picture);
- **P(X|Z)** - the probability distribution of pictures for given hidden factors; the same factors can lead to different pictures (the same person in the same conditions does not draw exactly the same numbers);
- **P(Z,X)** - joint distribution and, the most complete understanding of the data needed to generate new objects.

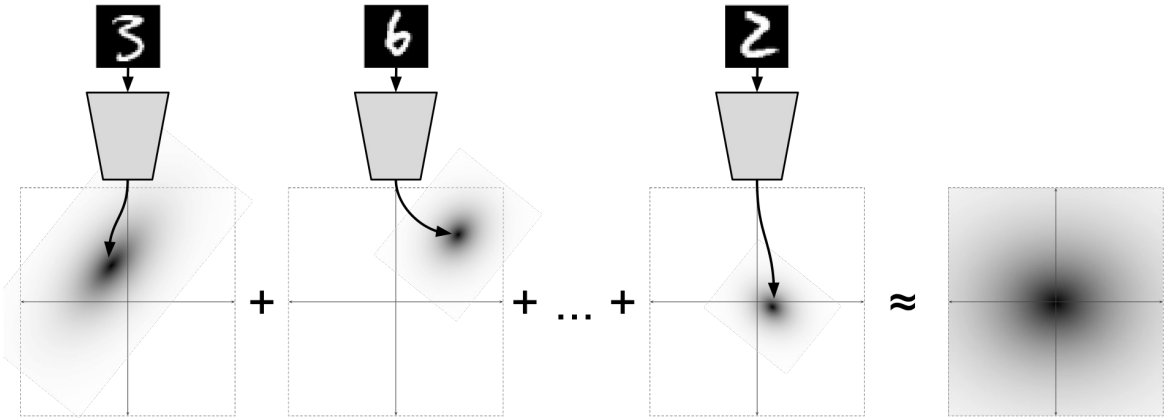
Learning latent space

ITMO UNIVERSITY

http://ijdykeman.github.io/ml/2016/12/21/cvae.html

It can be shown (introducing specific loss function to encoder) that:

$$P(Z|X) = N(\mu(X), \Sigma(X))$$



Encoder predict the parameters of some Normal distribution!

A variational autoencoder (VAE) is a generative model that learns to map objects into a given hidden space and than to generate the new objects as decoded samples from this hidden space.

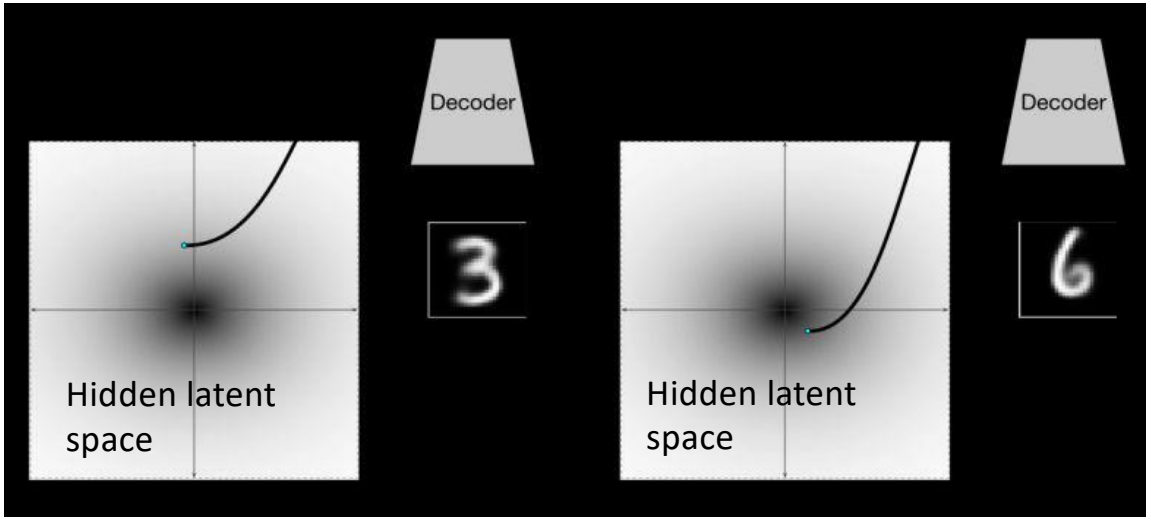
Variational Autoencoders

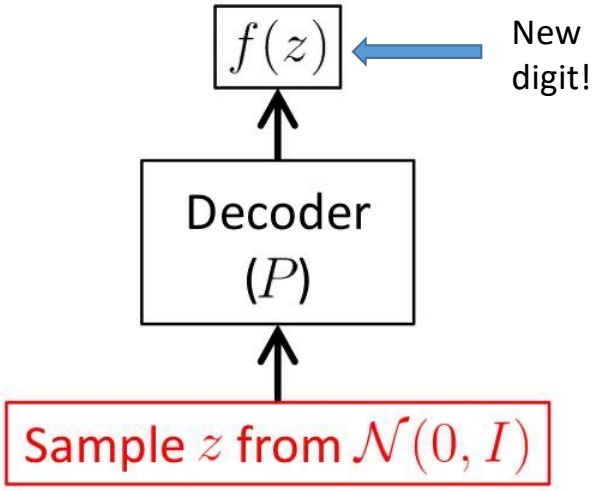
ITMO UNIVERSITY

In order to be able to generate new objects, the space of hidden variables (latent variables) must be predictable.
VAE are autoencoders that learn to map objects into a given latent space and, therefore, sample them.

Therefore, variational autoencoders are also referred to the family of generative models.

For example, we can sample from ***P(z) ~ N(0,1)*** distribution to decode new digits:





Conditional VAE: add new information to the input → new objects with desirable features (thickness or style)!

http://ijdykeman.github.io/ml/2016/12/21/cvae.html https://habr.com/ru/post/331664/

VAE architecture

ITMO UNIVERSITY

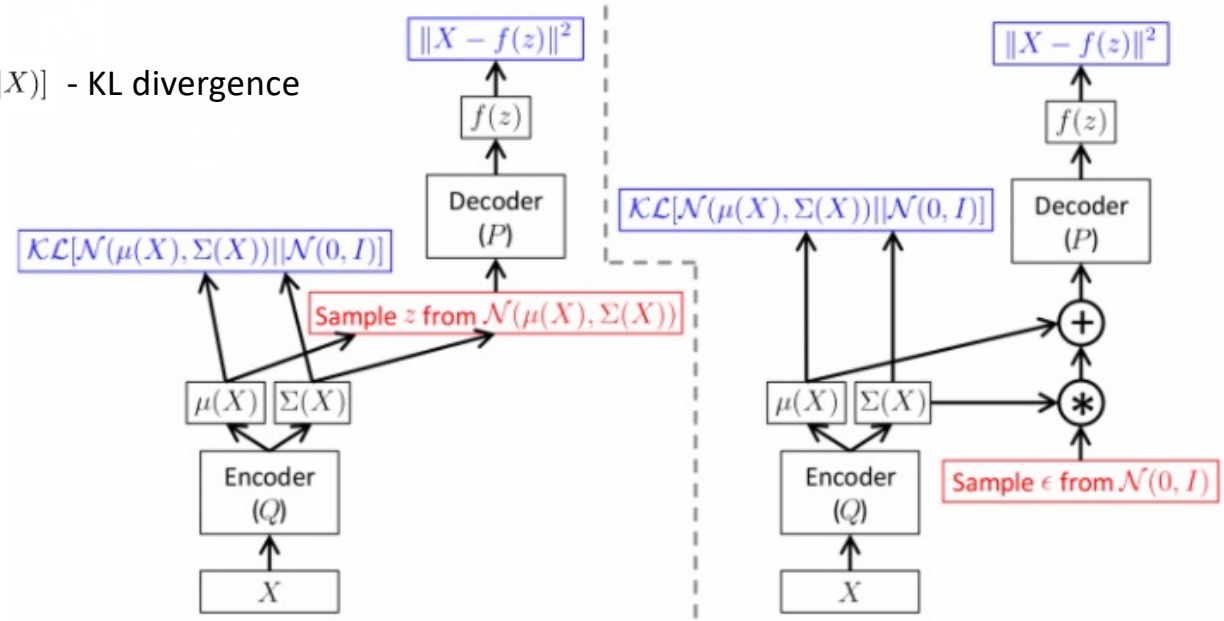
$P(X;\theta) = \int_z P(X|Z;\theta)P(Z)dZ$ - want to maximize

$P(X|Z;\theta) = N(X|f(Z;\theta), \sigma^2 I),$

$KL[Q(Z|X)||P(Z|X)] = \mathbb{E}_{Z\sim Q}[\log Q(Z|X) - \log P(Z|X)]$ - KL divergence

We can fix that encoder should generate hidden parameters from the multivariate Gaussian distribution:

$Q(Z|X;\theta_1) = N(\mu(X;\theta_1), \Sigma(X;\theta_1))$



ITMO More than a UNIVERSITY

It can be shown that:

$\log P(X) - KL[Q(Z|X)||P(Z|X)] = \mathbb{E}_{Z\sim Q}[\log P(X|Z)] - KL[Q(Z|X)||P(Z)]$

VAE example

ITMO UNIVERSITY

ITMOre than a UNIVERSITY

ITMO UNIVERSITY

GANs

ITMOre than a UNIVERSITY

GANs

ITMO UNIVERSITY

- **Generative**
  - Learn a generative model – to generate smth new
- **Adversarial**
  - Trained in an adversarial setting: generator vs. discriminator
- **Networks**
  - Use Deep Neural Networks

ITMOre than a UNIVERSITY

Magic of GANs…

ITMO UNIVERSITY

Generate 2D/3D shapes ➔ helpful for designers (rendering)

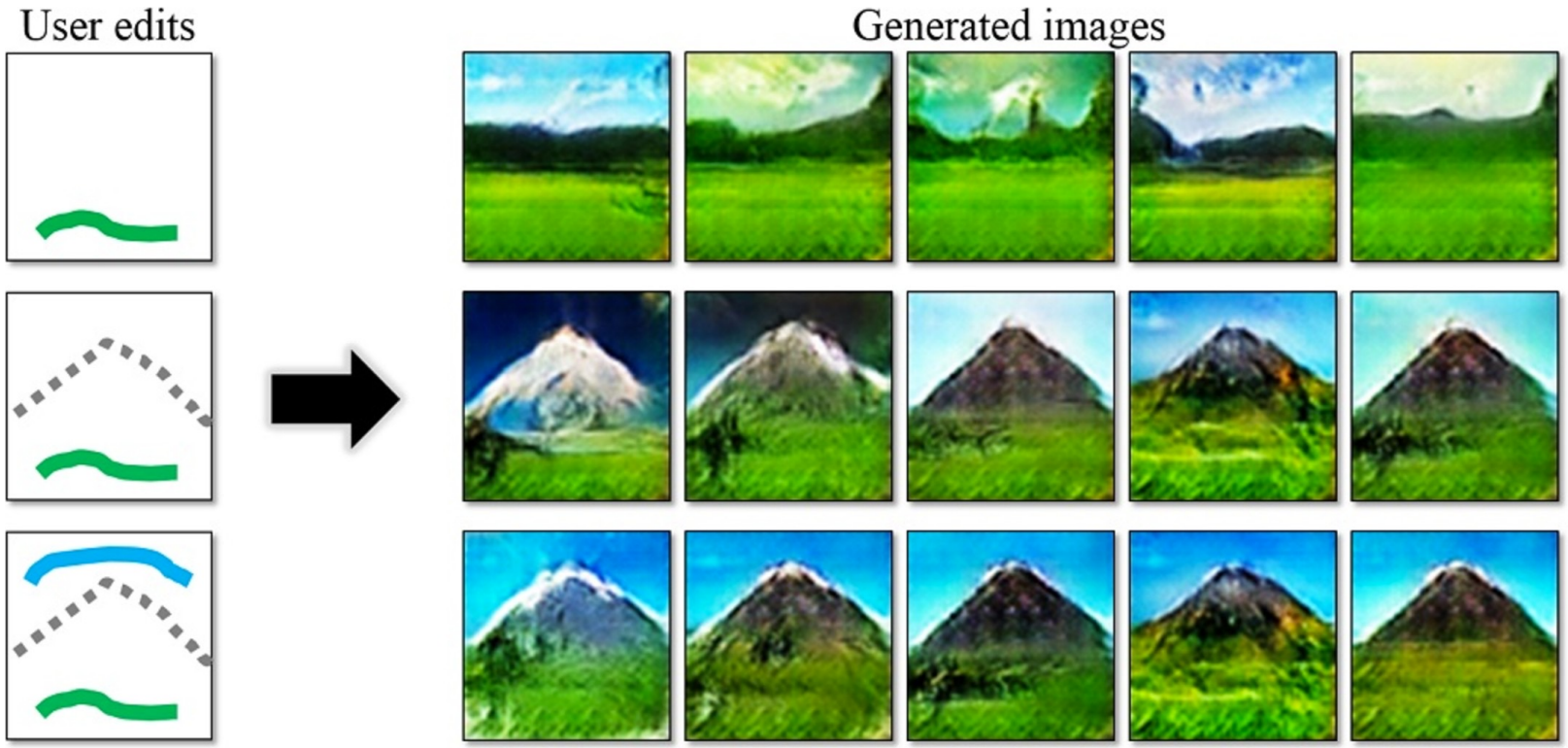
ITMO more than a UNIVERSITY

Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint arXiv:1609.04802* (2016).

Magic of GANs…

ITMO UNIVERSITY

Generate new images and videos ➔ helpful for artists



http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

Applications using GANs

ITMO UNIVERSITY

- Font generation
- Anime character generation
- Interactive Image generation
- Text2Image (text to image)
- 3D Object generation
- Image Editing
- Face Aging
- Human Pose Estimation
- Domain-transfer (e.g. style-transfer, pix2pix, sketch2image)
- Image Inpainting (hole filling)
- Super-resolution
- High-resolution image generation (large-scale image)
- Adversarial Examples (Defense vs Attack)
- Visual Saliency Prediction (attention prediction)
- Robotics
- Video (generation/prediction)
- Synthetic Data Generation
- Real-time face reconstruction
- Photorealistic Image generation (e.g. pix2pix, sketch2image)
- Human Pose Estimation
- …

ITMO more than a UNIVERSITY

Adversarial Training

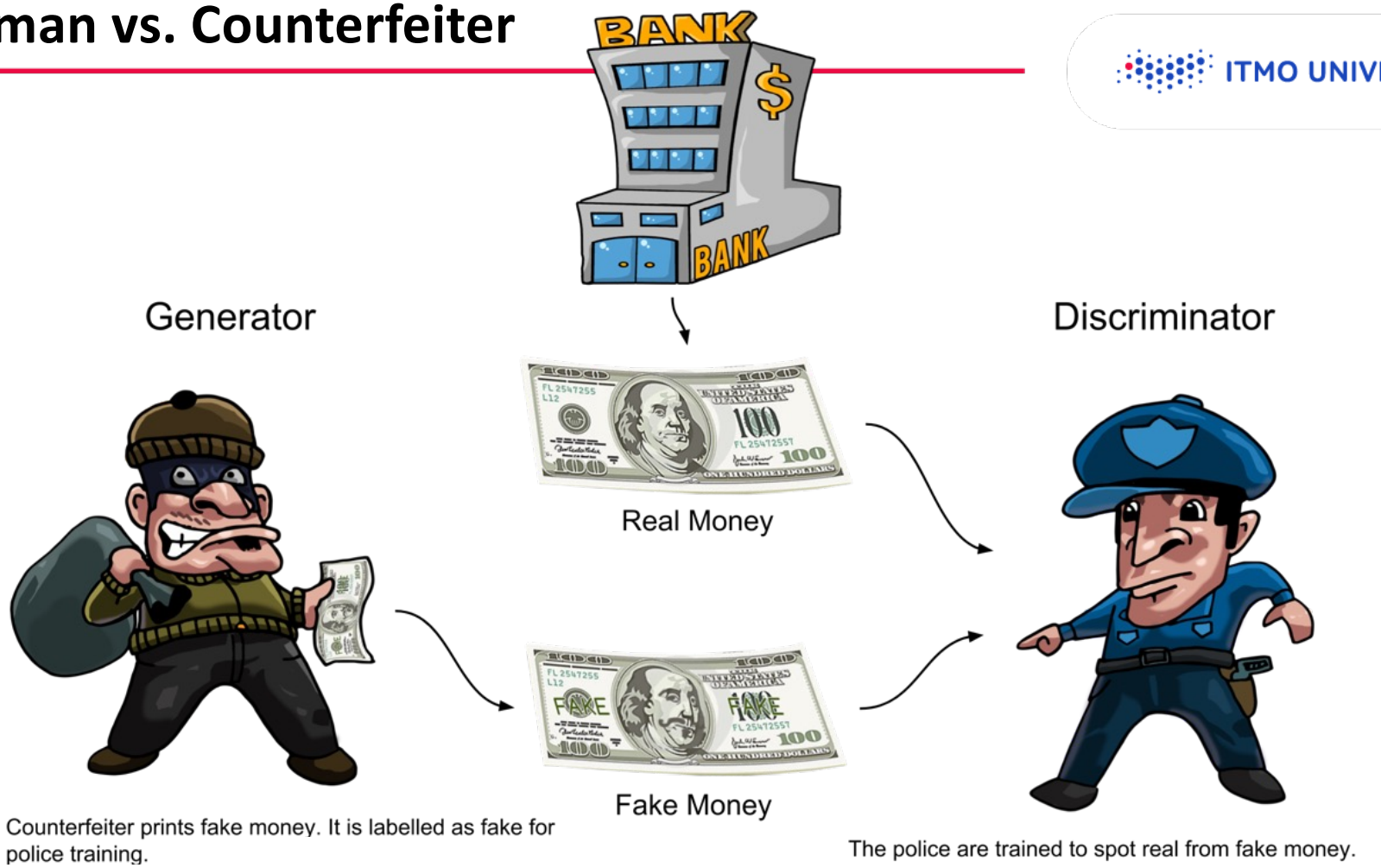
ITMO UNIVERSITY

- **GANs extend the idea of discriminative modeling:**
  - **Generator:** generate fake samples, tries to fool the Discriminator
  - **Discriminator:** tries to distinguish between real and fake samples
  - Train them against each other (separately)
  - Repeat this and we get better Generator and Discriminator
  - => receive more realistic generated objects!

ITMO more than a UNIVERSITY

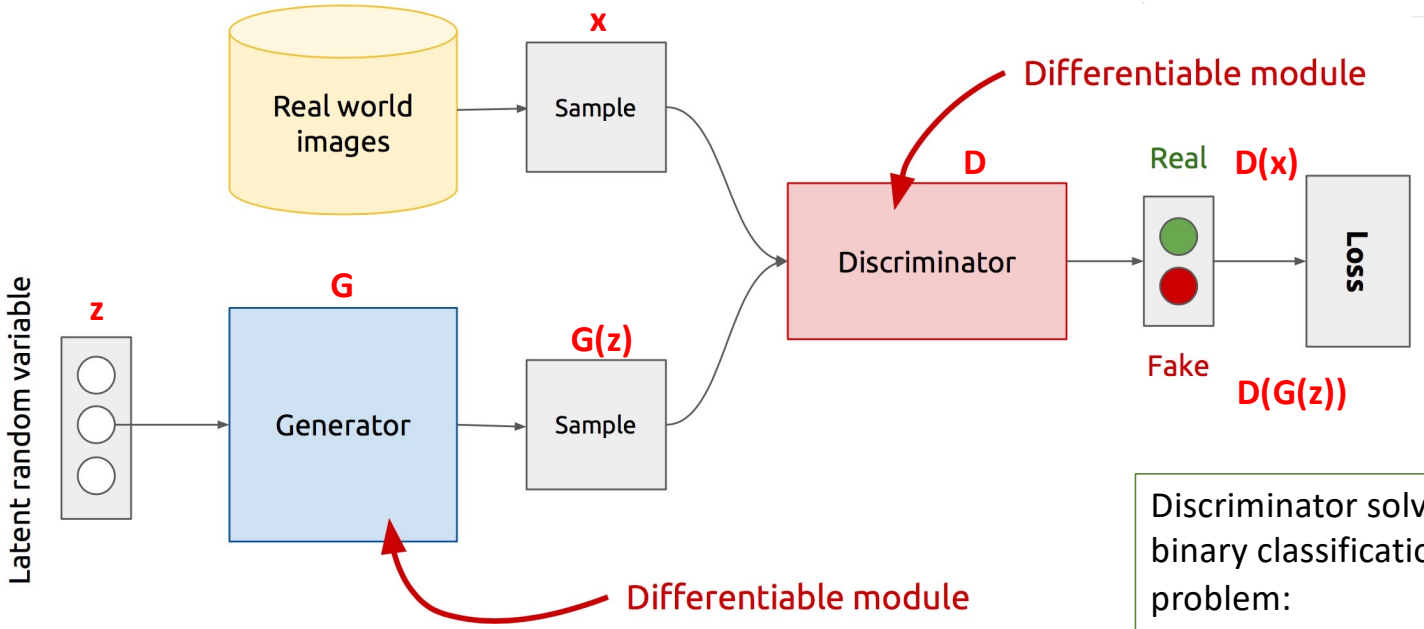
Policeman vs. Counterfeiter

ITMO UNIVERSITY



GAN's Architecture

ITMO UNIVERSITY



Discriminator solve binary classification problem:
«0» - is for the fake
"1" – is for the real

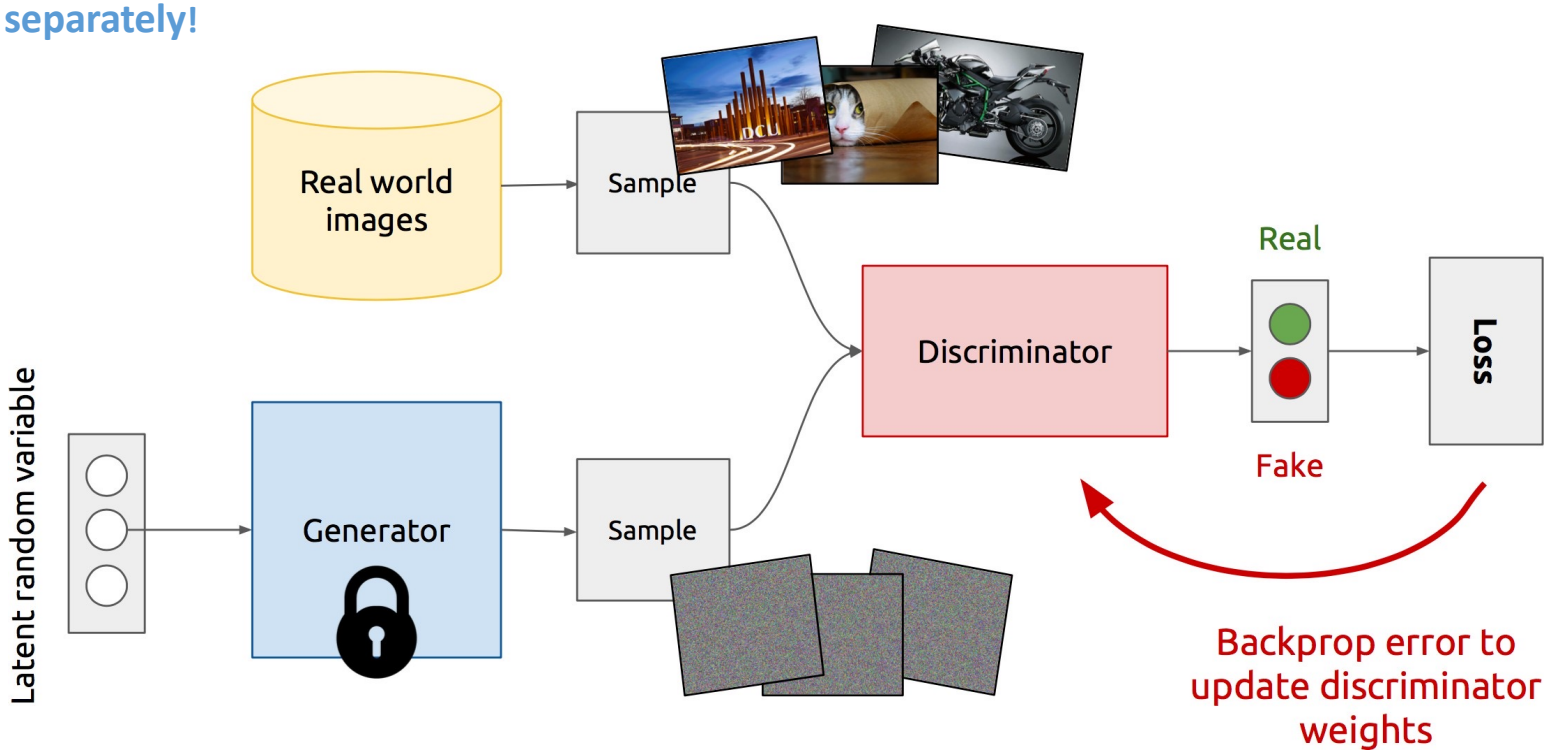
- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

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

Training Discriminator

ITMO UNIVERSITY

Training separately!

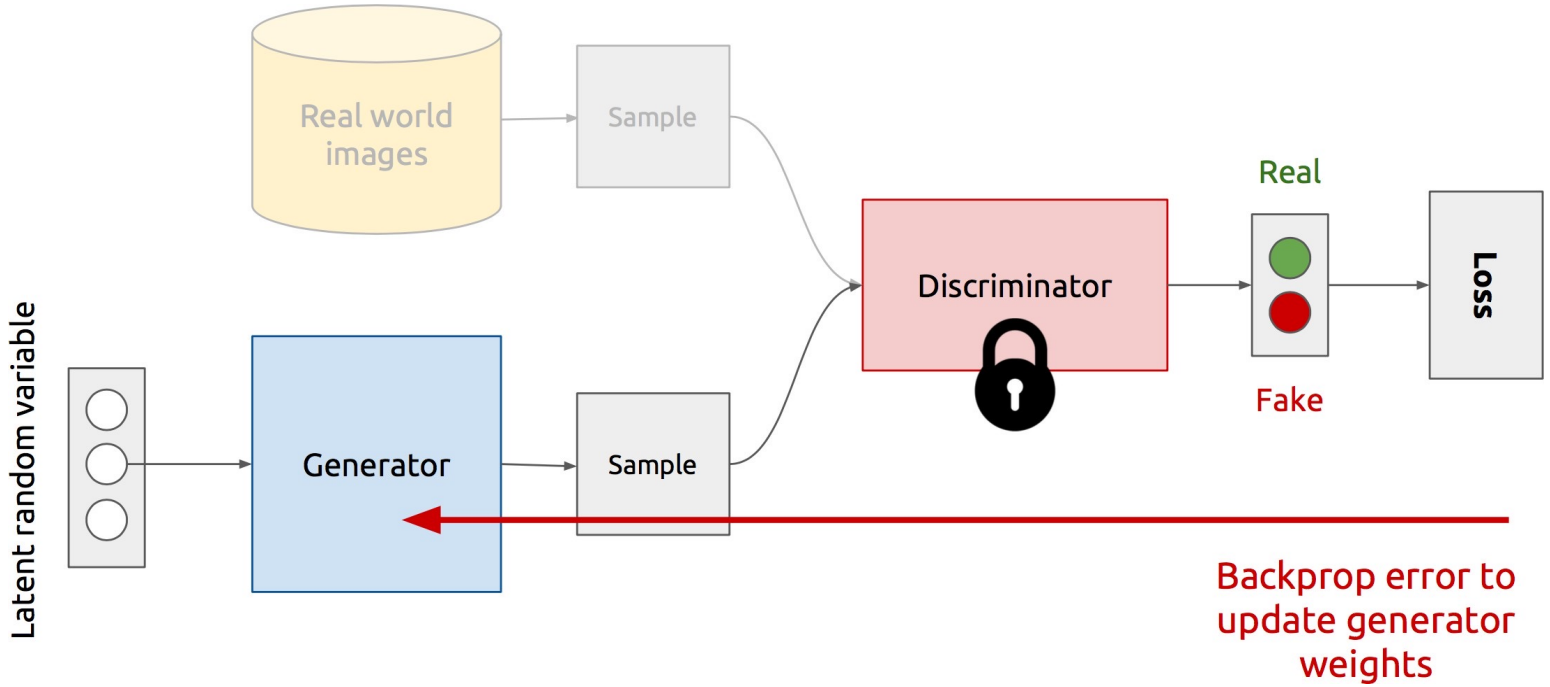


https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator

ITMO UNIVERSITY

Training separately!



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

GAN's formulation

ITMO UNIVERSITY

GANs training process is formulated as a **minimax game**, where:

- The **Discriminator** is trying to maximize its reward $\boldsymbol{V}(\boldsymbol{D}, \boldsymbol{G})$
- The **Generator** is trying to minimize Discriminator's reward (== maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

where:

x $\subset$ X – real samples

G(z) – generated samples

1 – $D(G(z))$ – probability of predicting "real" label

G – generator

D – discriminator

D(x)=1, if discriminator has labeled x as real

D(x)=0, if discriminator has labeled x as fake

GAN's training algorithm

ITMO UNIVERSITY

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**

Discriminator updates

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

Generator updates

- 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:

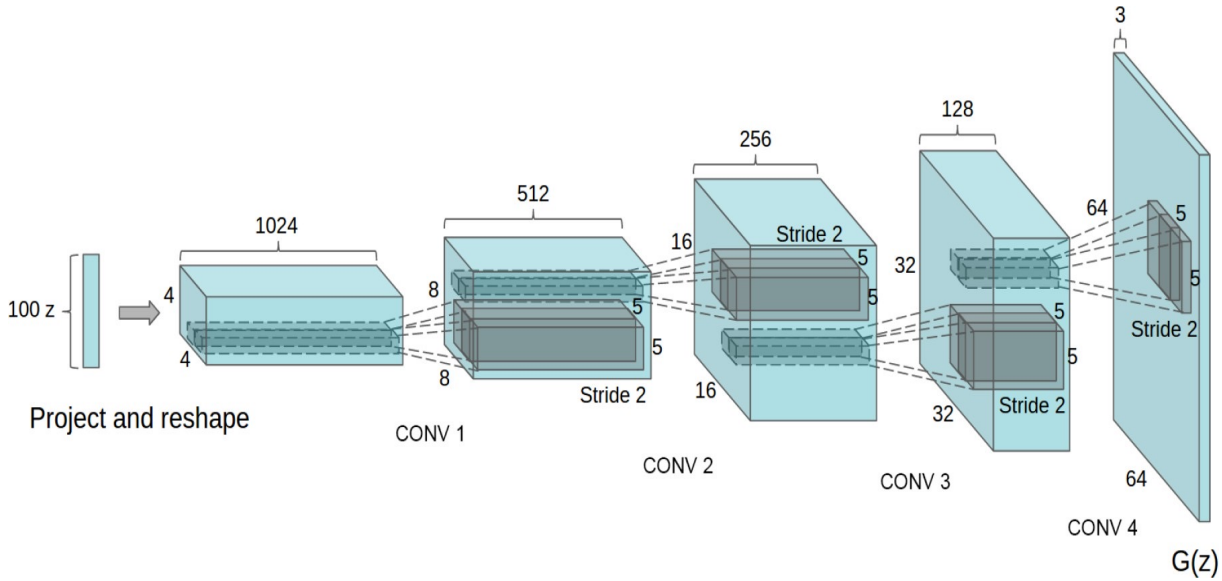
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Deep Convolutional GANs (DCGANs)

ITMO UNIVERSITY

Generator Architecture



Key ideas:

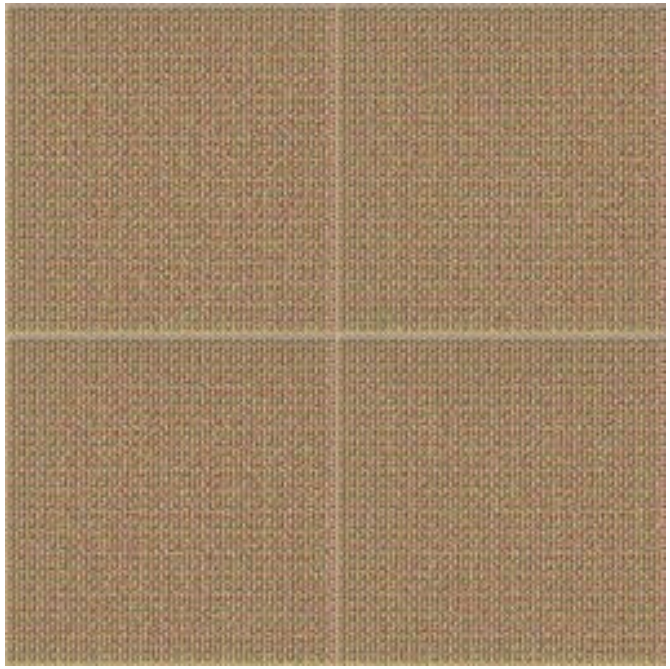
- Replace FC hidden layers with Convolutions
- **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

ITMOre than a UNIVERSITY

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

Example

ITMO UNIVERSITY



ITMOre than a UNIVERSITY

DCGAN: bedroom images

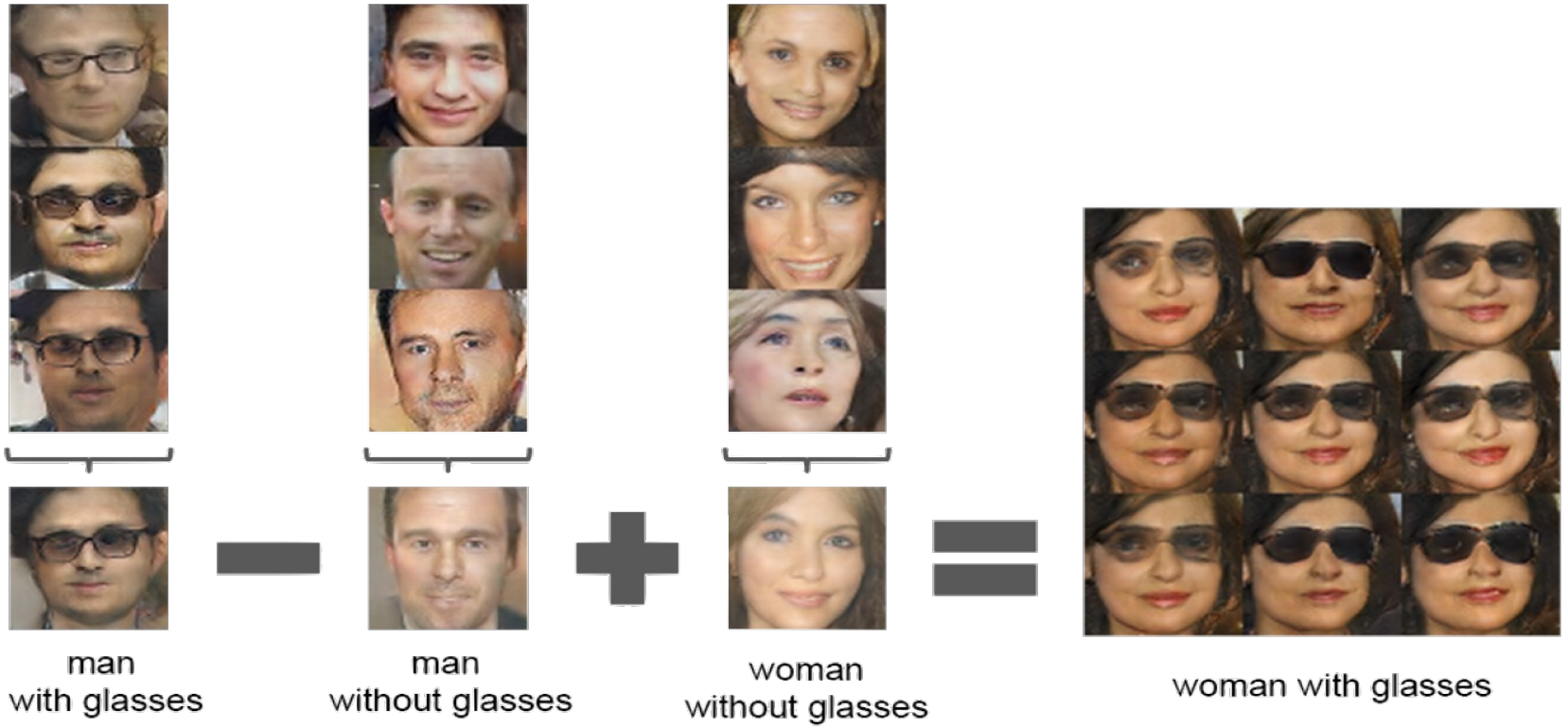
ITMO UNIVERSITY

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

Learning meaningful latent space

ITMO UNIVERSITY

Latent vectors with additional labels capture interesting patterns…



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

GANs and cGANs (I)

ITMO UNIVERSITY

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

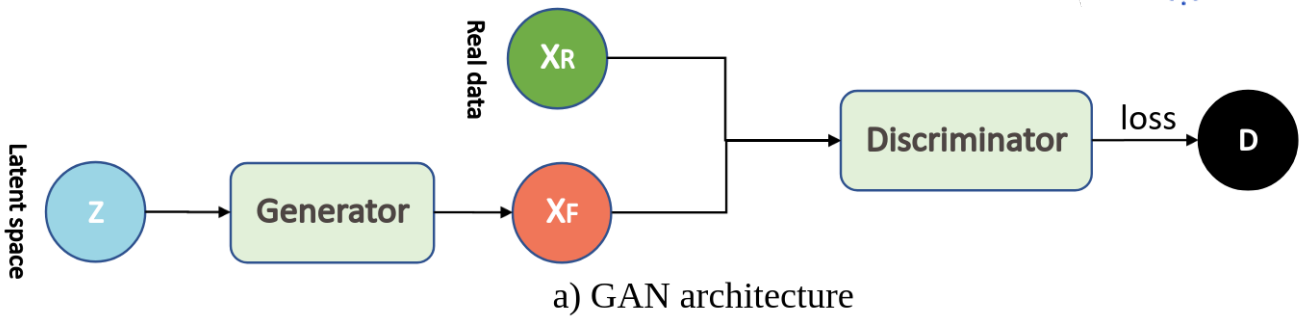
G – a generative model;
D – a discriminative model;
x – training real data;
z – data generated by G;

If both the generator and the discriminator are conditioned on some extra information *y*:

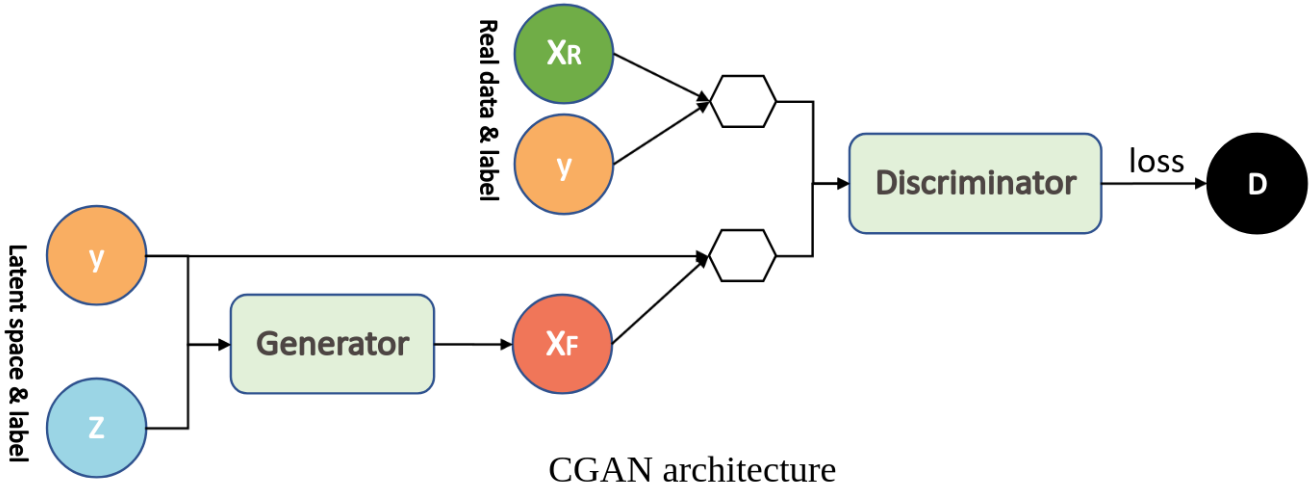
$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$ - conditional GAN (cGAN);

ITMO more than a UNIVERSITY

GANs and cGANs (II)

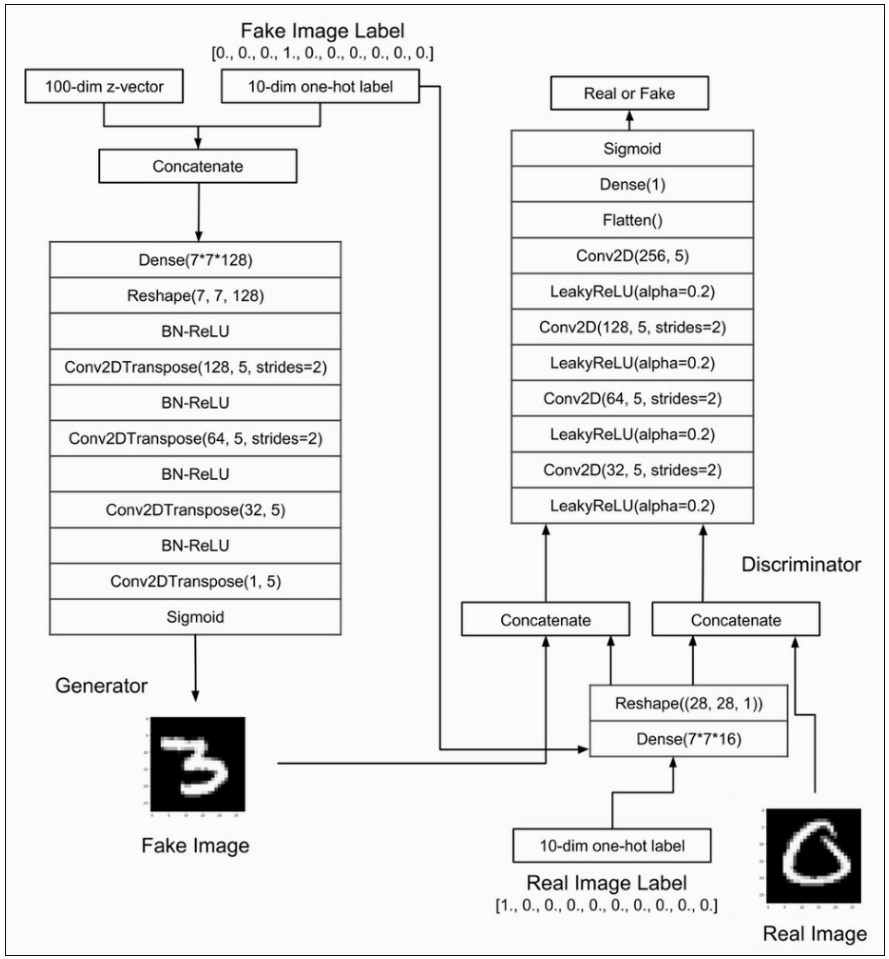


a) GAN architecture



CGAN architecture

GANs and cGANs (III)



https://medium.com/datadriveninvestor/an-introduction-to-conditional-gans-cgans-727d1f5bb011