보코더 정리

WaveNet (16.09. DeepMind)

WaveRNN (18.02. DeepMind)

WaveGlow (18.11. NVIDIA)

z ↔ x

z (isotropic Gaussian) ↔ x (audio distribution)

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; 0, \boldsymbol{I})$$
$$\boldsymbol{x} = \boldsymbol{f}_0 \circ \boldsymbol{f}_1 \circ \ldots \boldsymbol{f}_k(\boldsymbol{z})$$
$$\boldsymbol{z} = \boldsymbol{f}_k^{-1} \circ \boldsymbol{f}_{k-1}^{-1} \circ \ldots \boldsymbol{f}_0^{-1}(\boldsymbol{x})$$

Loss

$$\log p_\theta(\boldsymbol{x}) = \log p_\theta(\boldsymbol{z}) + \sum_{i=1}^{k} \log |\det(\boldsymbol{J}(\boldsymbol{f}_i^{-1}(\boldsymbol{x})))|$$

16,000 samples

GANSynth (19.02. Google AI)

Authors

Jesse Engel: **Music Guy**

Kumar Krishna Agrawal

Shuo Chen

Ishaan Gulrajani: **WGAN-GP**

Chris Donahue: **GAN + Audio + Music (Adversarial Audio Synthesis etc.)**

Adam Roberts

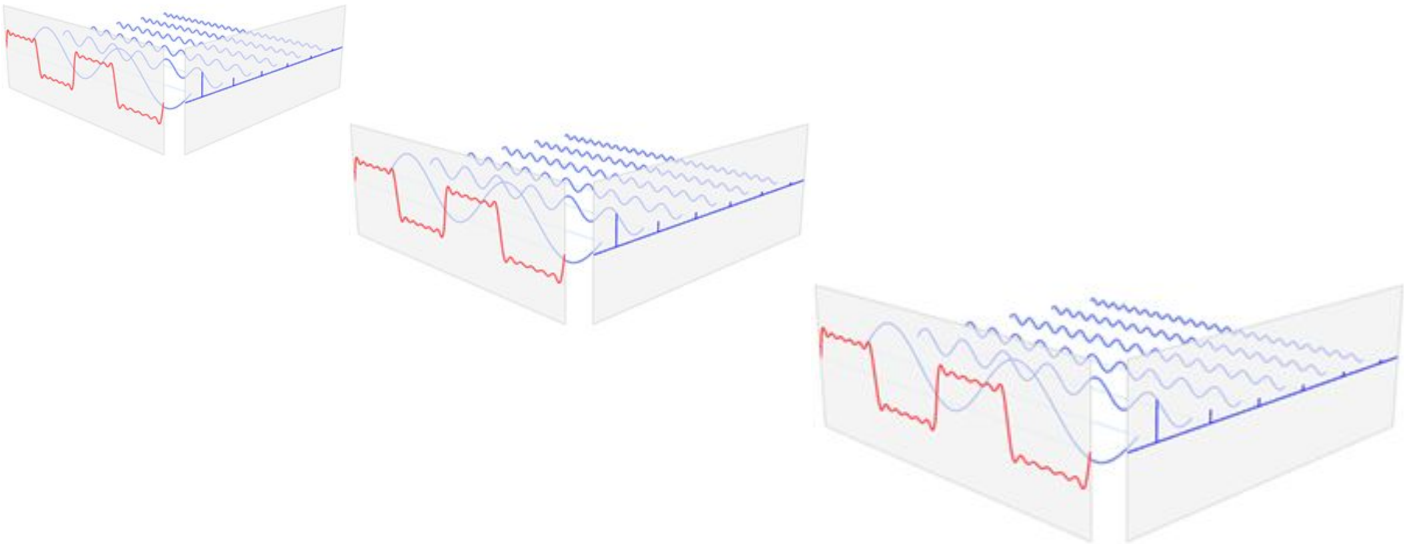
Goal

Explore effective audio representations for non-causal convolutional generation

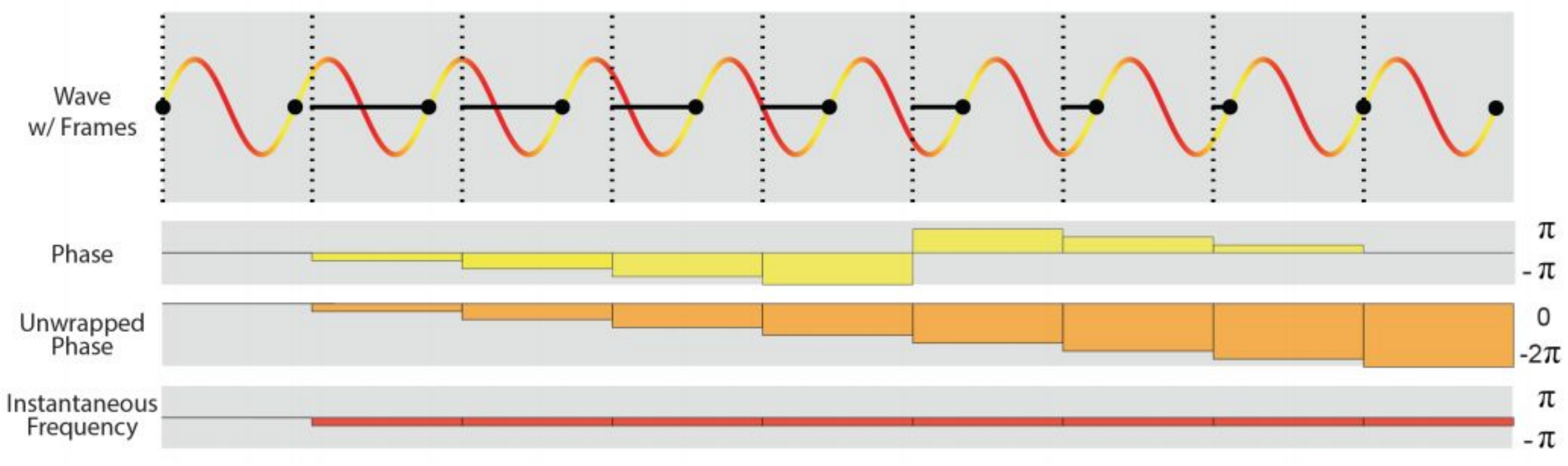
Spectrogram (magnitude & phase)

Blue bar: **mag**

Starting point within each frame: **phase**



Three representations of Phase



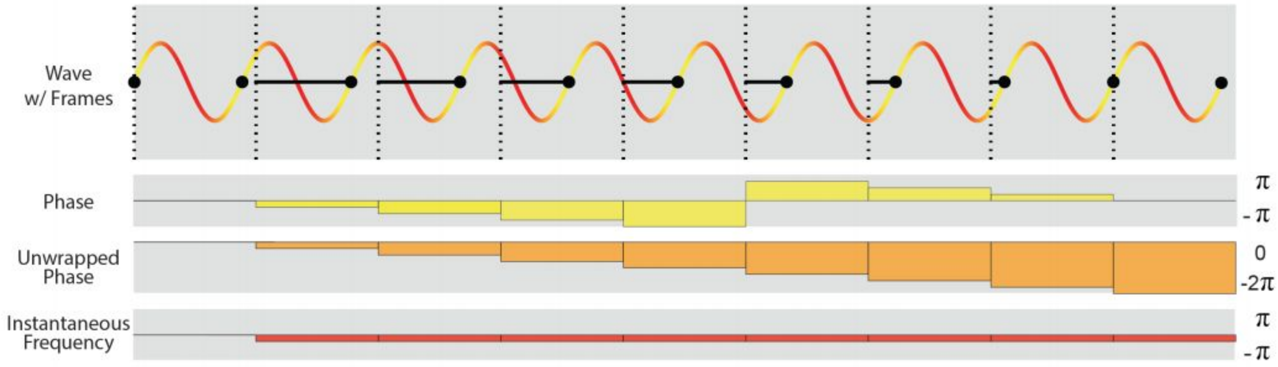
Three representations of Phase

Phase: alignment between the two processes

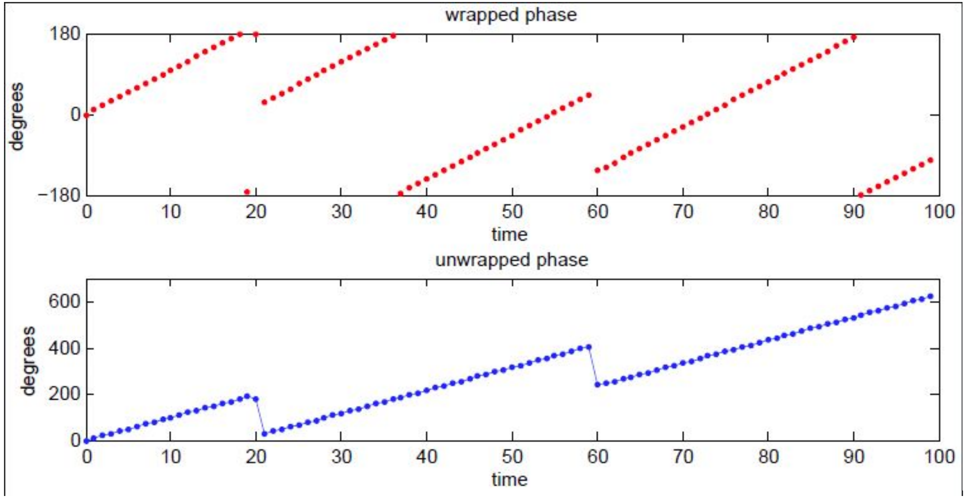
Unwrapped phase: unwrapped over 2pi boundary

Instantaneous frequency: derivative of unwrapped phase

Constant relationship between audio frequency and frame frequency



Phase Unwrapped



Inputs: Magnitude spectrum $v \in \mathbb{R}_+^F$,

phase in the previous time frame $\phi' \in \mathbb{R}^F$.

Peak localization f_p from v.

Frequencies ν_p with QIFFT on v around f_p.

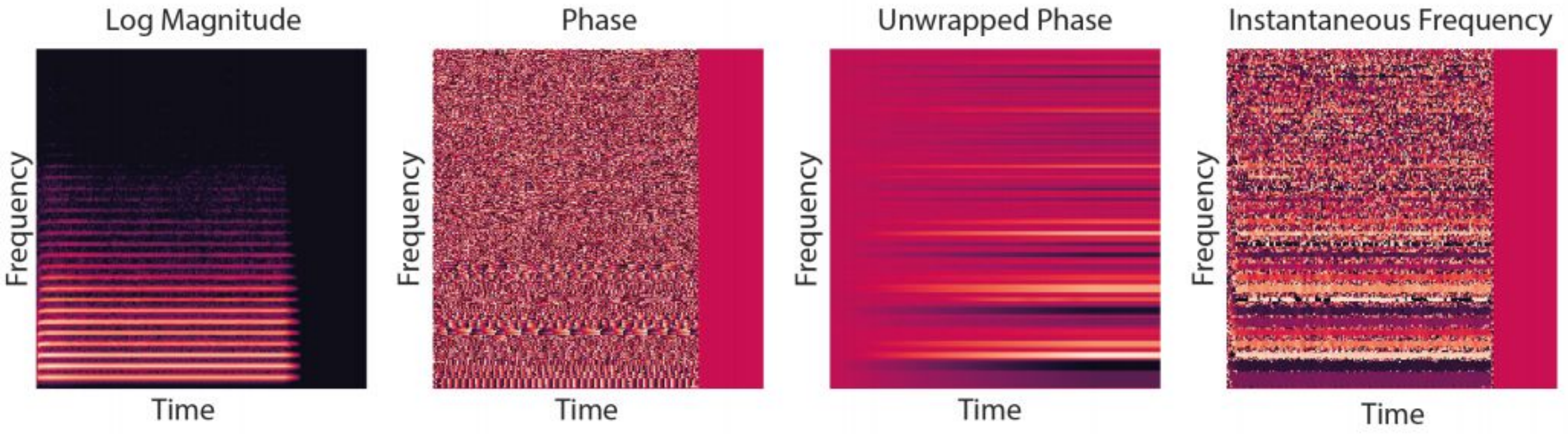
Regions of influence I_p and $\forall f \in I_p, \nu(f) = \nu_p$.

Phase unwrapping $\phi = \phi' + 2\pi S\nu$.

Output: $\phi \in \mathbb{R}^F$

Algorithm 1 Phase unwrapping

Magnitude & phase



Challenges

Human sensitive to discontinuities/irregularities in periodic signal

should maintain the regularity over short to mid timescale (1ms - 100ms)

Magnitudes (or conv filters) are not enough

Should take into account the phase

Data: NSynth

var:	: instruments, pitches, timbres, volumes
dur	: 4s
sr	: 16khz
total	: 300,000
subset	: 70,379 (32 - 1,000hz, acoustic, 80/20)

STFT

win_size = 1024 or 2048

hop_size = 256 or 512

log mag scaled to -1~1 (G 끝단의 tanh)

phase scaled to -1~1 (ver. 2 unwrapped, ver. 3 instantaneous frequency)

Maximum range over 100 examples shifted and scaled to [-0.8, 0.8]

shape = [256, 512, 2] or [128, 1024, 2], (time-pad at the end, nyquist trim)

Optionally to mel frequency WITHOUT compression (app. inv lin transf.)

Model

Progressive GAN (ACGAN) + pitch conditioning

append a one-hot representation of musical pitch to the latent vector

add an auxiliary classification loss to the discriminator

Model

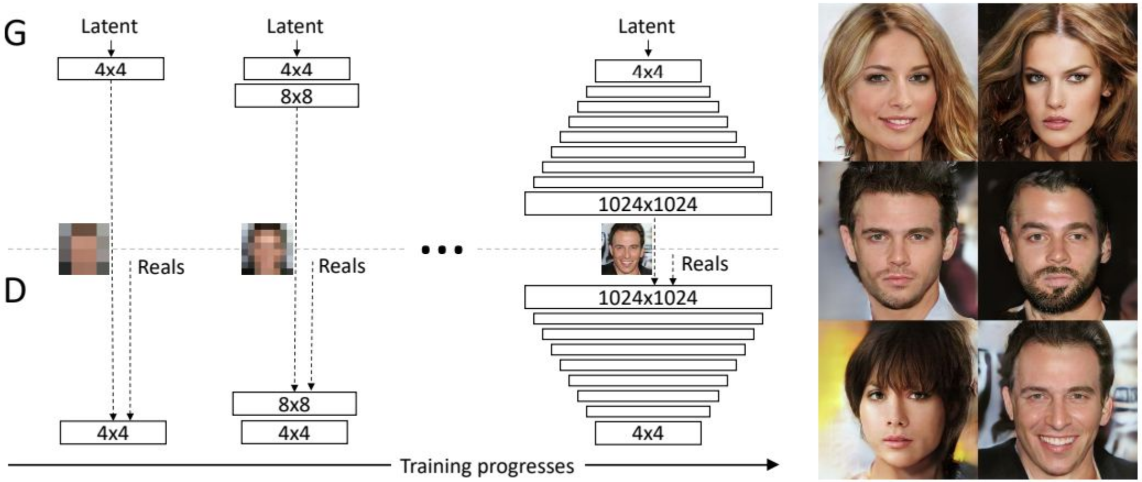


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

Tricks

Gradient penalty to promote Lipschitz continuity

Pixel normalization at each layer

$$x = x_{nhwc} / \left(\frac{1}{C} \sum_c x_{nhwc}^2\right)^{0.5}$$

Progressive training

Generator

Sample z from a spherical Gaussian

Upsample (a stack of transposed convolutions)

Pixelwise normalization

Generator

Generator	Output Size	k_{Width}	k_{Height}	$k_{Filters}$	Nonlinearity
concat(Z, Pitch)	(1, 1, 317)	-	-	-	-
conv2d	(2, 16, 256)	2	16	256	PN(LReLU)
conv2d	(2, 16, 256)	3	3	256	PN(LReLU)
upsample 2x2	(4, 32, 256)	-	-	-	-
conv2d	(4, 32, 256)	3	3	256	PN(LReLU)
conv2d	(4, 32, 256)	3	3	256	PN(LReLU)
upsample 2x2	(8, 64, 256)	-	-	-	-
conv2d	(8, 64, 256)	3	3	256	PN(LReLU)
conv2d	(8, 64, 256)	3	3	256	PN(LReLU)
upsample 2x2	(16, 128, 256)	-	-	-	-
conv2d	(16, 128, 256)	3	3	256	PN(LReLU)
conv2d	(16, 128, 256)	3	3	256	PN(LReLU)
upsample 2x2	(32, 256, 256)	-	-	-	-
conv2d	(32, 256, 128)	3	3	128	PN(LReLU)
conv2d	(32, 256, 128)	3	3	128	PN(LReLU)
upsample 2x2	(64, 512, 128)	-	-	-	-
conv2d	(64, 512, 64)	3	3	64	PN(LReLU)
conv2d	(64, 512, 64)	3	3	64	PN(LReLU)
upsample 2x2	(128, 1024, 64)	-	-	-	-
conv2d	(128, 1024, 32)	3	3	32	PN(LReLU)
conv2d	(128, 1024, 32)	3	3	32	PN(LReLU)
generator output	(128, 1024, 2)	1	1	2	Tanh

Sampling z

1 Set μ (기준 벡터)

2 어떤 다른 유닛 벡터 v는 μ와 이루는 각의 크기가 θ일 때

1−cosθ는 standard gaussian을 따른다.

Discriminator

Downsample (mirroring convolutions)

Measure a divergence between the real and generated distributions

Discriminator

Discriminator					
image	(128, 1024, 2)	-	-	-	-
conv2d	(128, 1024, 32)	1	1	32	-
conv2d	(128, 1024, 32)	3	3	32	LReLU
conv2d	(128, 1024, 32)	3	3	32	LReLU
downsample 2x2	(64, 512, 32)	-	-	-	-
conv2d	(64, 512, 64)	3	3	64	LReLU
conv2d	(64, 512, 64)	3	3	64	LReLU
downsample 2x2	(32, 256, 64)	-	-	-	-
conv2d	(32, 256, 128)	3	3	128	LReLU
conv2d	(32, 256, 128)	3	3	128	LReLU
downsample 2x2	(16, 128, 128)	-	-	-	-
conv2d	(16, 128, 256)	3	3	256	LReLU
conv2d	(16, 128, 256)	3	3	256	LReLU
downsample 2x2	(8, 64, 256)	-	-	-	-
conv2d	(8, 64, 256)	3	3	256	LReLU
conv2d	(8, 64, 256)	3	3	256	LReLU
downsample 2x2	(4, 32, 256)	-	-	-	-
conv2d	(4, 32, 256)	3	3	256	LReLU
conv2d	(4, 32, 256)	3	3	256	LReLU
downsample 2x2	(2, 16, 256)	-	-	-	-
concat(x, minibatch std.)	(2, 16, 257)	-	-	-	-
conv2d	(2, 16, 256)	3	3	256	LReLU
conv2d	(2, 16, 256)	3	3	256	LReLU
pitch classifier	(1, 1, 61)	-	-	61	Softmax
discriminator output	(1, 1, 1)	-	-	1	-

Training

ADAM

lr: 2e-4, 4e-4, **8e-4**

pitch λ: 0.1, 1.0, **10.0**

bs: 8

4,5 days on a single V100

Comparison

Baselines: WaveGAN (GAN), WaveNet (AR)

Model variations:

Phase or IF

Linear or MEL

Normal (256, 512, 2) or H (128, 1024, 2)

Result

54,000 times faster than WaveNet

Result

High > Normal

Mel > Linear

IF > Phase

* low frequency resolution matters

* phase matters

Examples	Human Eval (wins)	NDB	FID	IS	PA	PE
Real Data	549	2.2	13	47.1	98.2	0.22
IF-Mel + H	**485**	**29.3**	167	38.1	97.9	0.40
IF + H	308	36. 0	**104**	**41.6**	**98.3**	**0.32**
Phase + H	225	37.6	592	36.2	97.6	0.44
IF-Mel	479	37.0	600	29.6	94.1	0.63
IF	283	37.0	708	36.3	96.8	0.44
Phase	203	41.4	687	24.4	94.4	0.77
WaveNet	359	45.9	320	29.1	92.7	0.70
WaveGAN	216	43.0	461	13.7	82.7	1.40

Result: More Diverse

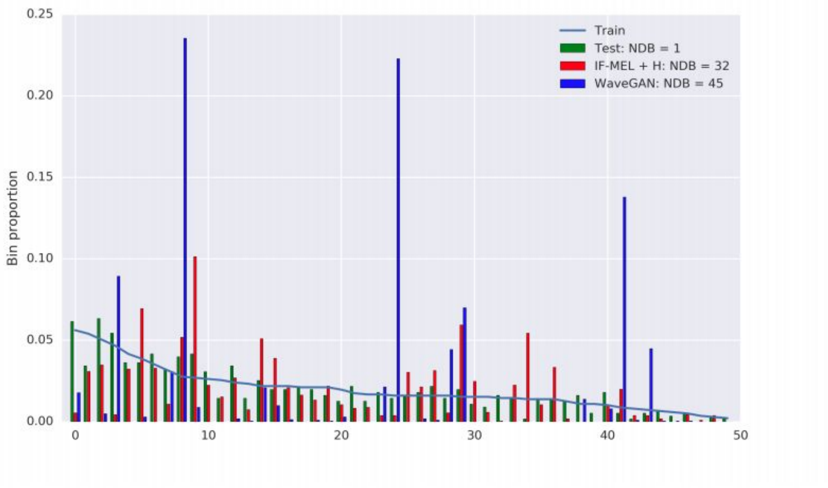


Figure 5: NDB bin proportions for the IF-Mel + H model and the WaveGAN baseline (evaluated with examples of pitch 60).

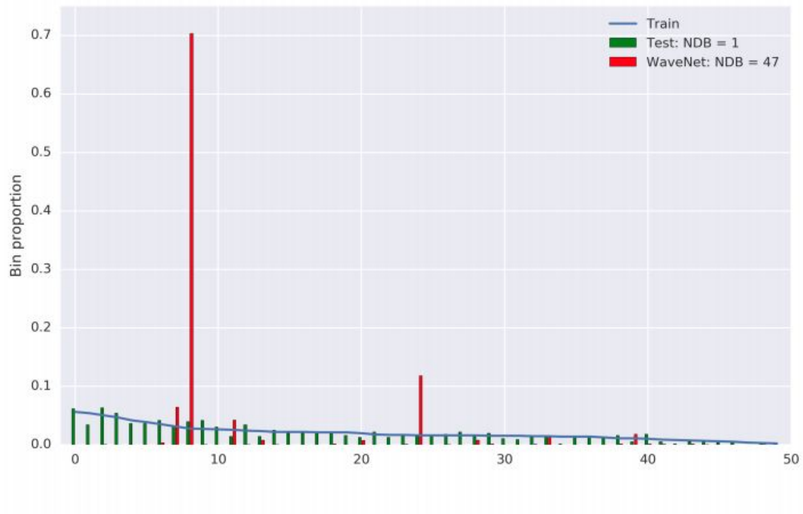


Figure 6: NDB bin proportions for the WaveNet baseline (evaluated with examples of pitch 60).

MelGAN (19.12., NIPS 2019)

Noticeable Differences

Generator

- No z
- Weight normalization (> instance normalization, spectral normalization)
- Inductive bias (dilation)
- Checkerboard artifacts (smart upsampling+dilation)

Discriminator

- Three discriminators
- Window-based loss
- Feature matching loss

Generator

Fully convolutional

Input:

mel spectrogram (256x lower temporal resolution than audio)

NO z: conditioning info is strong enough

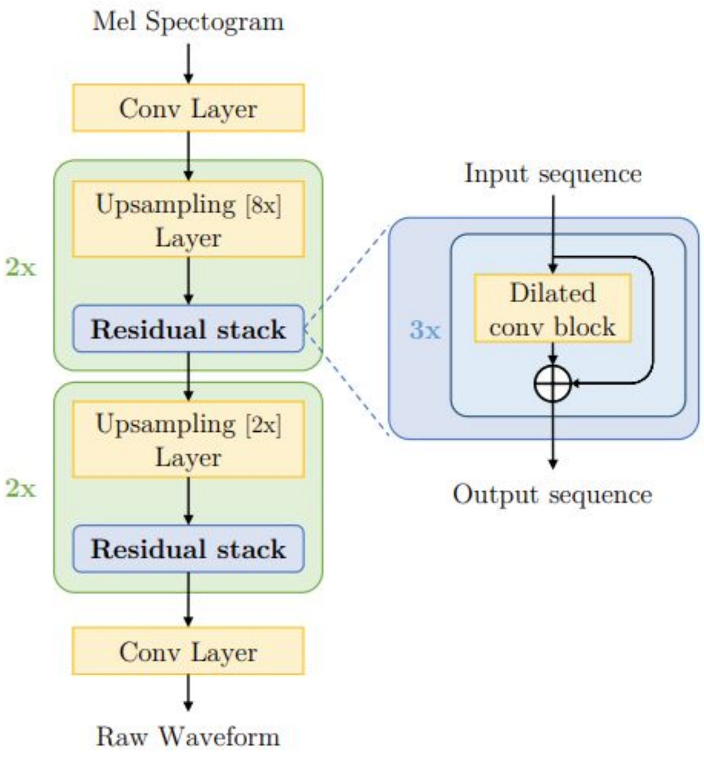
Output:

Waveform

Total # of params:

4,266,050

Generator



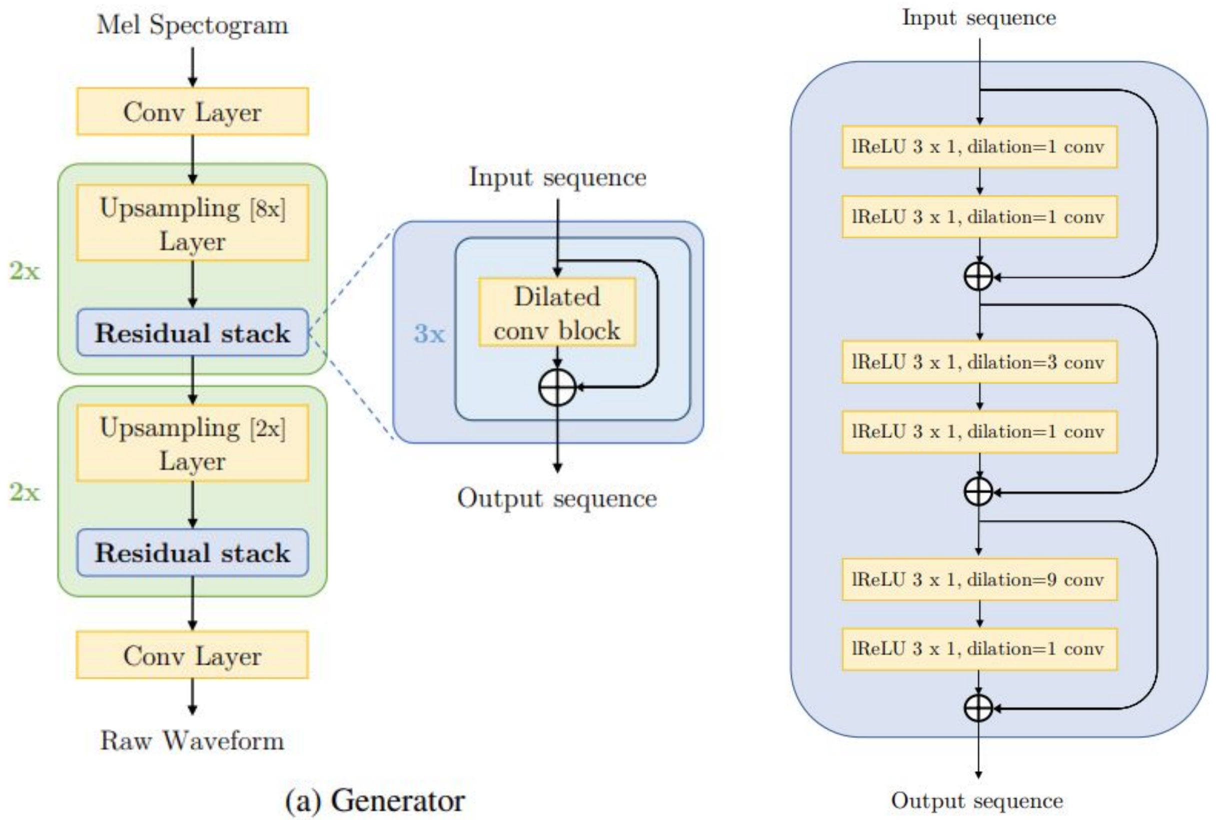
(a) Generator

7 × 1, stride=1 conv 512
lReLU 16 × 1, stride=8 conv transpose 256
Residual Stack 256
lReLU 16 × 1, stride=8 conv transpose 128
Residual Stack 128
lReLU 4 × 1, stride=2 conv transpose 64
Residual Stack 64
lReLU 4 × 1, stride=2 conv transpose 32
Residual Stack 32
lReLU 7 × 1, stride=1 conv 1 Tanh

(a) Generator Architecture

Input	B x 80 x n
Conv	B x 512 x n
Upsamp	B x 256 x n*8
Upsamp	B x 128 x n*64
Upsamp	B x 64 x n*128
Upsamp	B x 32 x n*256 (=s)
Conv	B x 1 x s

Generator: Inductive Bias



(a) Generator

Receptive field of 27 steps

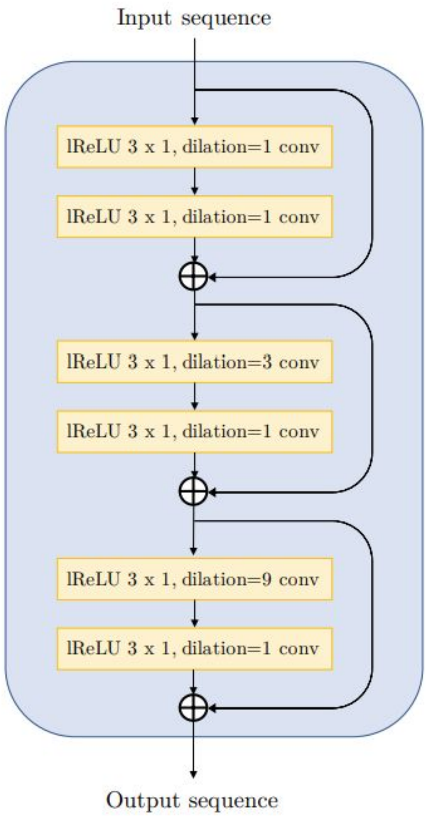
Dilation of 1, 3, 9

* Key for audio (periodicity can be captured in the samples that are far from each other)

Generator: Checkerboard Artifacts

7 × 1, stride=1 conv 512
lReLU 16 × 1, stride=8 conv transpose 256
Residual Stack 256
lReLU 16 × 1, stride=8 conv transpose 128
Residual Stack 128
lReLU 4 × 1, stride=2 conv transpose 64
Residual Stack 64
lReLU 4 × 1, stride=2 conv transpose 32
Residual Stack 32
lReLU 7 × 1, stride=1 conv 1 Tanh

(a) Generator Architecture



Upsampling layer

kernel-size = c * stride

Residual stack

dilation = kernel-size ** c

Generator: Weight Normalization

```
nn.ReflectionPad1d(3),
nn.utils.weight_norm(nn.Conv1d(mel_channel, 512, kernel_size=7, stride=1)),

nn.LeakyReLU(0.2),
nn.utils.weight_norm(nn.ConvTranspose1d(512, 256, kernel_size=16, stride=8, padding=4)),

ResStack(256),

nn.LeakyReLU(0.2),
nn.utils.weight_norm(nn.ConvTranspose1d(256, 128, kernel_size=16, stride=8, padding=4)),

ResStack(128),
```

• • •

Noticeable Differences

Generator

- No z
- Weight normalization (> instance normalization, spectral normalization)
- Inductive bias (dilation)
- Checkerboard artifacts (smart upsampling+dilation)

Discriminator

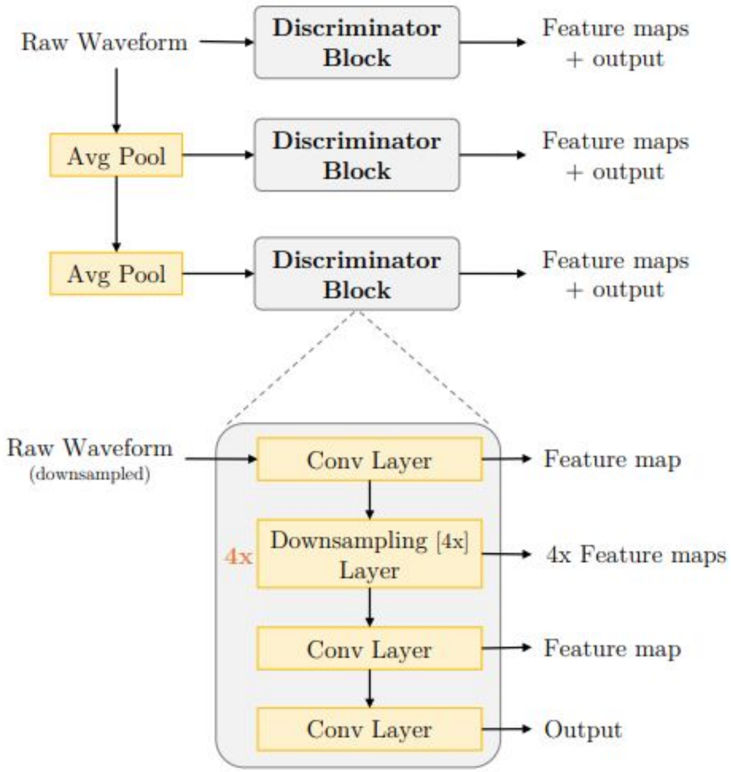
Fully Convolutional

Total # of params

5,641,362

Discriminator: Multi-scale & Identical

Input	B x 1 x (s or s/2 or s/4)
Conv1D (k=15, stride=1)	B x 16 x (s)
Conv1D (k=41, stride=4)	B x 64 x (s/4)
Conv1D (k=41, stride=4)	B x 256 x (s/16)
Conv1D (k=41, stride=4)	B x 1024 x (s/64)
Conv1D (k=41, stride=4)	B x 1024 x (s/256)
Conv1D (k=5, stride=1)	B x 1024 x (s/256)
Conv1D (k=3, stride=1)	B x 1 x (s/256 or s/512 or s/1024)

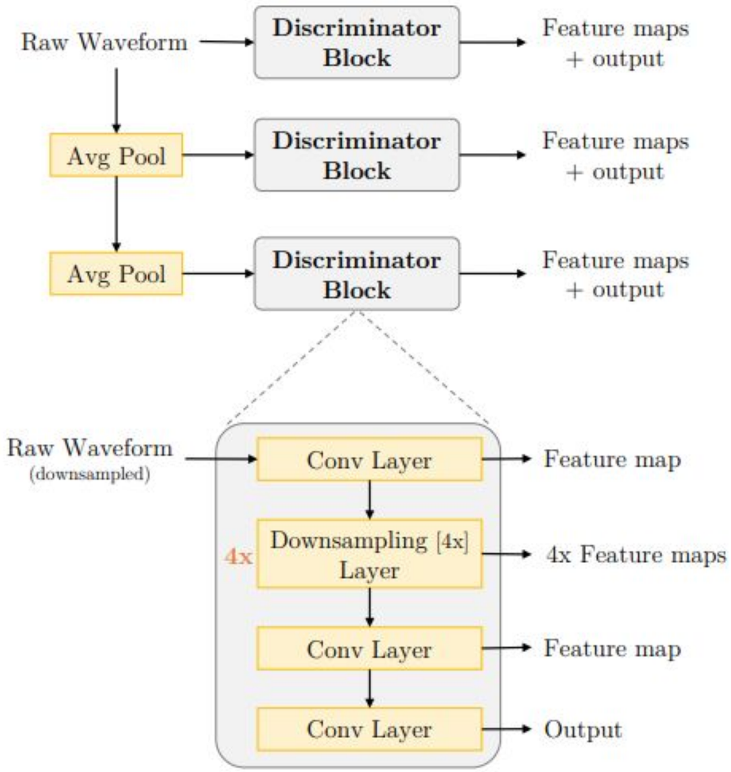


(b) Discriminator

Discriminator: Multi-scale & Identical

15 × 1, stride=1 conv 16 lReLU
41 × 1, stride=4 groups=4 conv 64 lReLU
41 × 1, stride=4 groups=16 conv 256 lReLU
41 × 1, stride=4 groups=64 conv 1024 lReLU
41 × 1, stride=4 groups=256 conv 1024 lReLU
5 × 1, stride=1 conv 1024 lReLU
3 × 1, stride=1 conv 1

(b) Discriminator Block Architecture



(b) Discriminator

Discriminator: Window-Based (loss)

Window-based

- Similar to image-patch
- Each window is equal to the receptive field of the discriminator

Discriminator: Feature-Matching (loss)

Q: Self-perceptual?

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

Noticeable Differences

Discriminator

- Three discriminators
- Window-based
- Feature matching

모든 과정의 Representation을 시간적으로도 여러번, 주파수대별로도 여러번 체크

주파수 대역별 너무 dominant하게 driving하지 않도록 강제

Loss

Hinge loss > LSGAN

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

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

Feature mapping (NOT in audio space)

$$\min_G \left(\mathbb{E}_{s,z} \left[\sum_{k=1,2,3} -D_k(G(s, z)) \right] + \lambda \sum_{k=1}^{3} \mathcal{L}_{\text{FM}}(G, D_k) \right)$$

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

Loss

```
loss_g = 0.0
for (feats_fake, score_fake), (feats_real, _) in zip(disc_fake, disc_real):
    loss_g += torch.mean(torch.sum(torch.pow(score_fake - 1.0, 2), dim=[1, 2]))
    for feat_f, feat_r in zip(feats_fake, feats_real):
        loss_g += hp.model.feat_match * torch.mean(torch.abs(feat_f - feat_r))
```

```
loss_d_sum = 0.0
for _ in range(hp.train.rep_discriminator):
    optim_d.zero_grad()
    disc_fake = model_d(fake_audio)
    disc_real = model_d(audioD)
    loss_d = 0.0
    for (_, score_fake), (_, score_real) in zip(disc_fake, disc_real):
        loss_d += torch.mean(torch.sum(torch.pow(score_real - 1.0, 2), dim=[1, 2]))
        loss_d += torch.mean(torch.sum(torch.pow(score_fake, 2), dim=[1, 2]))

    loss_d.backward()
    optim_d.step()
    loss_d_sum += loss_d
```

Result: inference speed

* Disc 5.64

Table 1: Comparison of the number of parameters and the inference speed. Speed of n kHz means that the model can generate $n \times 1000$ raw audio samples per second. All models are benchmarked using the same hardware [3].

Model	Number of parameters (in millions)	Speed on CPU (in kHz)	Speed on GPU (in kHz)
Wavenet (Shen et al., 2018)	24.7	0.0627	0.0787
Clarinet (Ping et al., 2018)	10.0	1.96	221
WaveGlow (Prenger et al., 2019)	87.9	1.58	223
MelGAN (ours)	**4.26**	**51.9**	**2500**

Result: ablation study

Weight norm > Spectral norm

*Weight norm effect is trivial

Feature map > audio space

Patch (window) > whole audio

Dilated conv (receptive field)

Multiscale discriminator

Model	MOS	95% CI
w/ Spectral Normalization	1.33	±0.07
w/ L1 loss (audio space)	2.59	±0.11
w/o Window-based Discriminator	2.29	±0.10
w/o Dilated Convolutions	2.60	±0.10
w/o Multi-scale Discriminator	2.93	±0.11
w/o Weight Normalization	3.03	±0.10
Baseline (MelGAN)	**3.09**	± **0.11**

Result: infer quality

Table 3: Mean Opinion Scores

Model	MOS	95% CI
Griffin Lim	1.57	±0.04
WaveGlow	4.11	±0.05
WaveNet	4.05	±0.05
MelGAN	3.61	±0.06
Original	**4.52**	**± 0.04**

Table 4: Mean Opinion Scores on the VCTK dataset (Veaux et al., 2017).

Model	MOS	95% CI
Griffin Lim	1.72	±0.07
MelGAN	3.49	±0.09
Original	**4.19**	**± 0.08**

Model	MOS	95% CI
Tacotron2 + WaveGlow	3.52	±0.04
Text2mel + WaveGlow	4.10	±0.03
Text2mel + MelGAN	3.72	±0.04
Text2mel + Griffin-Lim	1.43	±0.04
Original	4.46	± 0.04

Q: Text2Mel + WG > Taco + WG?

MOS

200 individuals

Asked to blindly evaluate a subset of 16 samples taken randomly

$$\hat{\mu}_i = \frac{1}{N_i}\sum_{k=1}^{N_i} m_{i,k}$$

$$CI_i = \left[\hat{\mu}_i - 1.96\frac{\hat{\sigma}_i}{\sqrt{N_i}}, \hat{\mu}_i + 1.96\frac{\hat{\sigma}_i}{\sqrt{N_i}}\right]$$

How to Improve?

ConvTranspose known to cause artifacts

- Replace with Upsample1D and Downsample1D?

WaveFlow (19.12.)

Focus

Flow concepts

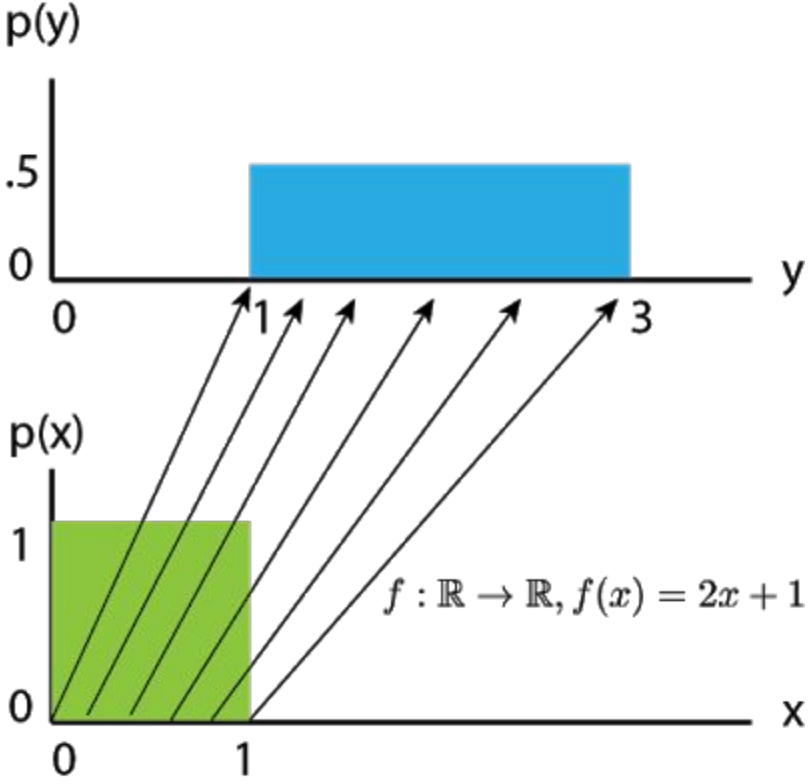
Conceptual overview of WaveFlow

Flow

Complex probability distribution → Manageable probability distribution

Via **invertible** transformations of distributions

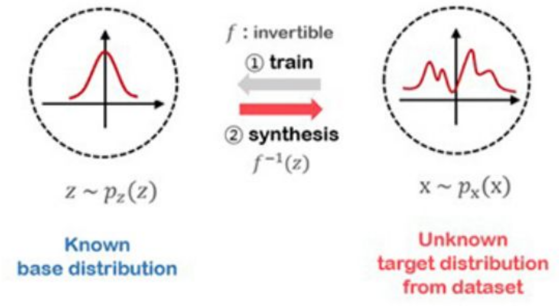
Flow



Why Flow?

Direct modelling of latent space

Intuitive training/synthesis



likelihood of the target probability as a loss

$$p(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right|$$

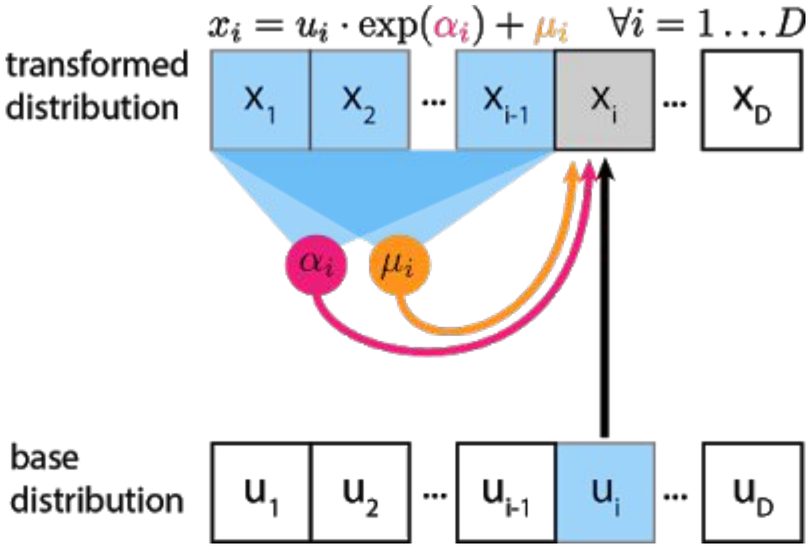
Flow Constraints

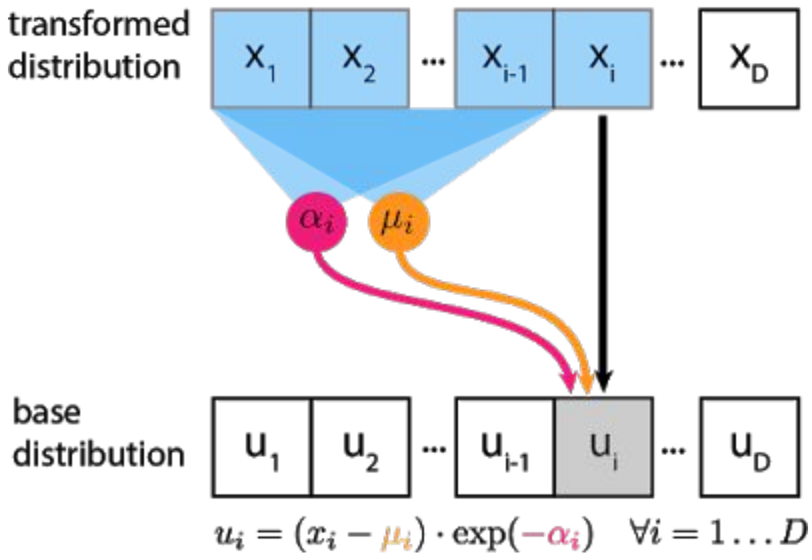
1. Invertible

2. Scalable determinant of Jacobian (normally $O(n^3)$)

*Left only with simple operations → more layers needed

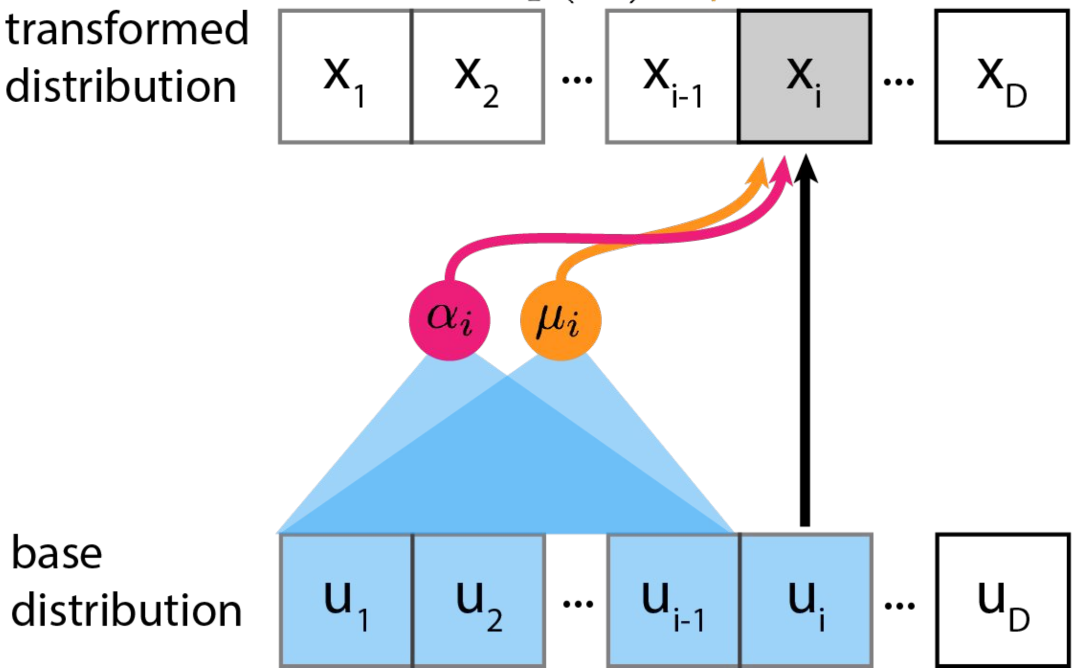
Autoregressive Flow



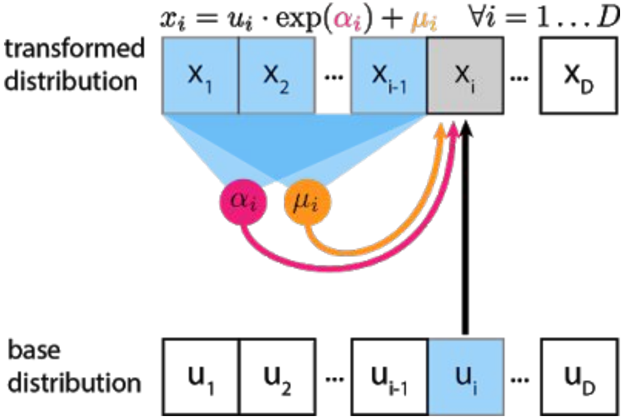


Inverse Autoregressive Flow

$$x_i = u_i \cdot \exp(\alpha_i) + \mu_i \quad \forall i = 1 \ldots D$$



Gaussian WaveNet (Autoregressive Flow)

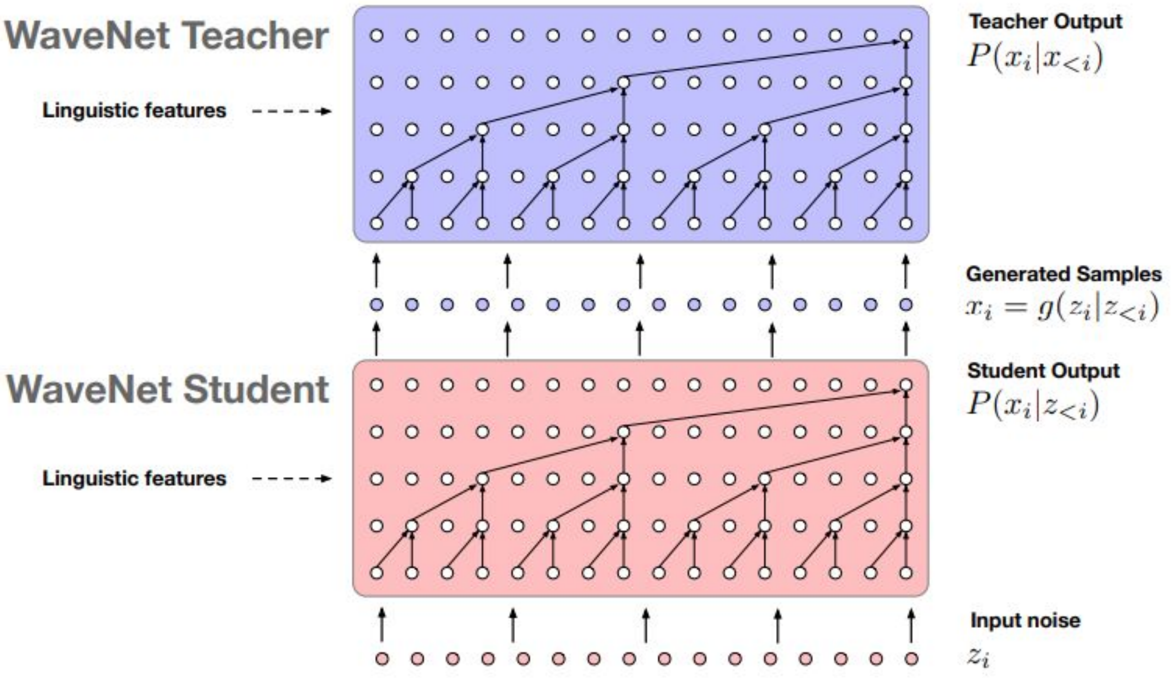


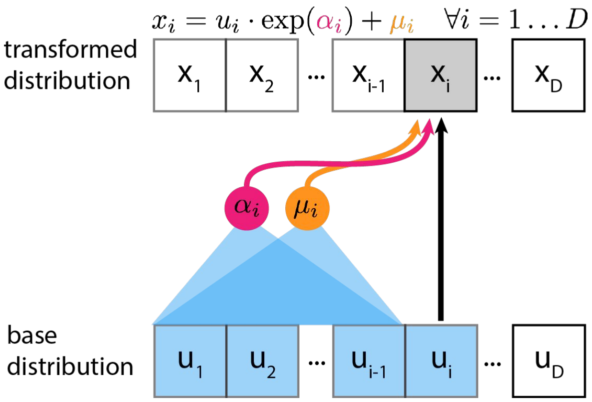
$$p(\boldsymbol{x} \mid \boldsymbol{c}\,;\, \boldsymbol{\theta}) = \prod_{t=1}^{T} p(x_t \mid x_{<t}, \boldsymbol{c}\,;\, \boldsymbol{\theta}),$$

$$p(x_t \mid x_{<t}; \boldsymbol{\theta}) = \mathcal{N}\big(\mu(x_{<t}; \boldsymbol{\theta}), \sigma(x_{<t}; \boldsymbol{\theta})\big),$$

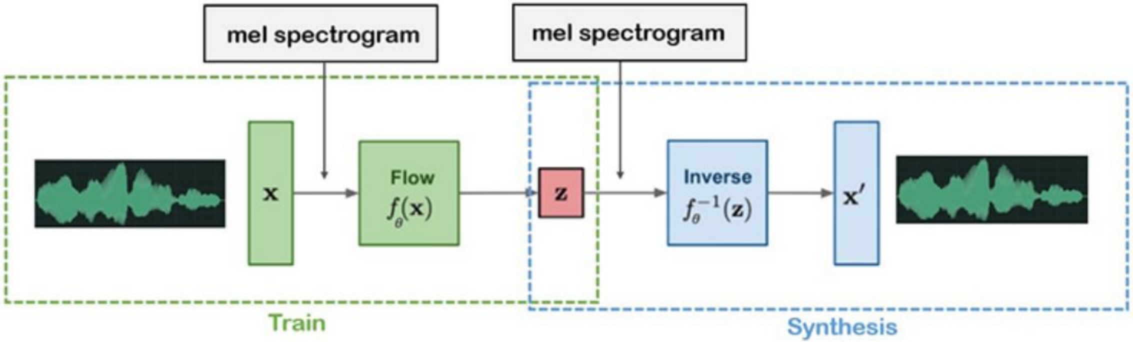
$$z_t = x_t \cdot \sigma_t(x_{<t}; \boldsymbol{\vartheta}) + \mu_t(x_{<t}; \boldsymbol{\vartheta}),$$

Parallel Wavenet (Inverse Autoregressive Flow)





WaveGlow (Flow)



WaveGlow: Block

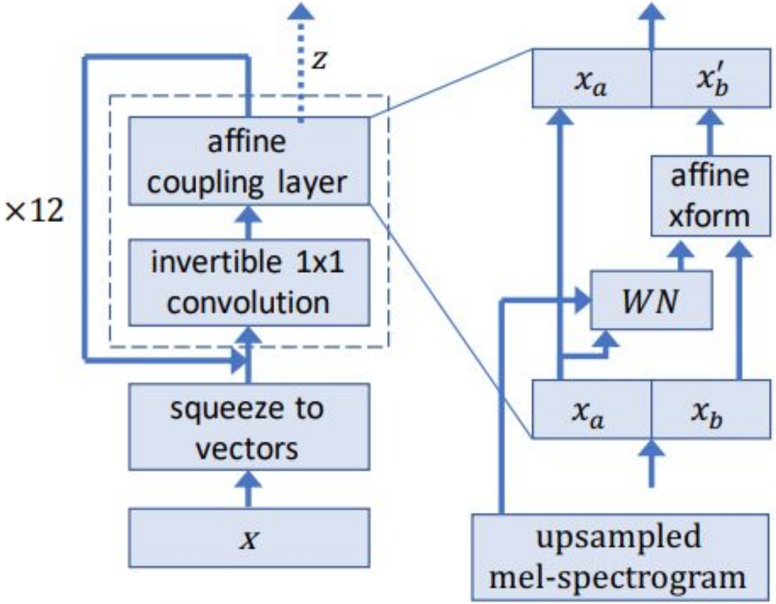


Fig. 1: WaveGlow network

WaveGlow: Two Operations

Affine Coupling (*bipartite)

$$\boldsymbol{x}_a, \boldsymbol{x}_b = split(\boldsymbol{x})$$
$$(\log \boldsymbol{s}, \boldsymbol{t}) = WN(\boldsymbol{x}_a, mel\text{-}spectrogram)$$

$$\boldsymbol{f}^{-1}_{coupling}(\boldsymbol{x}) = concat(\boldsymbol{x}_a, \boldsymbol{x}_b\prime)$$
$$\log|\det(\boldsymbol{J}(\boldsymbol{f}^{-1}_{coupling}(\boldsymbol{x})))| = \log|\boldsymbol{s}|$$

1x1 Conv

$$\boldsymbol{f}^{-1}_{conv} = \boldsymbol{W}\boldsymbol{x}$$
$$\log|\det(\boldsymbol{J}(\boldsymbol{f}^{-1}_{conv}(\boldsymbol{x})))| = \log|\det \boldsymbol{W}|$$

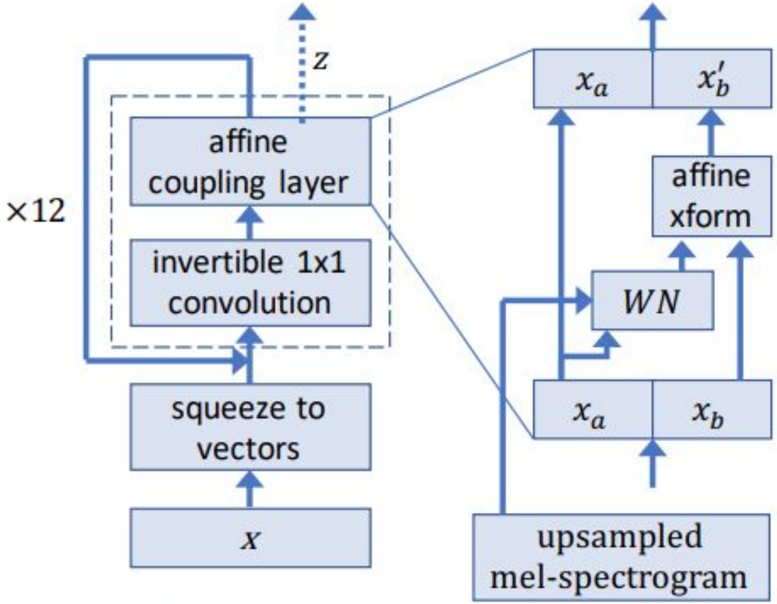


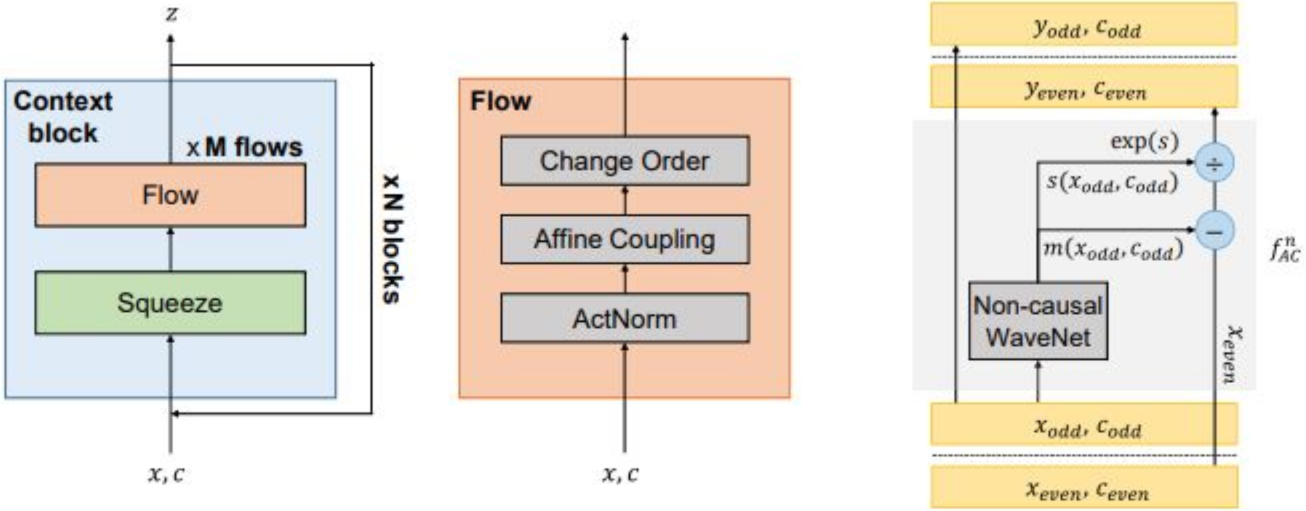
Fig. 1: WaveGlow network

WaveGlow: Loss

$$\log p_\theta(\boldsymbol{x}) = \log p_\theta(\boldsymbol{z}) + \sum_{i=1}^{k} \log |\det(\boldsymbol{J}(\boldsymbol{f}_i^{-1}(\boldsymbol{x})))|$$

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

FloWaveNet



z
Context block
x M flows
Flow
Squeeze
x N blocks
x, c
Flow
Change Order
Affine Coupling
ActNorm
x, c
y_{odd}, c_{odd}
y_{even}, c_{even}
$\exp(s)$
$s(x_{odd}, c_{odd})$
$m(x_{odd}, c_{odd})$
f_{AC}^{n}
Non-causal WaveNet
x_{even}
x_{odd}, c_{odd}
x_{even}, c_{even}

*Bonus

NCSoft에서는…

WaveGlow모델에 샘플 단위의 손실 함수(합성된 음성 샘플과 실제 음성 샘플간 차이)를 추가

Super Resolution 모듈 추가

WaveFlow

ICLR 2020 **REJECT**

The main concern of this paper is the novelty and depth of the analysis.

WaveFlow Contribution

A unified view of flow models

Dilated 2D Convolution over WaveNet-like structure

Permutation over 1x1 conv

Beyond Bipartition

AR vs Bipartite

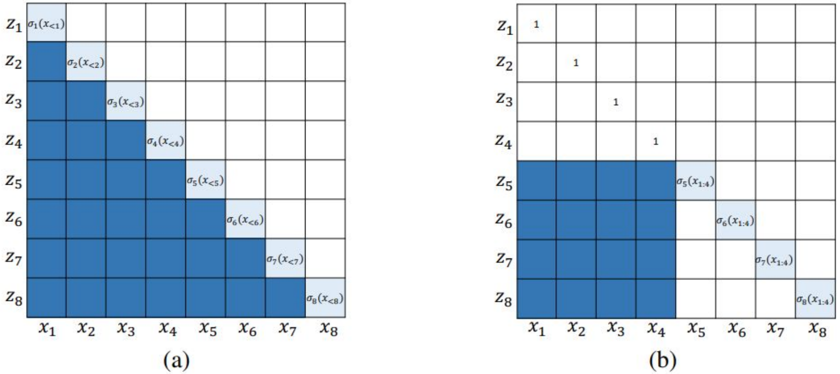
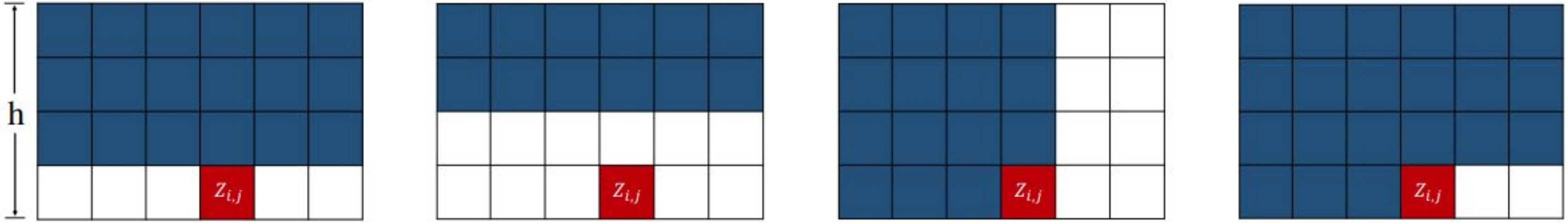


Figure 1: The Jacobian $\frac{\partial f^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ of (a) an autoregressive transformation, and (b) a bipartite transformation. The blank cells are zeros and represent the independent relations between z_i and x_j. The light-blue cells are scaling variables and represent the linear dependencies between z_i and x_i. The dark-blue cells represent complex non-linear dependencies defined by neural networks.

WaveFlow vs WaveGlow vs WaveNet

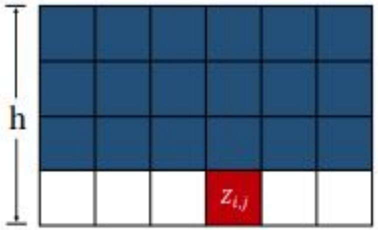


(a) (b) (c) (d)

Figure 2: The receptive fields over the squeezed inputs X for computing $Z_{i,j}$ in (a) WaveFlow, (b) WaveGlow, (c) autoregressive flow with column-major order (e.g., WaveNet), and (d) autoregressive flow with row-major order.

Squeeze

$X \in R^{h \times w}$



Operation

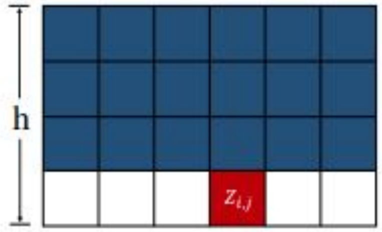
Shifting Variable $\mu_{i,j}(X_{<i,\bullet};\Theta)$

Scaling Variable $\sigma_{i,j}(X_{<i,\bullet};\Theta)$

obtained via dilated 2D convolution (no need to be WN)

Permutation instead of 1x1 conv (more efficient)

Inverse Transformation



$$Z_{i,j} = \sigma_{i,j}(X_{<i,\bullet};\Theta) \cdot X_{i,j} + \mu_{i,j}(X_{<i,\bullet};\Theta),$$

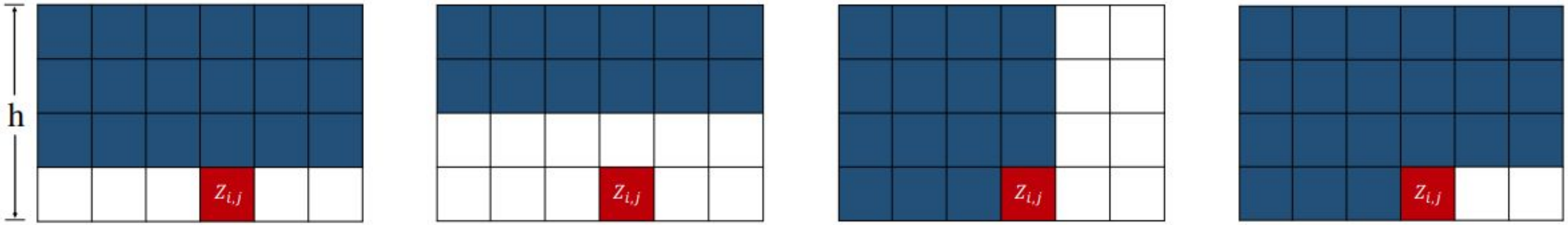
*Jacobian is triangular: $Z_{i,j}$ only depends on the current $X_{i,j}$ and previous $X_{<i,\bullet}$

$$\det\left(\frac{\partial f^{-1}(X)}{\partial X}\right) = \prod_{i=1}^{h}\prod_{j=1}^{w}\sigma_{i,j}(X_{<i,\bullet};\Theta).$$

Loss

$$\log p(X) = -\sum_{i=1}^{h}\sum_{j=1}^{w}\left(Z_{i,j}^2 + \frac{1}{2}\log(2\pi)\right) + \sum_{i=1}^{h}\sum_{j=1}^{w}\log \sigma_{i,j}(X_{<i,\bullet};\Theta),$$

Generalization



(a) (b) (c) (d)

Figure 2: The receptive fields over the squeezed inputs X for computing $Z_{i,j}$ in (a) WaveFlow, (b) WaveGlow, (c) autoregressive flow with column-major order (e.g., WaveNet), and (d) autoregressive flow with row-major order.

Result

Table 5: The synthesis speed over real-time and the 5-scale Mean Opinion Score (MOS) ratings with 95% confidence intervals. Models with bolded numbers are mentioned in the text.

Model	**flows×layers**	**res. channels**	**# param**	**syn. speed**	**MOS**
Gaussian WaveNet	1×30 = 30	128	4.57 M	0.002×	4.43 ± 0.14
ClariNet	6×10 = 60	64	2.17 M	21.64×	4.22 ± 0.15
WaveGlow	12×8 = 96	64	17.59 M	93.53×	2.17 ± 0.13
WaveGlow	12×8 = 96	128	34.83 M	69.88×	2.97 ± 0.15
WaveGlow	12×8 = 96	256	87.88 M	34.69×	4.34 ± 0.11
WaveGlow	12×8 = 96	512	268.29 M	8.08×	4.32 ± 0.12
WaveFlow ($h = 8$)	8×8 = 64	64	5.91 M	47.61×	4.26 ± 0.12
WaveFlow ($h = 16$)	8×8 = 64	64	**5.91** M	**42.60**×	$\mathbf{4.32 \pm 0.08}$
WaveFlow ($h = 16$)	8×8 = 64	96	12.78 M	26.23×	4.34 ± 0.13
WaveFlow ($h = 16$)	8×8 = 64	128	22.25 M	21.32×	4.38 ± 0.09
WaveFlow ($h = 16$)	8×8 = 64	256	86.18 M	8.42×	$\mathbf{4.43 \pm 0.10}$
Ground-truth	—	—	—	—	4.56 ± 0.09

SqueezeWave: Extremely Lightweight Vocoders for On-device Speech Synthesis (20.01)