ПОТОКИ В ГЛУБОКОМ ОБУЧЕНИИ

Сергей Николенко

СПбГУ — Санкт-Петербург

27 ноября 2025 г.

Random facts:

- 27 ноября 1095 г. Урбан II на Клермонском соборе по просьбе византийского императора Алексея I провозгласил Первый крестовый поход
- 27 ноября 1838 г. произошла битва при Сан-Хуан-де-Улуа в ходе Кондитерской войны между Мексикой и Францией; война началась по заявлению французского кондитера, якобы ограбленного мексиканскими мародёрами, Франция послала флот, чтобы вернуть долг, при бомбардировке Сан-Хуан-де-Улуа погибли более 60 человек, и президенту Мексики пришлось пообещать выплатить компенсацию Франции; впрочем, в итоге так и не выплатили
- 27 ноября 1895 г. Альфред Нобель подписал завещание, по которому бóльшая часть его состояния поступала в фонд Нобелевской премии
- 27 ноября — день авиакатастроф: в 1962 г. Boeing 707 врезался в гору при заходе на посадку под Лимой (97 погибших), в 1970 Douglas DC-8 выкатился со взлётной полосы и загорелся (47 погибших), в 1983 Boeing 747 под Мадридом зацепил несколько холмов и разрушился (181 погибших), а в 1989 г. в окрестностях Боготы террористы по указанию Пабло Эскобара взорвали Boeing 727 (110 погибших)
- 27 ноября 1992 г. была создана Высшая школа экономики

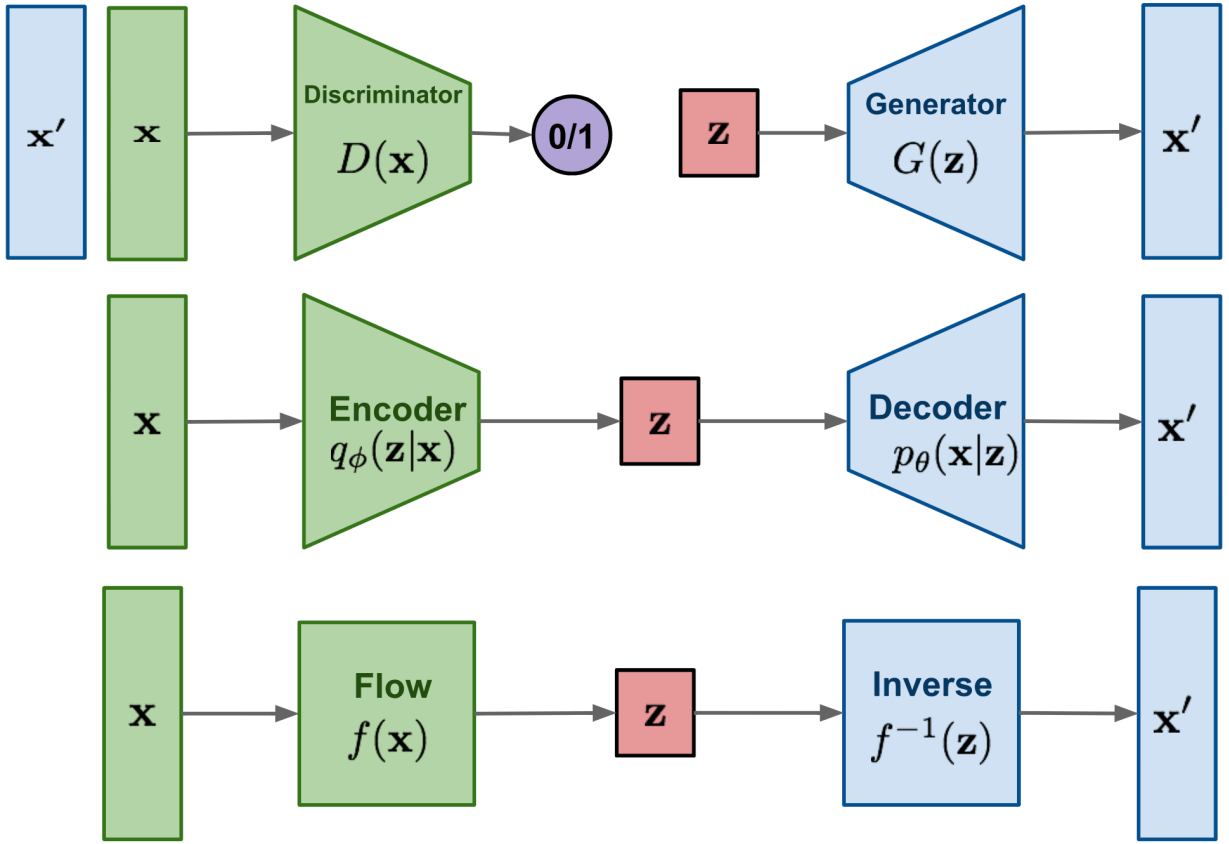
Нормализующие потоки

Идея

GAN: minimax the classification error loss.

VAE: maximize ELBO.

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



Нормализующие потоки

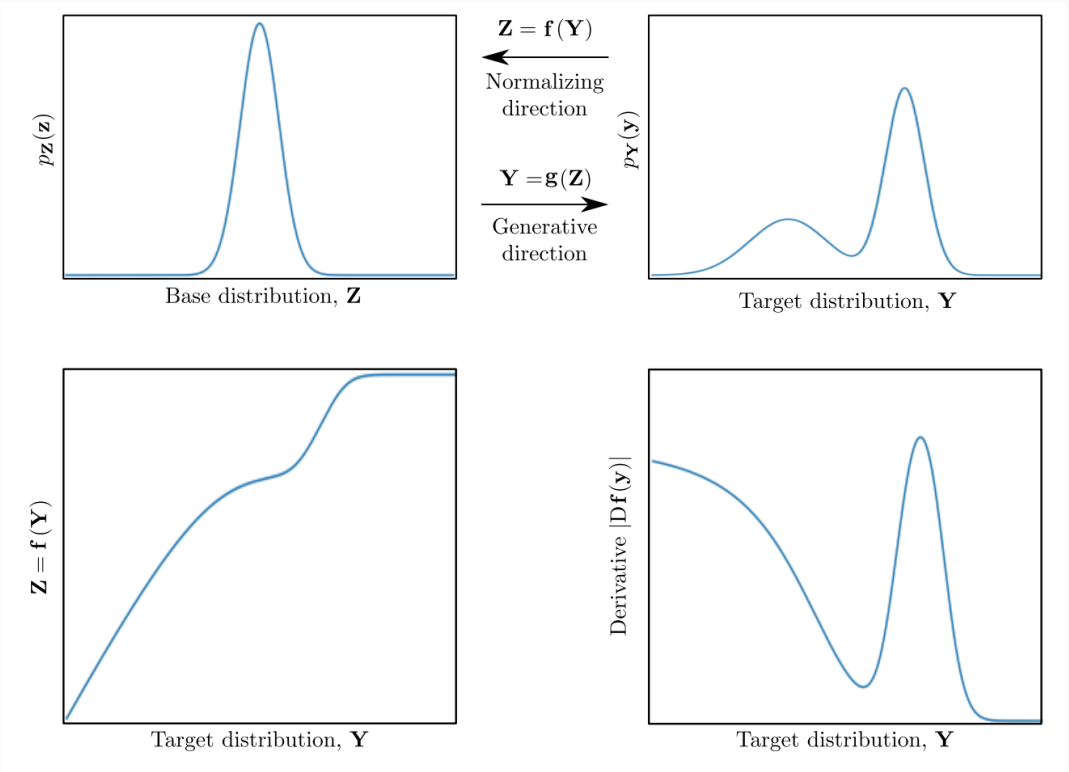
- И ещё один подход к порождающим моделям, совсем другой, но тоже очень интересный
- Нормализующие потоки (normalizing flows) – это метод построения сложных распределений из маленьких кусочков, выражаемых обратимыми преобразованиями (Rezende, Mohamed, 2015)
- Если $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{z} \sim q(\mathbf{z})$, и $f : \mathbb{R}^d \to \mathbb{R}^d$ – обратимая функция, то у случайной величины $y = f(\mathbf{z})$ будет плотность

$$q_y(\mathbf{z}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}.$$

Нормализующие потоки

- Если $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{z} \sim q(\mathbf{z})$, и $f : \mathbb{R}^d \to \mathbb{R}^d$ – обратимая функция, то у случайной величины $y = f(\mathbf{z})$ будет плотность

$$q_y(\mathbf{z}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}.$$



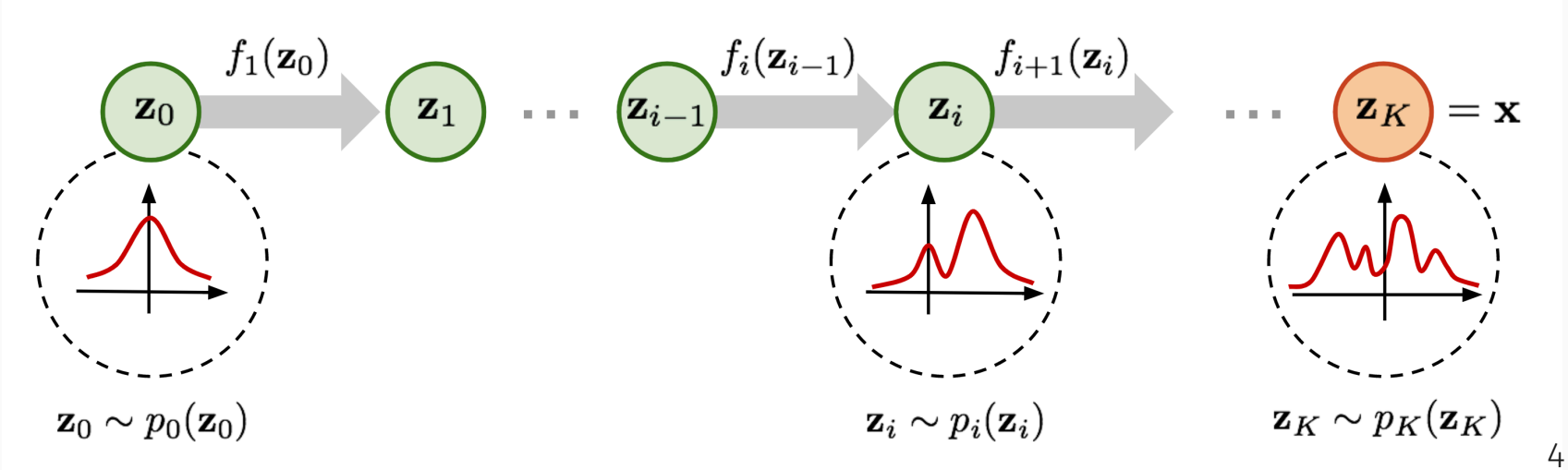
Нормализующие потоки

- И если сразу несколько взять таких функций подряд, то в общем тоже несложно посчитать, что получится:

$$\mathbf{z}_K = f_K \circ \ldots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0),$$

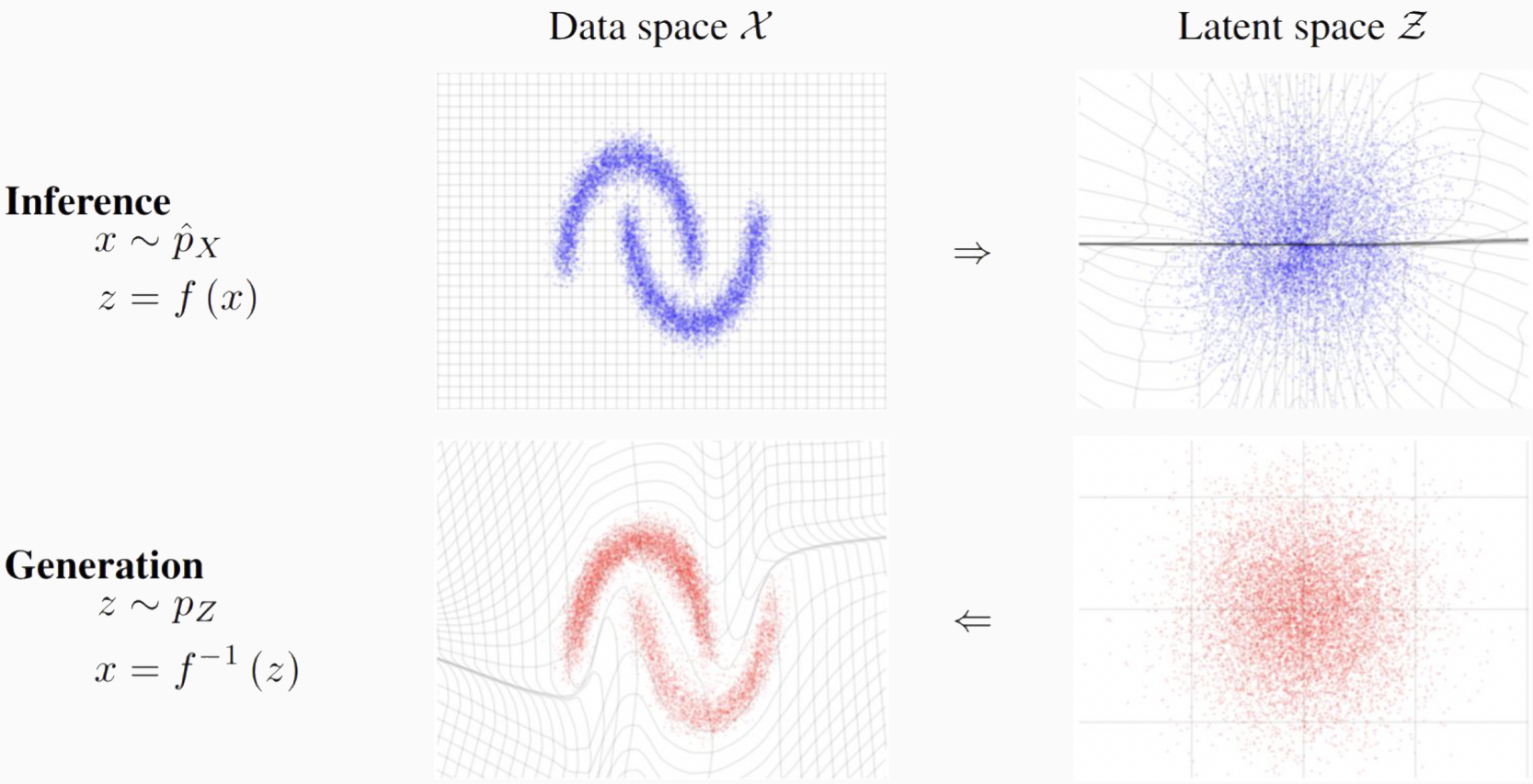
$$\mathbf{z}_K \sim q_K(\mathbf{z}) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1},$$

$$\log q_K(\mathbf{z}) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|.$$



Нормализующие потоки

- Так можно простое распределение (гауссиан, например) превратить в очень сложное:



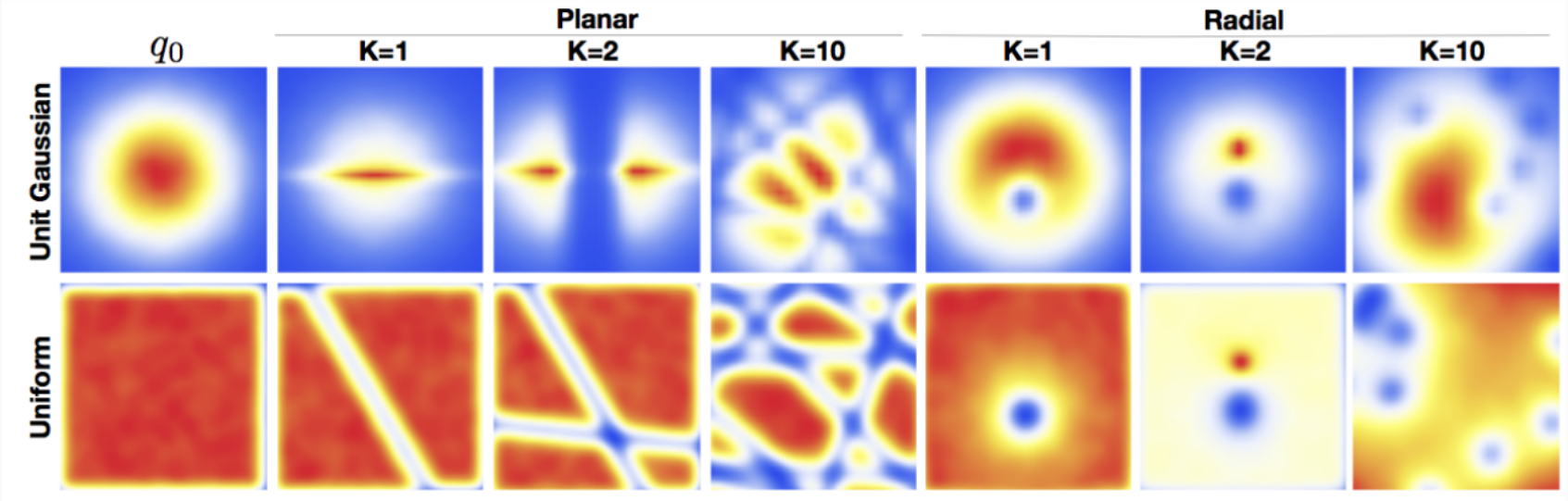
- Но мы ограничены преобразованиями, у которых легко подсчитать якобианы. Скоро увидим, насколько это серьёзное ограничение.

Нормализующие потоки

- Например, планарный поток (линейный) «разрезает» пространство гиперплоскостями:

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\top \mathbf{z} + b),$$

$$\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| = \left|1 + \mathbf{u}^\top \left(h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}\right)\right|.$$

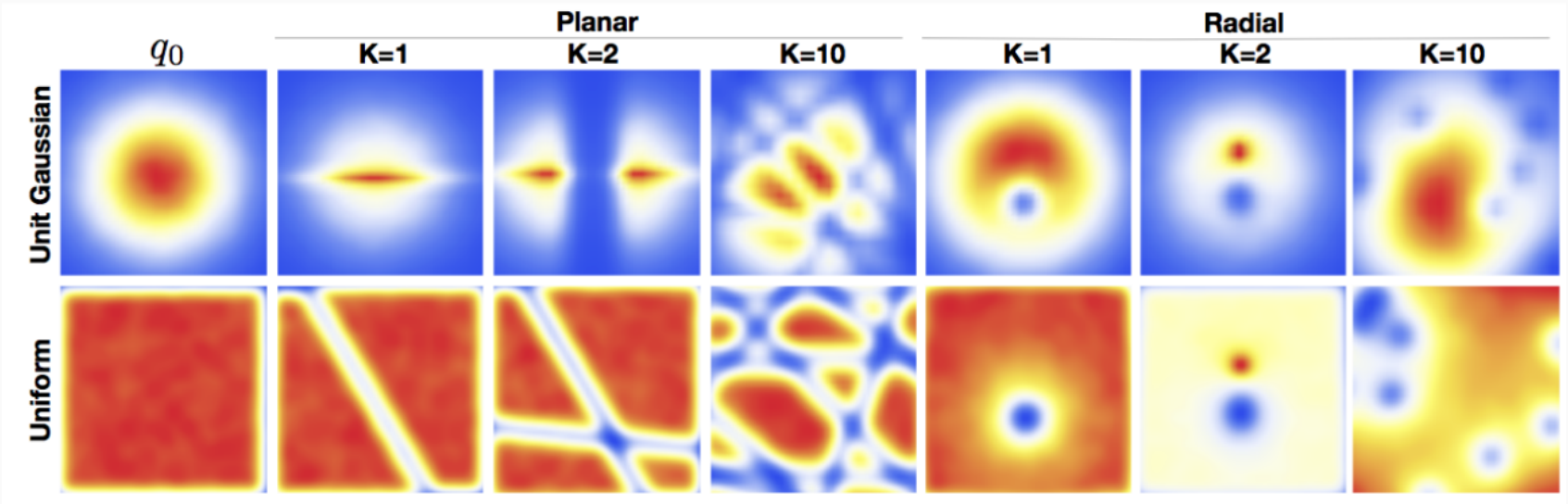


Нормализующие потоки

- Радиальный поток «разрезает» пространство сферами:

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)\,(\mathbf{z} - \mu)\,,$$

$$\text{где } r = \|\mathbf{z} - \mu\|_2\,,\ h(\alpha, r) = \frac{1}{\alpha + r}.$$



Нормализующие потоки

- А как сделать что-то поумнее? Нам надо уметь считать якобиан, т.е. определитель $\frac{\partial f}{\partial \mathbf{z}}$.
- Но если матрица треугольная, то определитель от неё несложно посчитать!
- Это в точности идея *авторегрессионных потоков* (autoregressive flows):

$$y_i = f(z_1, \dots, z_i), \quad \det \frac{\partial f}{\partial \mathbf{z}} = \prod_{i=1}^{d} \frac{\partial y_i}{\partial z_i}.$$

Нормализующие потоки

- Real Non-Volume Preserving Flows (R-NVP) – не слишком выразительные, но очень эффективные: для некоторых функция $\mu, \sigma : \mathbb{R}^k \to \mathbb{R}^{d-k}$ определим

$$\mathbf{y}_{1:k} = \mathbf{z}_{1:k}, \quad \mathbf{y}_{k+1:d} = \mathbf{z}_{k+1:d} \cdot \sigma(\mathbf{z}_{1:k}) + \mu(\mathbf{z}_{1:k}).$$

- Т.е. копируем первые k размерностей, потом для остальных применяем линейное преобразование; получается сначала единичная матрица, потом нижнетреугольная:

$$\det \frac{\partial \mathbf{y}}{\partial \mathbf{z}} = \prod_{i=1}^{d-k} \sigma_i(\mathbf{z}_{1:k}).$$

- Очень легко посчитать (всё параллельно), легко обратить даже для необратимых μ и σ, и можно использовать как параметризацию для VAE, а также как просто правдоподобие.

Нормализующие потоки

- Можно обобщить:

$$y_1 = \mu_1 + \sigma_1 z_1, \quad y_i = \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i.$$

- Более выразительная модель, якобиан такой же простой, $\prod_i \sigma(\mathbf{y}_{1:i-1})$, тоже можно легко обратить.
- Но теперь параллельно не вычисляется (по крайней мере в общем виде), преобразование по сути последовательное.

Нормализующие потоки

- Masked Autoregressive Flow (MAF):

$$p(\mathbf{x}) = \prod_{i=1}^{D} p\left(x_i | \mathbf{x}_{1:i-1}\right),$$

$$x_i \sim p\left(x_i | \mathbf{x}_{1:i-1}\right) = z_i \odot \sigma_i\left(\mathbf{x}_{1:i-1}\right) + \mu_i\left(\mathbf{x}_{1:i-1}\right).$$

- Функции μ и σ могут быть абсолютно произвольными! Например, параметризованными сложными нейросетями.
- Быстро работает для оценки плотности (обучения), но для сэмплирования (порождения) медленно, то есть авторегрессивно.

Нормализующие потоки

- Inverse Autoregressive Flow (IAF) работает по сути наоборот:

$$\tilde{x}_i \sim p\left(\tilde{x}_i | \mathbf{z}_{1:i}\right) = z_i \odot \sigma_i(\mathbf{z}_{1:i-1}) + \mu_i(\mathbf{z}_{1:i-1}).$$

- Теперь $\mathbf{y}$ зависит только от $\mathbf{z}$, и можно вычислить быстро (параллельно):

$$\tilde{\mathbf{x}} = \mathbf{z} \cdot \sigma(\mathbf{z}) + \mu(\mathbf{z}), \quad \det \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{z}} = \prod_{i=1}^{d} \sigma_i(\mathbf{z}),$$

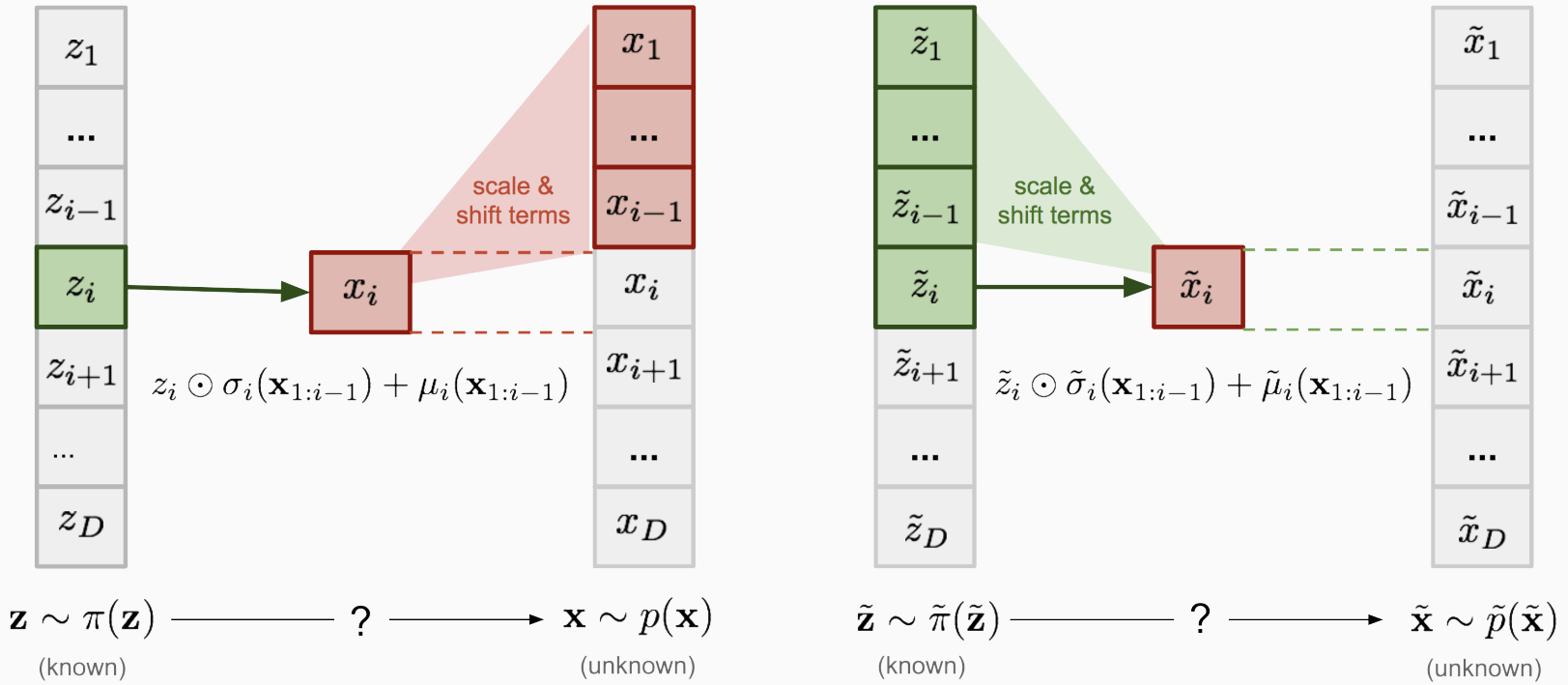
  и если собрать целую последовательность $f_K \circ \ldots \circ f_0(\mathbf{z})$, то

$$\log q_K(\mathbf{z}_K) = \log q(\mathbf{z}) - \sum_{k=0}^{K} \sum_{i=1}^{d} \log \sigma_{k,i}.$$

- Теперь сэмплировать легко, а плотность считать (обучать) сложно, нужно авторегрессивно.

Нормализующие потоки

- Получается такая конструкция:



Masked Autoregressive Flow (MAF)

Inverse Autoregressive Flow (IAF)

- В качестве всех этих μ и σ можно и нужно использовать нейронные сети.

Нормализующие потоки

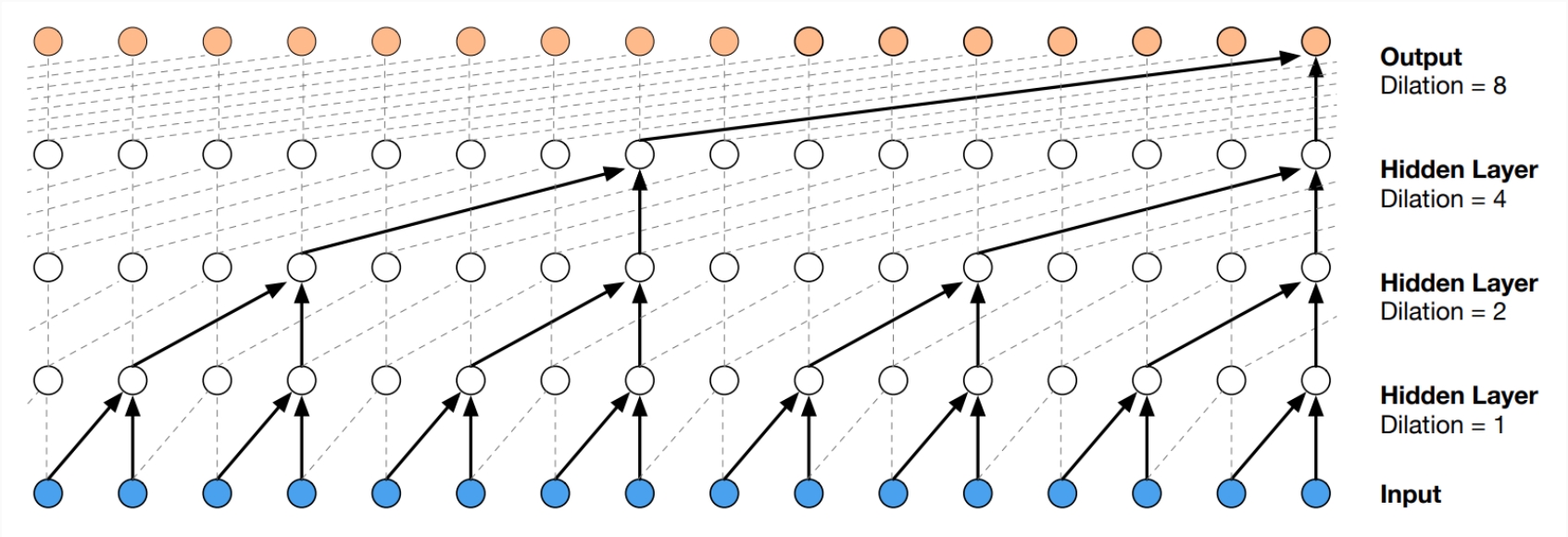
- Например, PixelRNN и MADE – это в точности нормализующие потоки.
- То есть MAF и IAF эффективны в разных направлениях:

	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

- Как это использовать?

Parallel WaveNet

- Следующий интересный шаг – Parallel WaveNet (van Oord, 2017), который ввёл метод параллельной дистилляции (parallel density distillation).



Parallel WaveNet

- WaveNet – это как раз модель, которая быстро (параллельно) обучается, но медленно (последовательно) сэмплирует.
- Чтобы быстро сэмплировать, было бы хорошо перейти к IAF:

$$\log p_X(\mathbf{x}) = \log p_Z(\mathbf{z}) - \log \left|\frac{\partial \mathbf{x}}{\partial \mathbf{z}}\right|, \quad \log \left|\frac{\partial \mathbf{x}}{\partial \mathbf{z}}\right| = \sum_t \log \frac{\partial f(\mathbf{z}_{\le t})}{\partial z_t}.$$

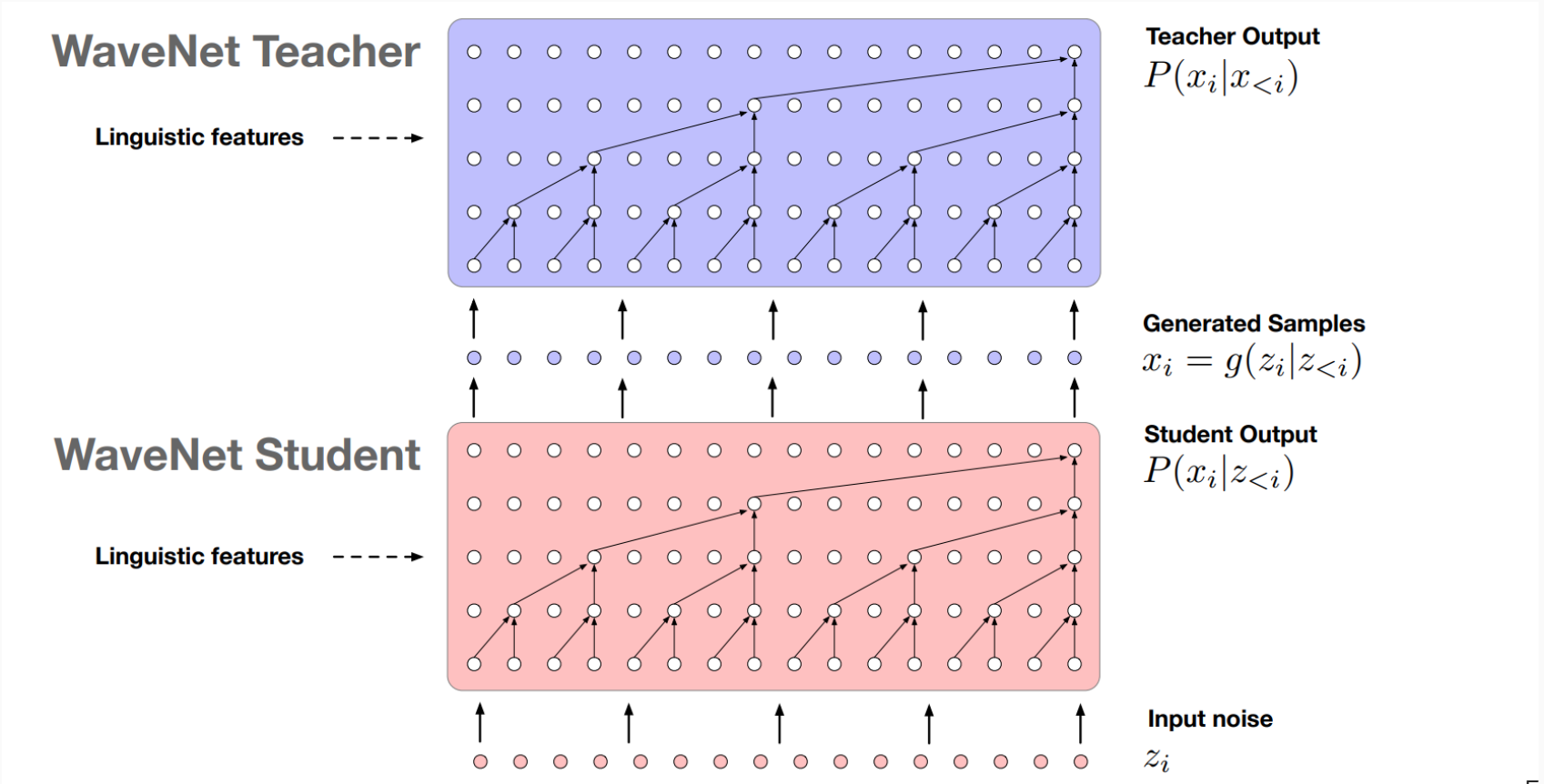
- Тогда сэмплирование – это просто $\mathbf{z} \sim q(\mathbf{z})$ (у WaveNet – логистическое), а дальше

$$x_t = z_t \cdot s(\mathbf{z}_{\le t}, \theta) + \mu(\mathbf{z}_{\le t}, \theta),$$

и для s и μ можно использовать те же свёрточные сети, что в WaveNet; можно в принципе одну, но оказывается получше, если композицию из четырёх сделать.

Parallel WaveNet

- А теперь главный трюк: обучать IAF очень долго, для WaveNet совсем непрактично. Поэтому parallel density distillation – давайте обучать студента $p_S(\mathbf{x})$ на основе учителя $p_T(\mathbf{x})$ с функцией ошибки $\mathrm{KL}\left(P_S \| P_T\right) = H(P_S, P_T) - H(P_S)$:



Parallel WaveNet

- Теперь и логистические распределения пригождаются:

$$H(P_S) = \mathbb{E}_{\mathbf{z}\sim L(0,1)}\left[\sum_{t=1}^{T} -\ln p_S(x_t \mid \mathbf{z}_{<t})\right] =$$

$$= \mathbb{E}_{\mathbf{z}\sim L(0,1)}\left[\sum_{t=1}^{T} \ln s(\mathbf{z}_{<t}, \theta)\right] + 2T,$$

  потому что энтропия $L(\mu, s)$ равна $\ln s + 2$

- А кросс-энтропия тоже нехитро:

$$H(P_S, P_T) = \int_{\mathbf{x}} p_S(\mathbf{x}) \ln p_T(\mathbf{x}) \mathrm{d}\mathbf{x} = \ldots =$$

$$= \sum_{t=1}^{T} \mathbb{E}_{p_S(\mathbf{x}_{<t})}\left[H\left(p_S(x_t \mid \mathbf{x}_{<t}), p_T(x_t \mid \mathbf{x}_{<t})\right)\right].$$

Parallel WaveNet

- Т.е. можно порождать из student $\mathbf{x}$, вычислять $p_T(x_t \mid \mathbf{x}_{<t})$ параллельно в teacher, а потом эффективно сэмплировать несколько x_t из $p_S(x_t \mid \mathbf{x}_{<t})$ для каждого t.
- Осталось добавить несколько дополнительных loss functions для student-сети (power loss для уровня мощности, perceptual loss на WaveNet-подобном классификаторе, contrastive loss при разных условиях).

Parallel WaveNet

- И получается такая же модель по качеству:

Method	Subjective 5-scale MOS
16kHz, 8-bit μ-law, 25h data:	
LSTM-RNN parametric [27]	3.67 ± 0.098
HMM-driven concatenative [27]	3.86 ± 0.137
WaveNet [27]	4.21 ± 0.081
24kHz, 16-bit linear PCM, 65h data:	
HMM-driven concatenative	4.19 ± 0.097
Autoregressive WaveNet	4.41 ± 0.069
Distilled WaveNet	4.41 ± 0.078

- Но авторегрессионный WaveNet сэмплирует 172 отсчёта в секунду, а дистиллированный – > 500000 отсчётов в секунду на том же железе...

GLOW

- Следующий шаг – давайте обобщать:

$$\mathbf{z} \sim p_\theta(\mathbf{z}), \quad \mathbf{x} = g_\theta(\mathbf{z}), \quad \mathbf{z} = f_\theta(\mathbf{x}) = g_\theta^{-1}(\mathbf{x}).$$

- И $f = f_1 \circ \ldots \circ f_K$, т.е.

$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{z}) + \sum_{i=1}^{K} \log \left\| \det \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\|,$$

  где $\mathbf{x} = \mathbf{h}_0$, $\mathbf{z} = \mathbf{h}_K$, и остальные в промежутке

- Даже ограничимся преобразованиями, где якобиан – треугольная матрица.

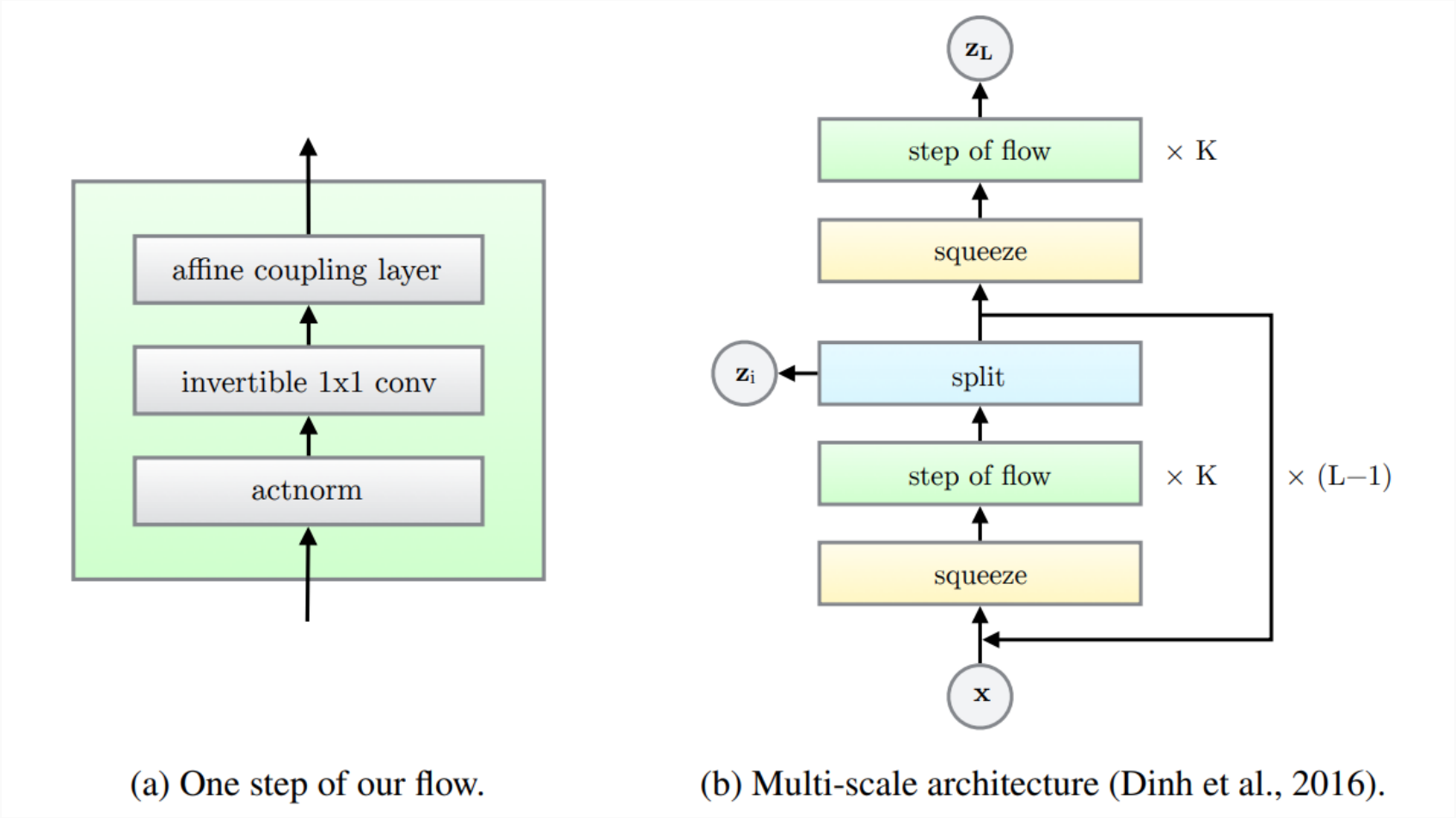
GLOW

- Но всё равно разнообразие довольно большое можно придумать:

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

GLOW

- И из этих компонентов составим модель; получится GLOW (Kingma, Dhariwal, 2018):



(a) One step of our flow.

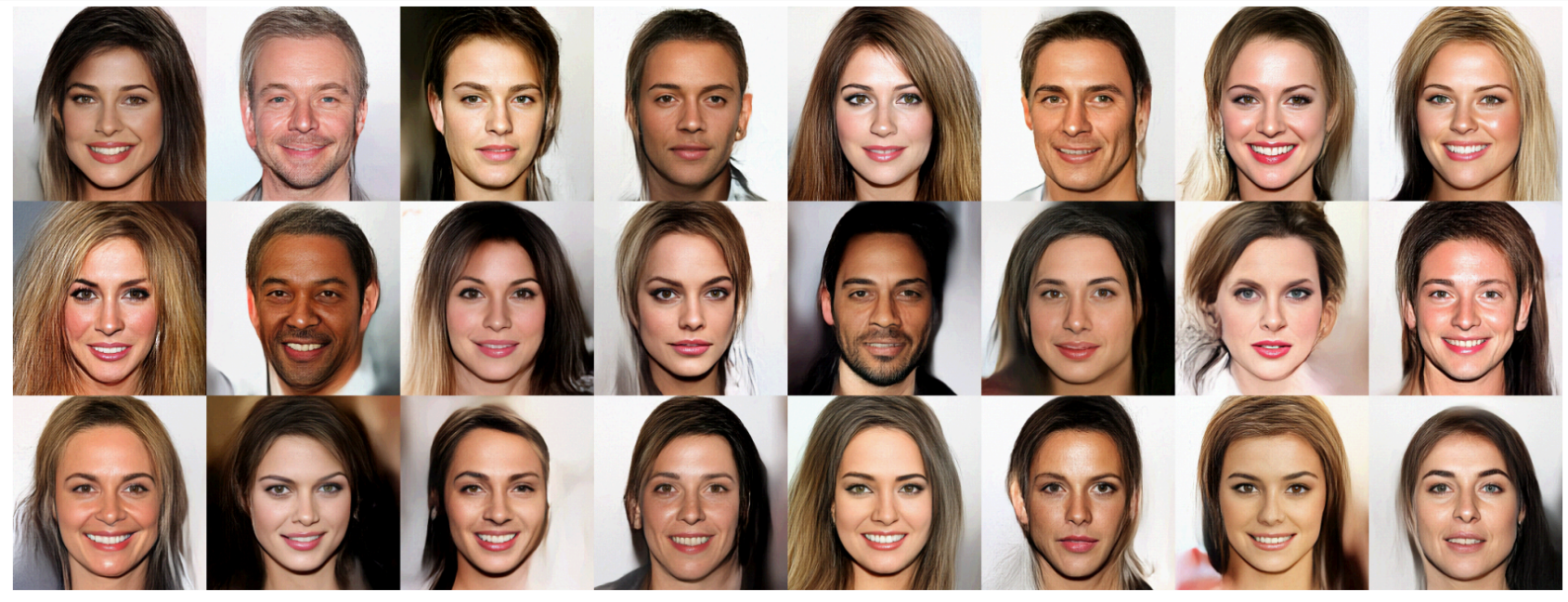
(b) Multi-scale architecture (Dinh et al., 2016).

GLOW

- И всё, это уже можно обучать делать неплохие, например, лица людей:

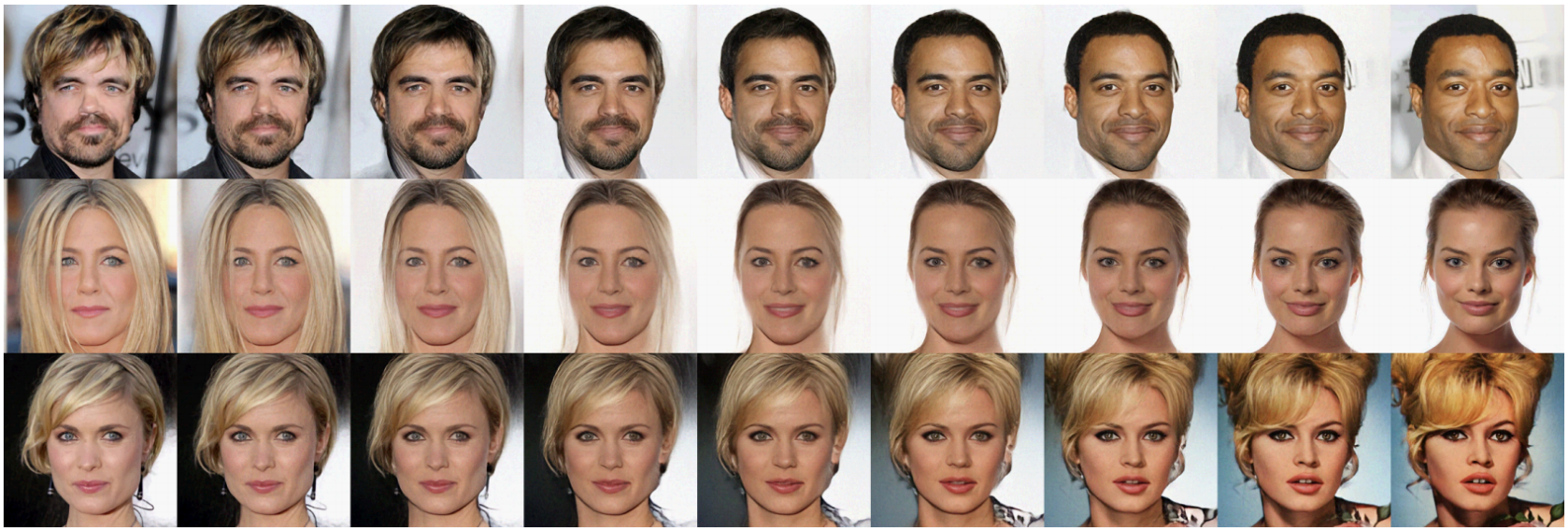
GLOW

- Вполне разнообразно:



GLOW

- По пространству признаков прогулки хорошие:



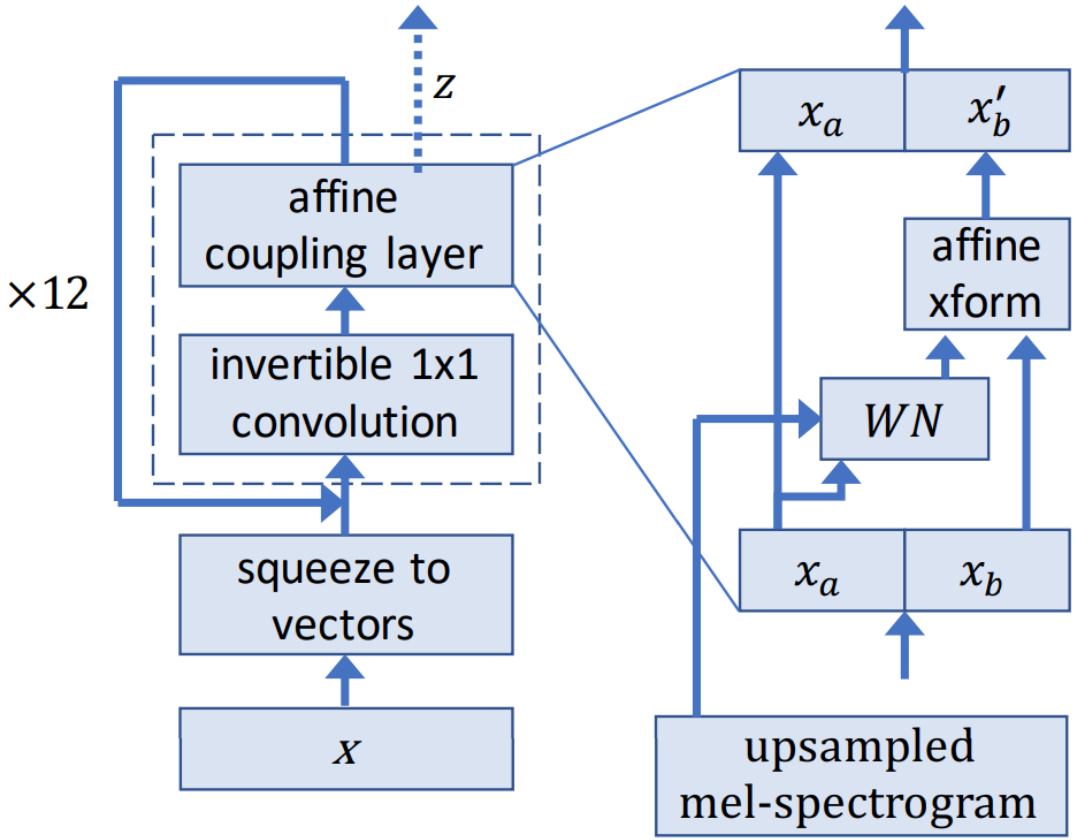
WaveGLOW

- А если совместить WaveNet и GLOW, получится

- WaveGlow генерирует речь из мел-спектрограммы (т.е. надо ещё отдельно её породить, но это нормально)

WaveGLOW

- Смысл в целом тот же, но идея в том, что можно сделать из необратимого WN (типа WaveNet) обратимое преобразование:



- И можно добраться до 2M отсчётов/сек, около 100x real-time

Flow++

- Flow++ (Ho et al., 2019) – несколько улучшений для потоковых моделей:
  - предыдущие модели используют деквантизацию, т.е. добавляют шум к дискретным данным, чтобы p_{model} не коллапсировала в смесь точечных распределений; тогда получается, что мы моделируем

$$P_{\text{model}}(\mathbf{x}) = \int_{[0,1]^D} p_{\text{model}}(\mathbf{x} + \mathbf{u}) \, \mathrm{d}\mathbf{u};$$

  - но равномерный шум не очень хорош для деквантизации: он просит нейросеть обучить равномерное распределение в кубиках вокруг набора точек, очень странно;
  - Flow++ предлагает использовать

Flow++

- Flow++ (Ho et al., 2019) – несколько улучшений для потоковых моделей:
  - Flow++ предлагает добавить распределение шума $q(\mathbf{u} \mid \mathbf{x})$ и оптимизировать вариационную оценку:

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log P_{\text{model}}(\mathbf{x})\right] = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\log \int_{[0,1]^D} p_{\text{model}}(\mathbf{x} + \mathbf{u}) \, \mathrm{d}\mathbf{u}\right] \geq$$

$$\geq \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\int_{[0,1]^D} q(\mathbf{u} \mid \mathbf{x}) \log \frac{p_{\text{model}}(\mathbf{x} + \mathbf{u})}{q(\mathbf{u} \mid \mathbf{x})} \mathrm{d}\mathbf{u}\right] =$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}, \mathbf{u} \sim q(\mathbf{u}|\mathbf{x})} \left[\log \frac{p_{\text{model}}(\mathbf{x} + \mathbf{u})}{q(\mathbf{u} \mid \mathbf{x})}\right];$$

  - и теперь $q(\mathbf{u} \mid \mathbf{x})$ можно взять тоже как потоковую порождающую модель $\mathbf{u} = q_{\mathbf{x}}(\epsilon)$, $\epsilon \sim N(0, \mathbf{I})$;

Flow++

- Flow++ (Ho et al., 2019) – несколько улучшений для потоковых моделей:
  - теперь

$$q\left(\mathbf{u} \mid \mathbf{x}\right) = p\left(q_{\mathbf{x}}^{-1}(\mathbf{u})\right)\left|\frac{\partial q_{\mathbf{x}}^{-1}}{\partial \mathbf{u}}\right|, \quad \text{и}$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log P_{\text{model}}\left(\mathbf{x}\right)\right] \geq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\epsilon}\left[\log \frac{p_{\text{model}}\left(\mathbf{x}+\mathbf{u}\right)}{p\left(\epsilon\right)\left|\frac{\partial q_{\mathbf{x}}}{\partial \epsilon}\right|^{-1}}\right],$$

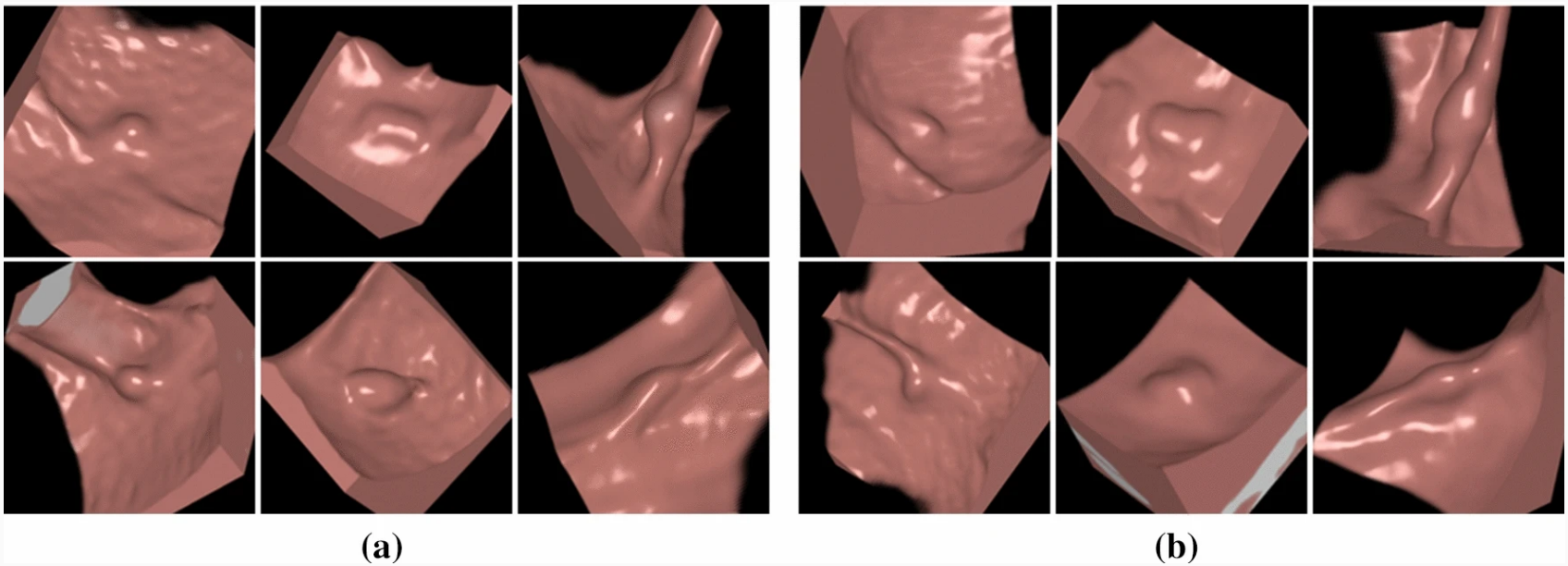
  и это теперь можно максимизировать совместно по p_{model} и q.

Flow++

- Flow++ (Ho et al., 2019) – несколько улучшений для потоковых моделей:
  - ещё улучшения для coupling layers;
  - улучшения для архитектуры нейросети, вычисляющей параметры этого преобразования (здесь трансформеры появились);
  - и в итоге лучше предыдущих работало на 64×64 картинках, но главное — идеи, они потом пошли дальше в потоковые модели.

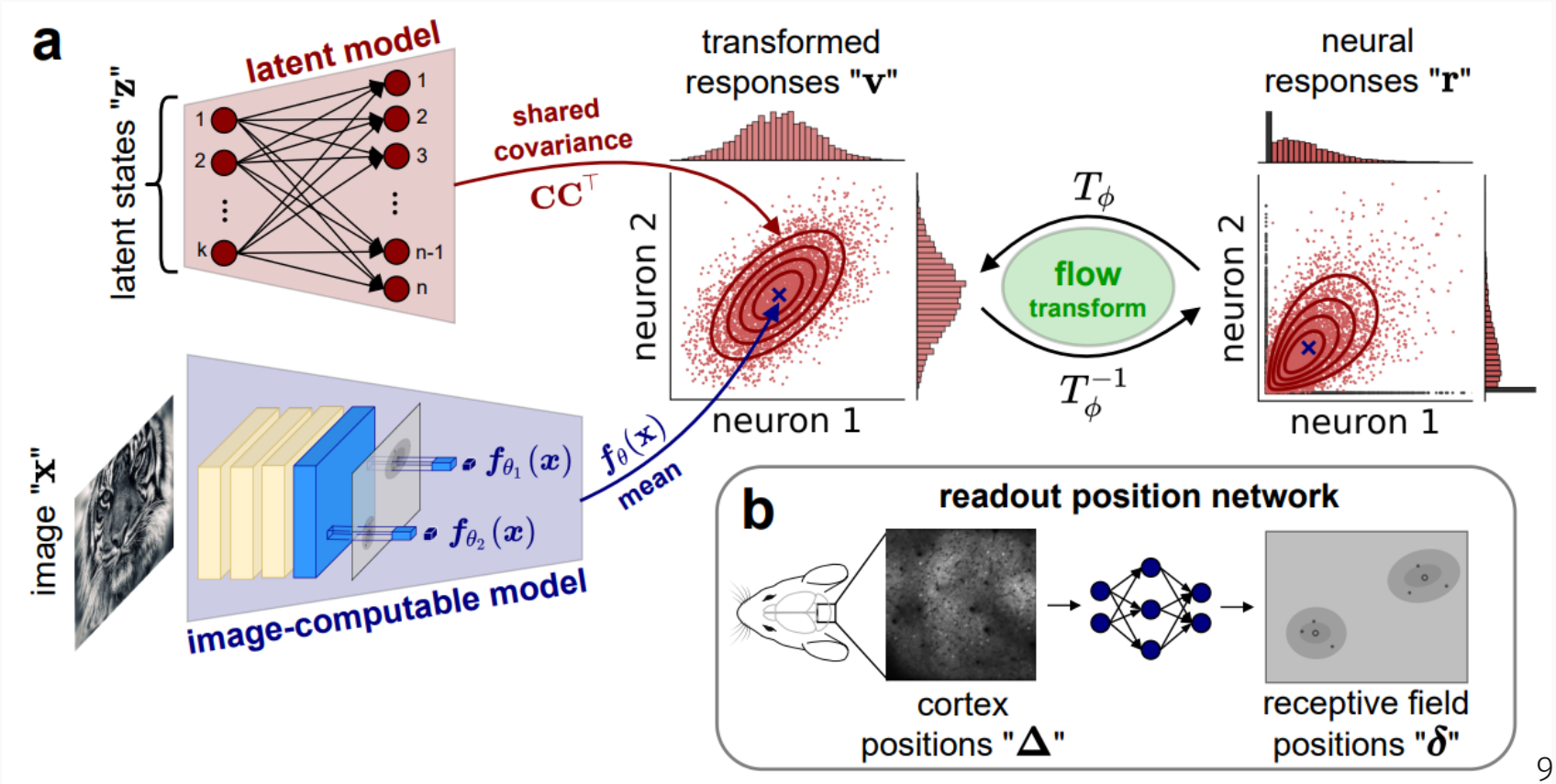
Применения потоковых моделей

- Потоковые модели активно используют для задач, где датасеты поменьше.
- (Uemura et al., 2020): 3D GLOW для моделирования полипов



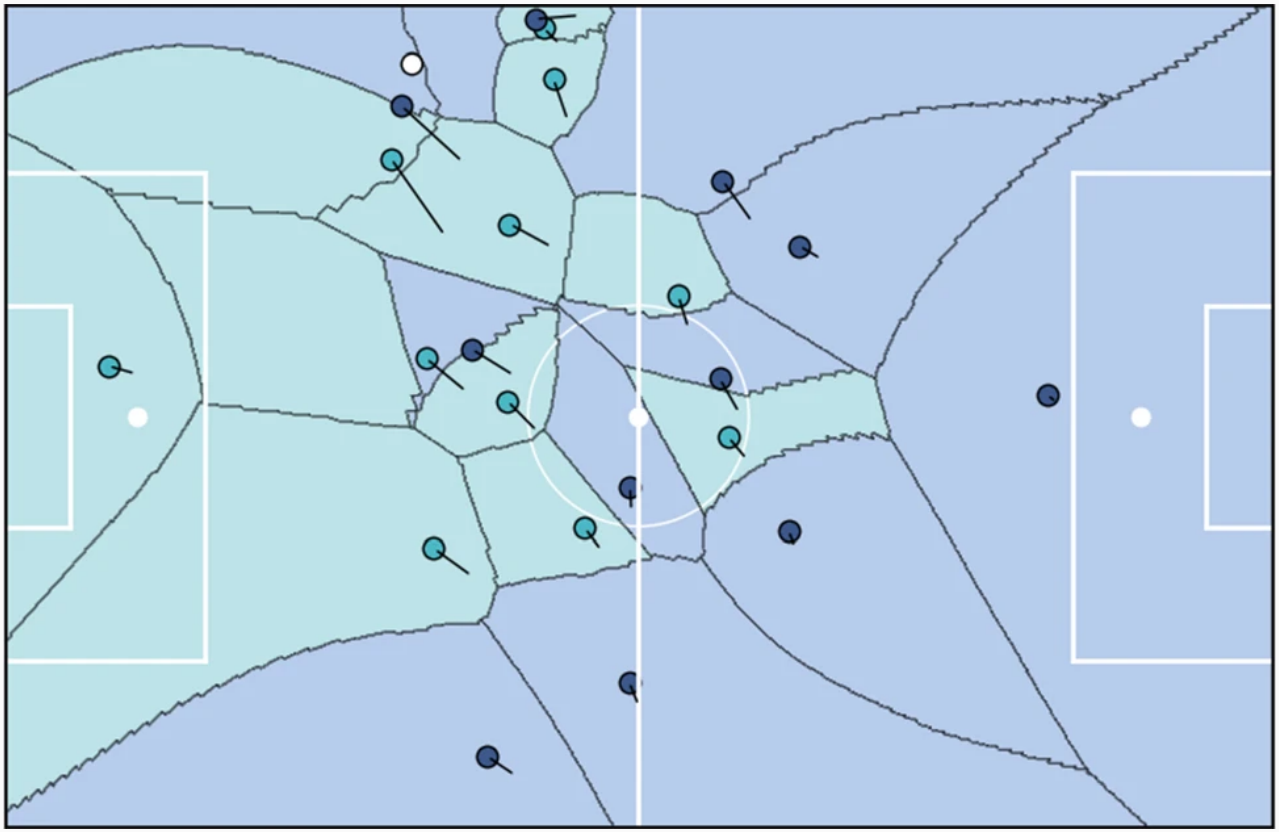
Применения потоковых моделей

- Потоковые модели активно используют для задач, где датасеты поменьше.
- (Bashiri et al., 2021): классификация активностей нейронов (настоящих, биологических)



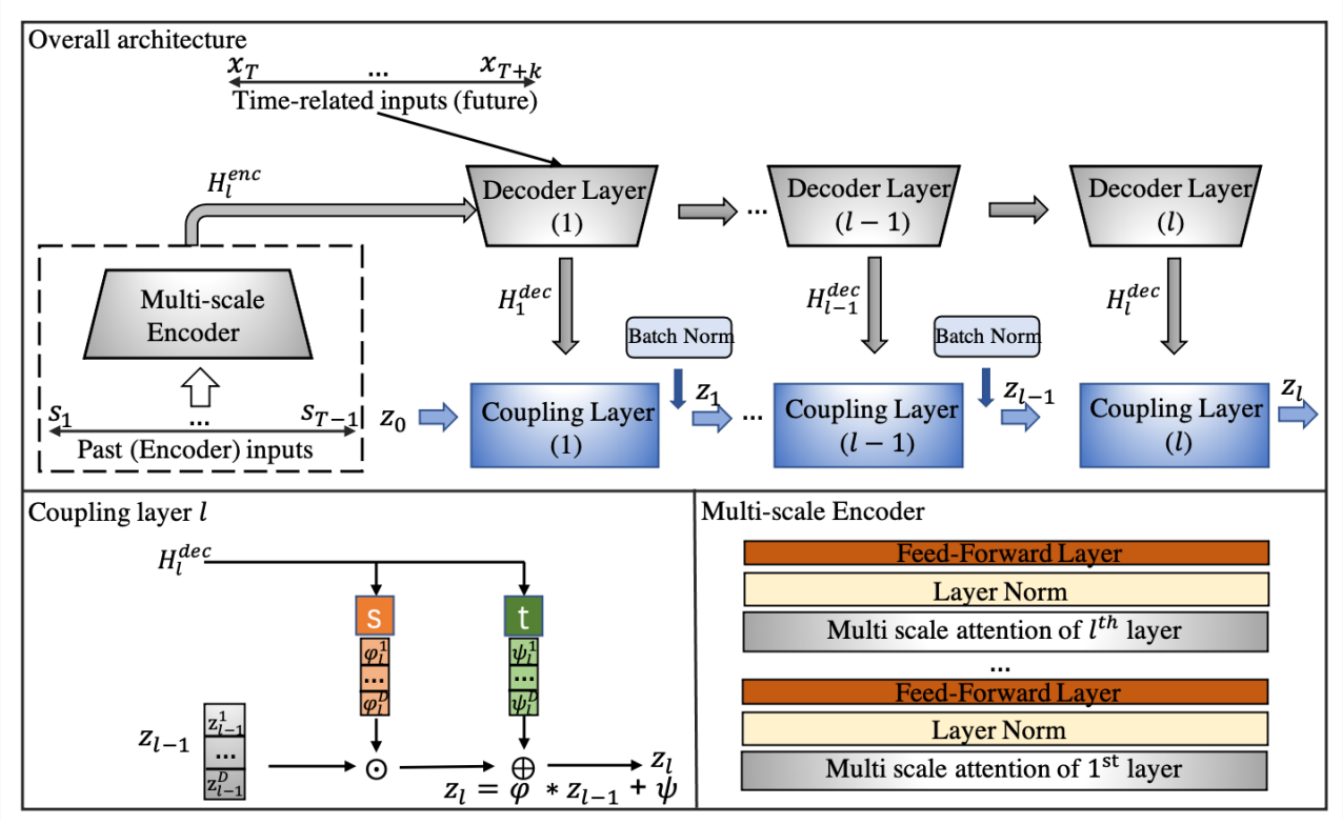
Применения потоковых моделей

- Потоковые модели активно используют для задач, где датасеты поменьше.
- (Fadel et al., 2021): моделирование траекторий футболистов



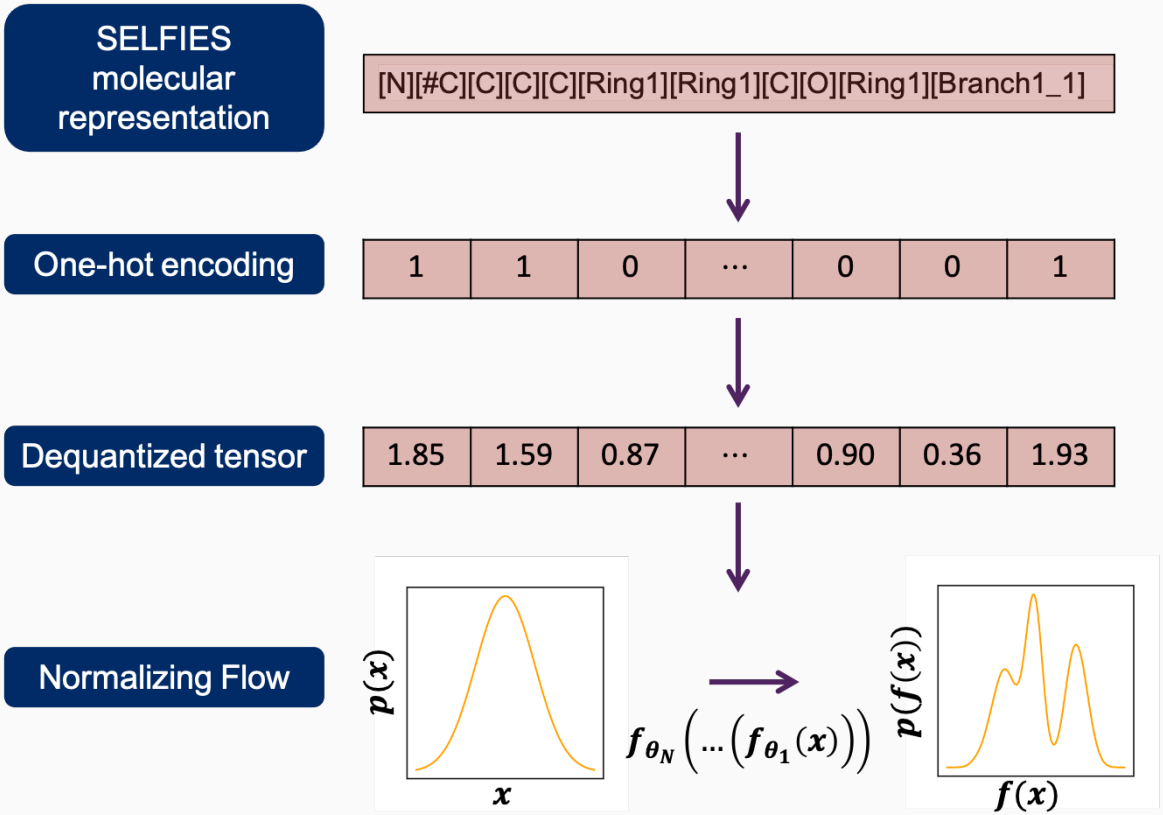
Применения потоковых моделей

- Потоковые модели активно используют для задач, где датасеты поменьше.
- (Feng et al., 2022): потоки для моделирования временных рядов (ещё multi-scale attention)



Применения потоковых моделей

- Потоковые модели активно используют для задач, где датасеты поменьше.
- (Frey et al., 2022): потоки для порождения молекул (SMILES fingerprints, как всегда)



Спасибо за внимание!