

# Синтез речи

## Лекция №3

Гриша Стерлинг, SberDevices

Вокодеры:

1. Griffin-Lim

2. WaveNet

3. Parallel WaveNet

4. WaveGlow

5. WaveRNN

6. LPCNet

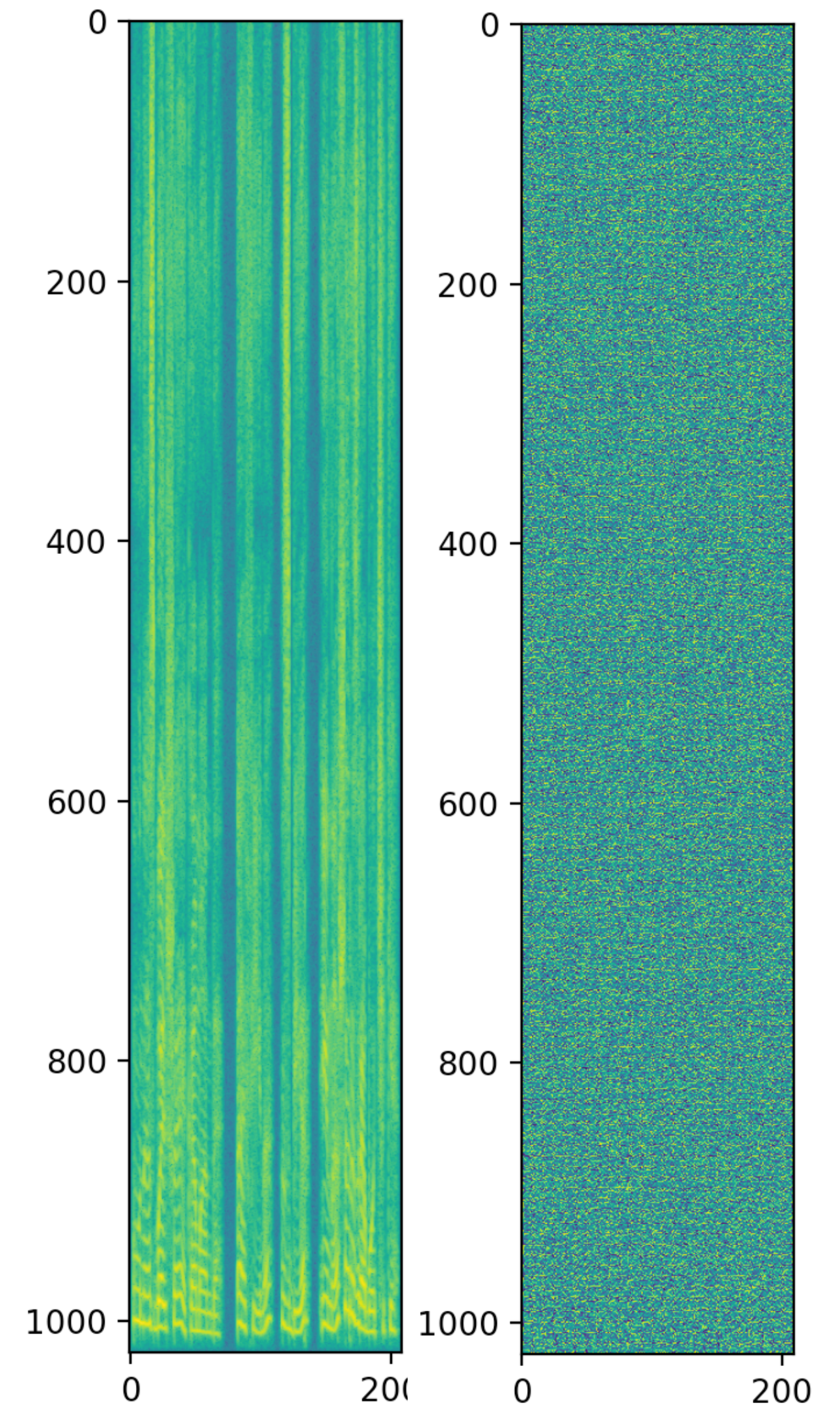
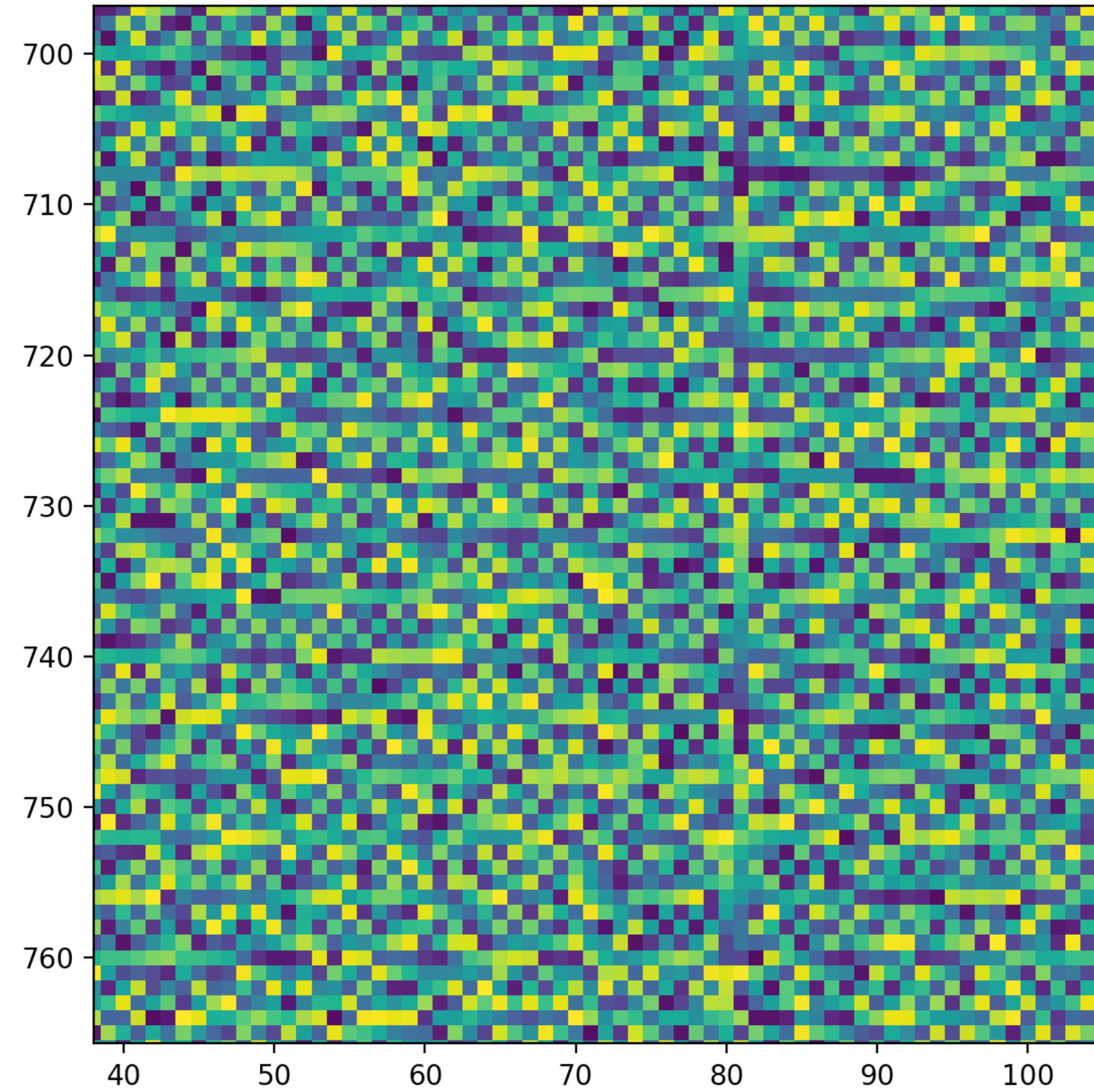
7. GAN-based

# Задача

$Y$  - сигнал

$X = |\text{STFT}(Y)|$

$\text{phase} = \text{angle}(\text{STFT}(Y))$



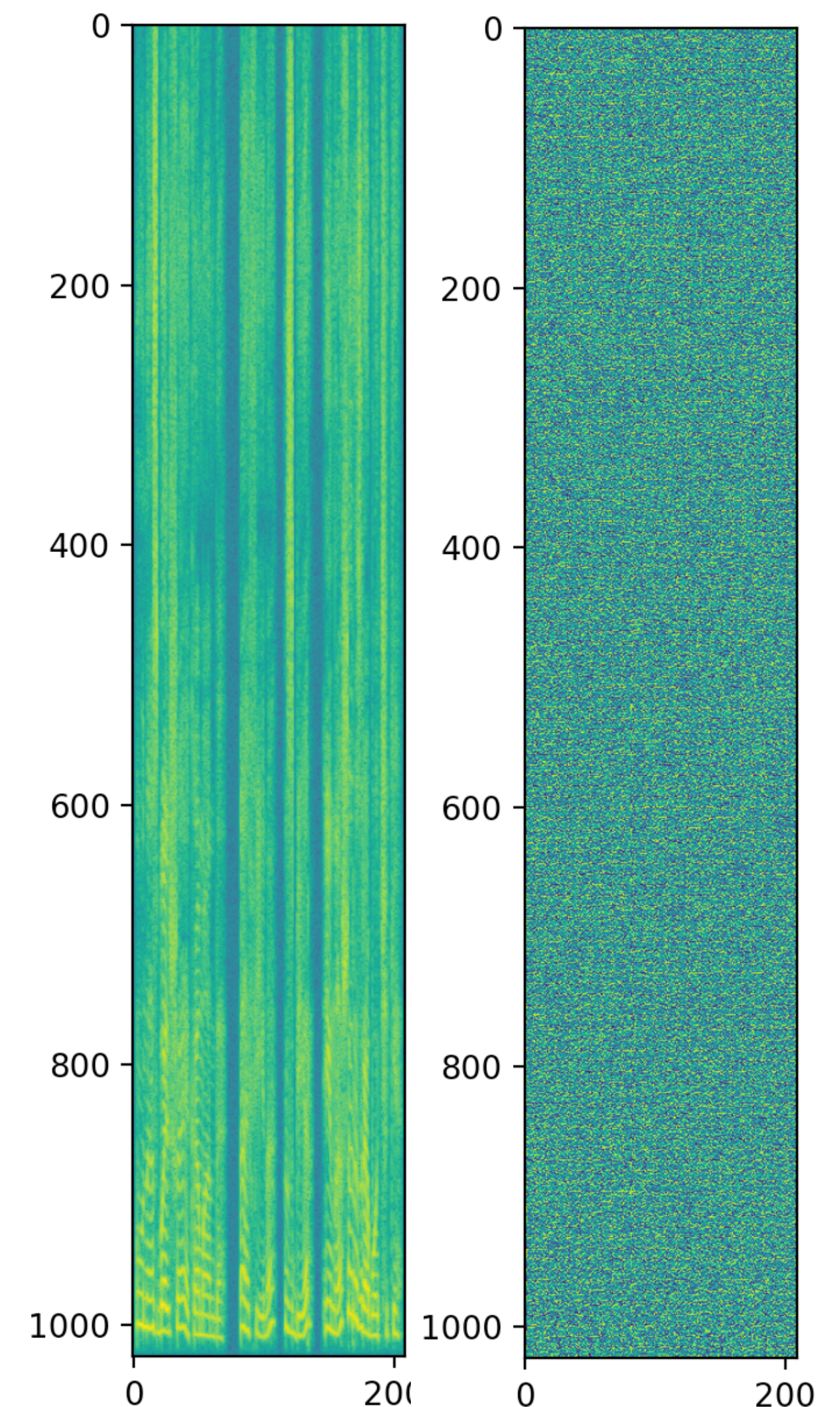
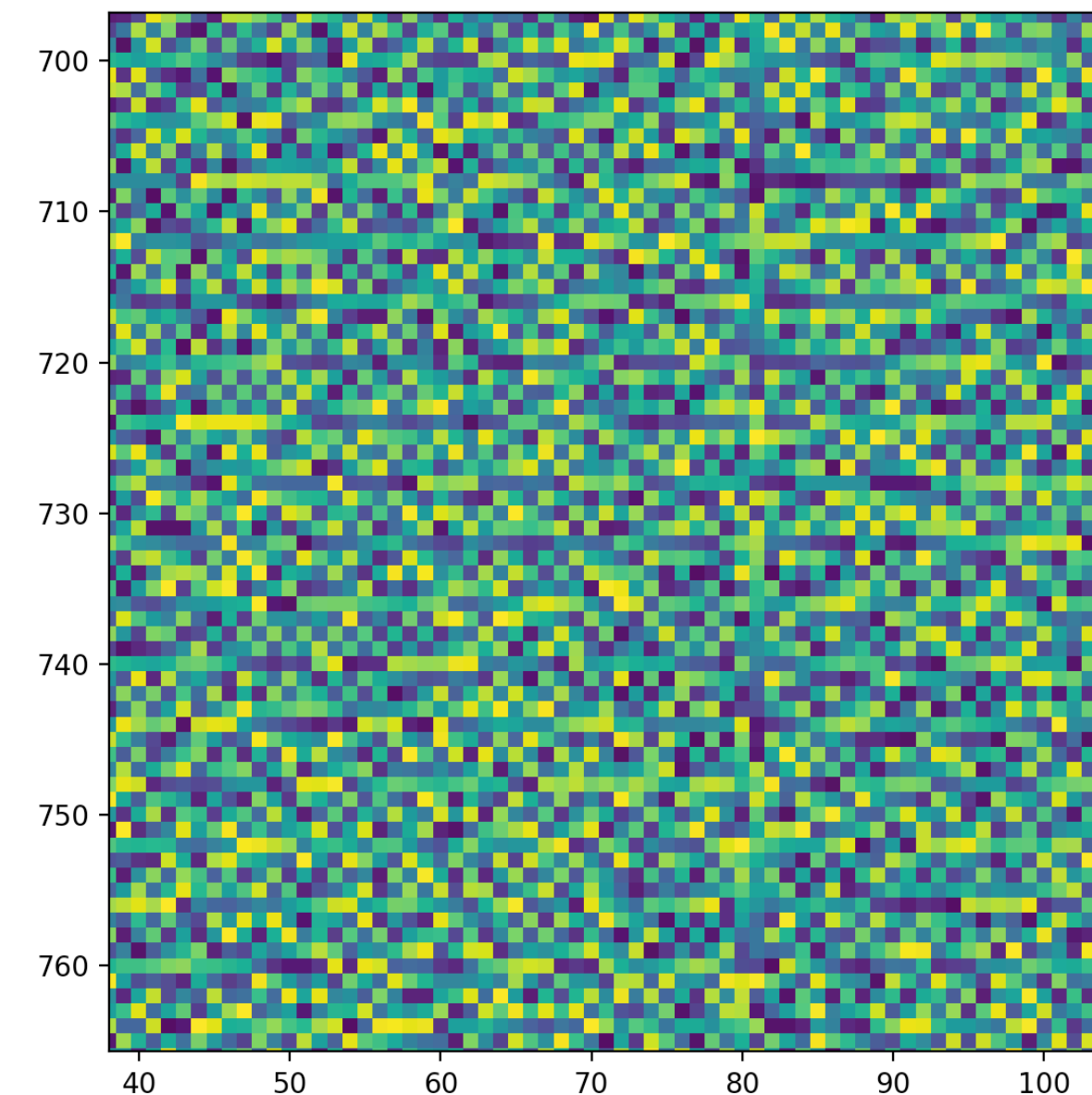


# Griffin-Lim algorithm

<https://paperswithcode.com/method/griffin-lim-algorithm>

Алгоритм:

1. Инициализируем фазу  $Y_0$  случайно
2.  $a_1 = \text{iFFT}(X e^{iY_0})$
3.  $Y_1 = \text{phase}(\text{FFT}(a_1))$
4. Итерируемся до сходимости  
( $a_j = \text{iFFT}(X e^{iY_{j-1}})$ )

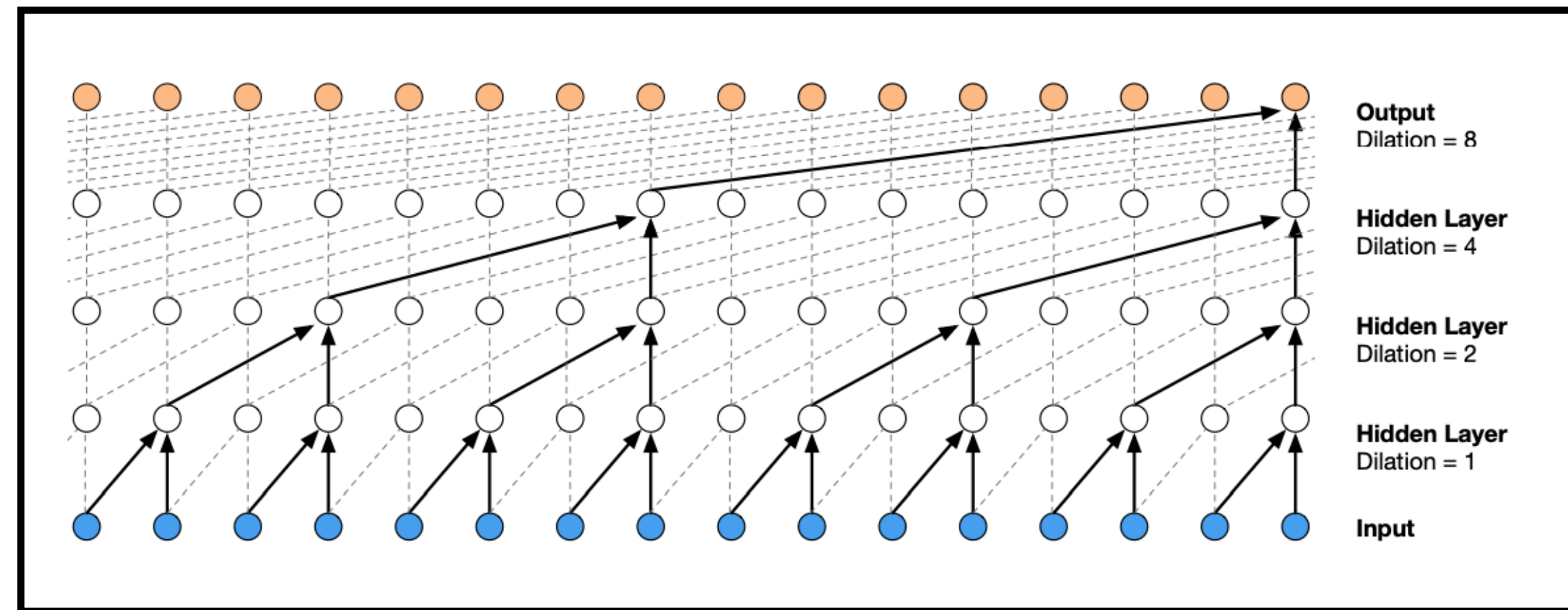




# WaveNet

$$1. \quad x_t = \sum_{p=1}^P a_p x_{t-p} + \epsilon_t$$

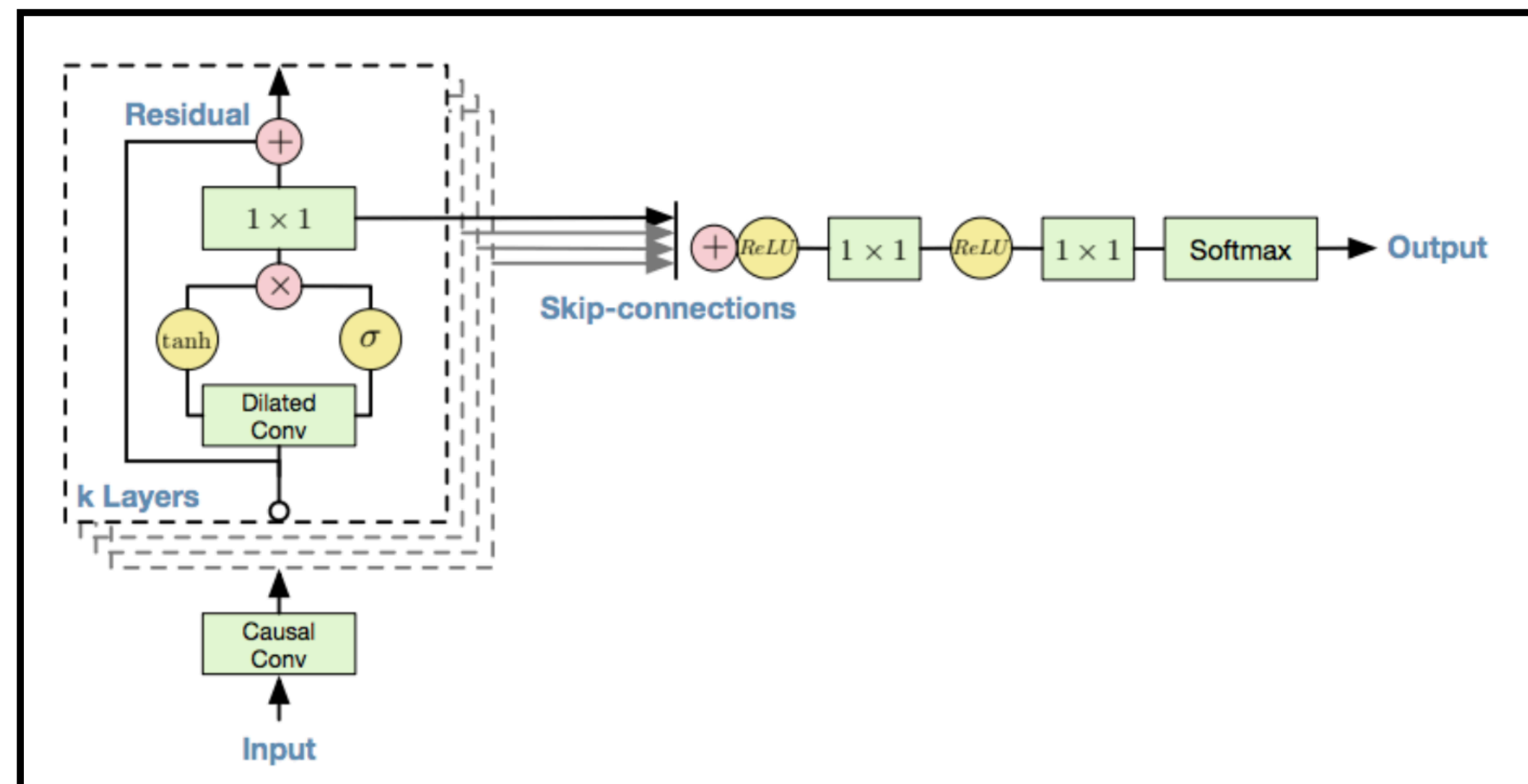
P = ?  
eps = ?



$$2. \quad p(\mathbf{x} | \mathbf{h}) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1}, \mathbf{h}).$$

$$3. \quad \mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k}^T \mathbf{h}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k}^T \mathbf{h}).$$

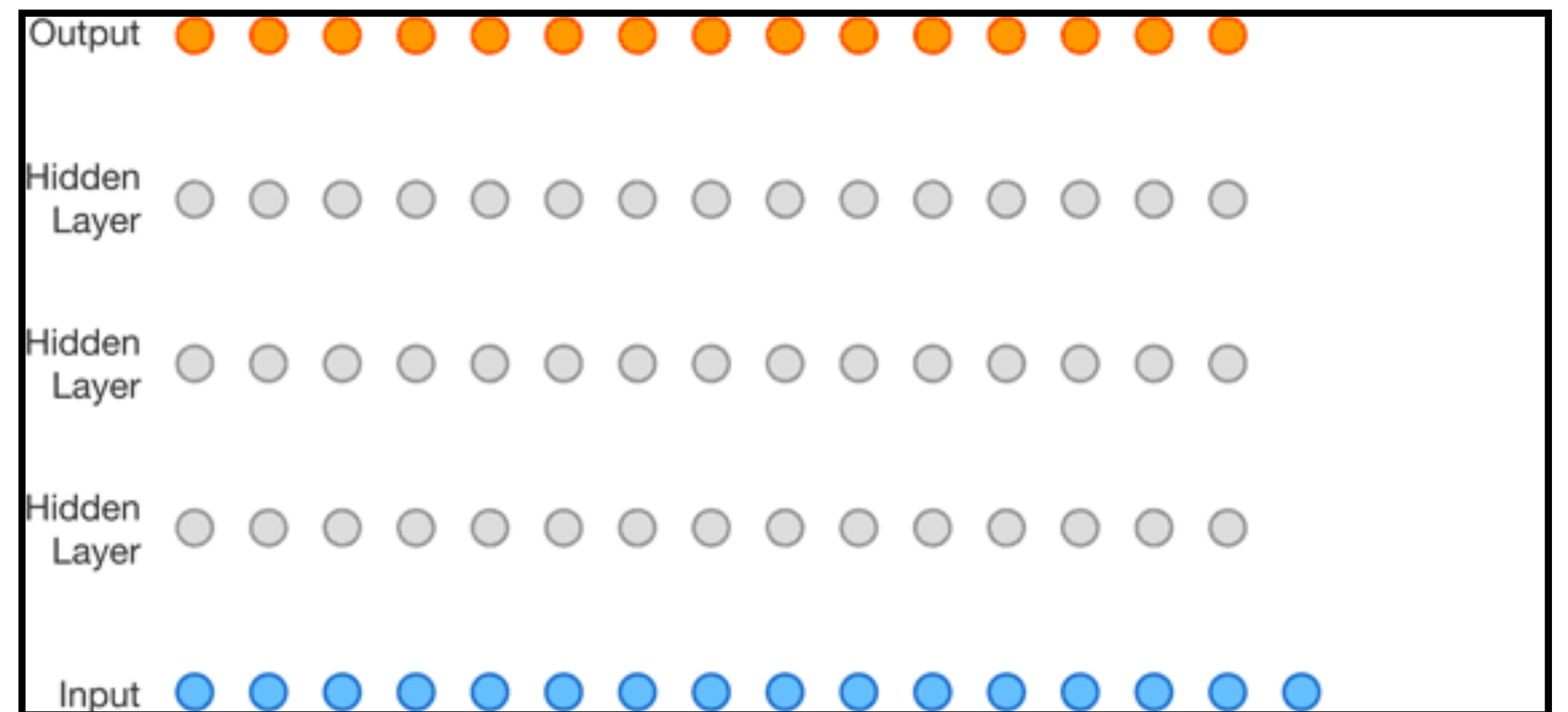
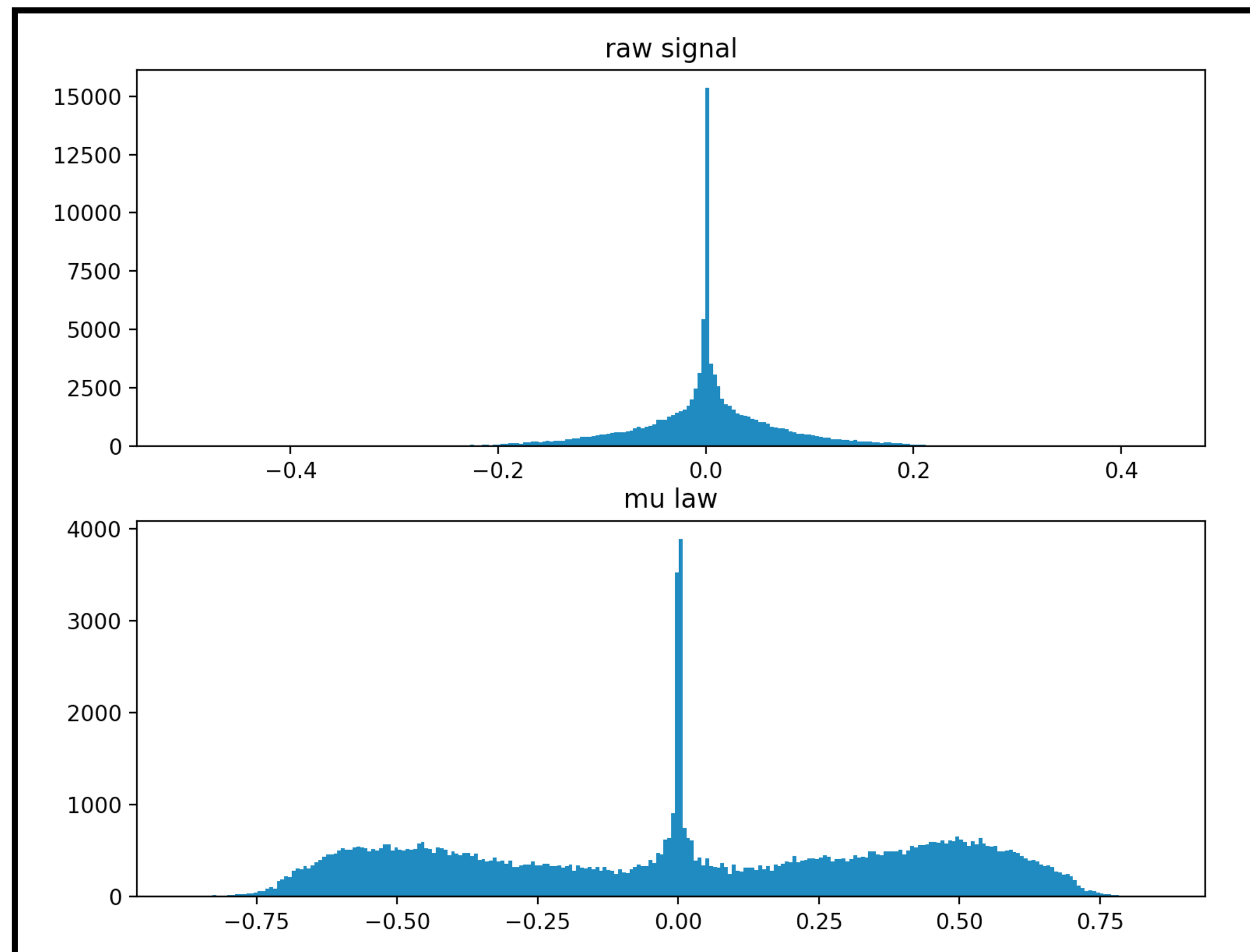
**h** - global conditioning:  
speaker + phonemes



# Wavenet

$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 + \mu |x_t|)}{\ln(1 + \mu)},$$

Inference:



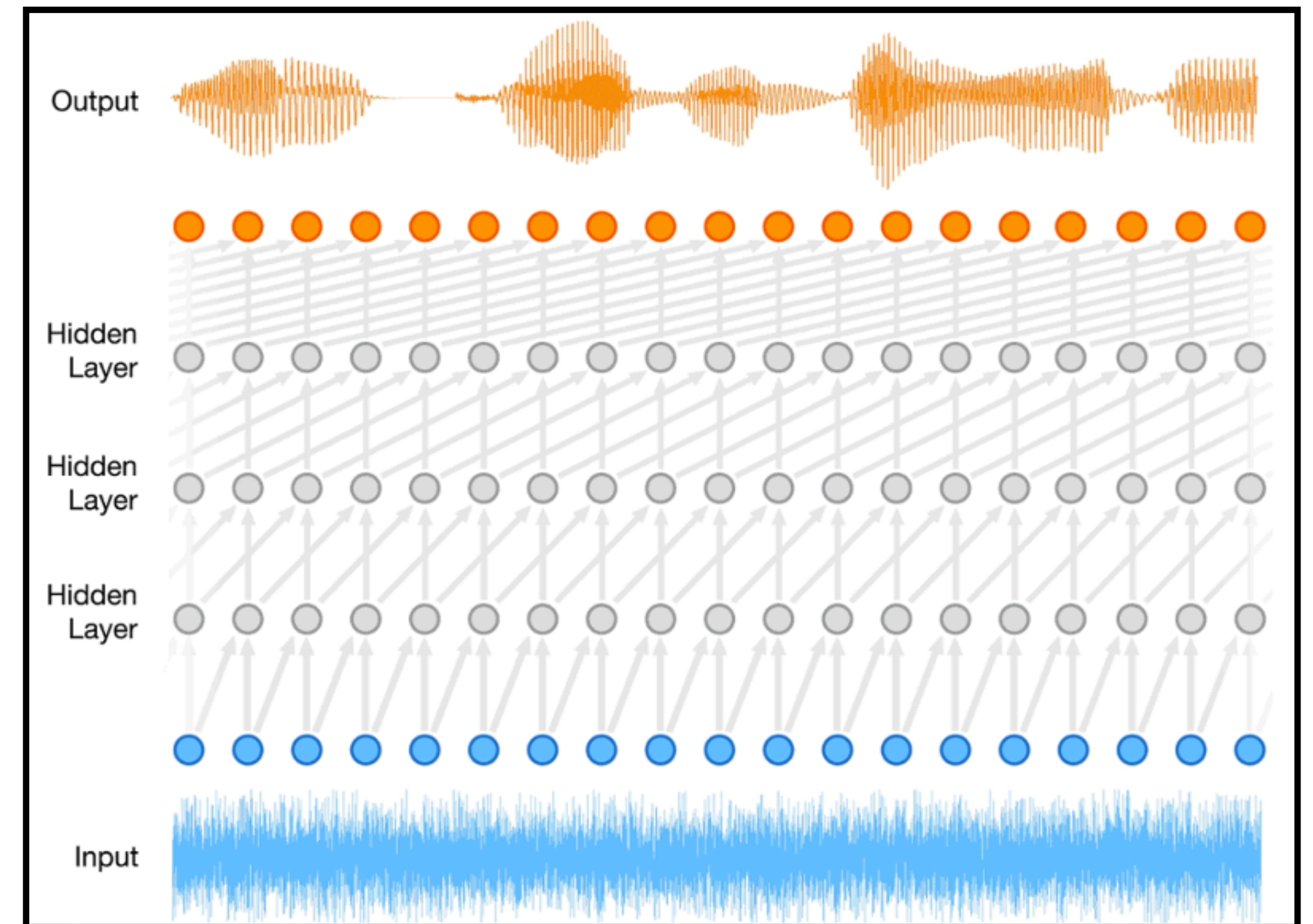
# Parallel WaveNet

WaveNet:

- + крутое качество
- в 20 раз медленнее realtime :)

Мотивация:

хочется синтезировать всю  
вавку за один инференс сетки



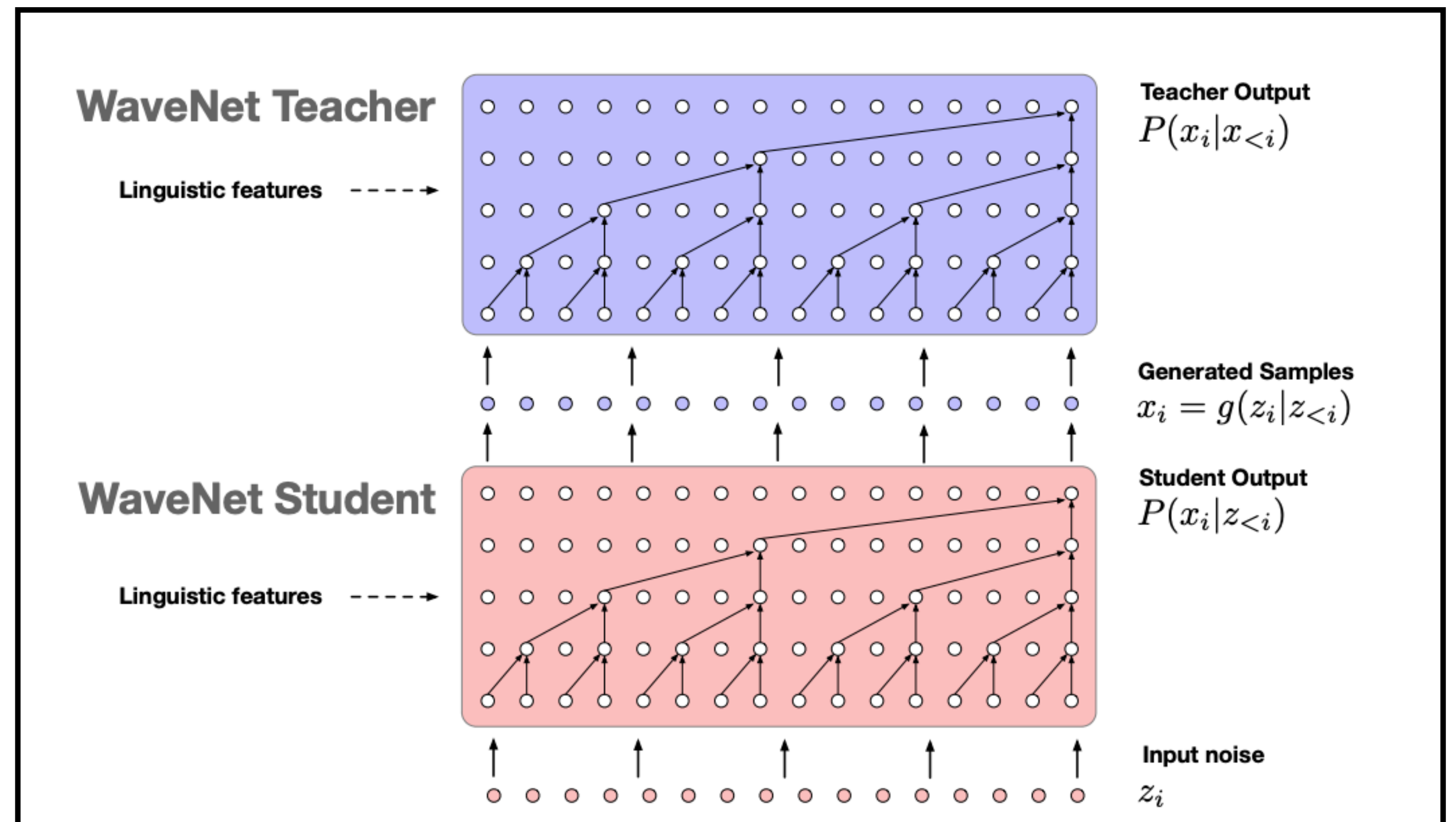
# Parallel WaveNet

Knowledge distillation:

$$D_{\text{KL}}(P_S || P_T) = H(P_S, P_T) - H(P_S)$$

Чтобы все лучше училось:

- stft loss
- perceptual loss
- contrastive loss





# WaveGlow

$$z \sim \mathcal{N}(z; 0, I)$$

$$x = f_0 \circ f_1 \circ \dots \circ f_k(z)$$

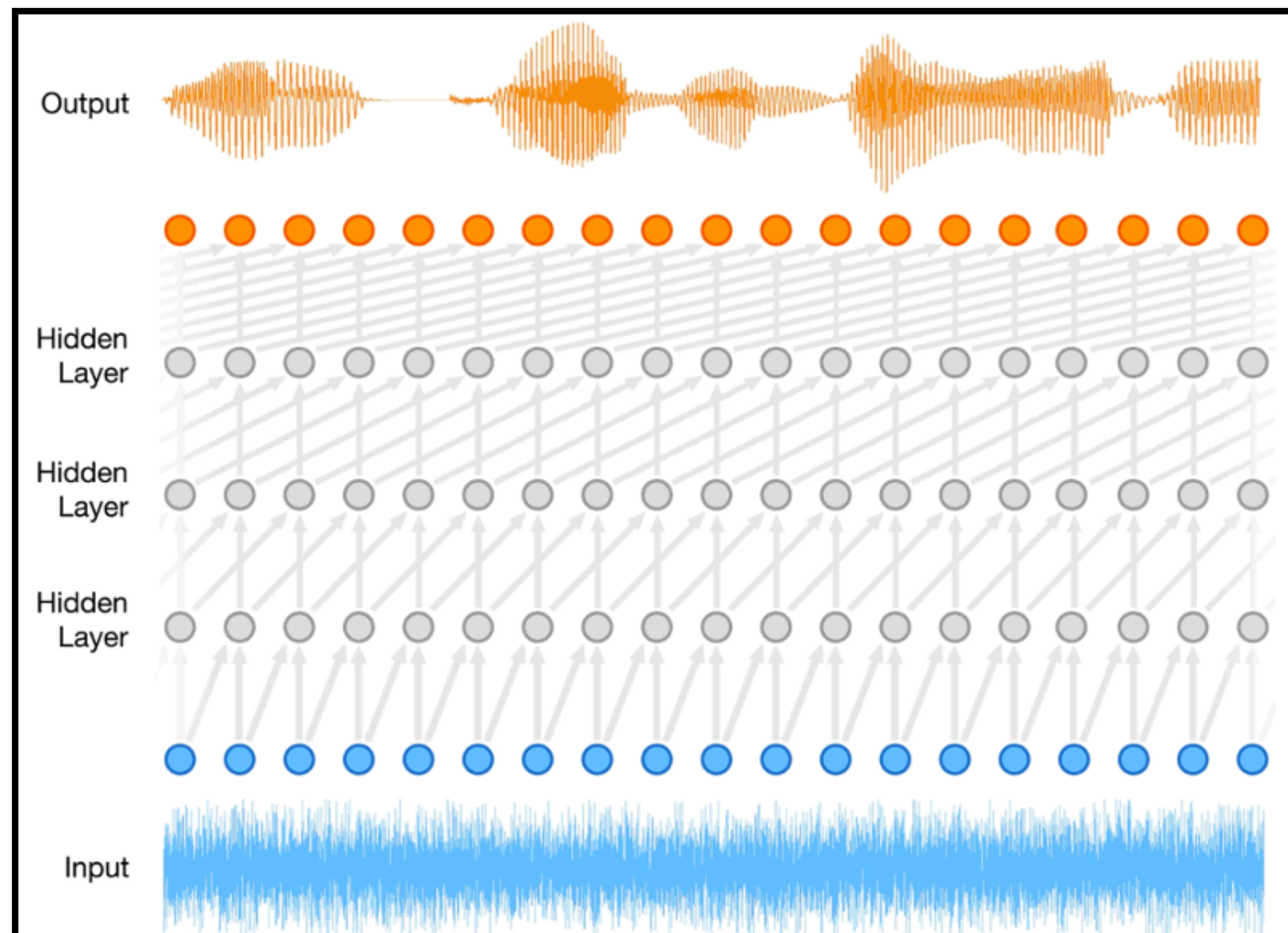
$$z = f_k^{-1} \circ f_{k-1}^{-1} \circ \dots \circ f_0^{-1}(x)$$

Обычно:

- для какого-то входа считаем выход
- считаем лосс и градиенты

WaveGlow:

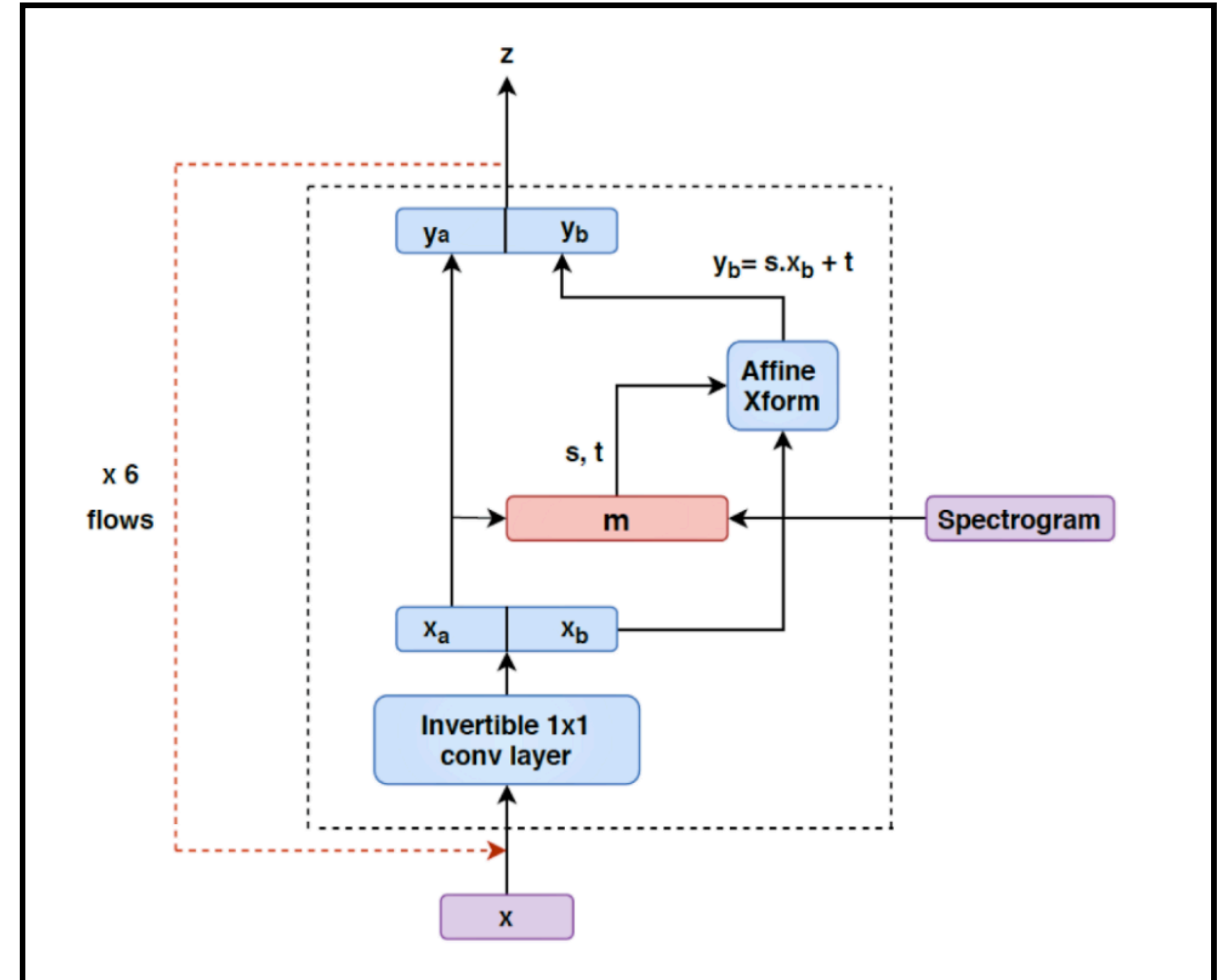
- подставляем вейвформу в конец сетки
- считаем с помощью обратных преобразований, из какого входа оно было получено
- считаем лосс между  $z \sim \mathcal{N}(0, 1)$  и  $z'$



# WaveGlow

$$\begin{aligned} \mathbf{x}_a, \mathbf{x}_b &= \text{split}(\mathbf{x}) \\ (\log \mathbf{s}, \mathbf{t}) &= \mathbf{W}\mathbf{N}(\mathbf{x}_a, \text{mel-spectrogram}) \\ \mathbf{x}_b' &= \mathbf{s} \odot \mathbf{x}_b + \mathbf{t} \\ \mathbf{f}_{\text{coupling}}^{-1}(\mathbf{x}) &= \text{concat}(\mathbf{x}_a, \mathbf{x}_b') \end{aligned}$$

$$\begin{aligned} \log p_{\theta}(\mathbf{x}) &= -\frac{\mathbf{z}(\mathbf{x})^T \mathbf{z}(\mathbf{x})}{2\sigma^2} \\ &+ \sum_{j=0}^{\# \text{coupling}} \log \mathbf{s}_j(\mathbf{x}, \text{mel-spectrogram}) \\ &+ \sum_{k=0}^{\# \text{conv}} \log \det |\mathbf{W}_k| \end{aligned}$$



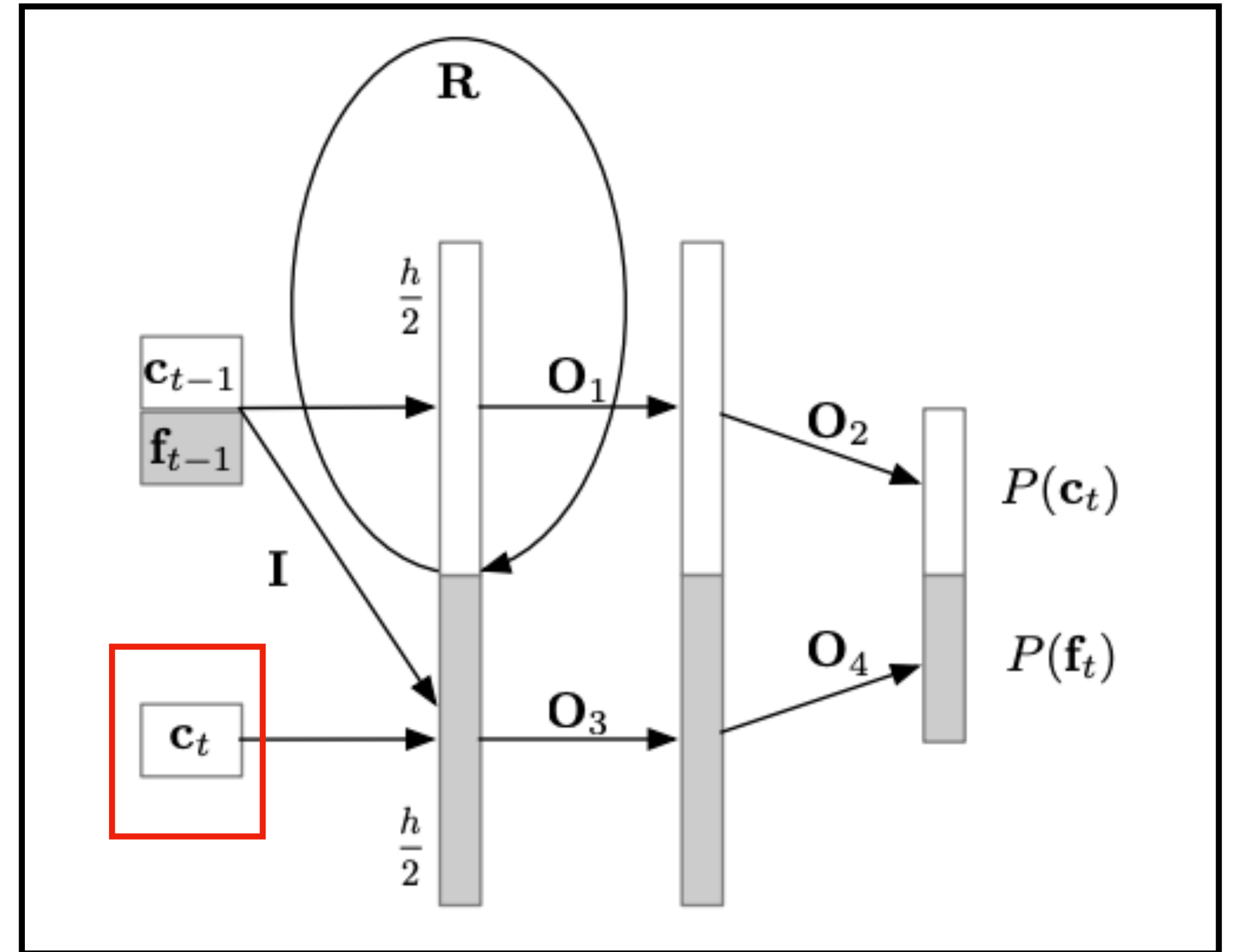


# WaveRNN

$$p(\mathbf{x} | \mathbf{h}) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1}, \mathbf{h}).$$

Идеи:

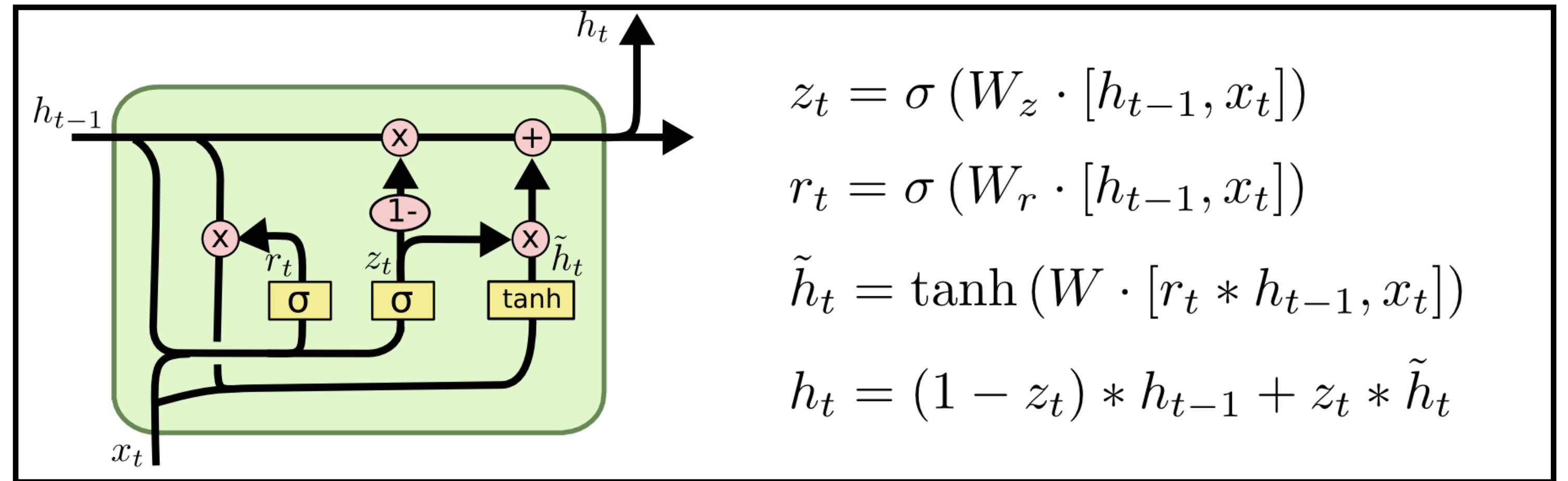
- использовать hidden state у GRU слоя вместо кучи сверток
- coarse и fine представление вместо mu-law
- оптимизации (спарсификация, subscale)



# WaveRNN

$$\begin{aligned}
 \mathbf{x}_t &= [\mathbf{c}_{t-1}, \mathbf{f}_{t-1}, \mathbf{c}_t] \\
 \mathbf{u}_t &= \sigma(\mathbf{R}_u \mathbf{h}_{t-1} + \mathbf{I}_u^* \mathbf{x}_t) \\
 \mathbf{r}_t &= \sigma(\mathbf{R}_r \mathbf{h}_{t-1} + \mathbf{I}_r^* \mathbf{x}_t) \\
 \mathbf{e}_t &= \tau(\mathbf{r}_t \circ (\mathbf{R}_e \mathbf{h}_{t-1}) + \mathbf{I}_e^* \mathbf{x}_t) \\
 \mathbf{h}_t &= \mathbf{u}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \circ \mathbf{e}_t \\
 \mathbf{y}_c, \mathbf{y}_f &= \text{split}(\mathbf{h}_t) \\
 P(\mathbf{c}_t) &= \text{softmax}(\mathbf{O}_2 \text{relu}(\mathbf{O}_1 \mathbf{y}_c)) \\
 P(\mathbf{f}_t) &= \text{softmax}(\mathbf{O}_4 \text{relu}(\mathbf{O}_3 \mathbf{y}_f))
 \end{aligned}$$

+ conditioning



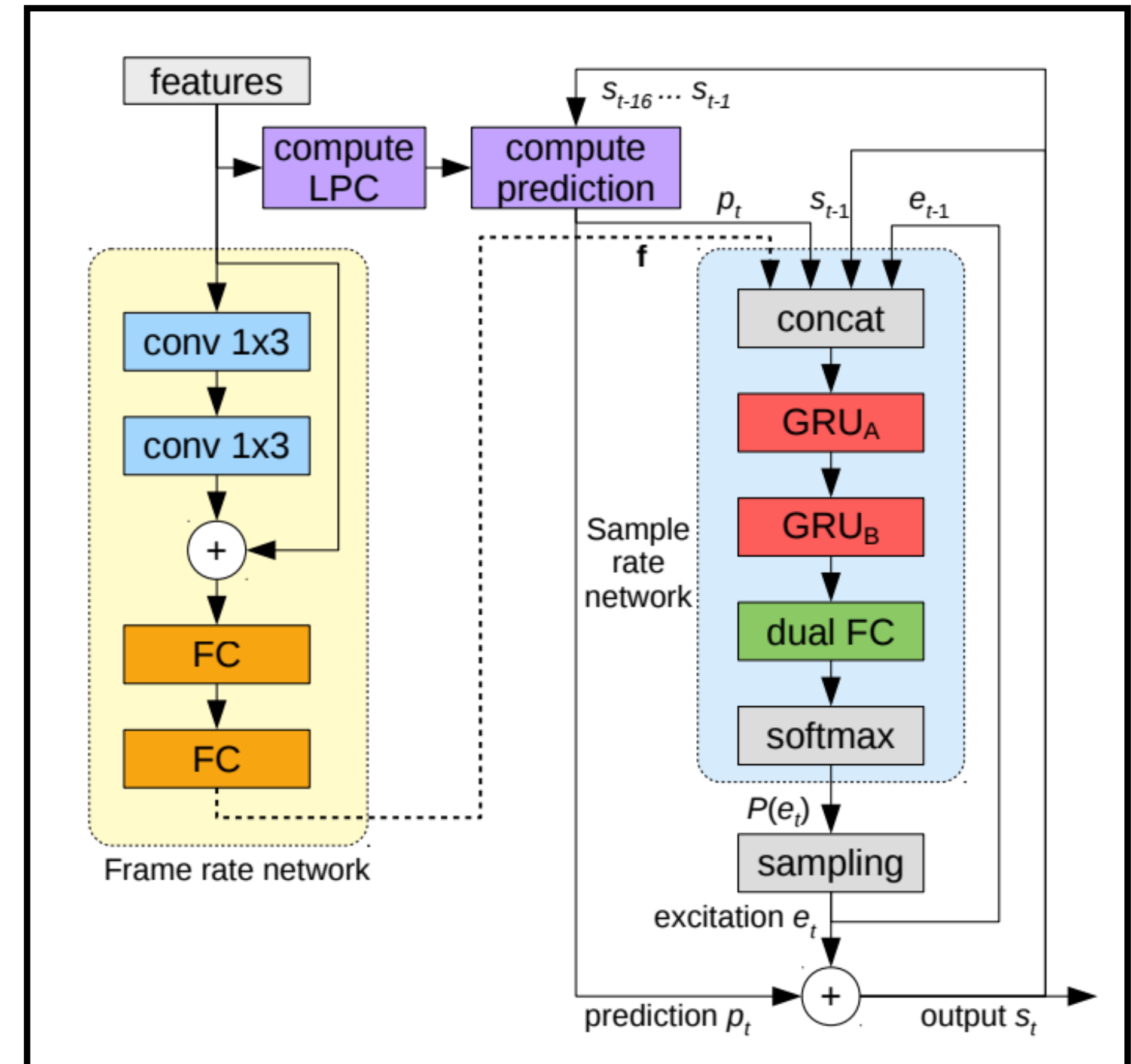


# LPCNet

Идея:  
помочь WaveRNN с помощью  
linear predictive coding

$$x_t = \sum_{p=1}^P a_p x_{t-p} + \epsilon_t$$

$a_p$  считаются по формулам  
 $e_t$  предсказываются сеткой



# LPCNet

Как считаются LP коэффициенты:

- 1) Есть теорема, что автокорреляционная функция - это обратное фффт от квадрата спектральной функции

$$\Psi(\tau) = \int_{-\infty}^{\infty} f(t) f^*(t - \tau) dt$$

$$\Psi(\tau) \sim \text{Re } \text{fft}^{-1} \left( |\text{fft}(\vec{x})|^2 \right)$$

- 2) LP коэффициенты получаются как решения системы линейных уравнений

$$\begin{bmatrix} R_{\hat{n}}(0) & R_{\hat{n}}(1) & \cdot & \cdot & R_{\hat{n}}(p-1) \\ R_{\hat{n}}(1) & R_{\hat{n}}(0) & \cdot & \cdot & R_{\hat{n}}(p-2) \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ R_{\hat{n}}(p-1) & R_{\hat{n}}(p-2) & \cdot & \cdot & R_{\hat{n}}(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_{\hat{n}}(1) \\ R_{\hat{n}}(2) \\ \cdot \\ \cdot \\ R_{\hat{n}}(p) \end{bmatrix}$$

<- это Теплицева матрица - симметричная, с одинаковыми элементами на диагонали, поэтому решение ищется за  $O(n^2)$



# LPCNet

RNN input:

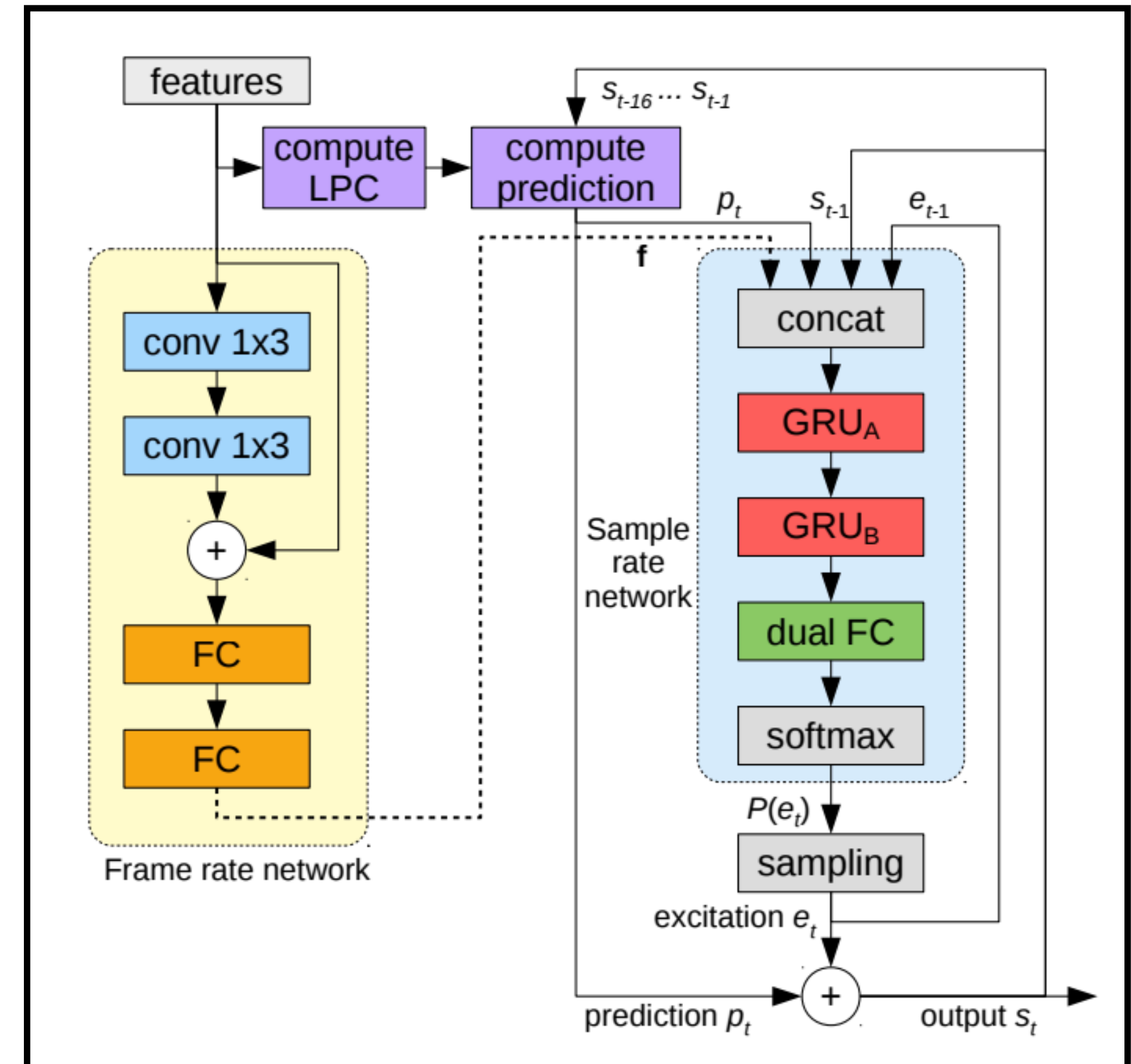
- LP предсказание  
(по  $P$  значениям сигнала)
- значение сигнала на прошлом шаге
- excitation на прошлом шаге

RNN output:

- excitation на текущем шаге

LPCNet output:

- $s_t = \text{LPC}(s_{<t}) + E_t$



# LPCNet

Архитектура nn унаследована от WaveRNN:

$$\begin{aligned} \mathbf{u}_t &= \sigma \left( \mathbf{W}_u \mathbf{h}_t + \mathbf{v}_{s_{t-1}}^{(u,s)} + \mathbf{v}_{p_{t-1}}^{(u,p)} + \mathbf{v}_{e_{t-1}}^{(u,e)} + \mathbf{g}^{(u)} \right) \\ \mathbf{r}_t &= \sigma \left( \mathbf{W}_r \mathbf{h}_t + \mathbf{v}_{s_{t-1}}^{(r,s)} + \mathbf{v}_{p_{t-1}}^{(r,p)} + \mathbf{v}_{e_{t-1}}^{(r,e)} + \mathbf{g}^{(r)} \right) \\ \tilde{\mathbf{h}}_t &= \tanh \left( \mathbf{r}_t \circ (\mathbf{W}_h \mathbf{h}_t) + \mathbf{v}_{s_{t-1}}^{(h,s)} + \mathbf{v}_{p_{t-1}}^{(h,p)} + \mathbf{v}_{e_{t-1}}^{(h,e)} + \mathbf{g}^{(h)} \right) \\ \mathbf{h}_t &= \mathbf{u}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \circ \tilde{\mathbf{h}}_t \\ P(e_t) &= \text{softmax}(\text{dual\_fc}(\text{GRU}_B(\mathbf{h}_t))) , \end{aligned} \quad (5)$$

- Никаких coarse и fine
- Dual FC одинаковы по виду и предсказывают одно и то же
- GRU\_a и GRU\_b размером 384 и 16 юнитов вместо 896
- Сложный инференс (из-за voiced и unvoiced звука)
- Спарсификация



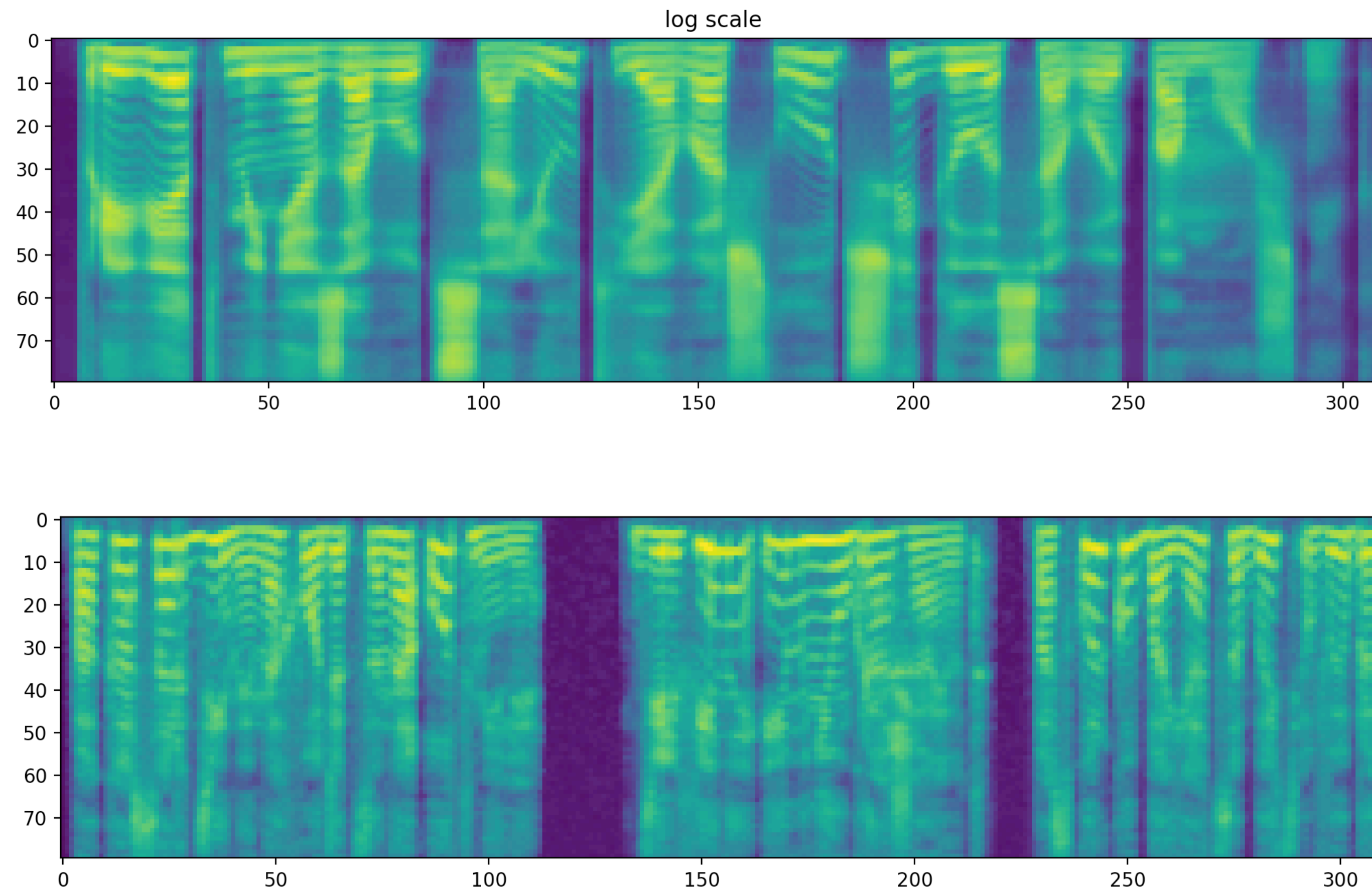
# Adversarial vocoder

Синтезированные спектрограммы  
всегда слишком сглажены  
(из-за MSE лосса)

Схема adversarial vocoding:

- 1)  $x$  = synthesized mel-spec
- 2)  $nn(x)$  - обычная спектрограмма
- 3) GL-алгоритм

Здесь  $nn$  - это генератор в GAN



# Остальные GAN-вокодеры

Идея:

Генератор синтезирует вейвформу

Дискриминатор пытается определить, настоящая ли это вейвформа

Примеры:

- WaveGAN: применили DCGAN к звуку
- MelGAN: добавили multiscale D и feature map loss
- StyleMelGan: G и D посложнее
- HiFiGAN: G и multiscale D + MSE от спектрограмм
- Multiband MelGAN: multiscale MSE от спектрограмм
- VocGAN: Сложный multiscale G и D + MSE от спектрограмм
- Parallel WaveGAN: G - кусок Parallel WaveNet

# HiFiGAN

## Generator G

