VOLGOGRAD STATE TECHNICAL UNIVERSITY

Машинное обучение и нейросетевые модели

Лекция 11. Вариационный автокодировщик

Лектор: Кравченя Павел Дмитриевич

Волгоград 2025

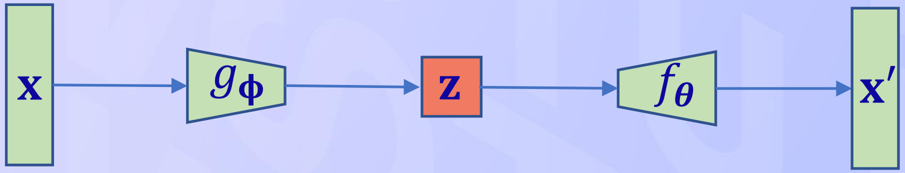
VOLGOGRAD STATE TECHNICAL UNIVERSITY

План лекции

1. Понятие автокодировщиков. Типы автокодировщиков, их свойства.
2. Вариационный автокодировщик, его архитектура и особенности.
3. Семейство распределений для VAE. Функция ошибки VAE. ELBO для VAE.
4. Явный вид функции ошибки VAE для нормального распределения данных.
5. Процесс обучения VAE и генерация новых объектов. Примеры VAE.
6. Распутанность представлений VAE. Модель β-VAE.
7. Коллапс апостериорного распределения в моделях VAE.
8. Структура латентного пространства автокодировщиков.
9. Условный вариационный автокодировщик, его обучение и генерация.
10. Особенности изображений, генерируемых VAE. Модель VAE-GAN.

- *Автокодировщик* представляет собой нейронную сеть, которая на выходе формирует сигнал, *эквивалентный* входному.
- Он *проектируется* таким образом, чтобы в процессе преобразований сигнала получить из него дополнительную информацию, например, *сжатое представление* сигнала в пространстве более *низкой размерности*.
- *Автокодировщик* состоит из *двух частей*:
  - ➢ *Кодировщик* (*encoder*) – преобразует входной сигнал $\mathbf{x} \in \mathcal{X}$ в латентный вектор $\mathbf{z} \in \mathcal{Z}$: $\mathbf{z} = g_{\boldsymbol{\phi}}(\mathbf{x})$, причём, как правило, $|\mathcal{Z}| \ll |\mathcal{X}|$.
  - ➢ *Декодировщик* (*decoder*) – восстанавливает первоначальный сигнал $\mathbf{x}' \in \mathcal{X}$, $\mathbf{x}' \approx \mathbf{x}$: $\mathbf{x}' = f_{\boldsymbol{\theta}}(\mathbf{z})$, *минимизируя* при этом *ошибку восстановления*, например: $\mathcal{L}_{AE}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathrm{MSE}(\mathbf{x}, \mathbf{x}')$.

- *Автокодировщики* подразделяются на несколько *основных типов*:
  - *Понижающие* (*undercomplete*).



  - *Повышающие* (*overcomplete*).
  - *Шумоподавляющие* (*denoising*): $$\mathcal{L}_{\mathrm{DAE}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \frac{1}{m}\sum_{i=1}^{m}\left[\mathbf{x}_i - f_{\boldsymbol{\theta}}\left(g_{\boldsymbol{\phi}}(\tilde{\mathbf{x}}_i)\right)\right]^2.$$
  - *Разреженные* (*sparse*):

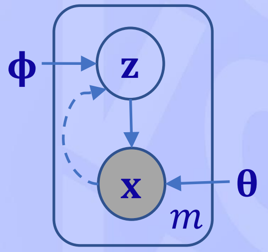
$$\mathcal{L}_{\mathrm{SAE}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) + \beta\sum_{l=1}^{L}\sum_{j=1}^{s_l}\mathrm{KL}\left(\mathcal{B}_{\rho}||\mathcal{B}_{\hat{\rho}_j^{(l)}}\right), \qquad \hat{\rho}_j^{(l)} = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(l)}(\mathbf{x}_i)\right].$$

  - *Сжимающие* (*contrastive*):

$$\mathcal{L}_{\mathrm{SAE}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) + \beta\left\|J_g(\mathbf{x})\right\|_F^2 = \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) + \beta\sum_{i,j}\left[\frac{\partial\left(g_{\boldsymbol{\phi}}(\mathbf{x})\right)_j}{\partial x_i}\right].$$

Вариационный автокодировщик

- *Вариационный автокодировщик* (*variational autoencoder, VAE*) является представителем глубоких порождающих моделей, основанный на оценке *латентных переменных* с помощью *вариационного инференса*.
- *Принцип работы VAE* заключается в следующем:
  - Каждому *объекту* (*изображению*) $\mathbf{x}_i \in \mathcal{X}$ соответствует *локальная латентная переменная* $\mathbf{z}_i \in \mathcal{Z}$.
  - $\mathbf{z} \sim q(\mathbf{z})$ отображается на $\mathbf{x}' \in \mathcal{X}$ с помощью *декодировщика*: $\mathbf{x}' = f_{\boldsymbol{\theta}}(\mathbf{z})$.
  - Переменные $\mathbf{z}_i \in \mathcal{Z}$ обучаются с использованием *амортизированного вариационного инференса*, в котором *кодировщик* играет роль *inference network*: $q(\mathbf{z} \mid \mathbf{x}) = q_{\boldsymbol{\phi}}\big(\mathbf{z} \mid g_{\boldsymbol{\phi}}(\mathbf{x})\big)$.
  - Обучение *VAE* сводится к *оптимизации*: $\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathrm{ELBO}\big(q(\mathbf{z} \mid \mathbf{x})\big)$.



Семейство вариационных распределений для VAE

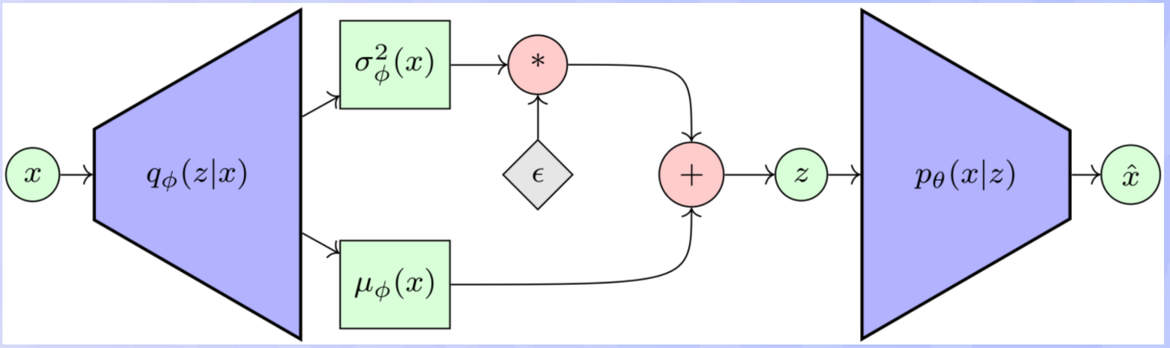
- *Вариационное семейство* для *VAE* определяется следующим образом:

$$q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{h}_{\boldsymbol{\phi}}^{(1)}(\mathbf{x}), \mathrm{diag}\left(\exp\left(\mathbf{h}_{\boldsymbol{\phi}}^{(2)}(\mathbf{x})\right)\right)\right).$$

- *Параметры* вариационного распределения $\boldsymbol{\mu} = \mathbf{h}_{\boldsymbol{\phi}}^{(1)}(\mathbf{x})$ и $\log \boldsymbol{\sigma}^2 = \mathbf{h}_{\boldsymbol{\phi}}^{(2)}(\mathbf{x})$ определяются *нейросетями*, которые, как правило, являются составными частями *кодировщика*: $\left(\mathbf{h}_{\boldsymbol{\phi}}^{(1)}, \mathbf{h}_{\boldsymbol{\phi}}^{(2)}\right) = g_{\boldsymbol{\phi}}(\mathbf{x})$.

- Для оценки *градиента ELBO* к *нормальному вариационному распределению* применяется *трюк репараметризации*:

$$\mathbf{z} = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu},$$
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}).$$



- Для *VAE* существует *модификация*, позволяющая снизить *дисперсию эстиматора градиента ELBO*. Перепишем *ELBO* следующим образом:

$$\mathrm{ELBO}(\boldsymbol{\phi},\boldsymbol{\theta}) = \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i,\mathbf{z}_i) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\right] =$$

$$= \sum_{i=1}^{m} \int q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i,\mathbf{z}_i) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\right] d\mathbf{z}_i =$$

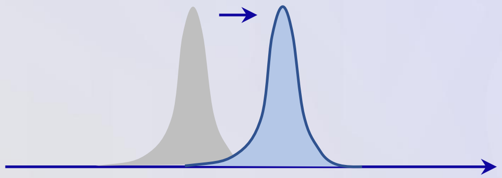
$$= \sum_{i=1}^{m} \int q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i) + \log p(\mathbf{z}_i) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\right] d\mathbf{z}_i =$$

$$= \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)\right] + \sum_{i=1}^{m} \int q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\left[\log p(\mathbf{z}_i) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)\right] d\mathbf{z}_i =$$

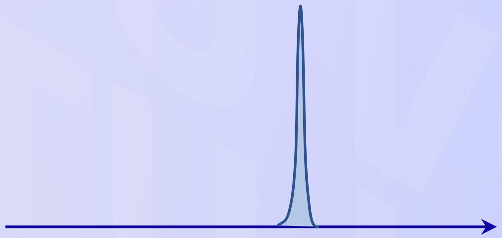
ELBO для VAE

$$= \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)] + \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}\left[\log \frac{p(\mathbf{z}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}\right] =$$

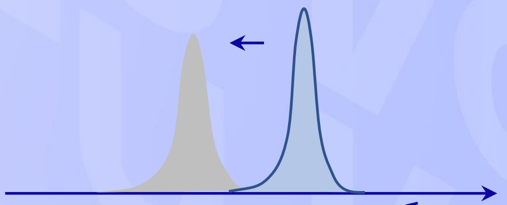
$$= \sum_{i=1}^{m} \left\{ \underbrace{\mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)]}_{\text{reconstruction loss}} - \underbrace{\mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,\|\, p(\mathbf{z}_i)\big)}_{\text{regularization term}} \right\} =$$



Вариационное распределение стремится отличаться от априорного, чтобы описать особенности в данных



Если нет регуляризации, вариационное распределение принимает узкую форму



Регуляризация ищет баланс, не позволяя вариационному распределению сильно отклониться от априорного

- В общем случае, вариационный автокодировщик описывает параметры *произвольных распределений*.
- Но если априорное распределение латентных переменных $p(\mathbf{z})$ считать стандартным нормальным $\mathcal{N}(\mathbf{0}, \mathbb{I})$, то дивергенция $\mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, p(\mathbf{z}_i)\big)$ может быть вычислена *аналитически*.
- Поскольку, часть эстиматора вычисляется *аналитически*, то отпадает необходимость в её *оценке Монте-Карло*, а это приводит к *уменьшению дисперсии эстиматора*.
- Если наблюдаемая переменная *непрерывна* и с достаточной степенью точности подчиняется *нормальному распределению*, то и *ошибка реконструкции* тоже может быть преобразована для упрощения расчетов.

Эквивалентность MSE и ELBO для VAE

- Пусть $p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i) \sim \mathcal{N}(f_{\boldsymbol{\theta}}(\mathbf{z}_i), \sigma_{\mathrm{dc}}\mathbb{I})$, σ_{dc} – некоторый *гиперпараметр*. Тогда:

$$\mathrm{ELBO}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{i=1}^{m} \left\{ \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)} [\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)] - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, p(\mathbf{z}_i)\big) \right\} =$$

$$= \sum_{i=1}^{m} \left\{ \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)} \left[\log \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{dc}}} - \frac{\|\mathbf{x}_i - f_{\boldsymbol{\theta}}(\mathbf{z}_i)\|_2^2}{2\sigma_{\mathrm{dc}}^2} \right] - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, p(\mathbf{z}_i)\big) \right\}.$$

- Таким образом, модель пытается *восстановить входной сигнал*, выполняя <u>*минимизацию*</u> $\mathrm{MSE}(\mathbf{x}, \mathbf{x}')$ (для задач *регрессии*).
- При этом модель также пытается поддерживать <u>*одинаковыми*</u> распределения $q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)$ и $p(\mathbf{z}_i)$.
- Если $p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$, то и *вариационное распределение* $q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \to \mathcal{N}(\mathbf{0}, \mathbb{I})$.

Явный вид функции ошибки VAE для случая нормального распределения выхода

- Поскольку KL-дивергенцию можно рассчитать *вручную*, то для расчета *функции ошибки* остаётся вычислить матожидание $\log p(\mathbf{x} \mid \mathbf{z})$, которое можно *оценить* с помощью *эстиматора Монте-Карло*.
- В таком случае, *функция ошибки* примет вид:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{X}) = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{K}\sum_{j=1}^{K}\log p_{\boldsymbol{\theta}}\left(\mathbf{x}_i|\mathbf{z}_j\right) + \frac{1}{2}\sum_{k=1}^{M}\left(1 + \ln\sigma_k^2(\mathbf{x}_i) - \mu_k^2(\mathbf{x}_i) - \sigma_k^2(\mathbf{x}_i)\right)\right].$$

Здесь M – размерность векторов $\boldsymbol{\mu}$ и $\boldsymbol{\sigma}$, $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = g_{\boldsymbol{\phi}}(\mathbf{X}, \boldsymbol{\epsilon})$.

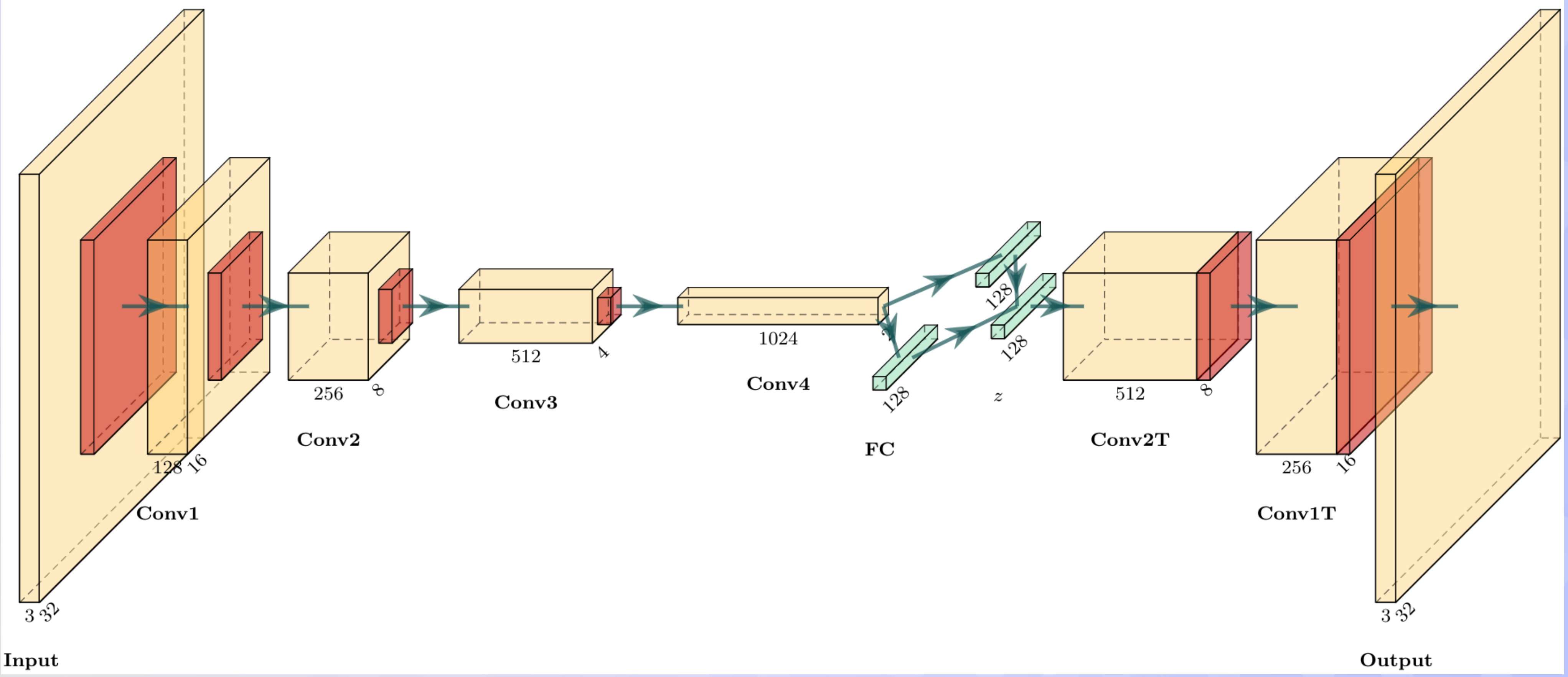
- При *достаточно большом* размере *мини-батча* количество *семплов* можно взять *небольшим* (например, *один* семпл на *100* элементов в батче).
- Поскольку применен *трюк репараметризации*, можно считать *градиент*.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Обучение VAE и генерация с его помощью

Обучение VAE	Генерация новых объектов с помощью VAE
Пусть дан *датасет* $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$. *Инициализировать* $\boldsymbol{\theta}$ и $\boldsymbol{\phi}$. **while** not converge: ➢ Сформировать случайный *минибатч*: $\mathrm{X}^m = \{x_1, x_2, \dots, x_m\} \subset \mathcal{X}$; ➢ Сформировать $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$; ➢ Вычислить *функцию ошибки*: $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathrm{X}) = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{K}\sum_{j=1}^{K}\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \vert \mathbf{z}_j) + \frac{1}{2}\sum_{k=1}^{M}\left(1 + \ln\sigma_k^2(\mathbf{x}_i) - \mu_k^2(\mathbf{x}_i) - \sigma_k^2(\mathbf{x}_i)\right)\right]$ ➢ Выполнить *backprop*(); ➢ Вычислить $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow \boldsymbol{\theta}, \boldsymbol{\phi} - \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathrm{X})$.	*Генерация* новых объектов выполняется с помощью *декодировщика*: $\mathbf{x} = f_{\boldsymbol{\theta}^*}(\mathbf{z}), \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}).$ *Кодировщик* для *генерации* новых объектов *не требуется*. Однако, иногда требуется оценить вероятность генерации моделью определенного объекта: $p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$ Для *оценки интеграла* требуется получить *семплы* $\mathbf{z}_i$. *Эффективнее* это сделать из *распределения* $q(\mathbf{z} \mid \mathbf{x})$, задаваемого *кодировщиком*, чем из $\mathcal{N}(\mathbf{0}, \mathbb{I})$.

Пример архитектуры VAE



Распутанность представлений VAE

- Если *каждая латентная переменная* чувствительна только к одному порождающему фактору и относительно инвариантна к другим, то говорят, что представление *распутано* (*disentangled*).
- Одним из преимуществ *распутанного представления* является хорошая интерпретируемость и простота обобщения. Например, модель, обученная на фотографиях человеческих лиц, может определять цвет кожи, цвет и длину волос, эмоции, и т.д.
- Модель *β-VAE* представляет собой *модификацию* вариационного автокодировщика, в которой *особое внимание* уделяется *обнаружению распутанных* скрытых факторов.

- В β-*VAE* требуется *максимизировать* вероятность *генерации* _реальных данных_, сохраняя при этом *небольшое расстояние* между *априорным* и *вариационным* распределениями латентных переменных:

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathbb{E}_{z\sim q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})} \log p_\theta(\mathbf{x}\,|\,\mathbf{z})\right], \qquad \text{при условии } \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p_{\boldsymbol{\theta}}(\mathbf{z})\right) < \epsilon.$$

- Данное выражение может быть записано в виде _Лагранжиана_ с _множителями Лагранжа_ β:

$$\mathcal{F}(\boldsymbol{\theta},\boldsymbol{\phi},\beta) = \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i\sim q_{\boldsymbol{\phi}}(\mathbf{z}_i\,|\,\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i\,|\,\mathbf{z}_i)] - \beta\left[\mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_i\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z}_i)\right) - \epsilon\right] =$$

$$= \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i\sim q_{\boldsymbol{\phi}}(\mathbf{z}_i\,|\,\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i\,|\,\mathbf{z}_i)] - \beta\mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_i\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z}_i)\right) + \beta\epsilon \geq$$

$$\geq \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)] - \beta \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, p(\mathbf{z}_i)\right), \qquad \text{поскольку } \beta, \epsilon \geq 0.$$

- Теперь можно ввести функцию ошибки для β-VAE:

$$\mathcal{L}_{\beta VAE} = -\sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i)] + \beta \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i) \,||\, p(\mathbf{z}_i)\right).$$

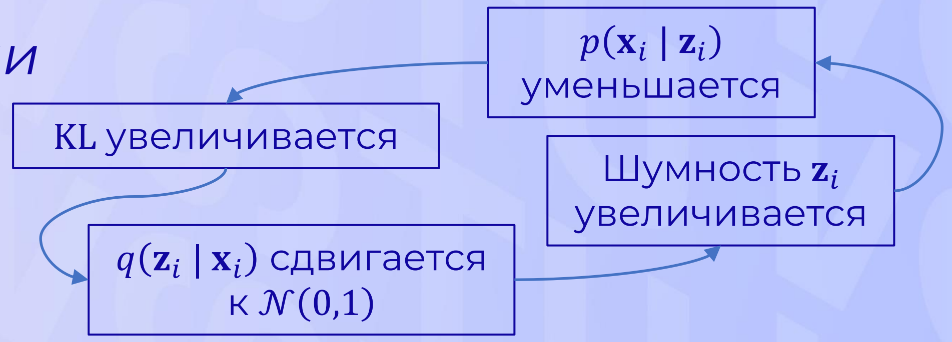
- *Отрицательное* значение *функции ошибки* является *нижней границей лагранжиана*. *Минимизация* функции ошибки приводит к *максимизации* лагранжиана и решению оптимизационной задачи.
- При $\beta = 1$ *β-VAE* функционирует как вариационный автокодировщик. При $\beta > 1$ *увеличиваются ограничения*, которые накладываются на *латентные переменные*.

Распутывание признаков в β-VAE

- Введённые условия *ограничивают* *информационную ёмкость* для $\mathbf{z}$, что в сочетании с необходимостью *максимизировать логарифмическую вероятность* обучающих данных $\mathbf{x}$ должно *стимулировать* модель к изучению *наиболее эффективного представления*.
- Модели приходится искать *компромисс* между *качеством реконструкции* и *KL-дивергенцией*.
- Так как *различные* факторы по-разному влияют на *потери модели* при *реконструкции*, то ей выгодно выполнить *распутывание признаков*, поскольку в этом случае она может *напрямую ранжировать* важность (с помощью KL-*дивергенции*) для *каждого* из них.

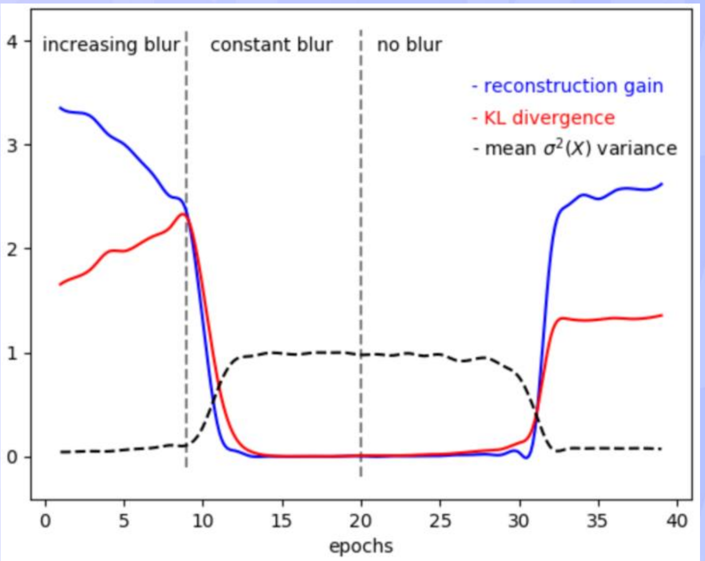
Коллапс апостериорного распределения

- Если во время *обучения VAE* вклад *ошибки восстановления* для *некоторой* латентной переменной *мал* по сравнению со *слагаемым Кульбака-Лейблера*, то распределение этой переменной из-за будет *сдвигаться* из-за *регуляризации* к априорному распределению $p(\mathbf{z}_i)$, не учитывающему $\mathbf{x}$.
- В результате, латентная переменная станет *ещё более шумной*, и *декодировщик перестанет учитывать* её при создании *новых* образцов.
- А это приведет к *постоянной генерации* некоторых «*усредненных*» объектов.
- Данный эффект называется *коллапсом апостериорного распределения*.



Воспроизведение коллапса апостериорного распределения в VAE

- Если *добавить* к латентной переменной *искусственный шум*, можно *воспроизвести* эффект *коллапса апостериорного распределения*.
- *Вклад* латентной переменной в *реконструкцию* можно вычислить как *разницу* между ошибками реконструкции с *учётом* и *без учета* переменной. Он называется *reconstruction gain*.
- Когда *reconstruction gain* переменной становится меньше, чем *KL-дивергенция*, сама переменная *игнорируется сетью*. *Дисперсия* переменной при этом *увеличивается*.
- Если *искусственный шум удалить*, латентная переменная вновь *активируется*.

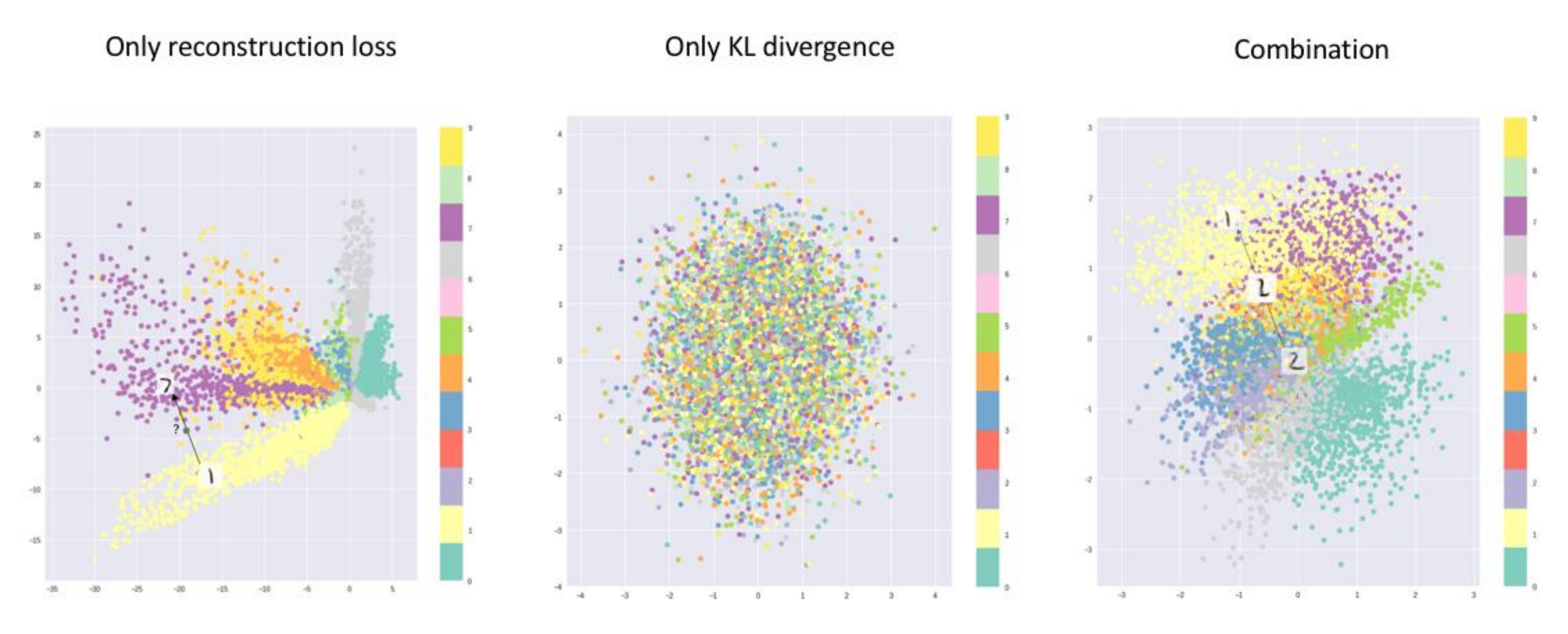


Структура латентного пространства стандартного автокодировщика

- *Стандартные автокодировщики* хорошо работают в задачах *восстановления данных*, однако, они *не подходят* в качестве *генеративной* модели, поскольку выбор *случайного* входного сигнала $\mathbf{z}'$ для декодера *не обязательно* приведет к тому, что декодер создаст *приемлемое* изображение.
- *Близкие* значения $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$, $\mathbf{x}_1 \approx \mathbf{x}_2$, *не обязательно* будут *отображаться* в *близкие* значения векторов латентного пространства: $g(\mathbf{x}_1) \not\approx g(\mathbf{x}_2)$.
- Близкие значения $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}$, $\mathbf{z}_1 \approx \mathbf{z}_2$, *не обязательно* будут *отображаться* в *близкие* значения векторов данных: $f(\mathbf{z}_1) \not\approx f(\mathbf{z}_2)$. Более того, если $\mathbf{x} \in \mathcal{X}$, $\mathbf{z} \in \mathcal{Z}$, $\mathbf{x} = f(\mathbf{z})$, то *не обязательно*, чтобы при $\mathbf{z}' \approx \mathbf{z}$ было справедливо: $f(\mathbf{z}') \in \mathcal{X}$.

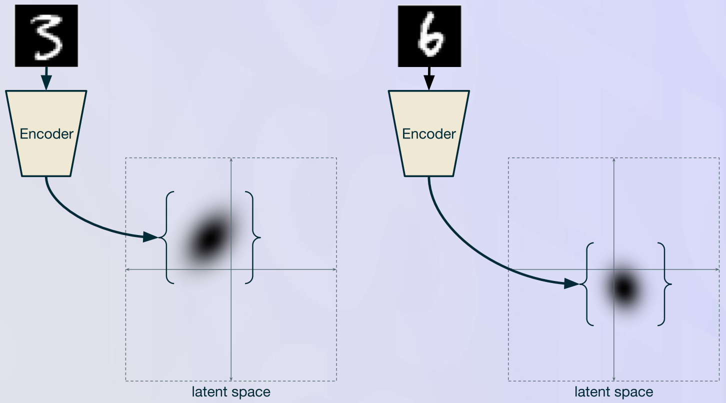
VOLGOGRAD STATE TECHNICAL UNIVERSITY

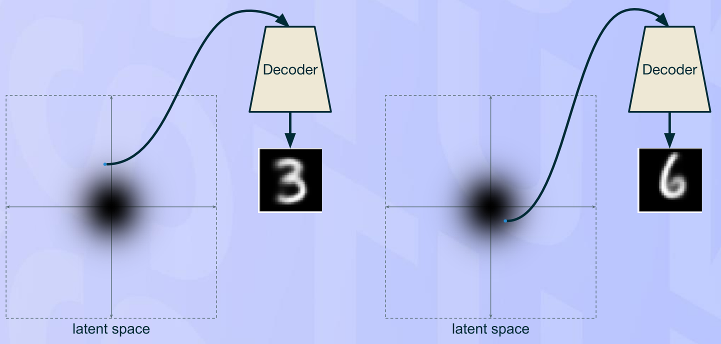
Структура латентного пространства AE и VAE



Структура латентного пространства вариационного автокодировщика

- Нужен какой-то способ *гарантировать*, что *декодер* готов преобразовывать любой *входной* сигнал в *разумное* изображение. Для этого нужно заранее определить <u>*распределение входных данных*</u>, которые *декодер* должен ожидать. В вариационном автокодировщике с этой целью обычно используется <u>*стандартное нормальное распределение*</u> $\mathcal{N}(\mathbf{0}, \mathbb{I})$.
- Онлайн-ресурс с VAE: https://www.siarez.com/projects/variational-autoencoder





Условный вариационный автокодировщик

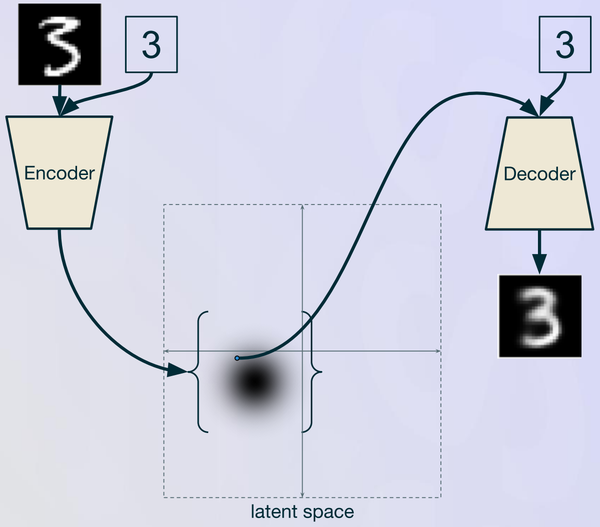
- Часто требуется *сгенерировать* не просто *какой-либо произвольный* объект из датасета, а относящийся *к конкретной группе или классу*.
- Для этого потребуем, чтобы *распределения*, участвующие при построении *вариационного автокодировщика*, были *условными* по некоторой *переменной* $\mathbf{y} \in \mathcal{Y}$, характеризующей *принадлежность объекта к требуемому классу*: $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$, $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{y})$.
- В этом случае, *функция ошибки* для *условного автокодировщика*:

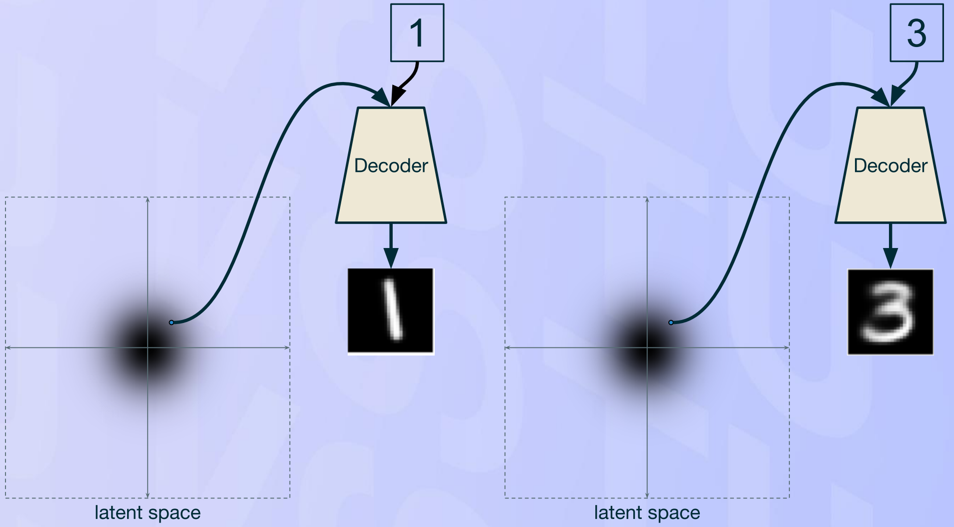
$$\mathrm{ELBO}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{i=1}^{m} \left\{ \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i, \mathbf{y}_i)} [\log p_{\boldsymbol{\theta}}(\mathbf{x}_i \mid \mathbf{z}_i, \mathbf{y}_i)] + \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_i \mid \mathbf{x}_i, \mathbf{y}_i) \mid\mid p(\mathbf{z}_i, \mathbf{y}_i)\big) \right\}.$$

- *Переменная* $\mathbf{y} \in \mathcal{Y}$ может иметь *любую природу* (быть, например, *меткой*).

Условный вариационный автокодировщик

- *Обучение CVAE отличается* от обучения *нескольких независимых VAE*, поскольку веса *CVAE* являются *общими* для *всех классов*.
- При *реализации CVAE* входы *кодировщика* и *декодировщика* обычно *конкатенируют* с тензором $\mathbf{y}$.



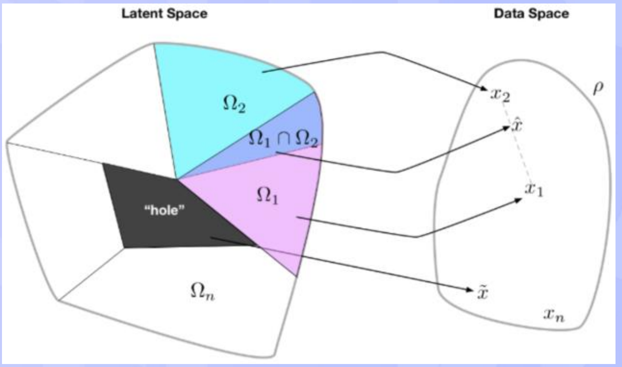


VSTU VOLGOGRAD STATE TECHNICAL UNIVERSITY

Особенности изображений, сгенерированных VAE

- Изображение *классического VAE* часто *размыто* (*blurry*).
- Одной из причин этого является *неоднозначность отображения* входных данных в латентное пространство.
- Для *разных векторов данных* могут наблюдаться *перекрывающиеся* латентные переменные.

Оптимальная реконструкция для *классических* *VAE без ограничений* предполагает расчет *среднего* между этими векторами, что приводит к *размытию*.



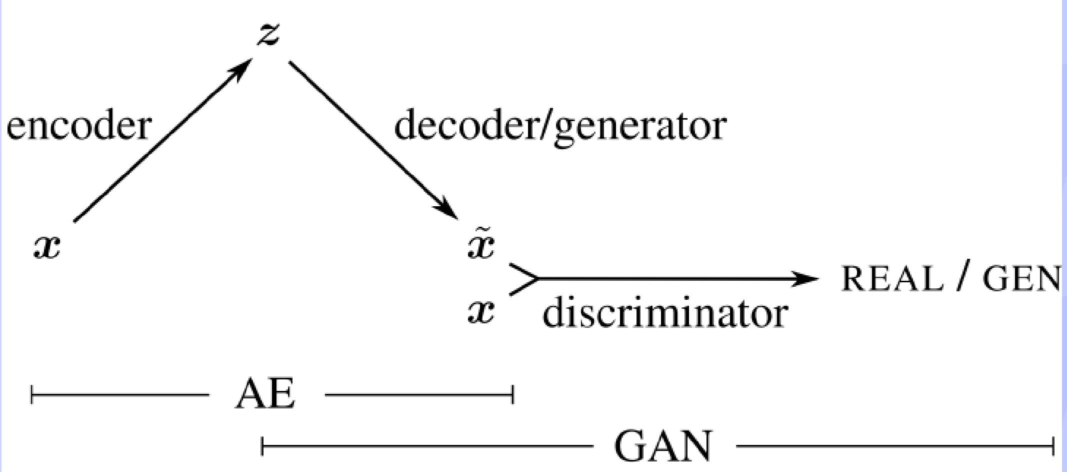
- Также, в VAE может наблюдаться проблема «*дыр*»: областях из $\mathcal{Z}$, в которых *декодер ничем не ограничен*, поэтому, может выдавать *произвольные выходные данные*.

Особенности изображений, сгенерированных VAE

- Еще одной *причиной* появления *размытости* является использование *попиксельных* ошибок (например, *среднеквадратичной*) *VAE* при расчете ошибки *восстановления*.
- Поскольку *архитектура VAE* предполагает *сжатие входного сигнала*, она *не сохраняет* всю информацию о входном изображении. Нюансы изображения часто *теряются*, и VAE будет прогнозировать что-то *среднее* между изученными изображениями, чтобы минимизировать ошибку *восстановления*. Это приводит к тому, что изображение становится *размытым*.
- Есть несколько *подходов*, предполагающих *замену функции ошибки* на более *подходящую* в конкретных случаях, например, *архитектура VAE-GAN*.

Подход к объединению VAE и GAN

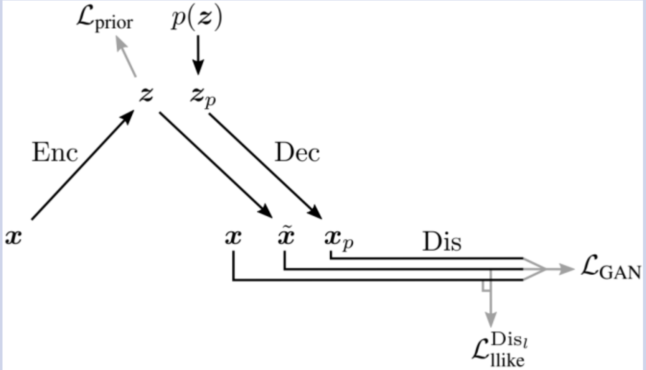
- Подход строится на том, чтобы обучить *вариационный автокодировщик* *генерировать* новые изображения, но требуемую *метрику* определять *автоматически* посредством *GAN*.
- *Объединим декодировщик VAE* и *генератор GAN* в единую нейросеть, которая занимается задачей формирования *восстановленных* объектов.
- *Дискриминатор GAN* пытается *отличить реальный* объект от *сгенерированного.*
- Для оценки *схожести объектов* требуется ввести *функцию ошибки* на основе *сравнения признаков*.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

Архитектура и процесс обучения VAE-GAN

Архитектура VAE-GAN



$$p\big(\text{Dis}_l(\mathbf{x})\big) = \mathcal{N}\big(\text{Dis}_l(\mathbf{x}) \mid \text{Dis}_l(\tilde{\mathbf{x}}), \mathbb{I}\big),$$

$$\mathcal{L}_{\text{llike}}^{\text{Dis}_l} = -\mathbb{E}_{q(\mathbf{z} \mid \mathbf{x})}\big[\log p\big(\text{Dis}_l(\mathbf{x}) \mid \mathbf{z}\big)\big],$$

$$\mathcal{L}_{\text{prior}} = \text{KL}\big(q(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z})\big),$$

$$\mathcal{L}_{\text{GAN}} = \log\big(\text{Dis}(\mathbf{x})\big) + \log\big(1 - \text{Dis}(\text{Dec}(\mathbf{z}))\big),$$
$$+ \log\big(1 - \text{Dis}(\text{Dec}(\text{Enc}(\mathbf{x})))\big)$$

$$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l} + \mathcal{L}_{\text{GAN}}.$$

Процесс обучения VAE-GAN

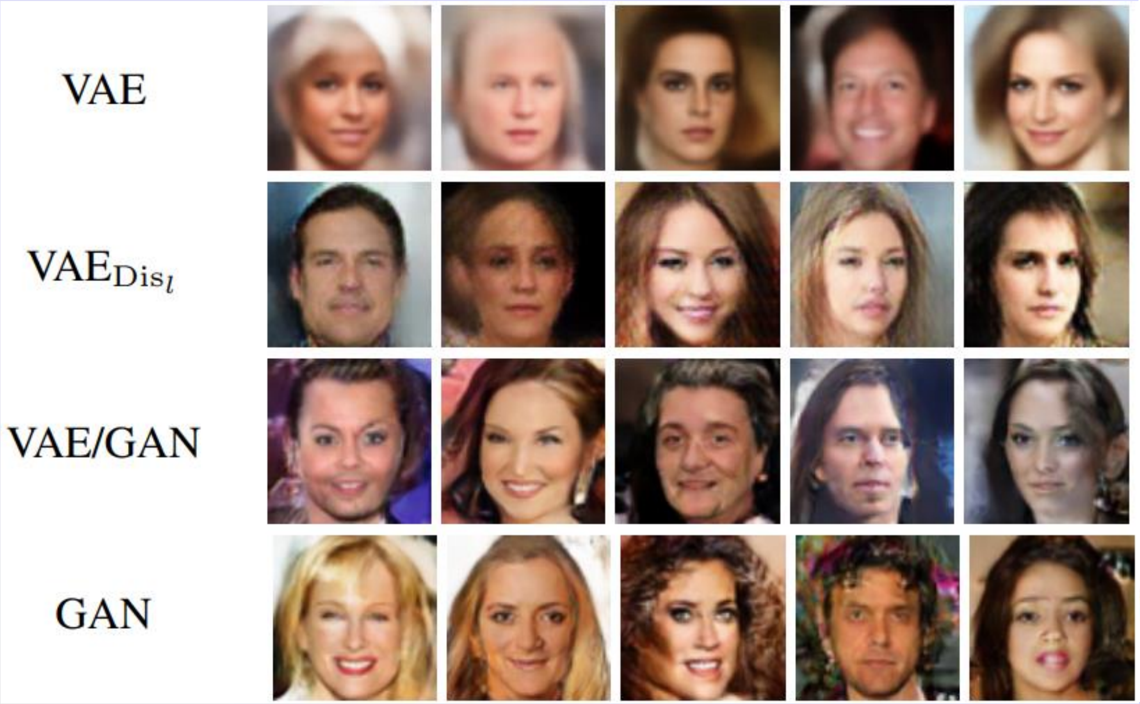
Инициализировать $\boldsymbol{\theta}_{\text{Enc}}, \boldsymbol{\theta}_{\text{Dec}}, \boldsymbol{\theta}_{\text{Dis}}$.

while not converge:

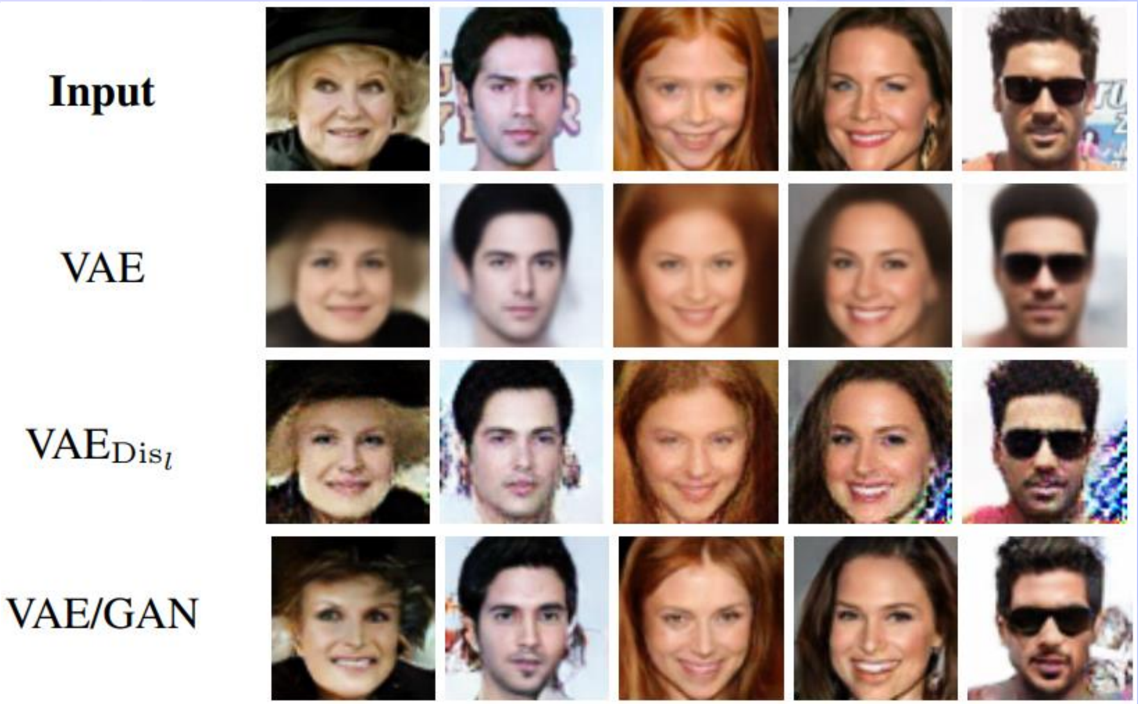
- Сформировать случайный *минибатч* $\mathbf{X}$;
- $\mathbf{Z} \leftarrow \text{Enc}(\mathbf{X})$;
- $\tilde{\mathbf{X}} \leftarrow \text{Dec}(\mathbf{Z})$;
- $\mathbf{Z}_p \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$;
- $\mathbf{X}_p \leftarrow \text{Dec}(\mathbf{Z}_p)$;
- Рассчитать $\mathcal{L}_{GAN}(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{X}_p)$;
- $\boldsymbol{\theta}_{\text{Enc}} \leftarrow -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}\big(\mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l}\big)$;
- $\boldsymbol{\theta}_{\text{Dec}} \leftarrow -\nabla_{\boldsymbol{\theta}_{\text{Dec}}}\big(\gamma\mathcal{L}_{\text{llike}}^{\text{Dis}_l} - \mathcal{L}_{\text{GAN}}\big)$;
- $\boldsymbol{\theta}_{\text{Dis}} \leftarrow -\nabla_{\boldsymbol{\theta}_{\text{Dis}}}\mathcal{L}_{\text{GAN}}$;

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Архитектура и процесс обучения VAE-GAN



Семплирование



Реконструкция

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Демонстрация практических примеров

Заключение

1. Ввели понятие новой глубокой порождающей модели – VAE – и поговорили про её особенности и свойства.
2. Получили функцию ошибки вариационного автокодировщика, разобрались с алгоритмом его обучения и генерации новых образцов.
3. Поговорили про особенности латентного пространства автокодировщиков.
4. Исследовали распутанность представлений VAE и ввели архитектуру β-VAE.
5. Ввели понятие условного вариационного автокодировщика.
6. Поговорили про особенности и проблемы, связанные с обучением VAE, ввели понятие коллапса апостериорного распределения.
7. Теоретически и на практическом примере разобрались с принципами функционирования вариационных автокодировщиков.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Спасибо за внимание!

Волгоград 2025