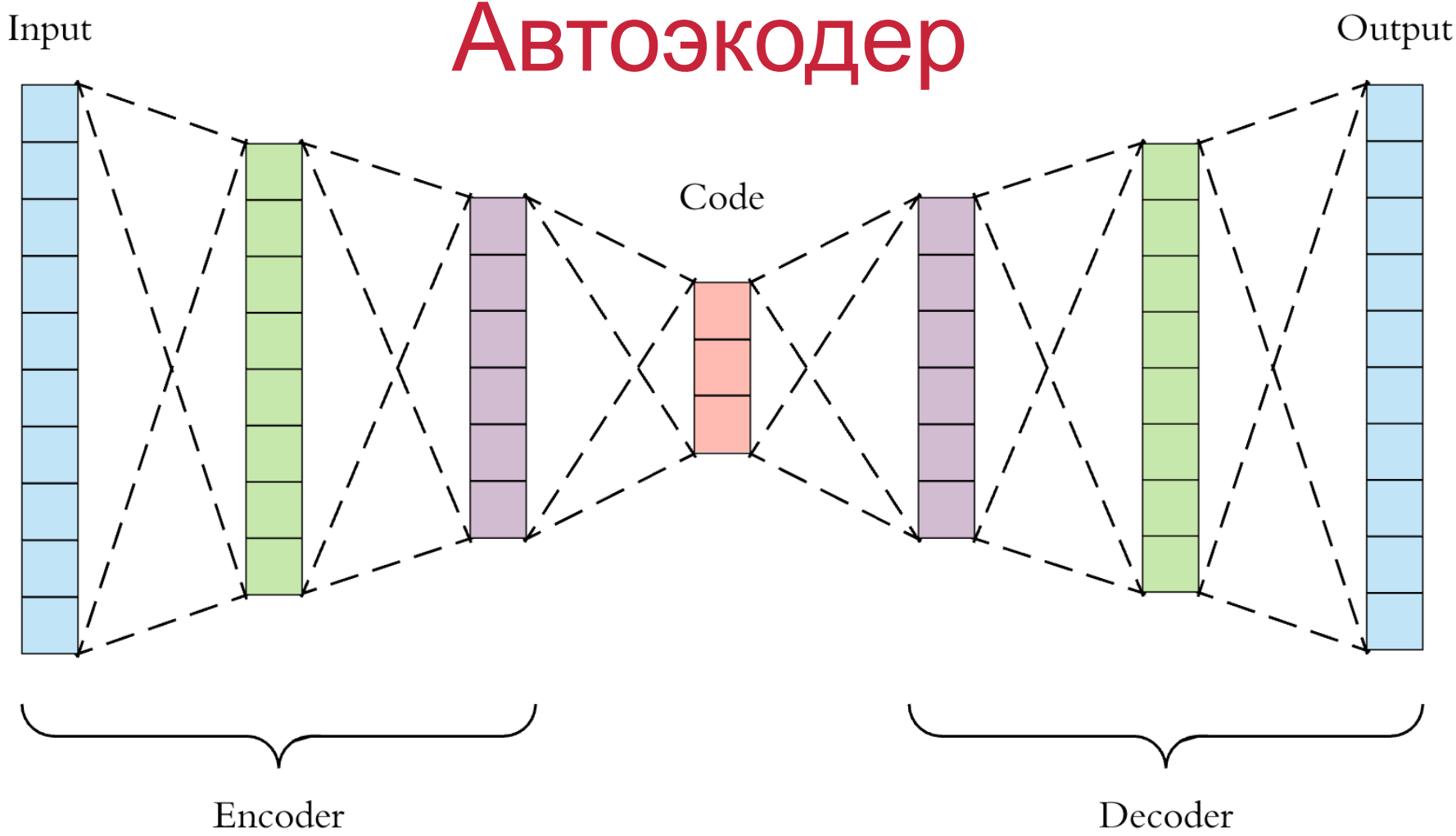
DL: Сверточные сети
Приложения

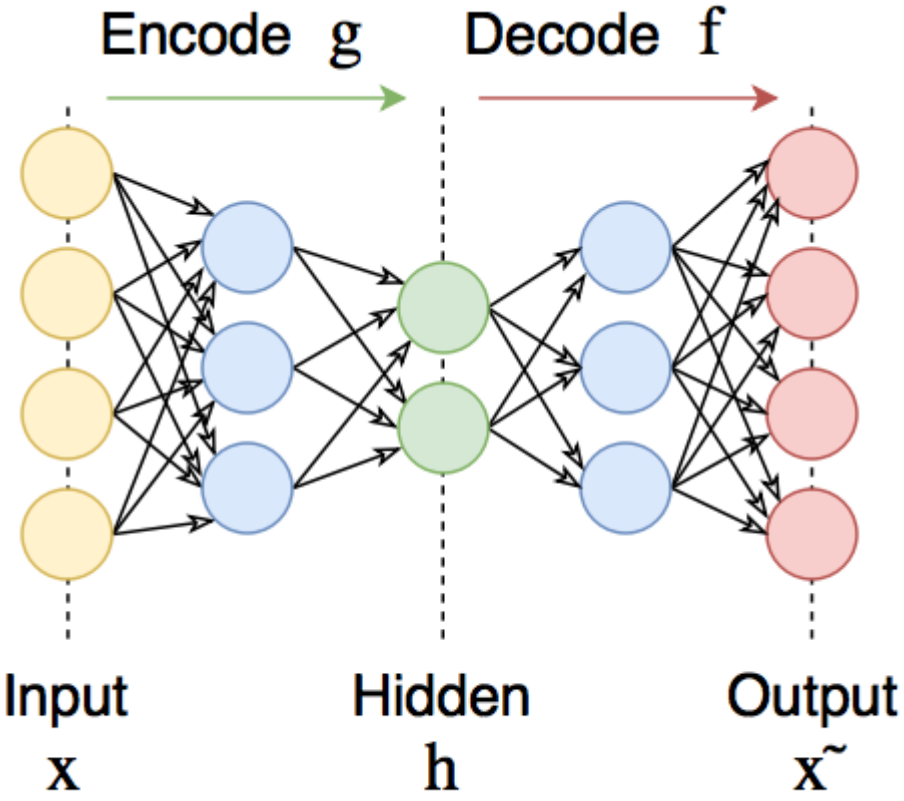
План

- Автоэнкодер
- Сиамские сети
- GAN

Автоэкодер



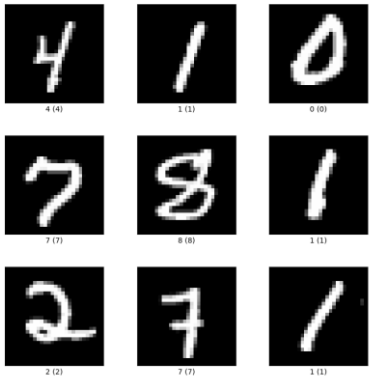
Автоэкодер



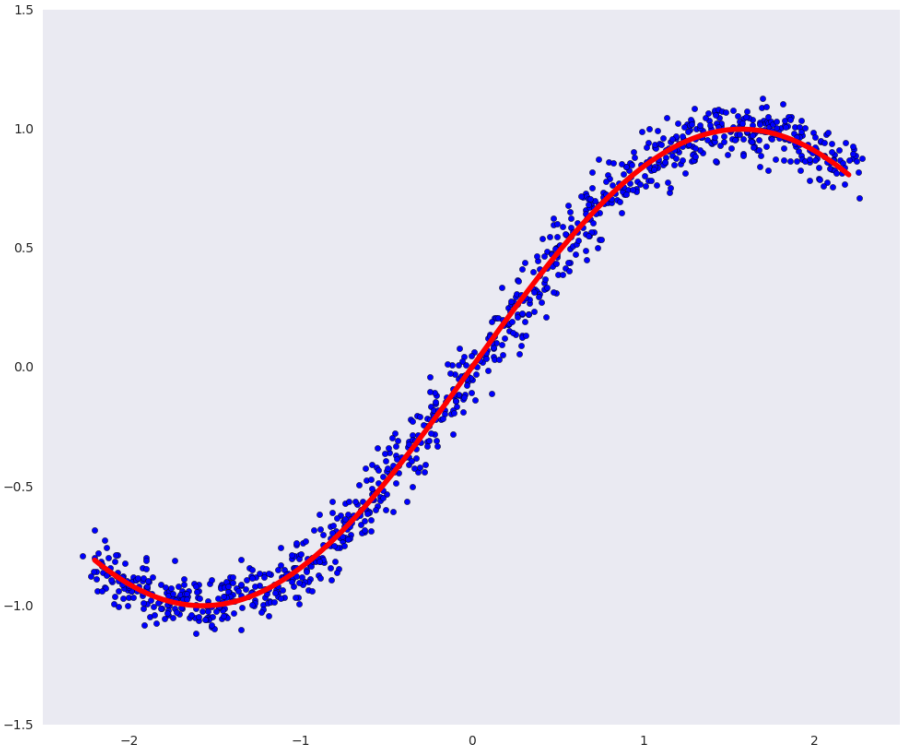
$h = g(x)$

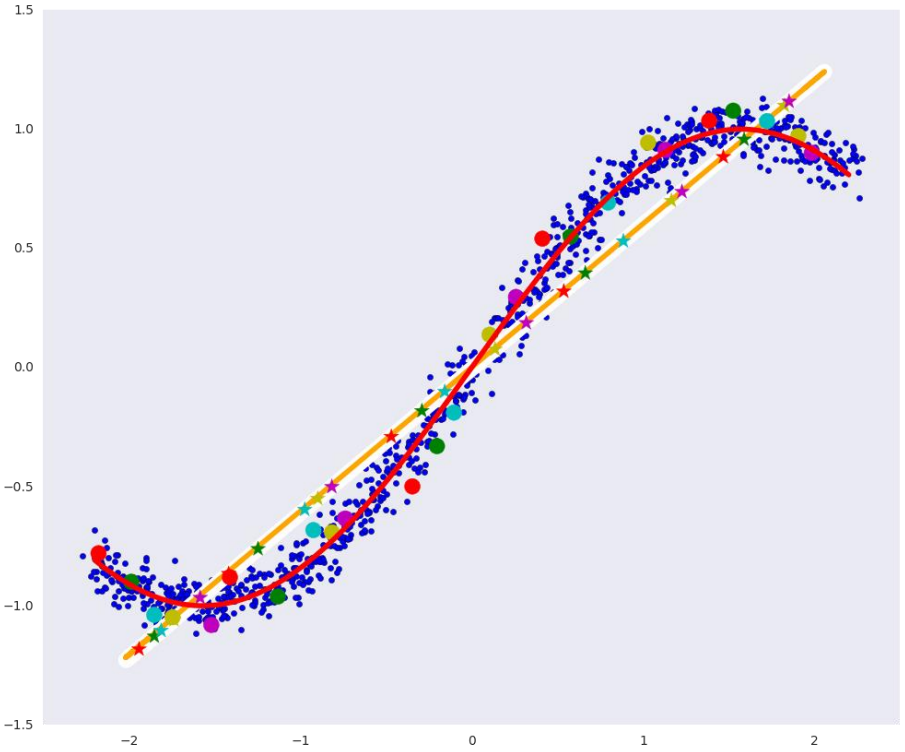
$\mathrm{x} = f(h)$

$E = \|x - g(f(x))\|$



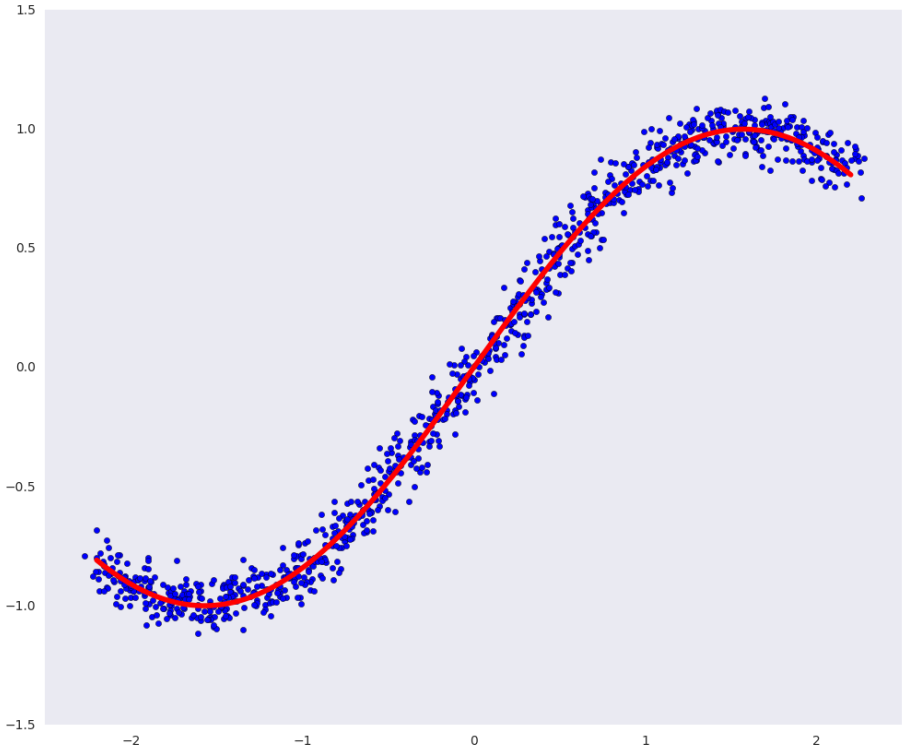
Автоэкодер: manifold learning

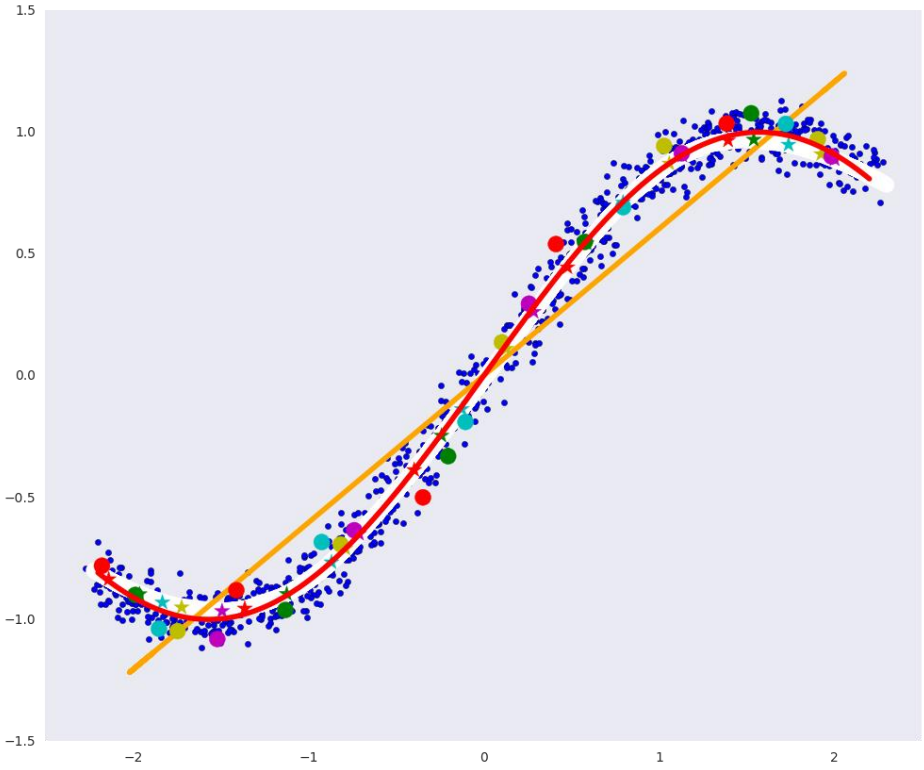




https://habr.com/ru/post/331382/

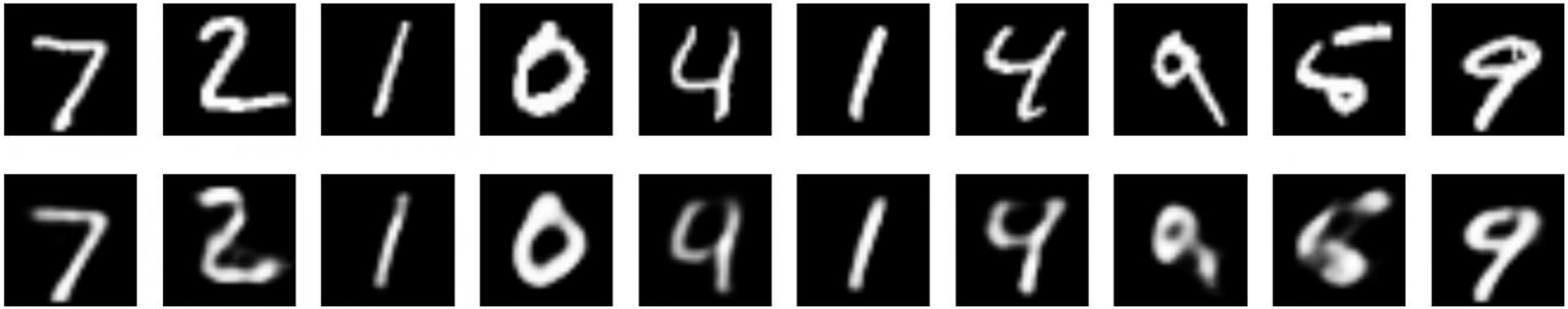
Автоэкодер: manifold learning



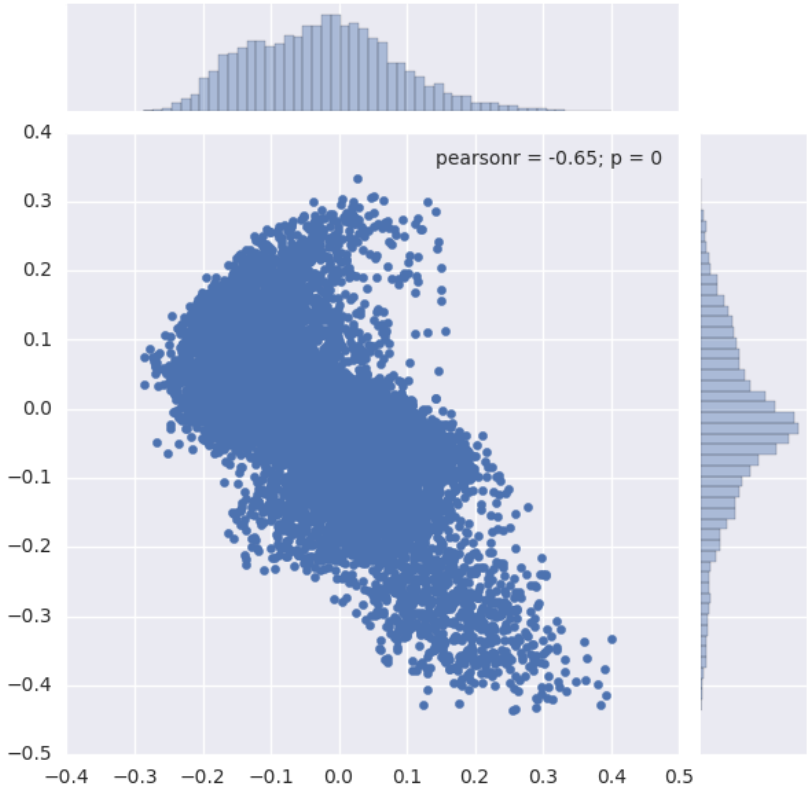


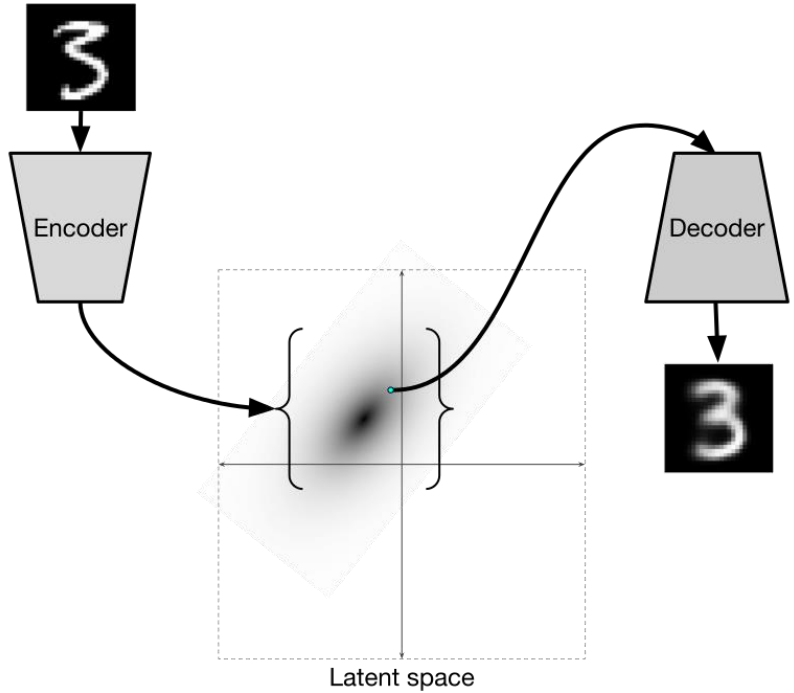
https://habr.com/ru/post/331382/

Автоэкодер: manifold learning

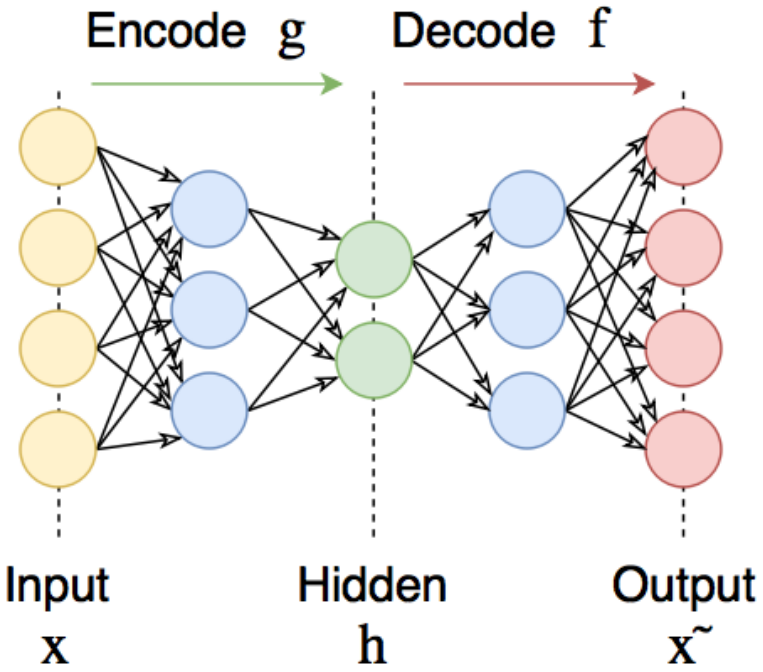


VAE: Вариационный автоэнкодер





VAE: Вариационный автоэнкодер



$$\mathrm{P}(X) = \int_H P(X|h)P(h)dh \qquad \mathrm{P}(X|h) = f(h) + \varepsilon$$

$$\mathrm{P}(X;\theta) = \int_H P(X|h;\theta)P(h)dh \qquad \mathrm{P}(X|h;\theta) = f(h;\theta) + \varepsilon$$

$$\mathrm{P}(X|h;\theta) = N(X|f(h;\theta), \sigma^2 I)$$

VAE: Вариационный автоэнкодер

$$\mathrm{P}(X) = \int_H P(X|h)P(h)dh \qquad \mathrm{P}(X|h;\theta) = N(X|f(h;\theta),\sigma^2 I)$$

Выберем подмножество $H' \in H$ из которого мы получаем множество X

Введем распределение $Q(H|X)$ которое даст нам те $H \sim Q$, которые привели к X

$$\mathrm{KL}[Q(H|X)||P(H,X)] = E_{H\sim Q}[logQ(H|X) - logP(H|X)]$$

$$\mathrm{KL}[Q(H|X)||P(H,X)] = E_{H\sim Q}[logQ(H|X) - logP(X|H) - \log P(H)] + logP(X)$$

$$\mathrm{KL}[Q(H|X)||P(H,X)] = KL[Q(H|X)||P(H)] - E_{H\sim Q}[logP(X|H)] + \mathrm{logP}(X)$$

$$\log P(X) - \mathrm{KL}[Q(H|X)||P(H,X)] = E_{H\sim Q}[logP(X|H)] - KL[Q(H|X)||P(H)]$$

VAE: Вариационный автоэнкодер

$$\mathrm{P}(X) = \int_H P(X|h)P(h)dh$$

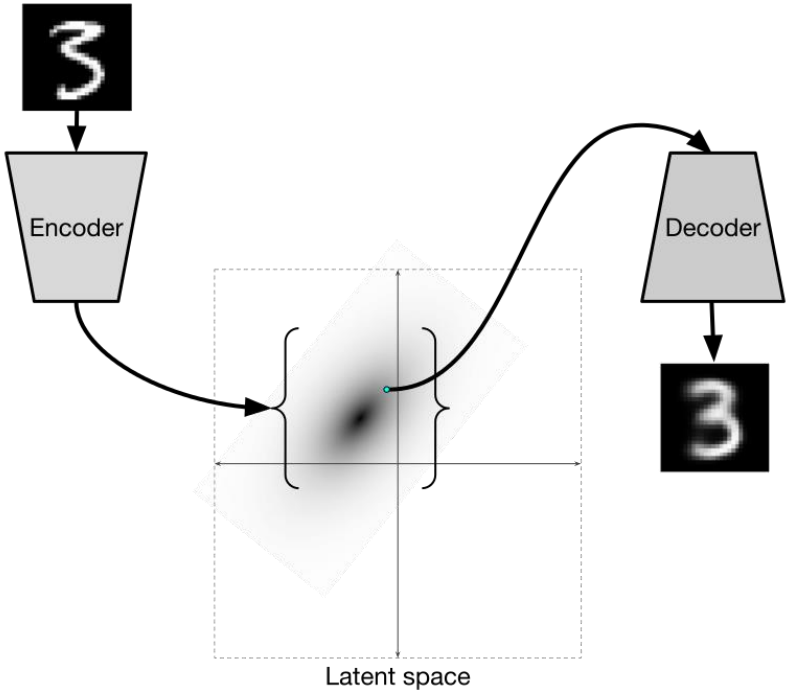
$$\log P(X) - \mathrm{KL}[Q(H|X)||P(H,X)] = E_{H\sim Q}[logP(X|H)] - KL[Q(H|X)||P(H)]$$

$$\log P(X|\theta_2) - \mathrm{KL}[Q(H|X;\theta_1)||P(H,X;\theta_2)] = E_{H\sim Q}[logP(X|H;\theta_2)] - KL[Q(H|X;\theta_1)||N(0,I)]$$

$$Q(H|X;\theta_1) - ?$$

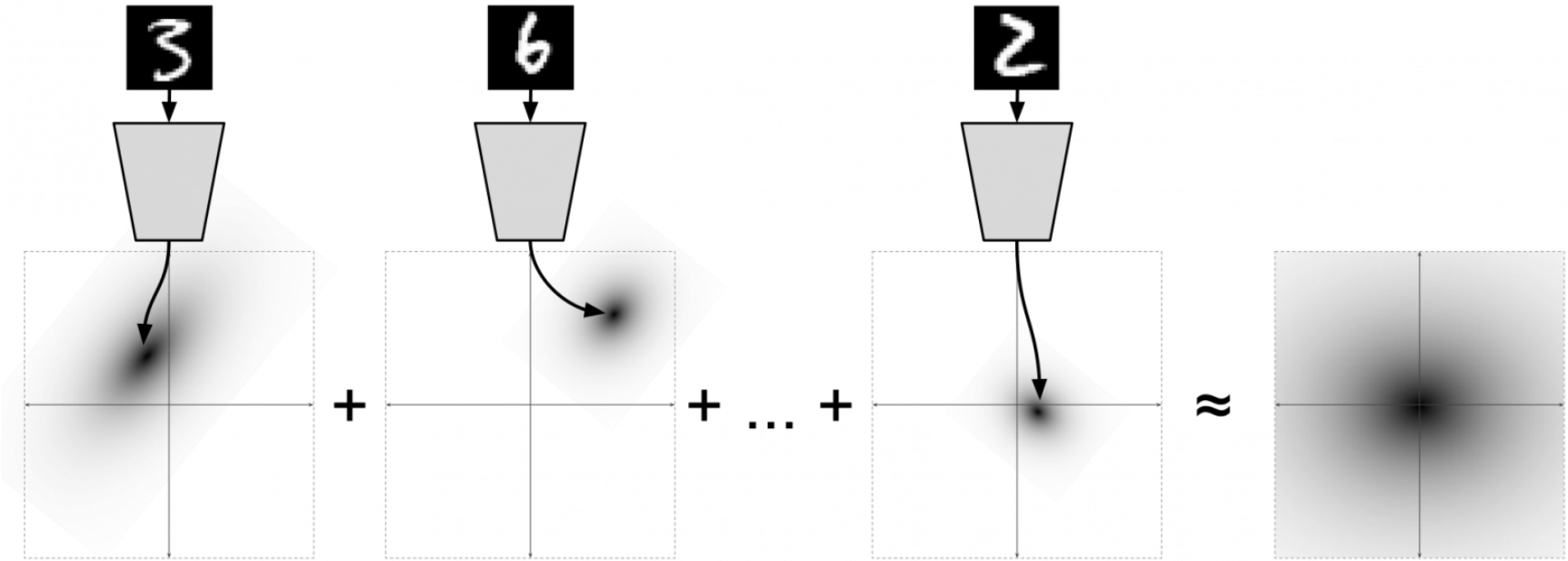
VAE: Вариационный автоэнкодер

$$Q(H|X;\theta_1) = N(\mu(X;\theta_1), \sum(X;\theta_1))$$



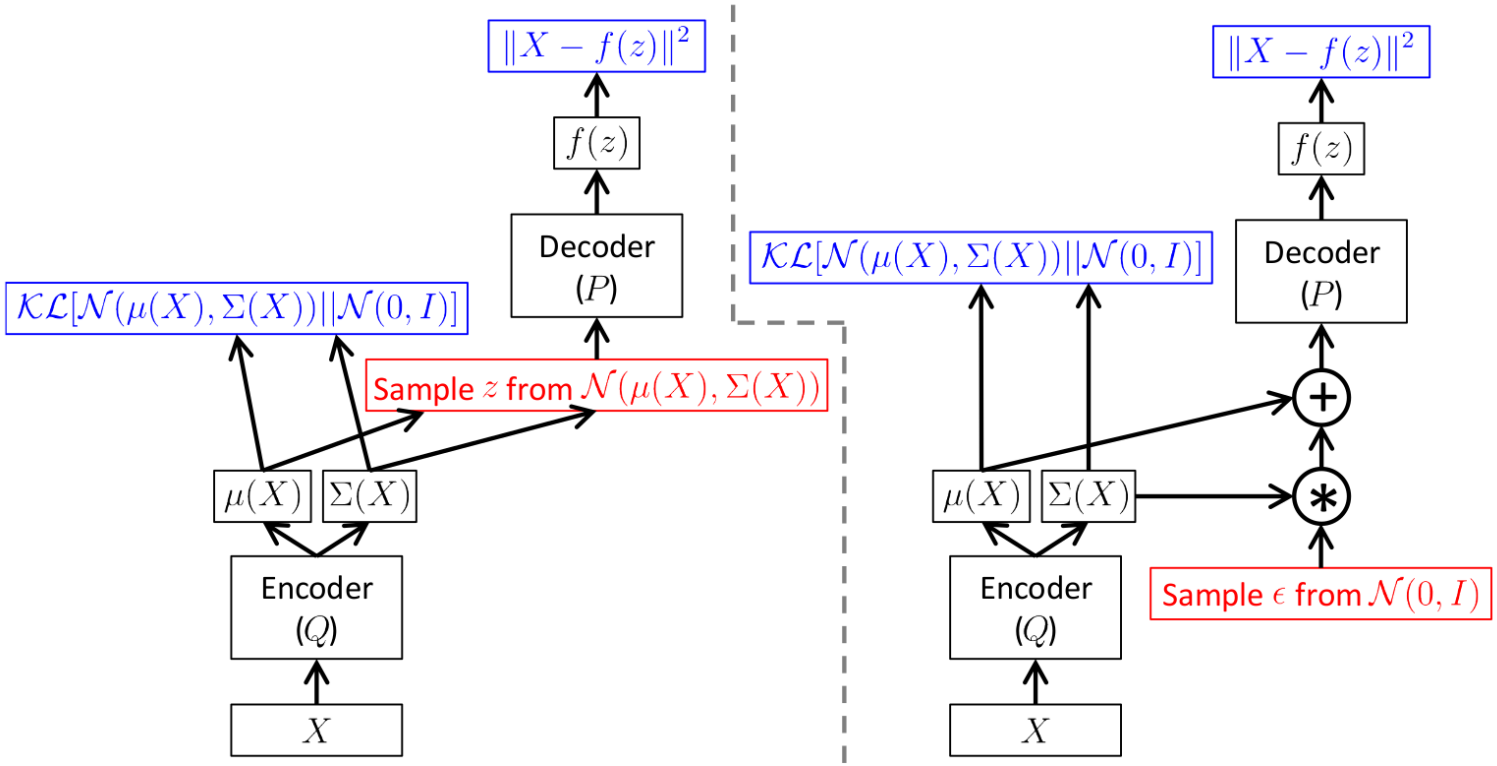
VAE: Вариационный автоэнкодер

$$P(H|X) = N(\mu(X), \sum(X)) \qquad P(H) = N(0, I)$$

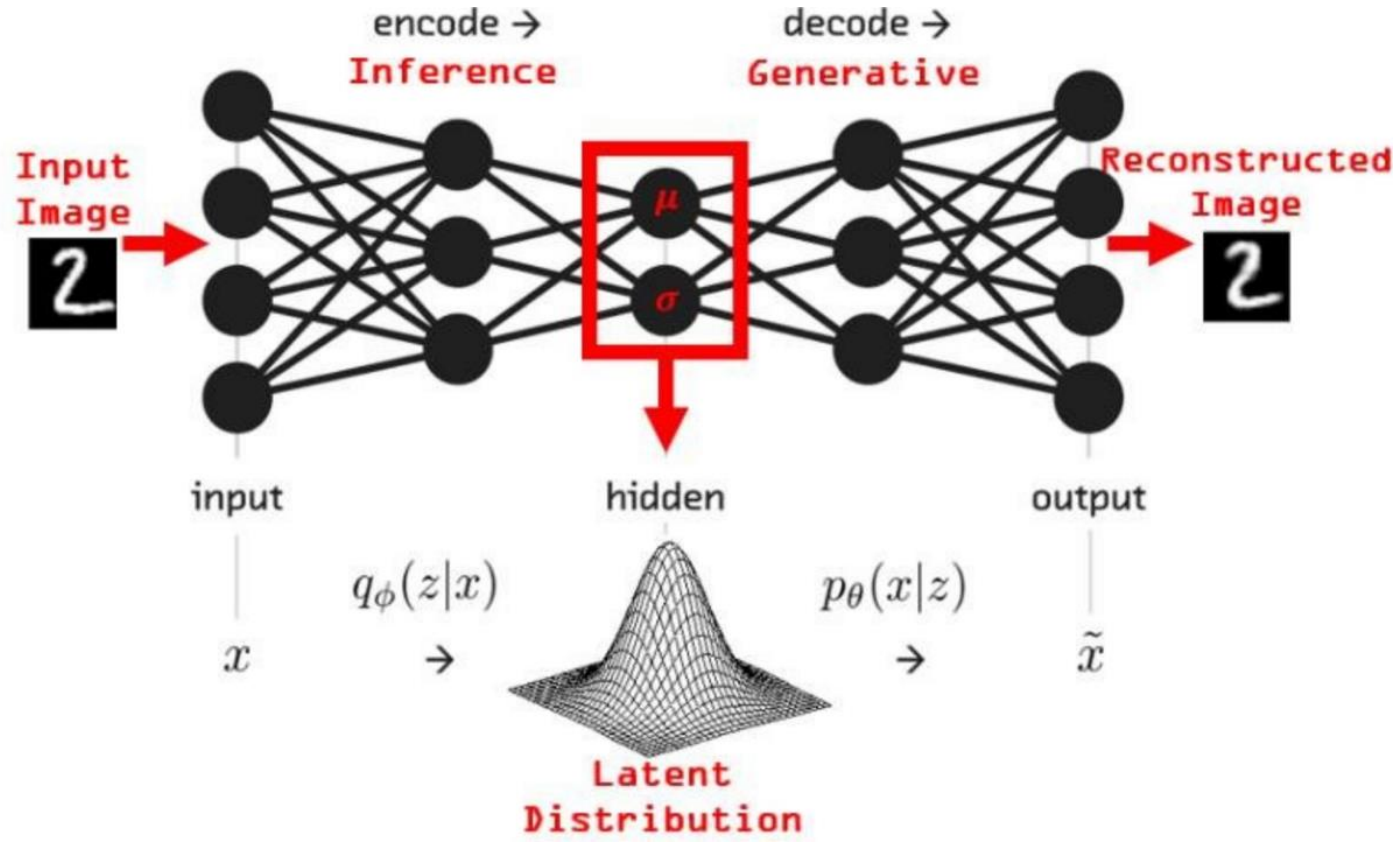


$$KL[Q(H|X;\theta_1)||N(0,I)] = 0.5(tr(\sum X) + \mu(X)^T\mu(X) - k - \log\det\sum(X))$$

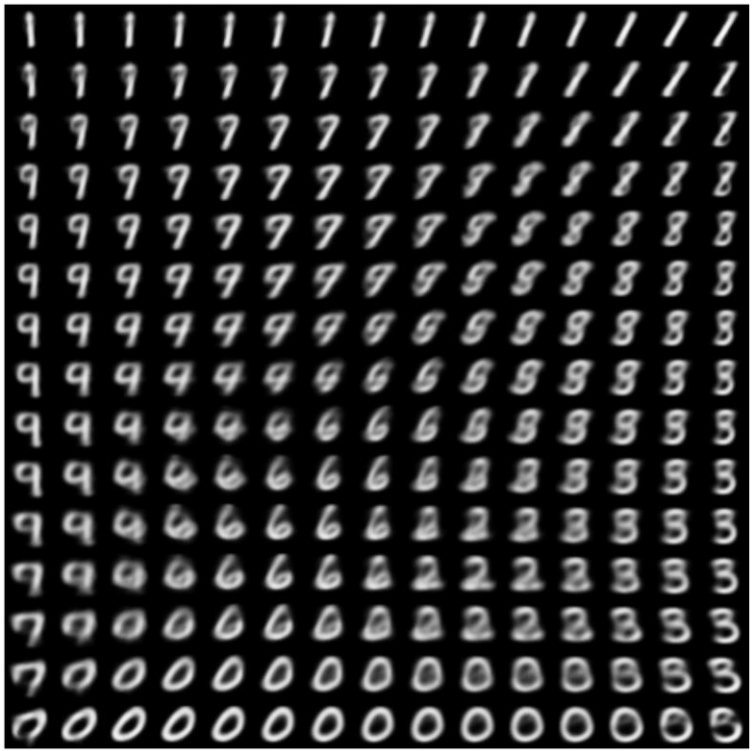
VAE: Вариационный автоэнкодер

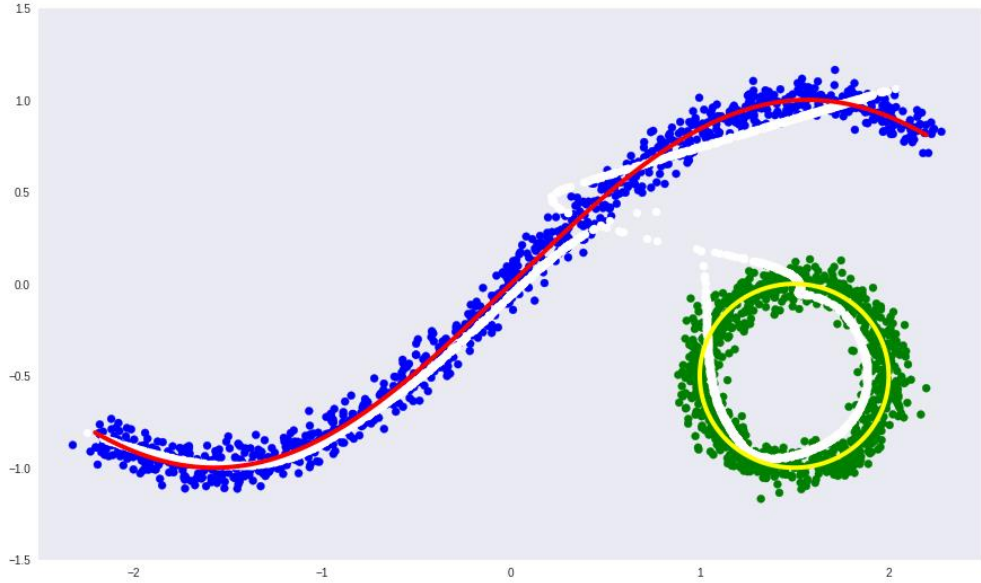


VAE: Вариационный автоэнкодер

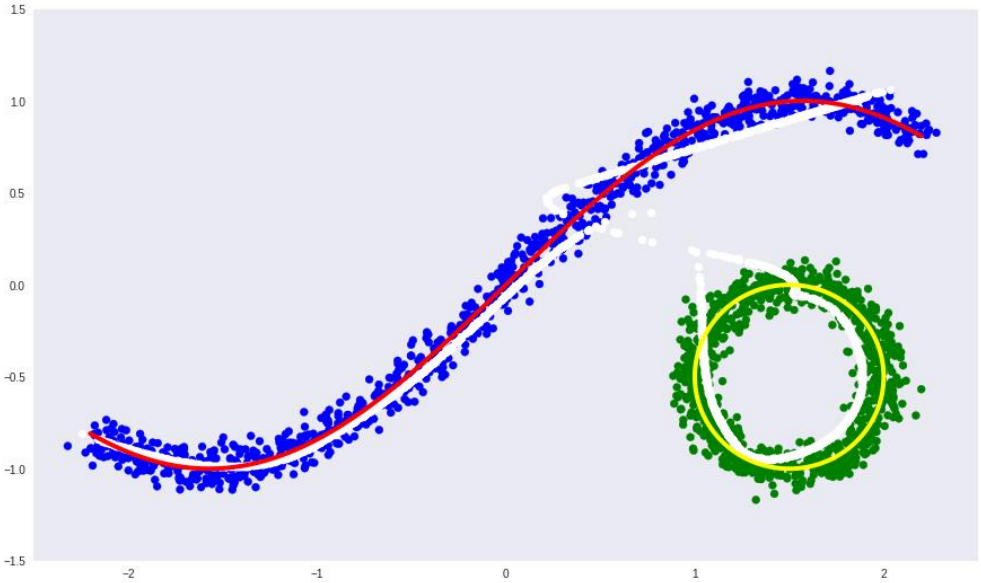


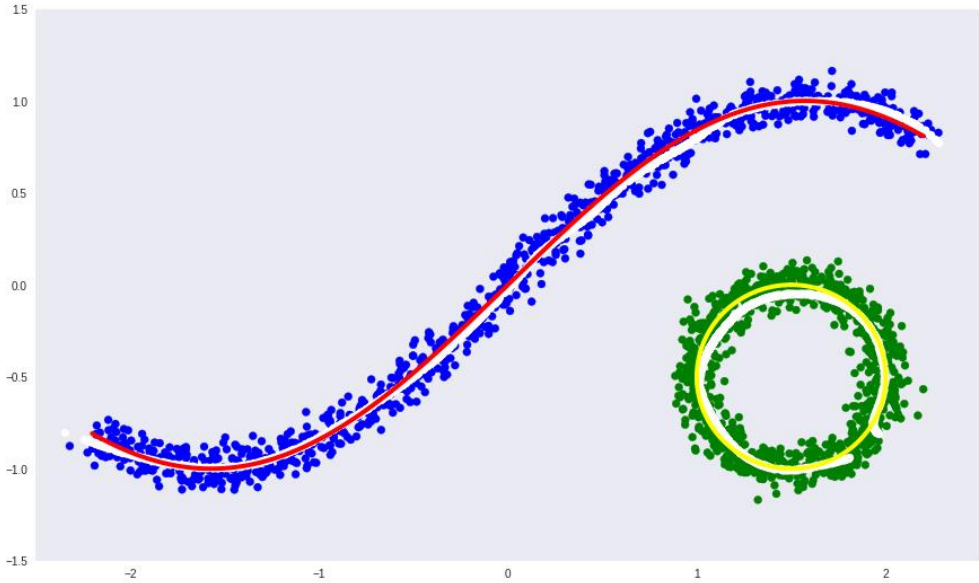
VAE: недостатки



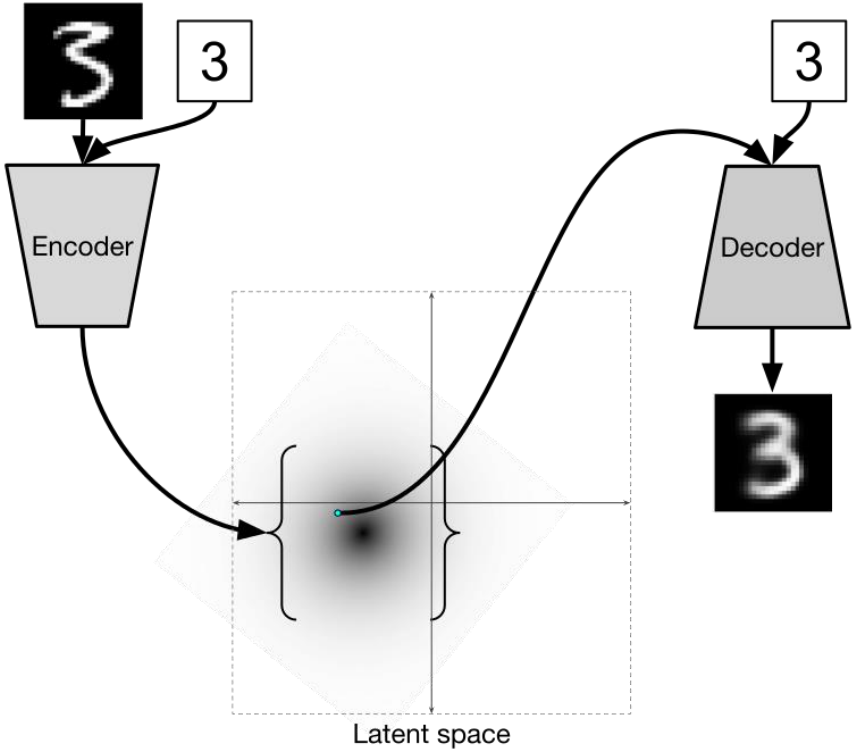


Conditional VAE



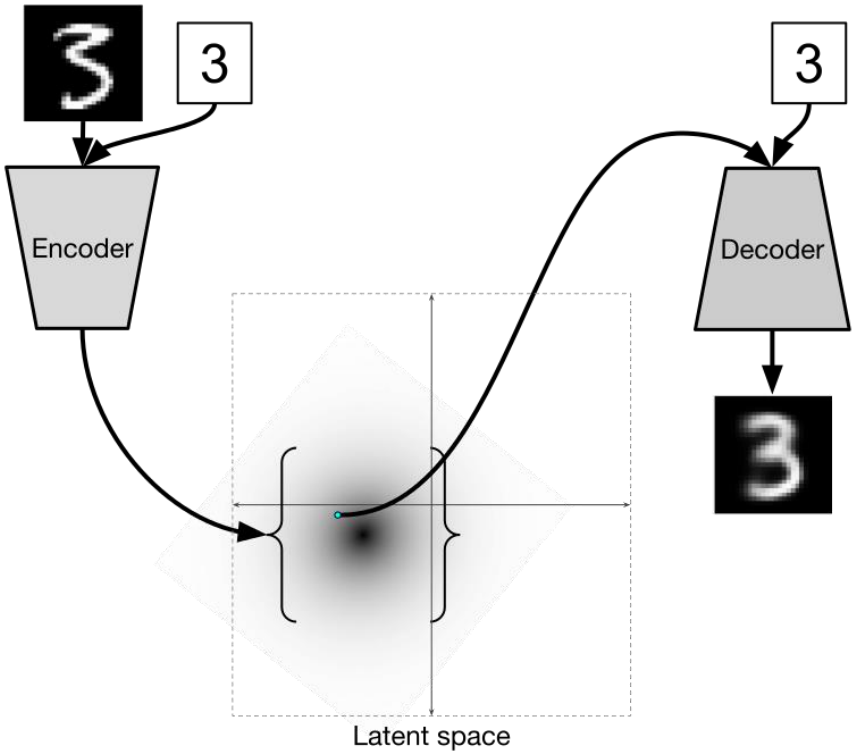


Conditional VAE



$$\log P(X|Y;\theta_2) - \mathrm{KL}[Q(H|X,Y;\theta_1)||P(H|X,Y;\theta_2)] = E_{H\sim Q}[logP(X|H,Y;\theta_2)] - KL[Q(H|X,Y;\theta_1)||N(0,I)]$$

Conditional VAE



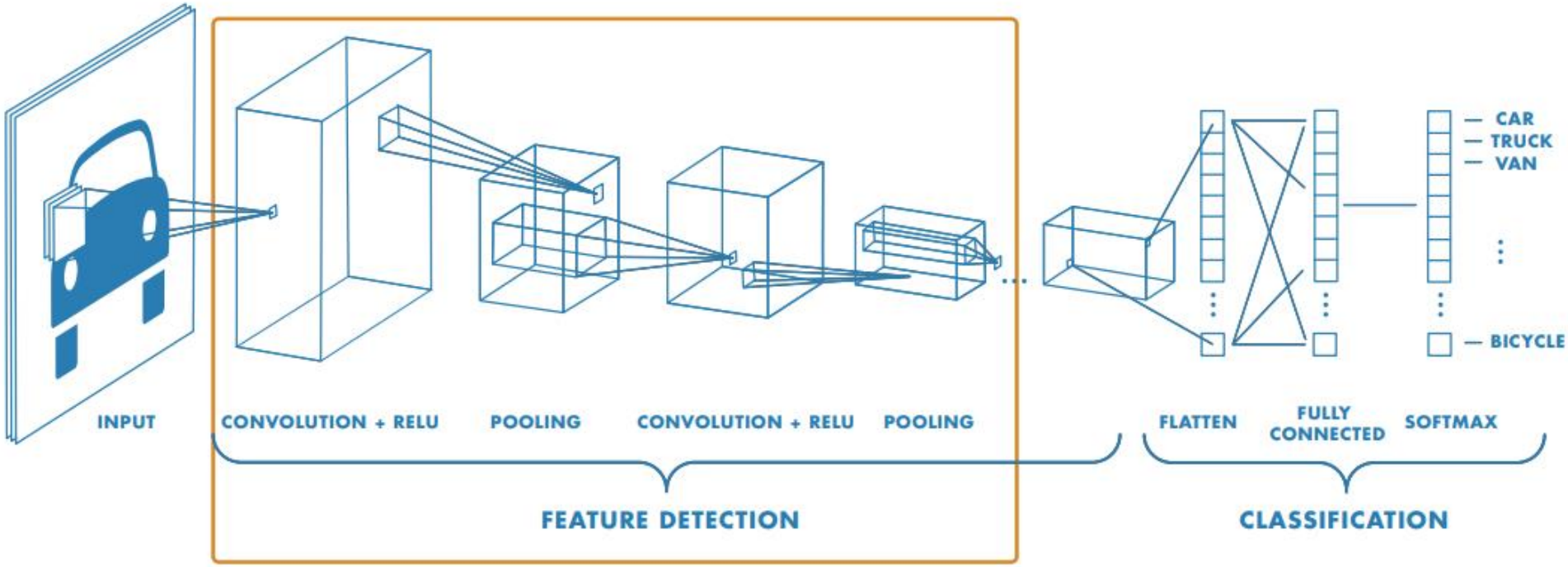
https://github.com/lyeoni/pytorch-mnist-VAE/blob/master/pytorch-mnist-VAE.ipynb

https://github.com/lyeoni/pytorch-mnist-CVAE/blob/master/pytorch-mnist-CVAE.ipynb

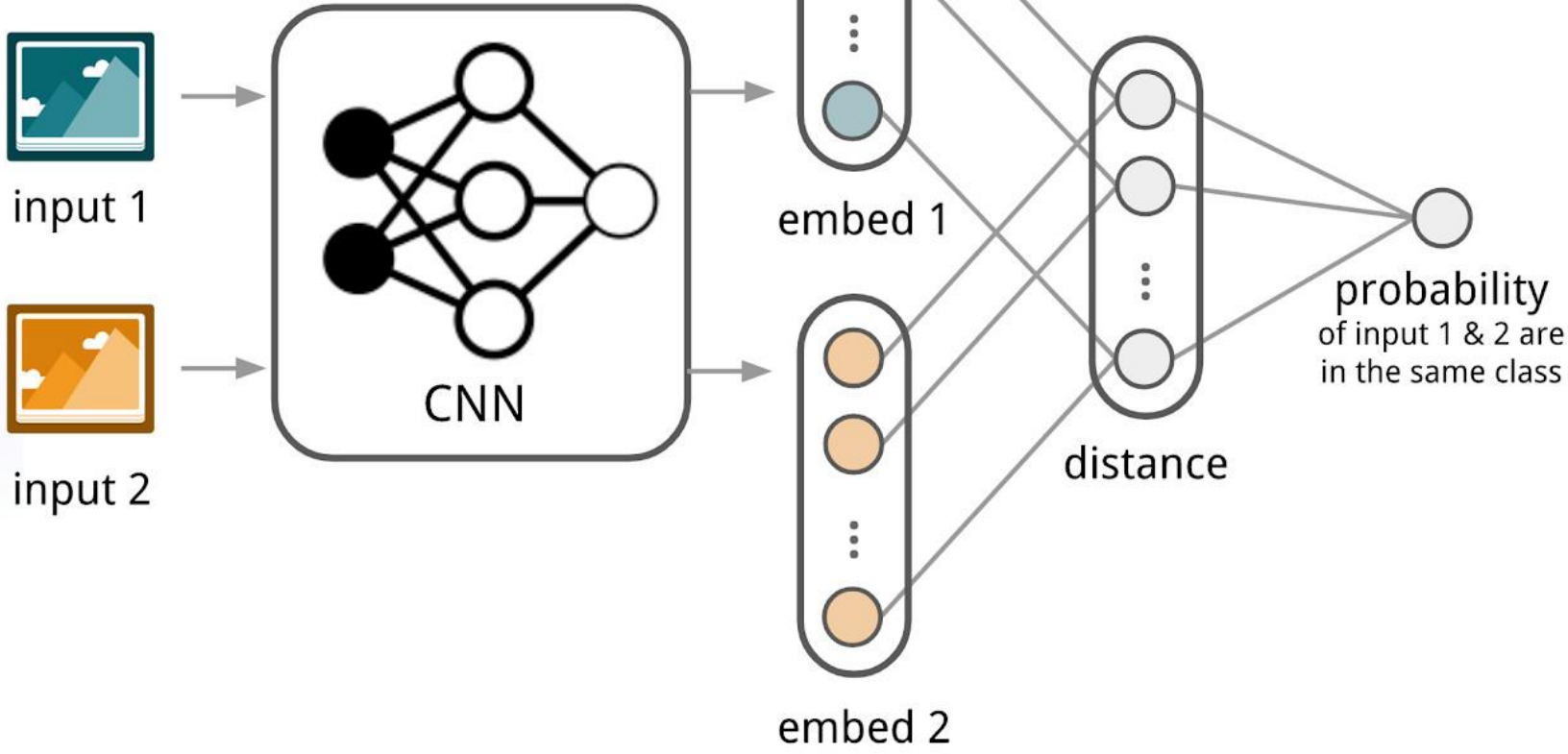
Conditional VAE: перенос стиля

- *Обучаем CVAE на картинках с метками*
- *Кодируем стиль заданной картинки в H*
- *Меняем метки Y, создаем из закодированного H новые картинки*

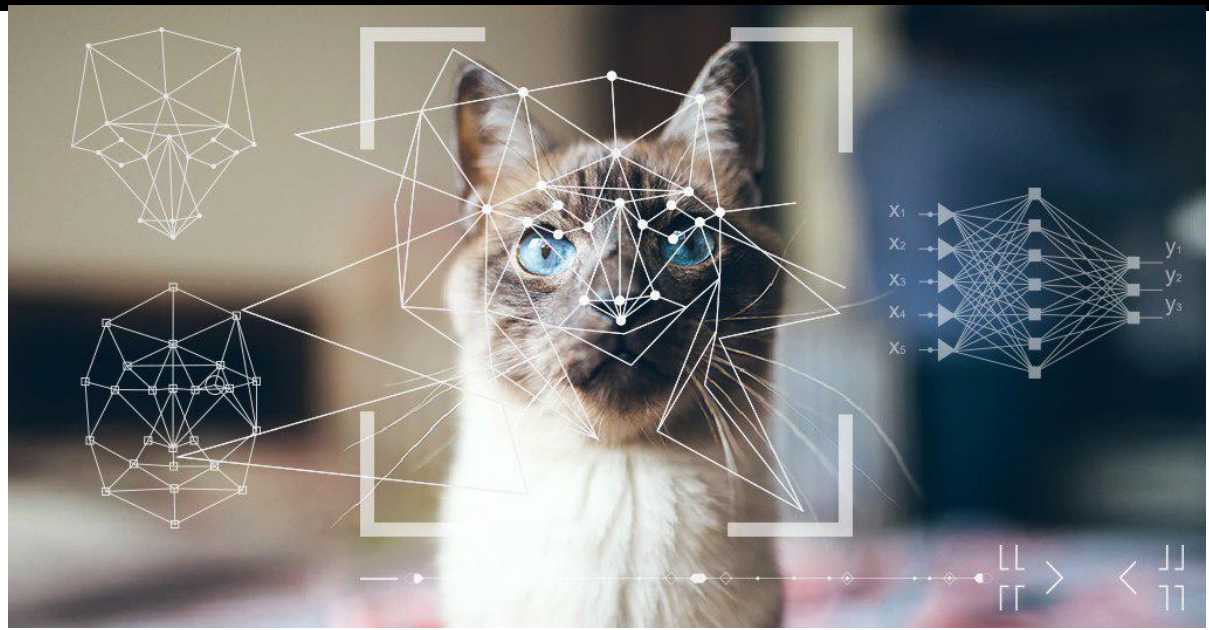
Передача обучения



Сиамские сети



Сиамские сети
Triplet loss



$$L = max(\|F(A) - F(P)\|^2 - \|F(A) - F(N)\|^2 + \alpha, 0)$$

$$L = YD^2 + (1 - Y)max(\alpha - D, 0)^2$$

$Y = 1$ для изображений одного класса, Y = 0 для изображений разного класса

Сиамские сети: Softmax loss

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{i=0}^{C} e^{z_i}}$$

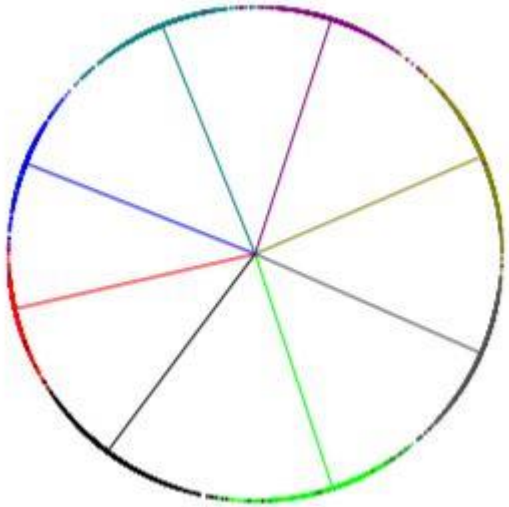
$$L_{softmax} = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{e^{W_{y_i}^T x_i + b_i}}{\sum_{j=1}^{C} e^{W_{y_j}^T x_i + b_j}}$$

W веса слоя классификации (центроиды)
X – embedding входного изображения
b - bias

Сиамские сети: Normalized Softmax Loss, N-Softmax (ArcFace)

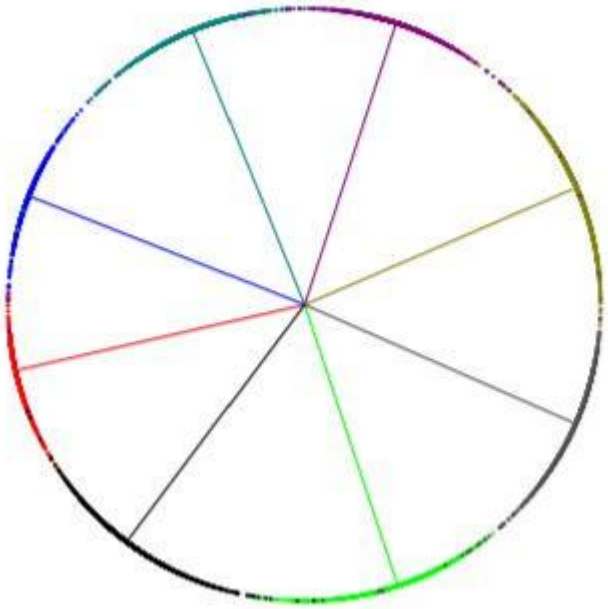
$$cos(\theta) = \frac{(x, y)}{\|x\|\|y\|}$$

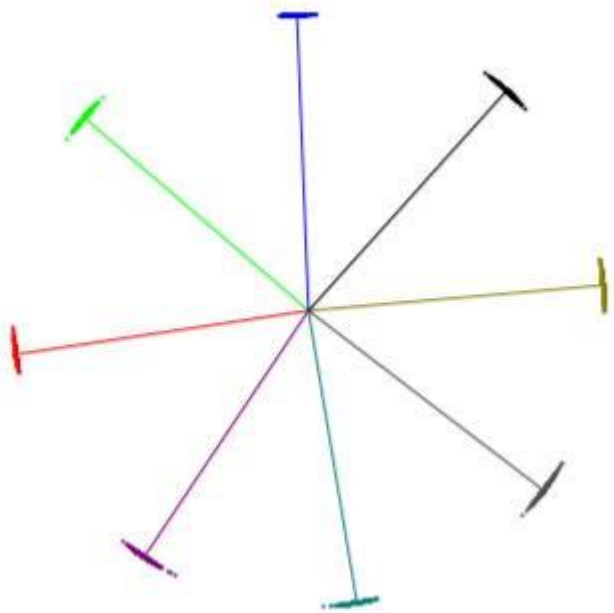
$$W^T X = s\, cos(\theta)$$



$$L_{Nsoftmax} = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{e^{s \cdot cos\left(\theta_{y_i}\right)}}{e^{s \cdot cos\left(\theta_{y_i}\right)} + \sum_{j=1, j \neq y_i}^{C} e^{s \cdot cos\left(\theta_j\right)}}$$

Сиамские сети: Margin-Base Loss

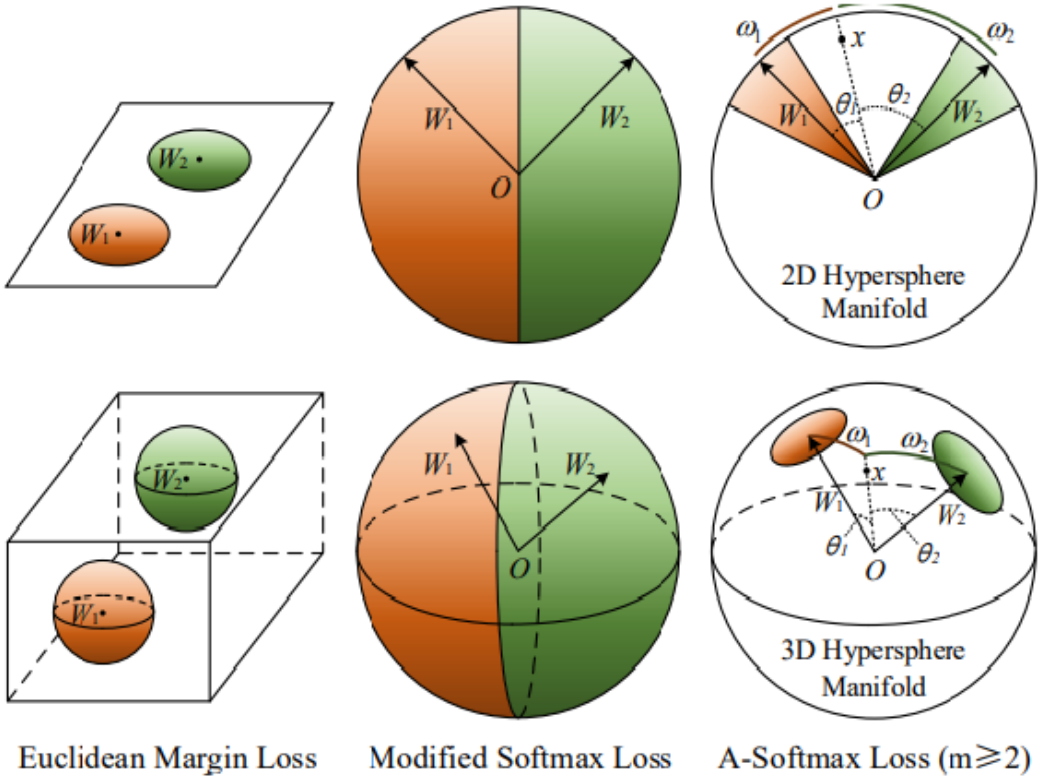




Сиамские сети: SphereFace Loss

$$L_{SphereFace} = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{e^{s\cdot cos\left(m\cdot\theta_{y_i}\right)}}{e^{s\cdot cos\left(m\cdot\theta_{y_i}\right)} + \sum_{j=1, j\neq y_i}^{C} e^{s\cdot cos\left(\theta_j\right)}}$$

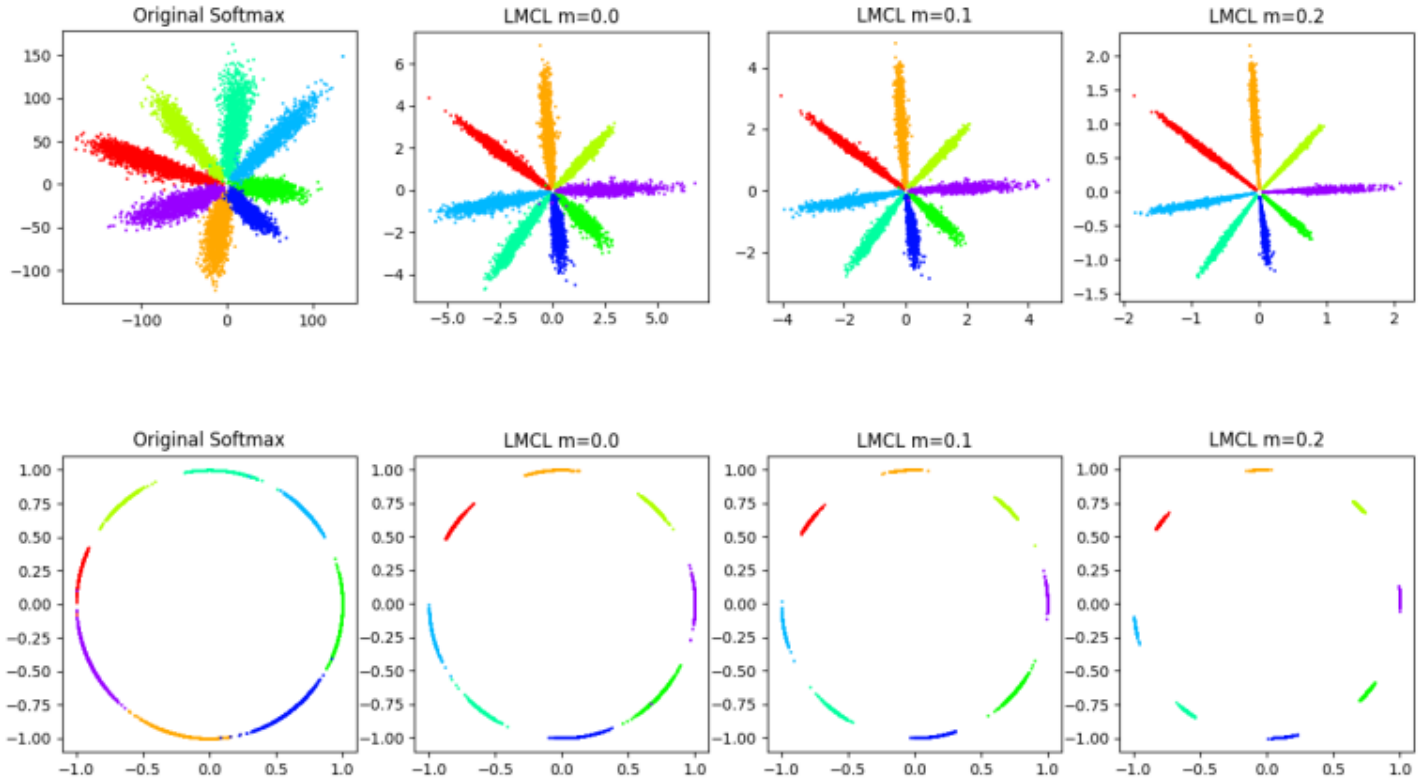
Сиамские сети: SphereFace Loss



Сиамские сети: CosFace Loss

$$L_{CosFace} = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{e^{s\cdot\left(cos\left(\theta_{y_i}\right)-m\right)}}{e^{s\cdot\left(cos\left(\theta_{y_i}\right)-m\right)} + \sum_{j=1,j\neq y_i}^{C} e^{s\cdot cos\left(\theta_j\right)}}$$

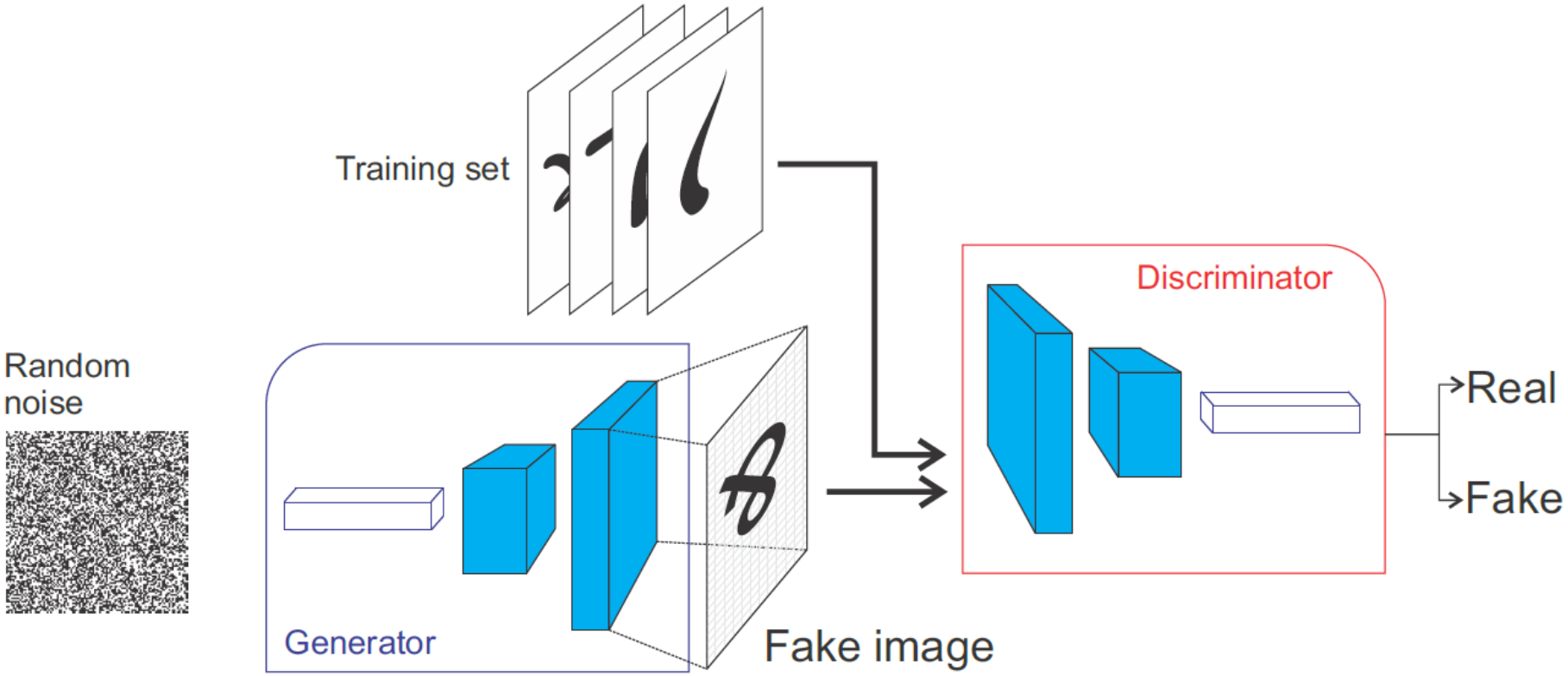
Сиамские сети: CosFace Loss



Сиамские сети: ArcFace Loss

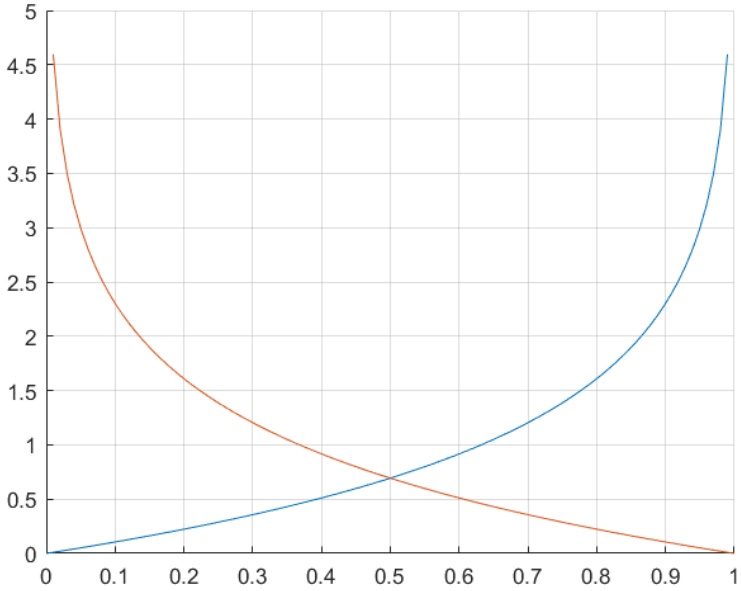
$$L_{SphereFace} = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{e^{s\cdot cos\left(\theta_{y_i}+m\right)}}{e^{s\cdot cos\left(\theta_{y_i}+m\right)} + \sum_{j=1, j\neq y_i}^{C} e^{s\cdot cos\left(\theta_j\right)}}$$

GAN

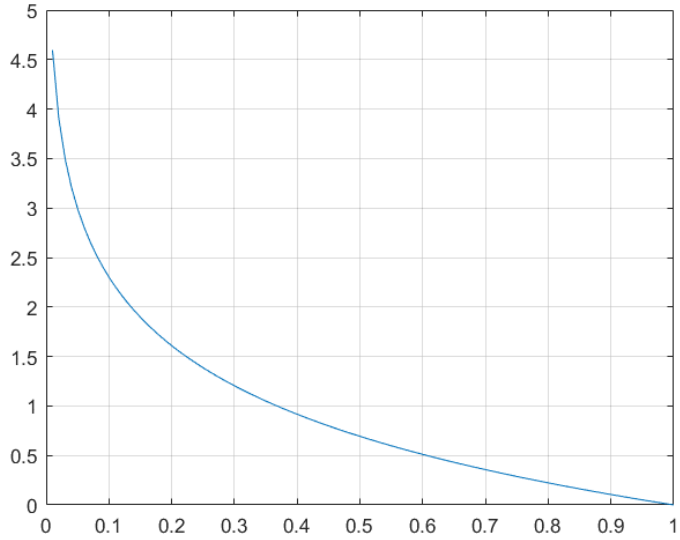

Training set
Random noise
Generator
Fake image
Discriminator
Real
Fake

GAN

$$-E_x[\log(D(x))] - E_z[\log(1 - D(G(z))]$$



$$-E_z[\log(D(G(z))]$$



GAN+VAE

https://github.com/csinva/gan-vae-pretrained-pytorch

https://github.com/rishabhd786/VAE-GAN-PYTORCH