



*Российская Академия Наук*

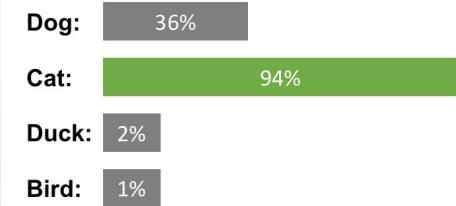
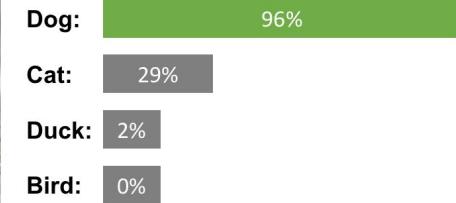
# Flow-based models: не GAN'ом единствим

Сергей Николенко

11 декабря 2021 г.

# Порождающие модели

- Большинство моделей машинного обучения **дискриминирующие**, то есть они умеют различать объекты разных типов (классификация) или выдавать одно-два числа (регрессия)
- **Проблема:** как сделать модель так, чтобы она не распознавала (классифицировала), а **порождала** новые объекты?



## Случайный шум



# Порождающие модели

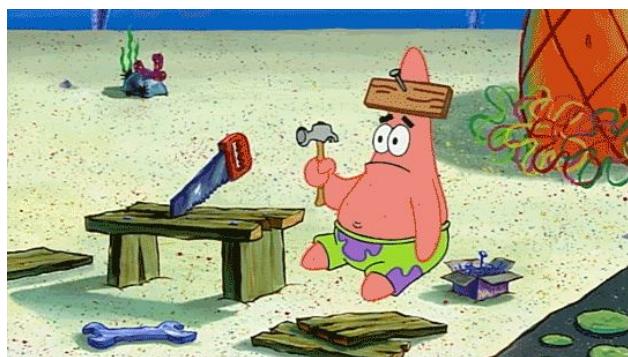
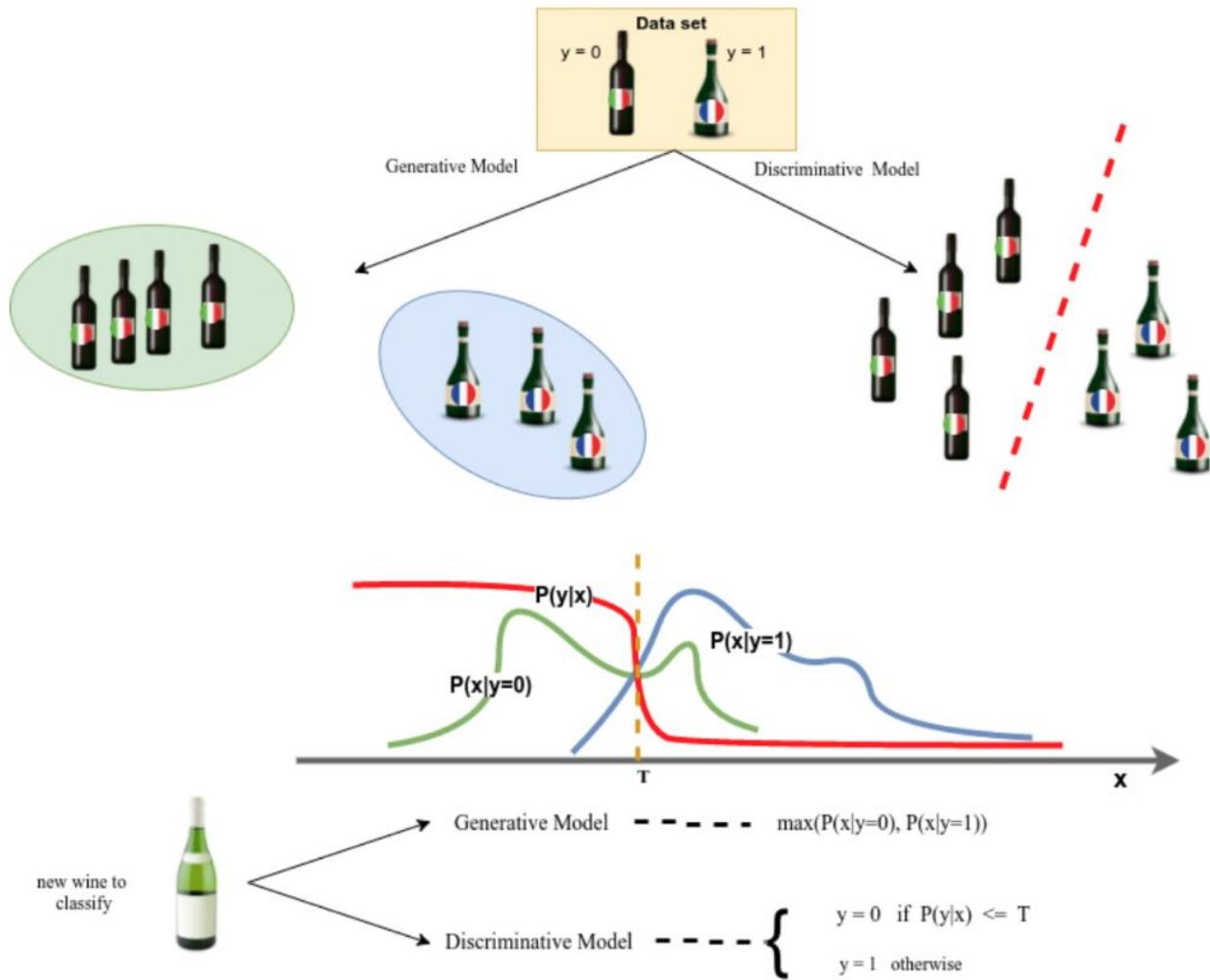
- Порождающие модели (generative models) очевидно сложнее:

$$p(\mathbf{x}, y) = p(y | \mathbf{x}) p(\mathbf{x})$$

- Хотя могут быть и простые:

## наивный Байес —

это порождающая модель, она просто делает очень сильные предположения



# Порождающие модели в deep learning

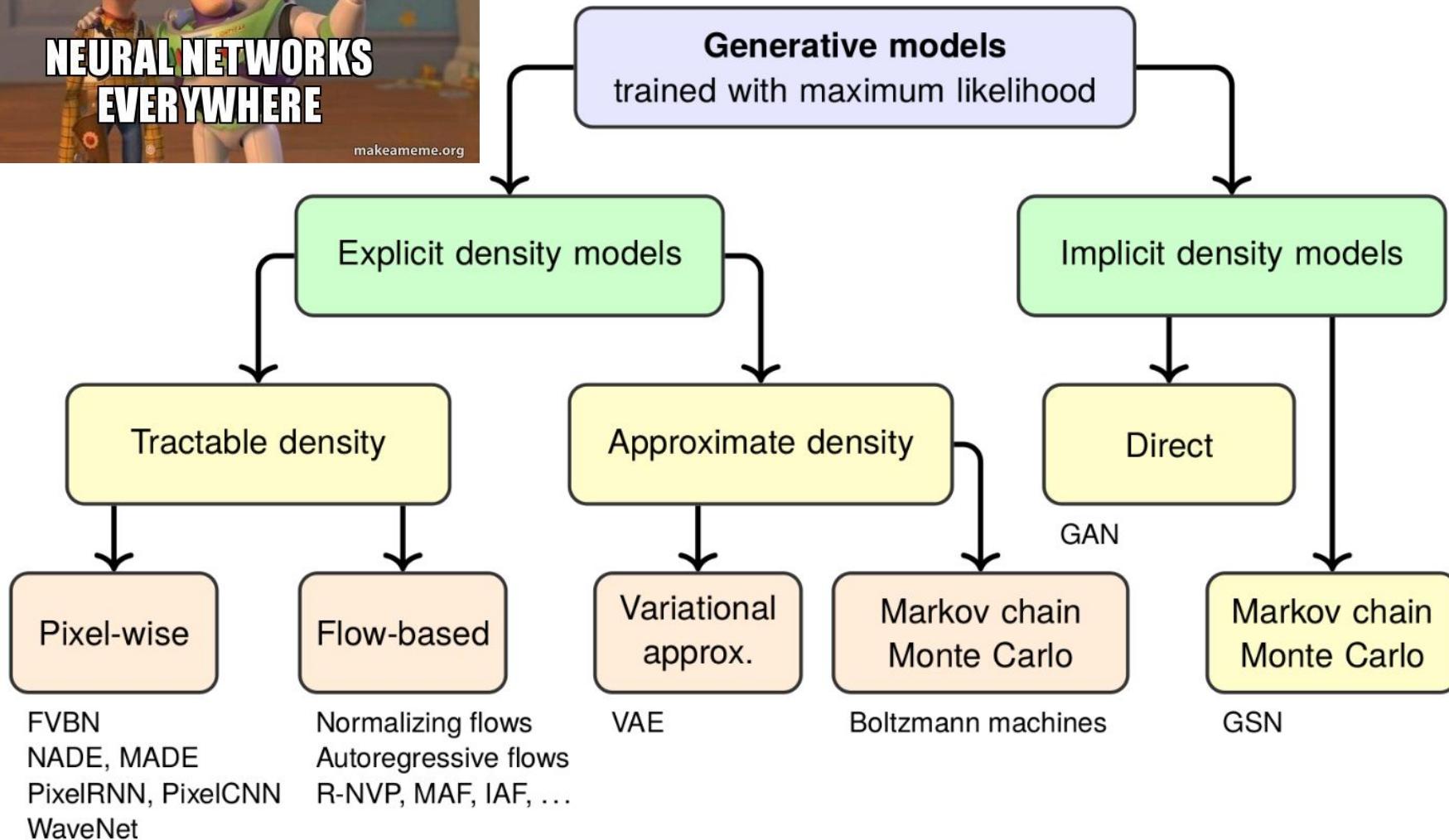


## NEURAL NETWORKS

NEURAL NETWORKS  
EVERWHERE

makeameme.org

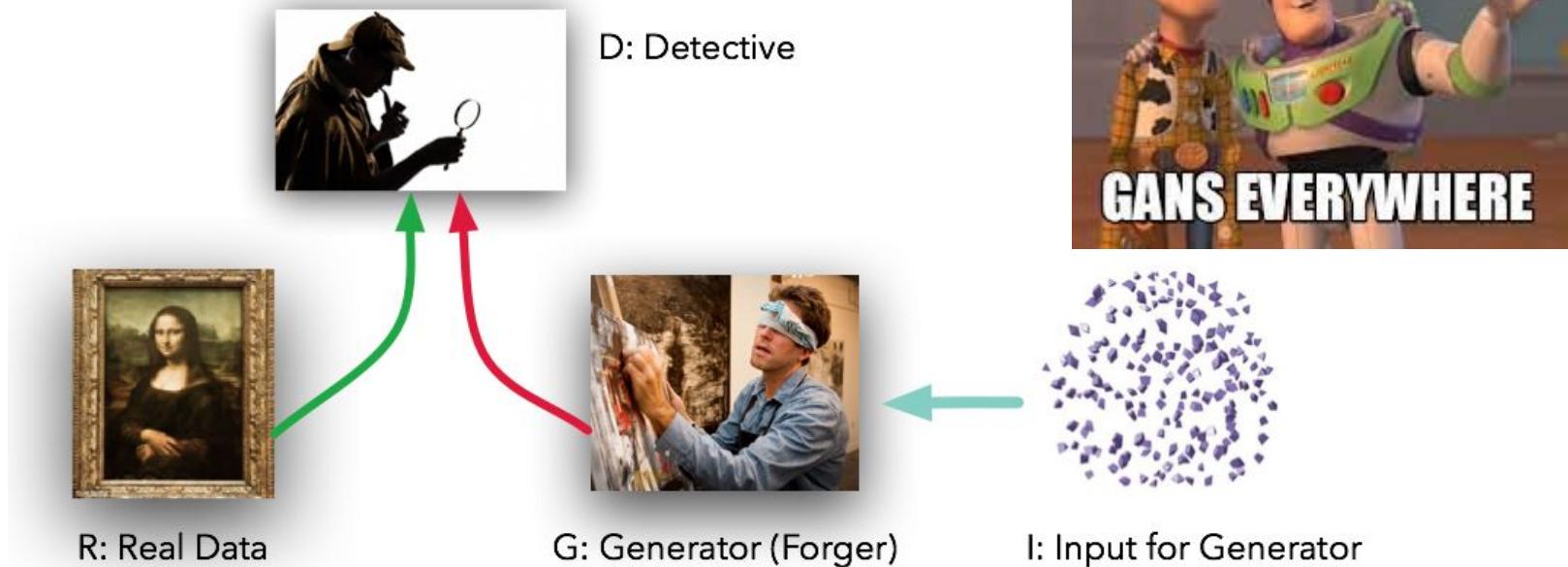
➤ Нейросети сюда можно воткнуть  
многими разными способами:



# Порождающие модели в deep learning



- Все вы слышали о **порождающих состязательных сетях** (generative adversarial networks, **GAN**):

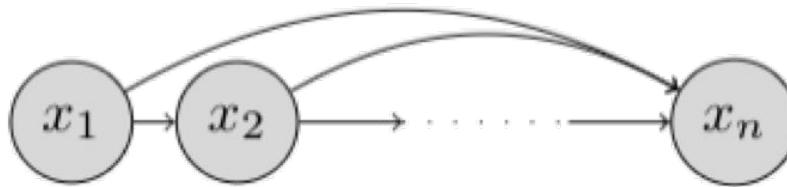


- GAN – это только сэмплирование, плотность задаётся неявно (**implicit density**)
- Поэтому их **сложно обучать**

# Авторегрессивные модели

- Если плотность дана явно (**tractable density**), **обучать легко**: просто максимизируем правдоподобие  $p(\mathbf{x})$
- Такие модели часто устроены **авторегрессивным** образом:

$$p(x_1, x_2, \dots, x_n) = p(x_1) p(x_2 | x_1) p(x_3 | x_1, x_2) \dots p(x_n | x_1, x_2, \dots, x_{n-1})$$



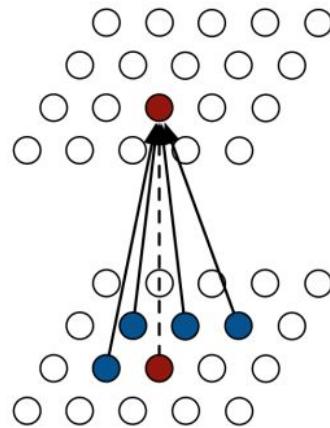
- Порождаем следующий кусочек объекта (пиксель, отчёт звука, слово) при условии всех предыдущих
- Такое разложение верно всегда, а дальше надо делать какие-то предположения, чтобы это стало возможно обучить и подсчитать
- Например, наивный Байес делает такое предположение:

$$p(y, x_1, x_2, \dots, x_n) = p(y) p(x_1 | y) p(x_2 | y) \dots p(x_n | y)$$

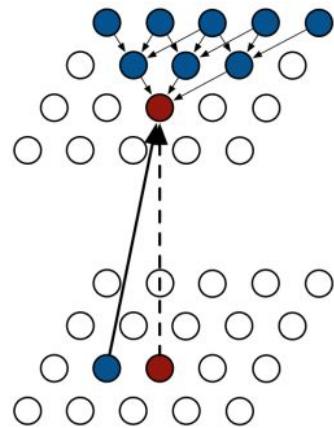
# Порождающие модели в deep learning



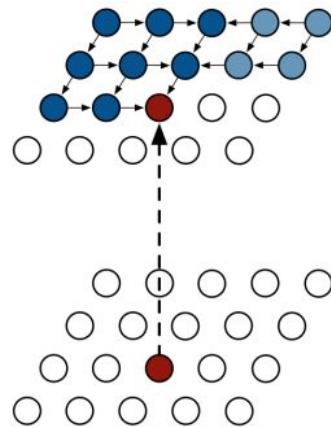
➤ Пример — **PixelCNN** и **PixelRNN** (van den Oord, 2016a; 2016b):



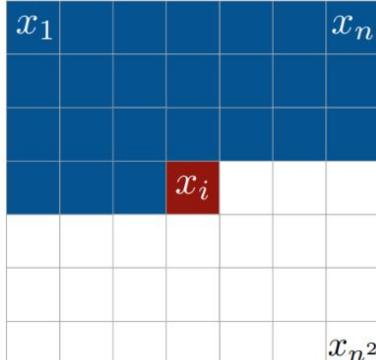
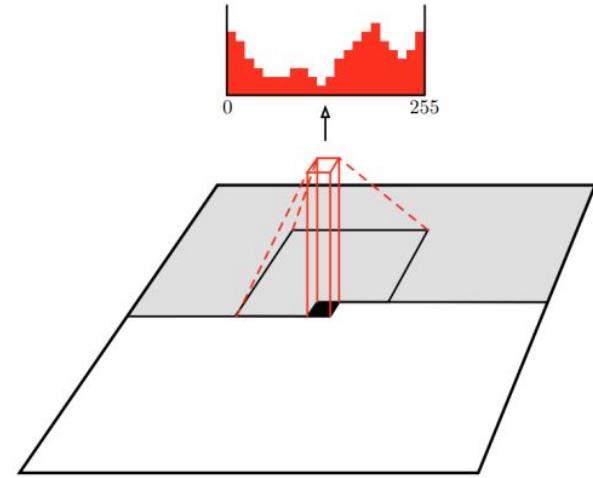
PixelCNN



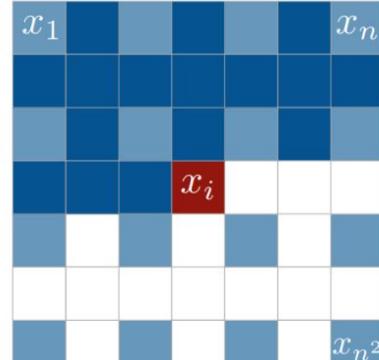
Row LSTM



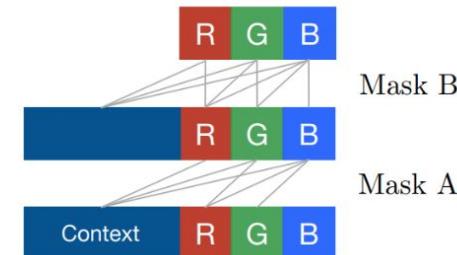
Diagonal BiLSTM



Context



Multi-scale context

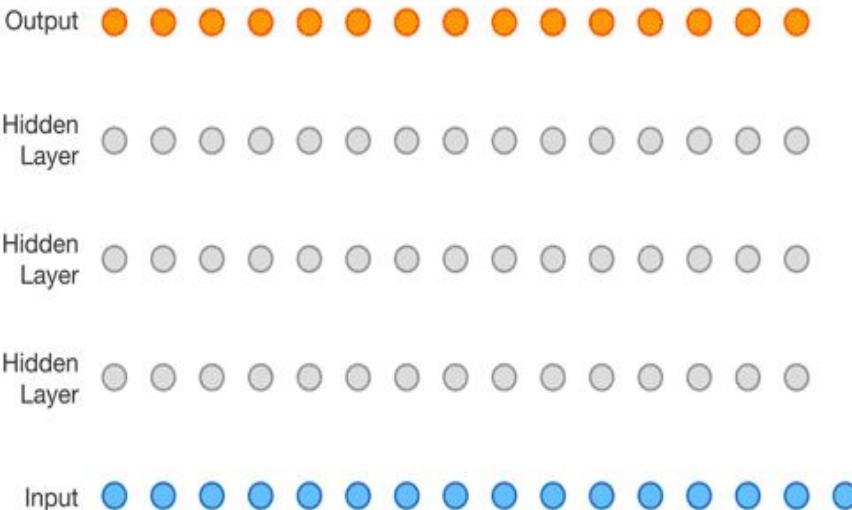
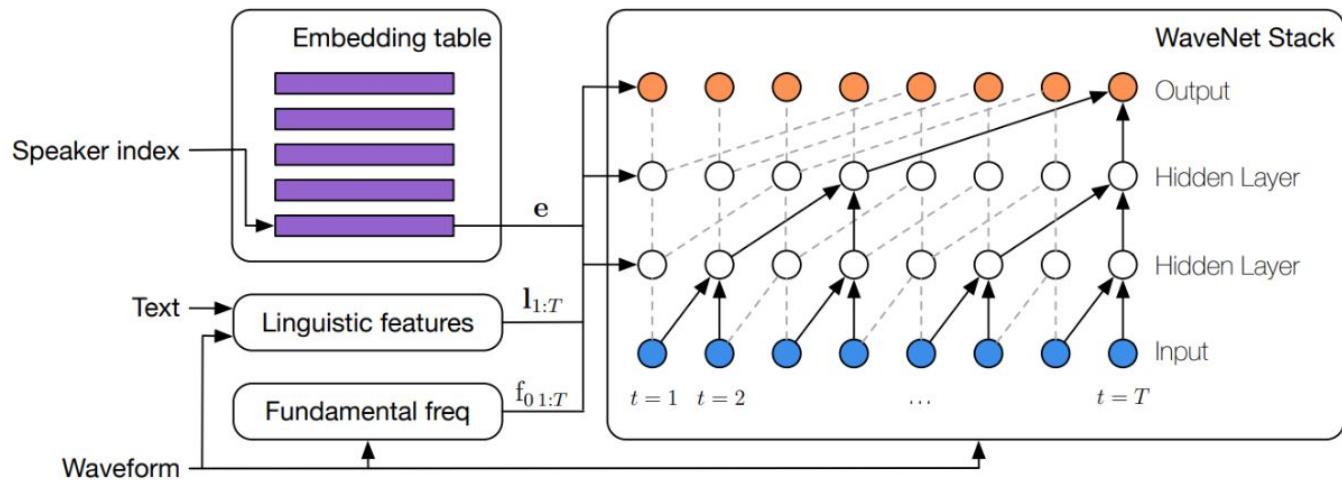


African elephant

# Порождающие модели в deep learning



- Другой пример – **WaveNet** (van den Oord et al., 2016; Chen et al., 2019)
- Оказалось, что можно речь порождать отсчёт за отсчётом

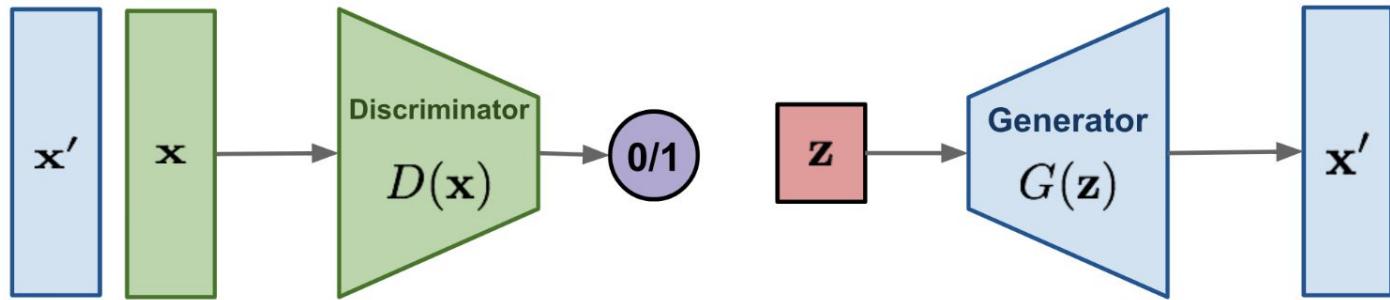


1 Second

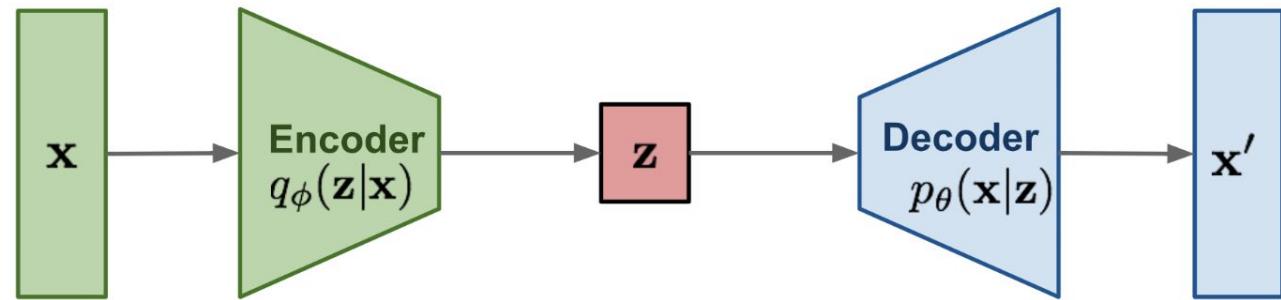
# Flow-based models

- Идея моделей, основанных на потоках: давайте строить кодировщик как **обратимое преобразование**; тогда проще будет обучить  $p(\mathbf{x})$

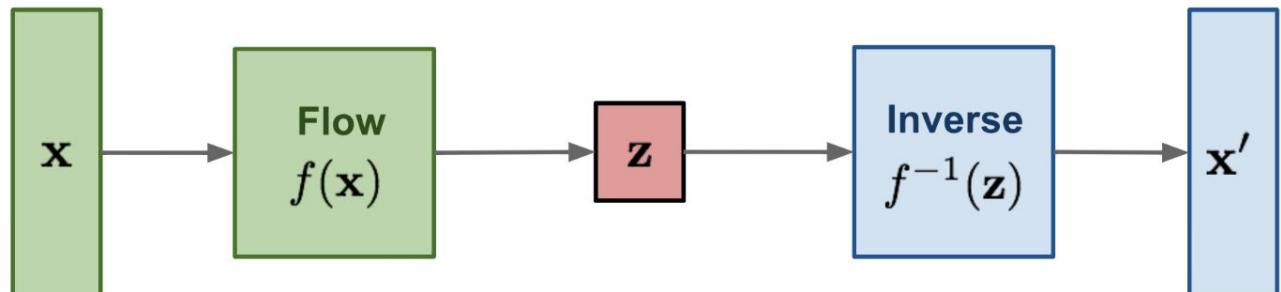
**GAN:** minimax the classification error loss.



**VAE:** maximize ELBO.



**Flow-based generative models:** minimize the negative log-likelihood



# Flow-based models



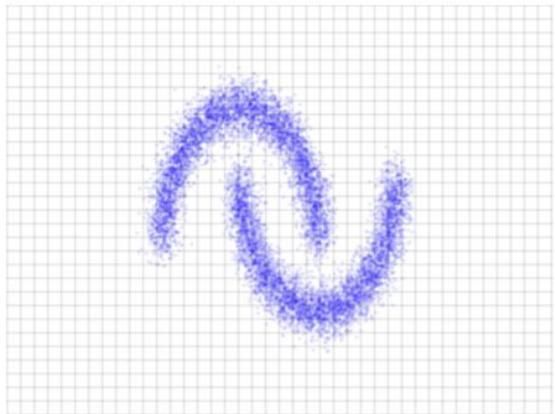
- Общий смысл происходящего примерно такой:

## Inference

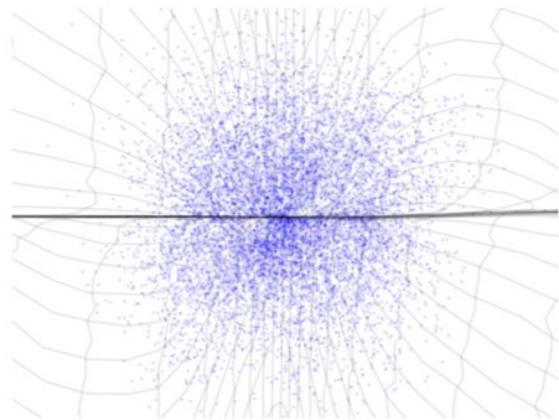
$$x \sim \hat{p}_X$$

$$z = f(x)$$

Data space  $\mathcal{X}$



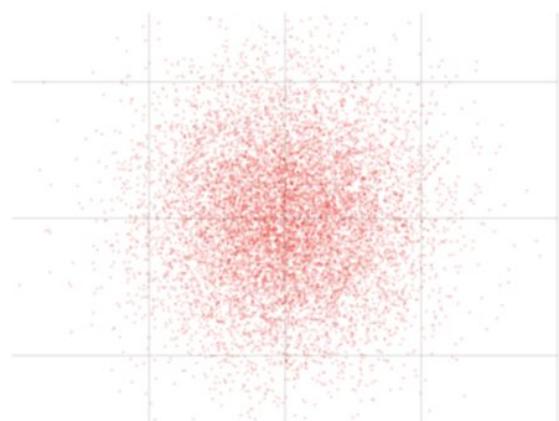
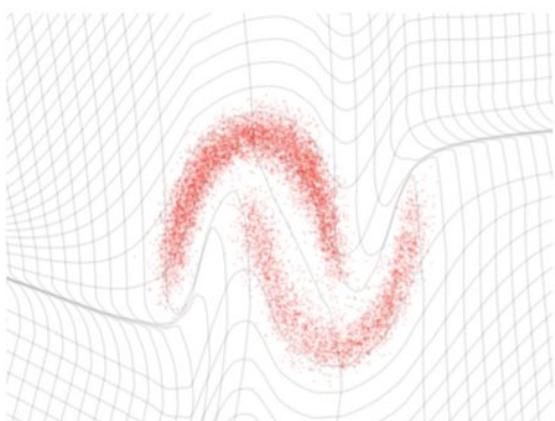
Latent space  $\mathcal{Z}$



## Generation

$$z \sim p_Z$$

$$x = f^{-1}(z)$$



# Flow-based models



➤ Самая главная идея:

- дано простое распределение  $\mathbf{z} \sim \pi(\mathbf{z})$ , а мы хотим построить новую случайную величину функцией  $\mathbf{x} = f(\mathbf{z})$ ; какое будет  $p(\mathbf{x})$ ?
- **теорема о замене переменных:**

$$\int p(x)dx = \int \pi(z)dz = 1$$

$$p(x) = \pi(z) \left| \frac{dz}{dx} \right| = \pi(f^{-1}(x)) \left| \frac{df^{-1}}{dx} \right| = \pi(f^{-1}(x)) |(f^{-1})'(x)|$$

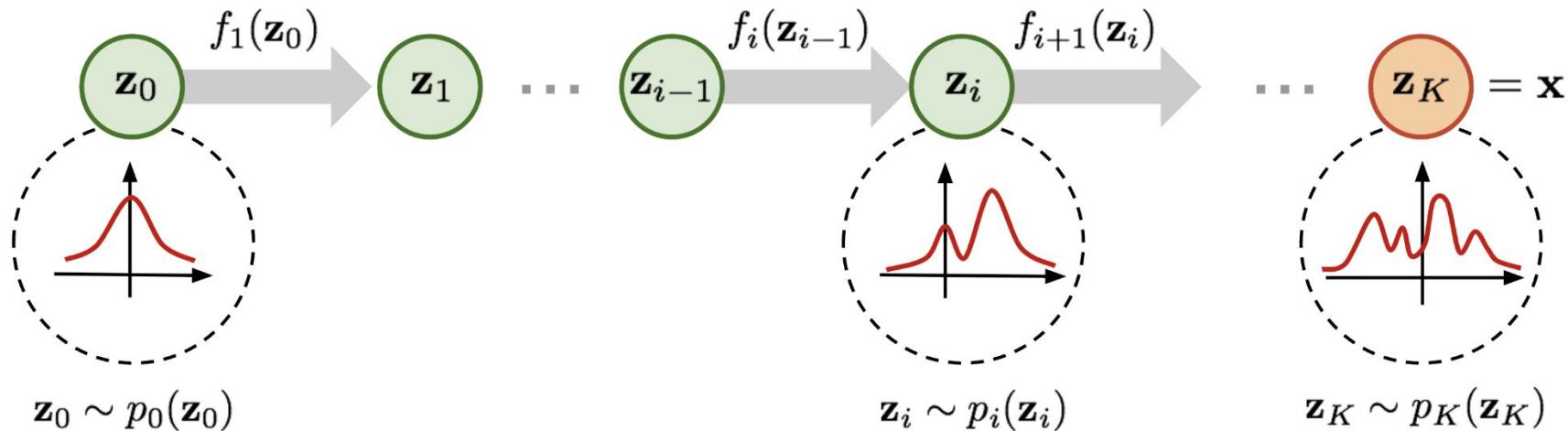
- а в многомерной версии появляется **якобиан** (определитель матрицы производных):

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$

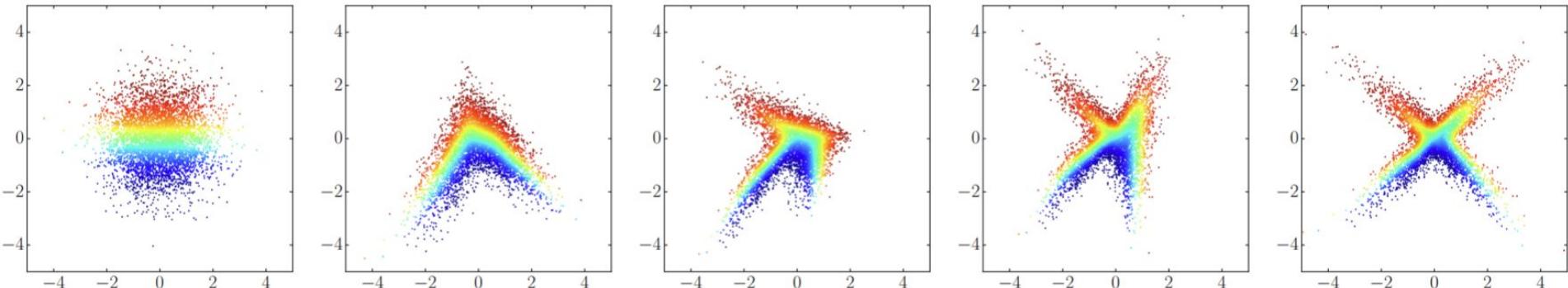
$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi(f^{-1}(\mathbf{x})) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|$$

# Flow-based models

- Набираем сложное преобразование как композицию простых:

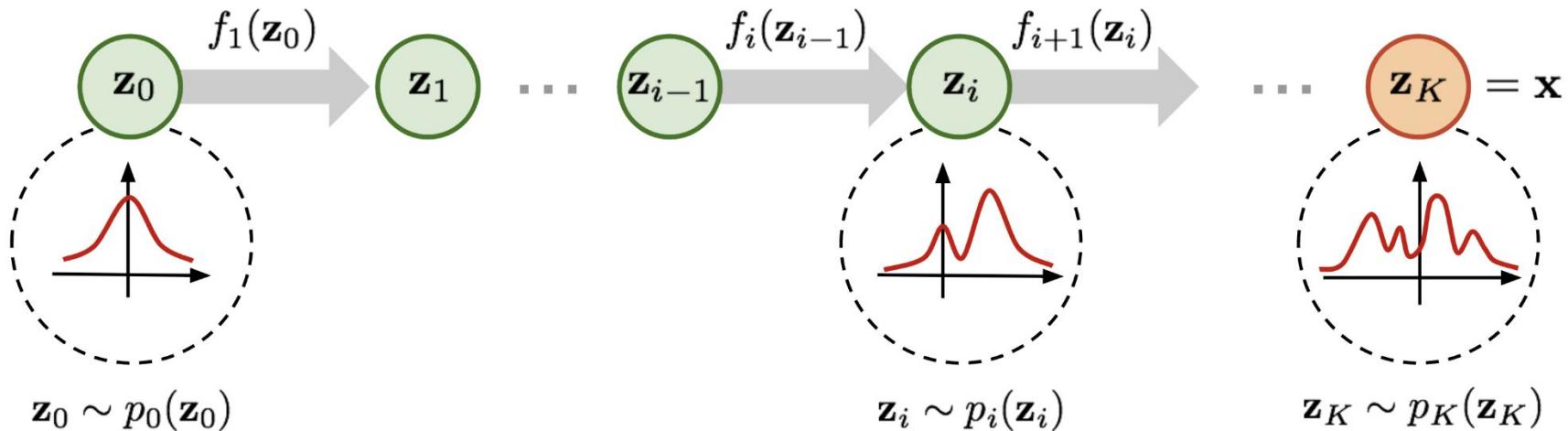


- Это может быть очень выразительно:



# Flow-based models

- Набираем сложное преобразование как композицию простых:



- И теперь те же формулы, но применённые несколько раз:

$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$

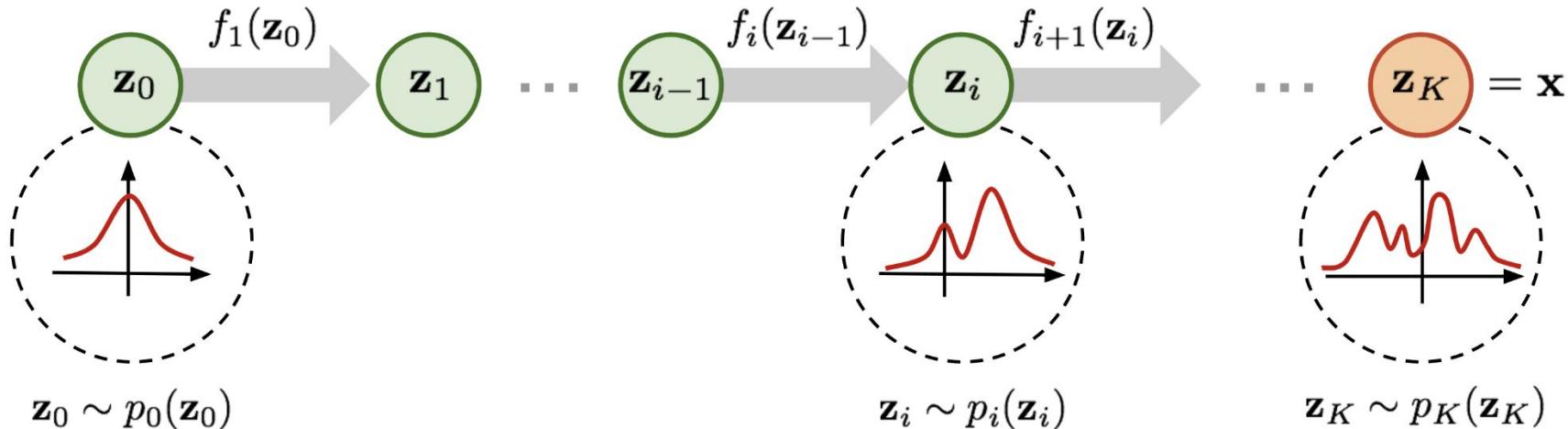
$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}),$$

$$\mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

# Flow-based models

- Набираем сложное преобразование как композицию простых:



- И теперь те же формулы, но применённые несколько раз:

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left( \frac{df_i}{d\mathbf{z}_{i-1}} \right)^{-1} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1}$$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|$$

# Flow-based models

- $\mathbf{z}_i$  — **поток** (flow), вся цепь — **нормализующий поток** (normalizing flow):

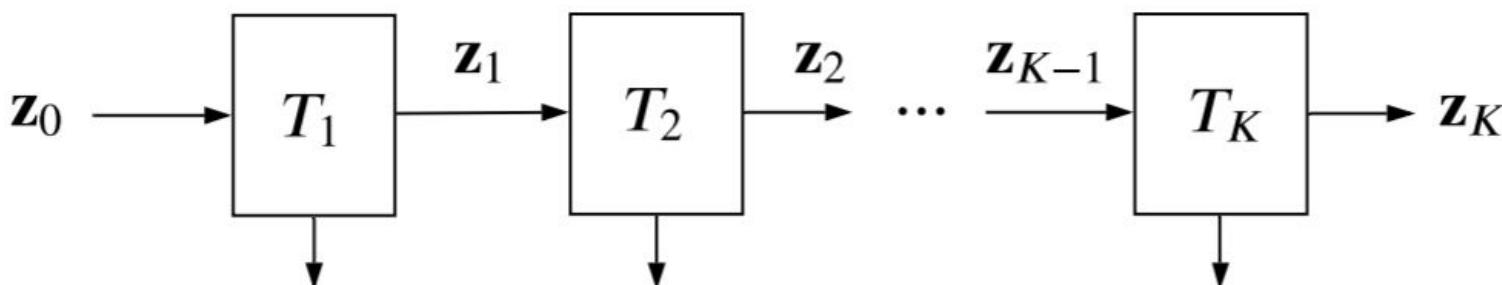
$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$$

$$\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) = \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right|$$

$$= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right|$$

= ...

$$= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^K \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|$$



$$\log | \det J_{T_1}(\mathbf{z}_0) | + \log | \det J_{T_2}(\mathbf{z}_1) | + \cdots + \log | \det J_{T_K}(\mathbf{z}_{K-1}) | = \log | \det J_T(\mathbf{z}_0) |$$

# Flow-based models



- И обучать можно просто максимизацией правдоподобия!

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

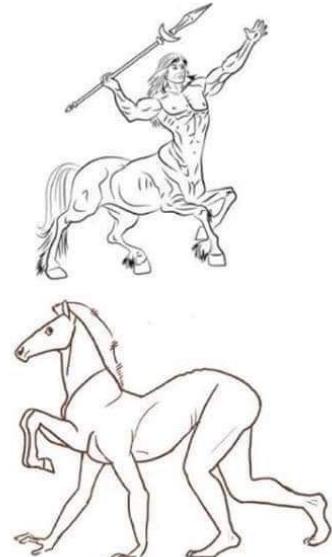
- Получается, что нам нужны две вещи:

$$f(x)$$

- чтобы  $\mathbf{f}_i$  были **обратимы**
- чтобы **якобиан** можно было  
**легко подсчитать**

$$f^{-1}(x)$$

- Мы будем стараться строить преобразования так, чтобы это получалось само собой, но при этом оставлять достаточно выразительности



- Первый настоящий пример: в **RealNVP** (Dinh et al., 2017) каждая  $f: \mathbf{x} \rightarrow \mathbf{y}$  оставляет первые  $d$  координат на месте, а остальные меняет

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$

- Получается легко обратить и легко найти якобиан (матрица вообще диагональная получается):

$$\mathbf{x}_{1:d} = \mathbf{y}_{1:d}$$

$$\mathbf{x}_{d+1:D} = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d}))$$

$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d \times (D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \text{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix}$$

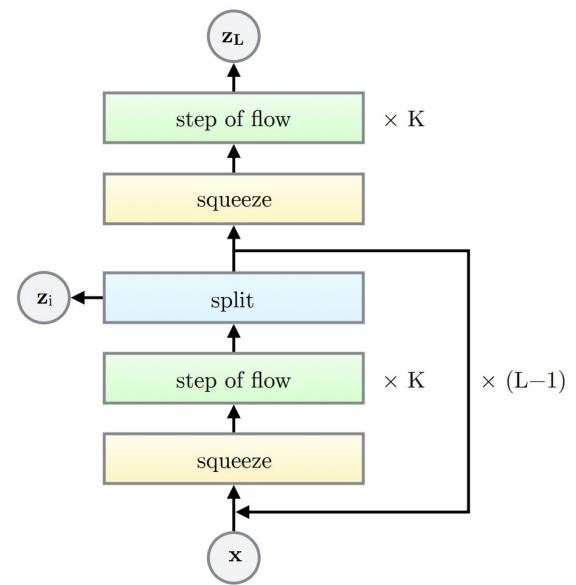
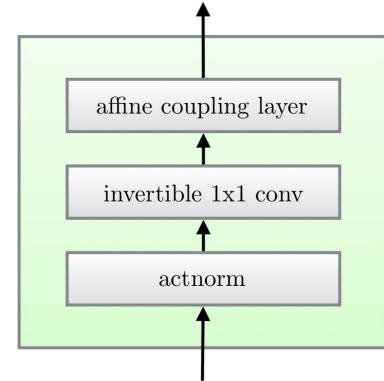
- Причём это легко обратить для любых функций  $s$  и  $t$ , то есть туда можно вставить сложные нейросети!



- Для 2016 года это выглядело очень даже хорошо (слева настоящие картинки из датасета, справа сэмплы из RealNVP):



- Следующий шаг: **GLOW** (Kingma, Dhariwal, 2018)
- Более общий способ добиться примерно того же самого (диагонального якобиана)

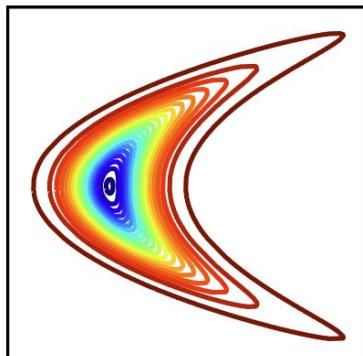


# Autoregressive flows

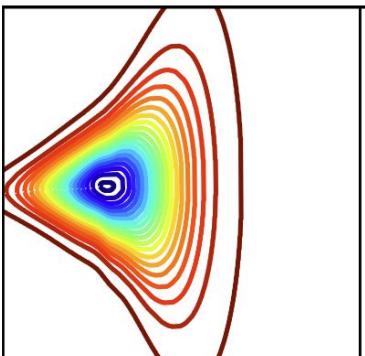
- Идея потоков отлично сочетается с идеей авторегрессивности
- **Авторегрессивные потоки** (autoregressive flows): такой же поток, где каждая размерность является функцией только от предыдущих
- **Masked Autoregressive Flow (MAF)** (Papamakarios et al., 2017)

$$p(\mathbf{x}) = \prod_{i=1}^D p(x_i | \mathbf{x}_{1:i-1}) \quad x_i \sim p(x_i | \mathbf{x}_{1:i-1}) = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1})$$

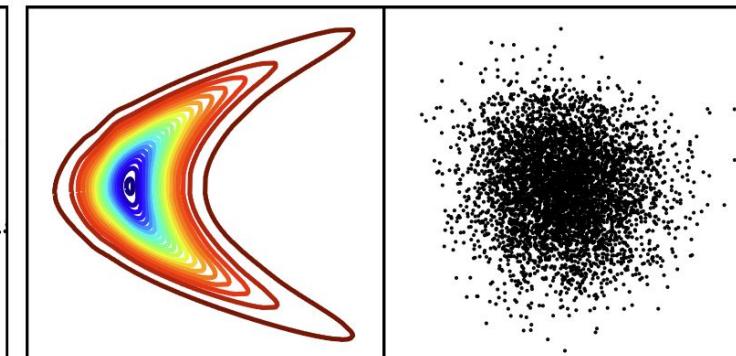
- Якобиан по определению диагональный, а функции  $\mu$  и  $\sigma$  можно делать сложными, то есть, опять же, нейросети подставлять



(a) Target density



(b) MADE with Gaussian conditionals



(c) MAF with 5 layers

# Autoregressive flows



- **Inverse Autoregressive Flow (IAF; Kingma et al., 2016)** делает наоборот:

$$z_i = \frac{x_i - \mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})} = -\frac{\mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})} + x_i \odot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}$$

- Выражаем  $\mathbf{x}$  авторегрессивно не по  $\mathbf{x}$ , а по  $\mathbf{z}$ :

$$\tilde{x}_i \sim p(\tilde{x}_i | \tilde{\mathbf{z}}_{1:i}) = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1}),$$

где  $\tilde{\mathbf{x}} = \mathbf{z}$ ,  $\tilde{p}(\cdot) = \pi(\cdot)$ ,  $\tilde{\mathbf{x}} \sim \tilde{p}(\tilde{\mathbf{x}})$

$\tilde{\mathbf{z}} = \mathbf{x}$ ,  $\tilde{\pi}(\cdot) = p(\cdot)$ ,  $\tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}})$

$$\tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1}) = \tilde{\mu}_i(\mathbf{x}_{1:i-1}) = -\frac{\mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})}$$

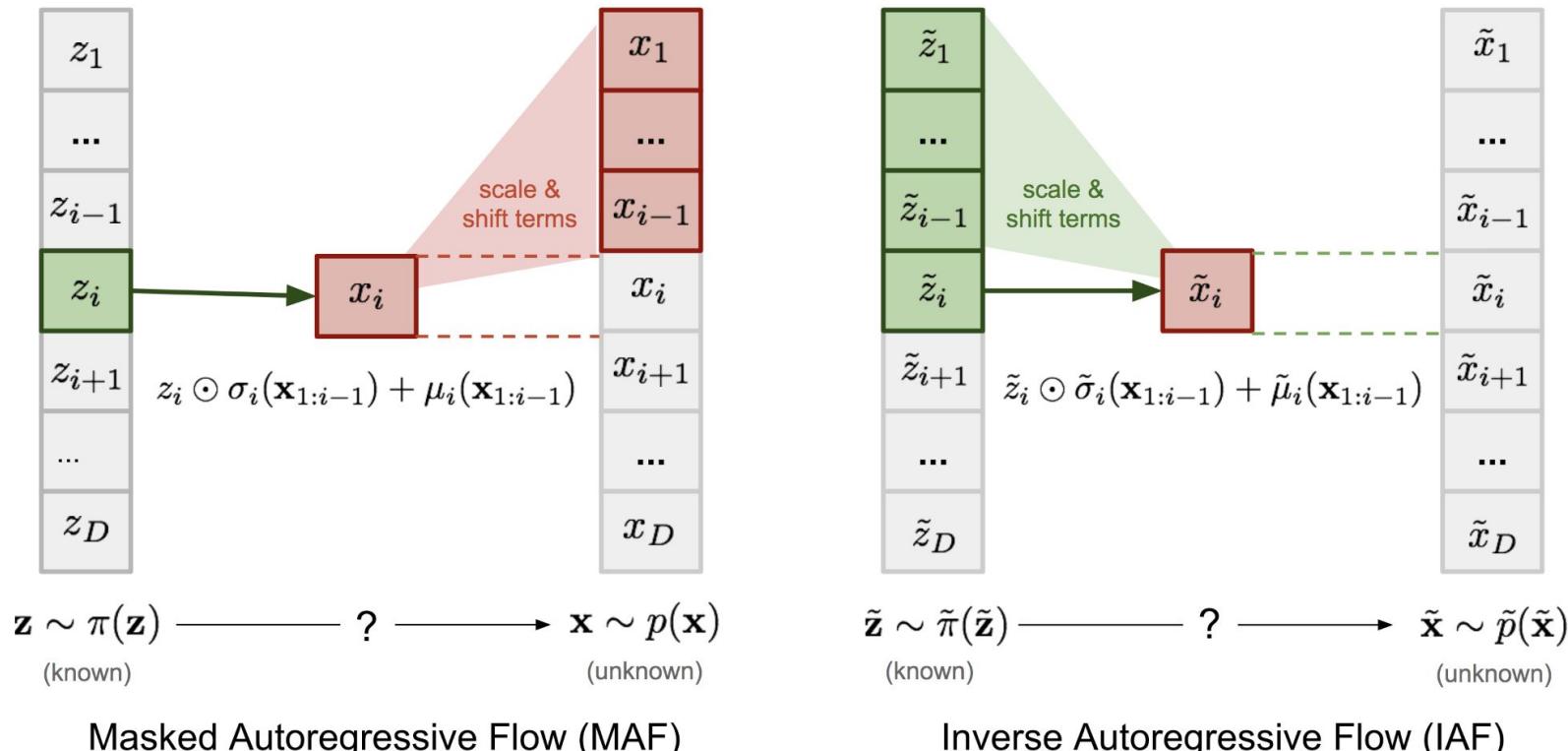
$$\tilde{\sigma}(\tilde{\mathbf{z}}_{1:i-1}) = \tilde{\sigma}(\mathbf{x}_{1:i-1}) = \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}$$

- Смысл тот же;  
в чём разница?



# Что быстро, а что медленно

Base distribution	Target distribution	Model	Data generation	Density estimation
MAF $\mathbf{z} \sim \pi(\mathbf{z})$	$\mathbf{x} \sim p(\mathbf{x})$	$x_i = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1})$	Sequential; slow	One pass; fast
IAF $\tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}})$	$\tilde{\mathbf{x}} \sim \tilde{p}(\tilde{\mathbf{x}})$	$\tilde{x}_i = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1})$	One pass; fast	Sequential; slow



# Best of both worlds



- Получается, что либо обучение, либо применение модели будет очень медленно
- Обучение нельзя (не дождёмся), и получается, что модель приходится авторегрессивно применять пиксель за пиксели
- А можно ли так, чтобы всё было быстро в обе стороны?



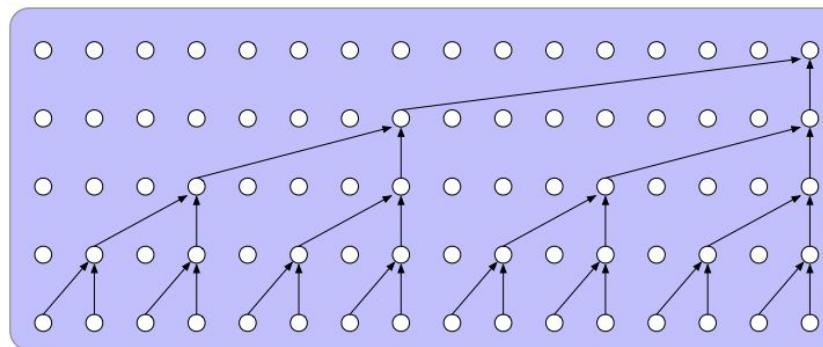
# Best of both worlds



- Можно! **Parallel WaveNet** (van den Oord et al., 2017): обучим обычный WaveNet (как MAF), а потом IAF-WaveNet на его результатах

## WaveNet Teacher

Linguistic features  $\dashrightarrow$

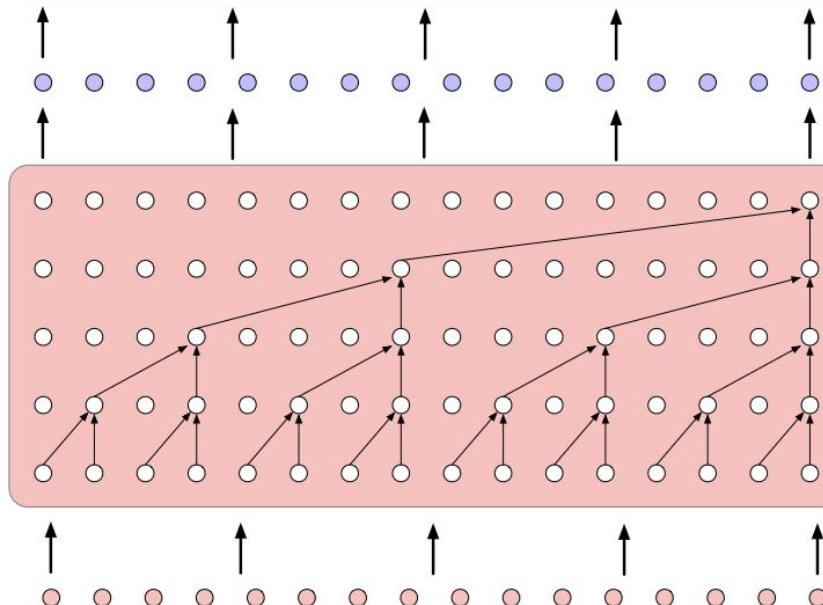


Teacher Output  
 $P(x_i|x_{<i})$

➤ В 1000 раз  
быстрее!

## WaveNet Student

Linguistic features  $\dashrightarrow$



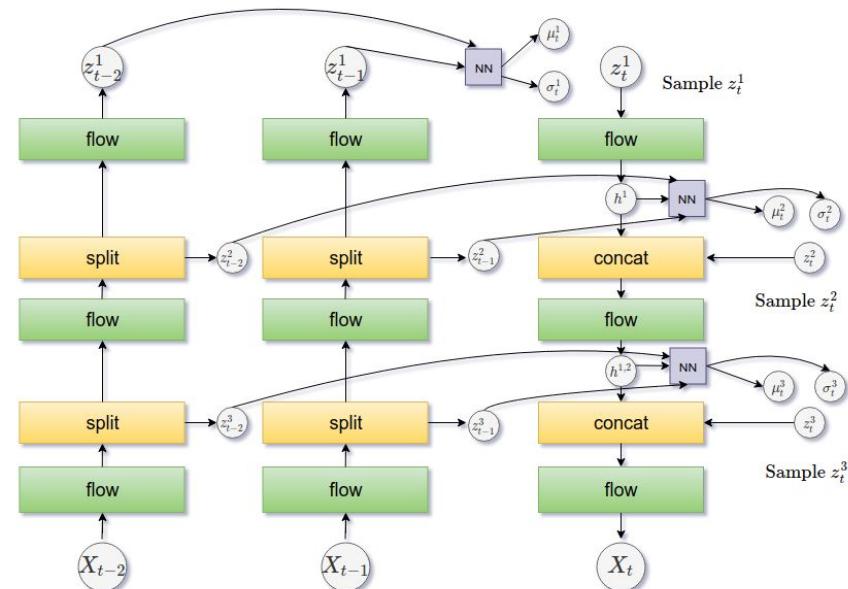
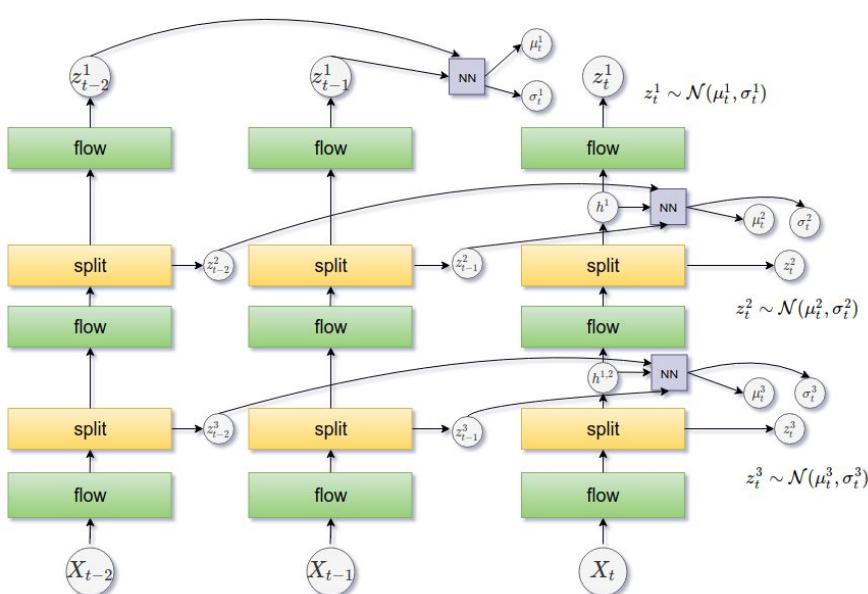
Generated Samples  
 $x_i = g(z_i|z_{<i})$

Student Output  
 $P(x_i|z_{<i})$



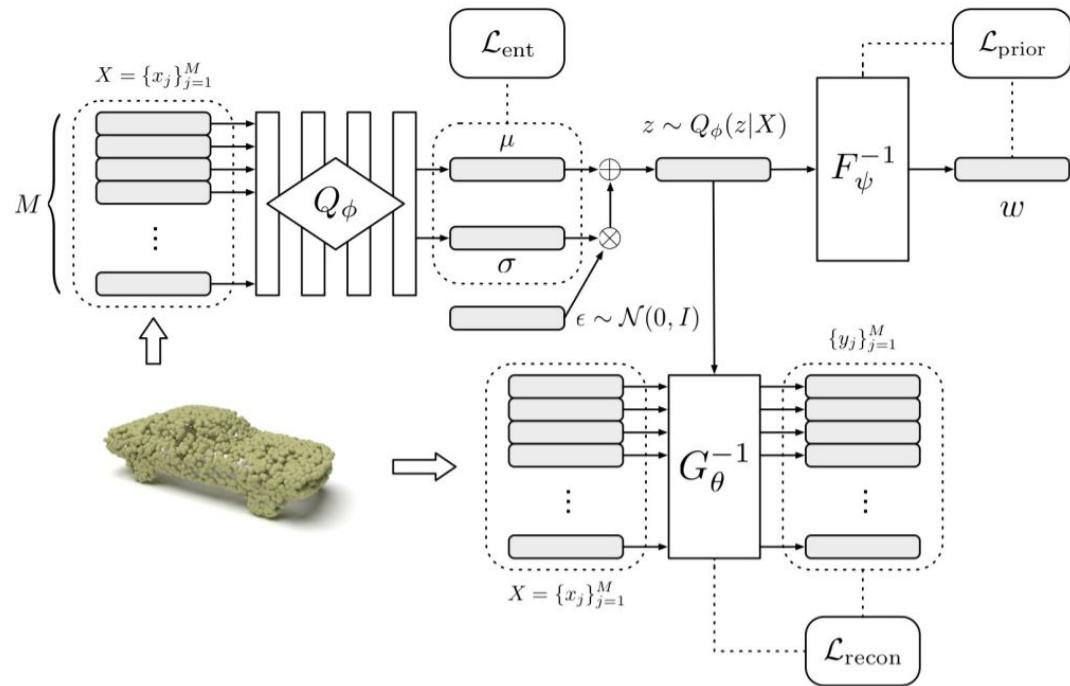
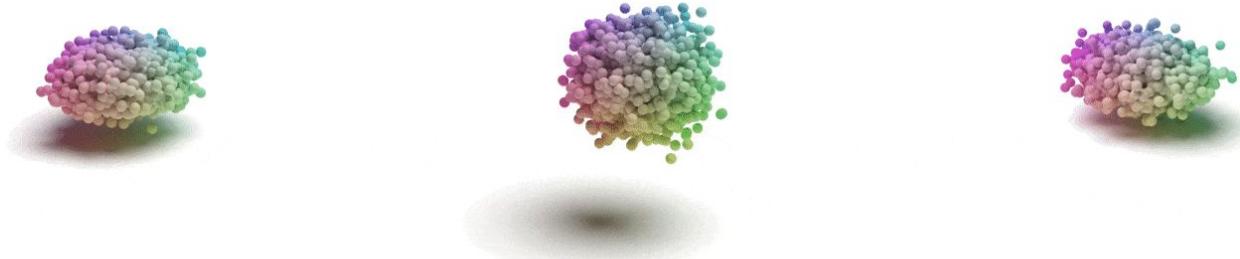
# VideoFlow

- **VideoFlow** (Kumar et al., 2019): порождаем видео по начальным кадрам, разнообразно, потому что моделируется всё распределение

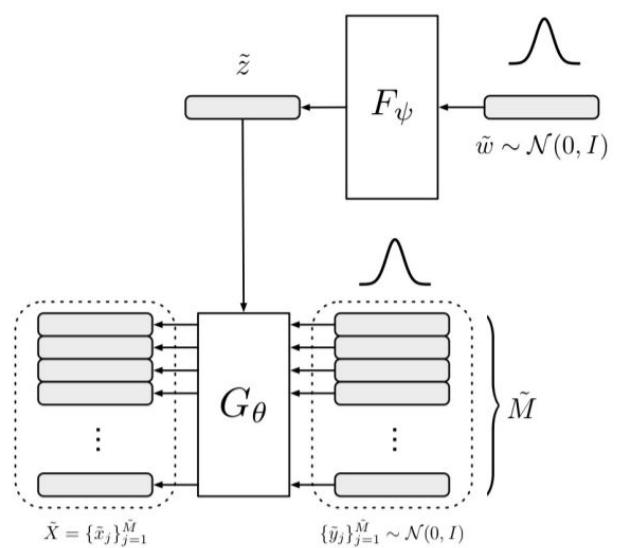


## ➤ PointFlow

(Yang et al.,  
2019):  
порождаем  
point clouds



(a) Training (Auto-encoding)

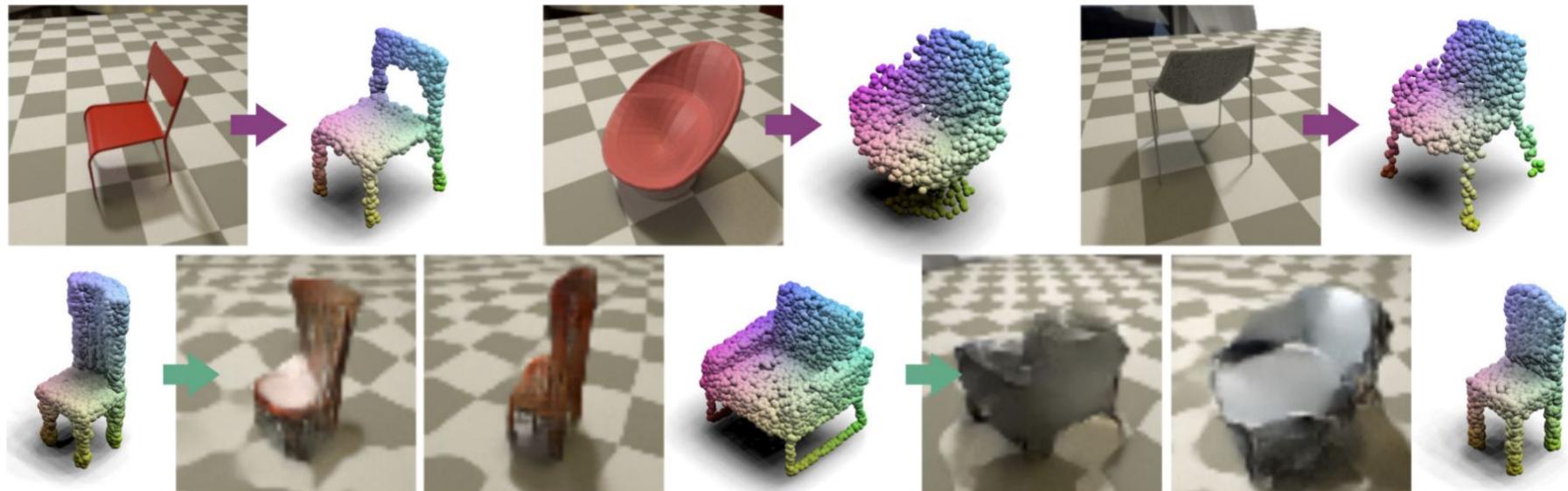
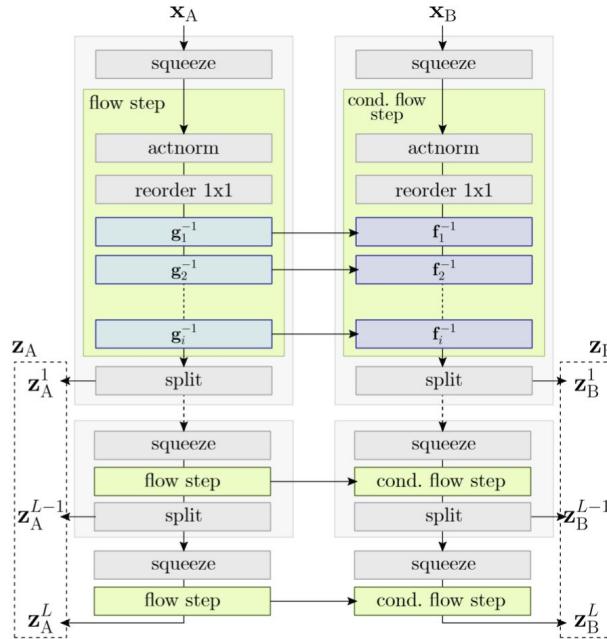


(a) Test (Sampling)

# CFlow



- **CFlow** (Pumarola et al., 2020):  
добавляем условие и получаем  
преобразование из картинок в  
point clouds и наоборот



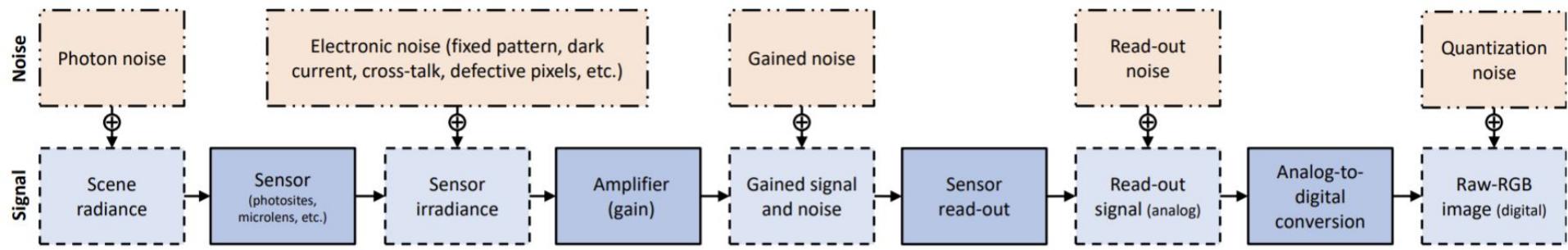
3D Reconstr.

Render

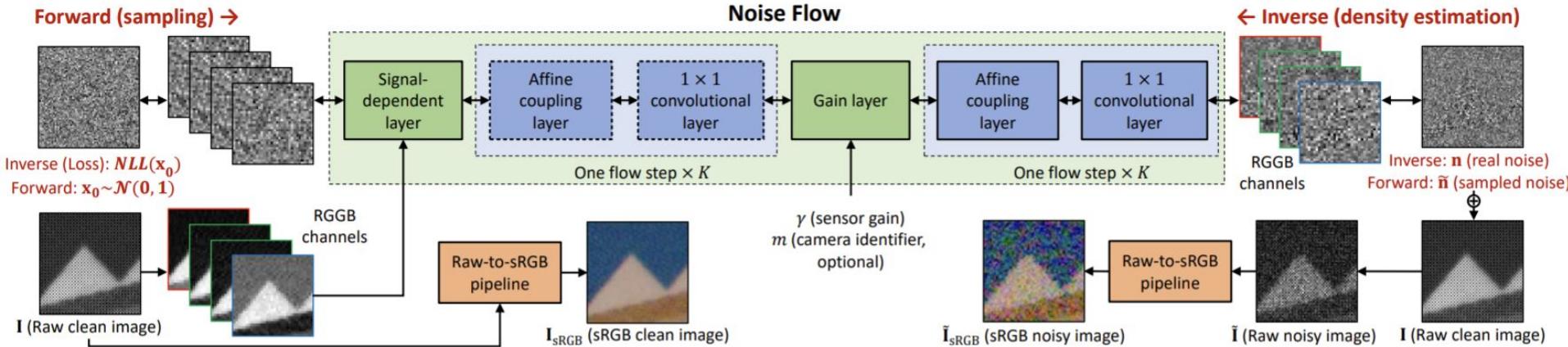
# NoiseFlow



- Интересное применение: чтобы улучшать картинки, надо понимать, какой шум вносит камера; это очень сложный процесс



- **NoiseFlow** (Abdelhamed et al., 2019): можно использовать это знание в построении потока, а потом обучить этот поток



# State of the art и итоги



- Модели, основанные на потоках, — очень интересная идея
- Уже есть несомненные успехи, но это всё достаточно ранний этап
- Можно запрыгнуть! Как мне кажется, очень интересны попытки соединить потоки и VAE (слишком технически сложно для нас сейчас): SurVAE ([Nielsen et al., 2020](#))
- Если вы хотите больше подробностей:
  - **Normalizing Flows: An Introduction and Review of Current Methods** ([Kobyzev et al., 2020](#))
  - **Normalizing Flows for Probabilistic Modeling and Inference** ([Papamakarios et al., 2021](#))
  - CVPR 2021 Tutorial  
**Normalizing Flows and Invertible Neural Networks in Computer Vision**  
<https://www.youtube.com/watch?v=8XufsgG066A> (4 часа! подробно и с самого начала)



Where androids dream of electric sheep

Спасибо за внимание!

neuromation.io

