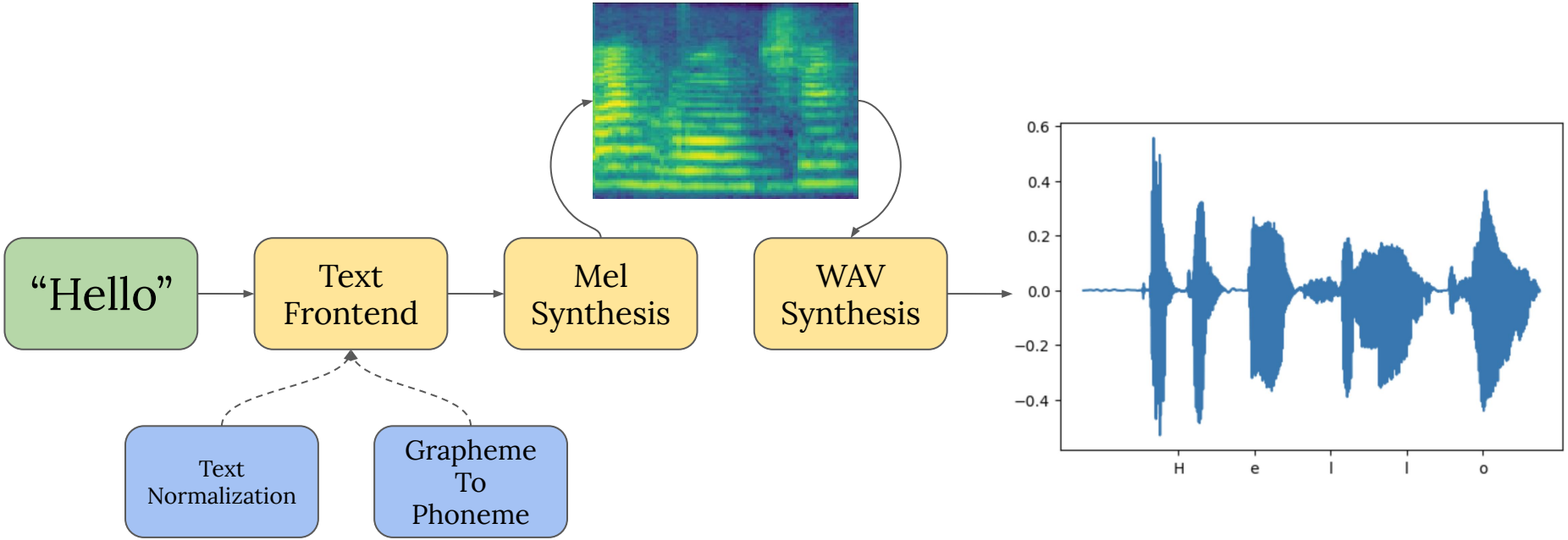
Speech Synthesis

What is a TTS in general?



Grapheme VS Phoneme

- A phoneme is simply a sound
- A grapheme is the smallest fundamental unit in written language
- There isn't bijection because depend on context

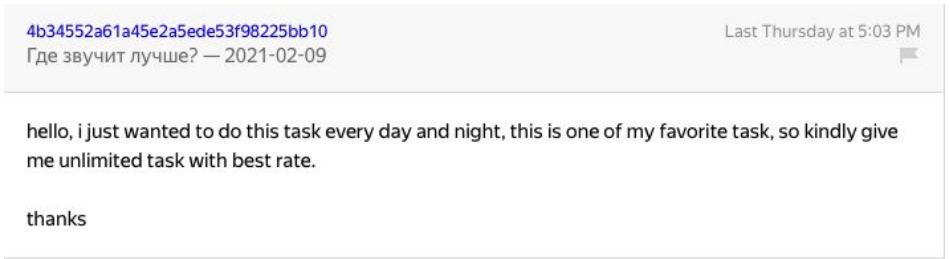
Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW

What About Public Datasets?

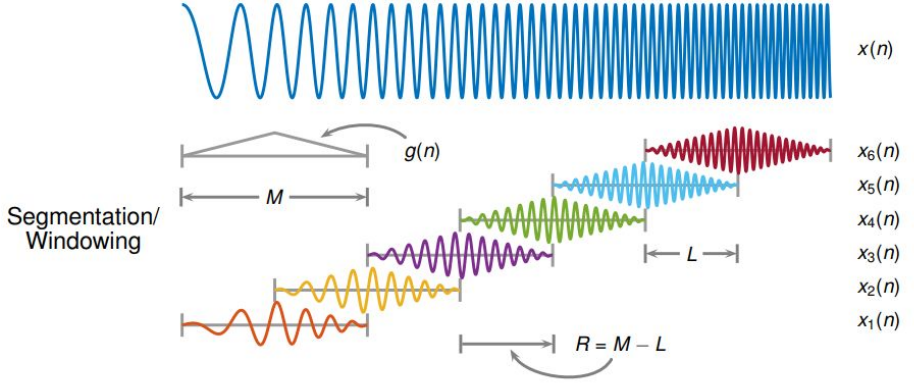
- LJSpeech
- LibtiTTS
- CommonVoice
- OpenTTS (Russian)

How to Measure Quality?

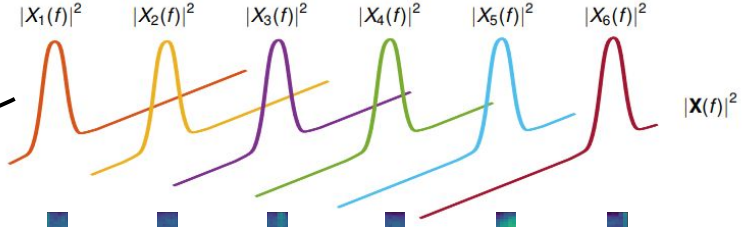
- There’s no correct answer
- Subjective perception
- A lot of types of mistakes
- One solution is MOS
- Another is Side-by-side (SBS)



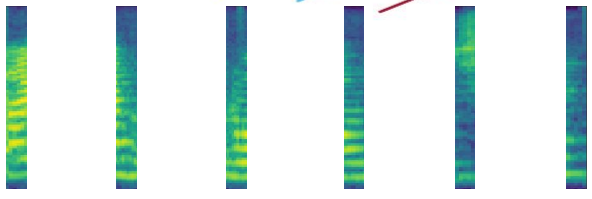
What is MelSpectrogram?

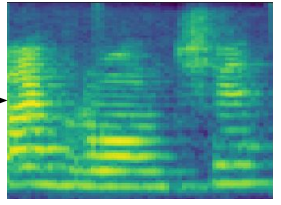


DFT



log(MelScale(x))

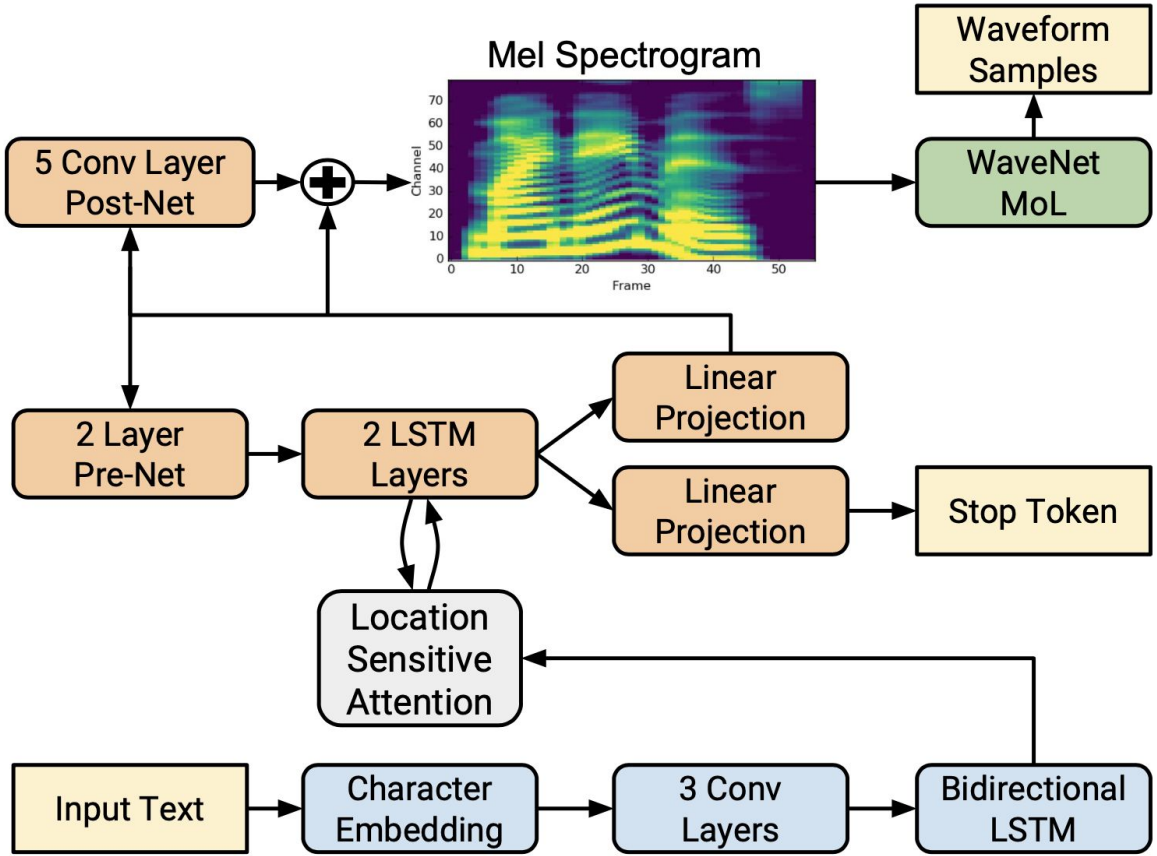




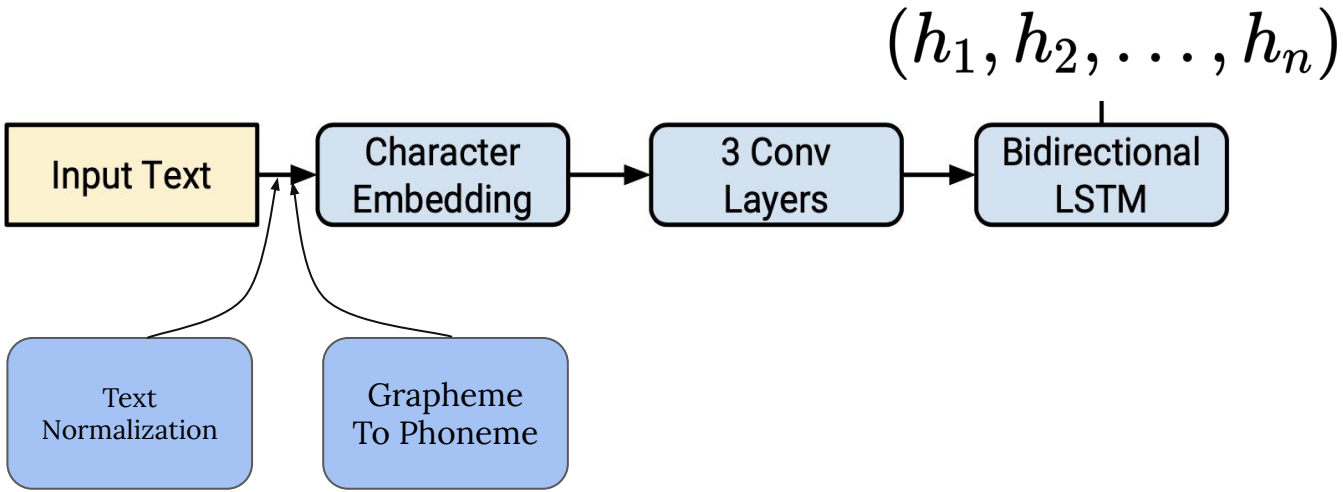
Wait! Why are we introducing MelSpectrogram?

- Train two component separately
  - Faster and easier
- MelSpectrogram is smoother than waveform
- Easier to train using a MSE

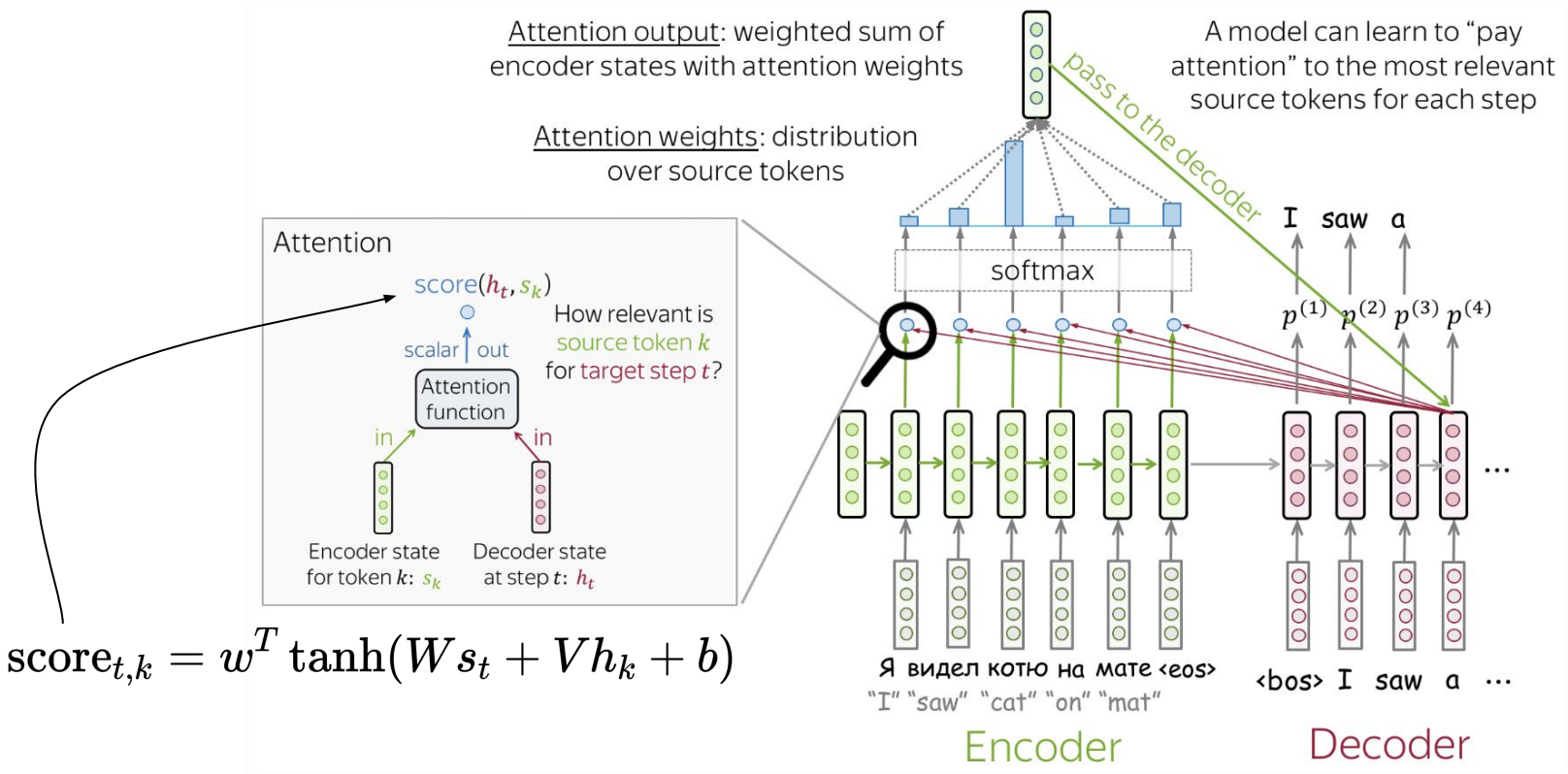
Tacotron 2


Mel Spectrogram
Channel
Frame
Waveform Samples
WaveNet MoL
5 Conv Layer Post-Net
2 Layer Pre-Net
2 LSTM Layers
Linear Projection
Linear Projection
Stop Token
Location Sensitive Attention
Input Text
Character Embedding
3 Conv Layers
Bidirectional LSTM

Text Encoder



Attention



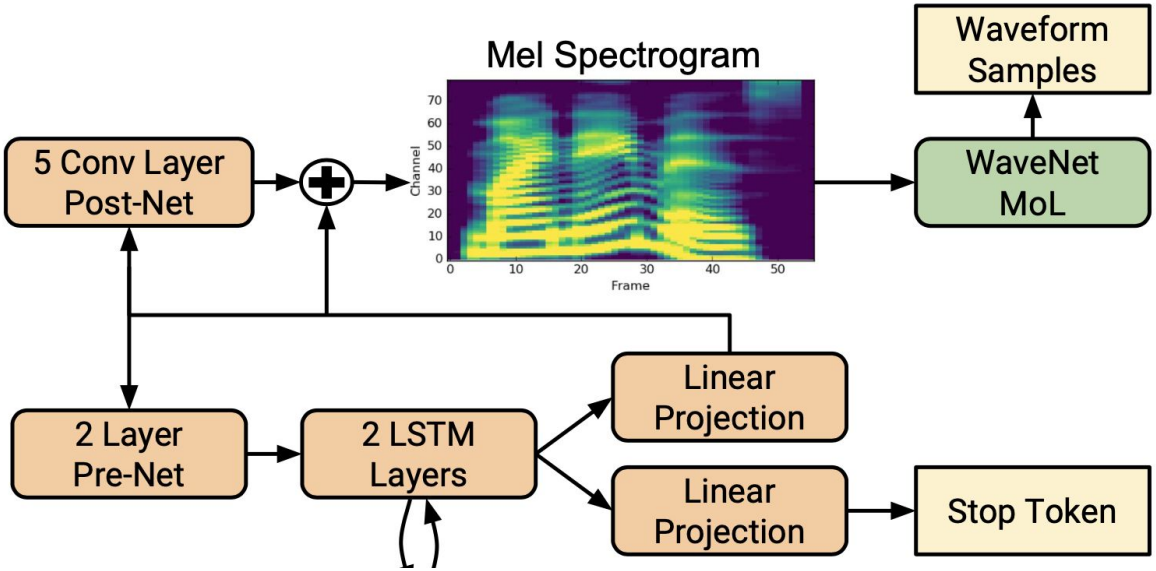
$$\text{score}_{t,k} = w^T \tanh(Ws_t + Vh_k + b)$$

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Decoder

$$\mathcal{L} = \mathcal{L}_{\text{pre}} + \mathcal{L}_{\text{post}} + \text{StopToken}$$
$$\mathcal{L}_{\text{pre}} = \text{MSE}(x, \hat{x}_{\text{pre}})$$
$$\mathcal{L}_{\text{pre}} = \text{MSE}(x, \hat{x}_{\text{post}})$$
$$\text{StopToken} = \text{CE}(h, \mathbb{I}[h = \text{stop}])$$

- Neighboring predicted mels are correlated
- Turn on dropout on inference in PreNet

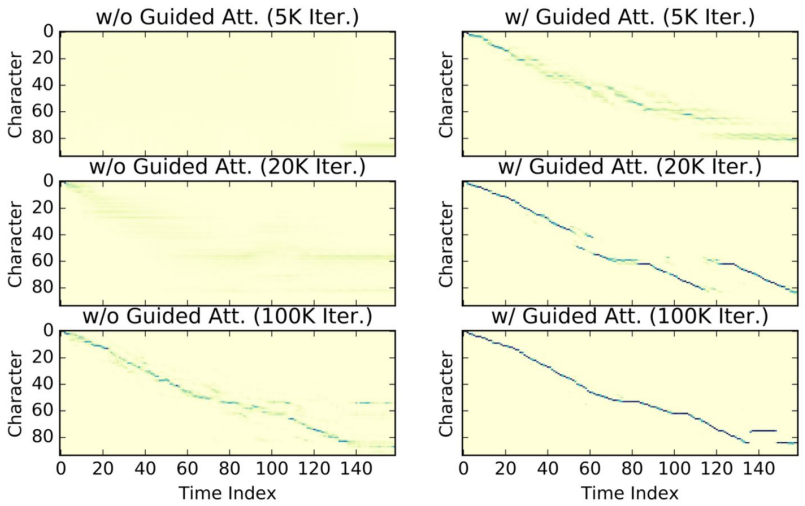


Guided Attention

- Hard to learn attention from scratch
- Text position n progresses linearly to time t:

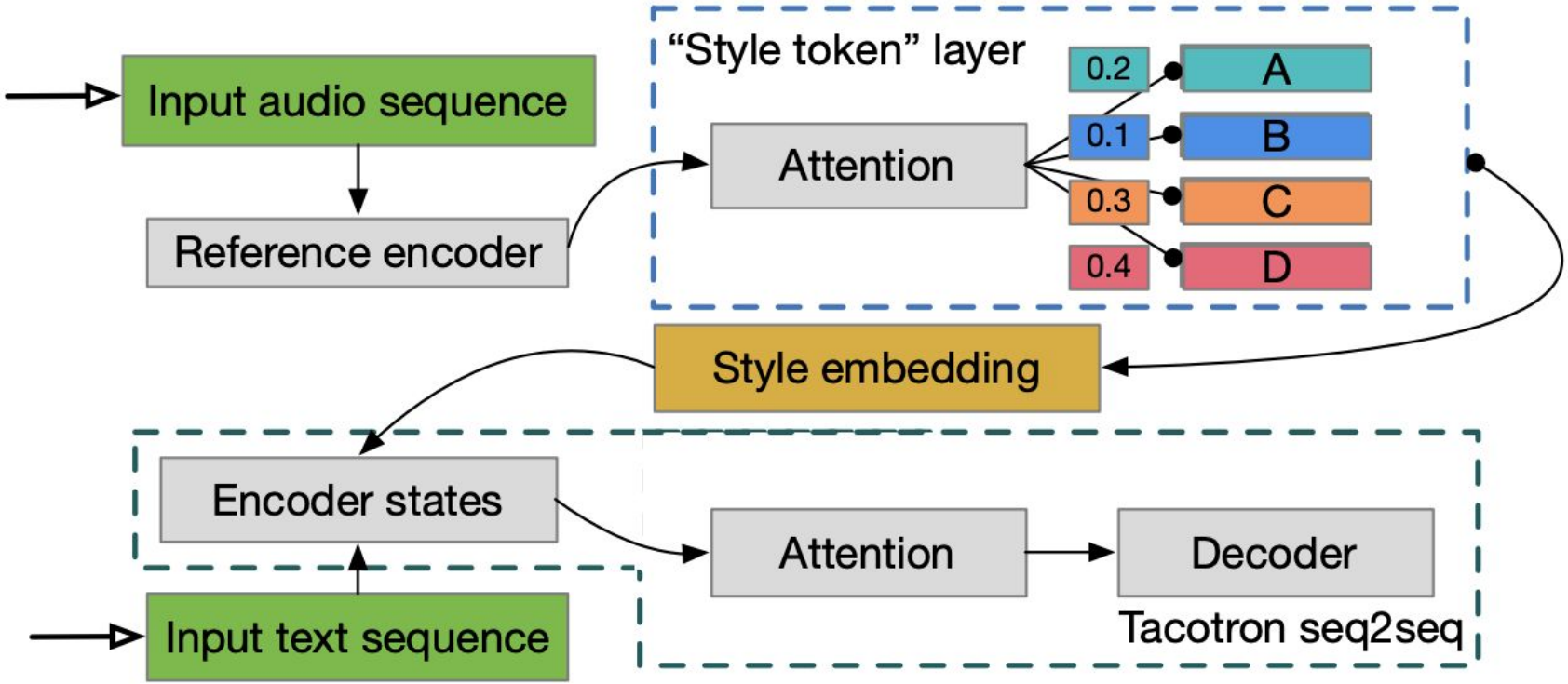
$$n \sim at, \text{ where } a \sim N/T$$

- Prompts attention matrix A to be “diagonal”



- $\mathcal{L}_{\text{att}}(A) = \mathbb{E}_{nt}[A_{nt} W_{nt}], \text{ where } W_{nt} = 1 - \exp\{-(n/N - t/T)^2 / 2g^2\}$

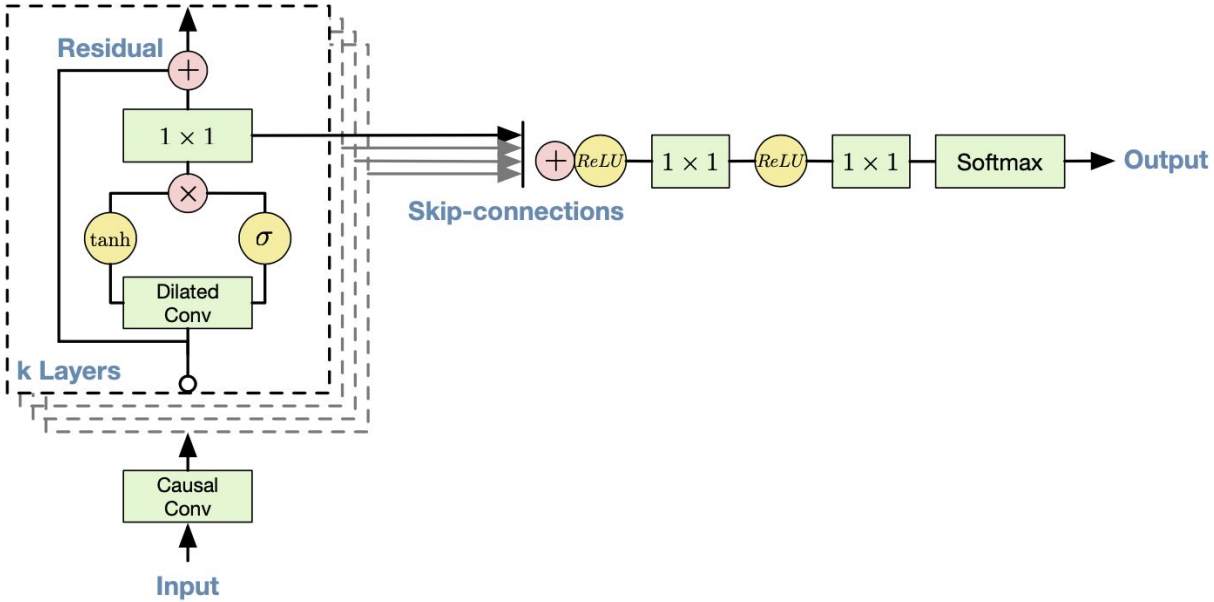
Global Style Token (GST)



WaveNet

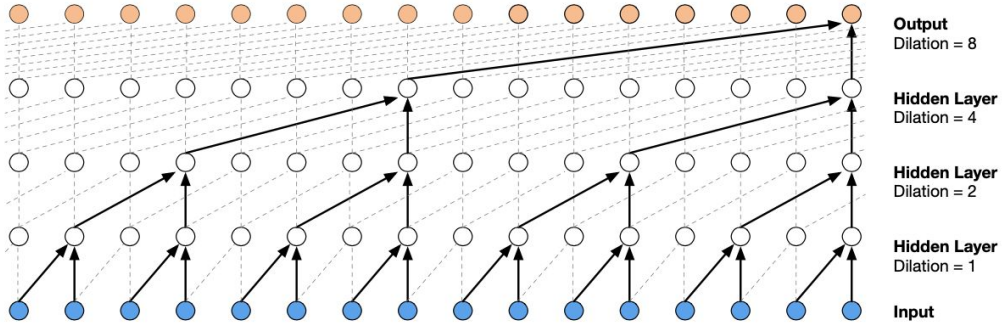
$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

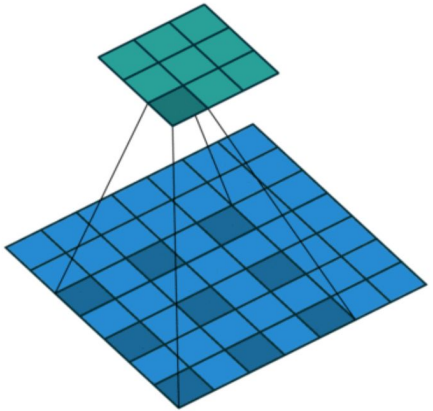
$$p(x_t | x_i, \ldots, x_{t-1}) \sim \mathrm{Cat}(\pi_\theta)$$



Dilated Convolution

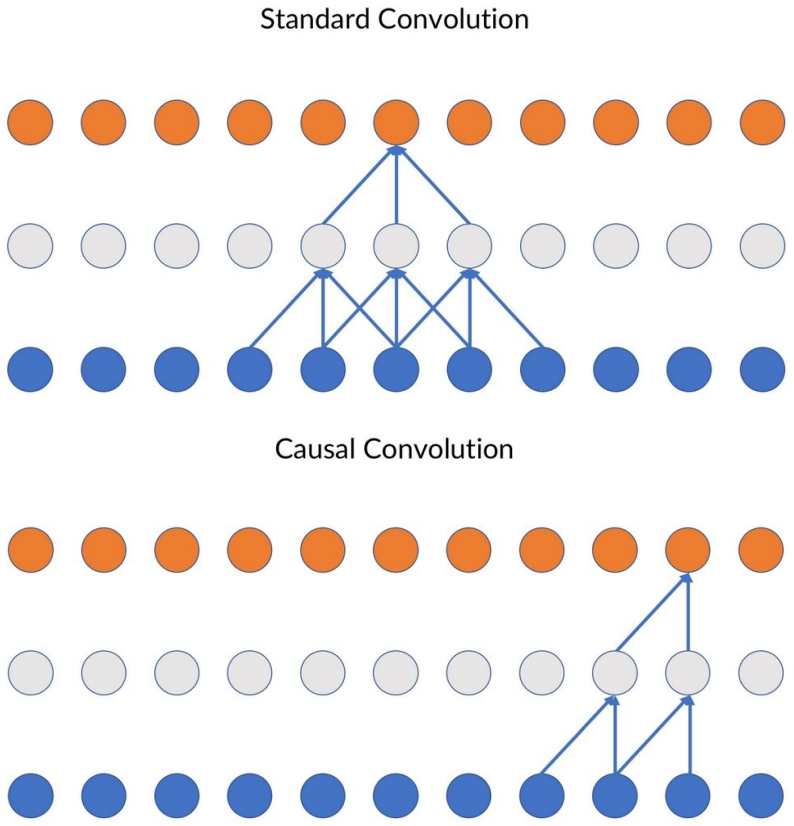
- Increase receptive field
- Allow modeling long time dependencies





Causal Convolution

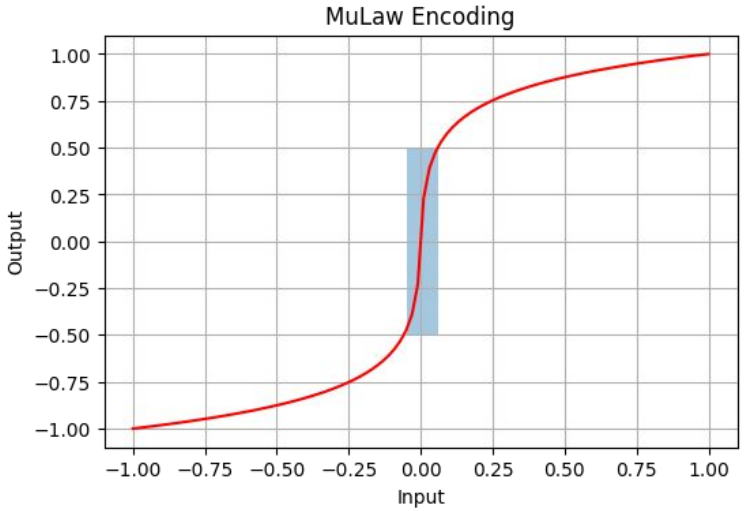
- $p(x_{t+1} \mid x_1, \ldots, x_t)$
- Don’t use padding in Conv
- Use a separate Pad



Mu Law Encoding

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

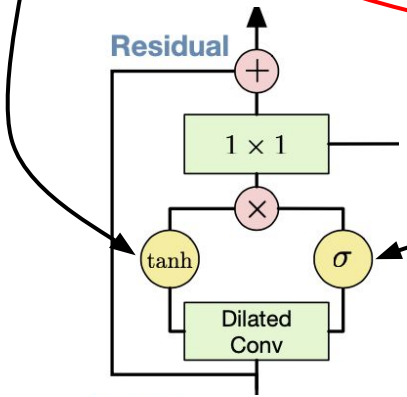
- 16-bit WAV contain 2^16 values
- Softmax will die :(
- Human hearing on a **logarithmic** scale
- **Low-amplitude** sounds in **high** resolution
- **High-amplitude** sounds in **low** resolution

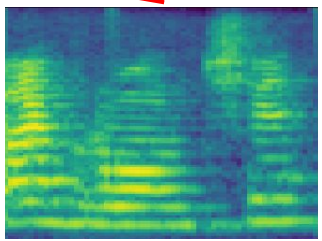


http://www.young-engineering.com/docs/YoungEngineering_ALaw_and_MuLaw_Companding.pdf

(Condition) Gated Mechanism

- $\mathbf{z} = \tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x})$
- $\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k} * \mathbf{y}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k} * \mathbf{y})$

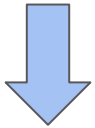




But how do we align the WAV and Mel?

But how do we align the WAV and Mel?

$$p(x_i|x_1,\ldots,x_{i-1})$$

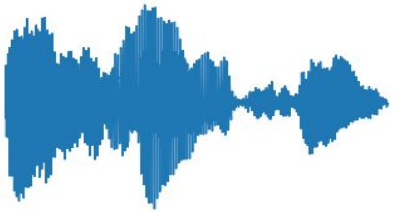


add a condition using **MelSpec**

$$p(x_i|x_1,\ldots,x_{i-1};c_1,\ldots,c_{j-1})$$

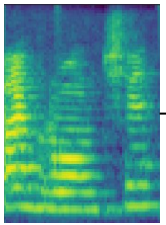
Samples and MelSpec are **not time-aligned**!

But how do we align the WAV and Mel?



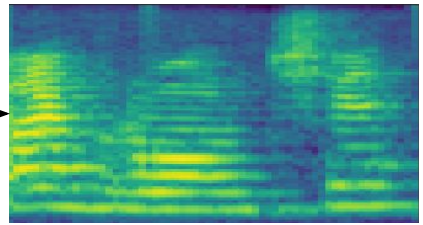
$(x_1, \ldots, x_{i-1})$

i is 256 times larger than j



$(c_1, \ldots, c_{j-1})$

Align with **Upsample**, **ConvTranspose** or something else



Upsample is our everything!

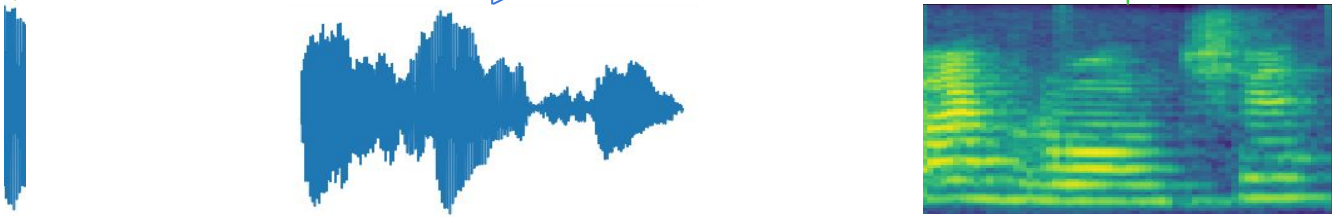

```
1  torch.nn.Upsample(hop_size, mode='linear')
```

```
Upsample(size=256, mode=linear)
```

```
1  torch.nn.ConvTranspose1d(..., stride=hop_size)
```

But how do we align the WAV and Mel?

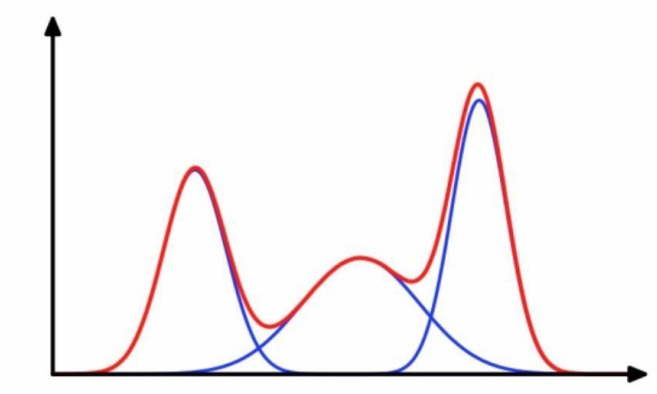
$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k} * \mathbf{y}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k} * \mathbf{y})$$



x_i $(x_1, \ldots, x_{i-1})$ $(c_1, \ldots, c_{i-1})$

What about a loss function?

- Categorical distribution
- Normal distribution
- Logistic distribution
- Mixture of Normals or Logistics
- Use torch.distributions :)



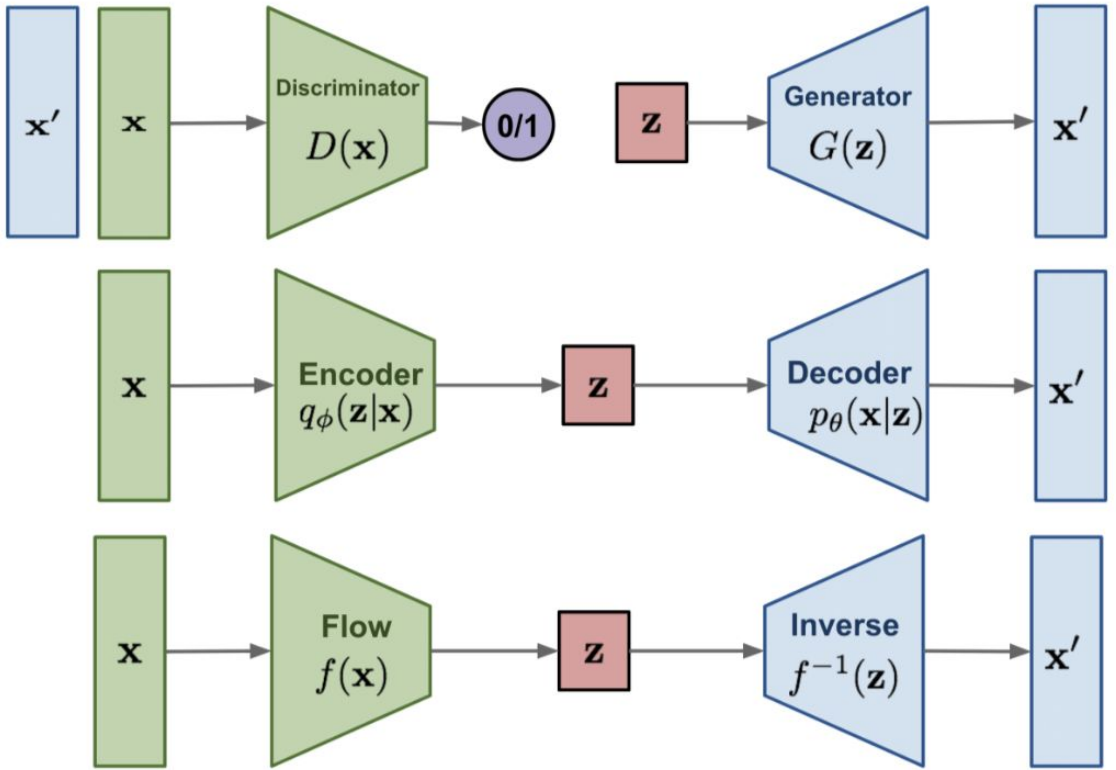
What if… we just learn to map Mels to WAVs?

Generative models

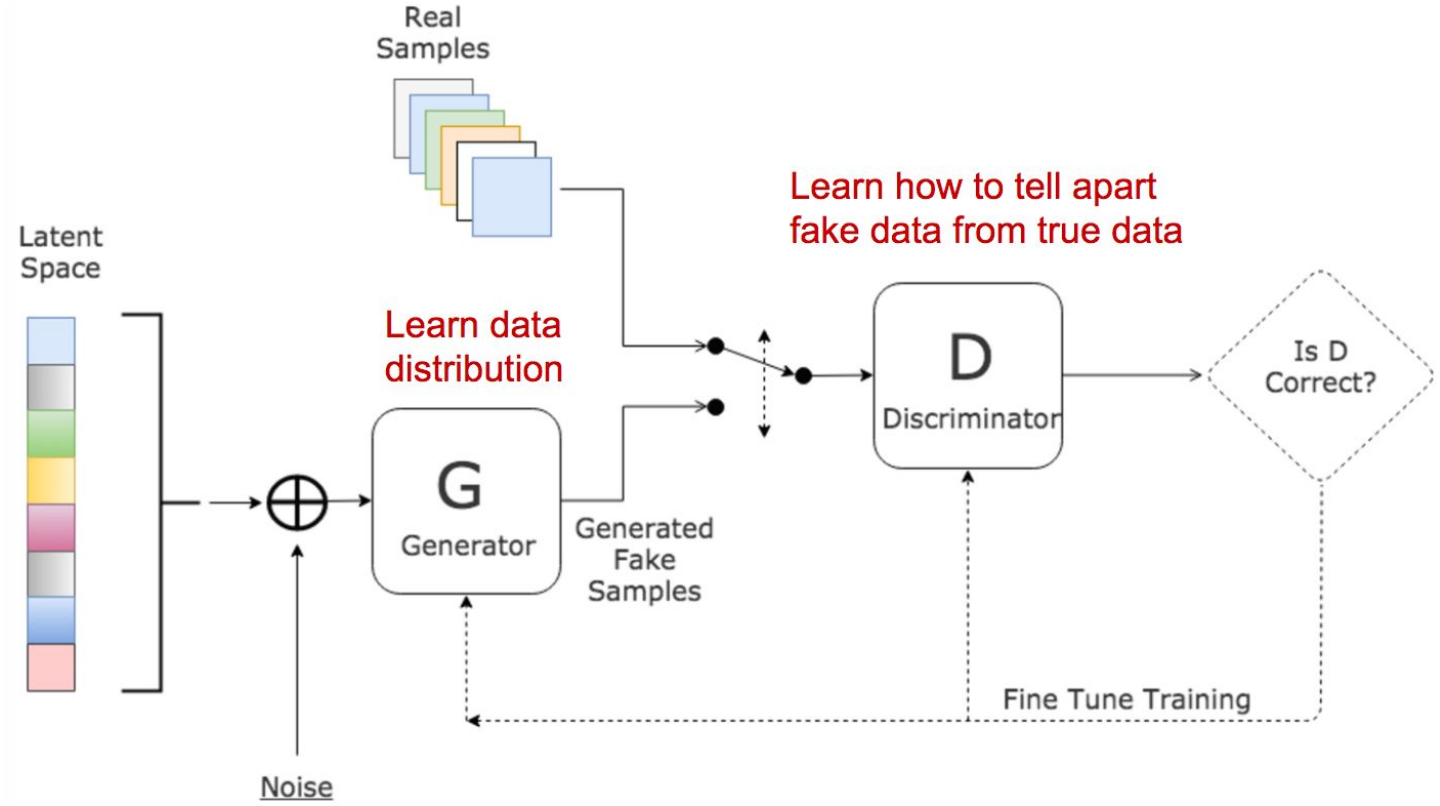
GAN: minimax the classification error loss.

VAE: maximize ELBO.

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



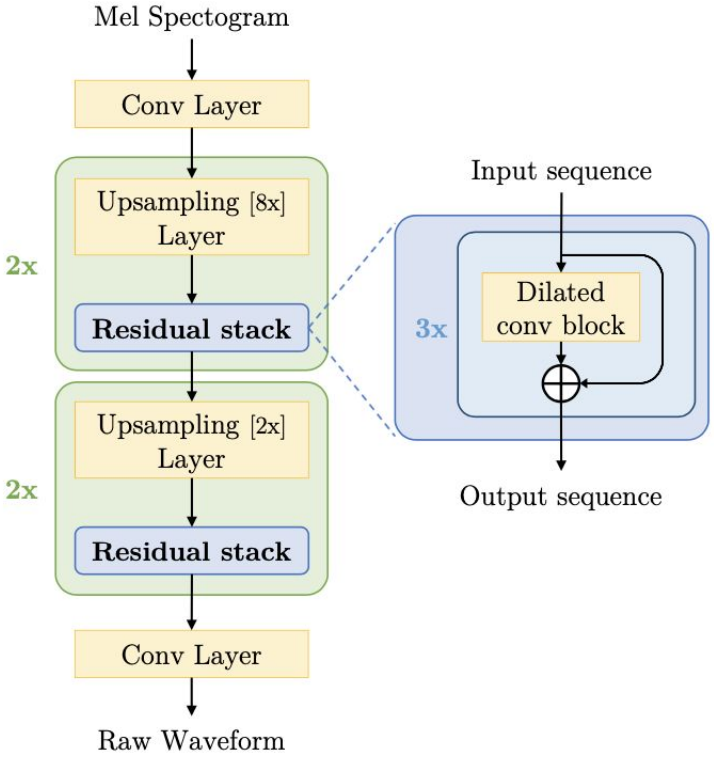
Generative Adversarial Networks

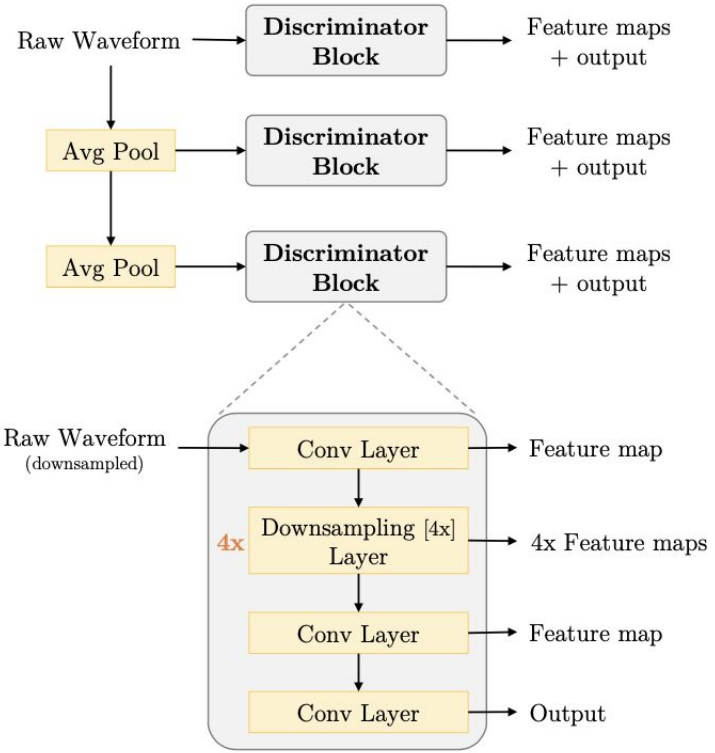


Generative Adversarial Networks

- A discriminator **D** estimates the probability of a given sample coming from the real or synthetic dataset
- A generator **G** outputs synthetic samples given a noise variable input **z**
- $$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

MelGAN


Mel Spectogram
Conv Layer
2x
Upsampling [8x] Layer
Residual stack
2x
Upsampling [2x] Layer
Residual stack
Conv Layer
Raw Waveform
Input sequence
3x
Dilated conv block
Output sequence


Raw Waveform
Discriminator Block
Feature maps + output
Avg Pool
Discriminator Block
Feature maps + output
Avg Pool
Discriminator Block
Feature maps + output
Raw Waveform (downsampled)
Conv Layer
Feature map
4x
Downsampling [4x] Layer
4x Feature maps
Conv Layer
Feature map
Conv Layer
Output

MelGAN

- Multiscale Discriminators
- Weight Normalization

$$\mathbf{w} = \frac{g}{\|\mathbf{v}\|}\mathbf{v}$$

- Markovian Discriminator
- Hinge Loss
- Feature Matching

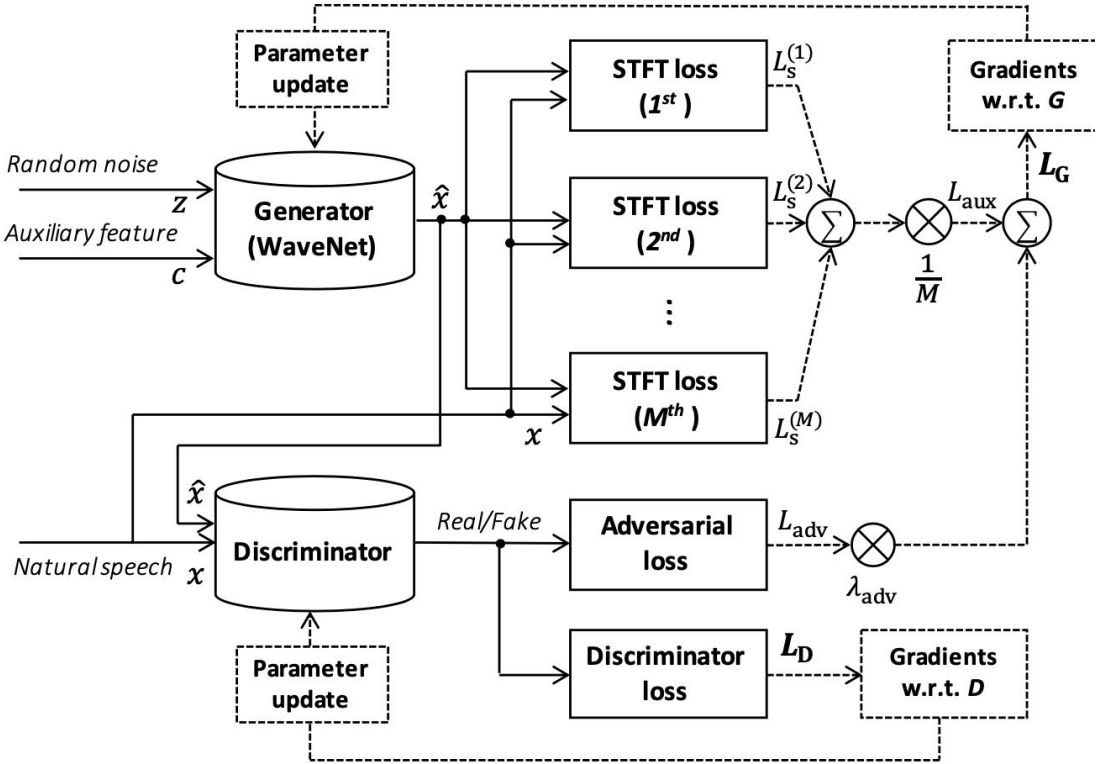
$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{x,s\sim p_{\text{data}}}\left[\sum_{i=1}^{T}\frac{1}{N_i}\left\|D_k^{(i)}(x) - D_k^{(i)}(G(s))\right\|_1\right]$$

$$\min_{D_k}\mathbb{E}_x[\min(0, 1 - D_k(x))] + \mathbb{E}_{s,z}[\min(0, 1 + D_k(G(s,z)))], \forall k = 1,2,3$$

$$\min_G \mathbb{E}_{s,z}\left[\sum_{k=1,2,3} -D_k(G(s,z))\right]$$

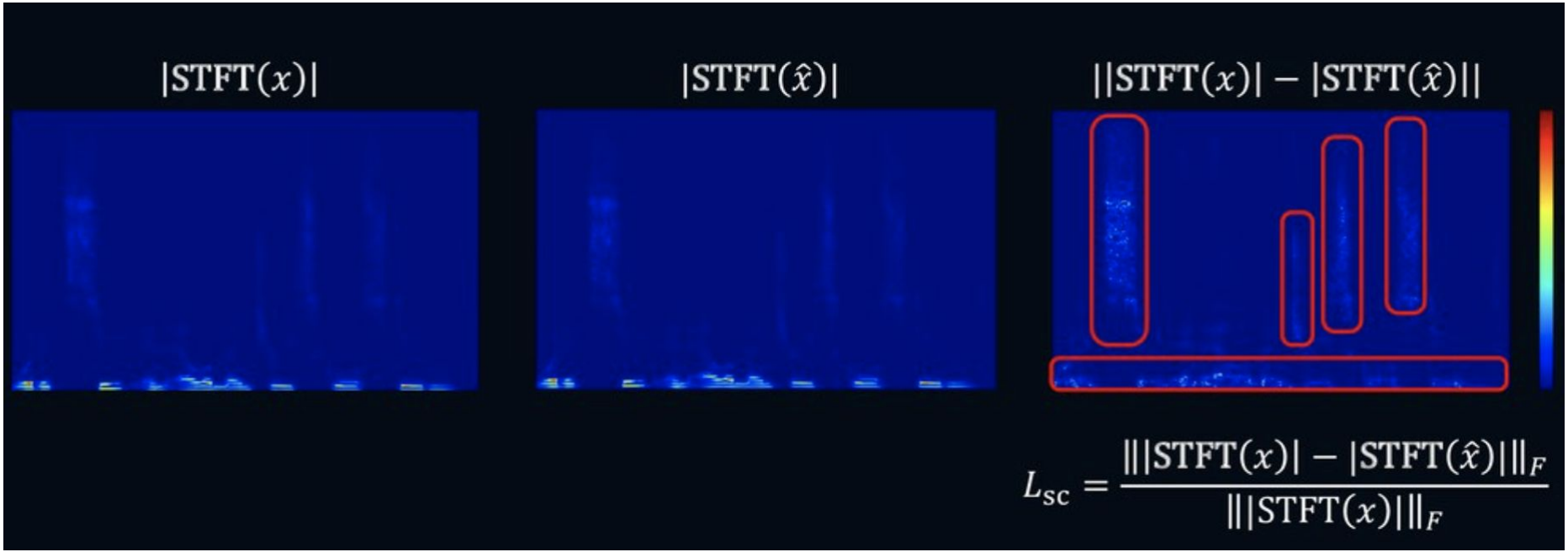
Parallel WaveGAN

- Use WaveNet based Generator
- Additionally use **multi-STFT loss**
- **LSGAN**



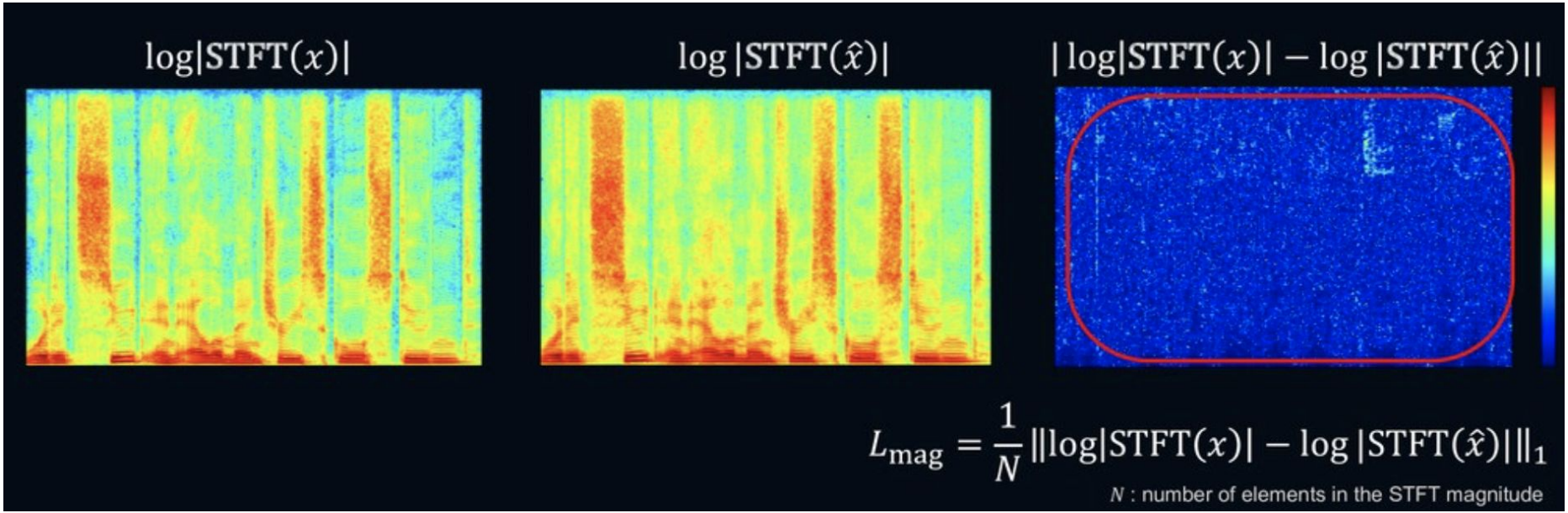
STFT Loss

Spectral Convergence part



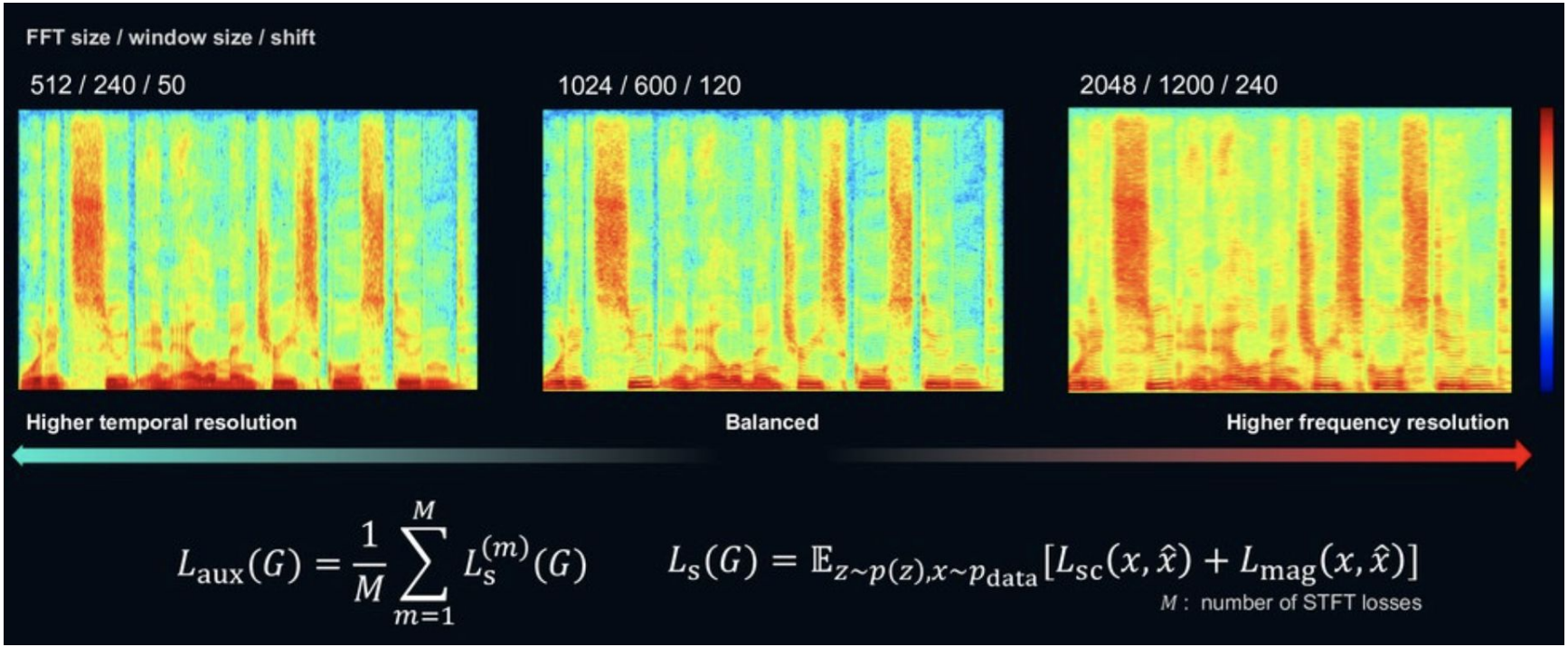
STFT Loss

Log scale STFT magnitude part



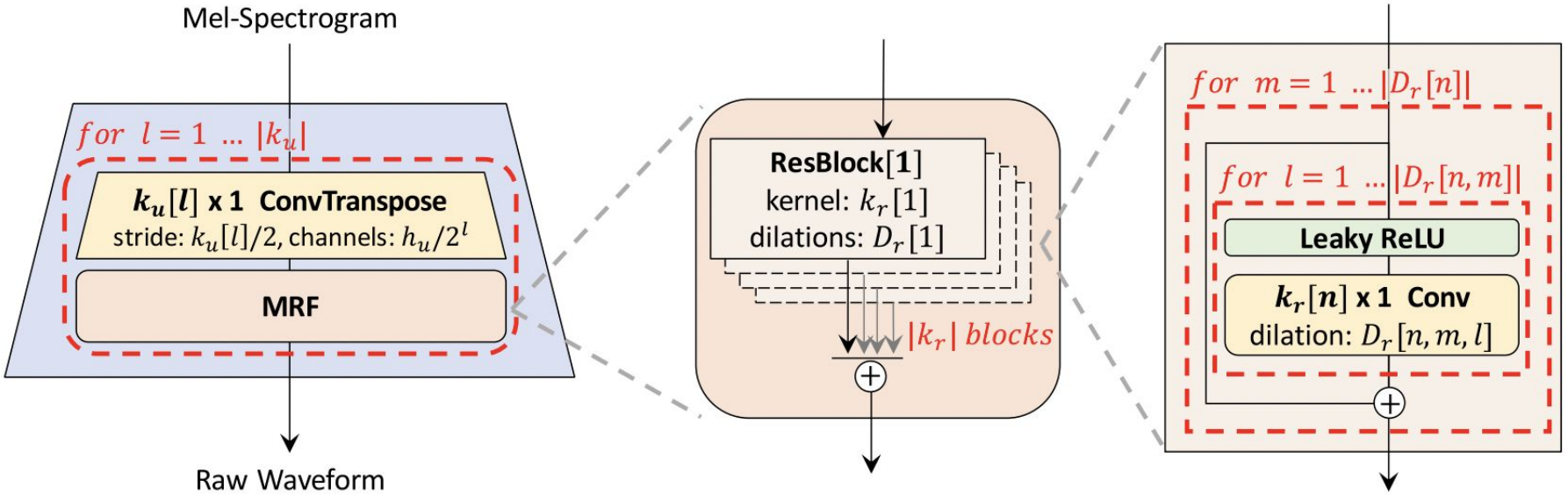
STFT Loss

Multi Resolution



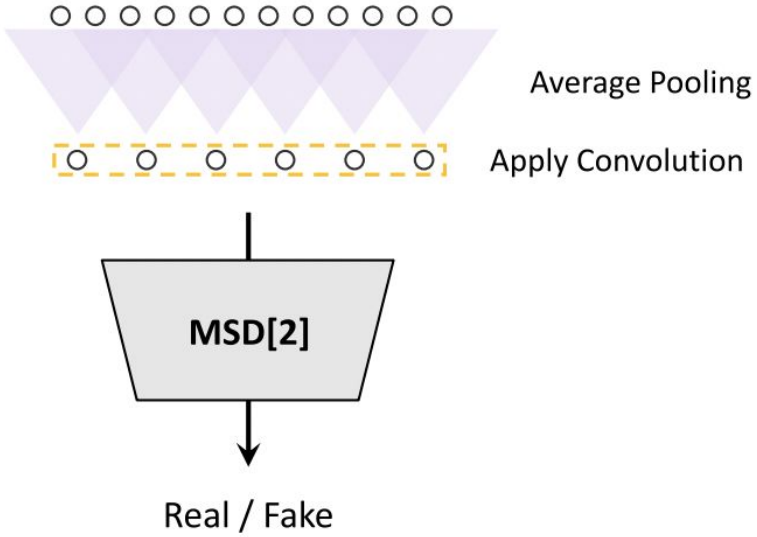
HiFi GAN

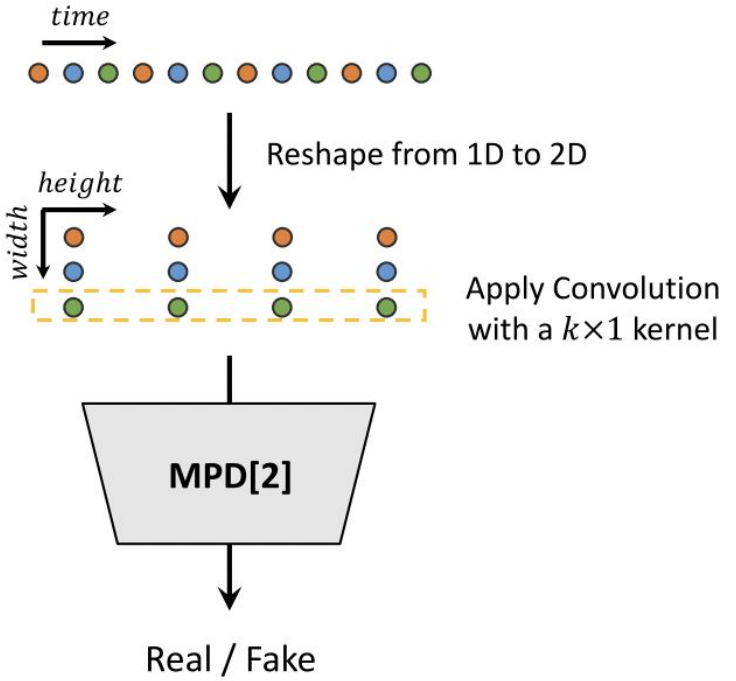
Generator



h_u	k_u	k_r	D_r
512	[16, 16, 4, 4]	[3, 7, 11]	[[1, 1], [3, 1], [5, 1]] ×3

HiFi GAN
Discriminator





HiFi GAN

Criterions

$$\mathcal{L}_{Adv}(D;G) = \mathbb{E}_{(x,s)}\left[(D(x)-1)^2 + (D(G(s)))^2\right]$$

$$\mathcal{L}_{Adv}(G;D) = \mathbb{E}_{s}\left[(D(G(s))-1)^2\right]$$

$$\mathcal{L}_{Mel}(G) = \mathbb{E}_{(x,s)}[\|\phi(x) - \phi(G(s))\|_1]$$

$$\mathcal{L}_{FM}(G;D) = \mathbb{E}_{(x,s)}\left[\sum_{i=1}^{T}\frac{1}{N_i}\left\|D^i(x) - D^i(G(s))\right\|_1\right]$$

Flow-based models

Basic idea:

- Use an invertible transformation $z = f_\theta(x)$ to transform x into z;
- Suppose that z has a simple distribution $p(z)$ (e.g. standard normal);
- Use maximum likelihood to fit θ.

$$\sum_{i=1}^{n} \log p_\theta(x_i) = \sum_{i=1}^{n} \log p(z_i) + \log \left| \det \left(\frac{dz_i}{dx_i} \right) \right| \to \max_\theta .$$

- $p(z)$ is easy to compute. But what about log-abs-det-Jacobian?

Flow-based models

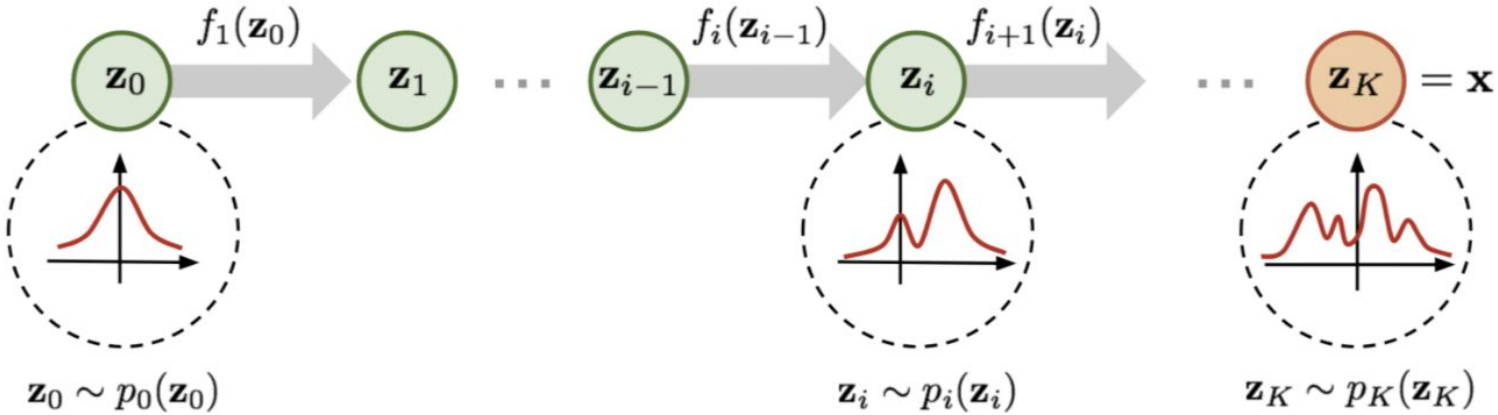
- We suppose that $f_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}$.
- If each of $f^{(j)}$ is invertible, then

$$\log \left| \det \left(\frac{dz}{dx} \right) \right| = \sum_{j=1}^{L} \log \left| \det \left(\frac{dh_j}{dh_{j-1}} \right) \right|,$$

  where $h_j = f_j(x)$, $h_0 = x$, and $h_L = z$.
- We now have to come up with invertible parametric transformations such that they have easily computable log-abs-det-Jacobian and their composition is expressive enough.
- STACK MORE LAYERS.

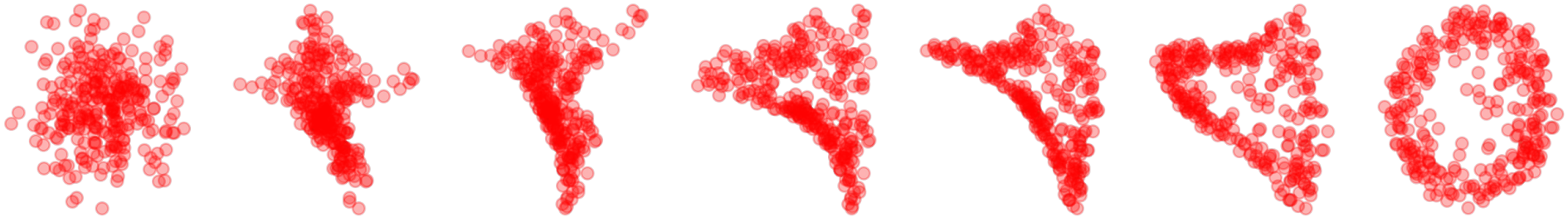
Flow-based models

- The resulting transformation f is called a *normalizing flow* or just *flow*.
- We can sample from the model just by applying the inverse transform $x = f_\theta^{-1}(z)$:



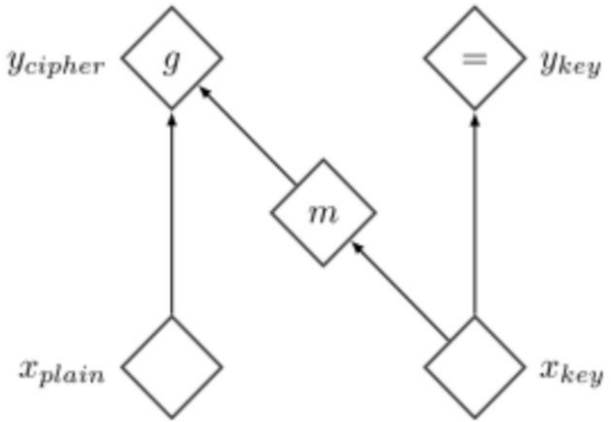
Flow-based models

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



Additive coupling

$$
\begin{aligned}
y_{I_1} &= x_{I_1} \\
y_{I_2} &= g(x_{I_2}, m(x_{I_1}))
\end{aligned}
.
$$



$$
\frac{\partial y}{\partial x} = \begin{bmatrix} I_d & 0 \\ \frac{\partial y_{I_2}}{\partial x_{I_1}} & \frac{\partial y_{I_2}}{\partial x_{I_2}} \end{bmatrix}
$$

$$
\det\left(\frac{\partial y}{\partial x}\right) = \prod_{i=1}^{n} \left(\frac{\partial y_{I_2}}{\partial x_{I_2}}\right)_{ii}
$$

Additive coupling

- [Dinh et al. (2016)] A simple extension to the previous approach allows the transform to have non-trivial Jacobian:

$$
\begin{aligned}
y_{1:d} &= x_{1:d}, \\
y_{d+1:D} &= \exp(s(x_{1:d})) \odot x_{d+1:D} + t(x_{1:d}).
\end{aligned}
$$

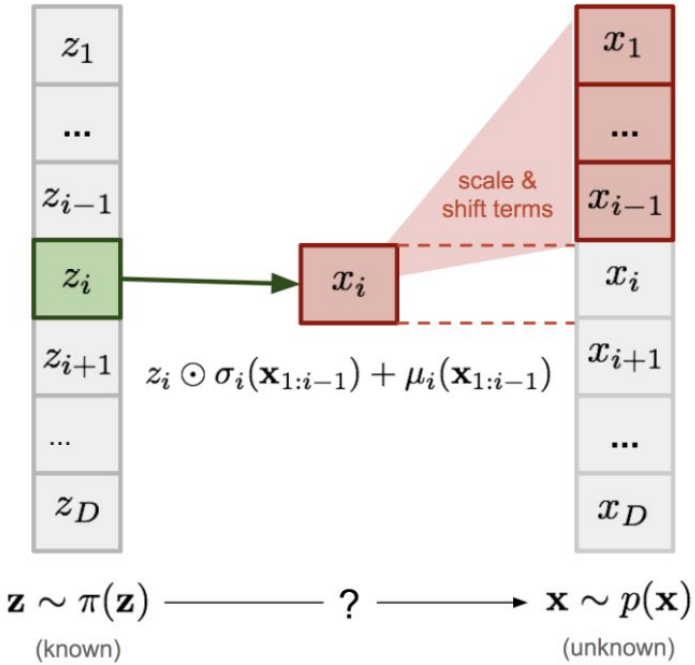
  The Log-abs-det-Jacobian is $sum(s(x_{1:d}))$ and inverse is also easy to compute.

- In case of image data, it makes sense to use CNNs for s and t and split dimensions over channels.

Autoregressive Normalizing Flows

Masked Autoregressive Flow

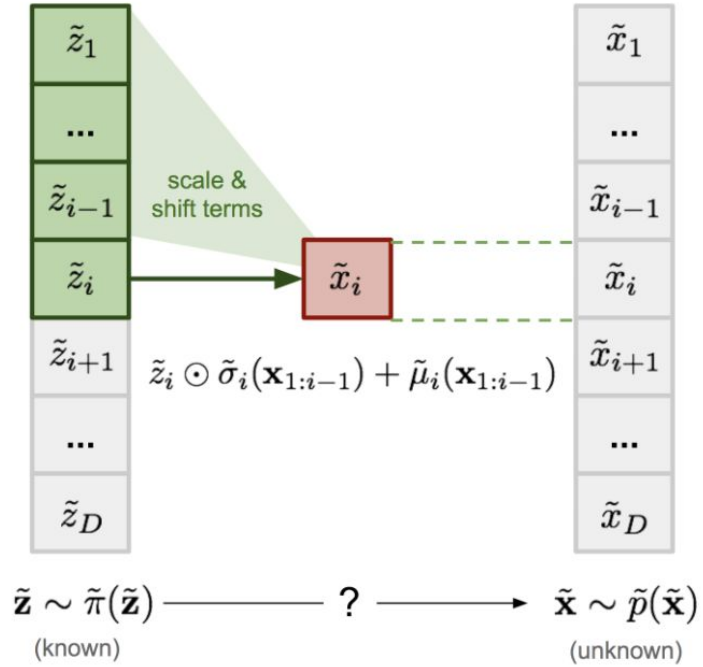
$$
\begin{aligned}
z_i &\sim \mathcal{N}(0, 1) \\
\mu_i &= f_{\mu_i}(\mathbf{x}_{1:i-1}) \\
\alpha_i &= f_{\alpha_i}(\mathbf{x}_{1:i-1}) \\
x_i &= z_i \exp \alpha_i + \mu_i \\
\left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right)\right| &= \exp\left(-\sum_i \alpha_i\right)
\end{aligned}
$$



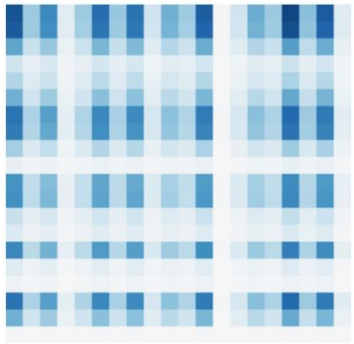
Autoregressive Normalizing Flows

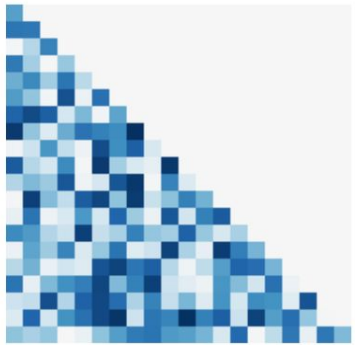
Inverse Autoregressive Flow

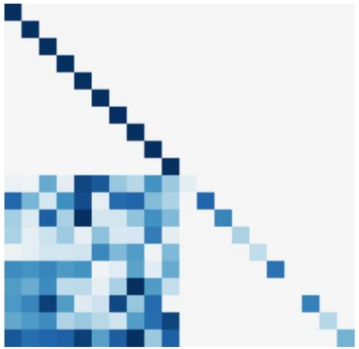
$$
\begin{aligned}
z_i &\sim \mathcal{N}(0, 1) \\
\mu_i &= f_{\mu_i}(\mathbf{z}_{1:i-1}) \\
\alpha_i &= f_{\alpha_i}(\mathbf{z}_{1:i-1}) \\
x_i &= z_i \exp \alpha_i + \mu_i \\
\left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| &= \exp \left(- \sum_i \alpha_i \right)
\end{aligned}
$$

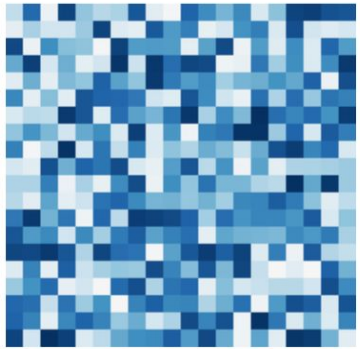


Jacobians



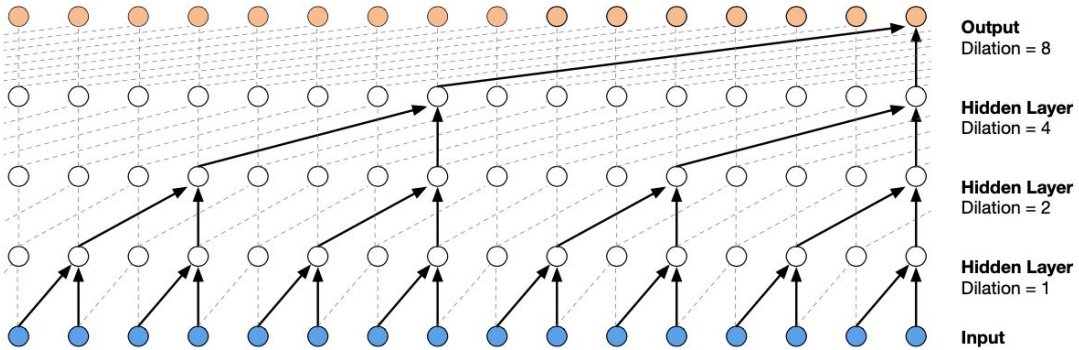






Recap WaveNet

- Dilated Convolution
- Casual Convolution
- Mu Law Encoding
- Gated Mechanism
- Global & Local Conditioning



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

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x})$$

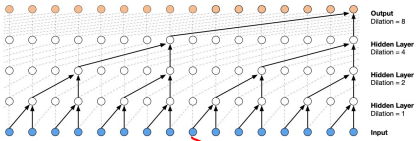
$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k} * \mathbf{y}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k} * \mathbf{y})$$

Parallel WaveNet

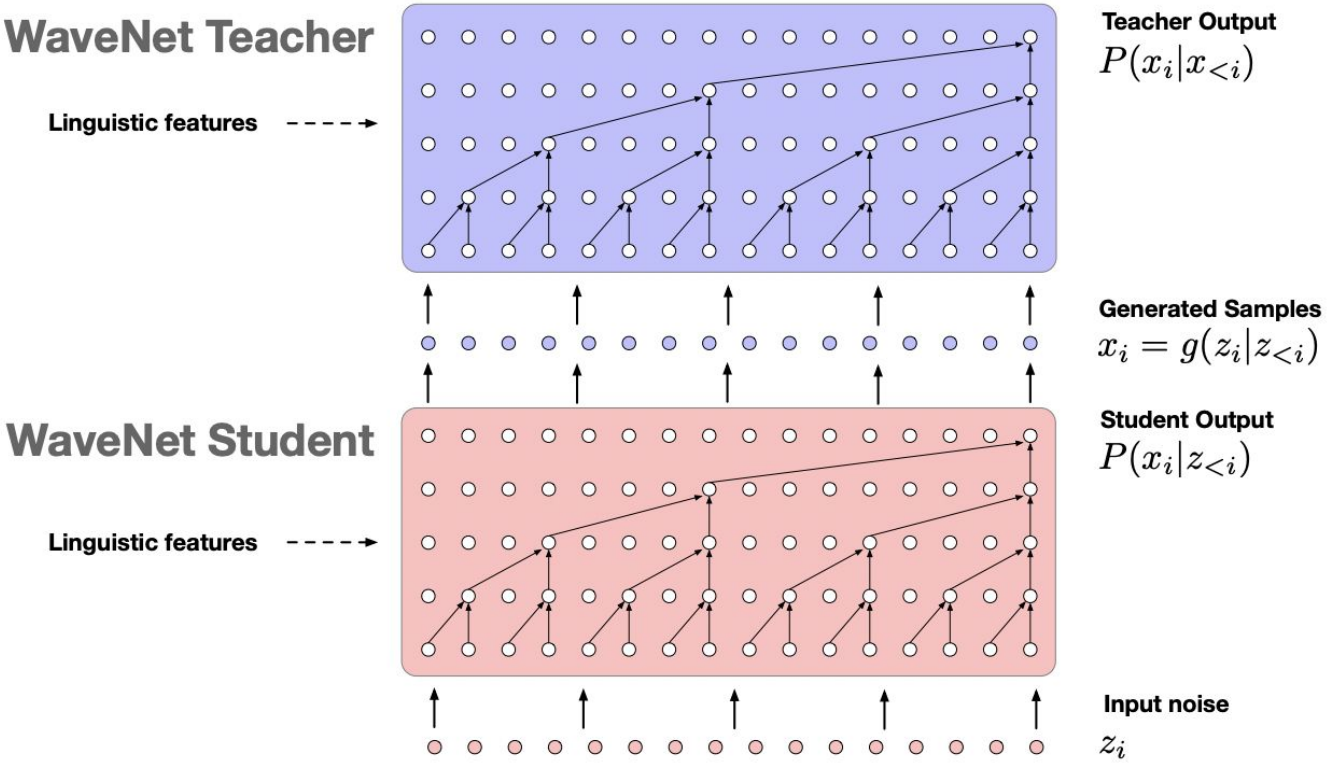
$$\boldsymbol{z} \sim \text{Logistic}(0, I)$$

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

$$p(x_t \mid \boldsymbol{z}_{<t}, \boldsymbol{\theta}) = \mathbb{L}(x_t \mid \dot{\boldsymbol{\mu}}(\boldsymbol{z}_{<t}, \boldsymbol{\theta}), \dot{\boldsymbol{s}}(\boldsymbol{z}_{<t}, \boldsymbol{\theta}))$$

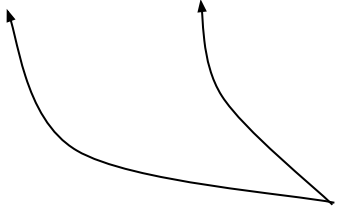


Parallel WaveNet



Parallel WaveNet

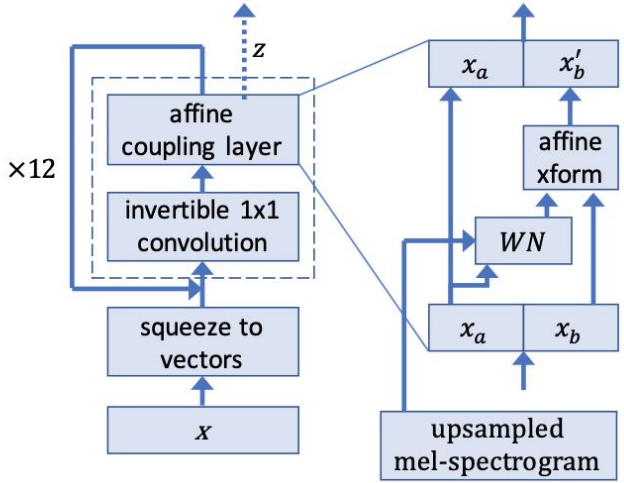
- Probability Density Distillation $D_{\mathrm{KL}}(P_S \| P_T) = H(P_S, P_T) - H(P_S)$
- Power Loss $\left\| \phi(g(\boldsymbol{z}, \boldsymbol{c})) - \phi(\boldsymbol{y}) \right\|^2$
- Perceptual Loss
- Contrastive Loss $D_{\mathrm{KL}}(P_S(\boldsymbol{c}_1) \| P_T(\boldsymbol{c}_1)) - \gamma D_{\mathrm{KL}}(P_S(\boldsymbol{c}_1) \| P_T \boldsymbol{c}_2))$



Linguistic features, speaker ID

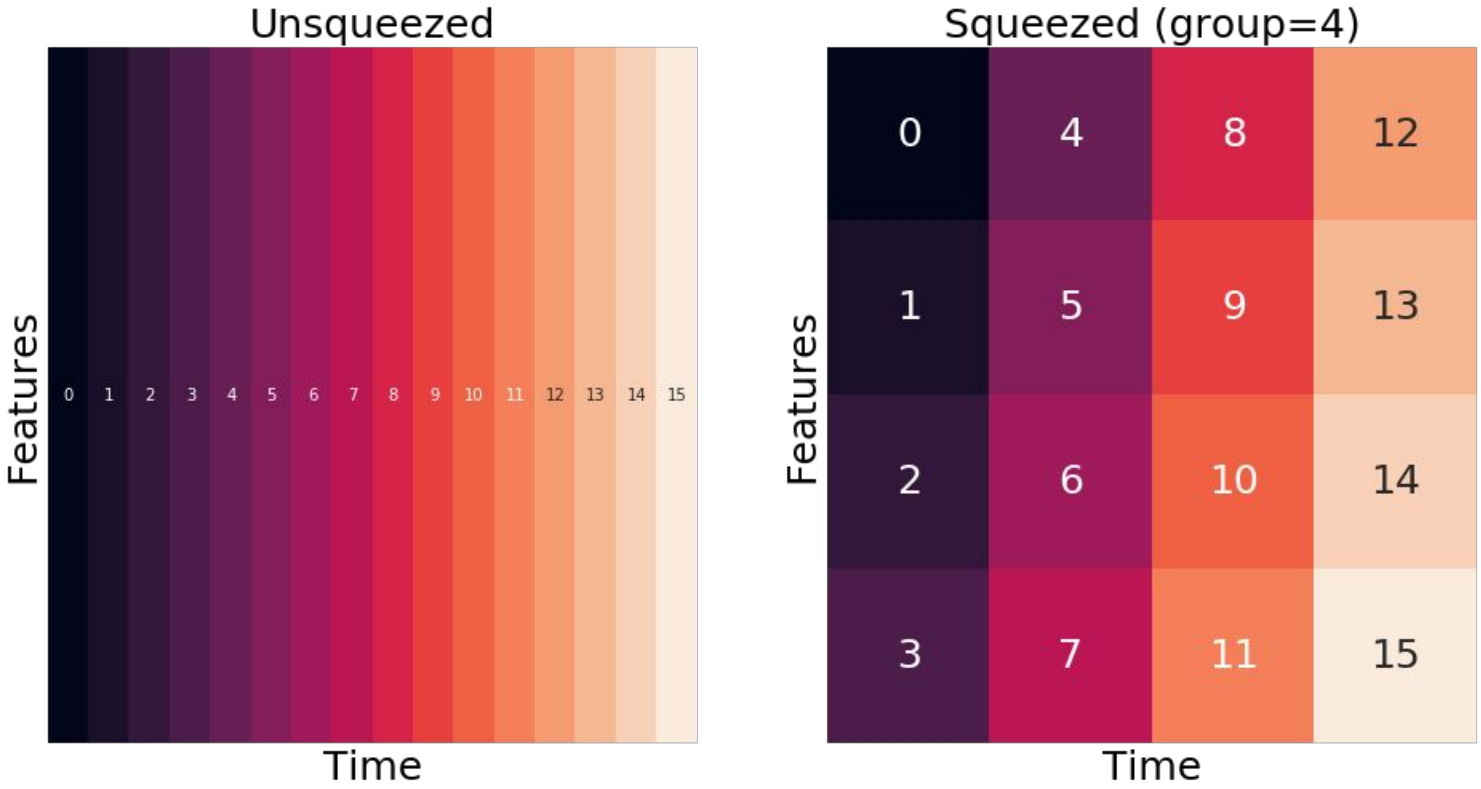
WaveGlow

- Combine insights from Glow and WaveNet
- Squeeze Operation
- Affine Couling Layer
- 1x1 Invertible Convolution
- Early outputs



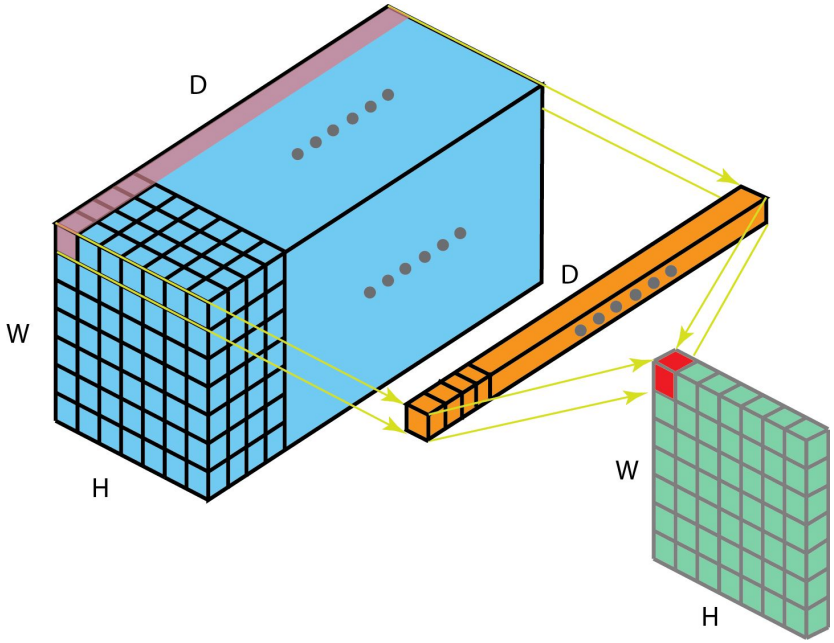
$$\boldsymbol{x}_a, \boldsymbol{x}_b = \text{split}(\boldsymbol{x})$$
$$(\log \boldsymbol{s}, \boldsymbol{t}) = \text{WaveNet}(\boldsymbol{x}_a, \text{MelSpectrogram})$$
$$\boldsymbol{x}_{b'} = \boldsymbol{s} \odot \boldsymbol{x}_b + \boldsymbol{t}$$
$$\boldsymbol{f}^{-1}_{\text{coupling}}(\boldsymbol{x}) = \text{concat}(\boldsymbol{x}_a, \boldsymbol{x}_{b'})$$

Squeeze Operation



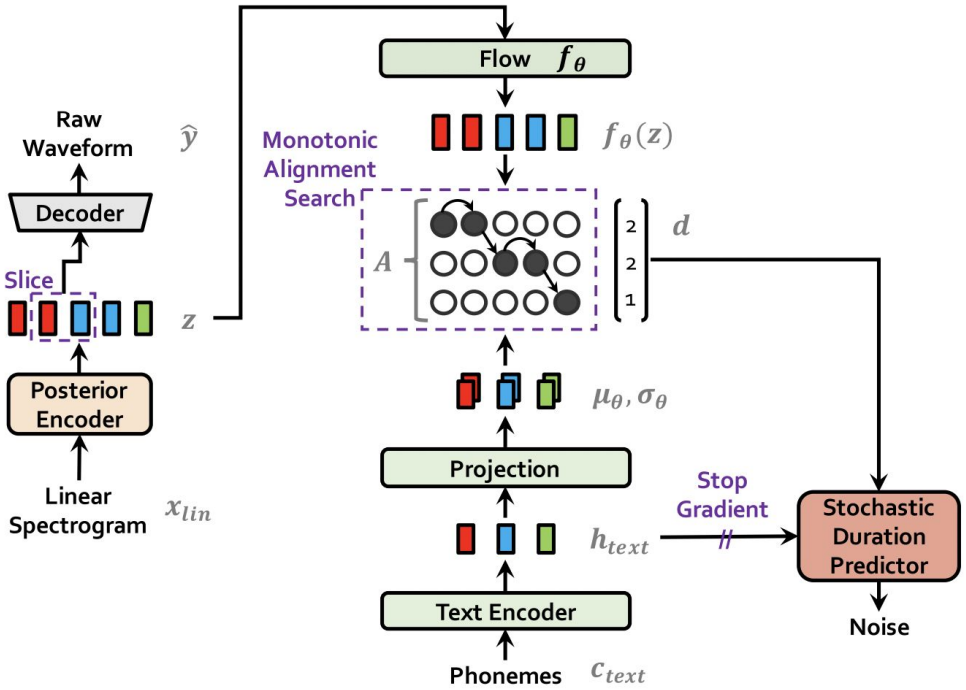
1x1 Invertible Convolution

- We want to permute **squeezed** channels
- Initialize weights as random **rotation** matrix



VITS

Training



$$L_{vae} = L_{recon} + L_{kl} + L_{dur} + L_{adv}(G) + L_{fm}(G)$$

VITS

Inference