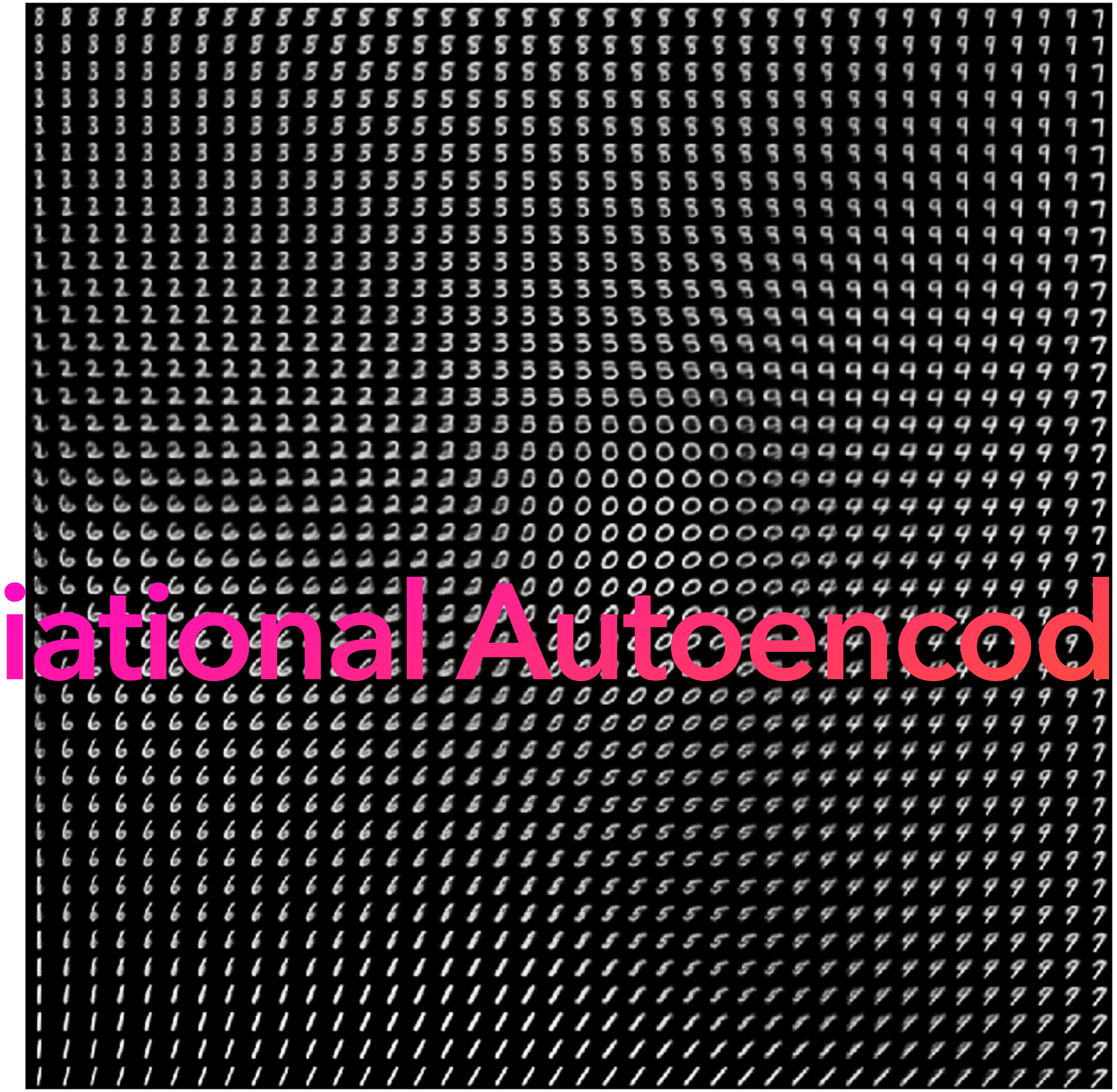
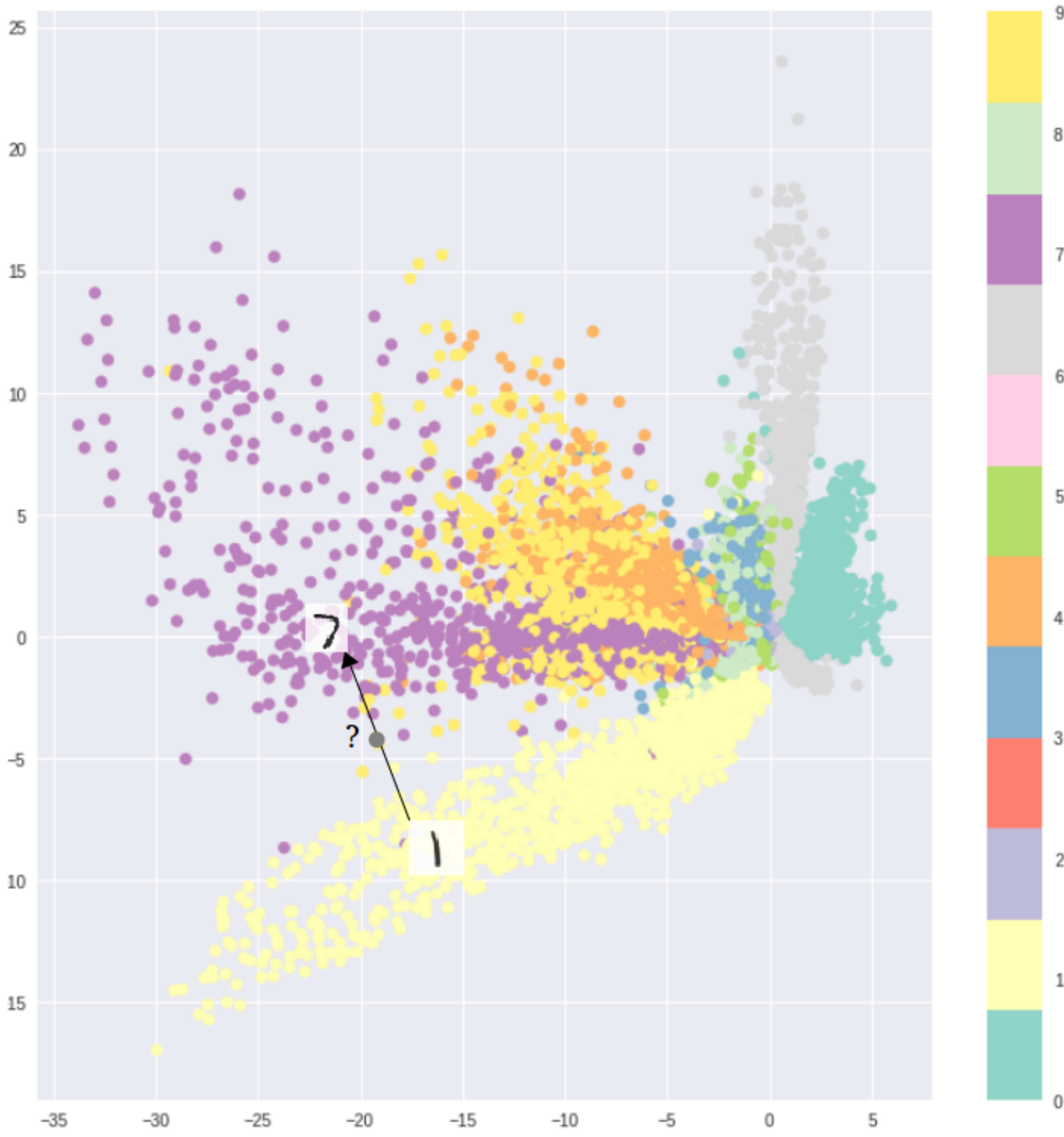
Variational Autoencoders

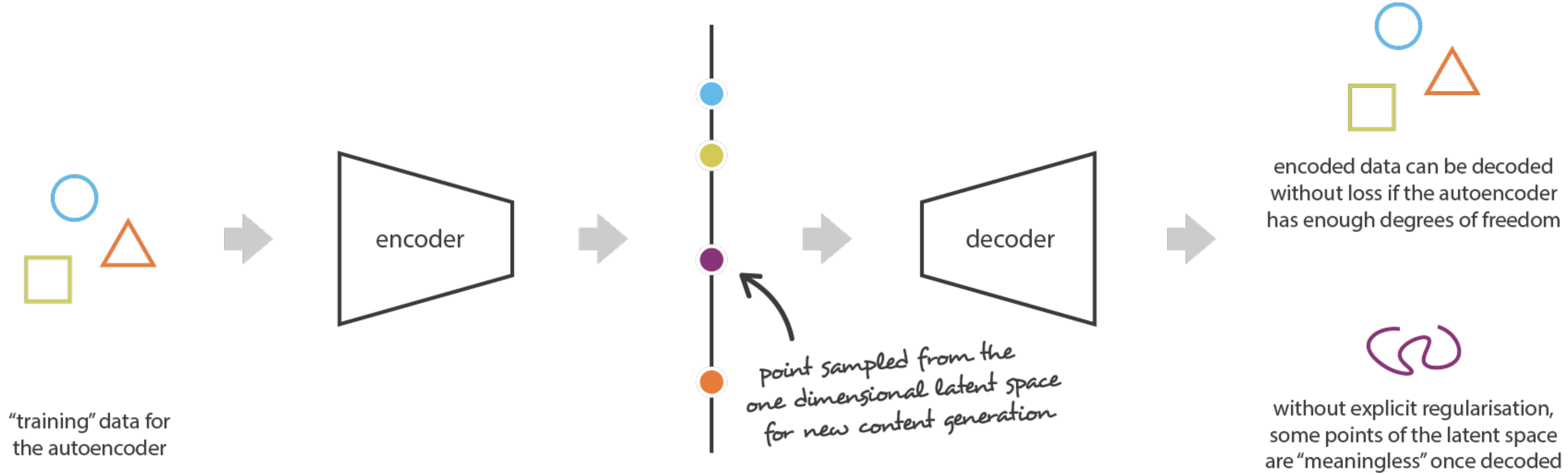


Credits: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Что не так с AE?

- С точки зрения задачи генерации ~Всё

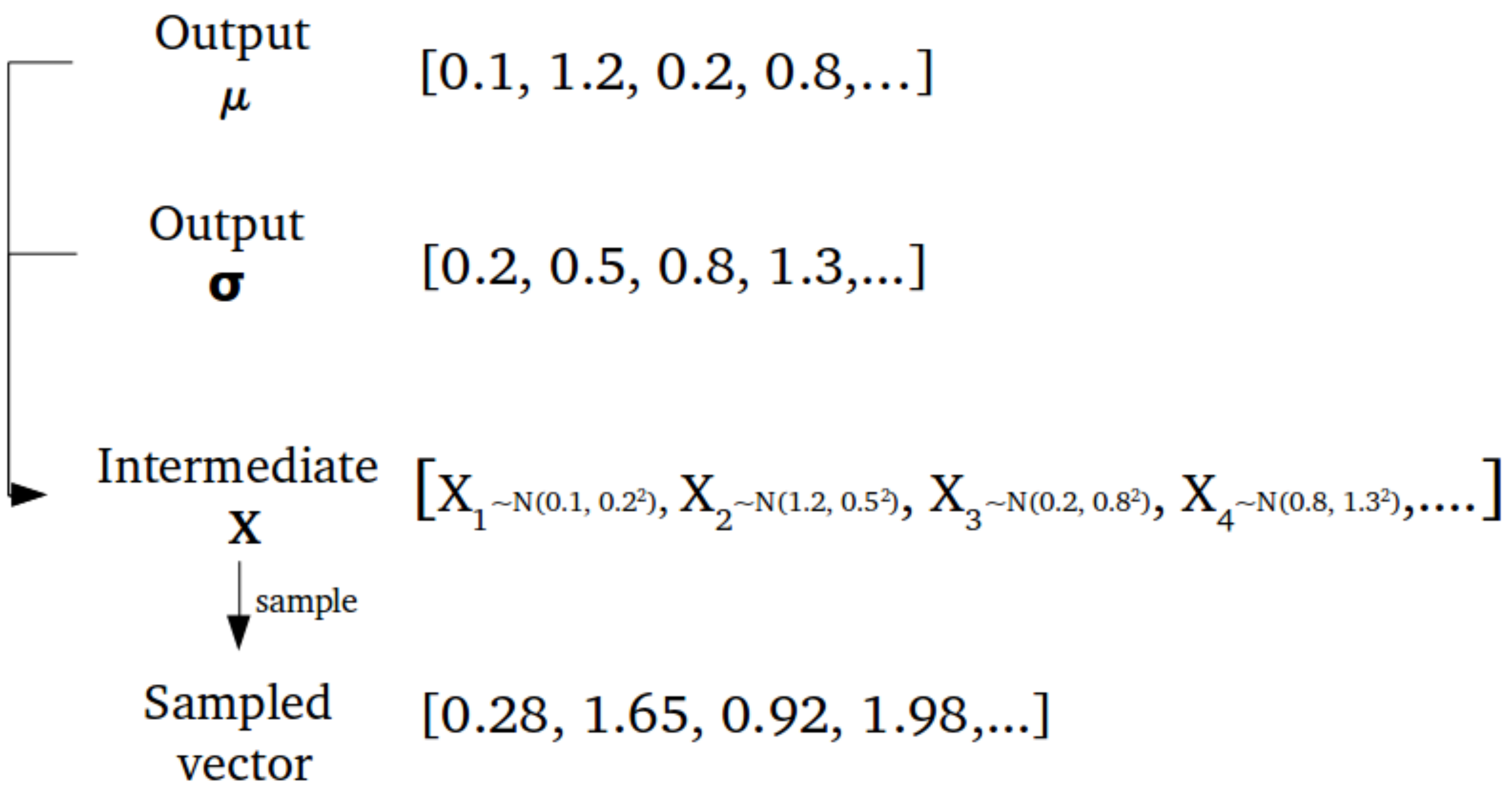


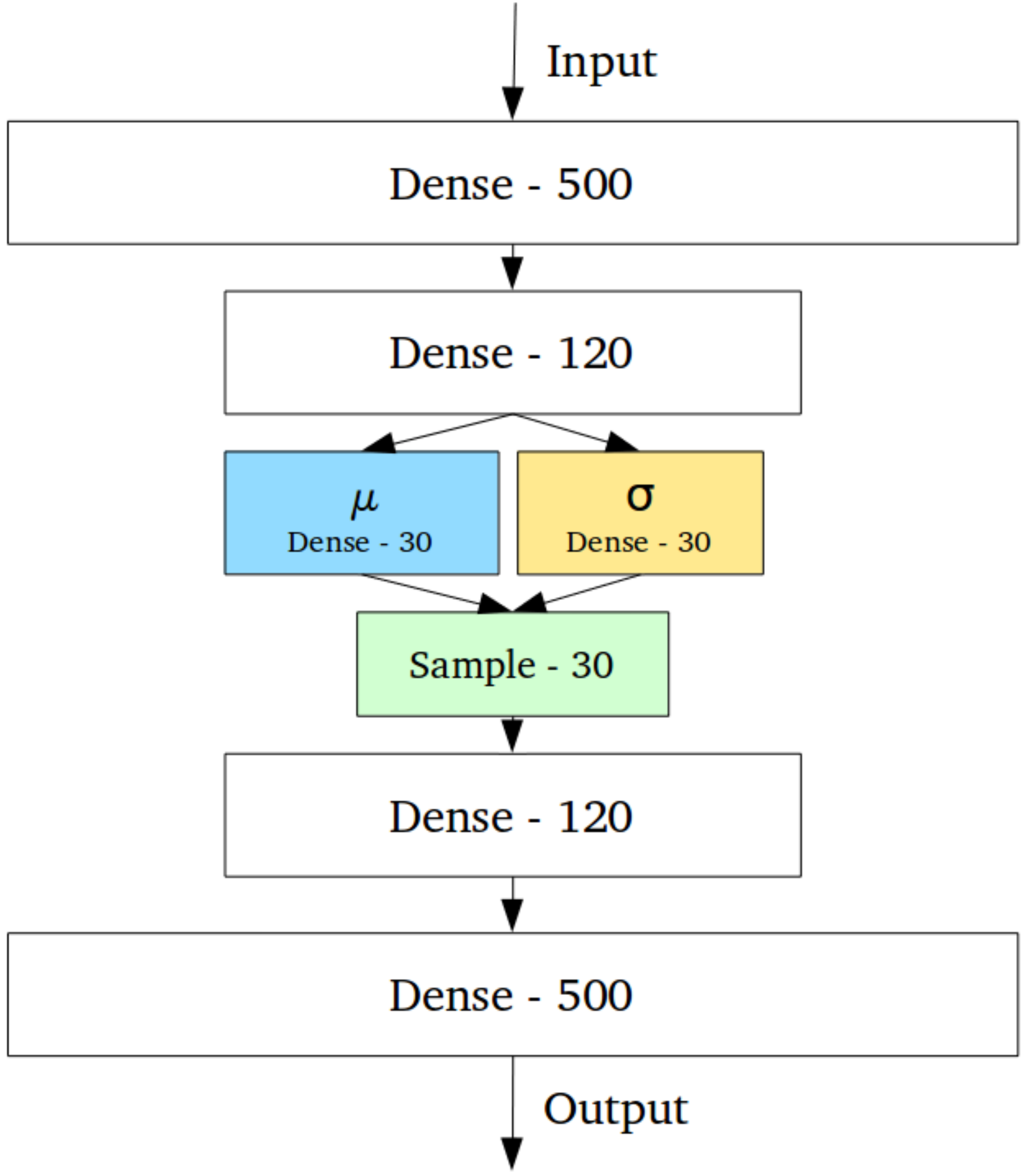


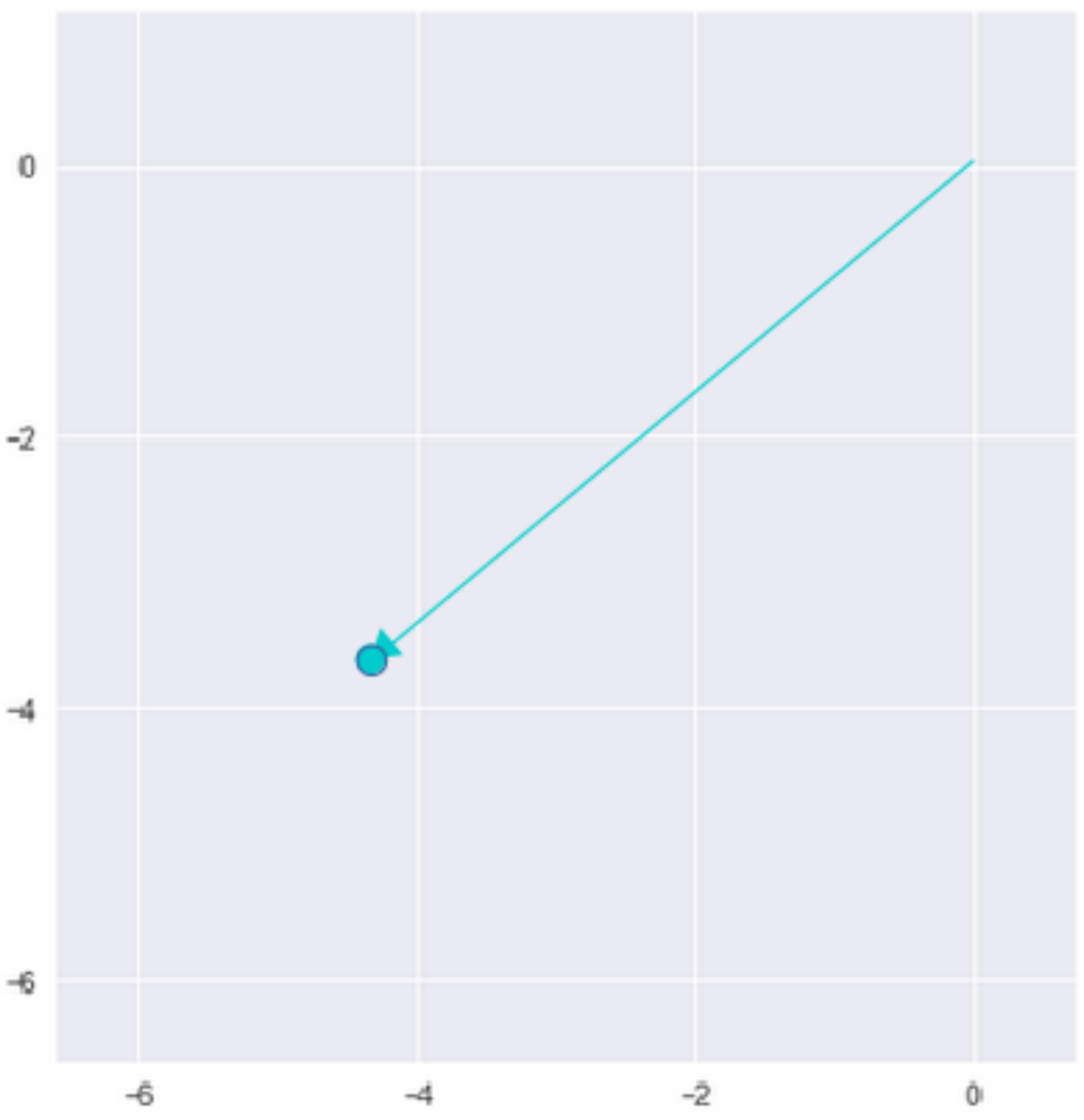
encoder
point sampled from the one dimensional latent space for new content generation
decoder
encoded data can be decoded without loss if the autoencoder has enough degrees of freedom
without explicit regularisation, some points of the latent space are "meaningless" once decoded
"training" data for the autoencoder

VAE (в чём смысОл?)

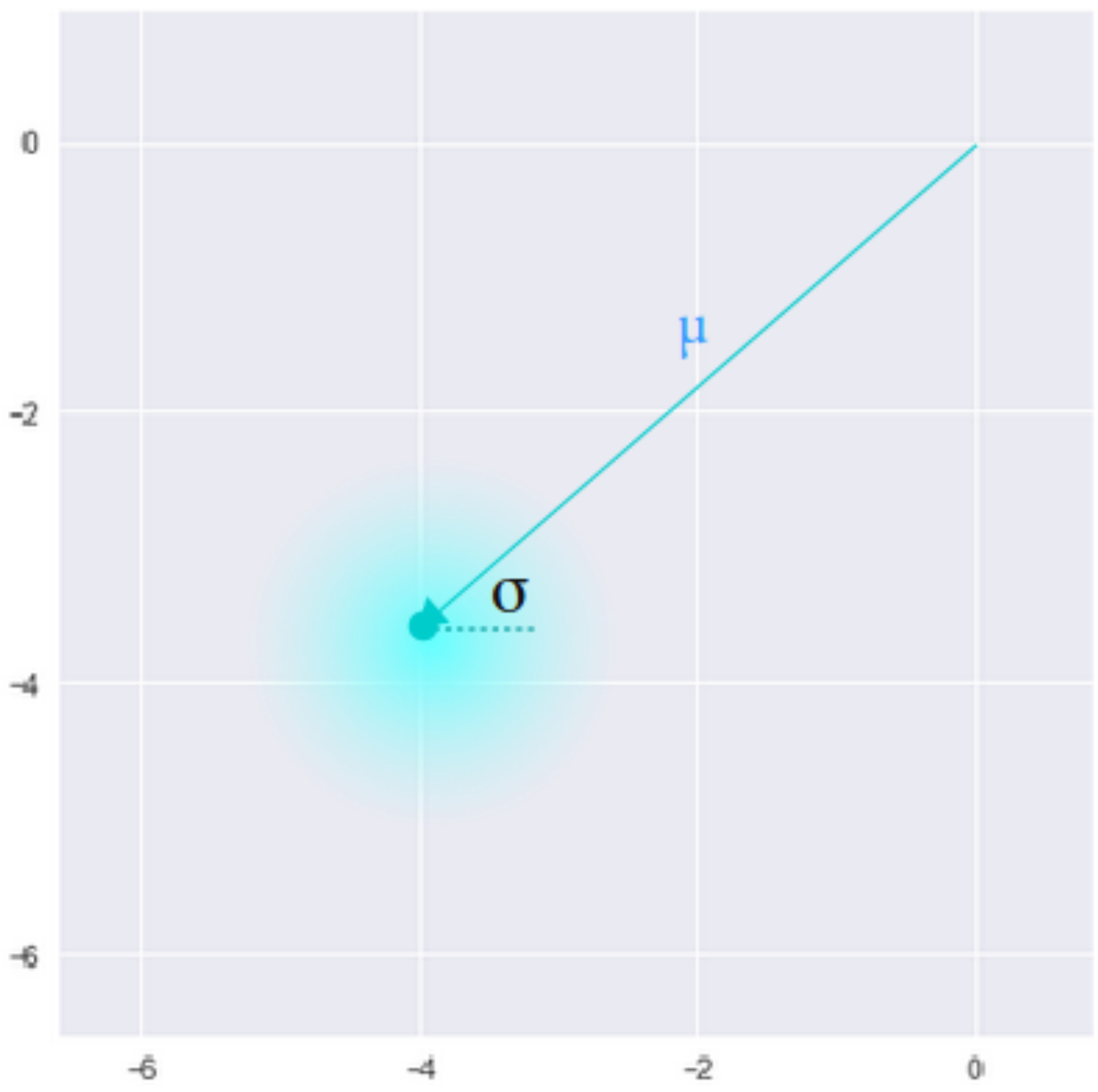
- Давайте как-то заставим сеть делать непрерывное латентное пространство





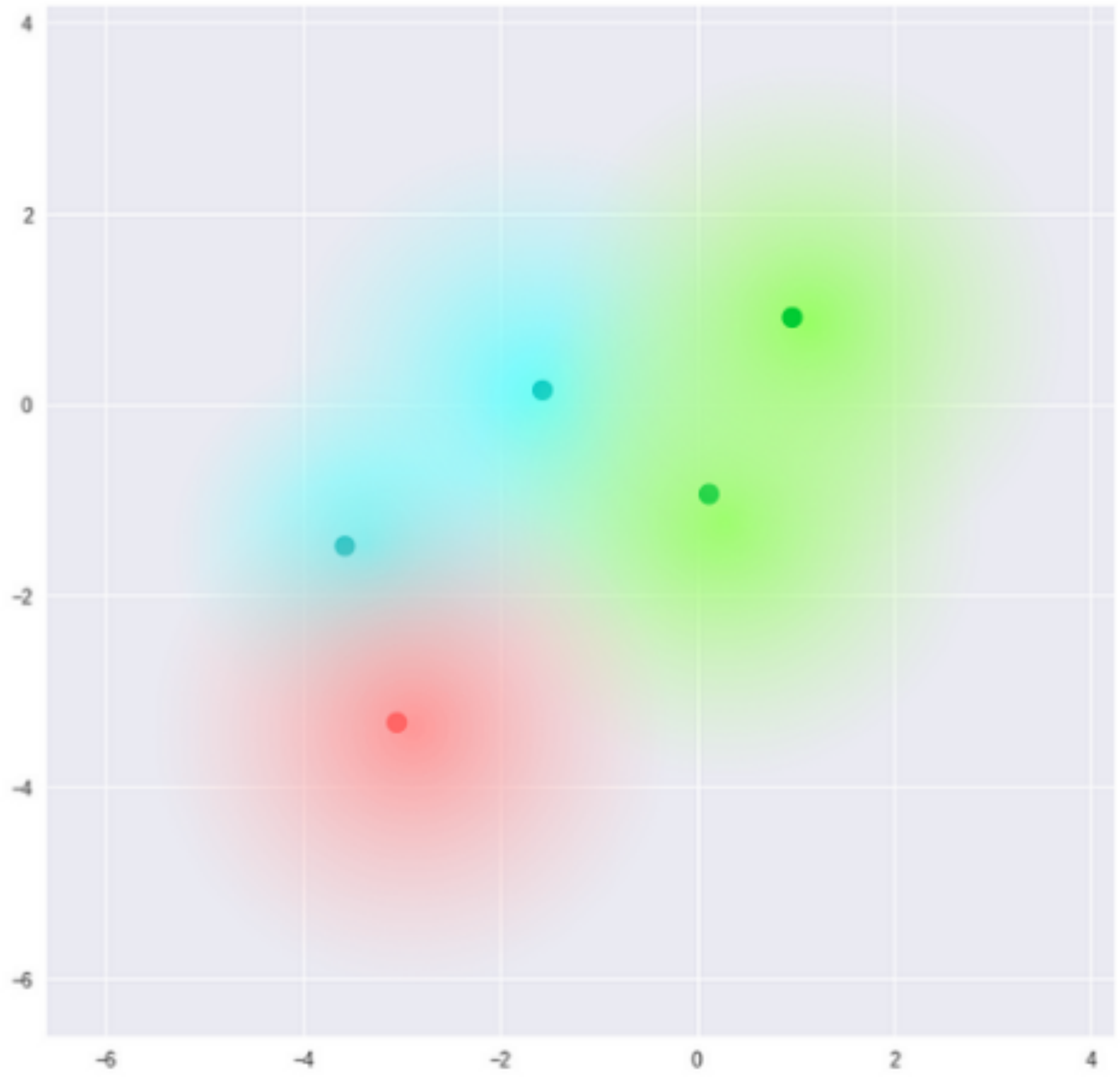


Standard Autoencoder
(direct encoding coordinates)

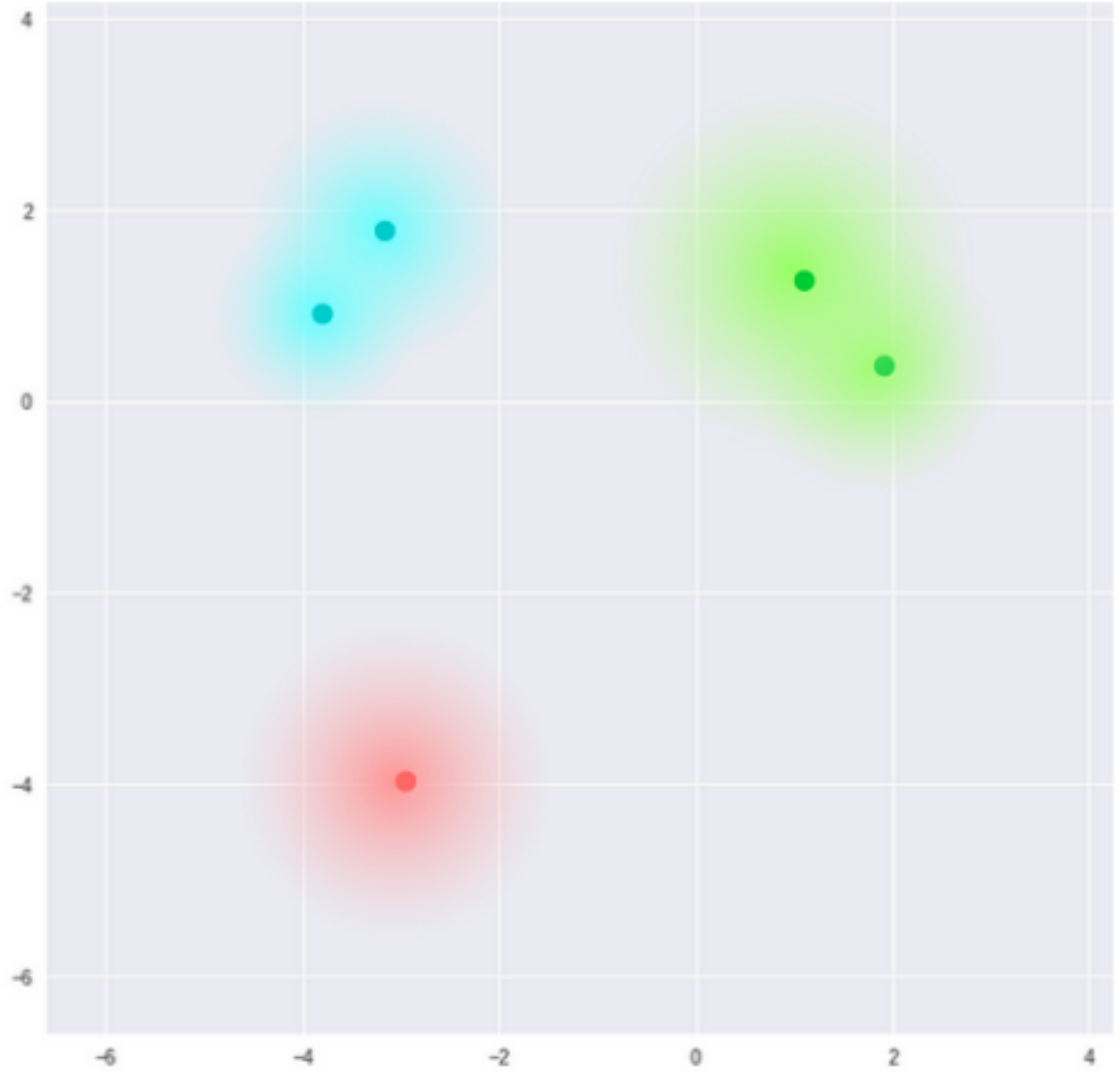


Variational Autoencoder
(μ and σ initialize a probability distribution)

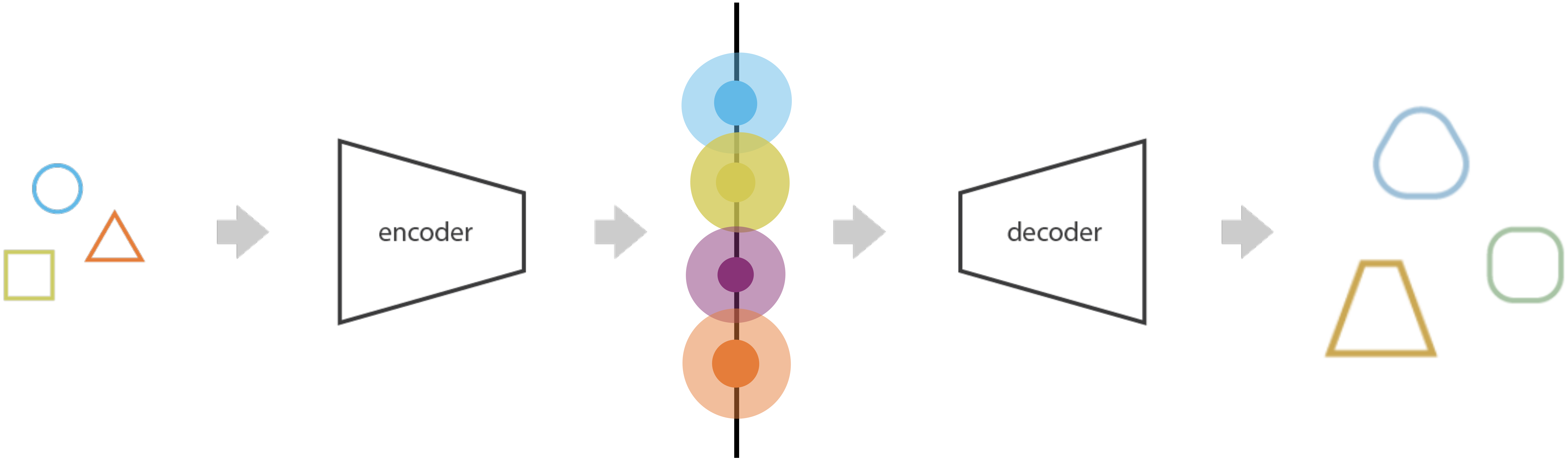
Просто добавить μ и σ is not enough



What we require



What we may inadvertently end up with

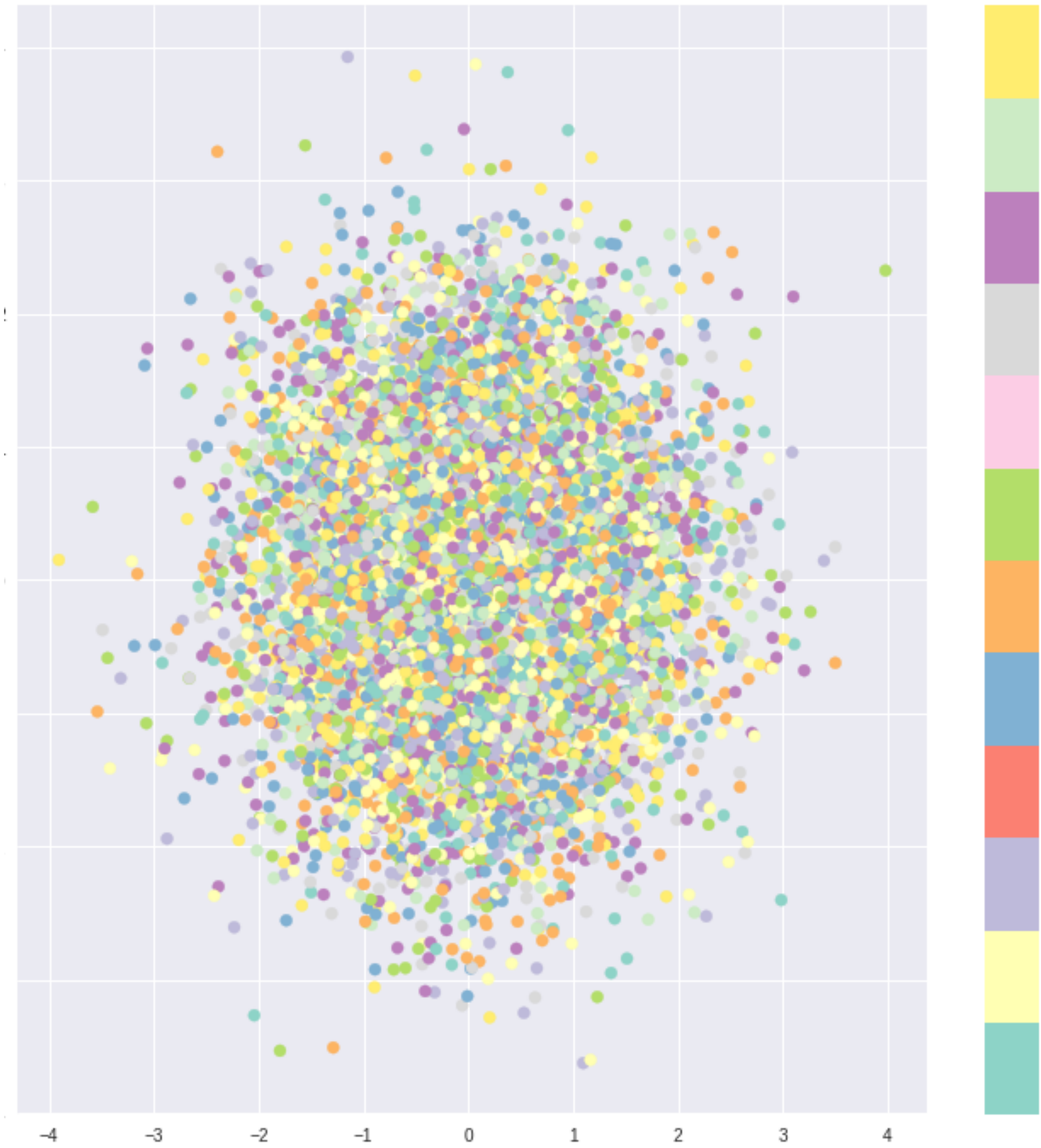

encoder
decoder

Добавим метрику схожести распределений

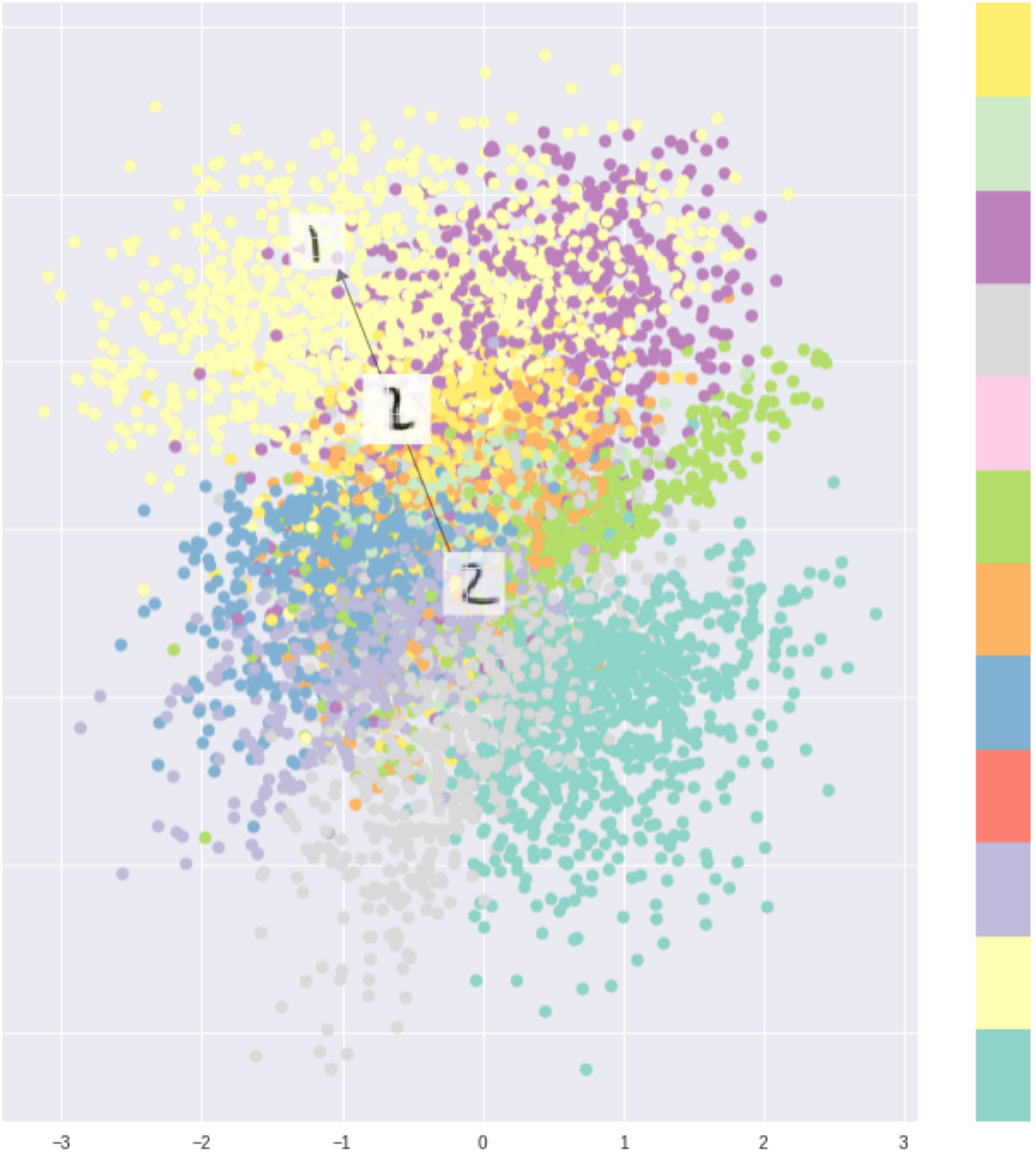
Kullback-Leibler расходимость

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

Но При использовании KL потерь области кодирования расположены случайным образом в окрестности выделенной точки в скрытом пространстве со слабым учётом сходства между образцами входных данных. Поэтому декодер не способен извлечь что-либо значащее из этого пространства

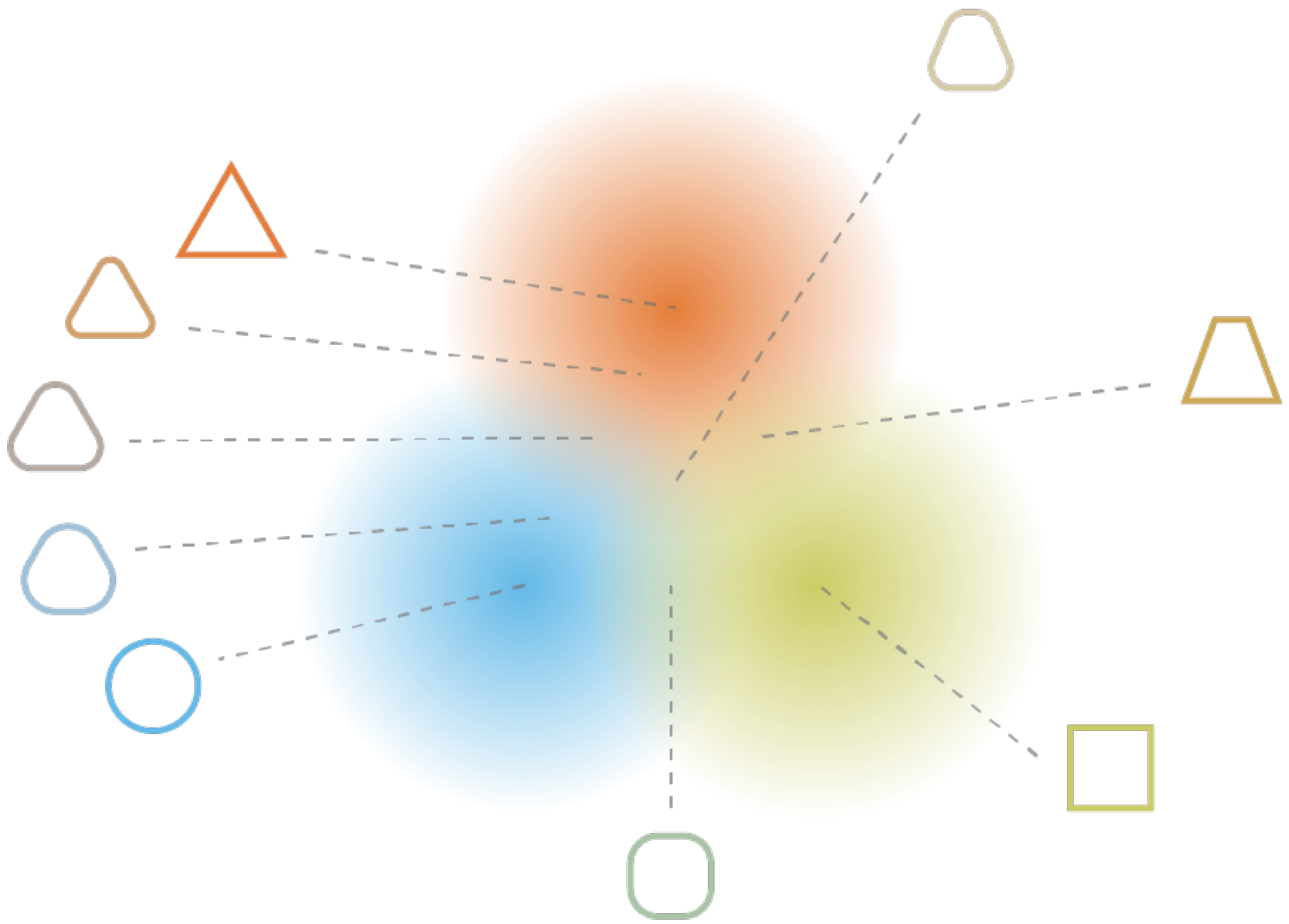


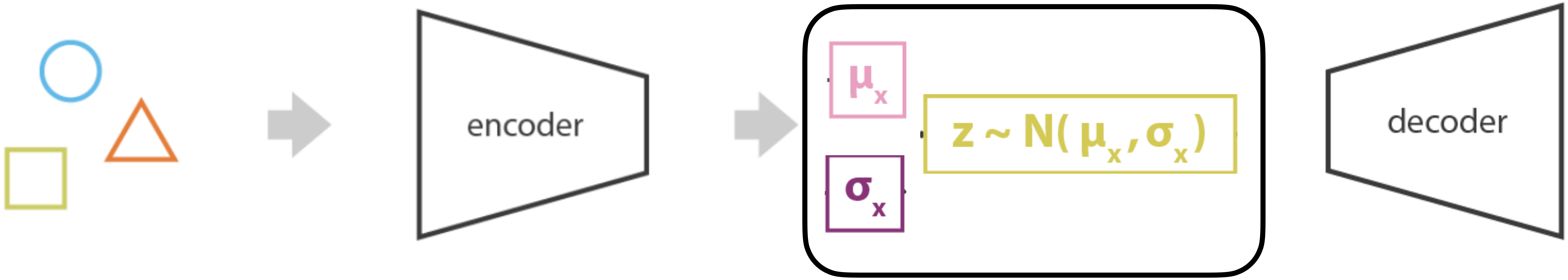
Добавим метрику потерь восстановления из пространства как у AE



Как выглядит пр-во в идеале?

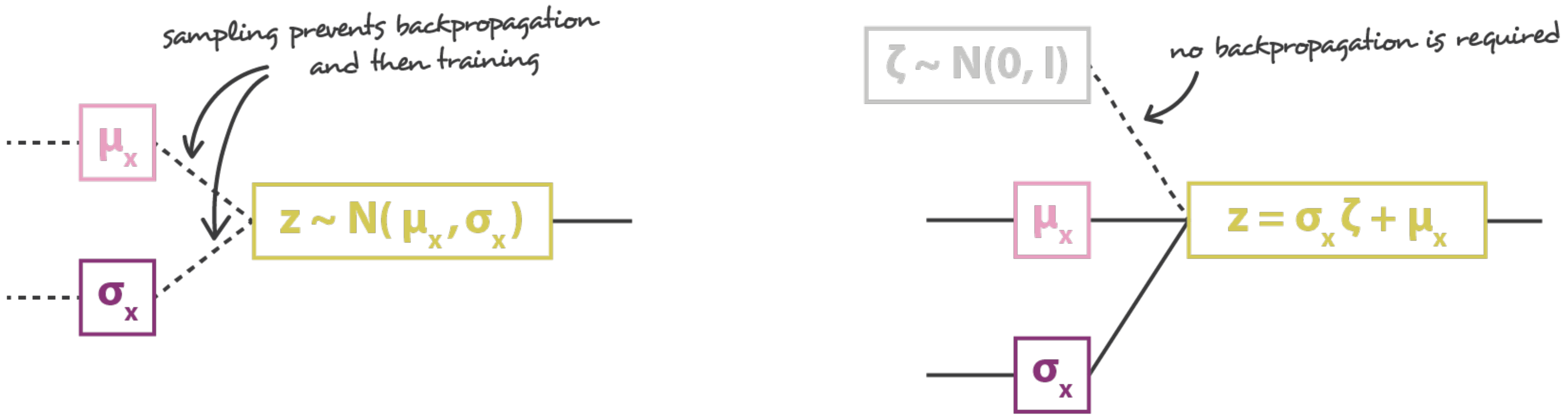
А вот так ->



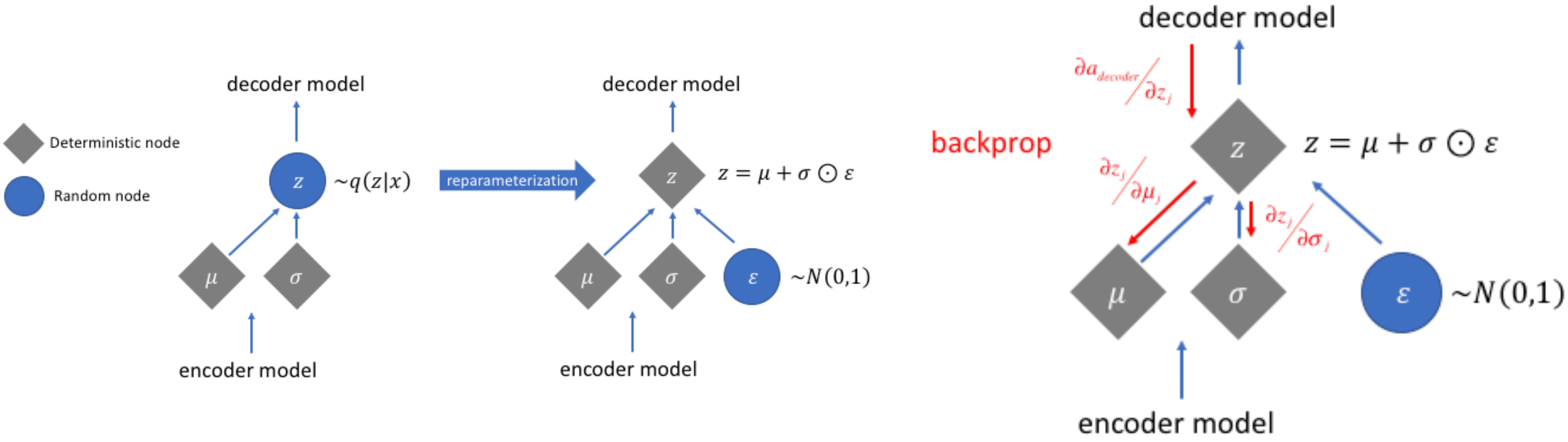

encoder
μ_x
σ_x
$z \sim N(\mu_x, \sigma_x)$
decoder

Reparametrization Trick

no problem for backpropagation — backpropagation is not possible due to sampling

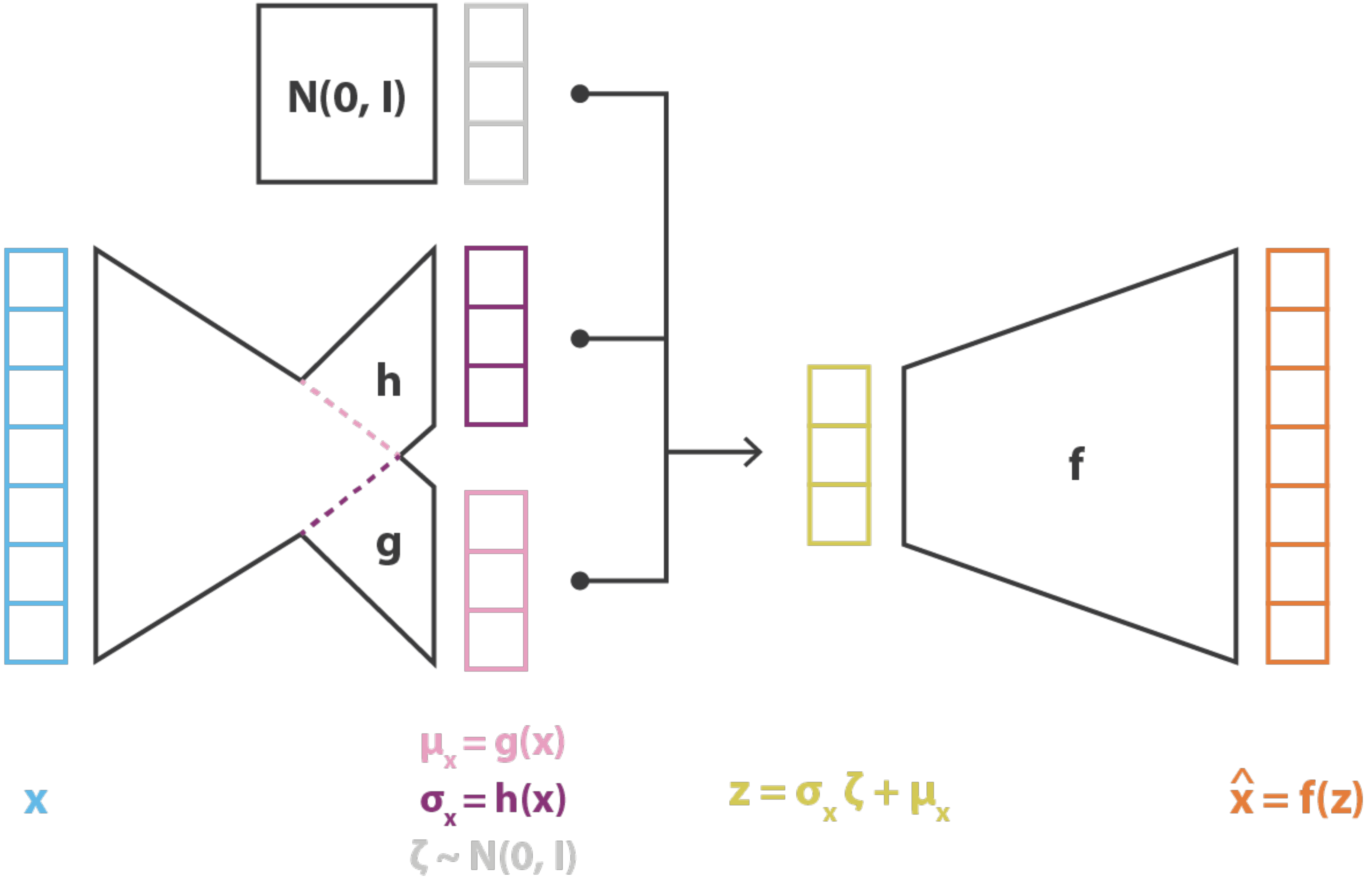


sampling without reparametrisation trick — **sampling with reparametrisation trick**



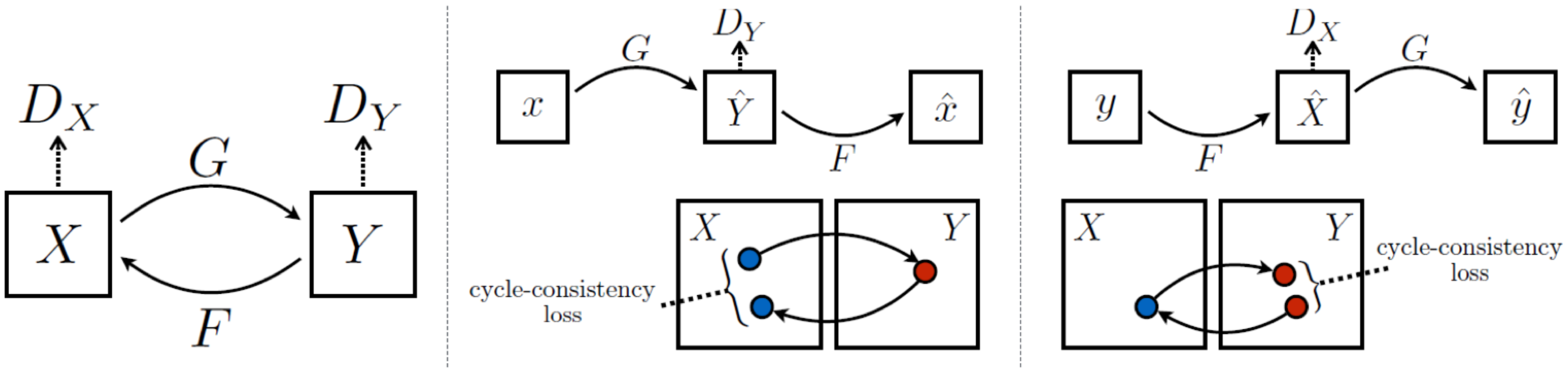
Deterministic node
Random node
decoder model
z
$\sim q(z|x)$
μ
σ
encoder model
reparameterization
decoder model
z
$z = \mu + \sigma \odot \varepsilon$
μ
σ
ε
$\sim N(0,1)$
encoder model
backprop
decoder model
$\partial a_{decoder} / \partial z_j$
z
$z = \mu + \sigma \odot \varepsilon$
$\partial z_j / \partial \mu_j$
$\partial z_j / \partial \sigma_j$
μ
σ
ε
$\sim N(0,1)$
encoder model

Итоговая схема архитектуры



$$\text{loss} = C\,||\,x - \hat{x}\,||^2 + KL[\,N(\mu_x, \sigma_x), N(0, I)\,] = C\,||\,x - f(z)\,||^2 + KL[\,N(g(x), h(x)), N(0, I)\,]$$

Cycle GAN TL;DR



- Вводится G : X -> Y и F : Y -> X и D_x с D_y
- D_y учит переводить изображения X в Y, D_x обратно
- В итоге получаем обратимые преобразования

$$\bar{x} \to G(\bar{x}) \to F(G(\bar{x})) \approx \bar{x}$$

$$\bar{y} \to F(\bar{y}) \to G(F(\bar{y})) \approx \bar{y}$$

Функции потерь

Состязательная функция потерь

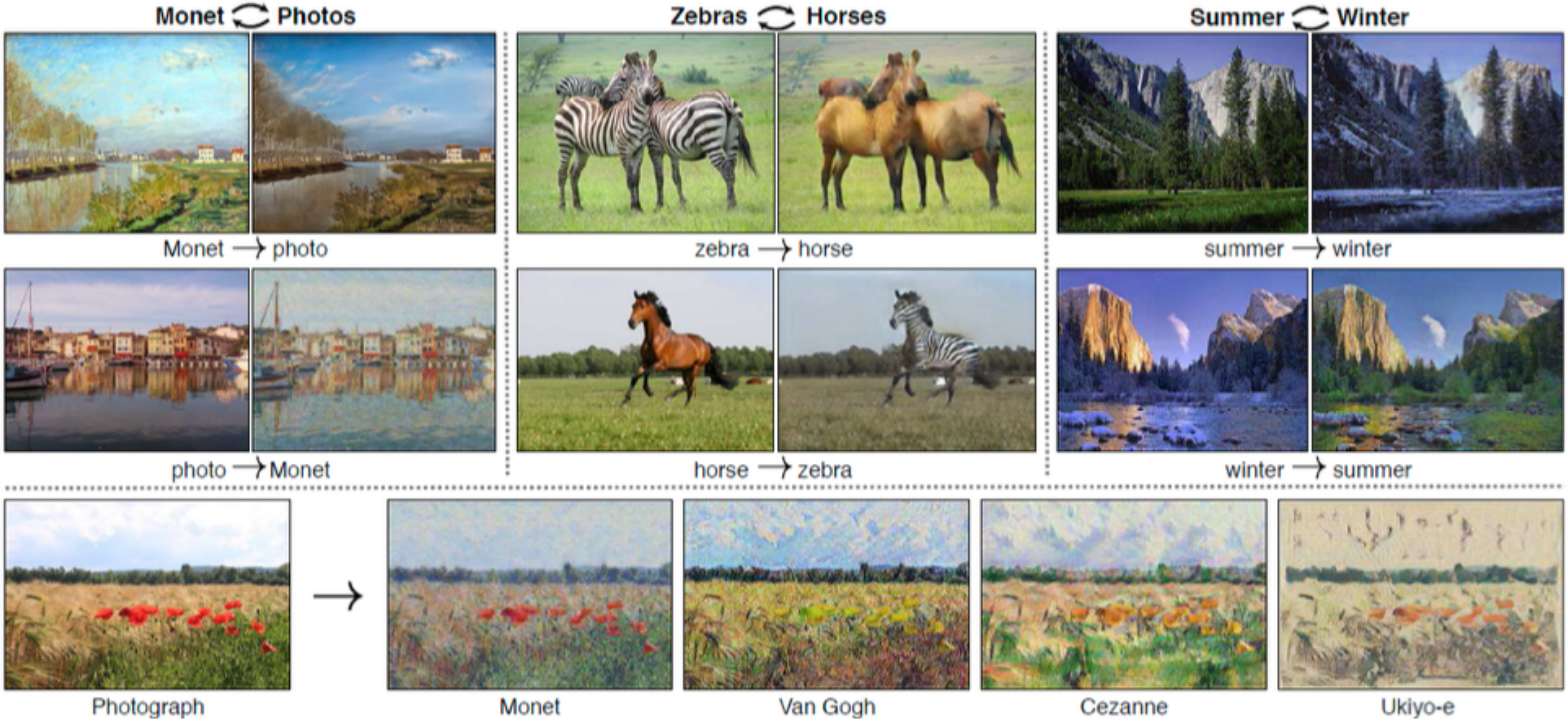
$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{\bar{y} \sim p_{data}(\bar{y})}[\log D_Y(\bar{y})] + \mathbb{E}_{\bar{x} \sim p_{data}(\bar{x})}[\log(1 - D_Y(G(\bar{x})))]$$

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{\bar{x} \sim p_{data}(\bar{x})}[\log D_X(\bar{x})] + \mathbb{E}_{\bar{y} \sim p_{data}(\bar{y})}[\log(1 - D_X(F(\bar{y})))]$$

Циклическая

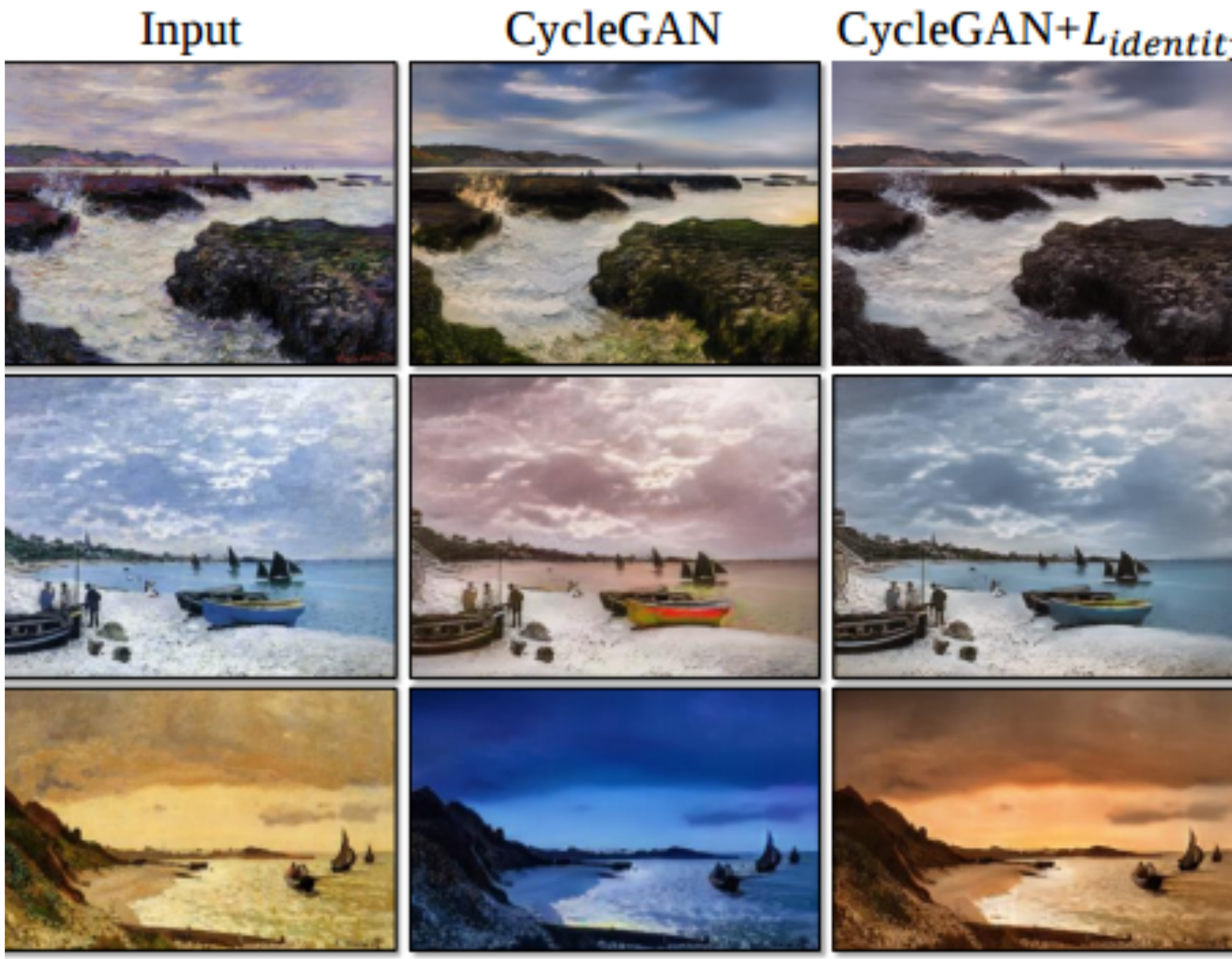
$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{\bar{x} \sim p_{data}(\bar{x})}[\|F(G(\bar{x})) - \bar{x}\|_1] + \mathbb{E}_{\bar{y} \sim p_{data}(\bar{y})}[\|G(F(\bar{y})) - \bar{y}\|_1]$$

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Спасаем цветовую палитру

$$L_{identity}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{data}(x)}[\|F(x) - x\|_1]$$



igure 9: The effect of the *identity mapping loss* o
Ionet→ Photo. From left to right: input paintings, Cycle
AN without identity mapping loss, CycleGAN with iden
ty mapping loss. The identity mapping loss helps preserv
ıe color of the input paintings.