курс «Глубокое обучение»


# Вариационный автокодировщик

Александр Дьяконов

11 мая 2020 года

# План

Генеративная модель

Variational Autoencoders (VAE)

Variational Bayesian Inference

Reparametrizaton trick

Векторная арифметика

Conditional VAE (CVAE)

Ladder Variational Autoencoders

Bidirectional-Inference Variational Autoencoder (BIVA)

Vector Quantised-Variational AutoEncoder (VQ-VAE)

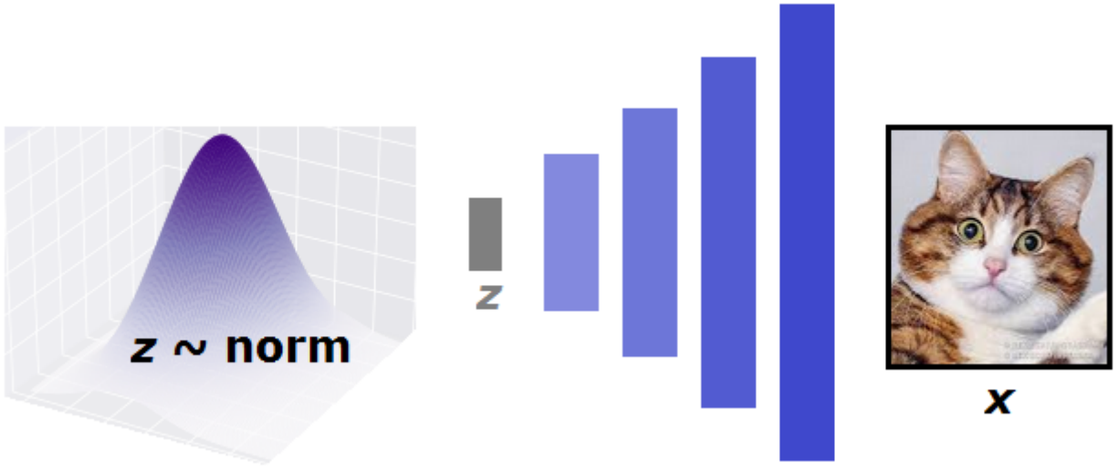
VQ-VAE-2

VAE: Image Colorization

VAE: Forecasting from Static Images

Adversarial Autoencoder

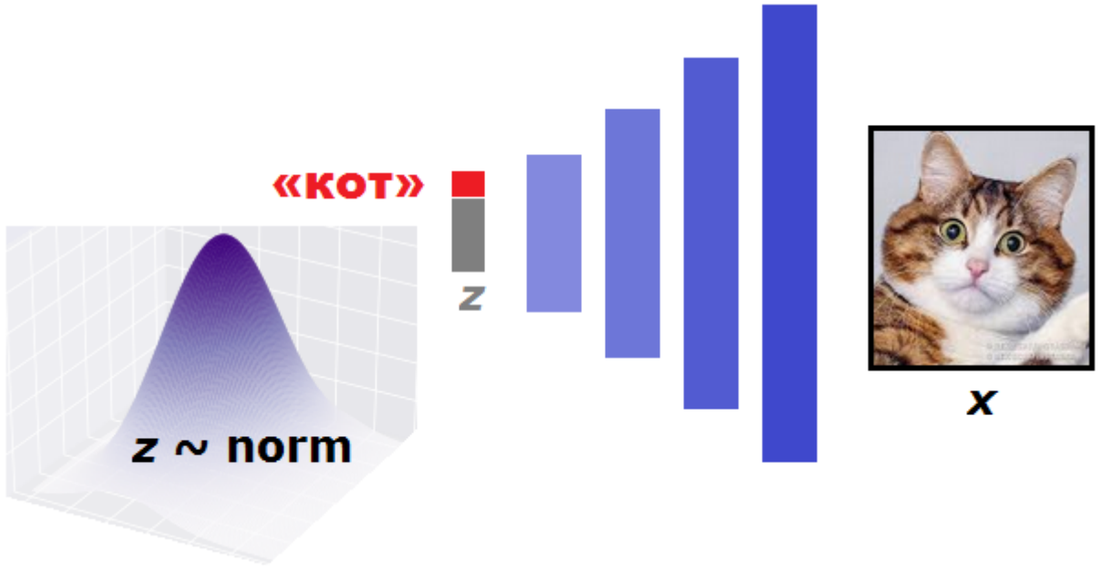
# Генеративная модель

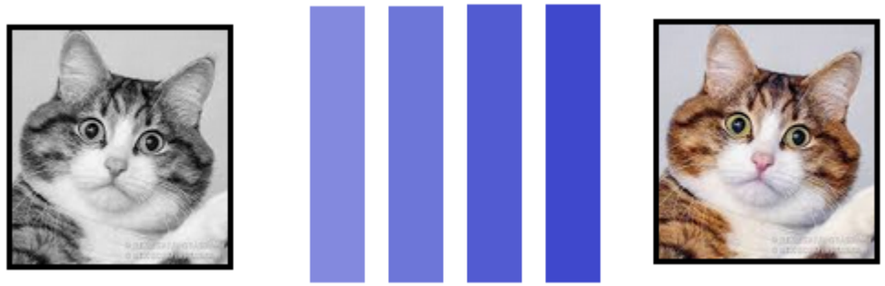



**Проблема – как сделать отображение «шум → что-то разумное»**

**дискриминативная постановка понятнее:**
**различаем несколько категорий**
**генеративная – порождение!**

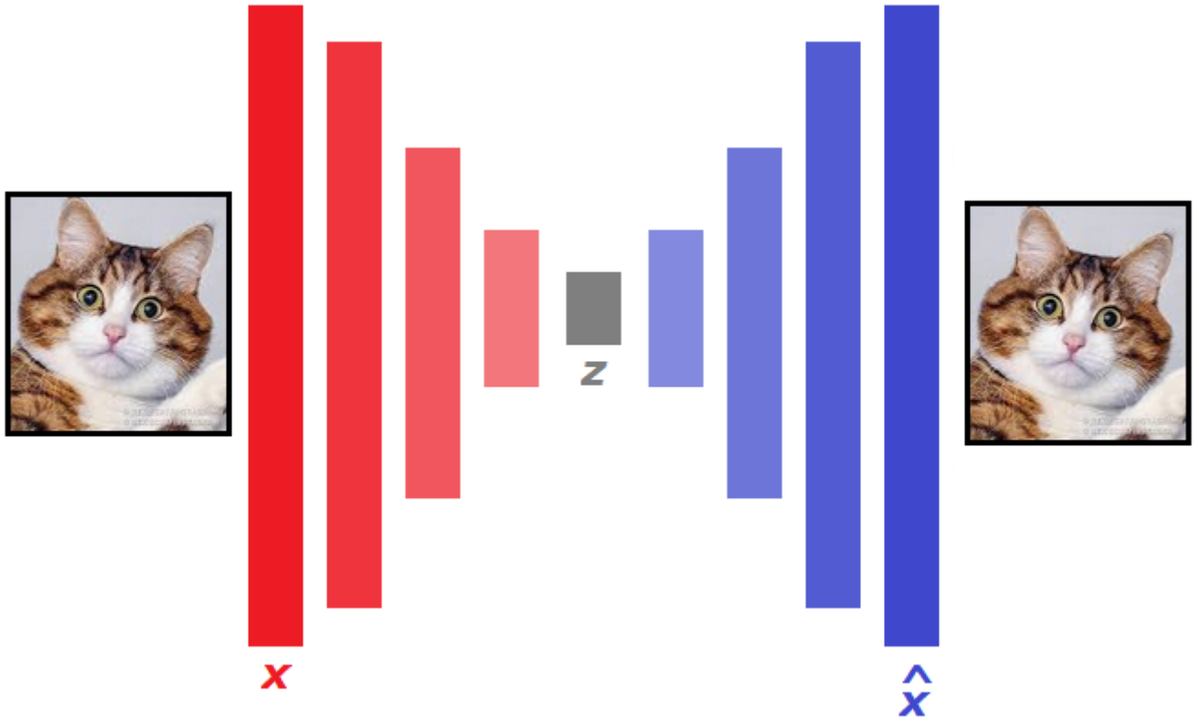
# Условная генеративная модель (Conditional Generative Model)



# Variational Autoencoders (VAE)

**Обучать автокодировщик так, чтобы скрытые переменные ~ какое-то распределение**




$$z \sim \text{norm}(0, I)$$

**можно потом сэмплировать отсюда**

**плохо, когда z кластеризуются… это запоминание данных**

# Variational Autoencoders (VAE)

**Раньше:**

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_1, \ldots, x_{i-1})$$

$$L \to \max_\theta$$

**Теперь: через скрытую переменную**

$$p_\theta(x) = \int p_\theta(z)\, p_\theta(x \mid z) \partial z$$

**нельзя напрямую оптимизировать $\Rightarrow$ оптимизируем оценку правдоподобия**

**Напрашивается**

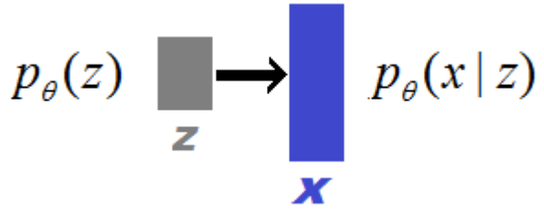
$$p_\theta(x) \approx \frac{1}{m} \sum_{t=1}^{m} p_\theta(x \mid z_t)$$

**но нужно много $\{z_t\}$, для большинства м.б. $p_\theta(x \mid z_t) \approx 0$**

**практически невычислимо… можно ли генерировать $\{z_t\}$: $p_\theta(x \mid z_t) >> 0$?**

# Variational Autoencoders (VAE)

«true prior» «true conditional»




**x – изображение, z – скрытая переменная**

$$p_\theta(x) = \int p_\theta(z)\, p_\theta(x \mid z) \partial z$$

$p_\theta(z)$ **~ пусть простое (нормальное)**

$p_\theta(x \mid z)$ **~ сложное, т.к. изображение по вектору**

**(КОДИРОВЩИК – пусть нейросеть)**

**Но невозможно вычислить** $p_\theta(x \mid z)$ **для всех z (интегрирование невозможно)**

Kingma and Welling «Auto-Encoding Variational Bayes», ICLR 2014

# Variational Autoencoders (VAE)

**Решение: кроме кодировщика сделать декодировщик $q_\varphi(z \mid x)$**

**это позволит получить оценку на правдоподобие**

**(заодно полезно для других задач как генератор признаков)**

**encoder (recognition/inference) network**

**decoder (generation) network**

# Variational Bayesian Inference

**хотим максимизировать:**

$$\log p_\theta(x) = \int q_\varphi(z \mid x) \log p_\theta(x) \partial z =$$

$$= \int q_\varphi(z \mid x) \log \frac{p_\theta(x, z)}{p_\theta(z \mid x)} \partial z =$$

$$= \int q_\varphi(z \mid x) \log \frac{p_\theta(x, z) q_\varphi(z \mid x)}{q_\varphi(z \mid x) p_\theta(z \mid x)} \partial z =$$

$$= \int q_\varphi(z \mid x) \log \left( p_\theta(x \mid z) \frac{p_\theta(z)}{q_\varphi(z \mid x)} \frac{q_\varphi(z \mid x)}{p_\theta(z \mid x)} \right) \partial z =$$

# Variational Bayesian Inference

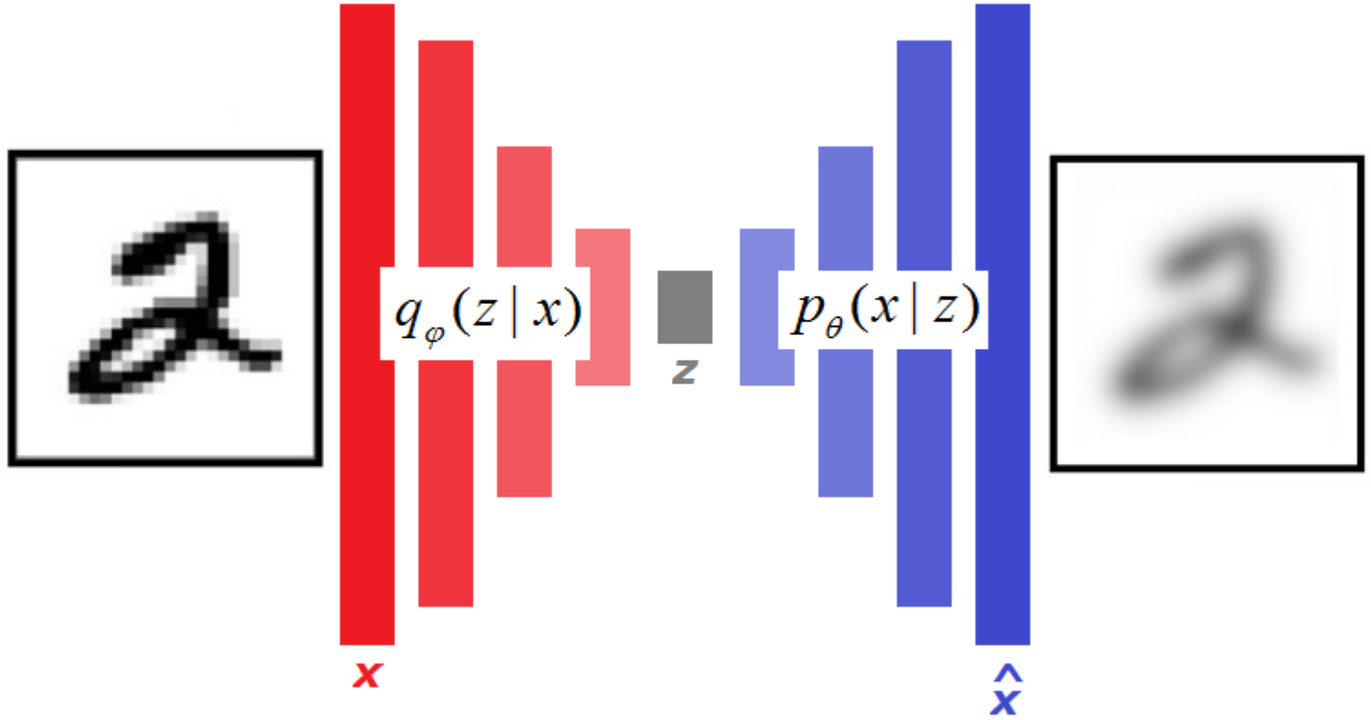
| | | |
|---|---|---|
| $= \int q_\varphi(z \mid x) \log p_\theta(x \mid z) \partial z -$ | $\sim \mathbf{E}_{z \sim q} \log p_\theta(x \mid z)$ | **с помощью сэмплирования и кодировщика – это реконструкция данных (Neural Decoder Reconstruction Loss)** |
| $-\int q_\varphi(z \mid x) \log \frac{q_\varphi(z \mid x)}{p_\theta(z)} \partial z +$ | $\sim D_{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z))$ | **KL между гауссианой для декодировщика и априорным (можно посчитать) – это близость априорного и апостериорного распределений (регуляризация латентного представления)** |
| $+\int q_\varphi(z \mid x) \log \frac{q_\varphi(z \mid x)}{p_\theta(z \mid x)} \partial z$ | $\sim D_{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z \mid x))$ | **уже говорили о невозможности вычисления, но $D_{KL} \geq 0$ отбросим это слагаемое** |

$$\log p_\theta(x) \geq \mathbf{E}_{z \sim q} \log p_\theta(x \mid z) + D_{KL}(q_\varphi(z \mid x) \mid\mid p_\theta(z)) \to \max$$

**аппроксимация истинного распределения вариационным**

# Variational Bayesian Inference

**кодировщик - декодировщик**




**Возьмём в качестве** $q_\varphi(z \mid x)$ **(а это моделирует DNN)**

**нормальное распределение** $\mathrm{norm}(z \mid \mu_z(x), \sigma_z(x))$

**тут reparametrization trick – дальше**

# Variational Autoencoders (VAE)

$$\log p_\theta(x) \geq \mathbf{E}_{z\sim q} \log p_\theta(x \mid z) + D_{KL}(q_\varphi(z \mid x) \,\|\, p_\theta(z)) \to \max$$

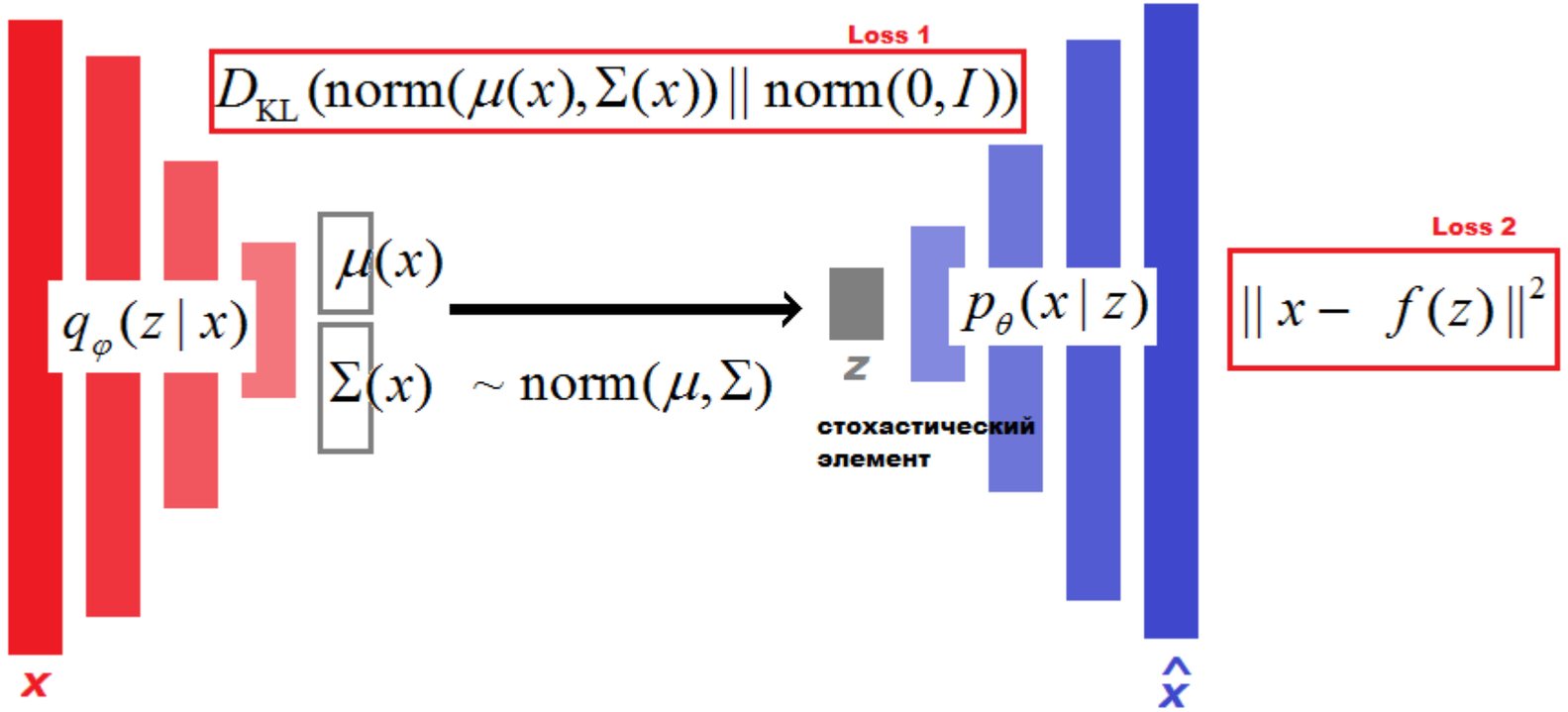
**Если считать распределение нормальным:**

$$\mathbf{E}_{z\sim q} \log p_\theta(x \mid z) \propto \| x - \quad f(z) \quad \|^2$$

выход сети

**Можно эффективно посчитать (closed form solution):**

$$D_{KL}(q_\varphi(z \mid x) \,\|\, p_\theta(z))$$

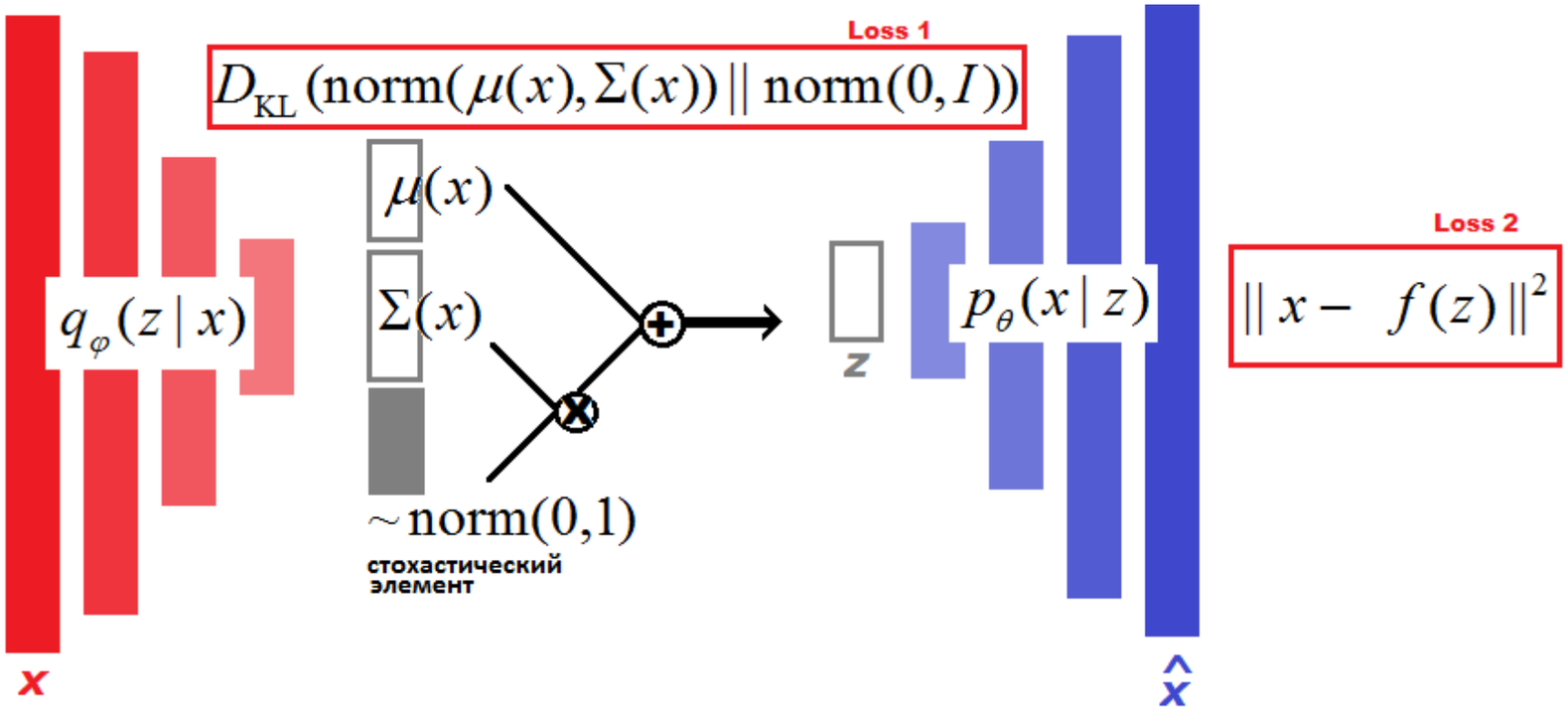



**Но как тут пропускать градиент? у нас полностью стохастический элемент!**

**кстати, ошибка = ошибка восстановления + регуляризация**

# Reparametrizaton trick: как делать BP через распределение?

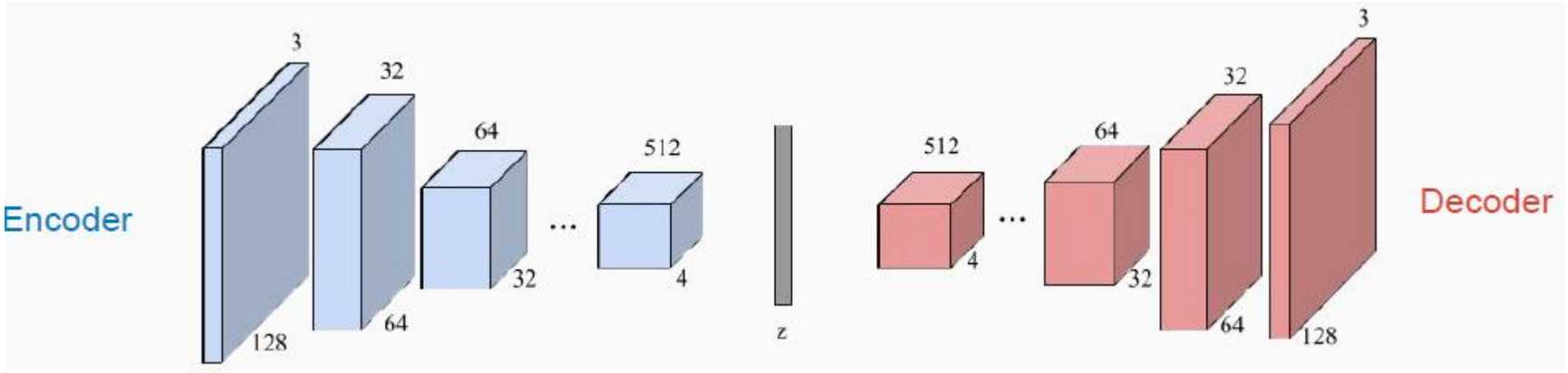
**DNN-кодировщик выдаёт вектор средних и матрицу ковариации (этого достаточно для моделирования распределения), будем сэмплировать ε!**




$$\text{norm}(\mu,\Sigma) \sim \mu + \Sigma \cdot \text{norm}(0,1)$$

**градиент будет свободно распространяться через нестохастические элементы**

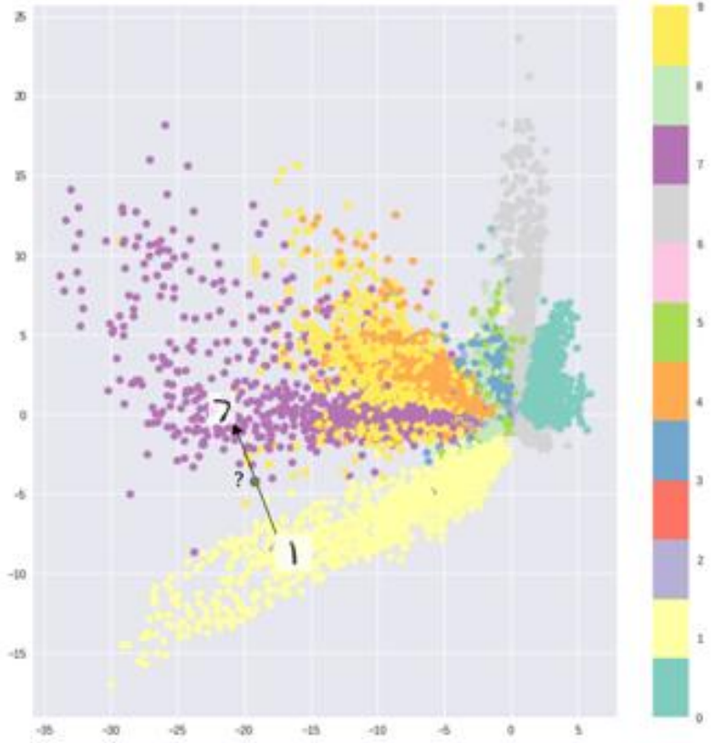
# Common VAE architecture



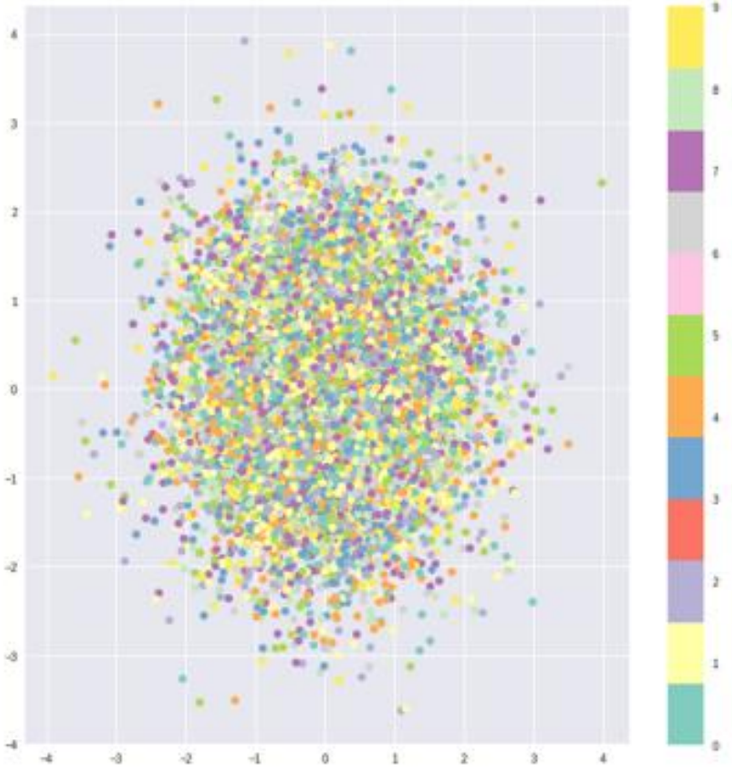

**похожа на DCGAN…**

# Variational Autoencoders (VAE)

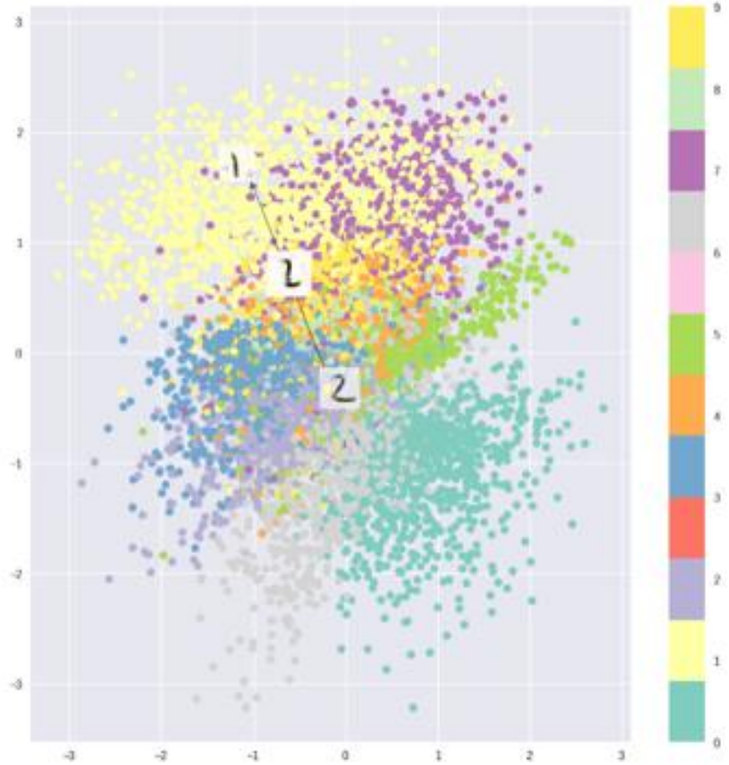
Only reconstruction loss



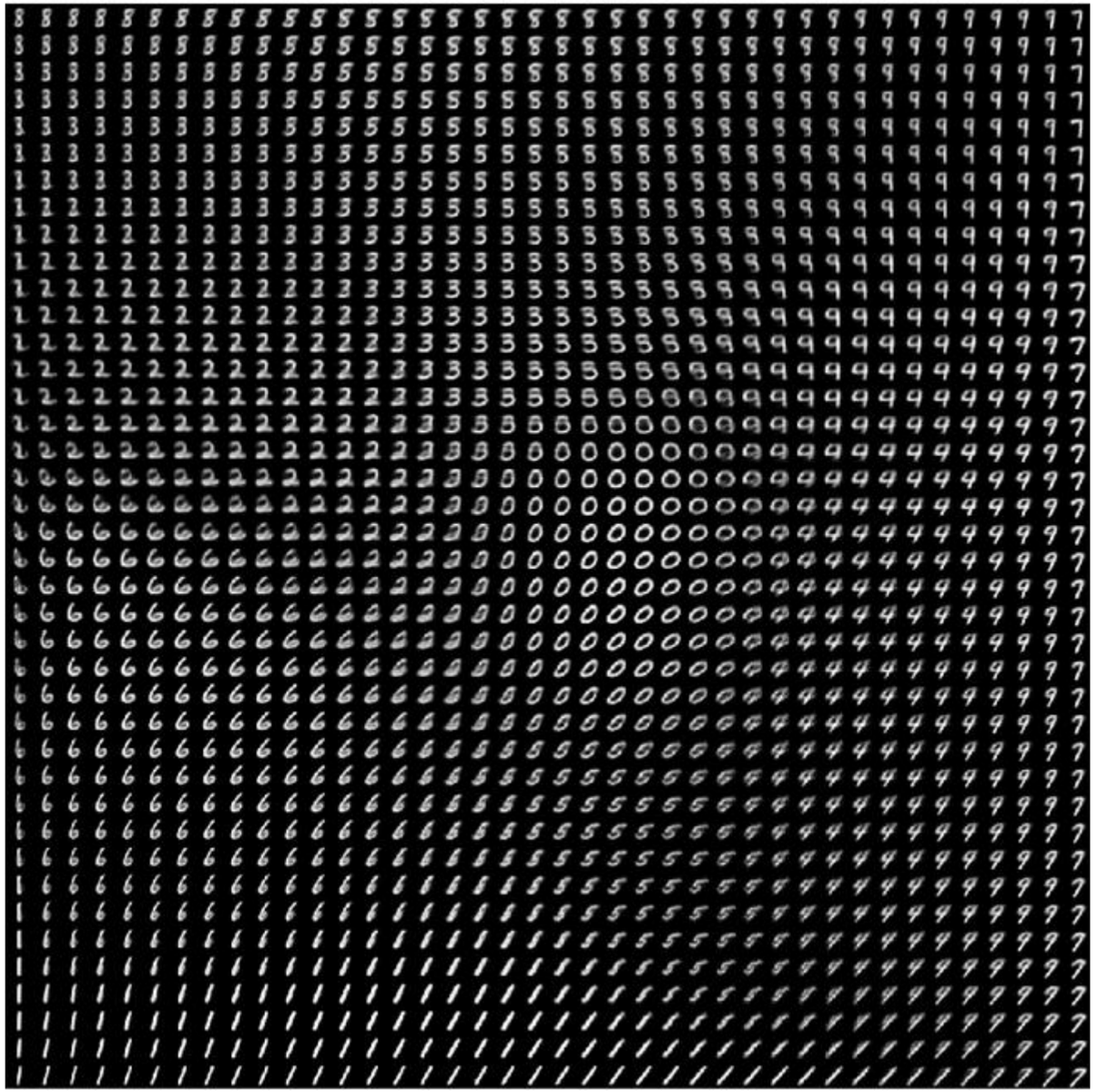
Only KL divergence



Combination

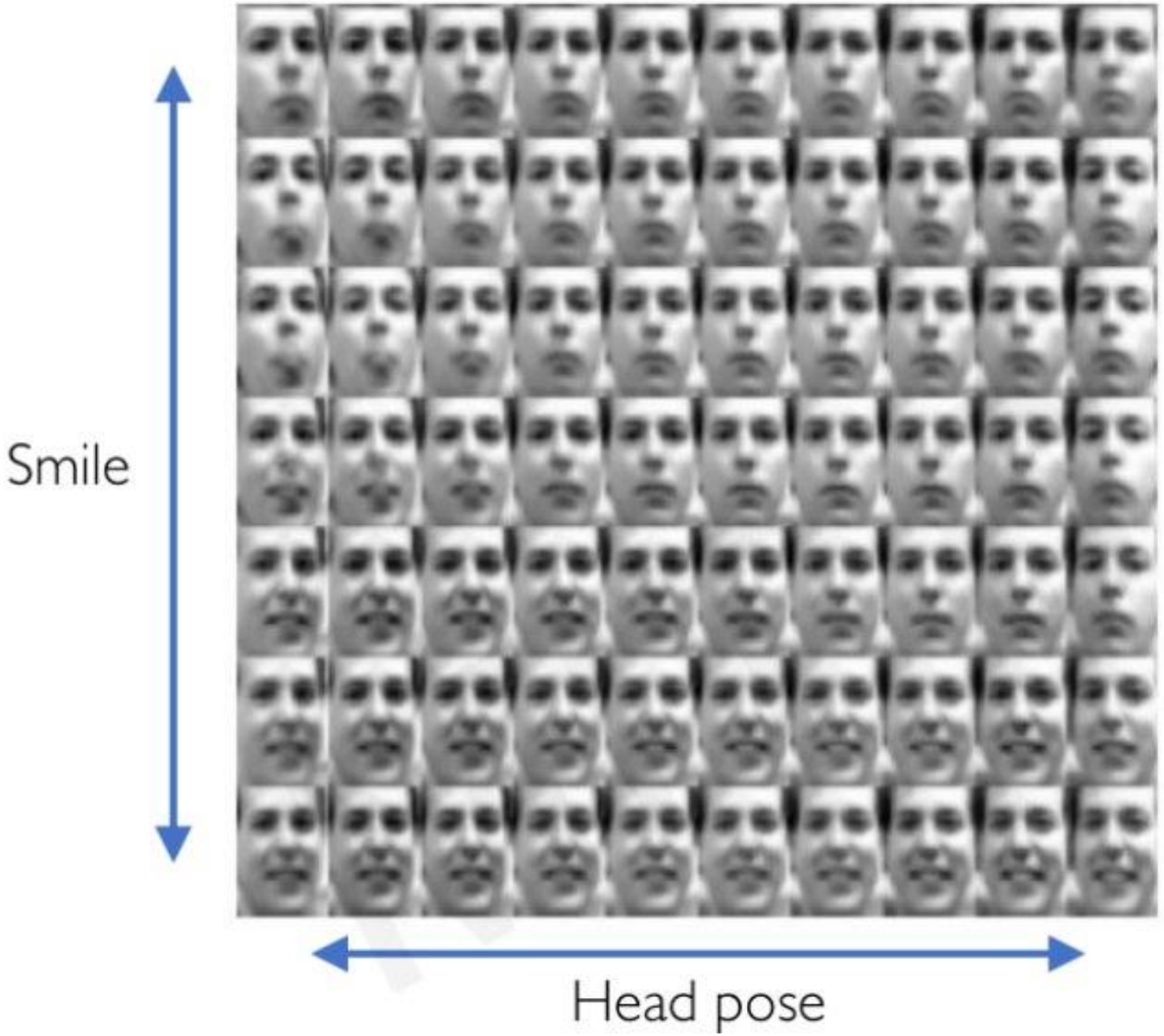


https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf



вариация двух координат z

# Могут быть интерпретируемые латентные переменные



# Функция ошибки в автокодировщиках – Feature Conceptual Loss

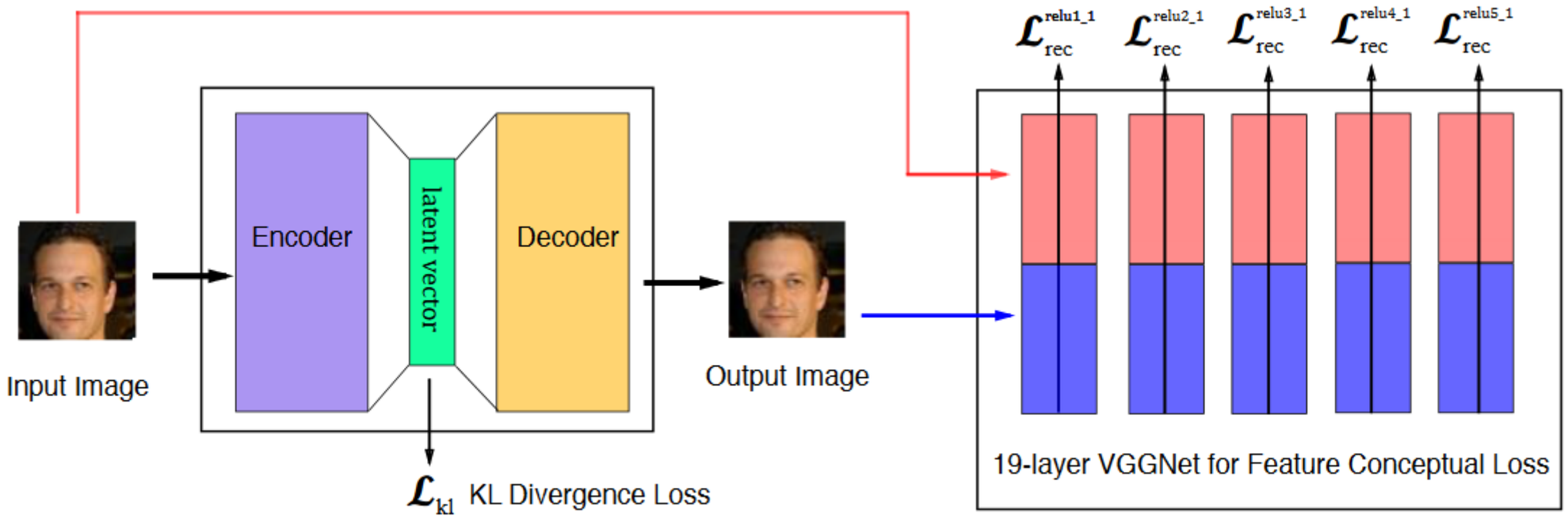



Figure 1. Model overview. The left is a deep CNN-based Variational Autoencoder and the right is a pretrained deep CNN used to compute feature perceptual loss.

**функция ошибки может быть хитрой, например, насколько похожи признаки изображений (вычисляются с помощью предобученной НС)**

Xianxu Hou, Linlin Shen, Ke Sun, Guoping Qiu «Deep Feature Consistent Variational Autoencoder» https://arxiv.org/abs/1610.00291

# Векторная арифметика

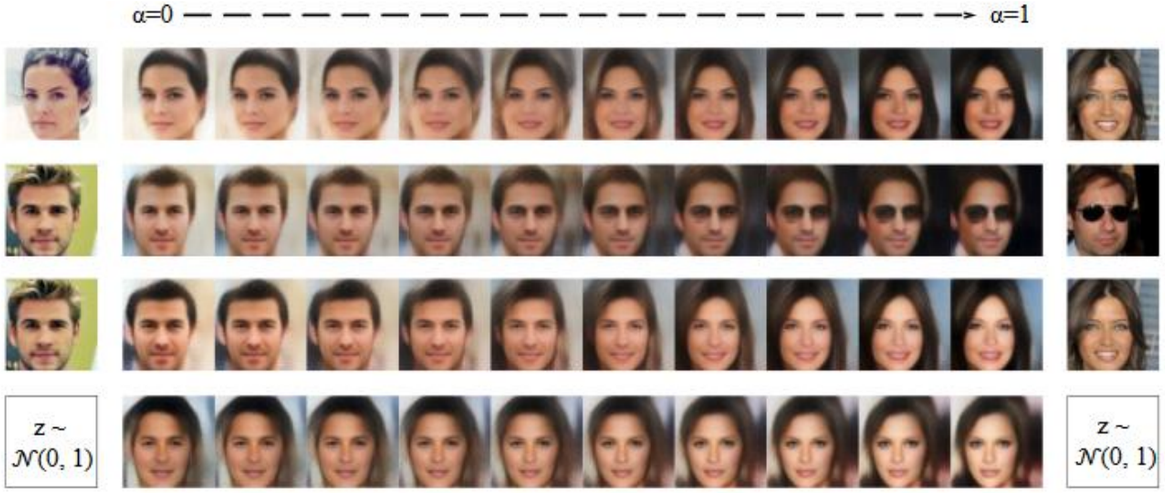



Figure 5. Linear interpolation for latent vector. Each row is the interpolation from left latent vector $z_{left}$ to right latent vector $z_{right}$. e.g. $(1-\alpha)z_{left} + \alpha z_{right}$. The first row is the transition from a non-smiling woman to a smiling woman, the second row is the transition from a man without eyeglass to a man with eyeglass, the third row is the transition from a man to a woman, and the last row is the transition between two fake faces decoded from $z \sim \mathcal{N}(0, 1)$.

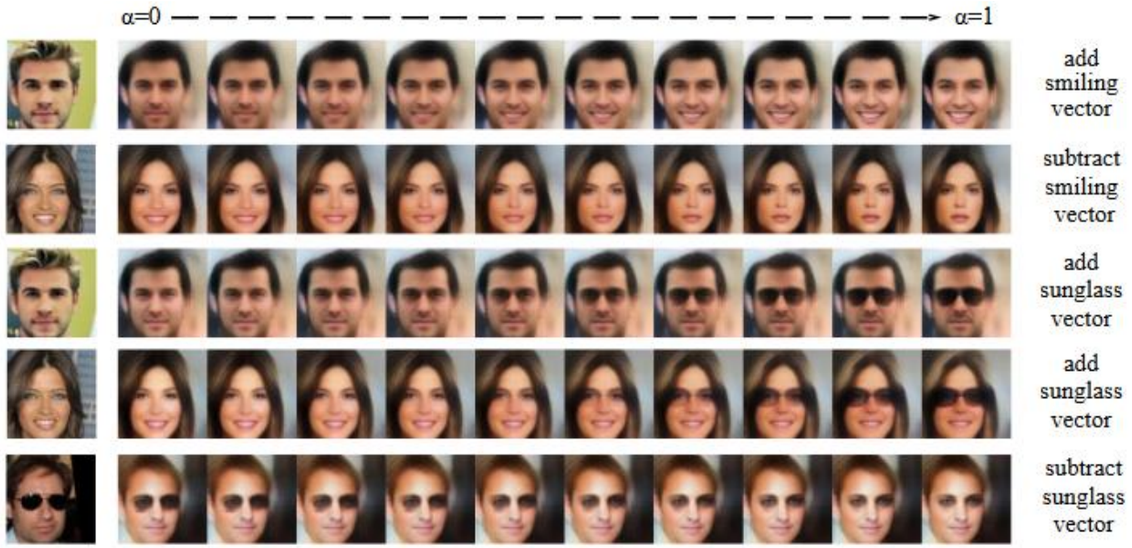



Figure 6. Vector arithmetic for visual attributes. Each row is the generated faces from latent vector $z_{left}$ by adding or subtracting an attribute-specific vector, i.e., $z_{left} + \alpha\ z_{smiling}$, where $\alpha = 0, 0.1, \ldots, 1$. The first row is the transition by adding a smiling vector with a linear factor $\alpha$ from left to right, the second row is the transition by subtracting a smiling vector, the third and fourth row are the results by adding a eyeglass vector to the latent representation for a man and women, and the last row shows results by subtracting an eyeglass vector.

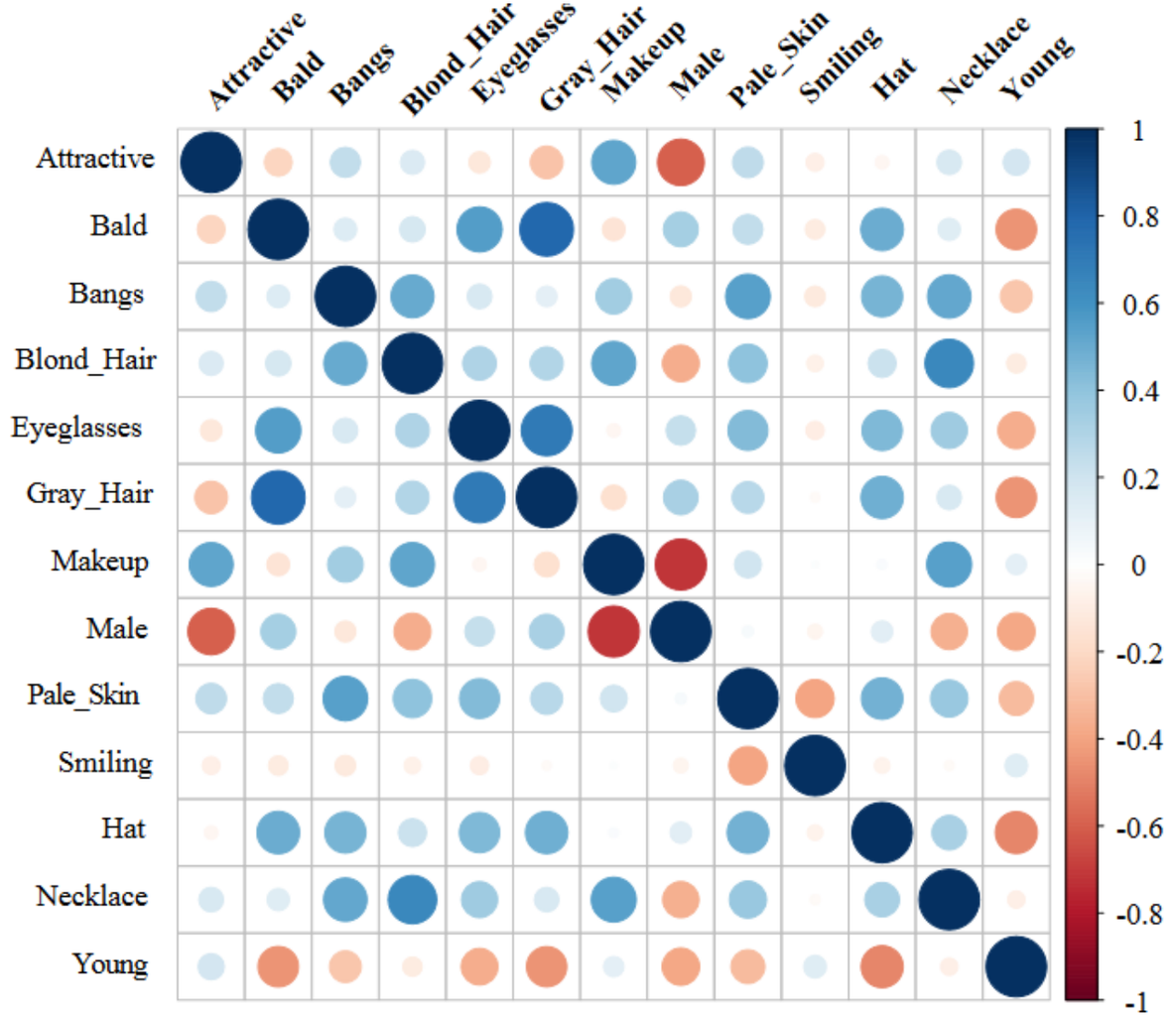
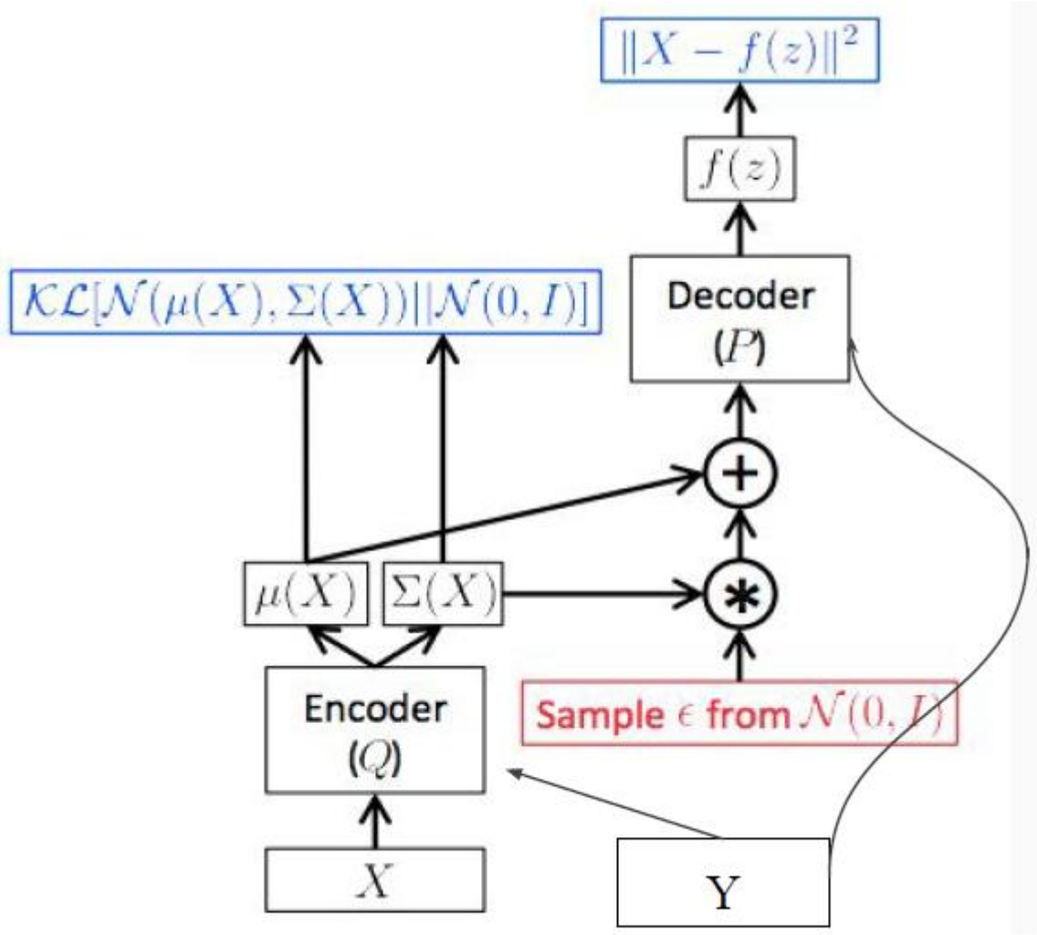



Figure 7. Diagram for the correlation between selected facial attribute-specific vectors. The blue indicates positive correlation, while red represents negative correlation, and the color shade and size of the circle represent the strength of the correlation.

# Conditional VAE (CVAE)




**Если есть какие-нибудь дополнительные метки…**

$$p_\theta(x \mid z) \to p_\theta(x \mid z, y)$$

Xinchen Yan, Jimei Yang, Kihyuk Sohn, Honglak Lee «Attribute2Image: Conditional Image Generation from Visual Attributes» https://arxiv.org/abs/1512.00570

# Conditional VAE (CVAE)

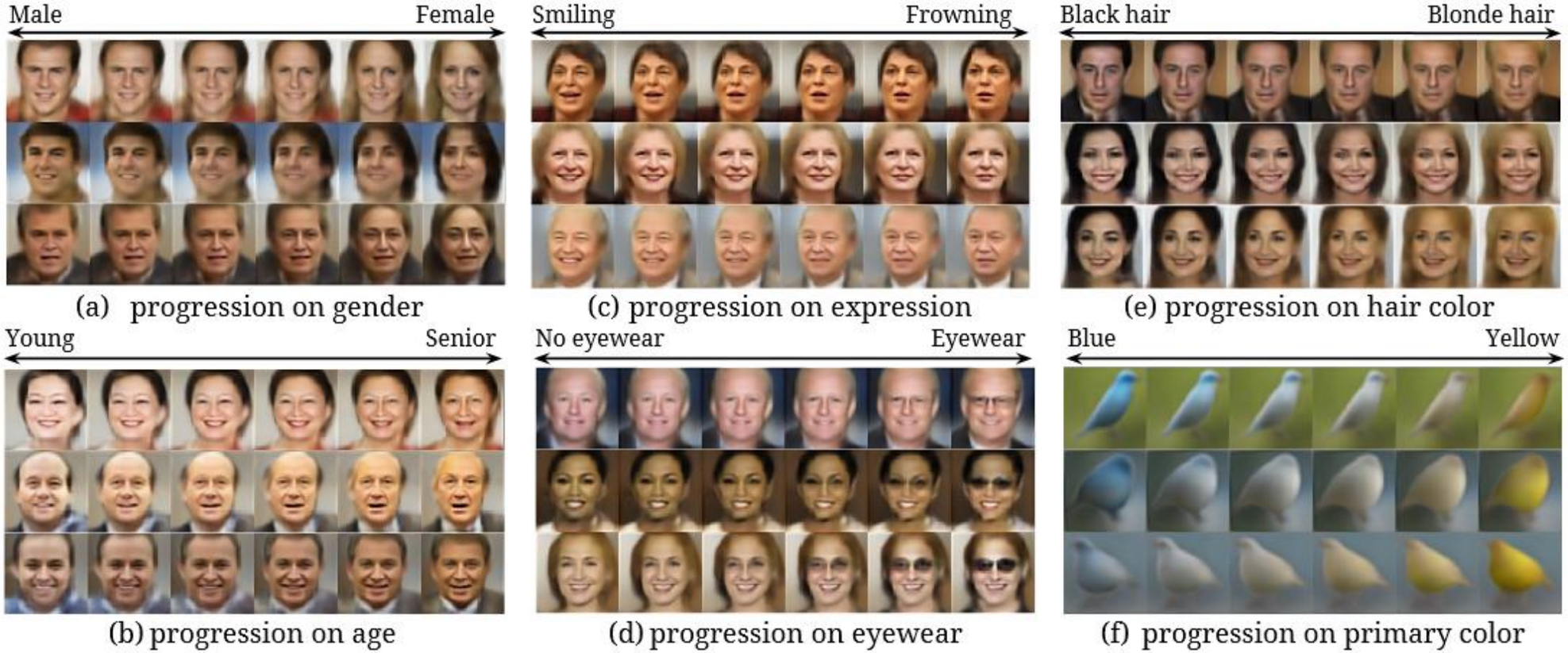



**Fig. 4.** Attribute-conditioned image progression. The visualization is organized into six attribute groups (e.g., "gender", "age", "facial expression", "eyewear", "hair color" and "primary color (blue vs. yellow)"). Within each group, the images are generated from $p_\theta(x|y,z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_\alpha, y_{rest}]$, where $y_\alpha = (1-\alpha)\cdot y_{min} + \alpha \cdot y_{max}$. Here, $y_{min}$ and $y_{max}$ stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.

# Ladder Variational Autoencoders (вариационный автокодировщик «стремянка»)

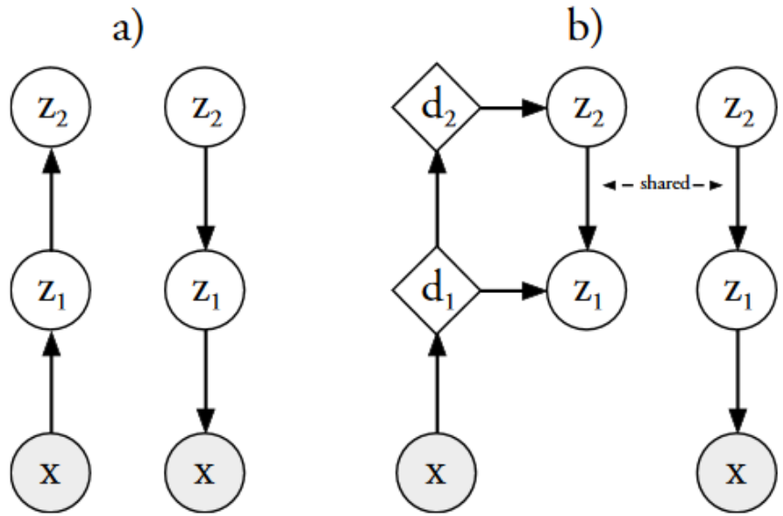



Figure 1: Inference (or encoder/recognition) and generative (or decoder) models for a) VAE and b) LVAE. Circles are stochastic variables and diamonds are deterministic variables.

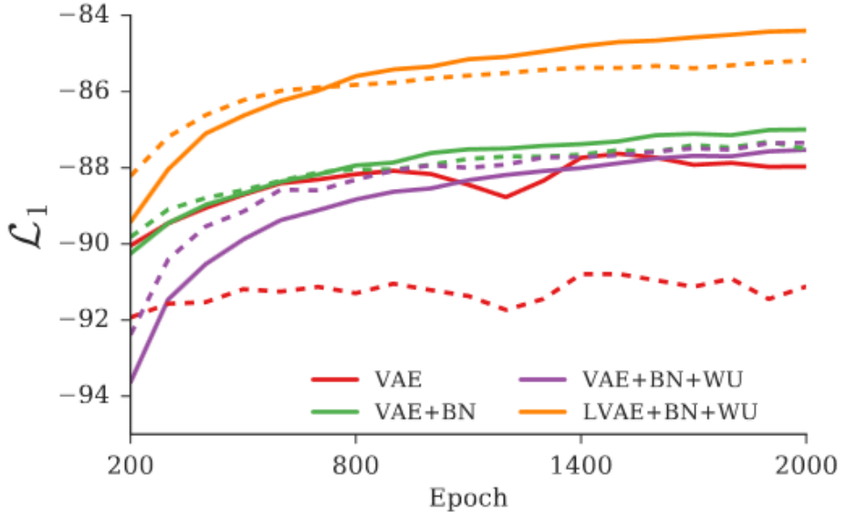



Figure 2: MNIST train (*full lines*) and test (*dashed lines*) set log-likelihood using one importance sample during training. The LVAE improves performance significantly over the regular VAE.

**латентные переменные разбиваются на слои:**

$$p_\theta(\mathbf{z}) = p_\theta(\mathbf{z}_L) \prod_{i=1}^{L-1} p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) \quad (1)$$

$$p_\theta(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}\left(\mathbf{z}_i|\mu_{p,i}(\mathbf{z}_{i+1}), \sigma^2_{p,i}(\mathbf{z}_{i+1})\right), \quad p_\theta(\mathbf{z}_L) = \mathcal{N}\left(\mathbf{z}_L|\mathbf{0}, \mathbf{I}\right) \quad (2)$$

$$p_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{N}\left(\mathbf{x}|\mu_{p,0}(\mathbf{z}_1), \sigma^2_{p,0}(\mathbf{z}_1)\right) \text{ or } P_\theta(\mathbf{x}|\mathbf{z}_1) = \mathcal{B}\left(\mathbf{x}|\mu_{p,0}(\mathbf{z}_1)\right) \quad (3)$$

# Ladder Variational Autoencoders (вариационный автокодировщик «стремянка»)

**warm-up (WU) – регуляризатор с коэффициентом, который растёт от 0 до 1**

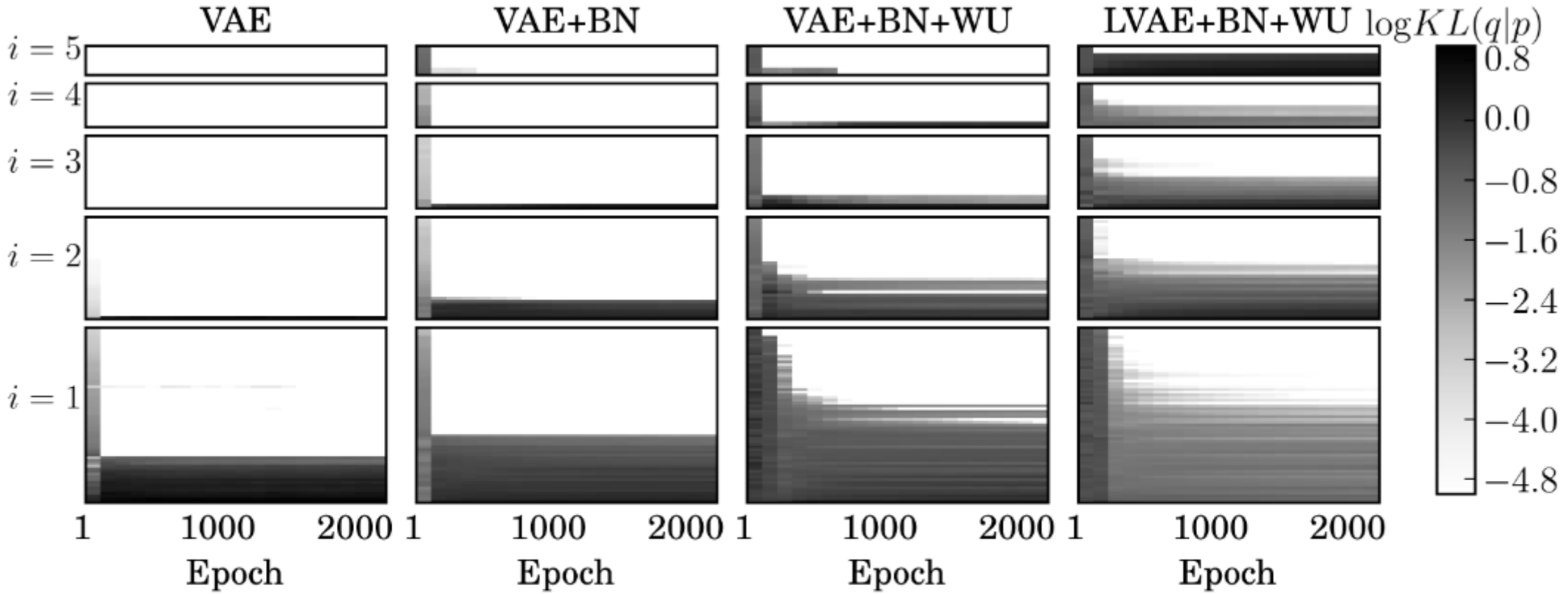



Figure 4: $\log KL(q|p)$ for each latent unit is shown at different training epochs. Low $KL$ (white) corresponds to an inactive unit. The units are sorted for visualization. It is clear that vanilla VAE cannot train the higher latent layers, while introducing batch normalization helps. Warm-up creates more active units early in training, some of which are then gradually pruned away during training, resulting in a more distributed final representation. Lastly, we see that the LVAE activates the highest number of units in each layer.

# Ladder Variational Autoencoders (вариационный автокодировщик «стремянка»)

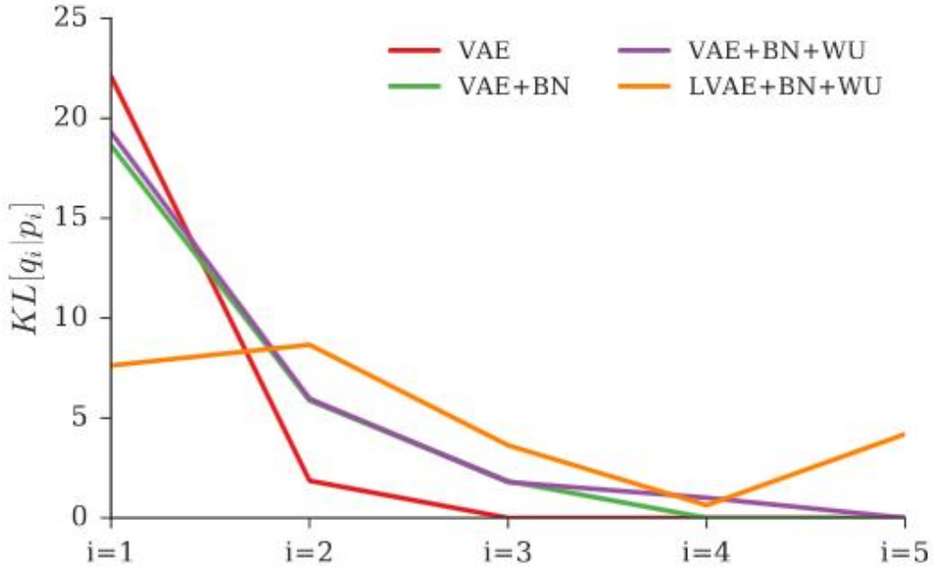



Figure 5: Layer-wise $KL[q|p]$ divergence going from the lowest to the highest layers. In the VAE models the KL divergence is highest in the lowest layers whereas it is more distributed in the LVAE model

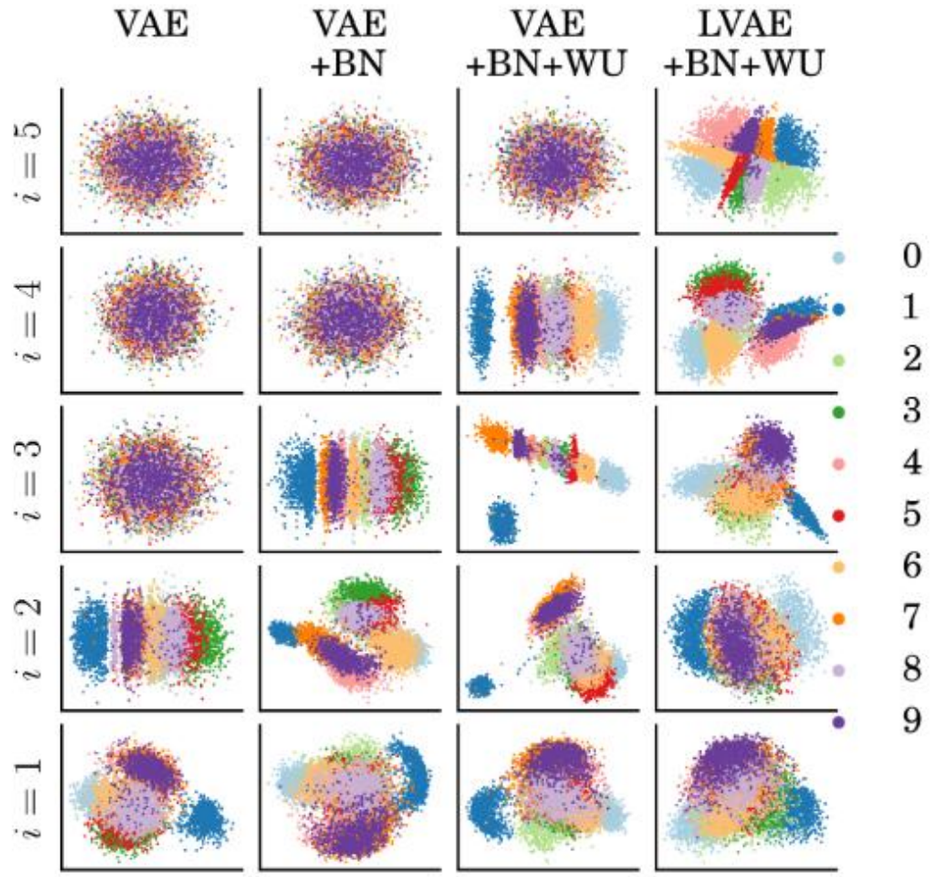
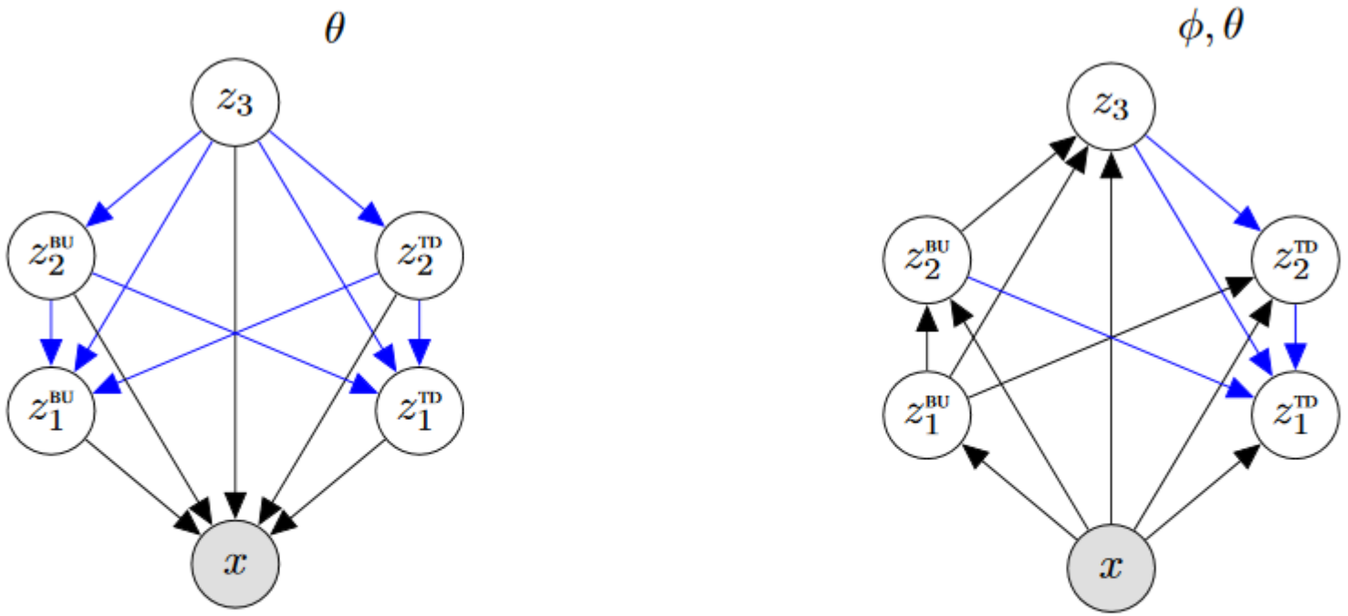



Figure 6: PCA-plots of samples from $q(z_i|z_{i-1})$ for 5-layer VAE and LVAE models trained on MNIST. Color-coded according to true class label

Casper Kaae Sønderby «Ladder Variational Autoencoders» // https://arxiv.org/abs/1602.02282

# Bidirectional-Inference Variational Autoencoder (BIVA)




(a) Generative model (b) Inference model

Figure 1: A $L = 3$ layered BIVA with (a) the generative model and (b) inference model. Blue arrows indicate that the deterministic parameters are shared between the inference and generative models. See Appendix B for a detailed explanation and a graphical model that includes the deterministic variables.

**Lars Maaløe «BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling» // https://arxiv.org/abs/1902.02102**

# Bidirectional-Inference Variational Autoencoder (BIVA)

**На каждом слое переменные расщепляются на две части:**

**BU-путь (bottom-up) и TD-путь (top-down)**

$$p_\theta(x, \mathbf{z}) = p_\theta(x|\mathbf{z})p_\theta(z_L)\prod_{i=1}^{L-1} p_\theta(z_i^{\text{BU}}|z_{>i})p_\theta(z_i^{\text{TD}}|z_{>i})$$

$$q_\phi(\mathbf{z}|x) = q_\phi(z_L|x, z_{<L}^{\text{BU}})\prod_{i=1}^{L-1} q_\phi(z_i^{\text{BU}}|x, z_{<i}^{\text{BU}})q_{\phi,\theta}(z_i^{\text{TD}}|x, z_{<i}^{\text{BU}}, z_{>i}^{\text{BU}}, z_{>i}^{\text{TD}})$$

**BIVA = LVAE +**

**1) a deterministic top-down path**

**2) apply a bidirectional inference network.**

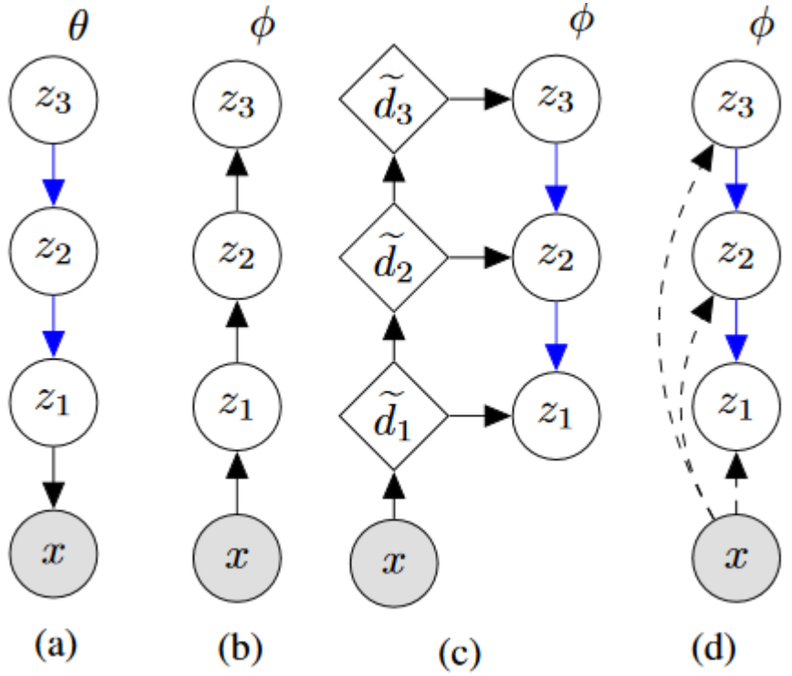
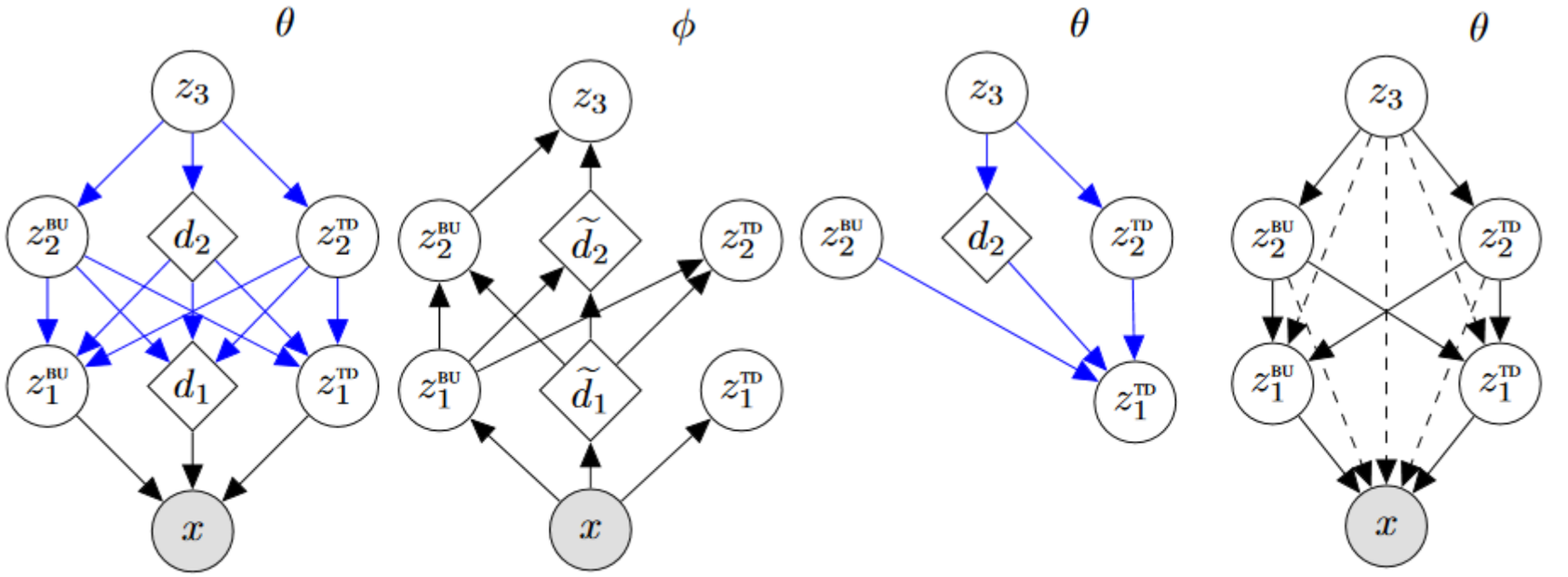



Figure 5: (a) Generative model of a VAE/LVAE with $L = 3$ stochastic variables, (b) VAE inference model, (c) LVAE inference model, and (d) skip connections among stochastic variables in the LVAE where dashed lines denote a skip-connection. Blue arrows indicate that there are shared parameters between the inference and generative model.

# Bidirectional-Inference Variational Autoencoder (BIVA)




(a) Generative model (b) BU inference (c) TD inference (d) Variable dependency

Figure 6: A $L = 3$ layered BIVA with (a) the generative model, (b) bottom-up (BU) inference path, (c) top-down (TD) inference path, and (d) variable dependency of the generative models where dashed lines denote a skip-connection. Blue arrows indicate that the deterministic parameters are shared within the generative model or between the generative and inference model.

**в генеративной модели много прокидывания связей**

**двунаправленный вывод (Bidirectional inference)**

# Bidirectional-Inference Variational Autoencoder (BIVA)

Figure 10: 64x64 CelebA samples generated from a BIVA with increasing levels of stochasticity in the model (going from close to the mode to the full distribution). In each column the latent variances are scaled with factors $0.1, 0.3, 0.5, 0.7, 0.9, 1.0$. Images in a row look similar because they use the same Gaussian random noise $\epsilon$ to generate the latent variables. BIVA has $L = 20$ stochastic latent layers connected by three layer ResNet blocks.

# Vector Quantised-Variational AutoEncoder (VQ-VAE)

**отличия от стандартного VAE**

- **выход кодировщика дискретный (дискретное латентное состояние выбирается 1-NN)**
- **prior обучаем ( над латент. пр-ом – авторегрессионное, например с помощью PixelCNN)**

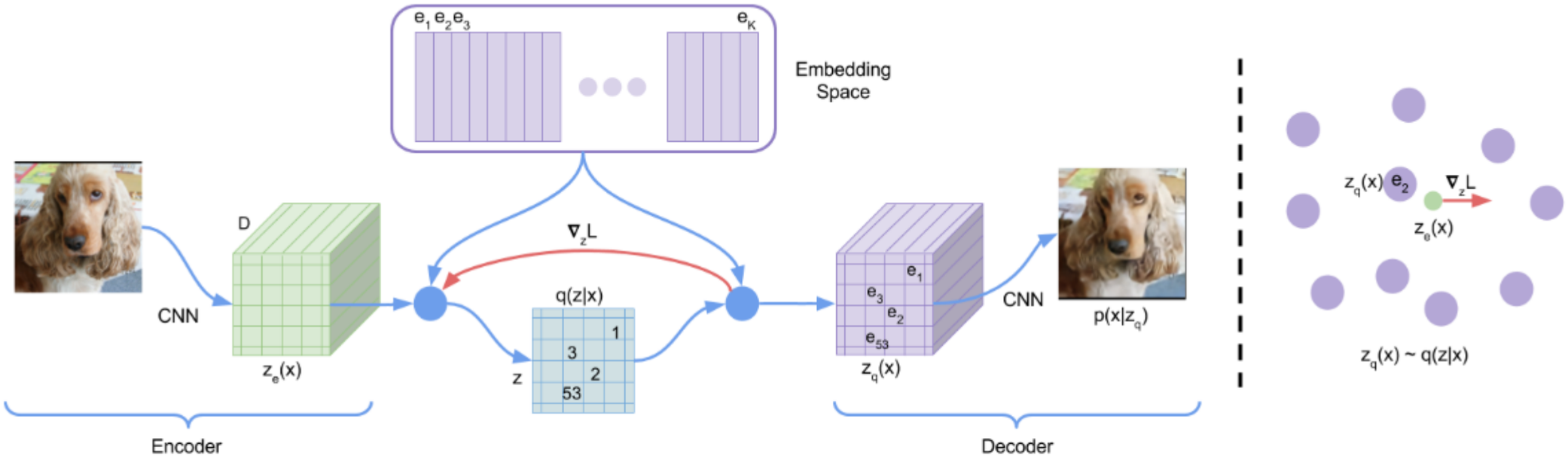



Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point $e_2$. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

**VQ-VAE: Neural Discrete Representation Learning, A. van den Oord, O. Vinyals and K. Kavukcuoglu, NeurIPS 2017, https://arxiv.org/pdf/1711.00937.pdf**

Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512.

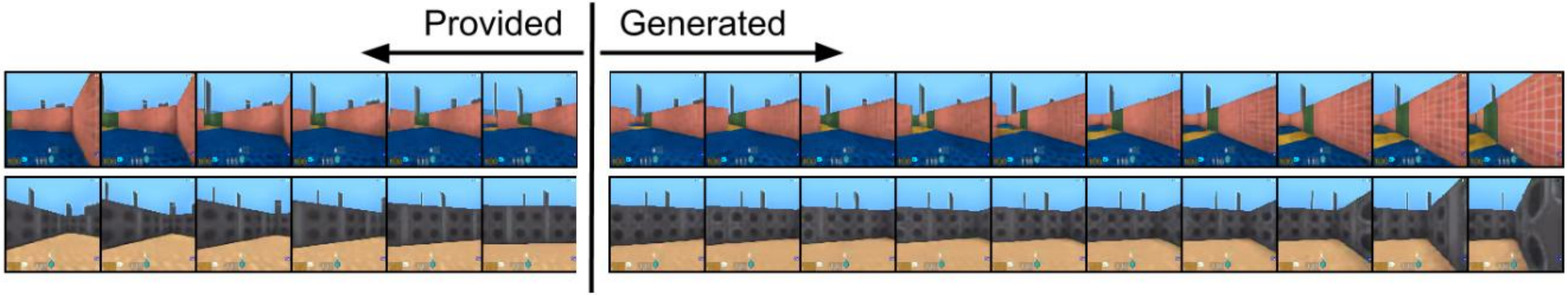



Figure 7: First 6 frames are provided to the model, following frames are generated conditioned on an action. Top: repeated action "move forward", bottom: repeated action "move right".

# VQ-VAE-2

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

**кодировщик и декодировщик – простые сети прямого распространения**
**VQ-VAE требует авторегрессионную модель в латентном пространстве**
**(а не в пространстве пикселей)**
**в отличие от GAN большое разнообразие и нет «mode collapse»**

# VQ-VAE-2

**здесь изображение кодируется на нескольких уровнях (абстракции) с помощью кодировщика см. ниже**

**Algorithm 1** VQ-VAE training (stage 1)

**Require:** Functions $E_{top}$ , $E_{bottom}$, $D$, $\mathbf{x}$ (batch of training images)

$\mathbf{h}_{top} \leftarrow E_{top}(\mathbf{x})$

▷ quantize with top codebook eq 1

$\mathbf{e}_{top} \leftarrow Quantize(\mathbf{h}_{top})$

$\mathbf{h}_{bottom} \leftarrow E_{bottom}(\mathbf{x}, \mathbf{e}_{top})$

▷ quantize with bottom codebook eq 1

$\mathbf{e}_{bottom} \leftarrow Quantize(\mathbf{h}_{bottom})$

$\hat{\mathbf{x}} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$

▷ Loss according to eq 2

$\theta \leftarrow Update(\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}))$

**Algorithm 2** Prior training (stage 2)

$\mathbf{T}_{top}, \mathbf{T}_{bottom} \leftarrow \emptyset$ ▷ training set

**for** $\mathbf{x} \in$ training set **do**

  $\mathbf{e}_{top} \leftarrow Quantize(E_{top}(\mathbf{x}))$

  $\mathbf{e}_{bottom} \leftarrow Quantize(E_{bottom}(\mathbf{x}, \mathbf{e}_{top}))$

  $\mathbf{T}_{top} \leftarrow \mathbf{T}_{top} \cup \mathbf{e}_{top}$

  $\mathbf{T}_{bottom} \leftarrow \mathbf{T}_{bottom} \cup \mathbf{e}_{bottom}$

**end for**

$p_{top} = \texttt{TrainPixelCNN}(\mathbf{T}_{top})$

$p_{bottom} = \texttt{TrainCondPixelCNN}(\mathbf{T}_{bottom}, \mathbf{T}_{top})$

▷ Sampling procedure

**while** true **do**

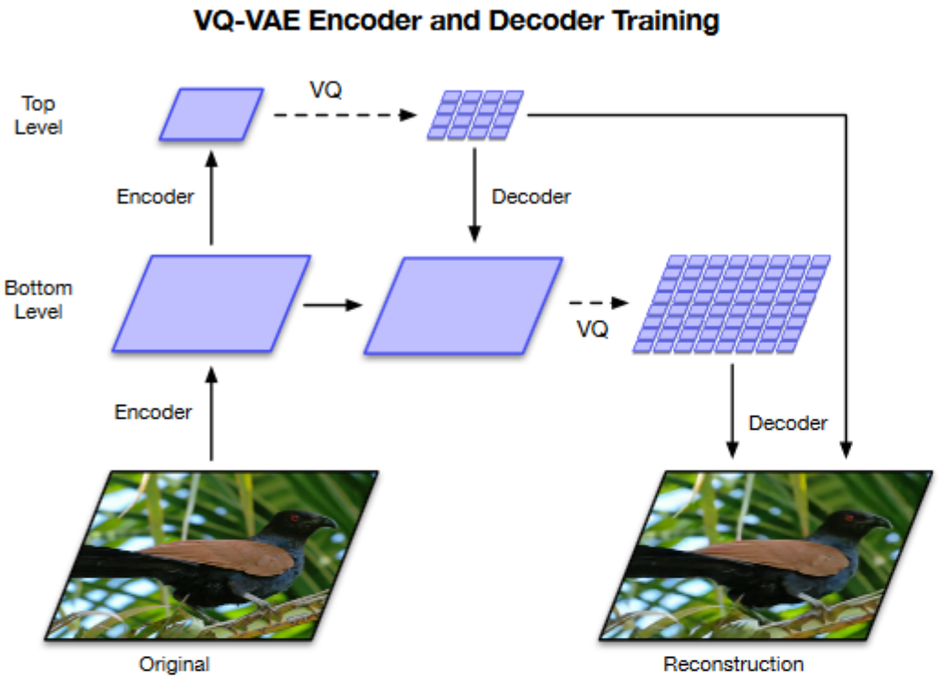
  $\mathbf{e}_{top} \sim p_{top}$

  $\mathbf{e}_{bottom} \sim p_{bottom}(\mathbf{e}_{top})$

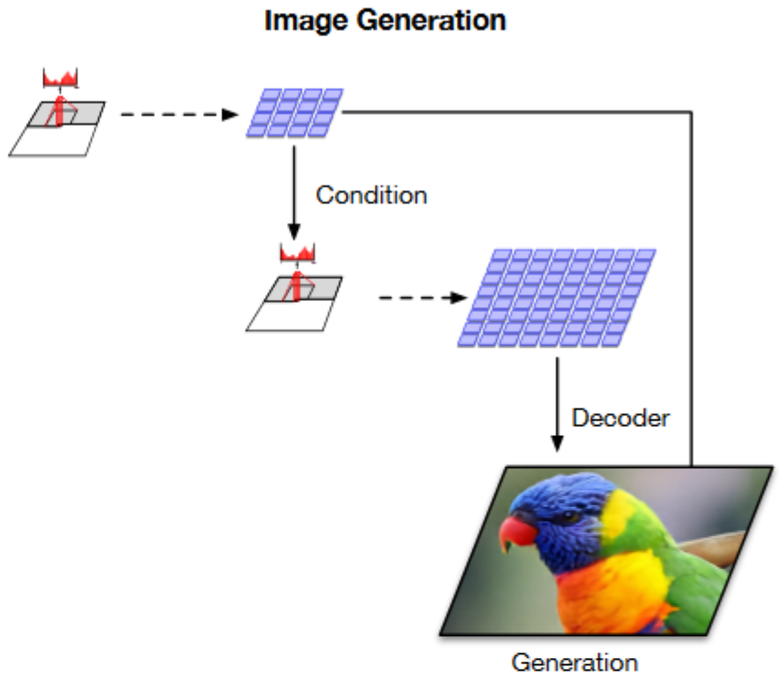
  $\mathbf{x} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$

**end while**

# VQ-VAE-2




(a) Overview of the architecture of our hierarchical VQ-VAE. The encoders and decoders consist of deep neural networks. The input to the model is a $256 \times 256$ image that is compressed to quantized latent maps of size $64 \times 64$ and $32 \times 32$ for the *bottom* and *top* levels, respectively. The decoder reconstructs the image from the two latent maps.




(b) Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

Figure 2: VQ-VAE architecture.

# VQ-VAE-2


Figure 3: Reconstructions from a hierarchical VQ-VAE with three latent maps (top, middle, bottom). The rightmost image is the original. Each latent map adds extra detail to the reconstruction. These latent maps are approximately 3072x, 768x, 192x times smaller than the original image (respectively).

**результат учёта разных уровней иерархии кодирования**

**VQ-VAE (Proposed)** **BigGAN deep**

Figure 5: Sample diversity comparison for the proposed method and BigGan Deep for Tinca-Tinca (1st ImageNet class) and Ostrich (10th ImageNet class). BigGAN samples were taken with the truncation level 1.0, to yield its maximum diversity. There are several kinds of samples such as top view of the fish or different kinds of poses such as a close up ostrich absent from BigGAN's samples. Please zoom into the pdf version for more details and refer to the Supplementary material for diversity comparison on more classes.

# VAE: Image Colorization

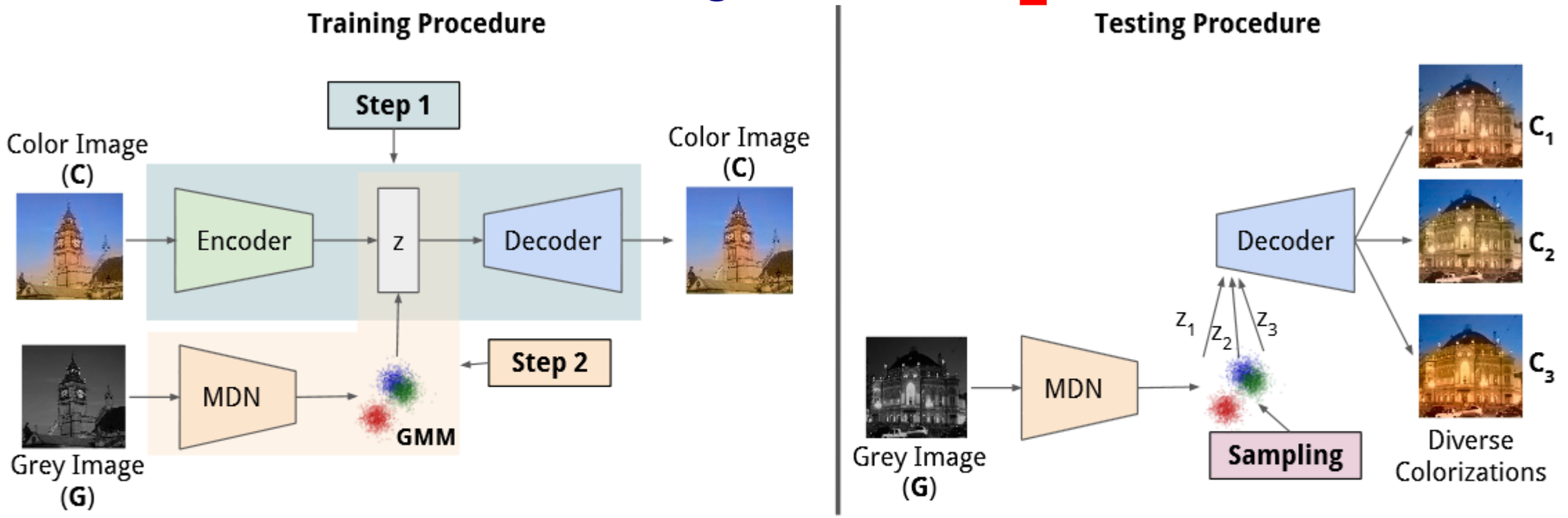



Figure 1: Step 1, we learn a low-dimensional embedding $\mathbf{z}$ for a color field $\mathbf{C}$. Step 2, we train a multi-modal conditional model $P(\mathbf{z}|\mathbf{G})$ that generates the low-dimensional embedding from grey-level features $\mathbf{G}$. At test time, we can sample the conditional model $\{\mathbf{z}_k\}_{k=1}^N \sim P(\mathbf{z}|\mathbf{G})$ and use the VAE decoder to generate the corresponding diverse color fields $\{\mathbf{C}_k\}_{k=1}^N$.

**Тонкости:**

- **функция ошибки «хитрая»**
- **разнообразие палитры (отличалась от часто встречающихся)**

Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, David Forsyth «Learning Diverse Image Colorization» https://arxiv.org/abs/1612.01958

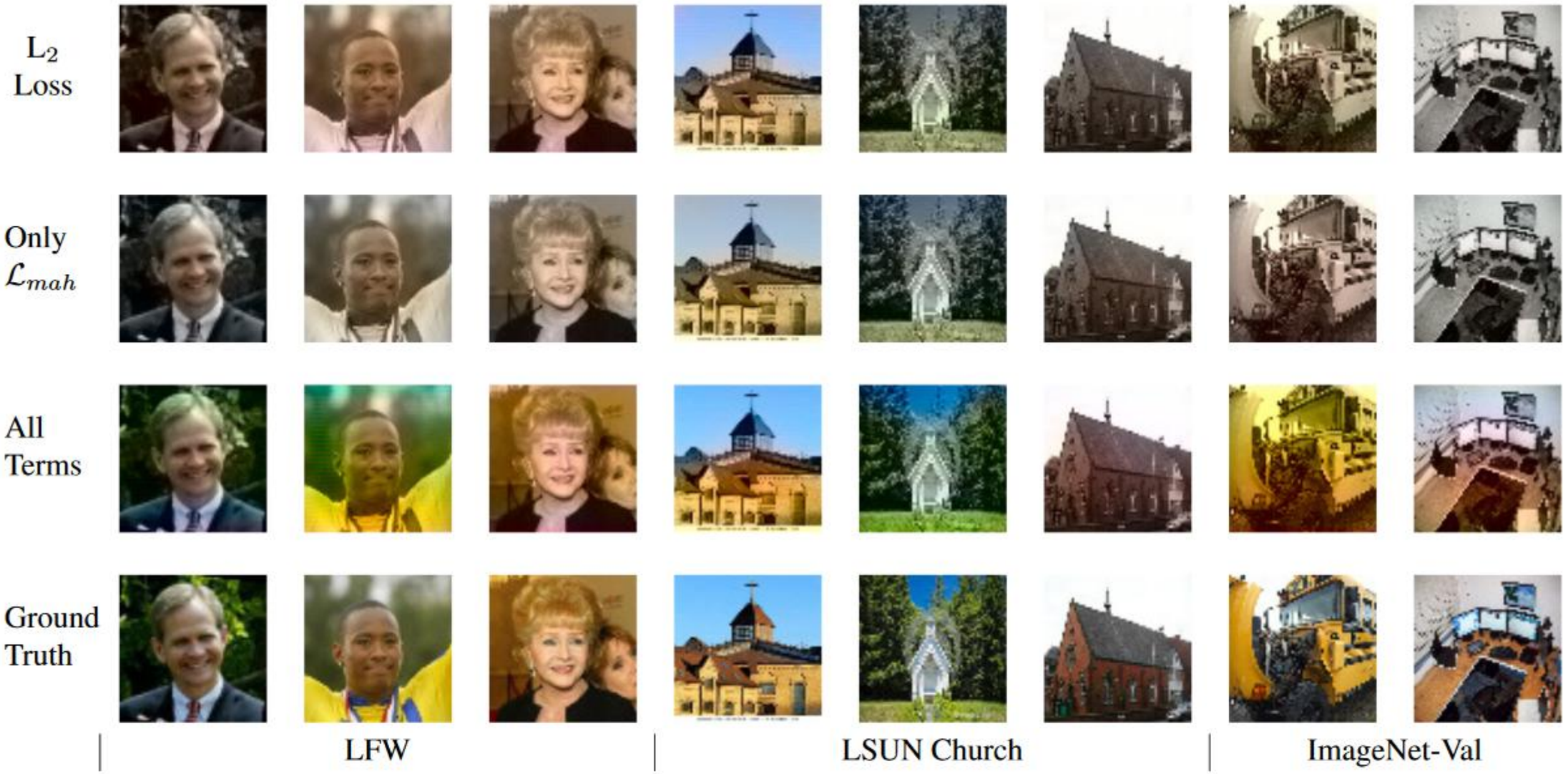
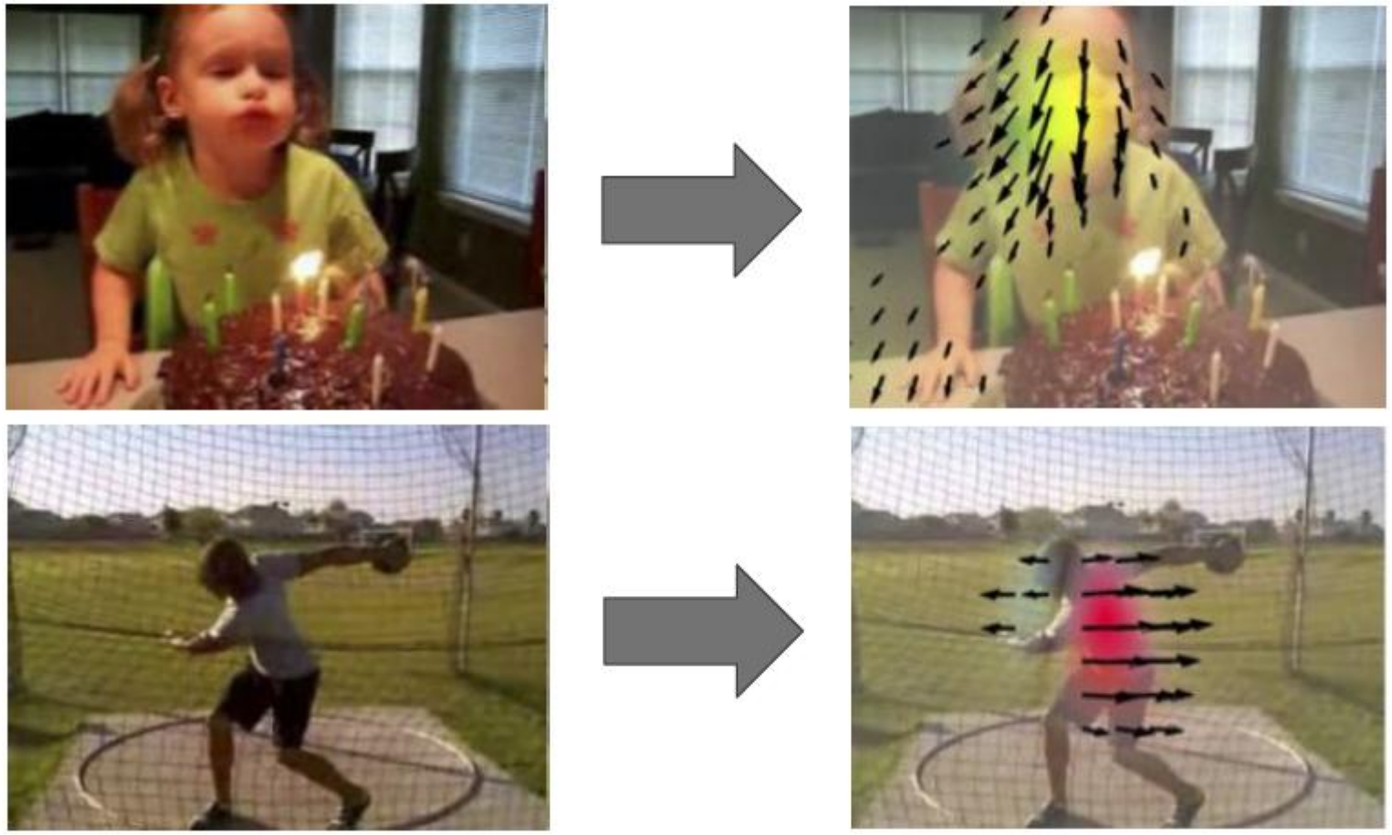



Figure 5: Qualitative results with different loss terms for the VAE decoder network. Top or $1^{st}$ Row uses only the $L_2$ loss, $2^{nd}$ row uses $\mathcal{L}_{mah}$, $3^{rd}$ row uses all the loss terms: mahalanobis, colorfulness and gradient (See $\mathcal{L}_{dec}$ of Equation 4) and last row is the ground-truth color field. These qualitative results show that using all our loss terms generates better quality color fields as compared to the standard $L_2$ loss for VAE decoders.

# VAE: Forecasting from Static Images



Jacob Walker, Carl Doersch, Abhinav Gupta, Martial Hebert An Uncertain Future: Forecasting from Static Images using Variational Autoencoders https://arxiv.org/abs/1606.07873

# VAE: Forecasting from Static Images

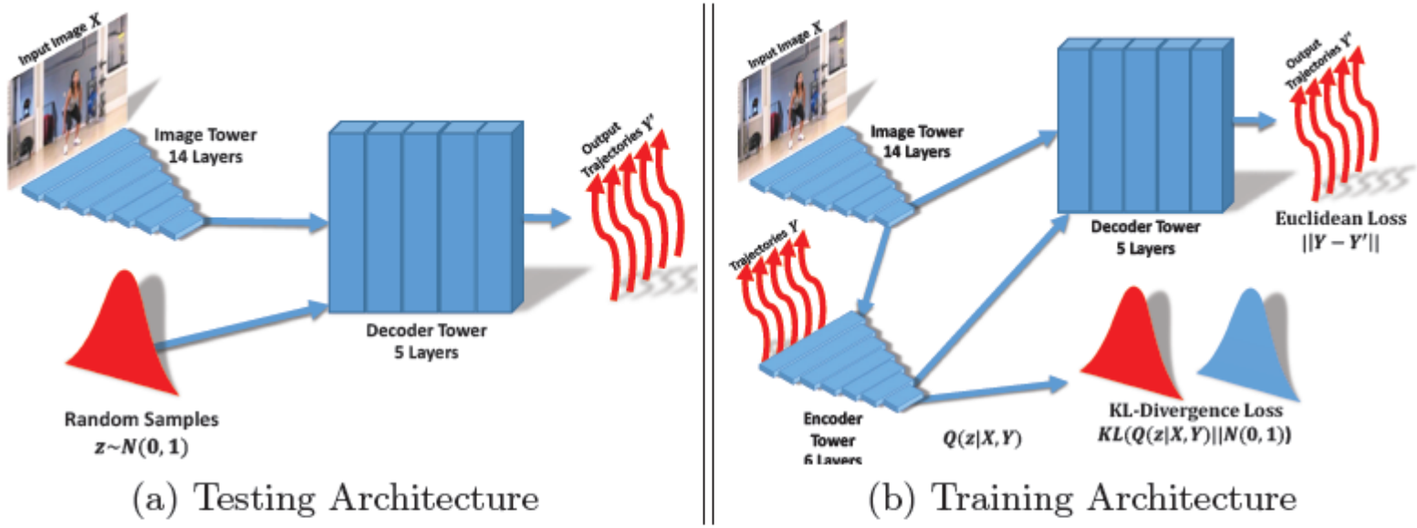



**Fig. 2.** Overview of the architecture. During training, the inputs to the network include both the image and the ground truth trajectories. A variational autoencoder encodes the joint image and trajectory space, while the decoder produces trajectories depending both on the image information as well as output from the encoder. During test time, the only inputs to the decoder are the image and latent variables sampled from a normal distribution.

## Adversarial Autoencoder

**Здесь в узком горле будет работать дискриминатор, чтобы распределение было похоже на заданное**

**чтобы понять, надо знать, что такое GAN**

# Итог

**VAE – естественная реализация автокодировщика**

**векторная арифметика в латентном пространстве**

**есть специальные латентные пространства и функции ошибок**

**далее сравним с другими решениями**