Inter-University Research Institute Corporation /
Research Organization of Information and Systems
National Institute of Informatics

SLP-L8.6 ICASSP 2019
Brighton, United Kingdom

Neural source-filter waveform model

Xin WANG, Shinji TAKAKI, Junichi YAMAGISHI
National Institute of Informatics, Japan

2019-02-27

Note: Japanese natural speech waveforms are deleted due to license

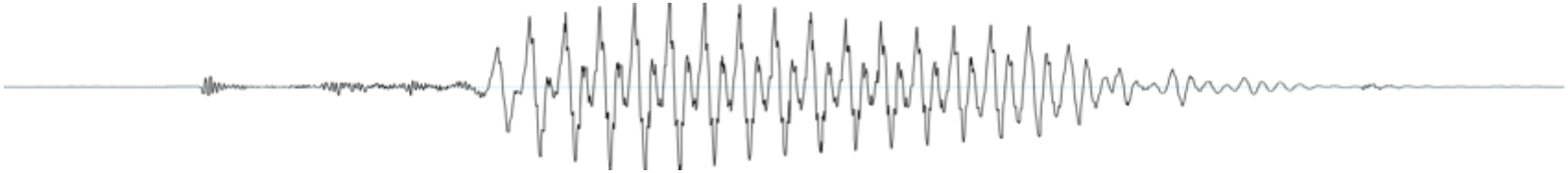
contact: wangxin@nii.ac.jp
we welcome critical comments, suggestions, and discussion

CONTENTS

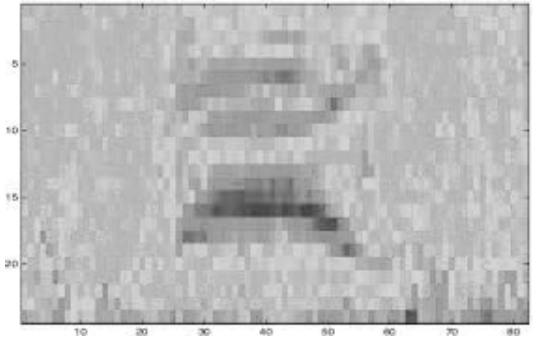
- Introduction
- Proposed model
- Experiments
- Summary

INTRODUCTION

Task definition

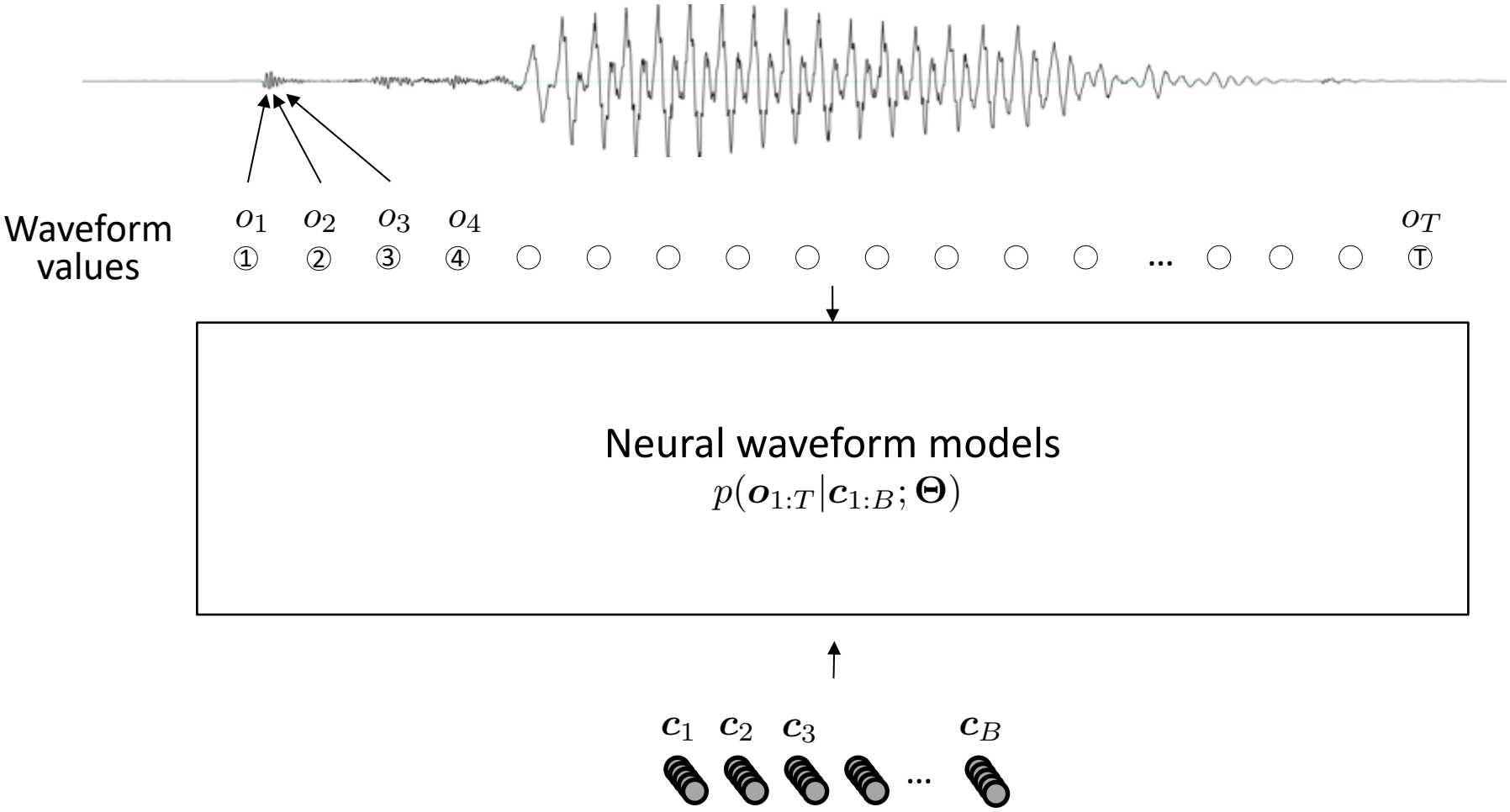


Mel-spectrogram, F0, etc.



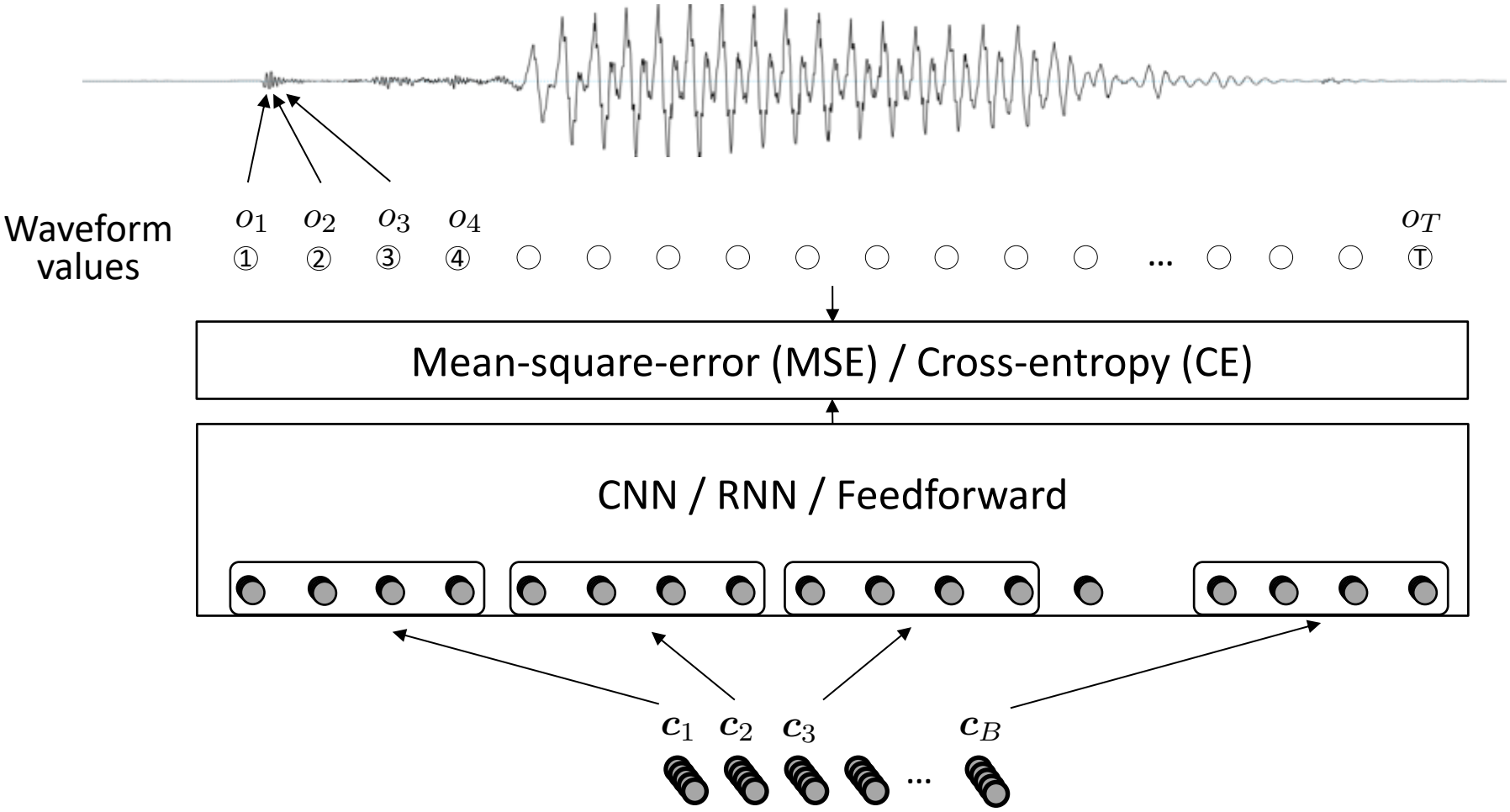
INTRODUCTION

Task definition



INTRODUCTION

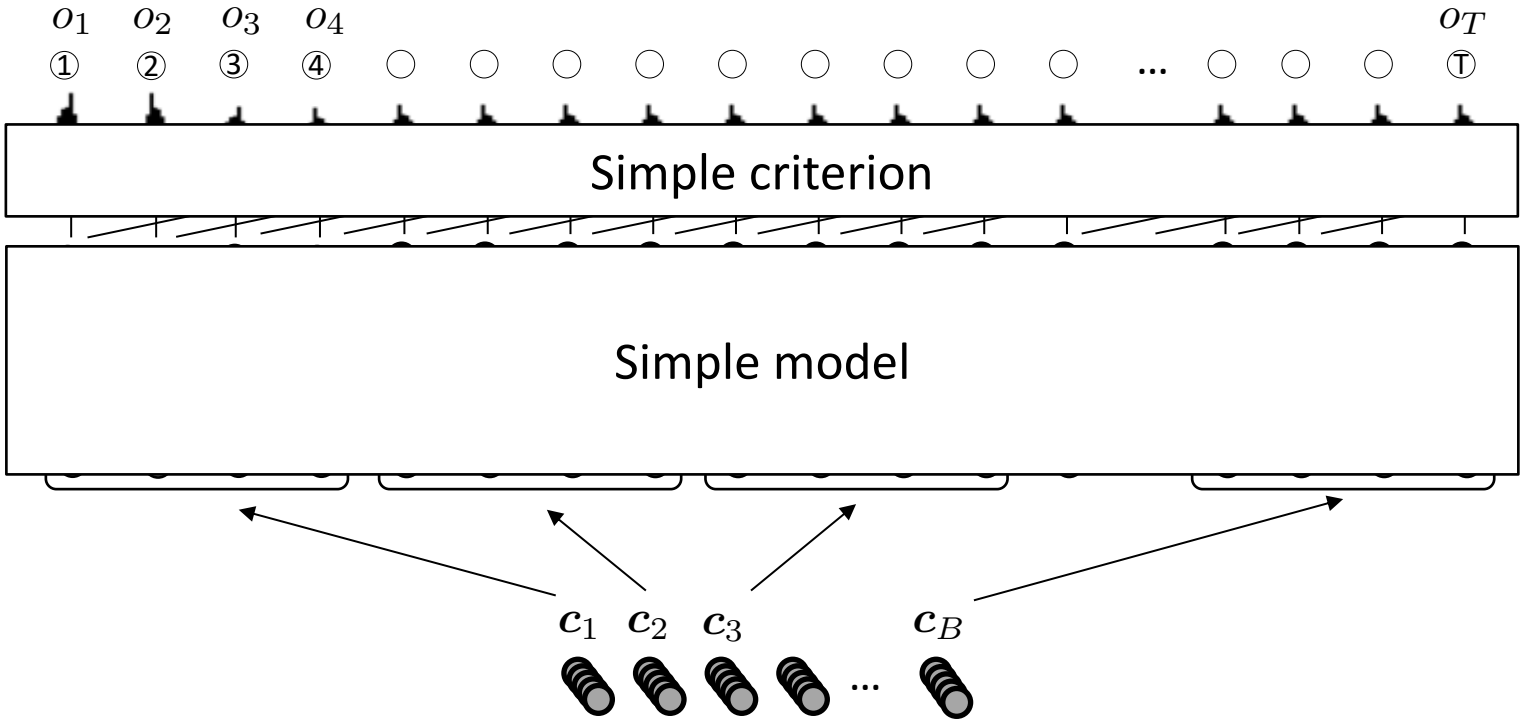
Naïve model



INTRODUCTION

Naïve model

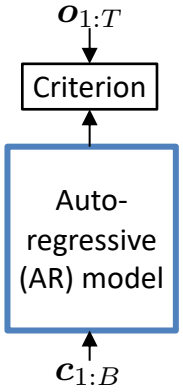
$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B};\boldsymbol{\Theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{c}_{1:B};\boldsymbol{\Theta})$$



INTRODUCTION

Recent neural waveform models

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{<t}, \boldsymbol{c}_{1:B})$$



WaveNet

SampleRNN

WaveRNN

FFTNet

Universal neural vocoder

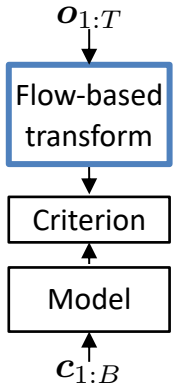
Flow-based models

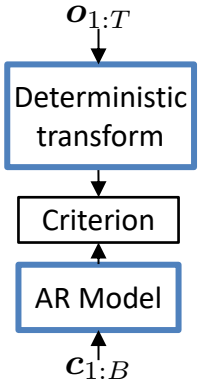
Parallel WaveNet

FloWaveNet

ClariNet

WaveGlow





AR + expert knowledge

LPCNet

ExcitNet

GlotNet

LP-WaveNet

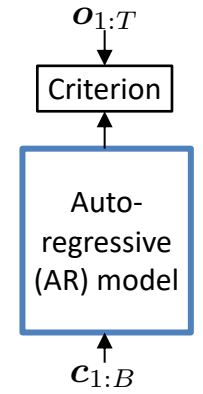
☛ Reference in appendix

☛ Tutorial slides: https://www.slideshare.net/jyamagis/

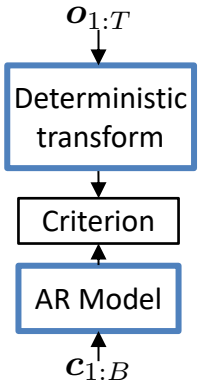
INTRODUCTION

Recent neural waveform models

$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:B}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{<t}, \boldsymbol{c}_{1:B})$$



- Powerful model
- ! Sequential generation

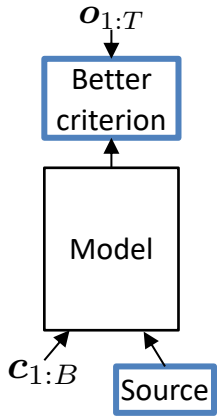


Flow-based models

- Non-sequential generation
- ! Complicated
- ! Slow in training

Neural source-filter model (NSF)

- Spectral-domain criterion
- Source-filter architecture



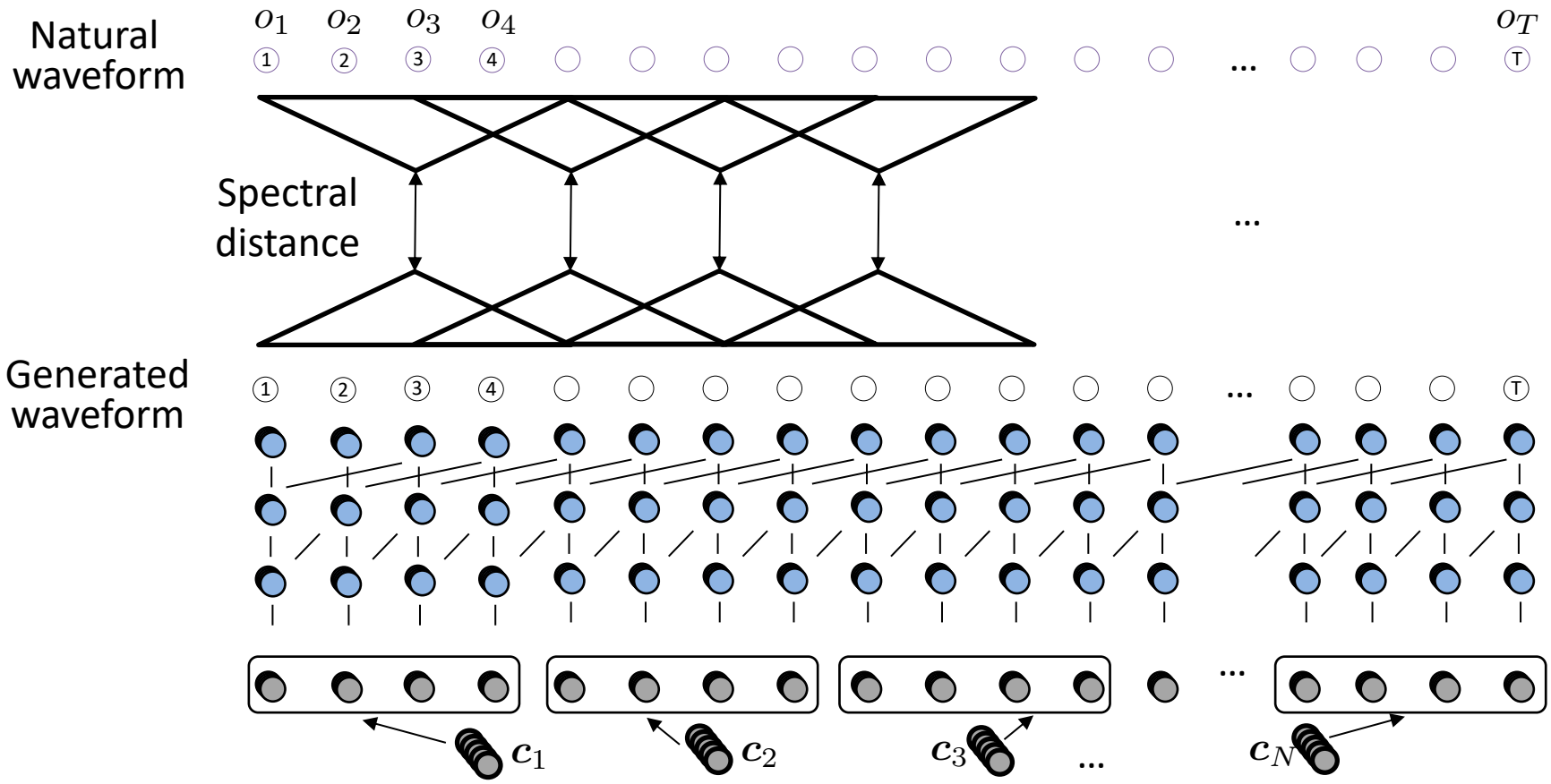
☛ Reference in appendix
☛ Tutorial slides: https://www.slideshare.net/jyamagis/

CONTENTS

- Introduction
- NSF model
- Experiments
- Summary

NEURAL SOURCE-FILTER MODEL

Idea 1: spectral domain criterion

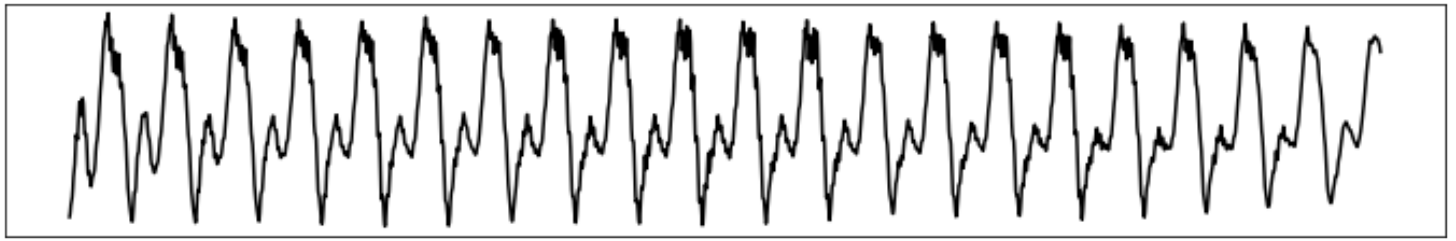


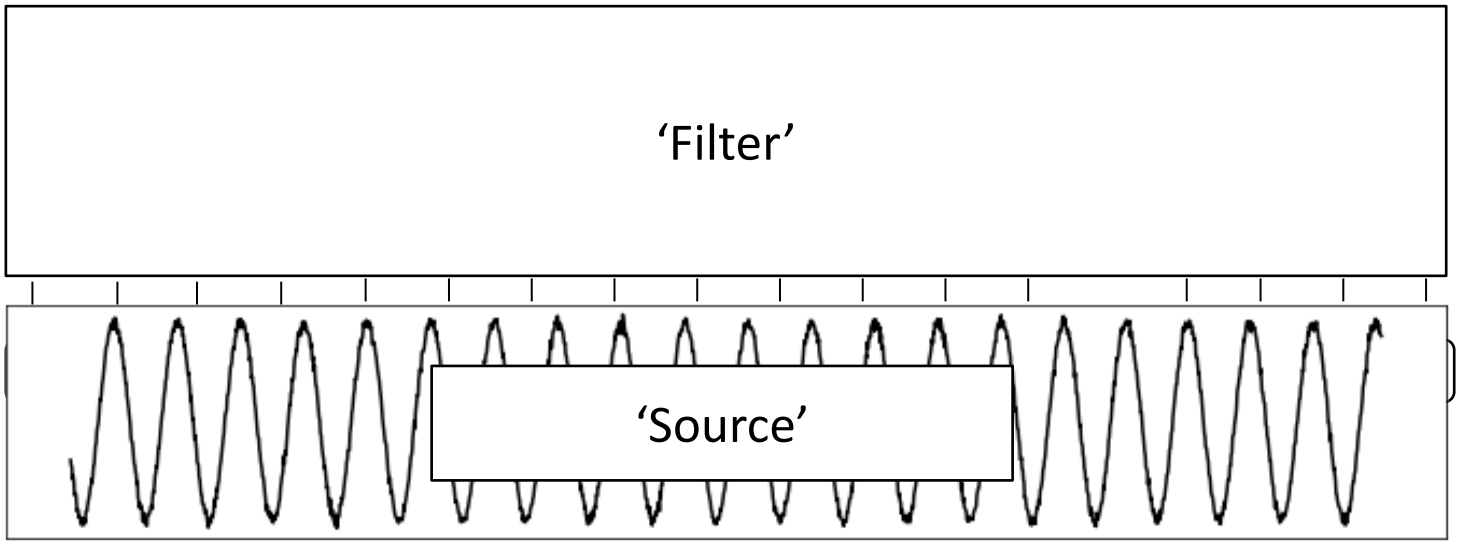
- Avoid naïve model assumption $p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:N}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{c}_{1:N})$

NEURAL SOURCE-FILTER MODEL

Idea 2: sine-based source excitation

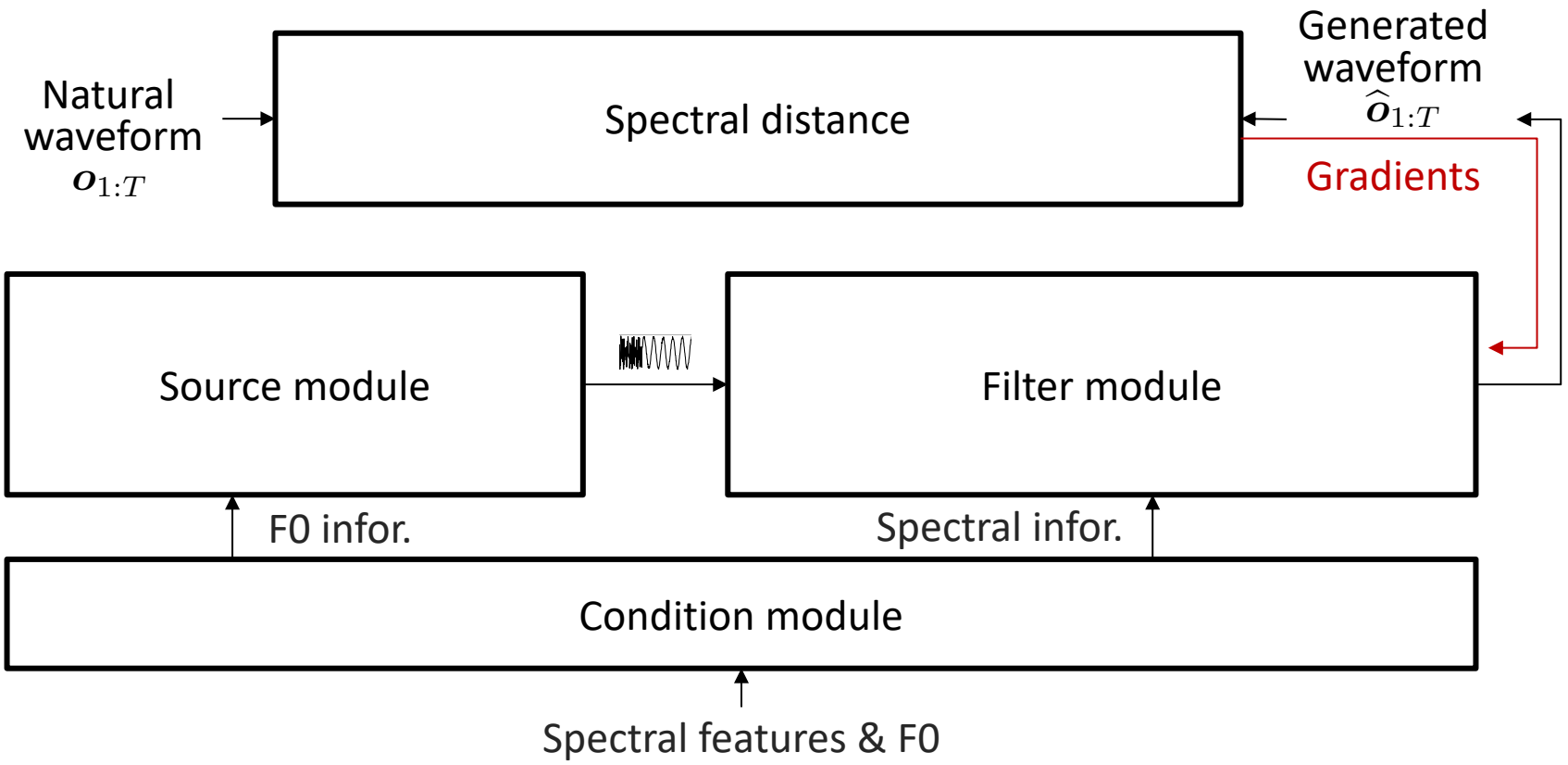
Natural waveform





NEURAL SOURCE-FILTER MODEL

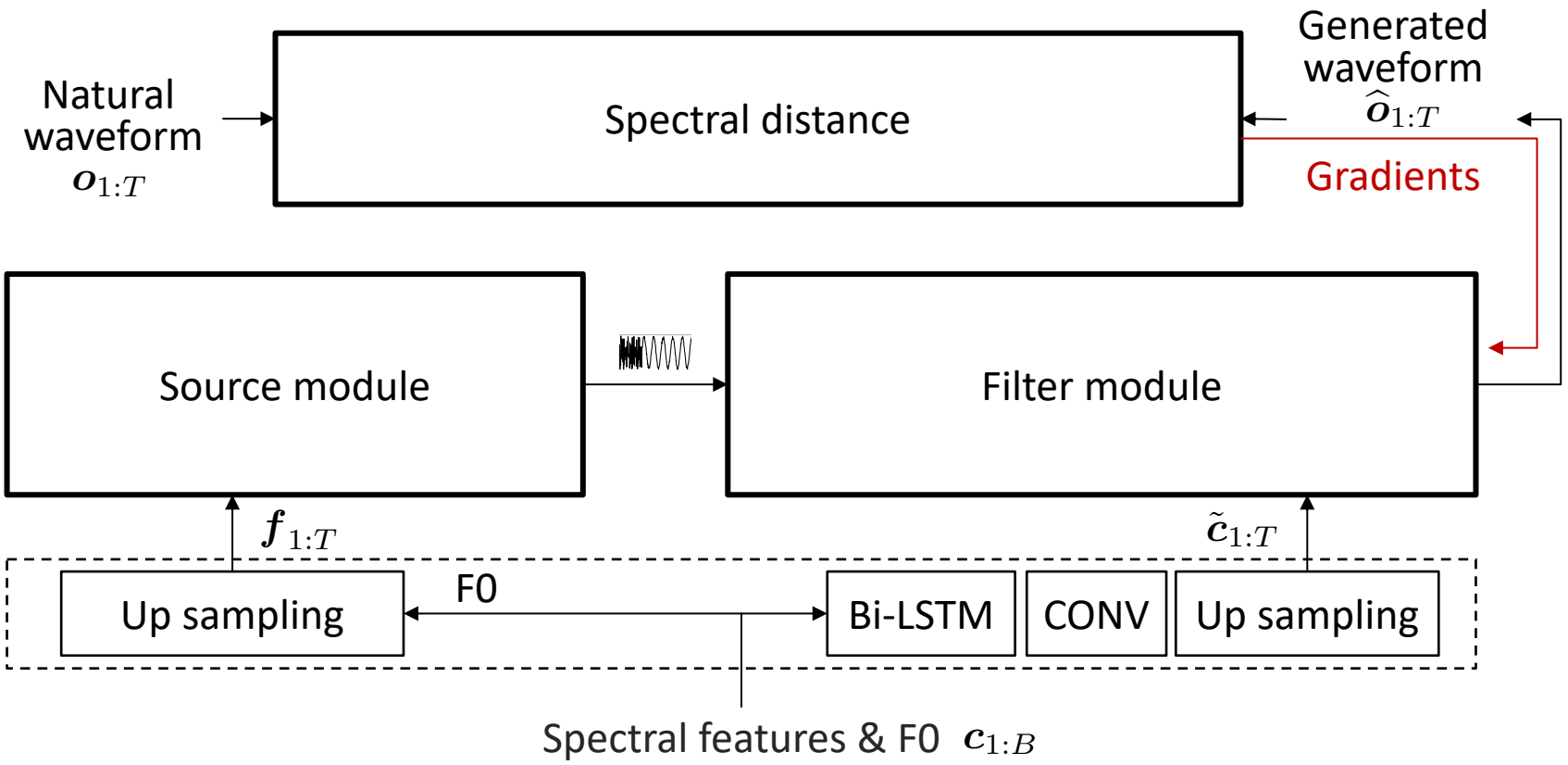
Model structure



- Without AR, flow, or knowledge-distilling

NEURAL SOURCE-FILTER MODEL

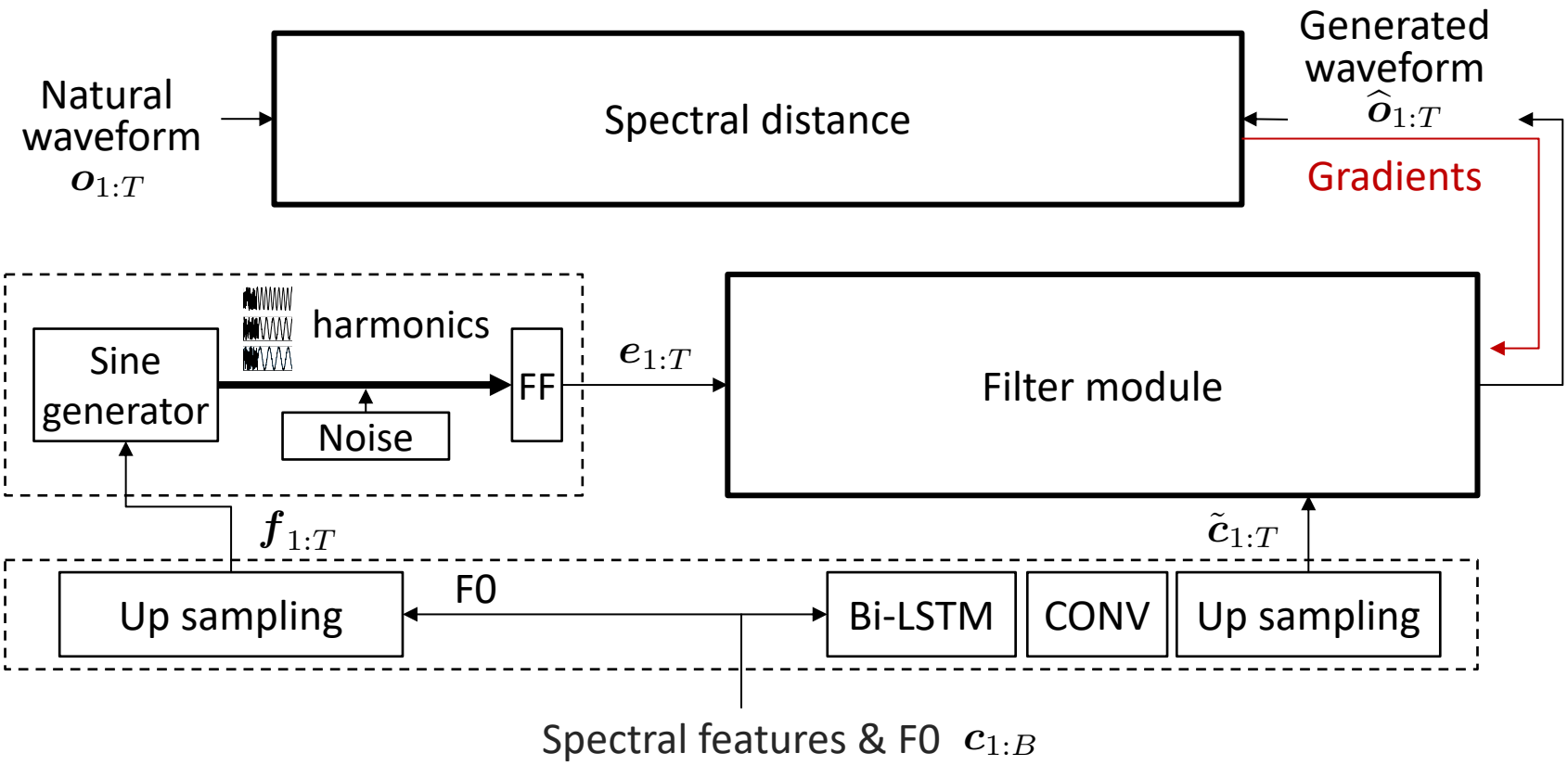
Condition module



- Up sampling by replication

NEURAL SOURCE-FILTER MODEL

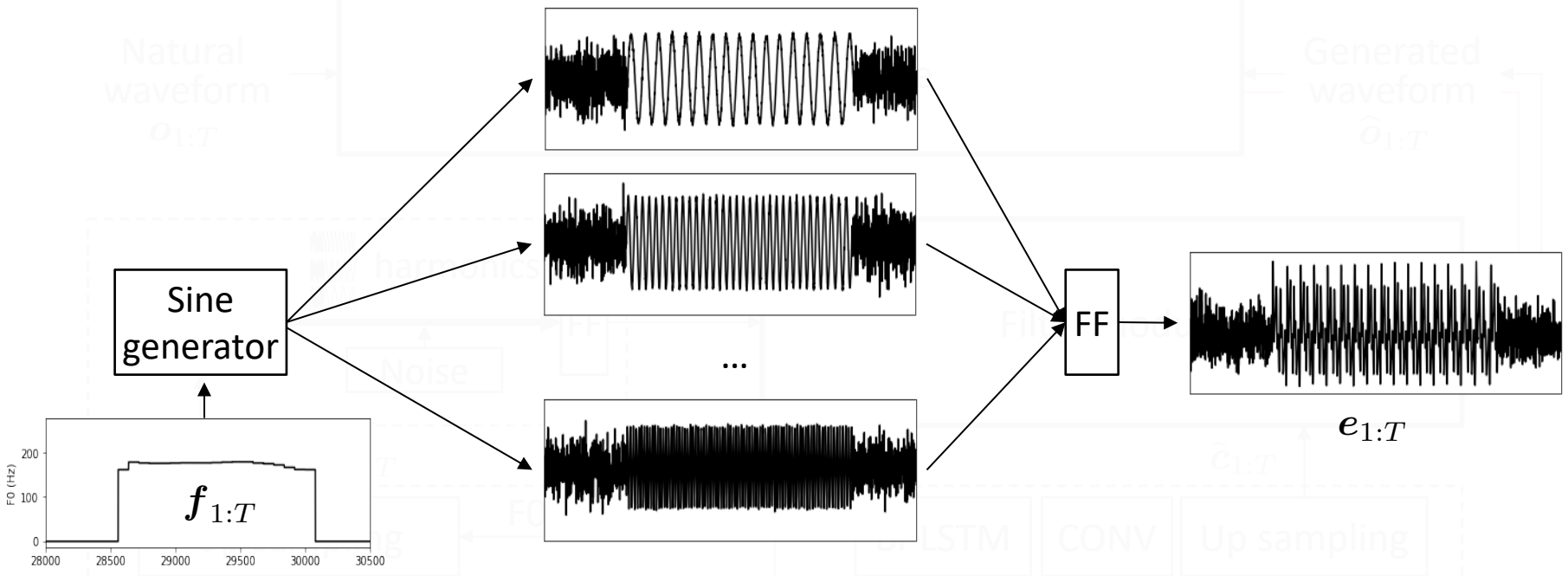
Source module



- FF: feedforward layer with Tanh
- $f_t \in \{0\} \cup \mathbb{R}^+$, $e_t \in \mathbb{R}, \forall t \in \{1, \cdots, T\}$

NEURAL SOURCE-FILTER MODEL

Source module



Fundamental component

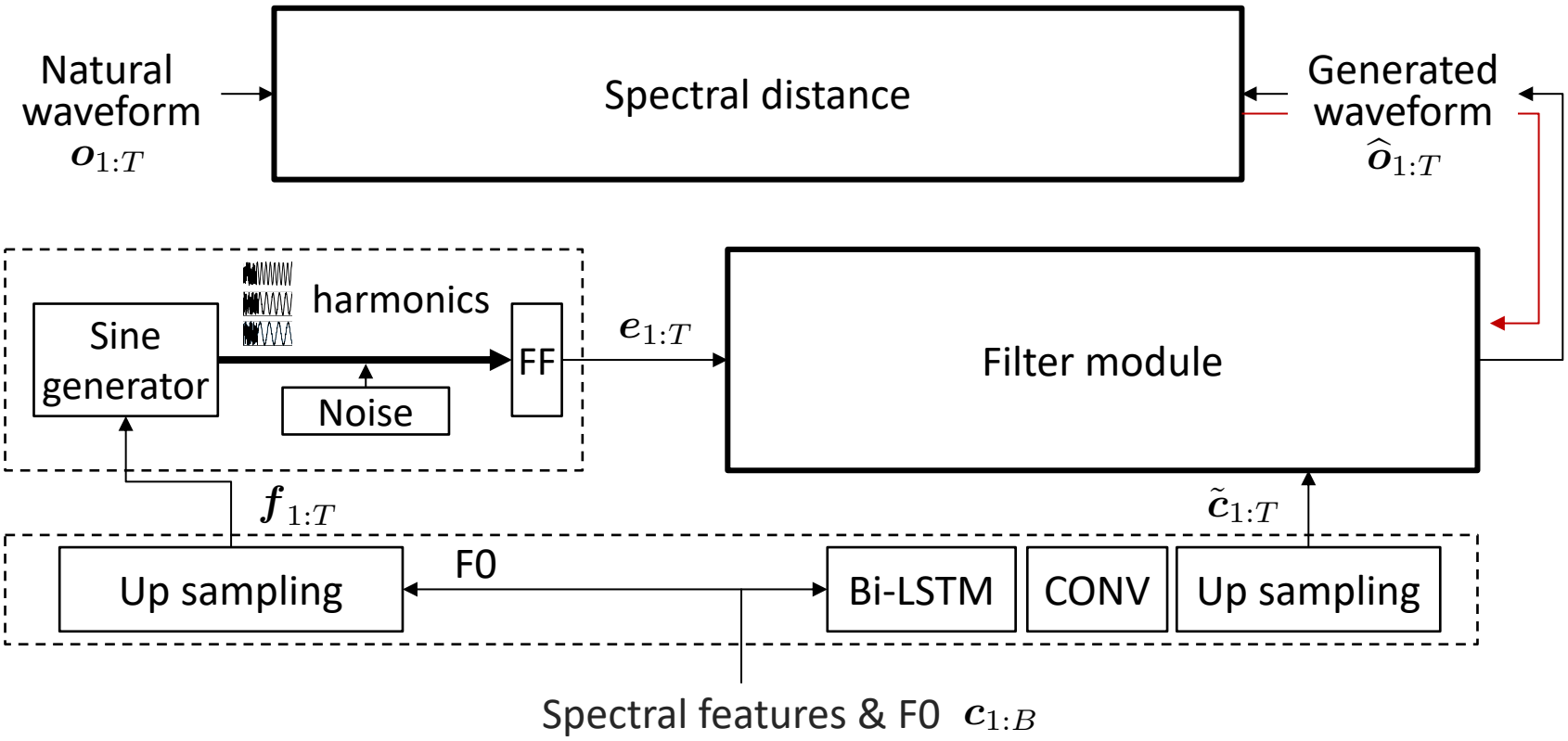
Voiced: $\sin(\sum_{k=1}^{t} 2\pi \frac{f_k}{N_s} + \phi) + n_t$

- n_t → Noise
- ϕ → Random initial phase
- N_s → Sampling rate

Unvoiced: noise

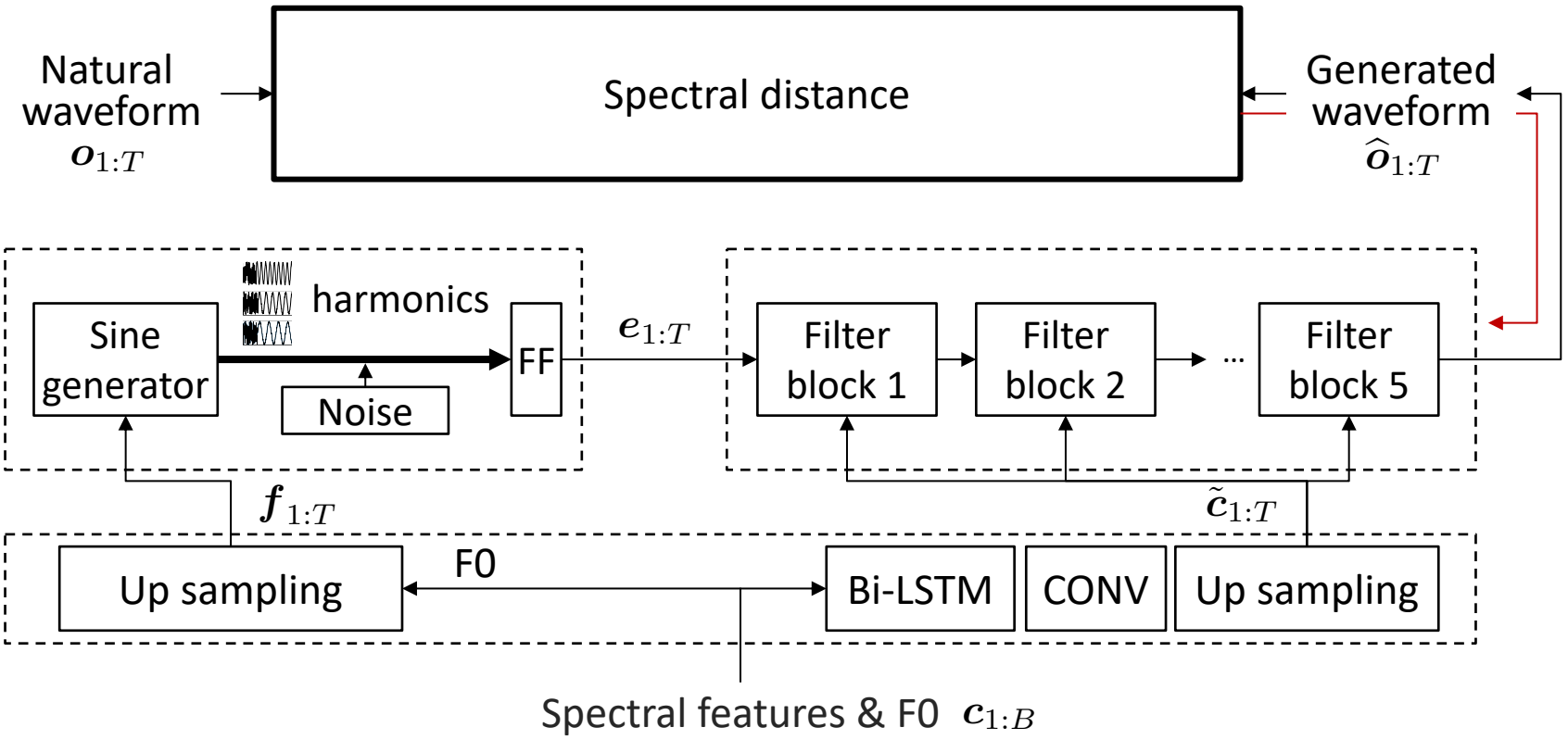
NEURAL SOURCE-FILTER MODEL

Filter module



NEURAL SOURCE-FILTER MODEL

Filter module

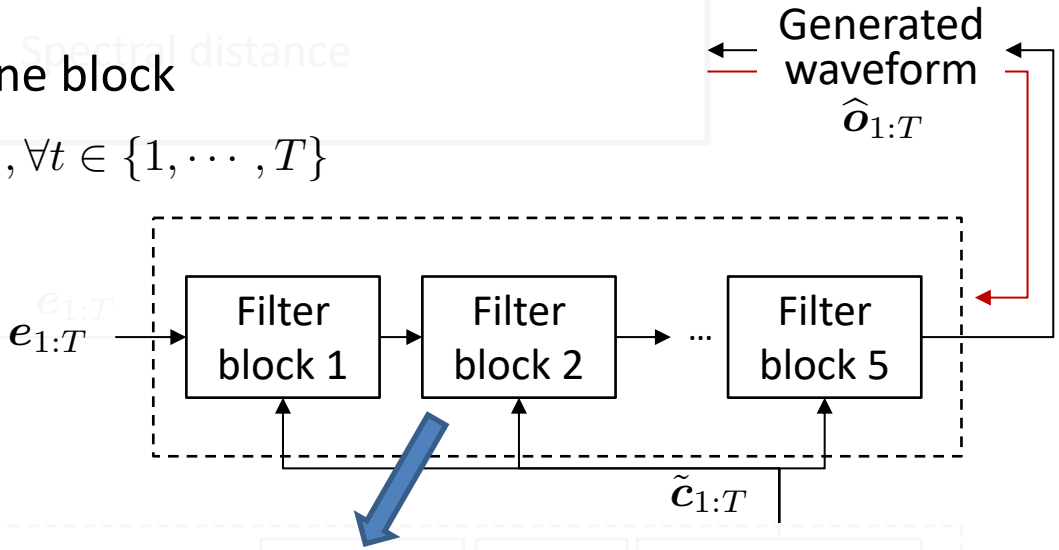


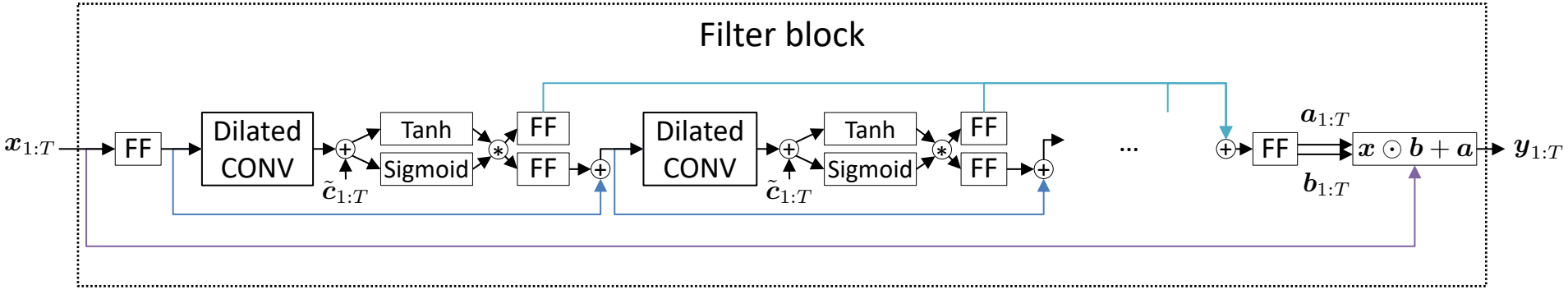
- Neural filter blocks based on dilated convolution (CONV)

NEURAL SOURCE-FILTER MODEL

Filter module

- Similar to ClariNet
- Ten dilated-CONV in one block
- $x_t, y_t, \hat{o}_t, a_t \in \mathbb{R}, b_t \in \mathbb{R}^+, \forall t \in \{1, \cdots, T\}$

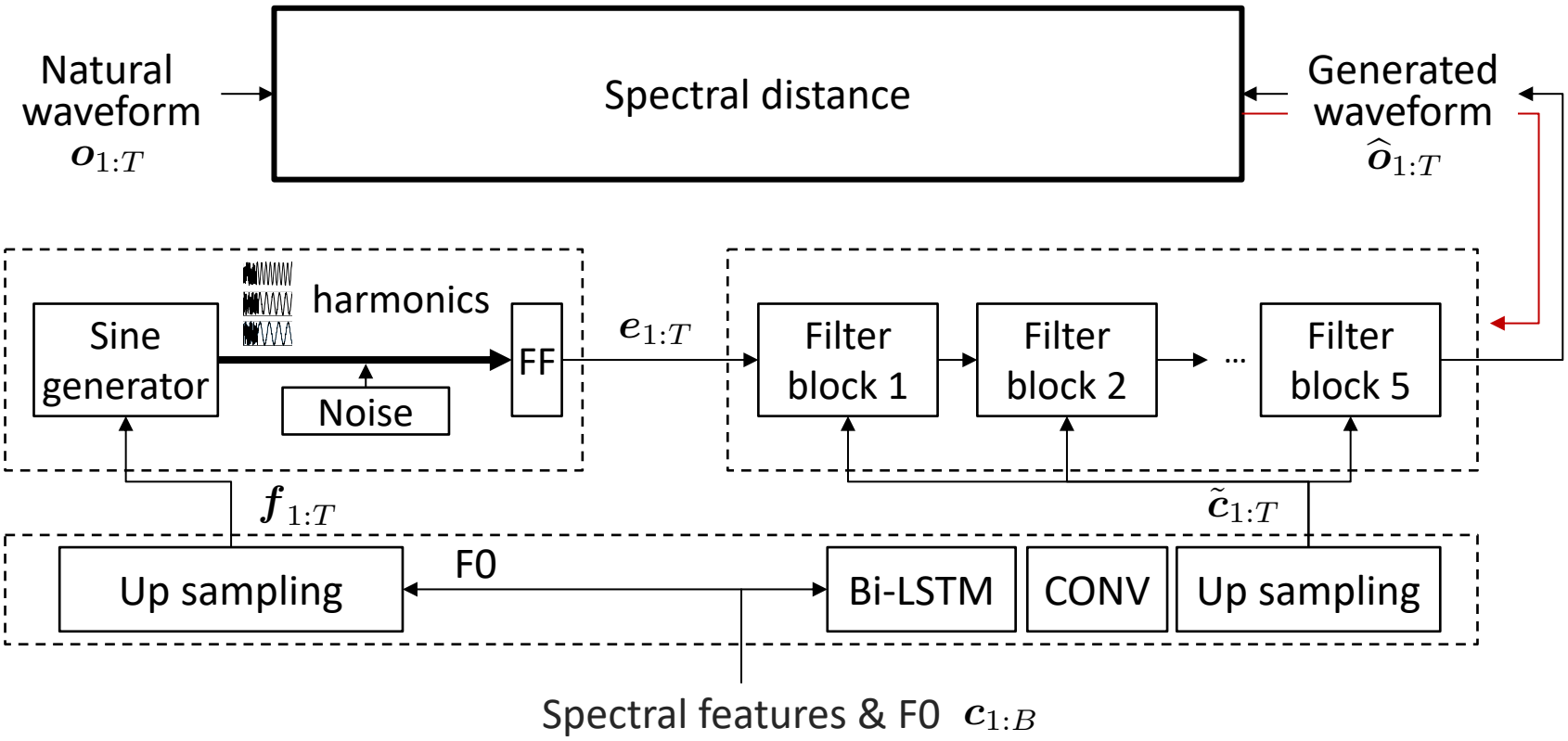




☛ Our WaveNet implementation: http://tonywangx.github.io/pdfs/wavenet.pdf

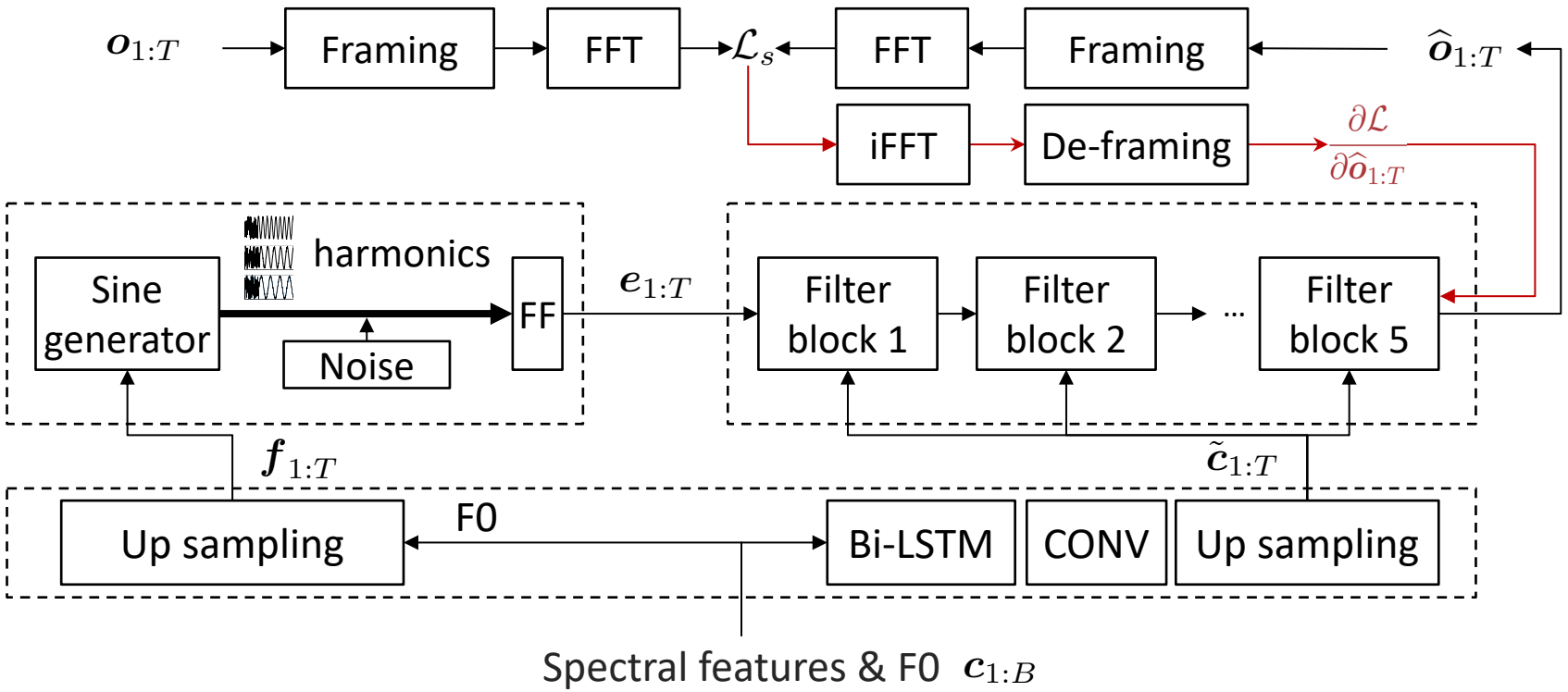
NEURAL SOURCE-FILTER MODEL

Training criterion



NEURAL SOURCE-FILTER MODEL

Training criterion

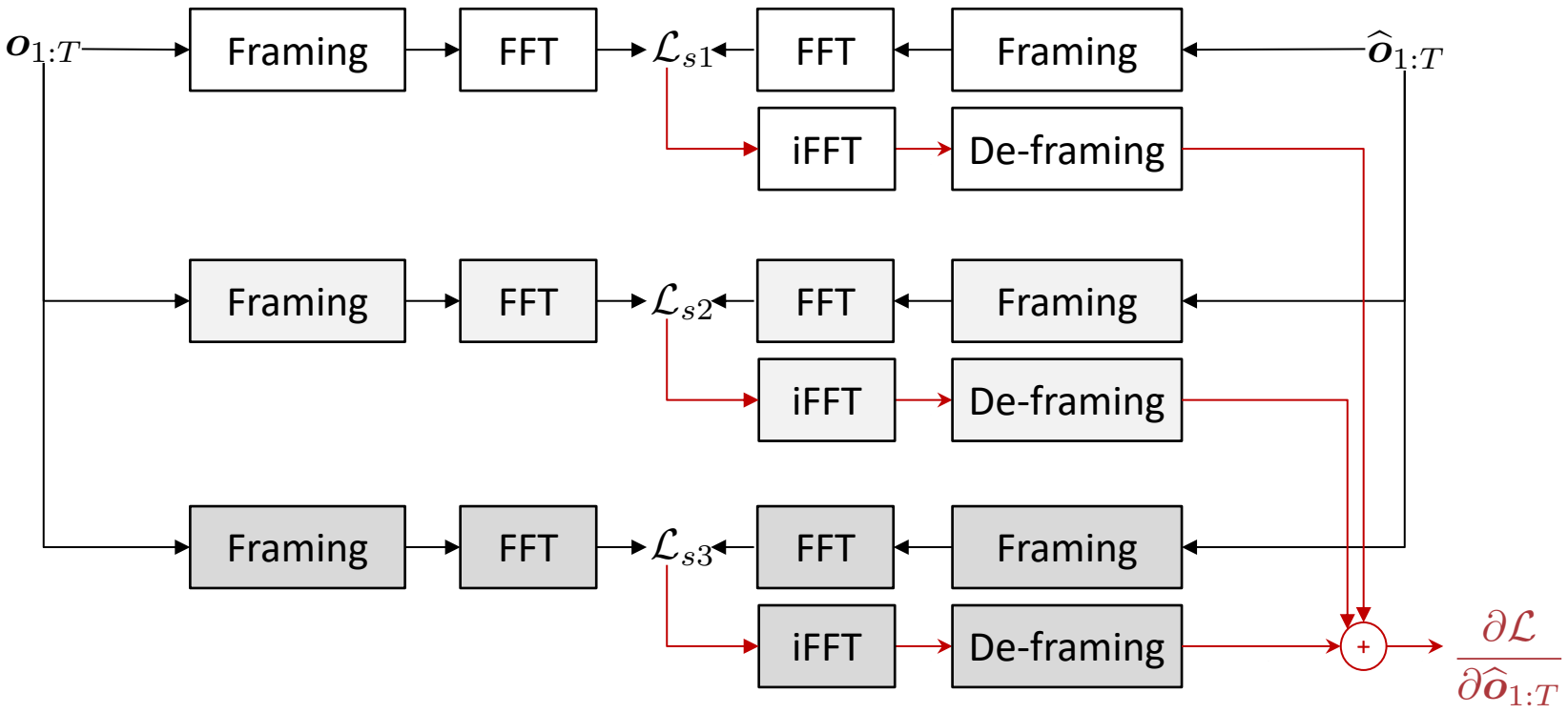


- Please check paper and:

  SLP-P22.8, *STFT spectral loss for training a neural speech waveform model*

NEURAL SOURCE-FILTER MODEL

Training criterion



- Different frame shifts / window lengths / FFT points
- Homogenous distances $\mathcal{L} = \mathcal{L}_{s1} + \mathcal{L}_{s2} + \mathcal{L}_{s3}$
- $$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{o}}_{1:T}} = \frac{\partial \mathcal{L}_{s1}}{\partial \hat{\boldsymbol{o}}_{1:T}} + \frac{\partial \mathcal{L}_{s2}}{\partial \hat{\boldsymbol{o}}_{1:T}} + \frac{\partial \mathcal{L}_{s3}}{\partial \hat{\boldsymbol{o}}_{1:T}}$$

CONTENTS

- Introduction
- Proposed model
- Experiments
  - Comparison with WaveNet
  - Ablation test
  - Analysis
- Summary

EXPERIMENTS

Test 1: comparison with WaveNet

- Data and feature

Corpus	Size	Note
ATR Ximera F009 [1]	15 hours	16kHz, Japanese, neutral style

	Feature	Dimension
Acoustic	Mel-generalized cepstrum coefficients (MGC)	60
	F0	1

- Models

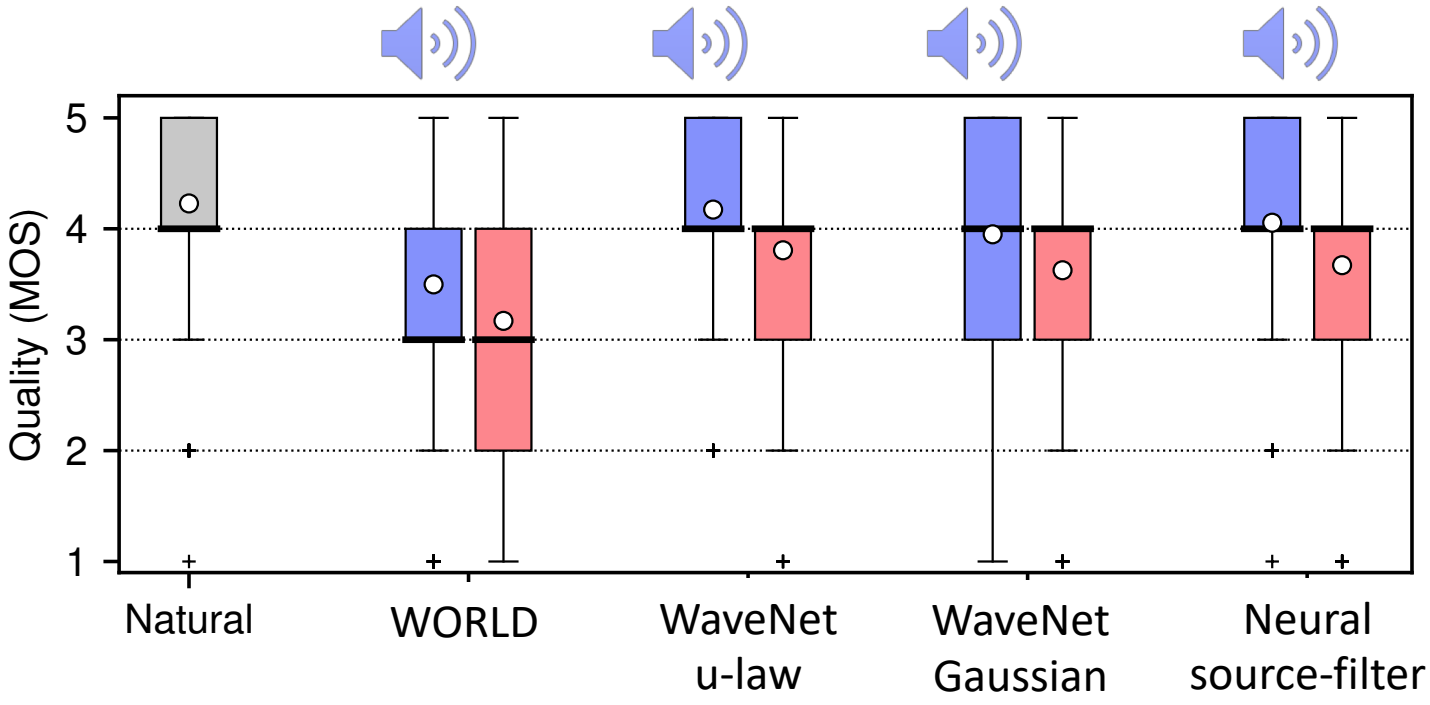
WOR	WORLD vocoder
WAD	WaveNet-vocoder for 10-bit discrete μ-law waveform
WAC	WaveNet-vocoder using Gaussian dist. for raw waveform
NSF	Proposed model for raw waveform

[1] Kawai, H., Toda, T., Ni, J., Tsuzaki, M., and Tokuda, K. (2004). Ximera: A new TTS from ATR based on corpus-based technologies. In Proc. SSW5, pages 179–184..

EXPERIMENTS

Test 1: comparison with WaveNet

- Speech quality
  - 245 paid evaluators, 1450 evaluation sets
    - Copy-synthesis: given natural MGC/F0
    - Pipeline TTS: given generated MGC/F0 from acoustic models

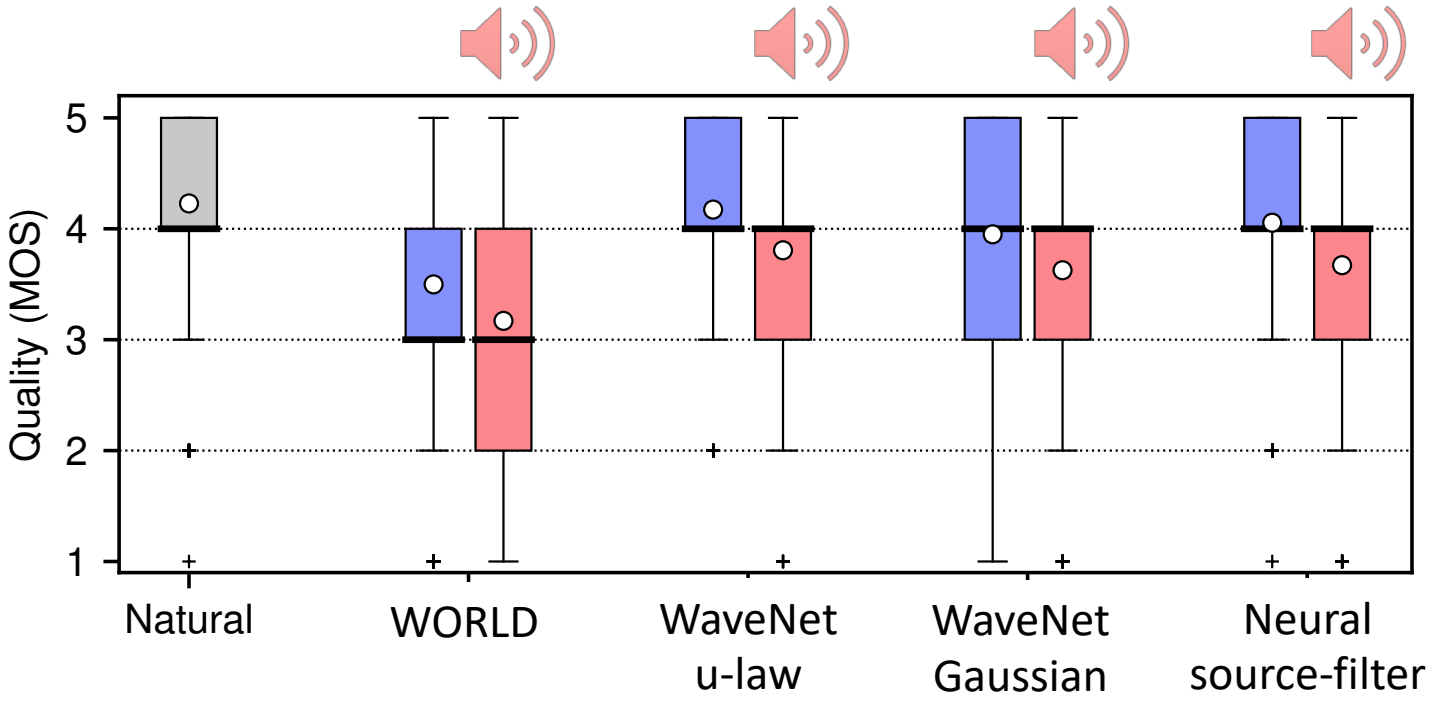


☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 1: comparison with WaveNet

- Speech quality
  - 245 paid evaluators, 1450 evaluation sets
    - Copy-synthesis: given natural MGC/F0
    - Pipeline TTS: given generated MGC/F0 from acoustic models



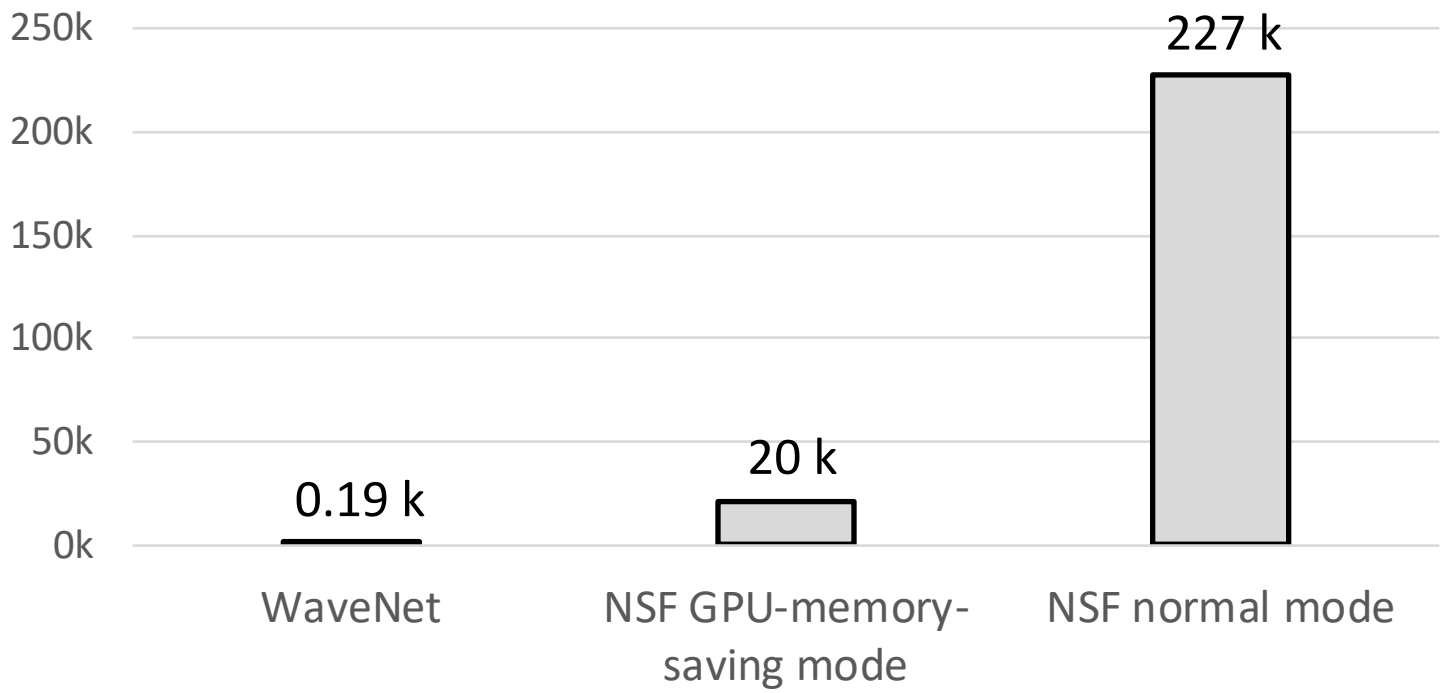
☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 1: comparison with WaveNet

- Generation speed

How many waveform sampling points can be generated in 1s GPU time?



- Memory-saving: release and allocate GPU memory layer by layer (pp. 69-73)
- Single GPU card, same DNN toolkit

☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 2: Ablation test

- One more corpus

Corpus	Size	Note
CMU-arctic SLT [2]	0.83 hours	16kHz, English

	Feature	Dimension
Acoustic	Mel-spectrogram	80
	F0	1

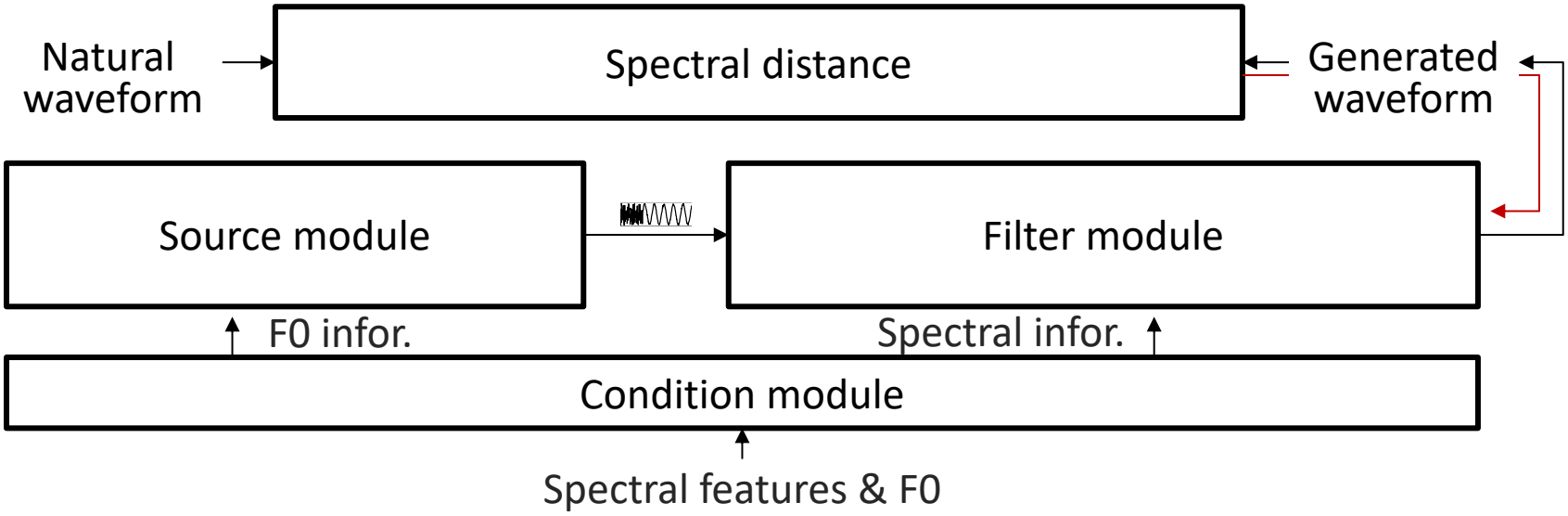
- Copy-synthesis

[2] J. Kominek and A. W. Black. The cmu arctic speech databases. In Fifth ISCA workshop on speech synthesis, 2004.

EXPERIMENTS

Test 2: Ablation test

	Natural	NSF	
ATR Ximera F009			
CMU-arctic SLT			

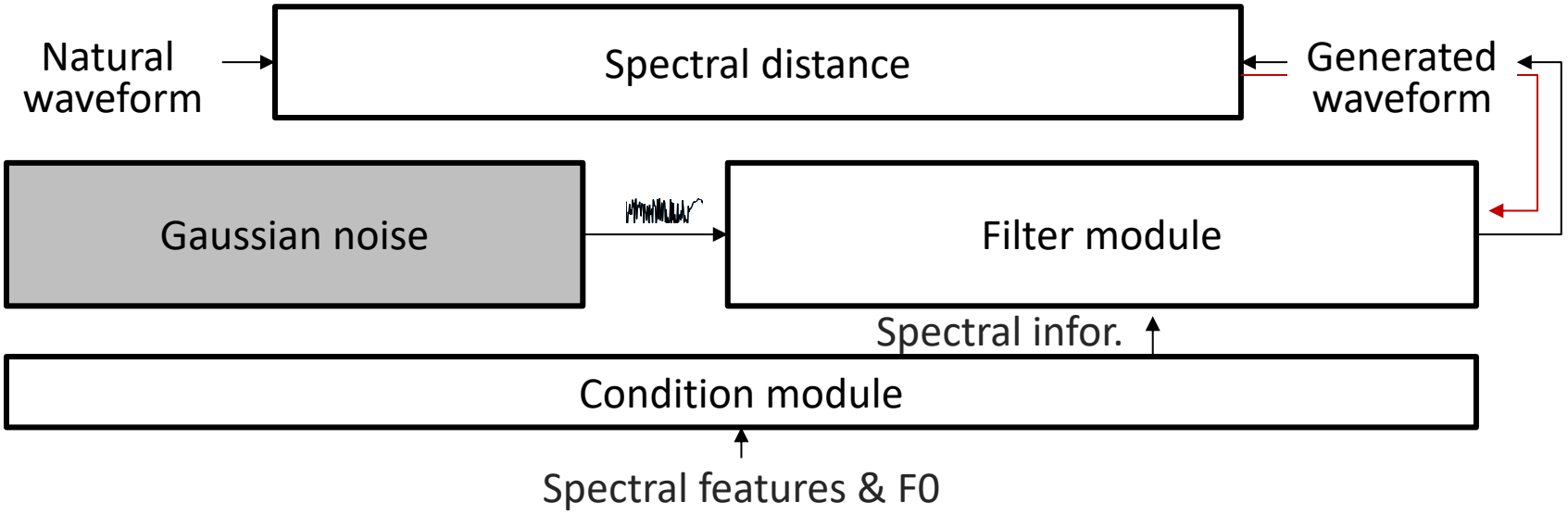


☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 2: Ablation test

	Natural	NSF	NSF w/o sine excitation
ATR Ximera F009			
CMU-arctic SLT			

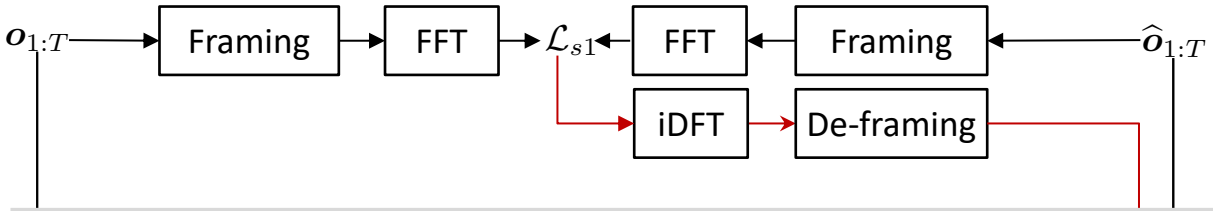


☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 2: Ablation test

	Natural	NSF	NSF only L_{s1}	
ATR Ximera F009				
CMU-arctic SLT				



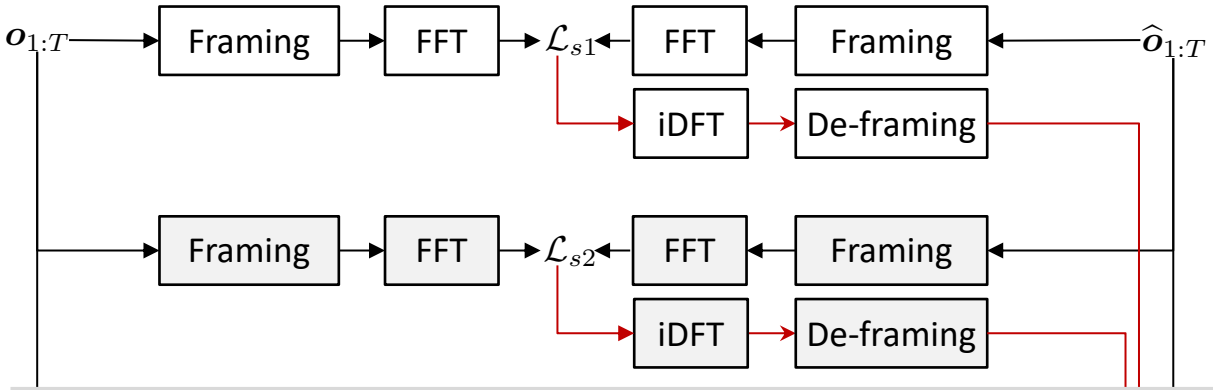
L_{s1}: frame shift 5ms, window length 20ms

☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

Test 2: Ablation test

	Natural	NSF	NSF only L_{s1}	NSF L_{s1}+L_{s2}
ATR Ximera F009				
CMU-arctic SLT				

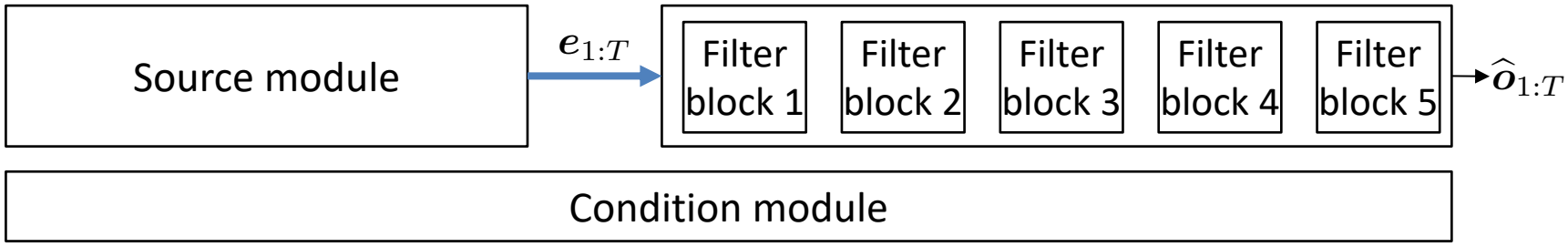


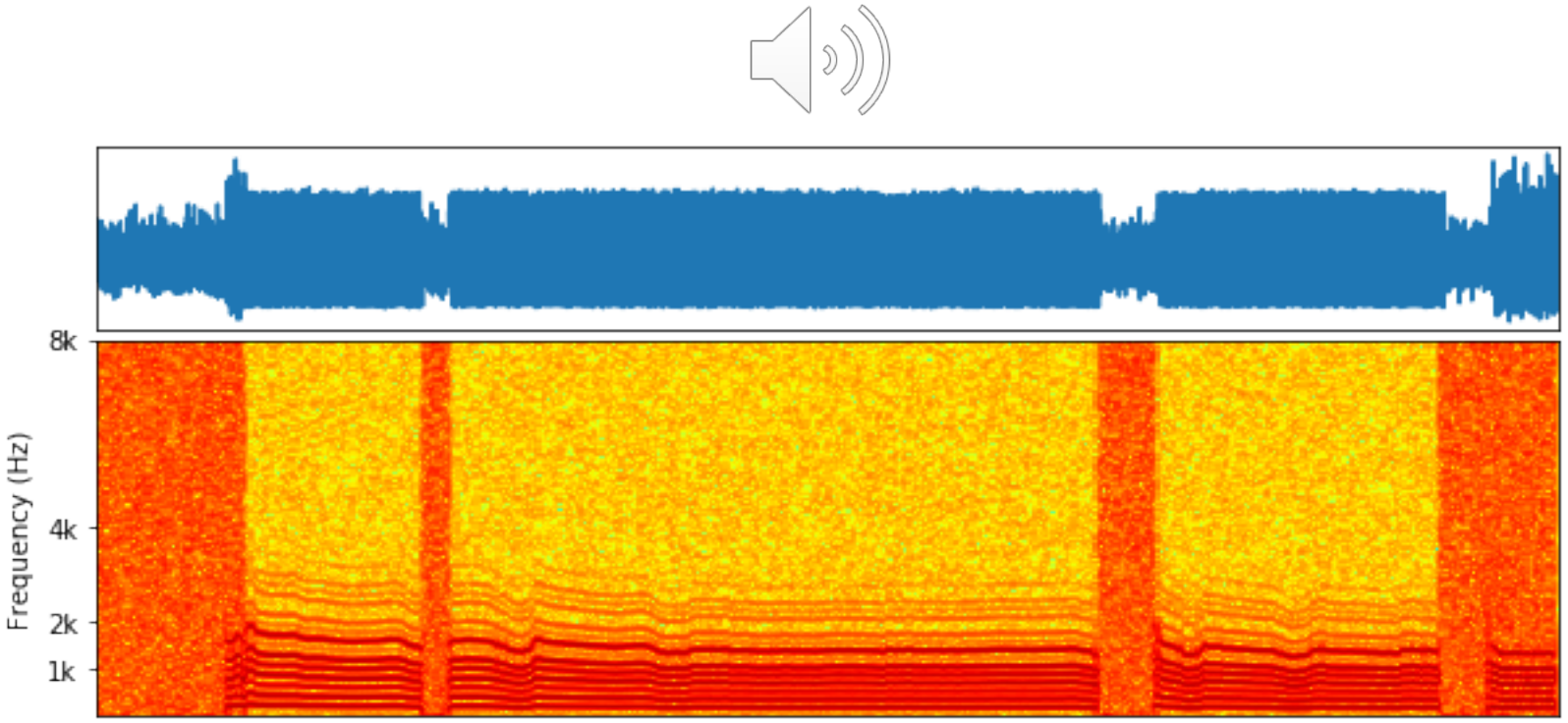
L_{s1}: frame shift 5ms, window length 20ms
L_{s2}: frame shift 2.5ms, window length 5ms

☛ Samples, models, codes: https://nii-yamagishilab.github.io/samples-nsf/index.html

EXPERIMENTS

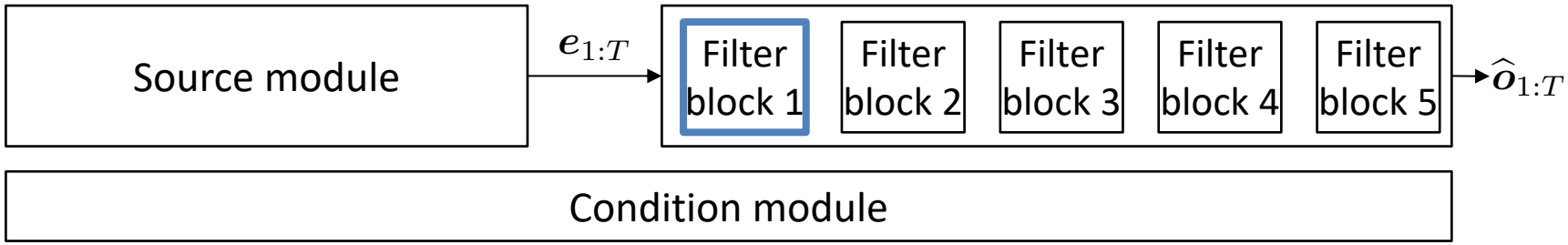
Analysis

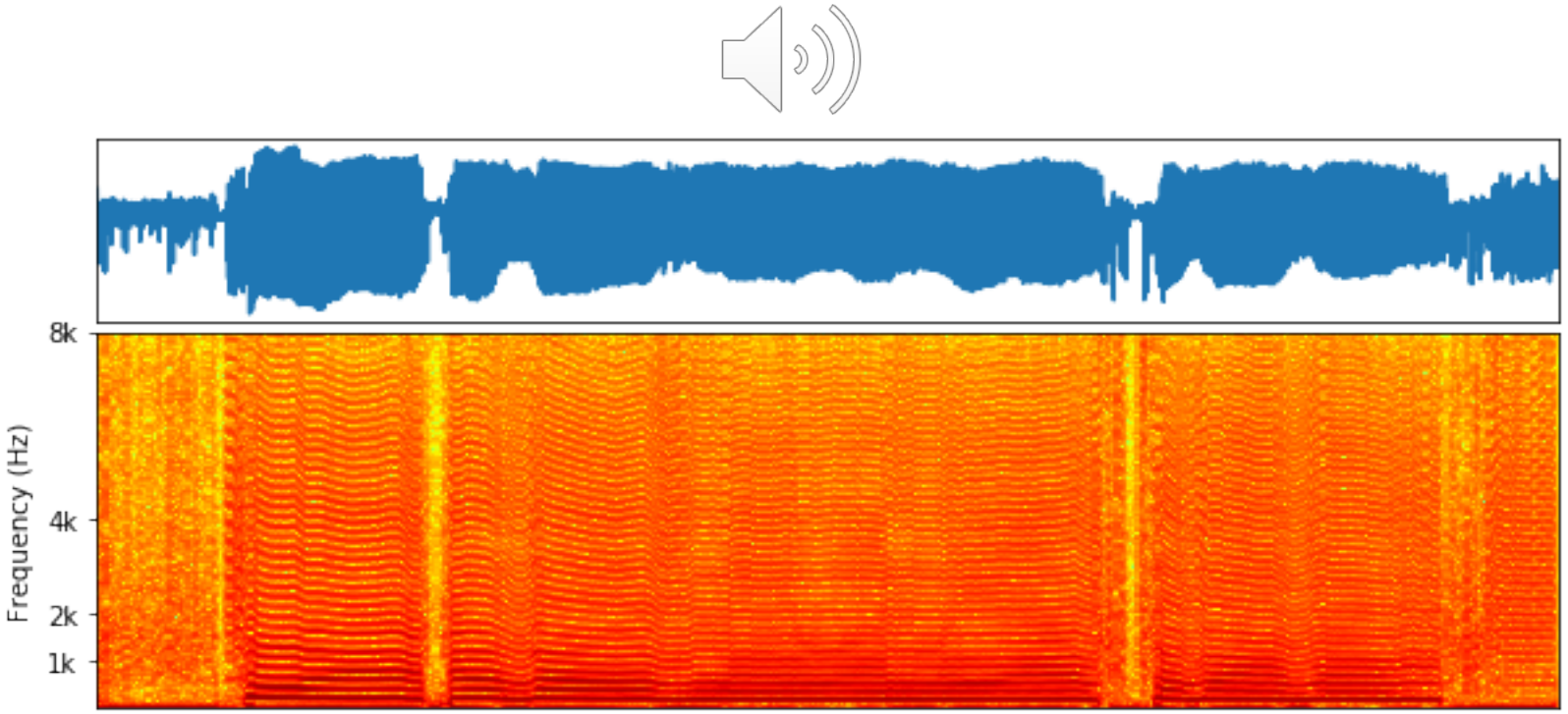




EXPERIMENTS

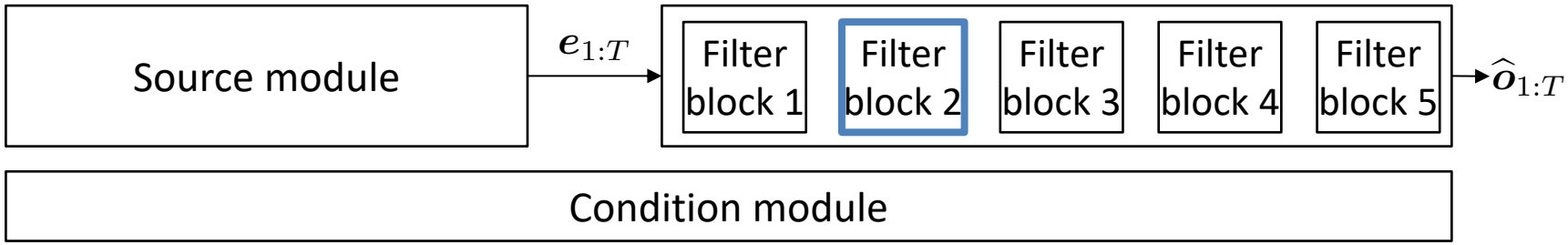
Analysis

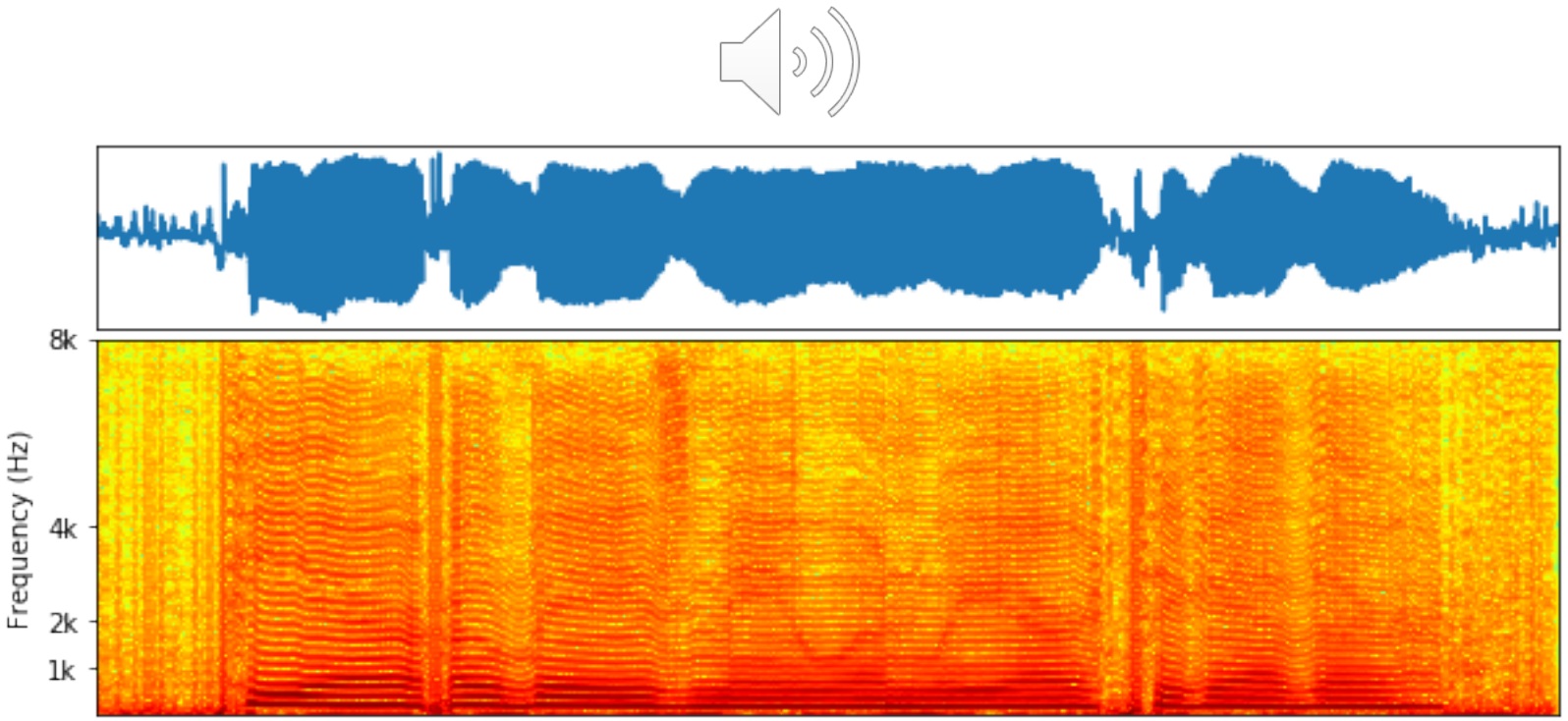




EXPERIMENTS

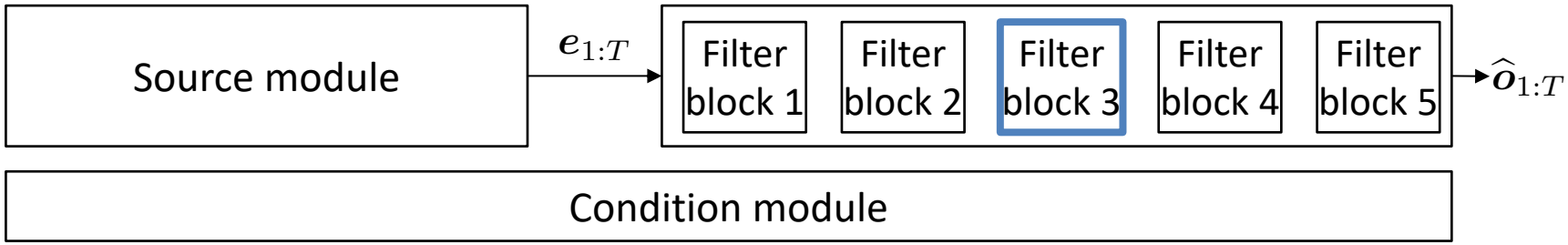
Analysis

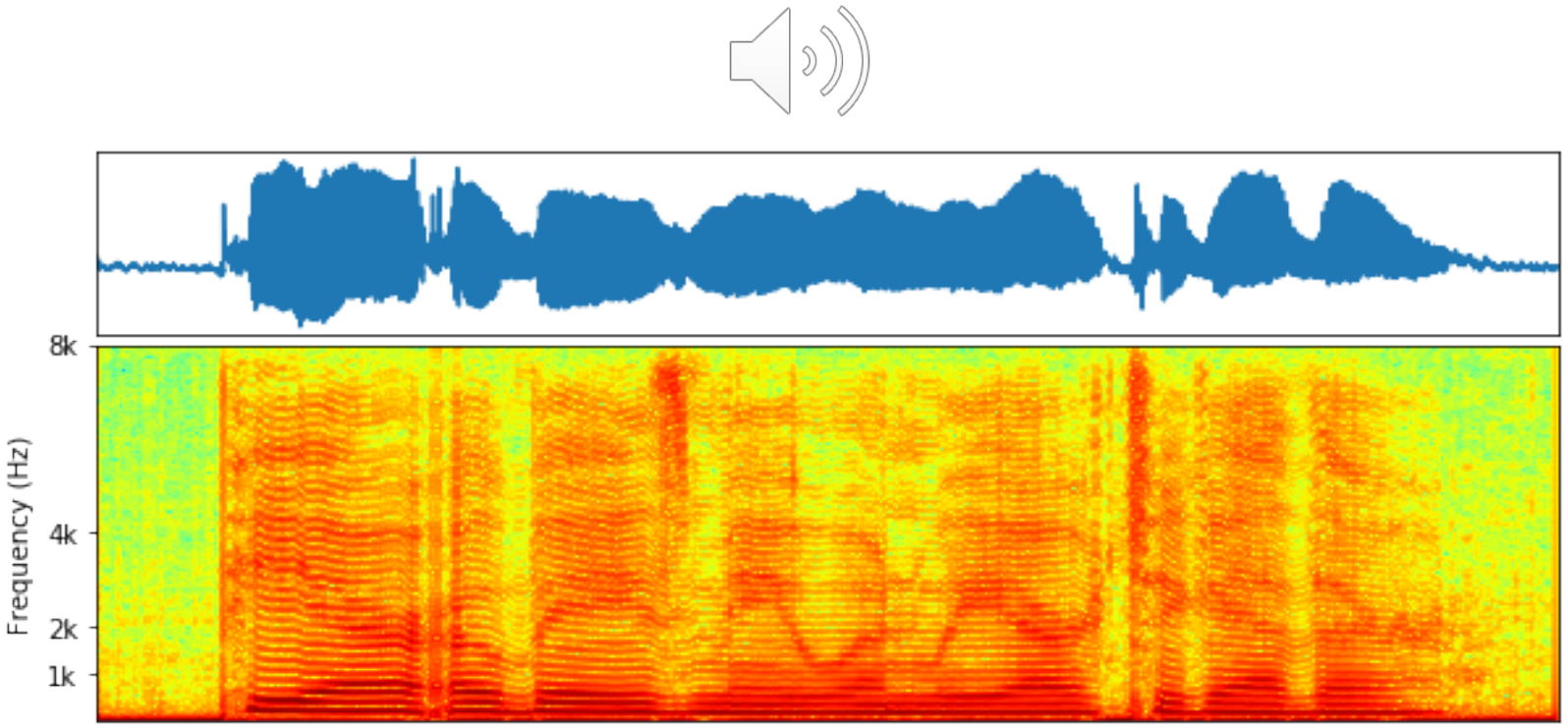




EXPERIMENTS

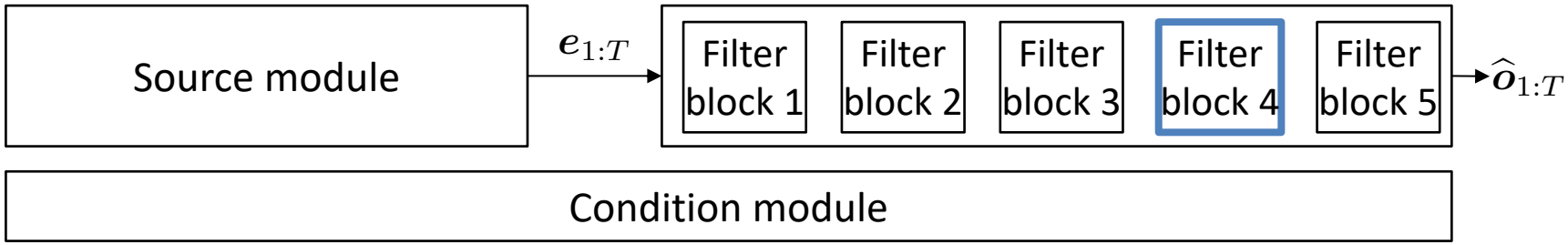
Analysis

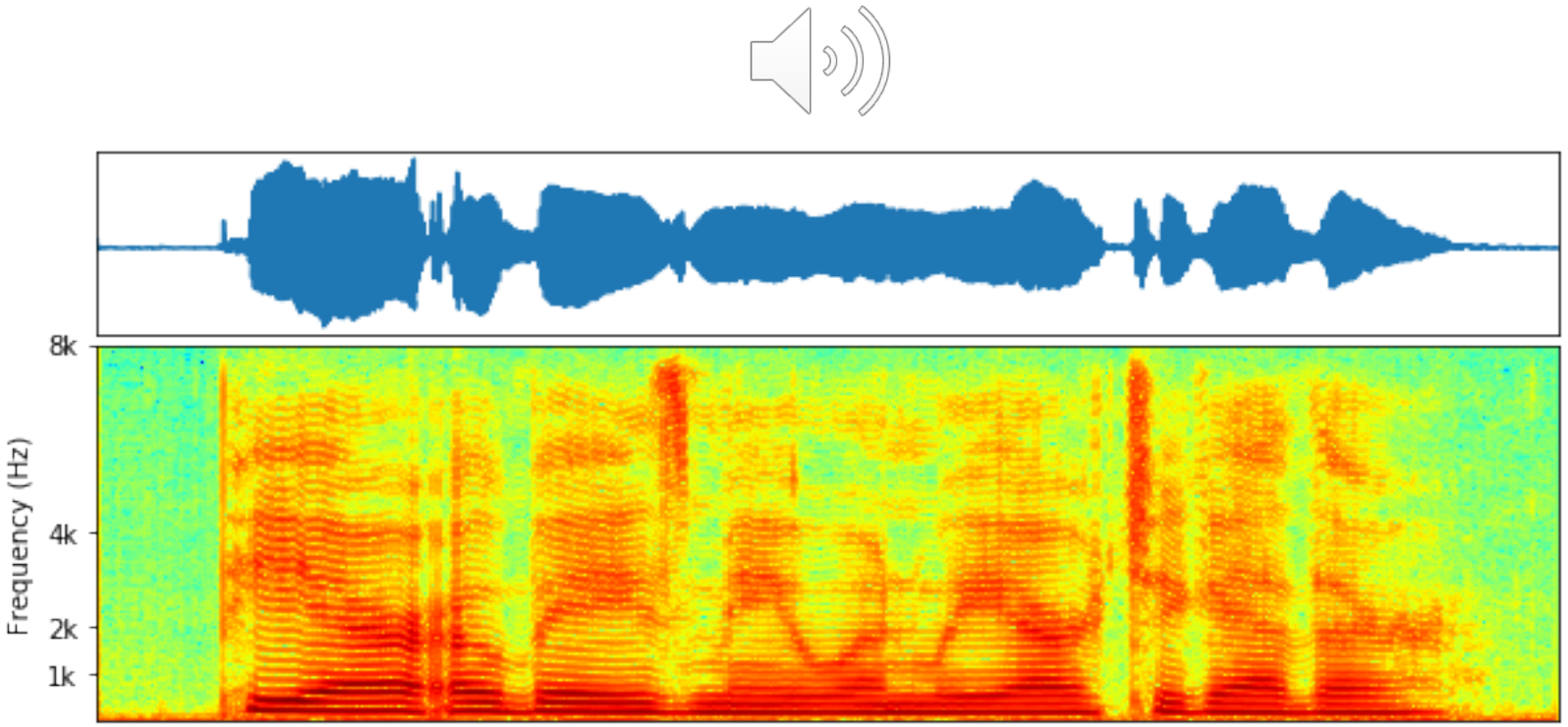




EXPERIMENTS

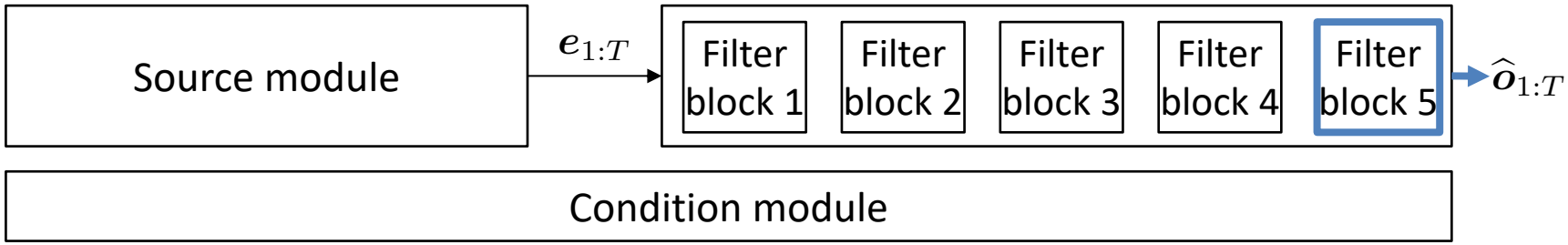
Analysis

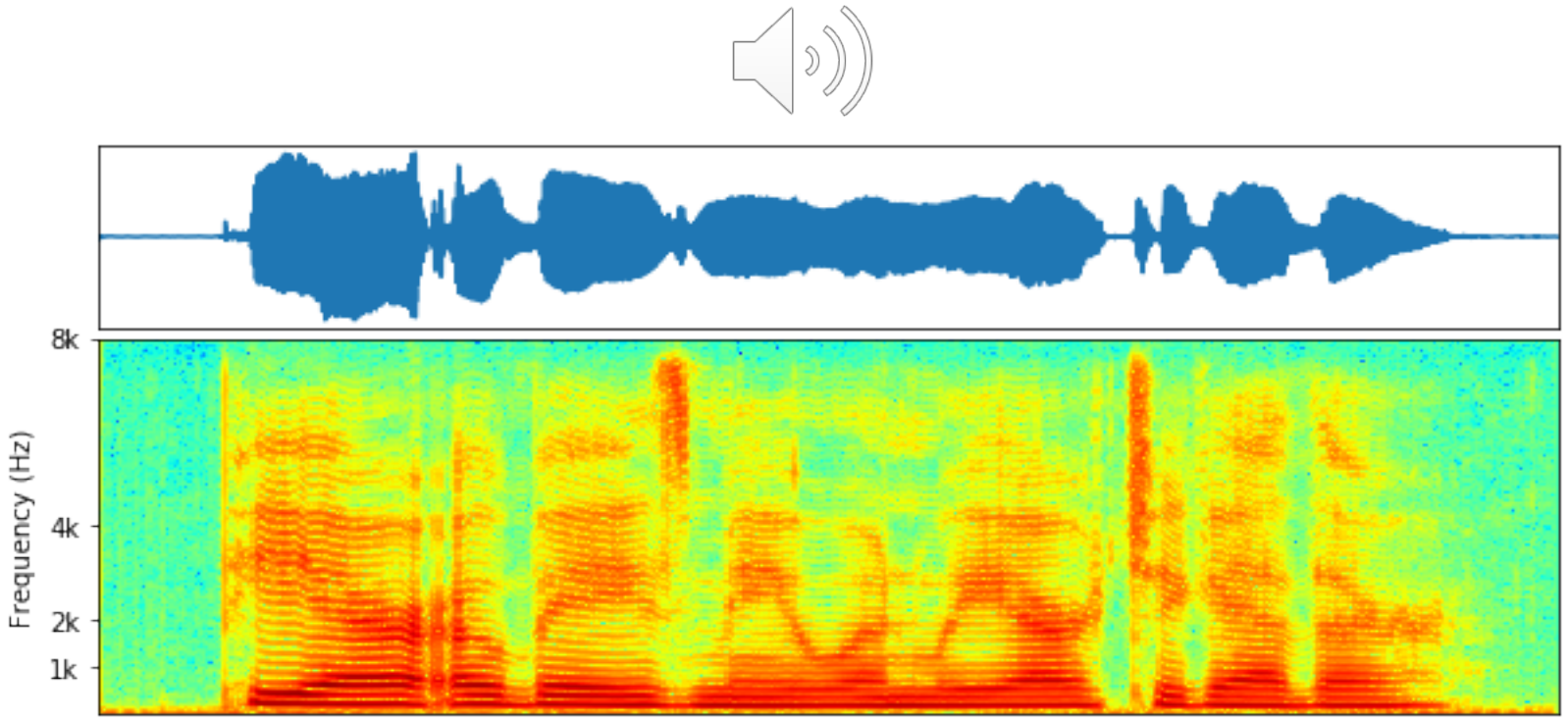




EXPERIMENTS

Analysis



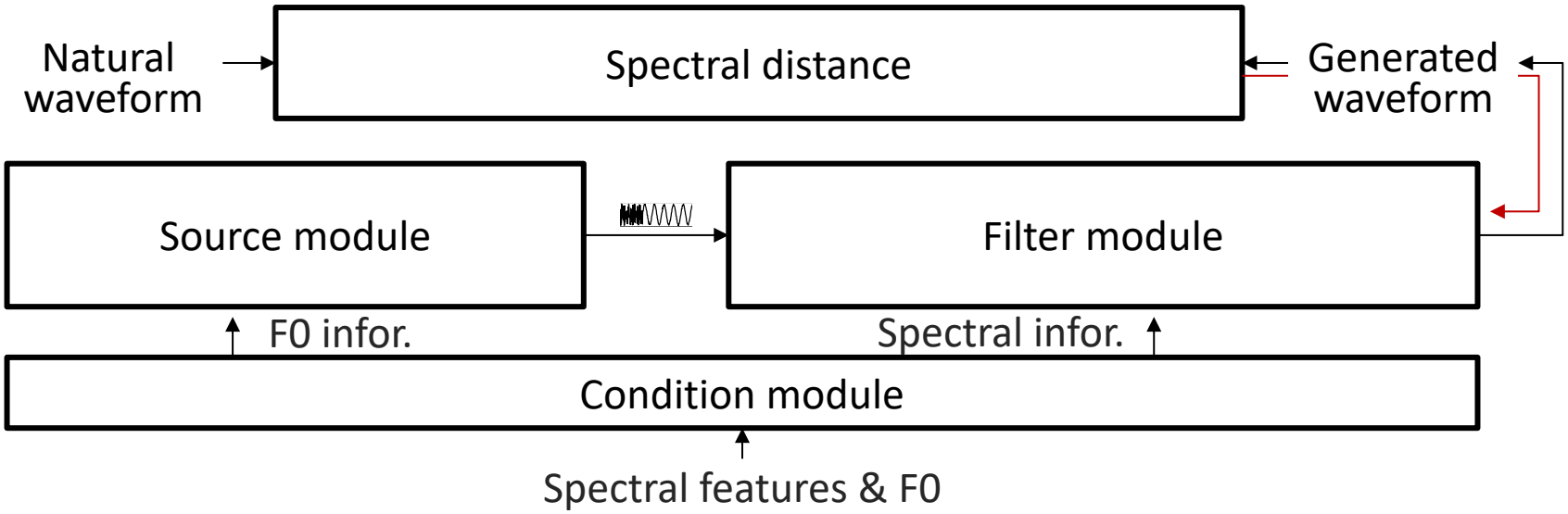


CONTENTS

- Introduction
- Proposed model
- Experiments
- Summary

SUMMARY

NSF



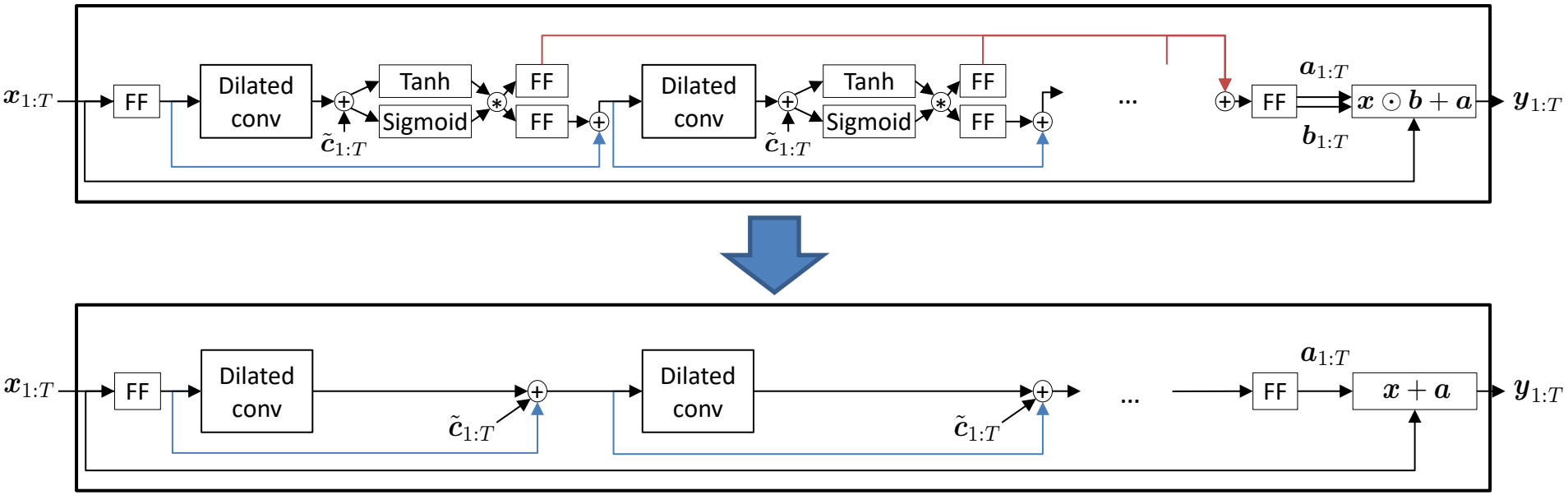
- Condition: up sampling
- Source: sine excitation
- Filter: dilated-CONV
- Criterion: STFT

Easy training
Fast generation
High-quality waveforms

SUMMARY

Recent work

- Simplified NSF: faster generation speed



- Simplified NSF -> Harmonic-plus-noise NSF
  - Best quality

Questions & Comments are always Welcome!

https://nii-yamagishilab.github.io/samples-nsf/index.html

https://arxiv.org/abs/1904.12088

REFERENCE

WaveNet: A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.

SampleRNN: S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. Samplernn: An unconditional end-to-end neural audio generation model. arXiv preprint arXiv:1612.07837, 2016.

WaveRNN: N. Kalchbrenner, E. Elsen, K. Simonyan, et.al. Efficient neural audio synthesis. In J. Dy and A. Krause, editors, Proc. ICML, volume 80 of Proceedings of Machine Learning Research, pages 2410–2419, 10–15 Jul 2018.

FFTNet: Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu. FFTNet: A real-time speaker-dependent neural vocoder. In Proc. ICASSP, pages 2251–2255. IEEE, 2018.

Universal vocoder: J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, and R. Barra-Chicote. Robust universal neural vocoding. arXiv preprint arXiv:1811.06292, 2018.

Subband WaveNet: T. Okamoto, K. Tachibana, T. Toda, Y. Shiga, and H. Kawai. An investigation of subband wavenet vocoder covering entire audible frequency range with limited acoustic features. In Proc. ICASSP, pages 5654–5658. 2018.

Parallel WaveNet: A. van den Oord, Y. Li, I. Babuschkin, et. al.. Parallel WaveNet: Fast high-fidelity speech synthesis. In Proc. ICML, pages 3918–3926, 2018.

ClariNet: W. Ping, K. Peng, and J. Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. arXiv preprint arXiv:1807.07281, 2018.

FlowWaveNet: S. Kim, S.-g. Lee, J. Song, and S. Yoon. Flowavenet: A generative flow for raw audio. arXiv preprint arXiv:1811.02155, 2018.

WaveGlow: R. Prenger, R. Valle, and B. Catanzaro. Waveglow: A flow-based generative network for speech synthesis. arXiv preprint arXiv:1811.00002, 2018.

RNN+STFT: S. Takaki, T. Nakashika, X. Wang, and J. Yamagishi. STFT spectral loss for training a neural speech waveform model. In Proc. ICASSP (submitted), 2018.

NSF: X. Wang, S. Takaki, and J. Yamagishi. Neural source-filter-based waveform model for statistical para- metric speech synthesis. arXiv preprint arXiv:1810.11946, 2018.

LP-WavNet: M.-J. Hwang, F. Soong, F. Xie, X. Wang, and H.-G. Kang. Lp-wavenet: Linear prediction-based wavenet speech synthesis. arXiv preprint arXiv:1811.11913, 2018.

GlotNet: L. Juvela, V. Tsiaras, B. Bollepalli, M. Airaksinen, J. Yamagishi, and P. Alku. Speaker-independent raw waveform model for glottal excitation. arXiv preprint arXiv:1804.09593, 2018.

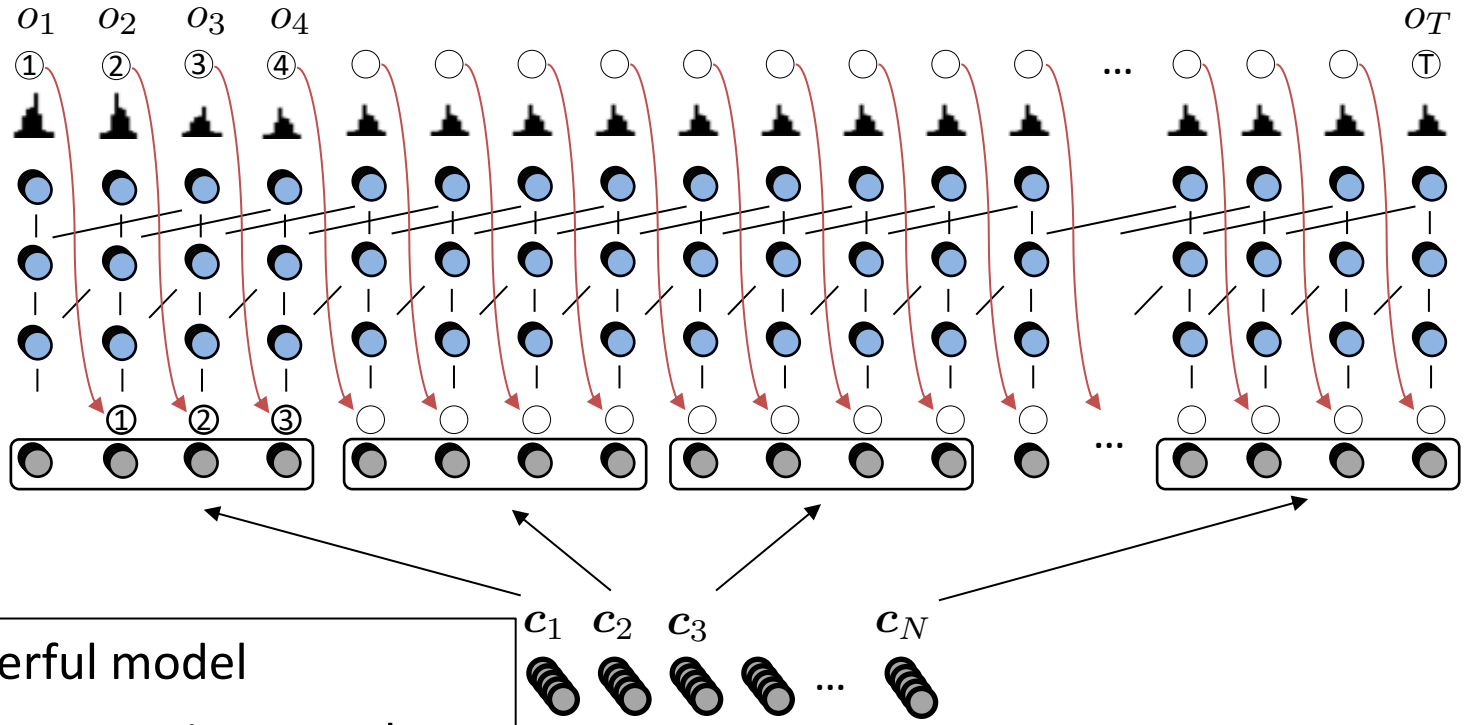
ExcitNet: E. Song, K. Byun, and H.-G. Kang. Excitnet vocoder: A neural excitation model for parametric speech synthesis systems. arXiv preprint arXiv:1811.04769, 2018.

LPCNet: J.-M. Valin and J. Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. arXiv preprint arXiv:1810.11846, 2018.

INTRODUCTION

Autoregressive (AR) model

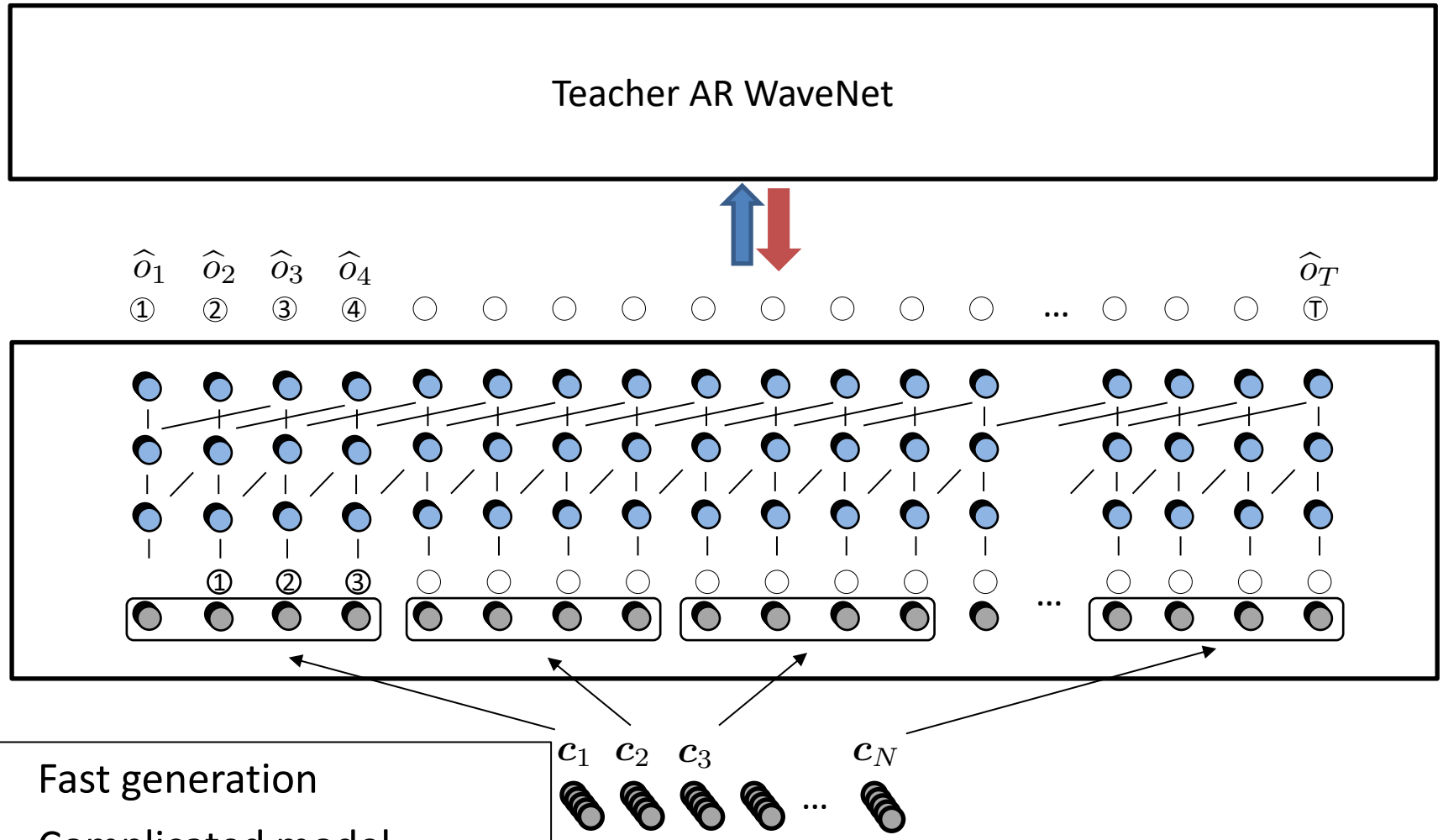
$$p(\boldsymbol{o}_{1:T}|\boldsymbol{c}_{1:N};\boldsymbol{\Theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1},\boldsymbol{c}_{1:N};\boldsymbol{\Theta})$$



- Powerful model
- ! Slow generation speed

INTRODUCTION

Normalizing flow + knowledge distilling

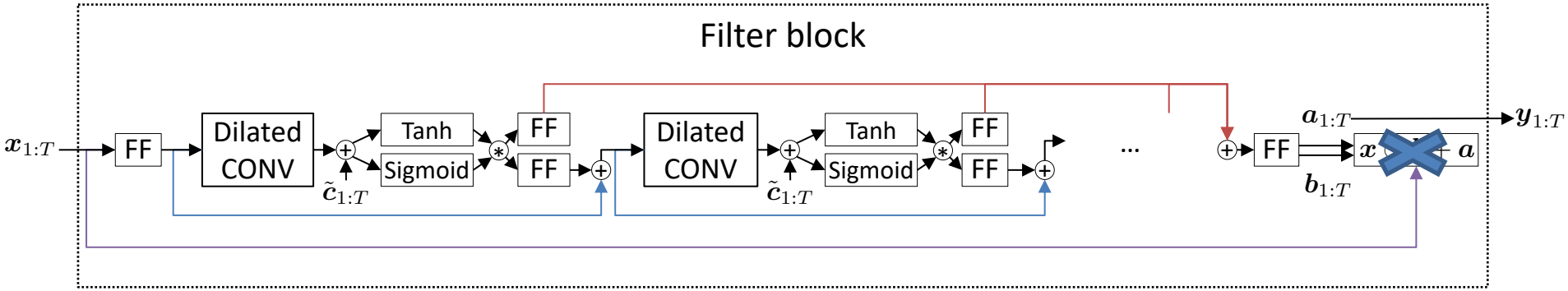


- Fast generation
- ! Complicated model

EXPERIMENTS

E2: Ablation test

	Natural	NSF	NSF w/o 'residual' connection in filter module
F009			
CMU-arctic			



https://nii-yamagishilab.github.io/samples-nsf/index.html

SUMMARY

Recent work

- ❑ Input F0?

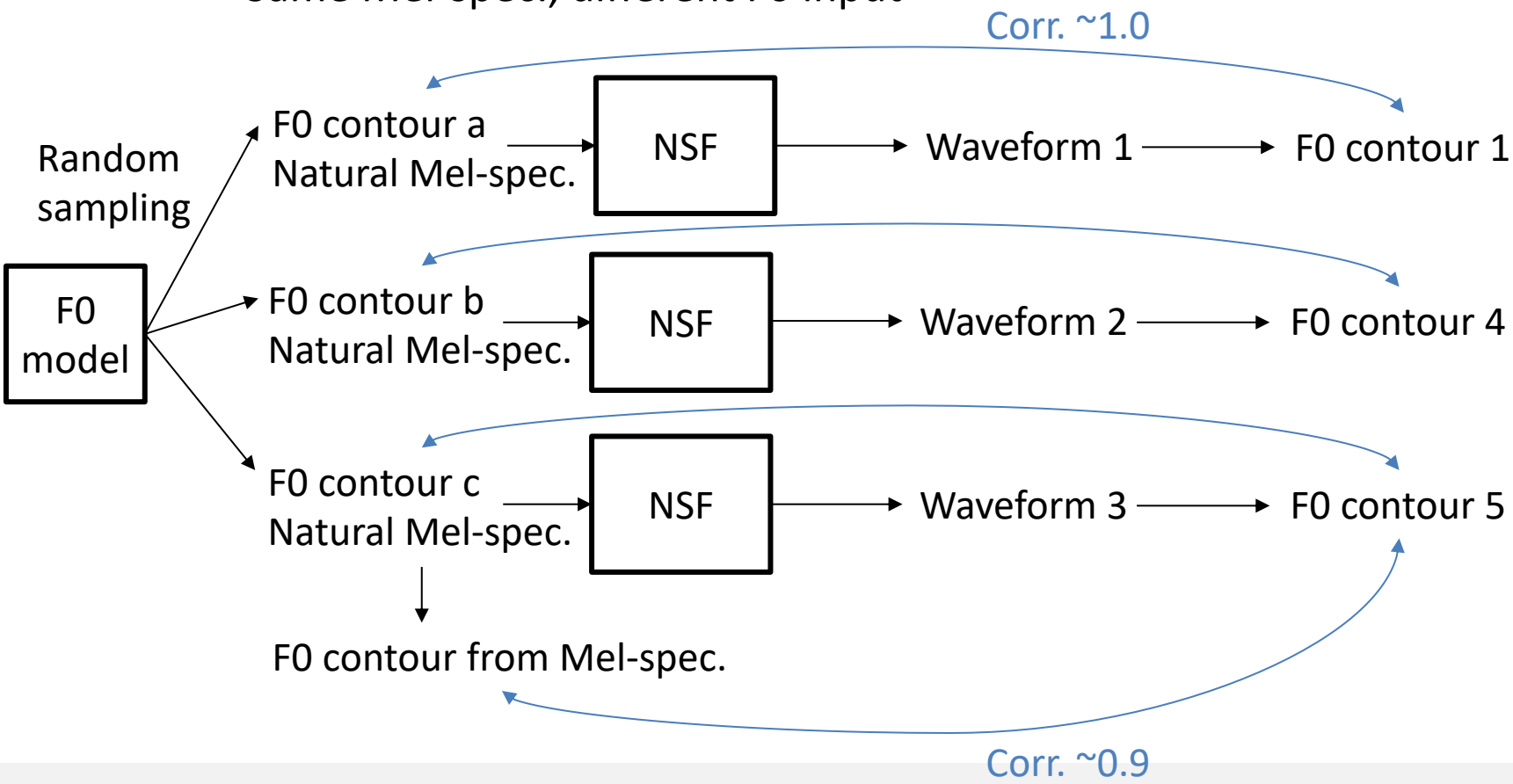
  - VCTK corpus, Mel-spectrogram + F0
  - Samples generated from NSF, by changing F0 only

F0	0.5 * F0	2.0 * F0

SUMMARY

Recent work

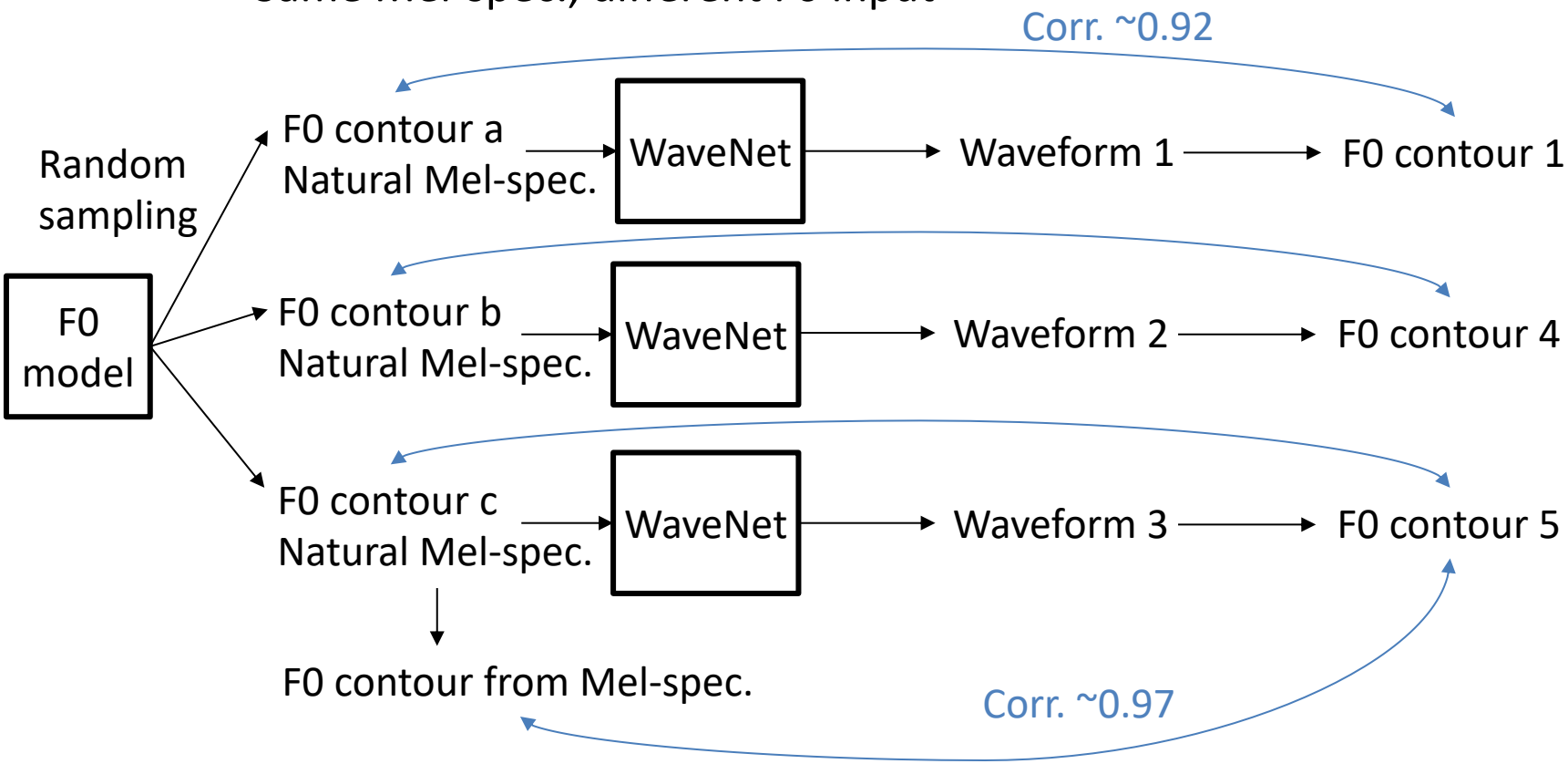
- Input F0?
  - https://arxiv.org/abs/1904.12088
  - Same Mel-spec., different F0 input



SUMMARY

Recent work

- Input F0?
  - https://arxiv.org/abs/1904.12088
  - Same Mel-spec., different F0 input



- Not easy to control F0 of generated waveforms from WaveNet

SIMPLIFIED NSF

Comparison with original NSF

- Waveform generation speed

Number of waveform values generated in 1s GPU time

AR WaveNet	Original NSF		Simplified NSF	
	Memory save	Full speed	Memory save	Full speed
190	20,000	227,000	**88,000**	**327,000**

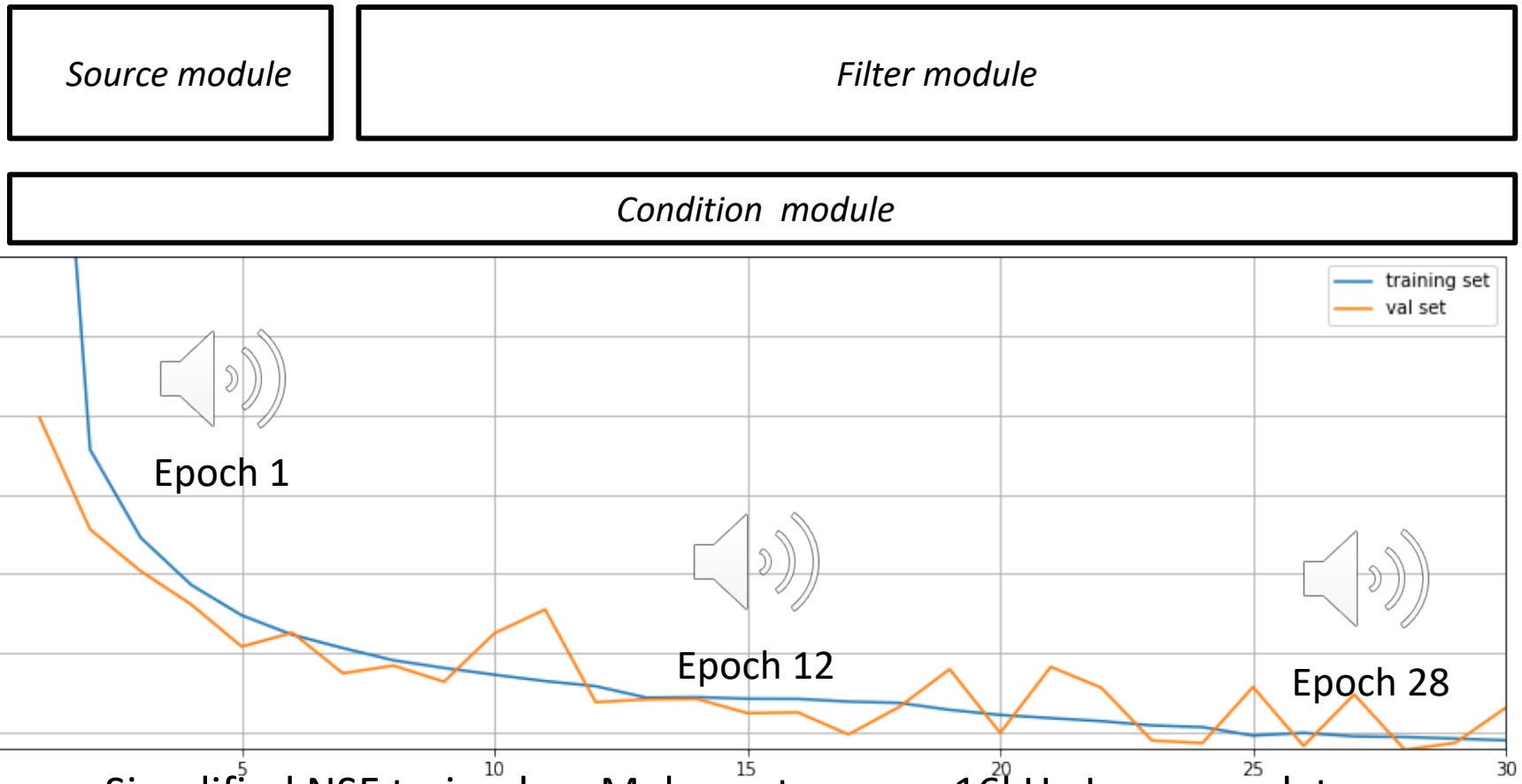
- Number of model parameters

AR WaveNet	Original NSF	Simplified NSF
~ 2.9 M	~1.8 M	~1.07 M

  - Memory-save mode: release and allocate GPU memory layer by layer

SIMPLIFIED NSF

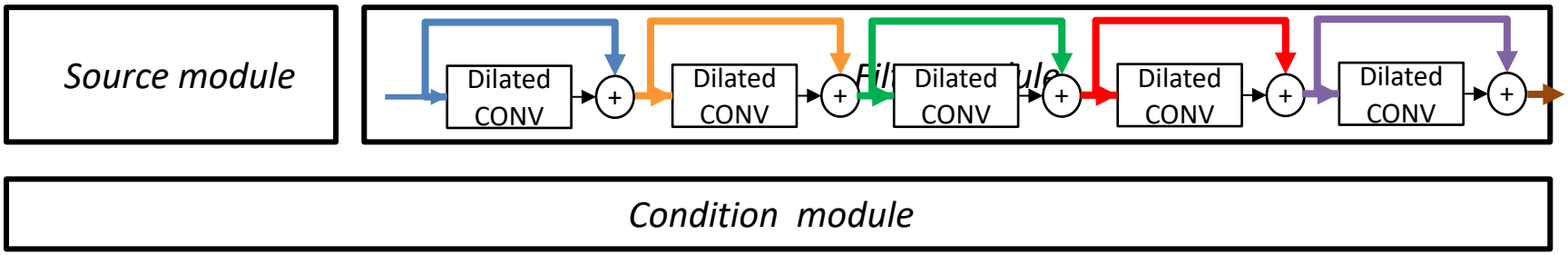
Improve the simplified NSF?



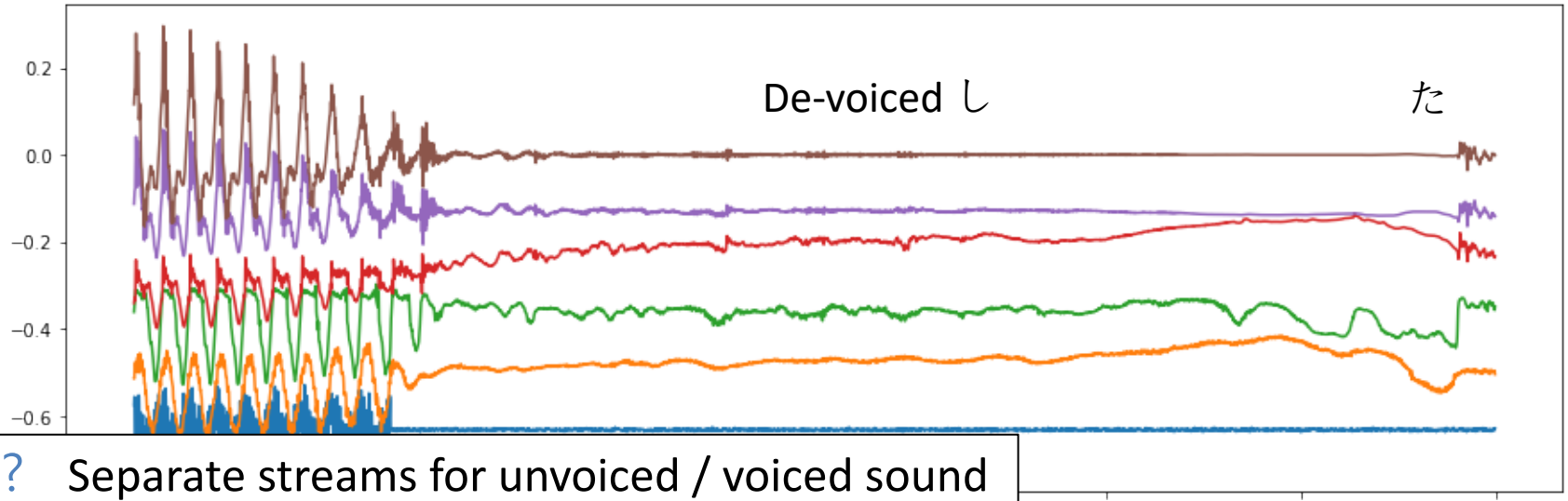
- Simplified NSF trained on Mel-spectrogram, 16kHz Japanese data
- Unvoiced sound disappears

SIMPLIFIED NSF

Improve the simplified NSF?



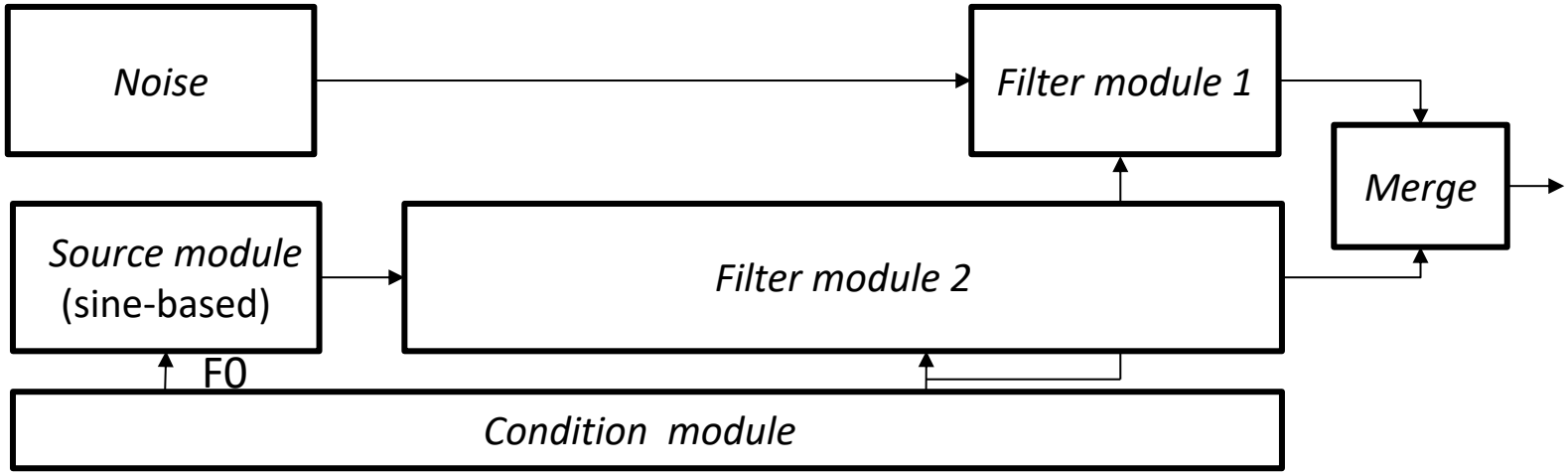
… 金融調節を実施した。



? Separate streams for unvoiced / voiced sound

HARMONIC-PLUS-NOISE NSF

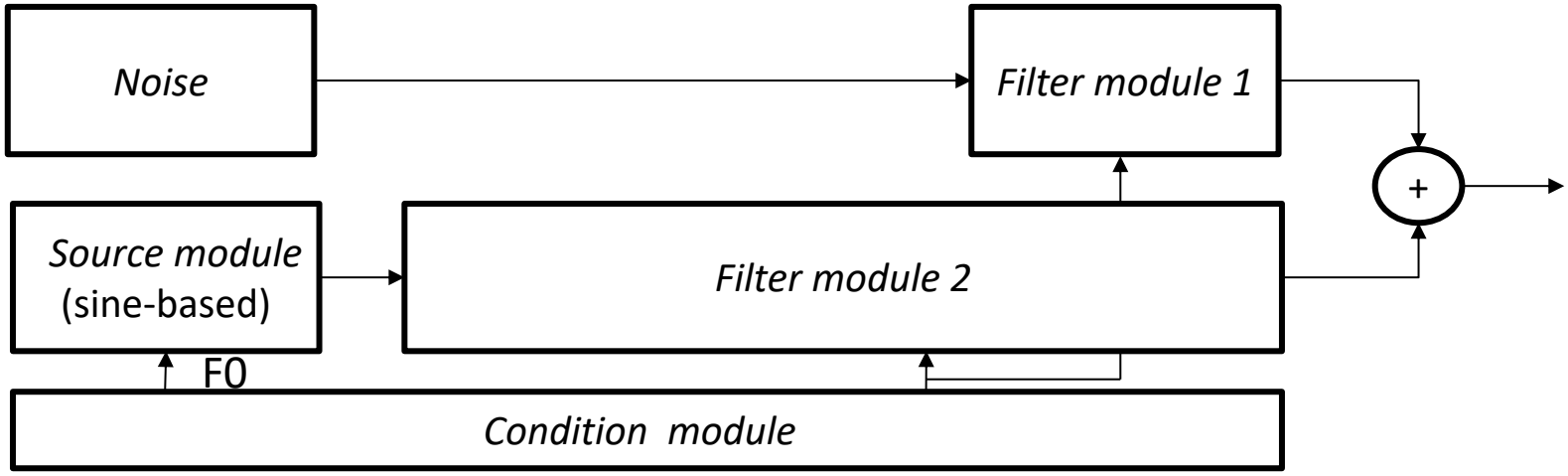
HPN NSF



- Harmonic + noise style
- How to merge?

HARMONIC-PLUS-NOISE NSF

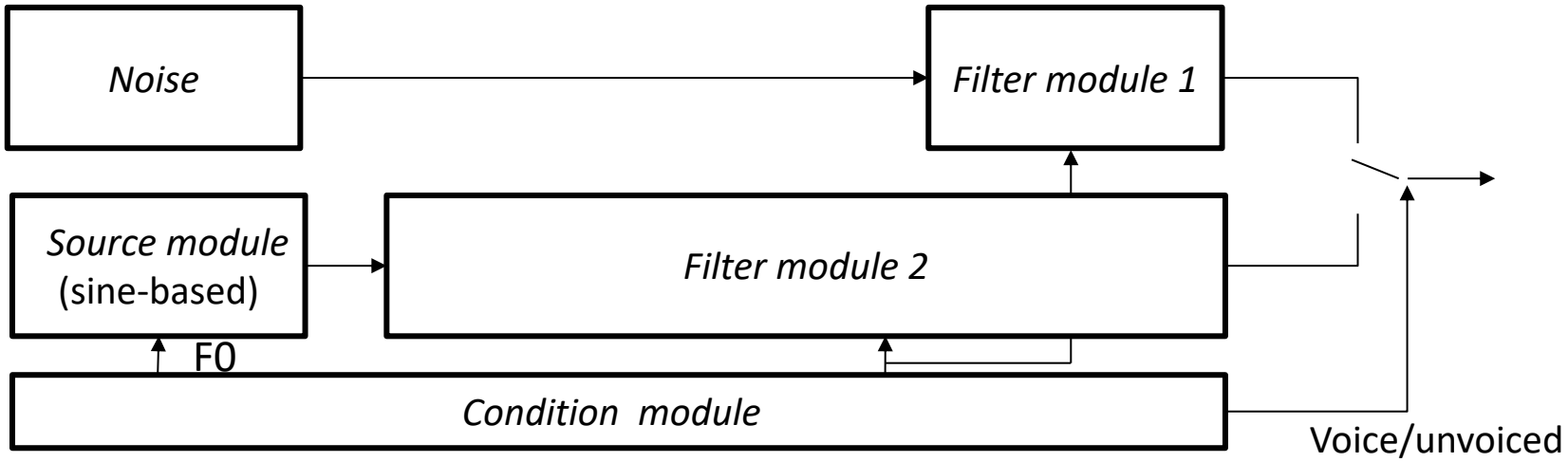
HPN NSF



- Harmonic + noise style
- How to merge?
  - Summation: may not separate harmonic from noise

HARMONIC-PLUS-NOISE NSF

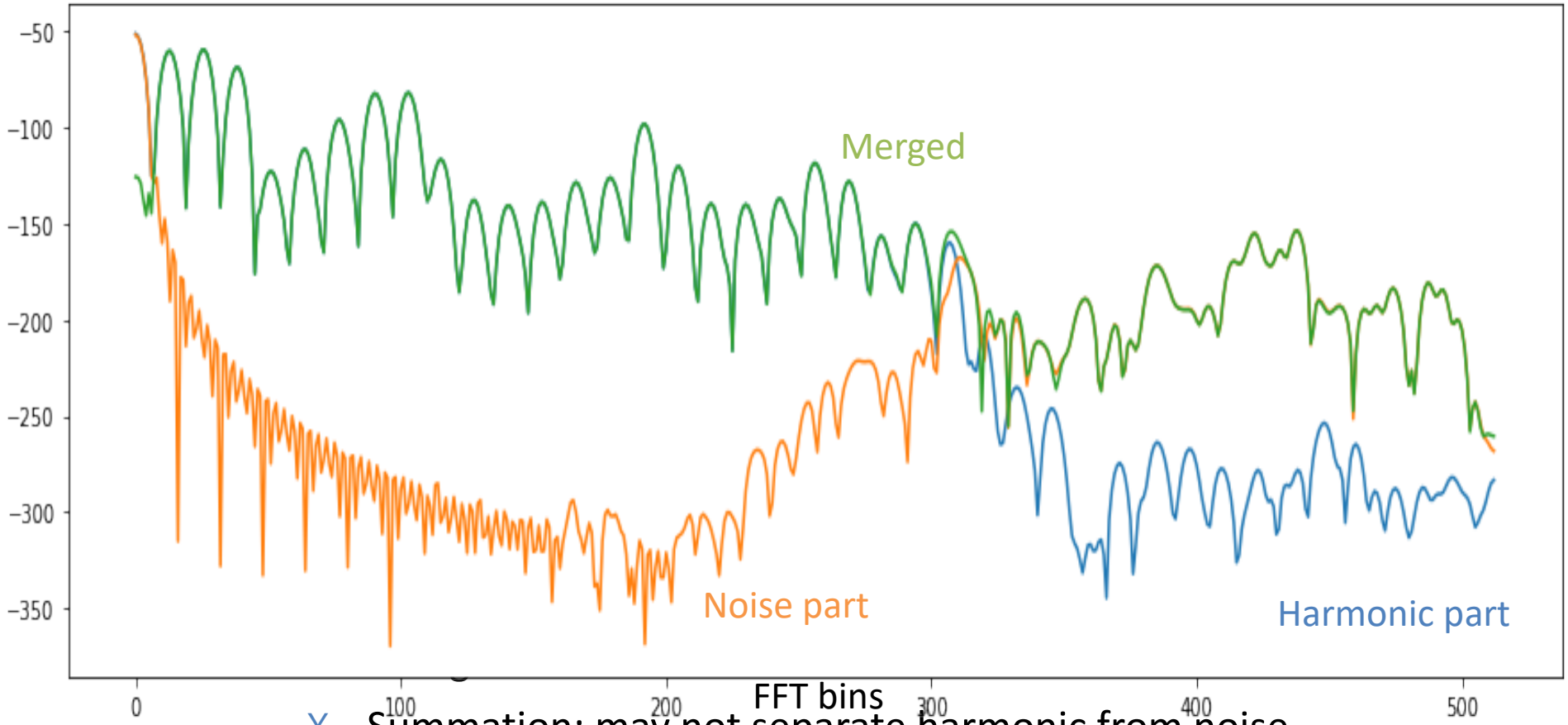
HPN NSF



- Harmonic + noise style
- How to merge?
  - X Summation: may not separate harmonic from noise
  - X Merge in time domain: hard / soft switch

HARMONIC-PLUS-NOISE NSF

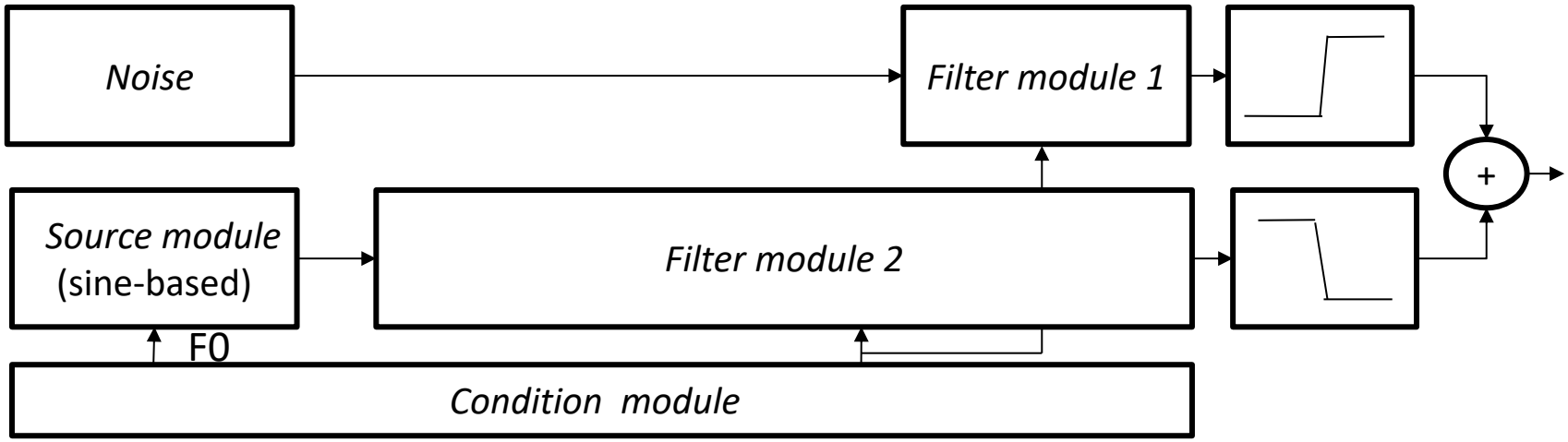
HPN NSF



- X Summation: may not separate harmonic from noise
- X Merge in time domain: hard / soft switch
- ✓ Merge in frequency domain: FIR filters

HARMONIC-PLUS-NOISE NSF

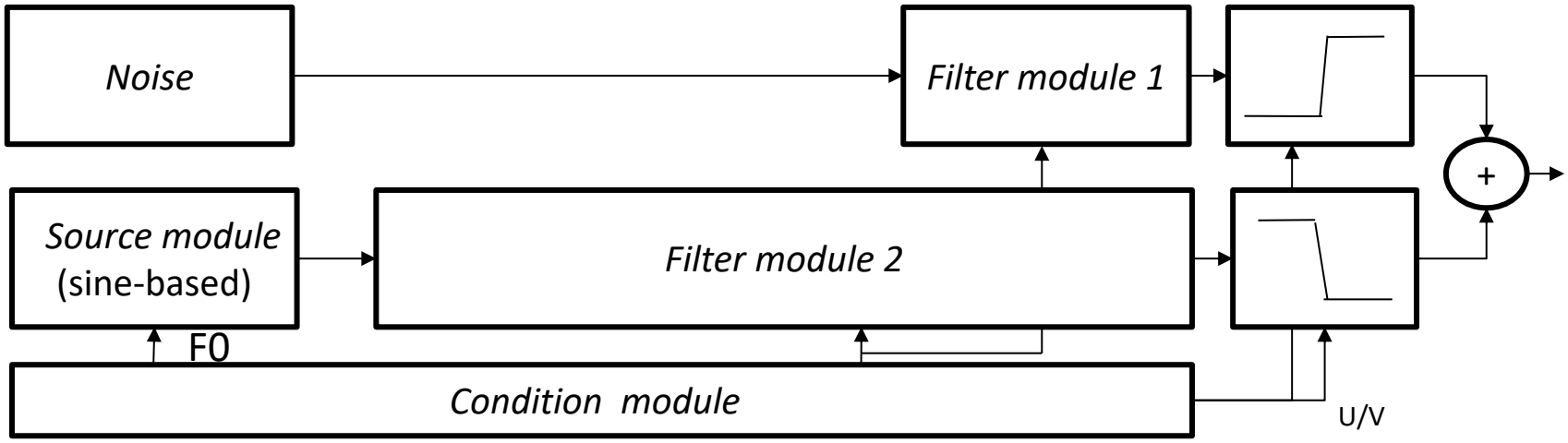
HPN NSF



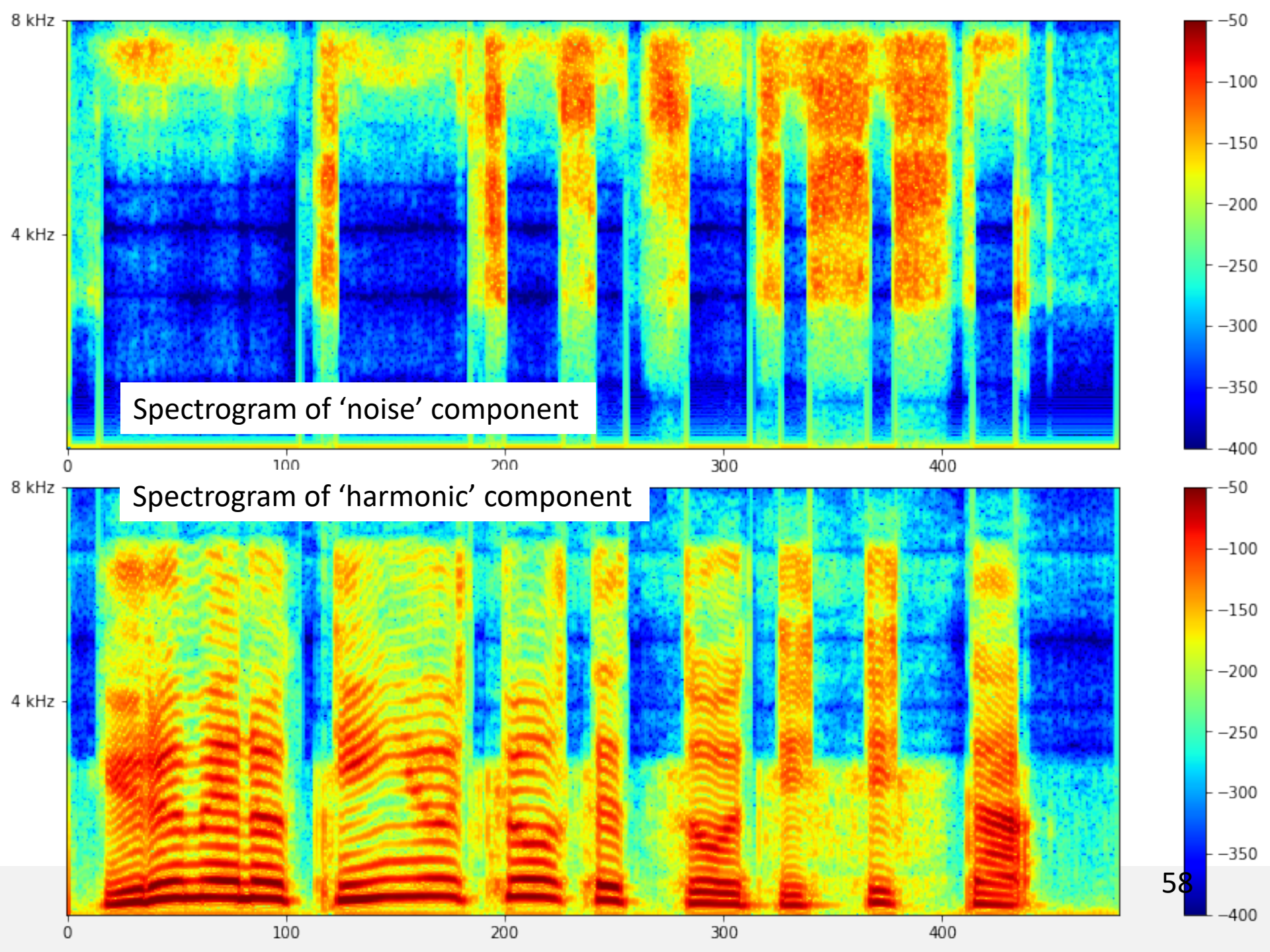
- Harmonic + noise style
- How to merge?
  - X Summation: may not separate harmonic from noise
  - X Merge in time domain: hard / soft switch
  - ✓ Merge in frequency domain: FIR filters

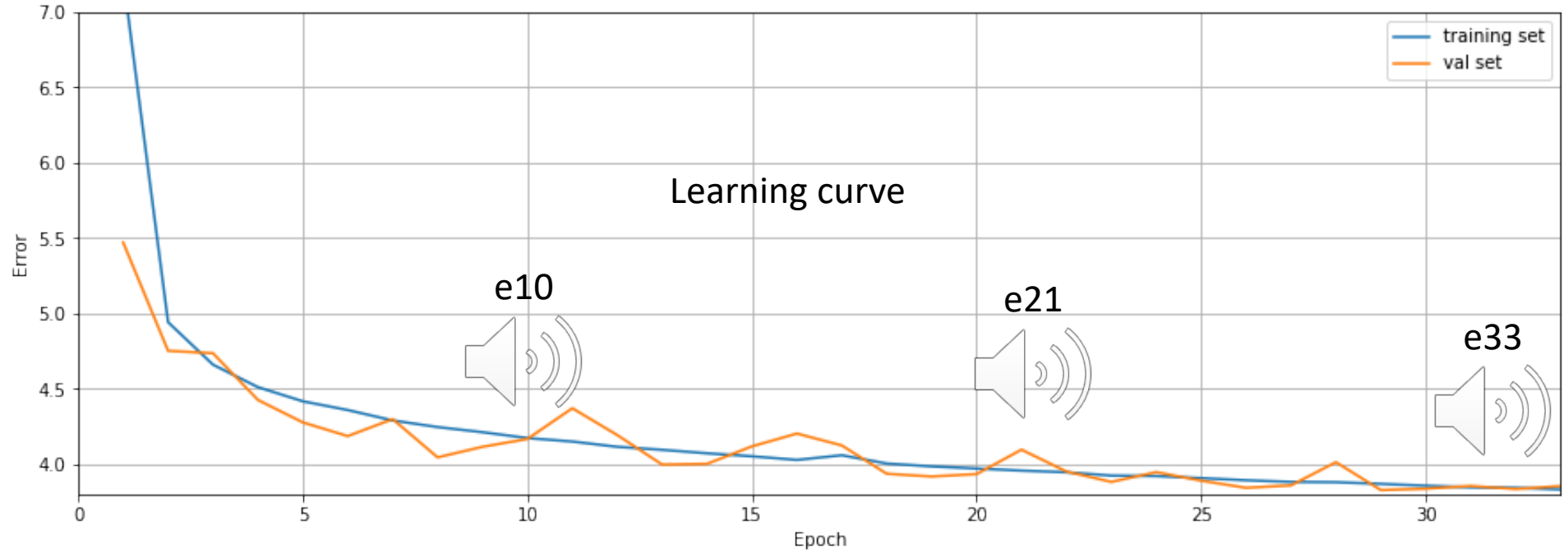
HARMONIC-PLUS-NOISE NSF

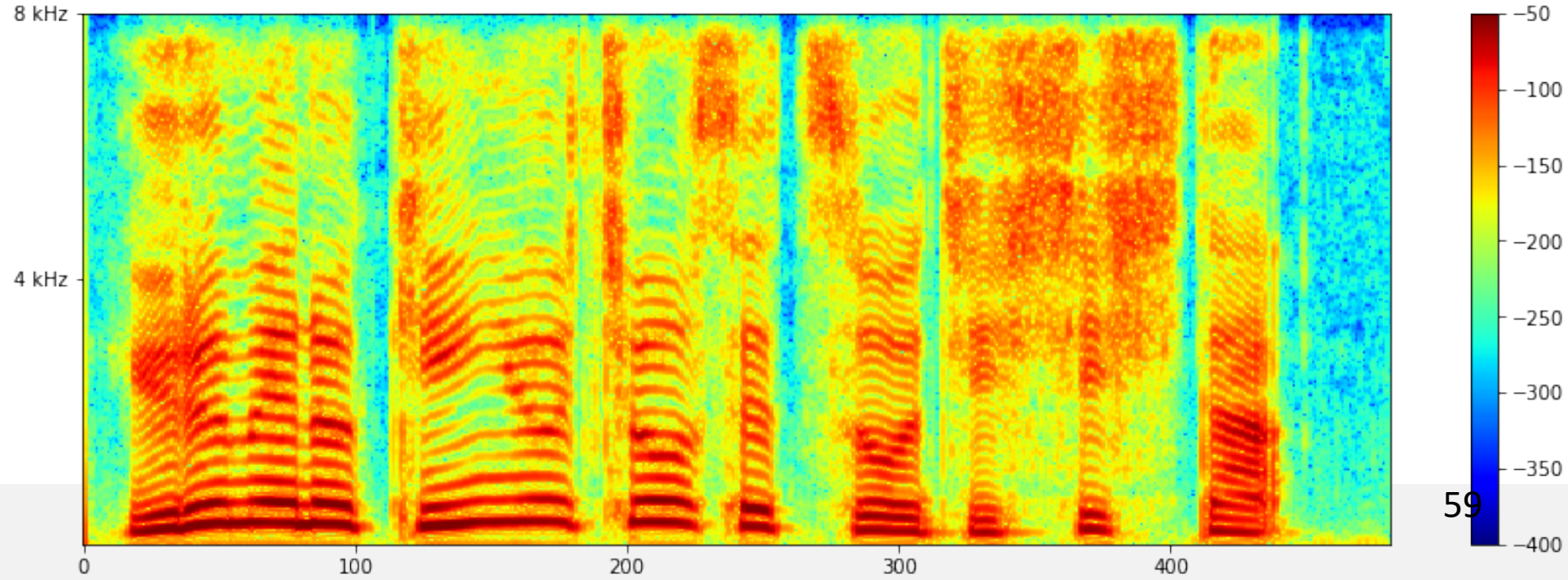
HPN NSF



- Harmonic + noise style
- How to merge?
  - ✓ Merge in frequency domain: FIR filters
    - Change cut-off frequency based on voicing condition


8 kHz
4 kHz
0
100
200
300
400
−50
−100
−150
−200
−250
−300
−350
−400
Spectrogram of ‘noise’ component
Spectrogram of ‘harmonic’ component


Learning curve
training set
val set
Error
Epoch
e10
e21
e33


8 kHz
4 kHz

SUMMARY

Samples (16kHz waveform, natural acoustic features)

Natural	Original NSF		Simplified NSF		Simplified & improved NSF	
	MGC + F0	Mel-spec + F0	MGC + F0	Mel-spec + F0	MGC + F0	Mel-spec + F0

- Natural MGC / Mel-spectrogram
- Natural F0

SUMMARY

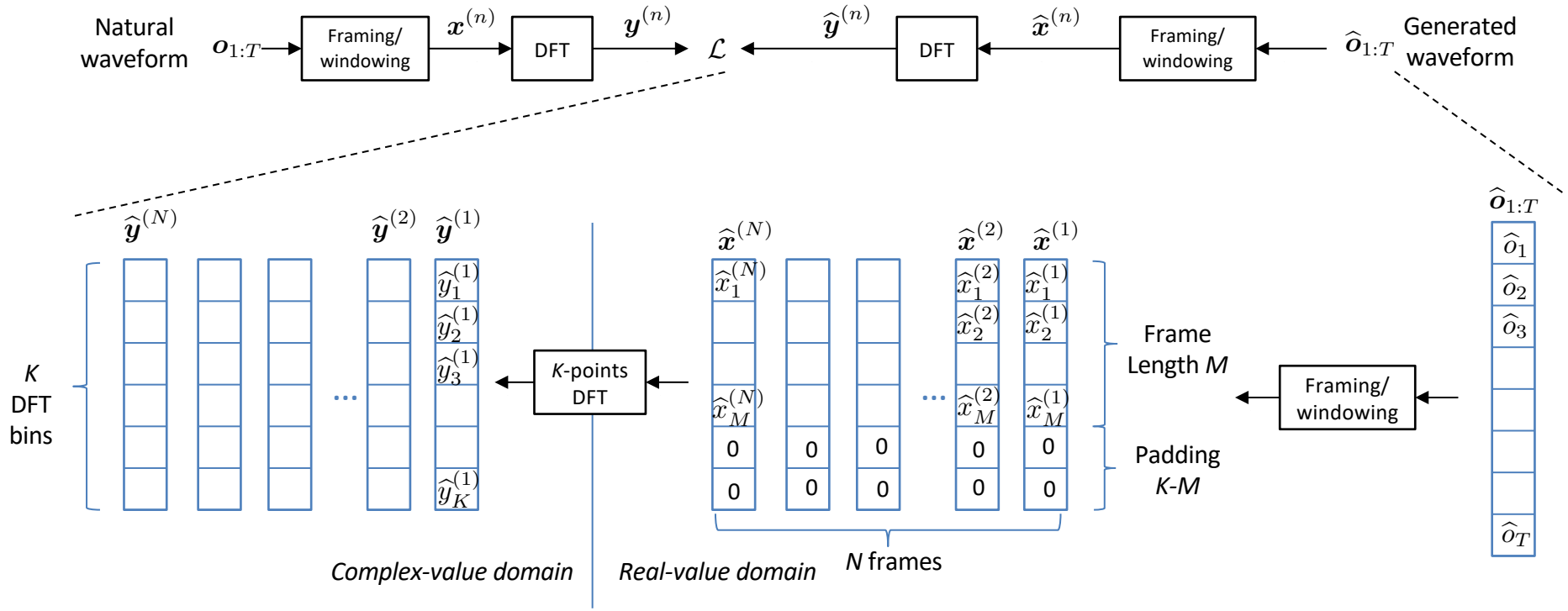
Samples (16kHz waveform, text-to-speech)

Natural	Original NSF		Simplified NSF		Simplified & improved NSF	
	MGC + F0	Mel-spec + F0	MGC + F0	Mel-spec + F0	MGC + F0	Mel-spec + F0

- Predicted MGC / Mel-spectrogram from text
- Natural F0

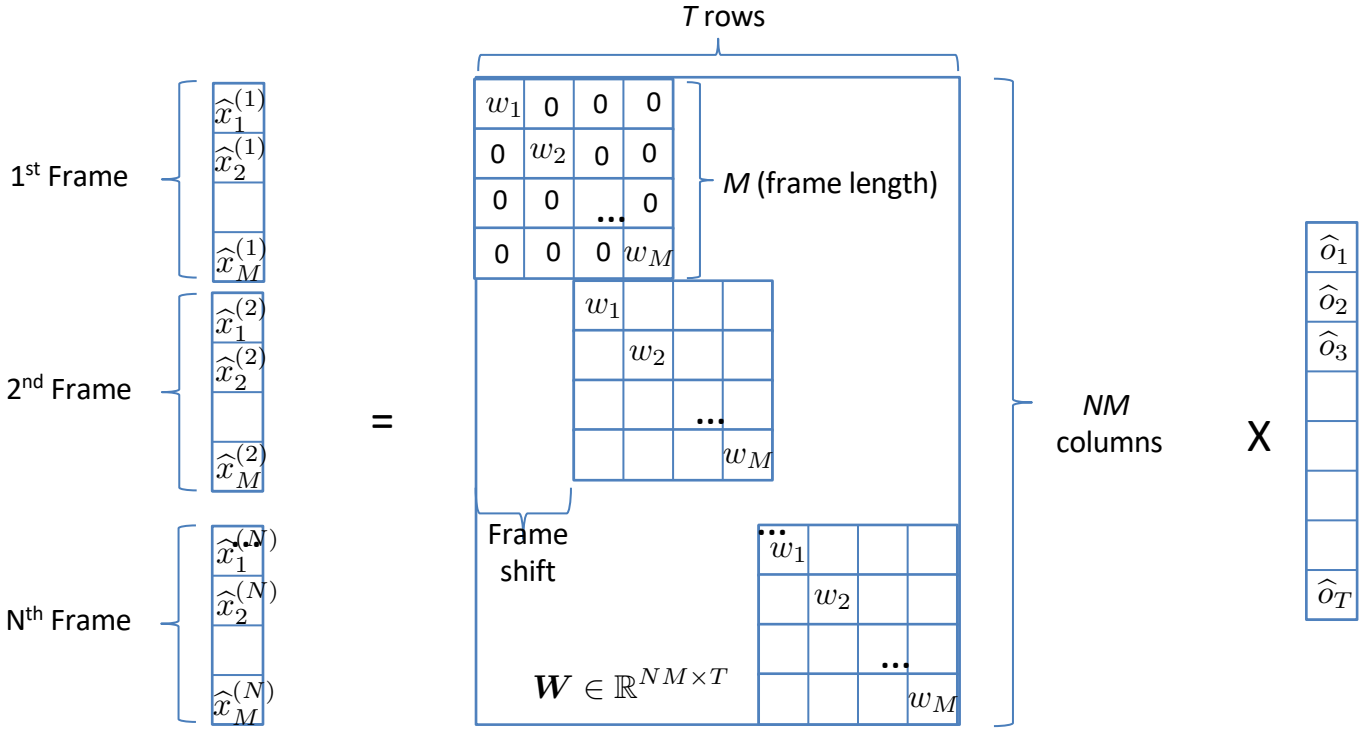
NEURAL SOURCE-FILTER MODEL

Training criterion



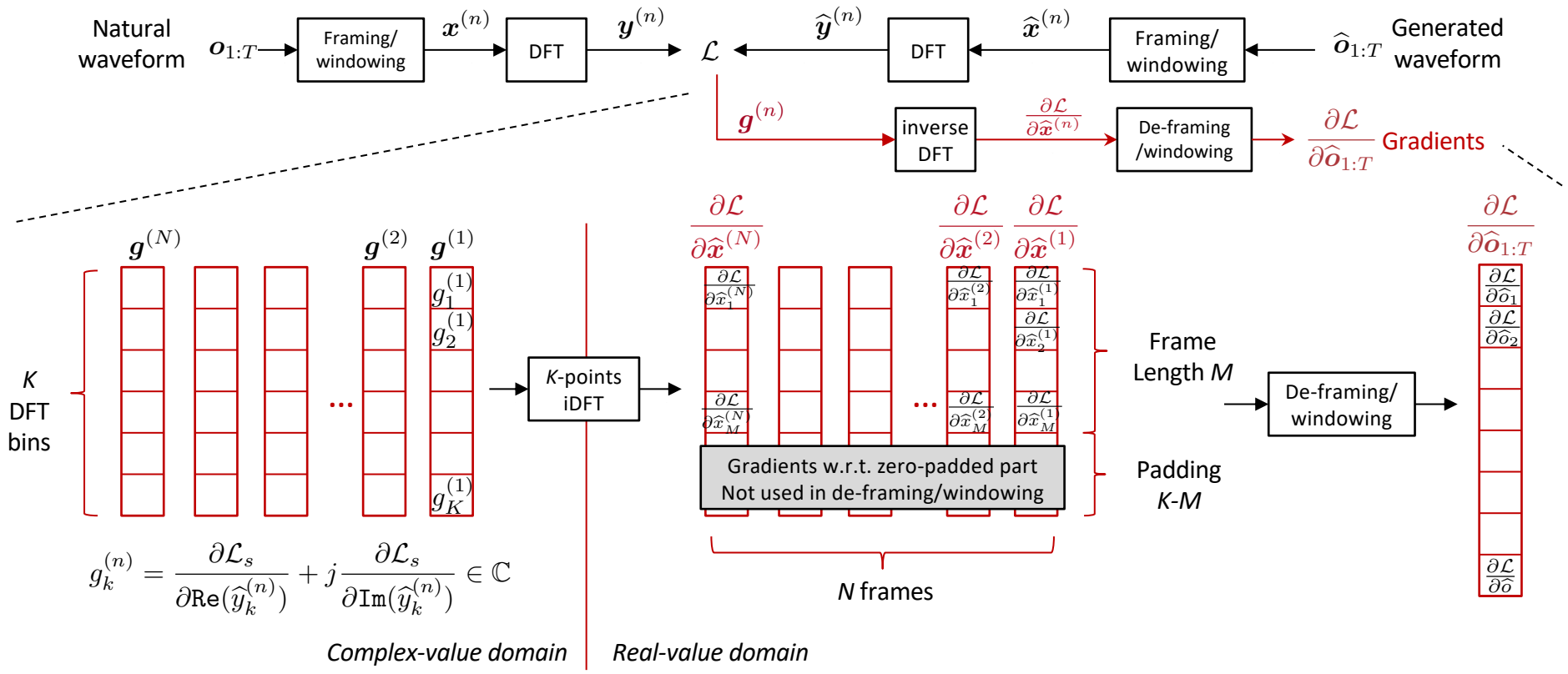
NEURAL SOURCE-FILTER MODEL

Training criterion



NEURAL SOURCE-FILTER MODEL

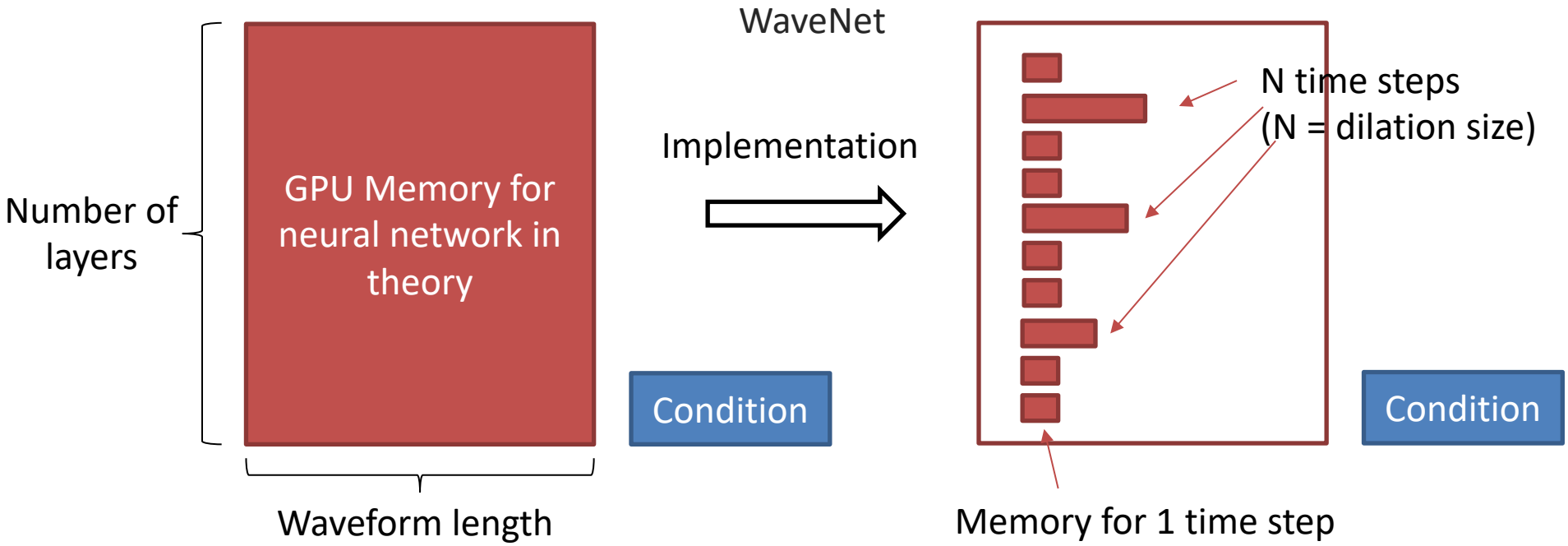
Training criterion



PART 3

Implementation issue

- WaveNet memory consumption in generation

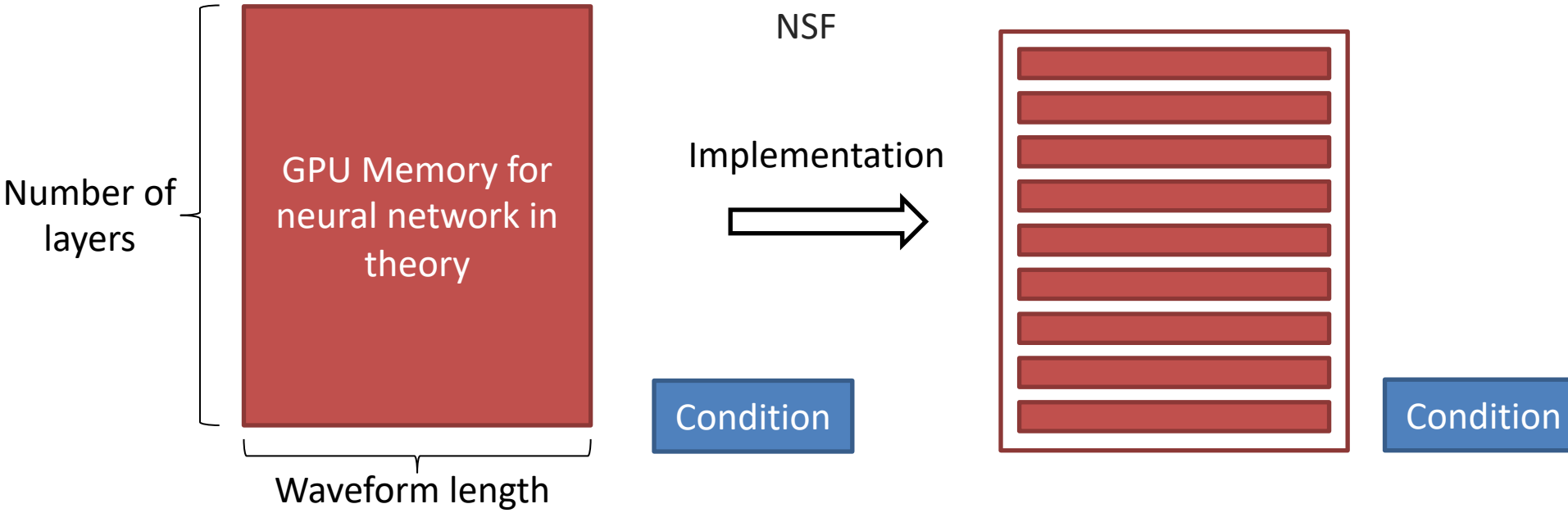


- Only allocate GPU memory for generating 1 waveform point
- For t = 1 : T, reuse the memory space
- Check: http://tonywangx.github.io/pdfs/CURRENNT_WAVENET.pdf (pp.44-55)

PART 3

Implementation issue

- NSF normal node

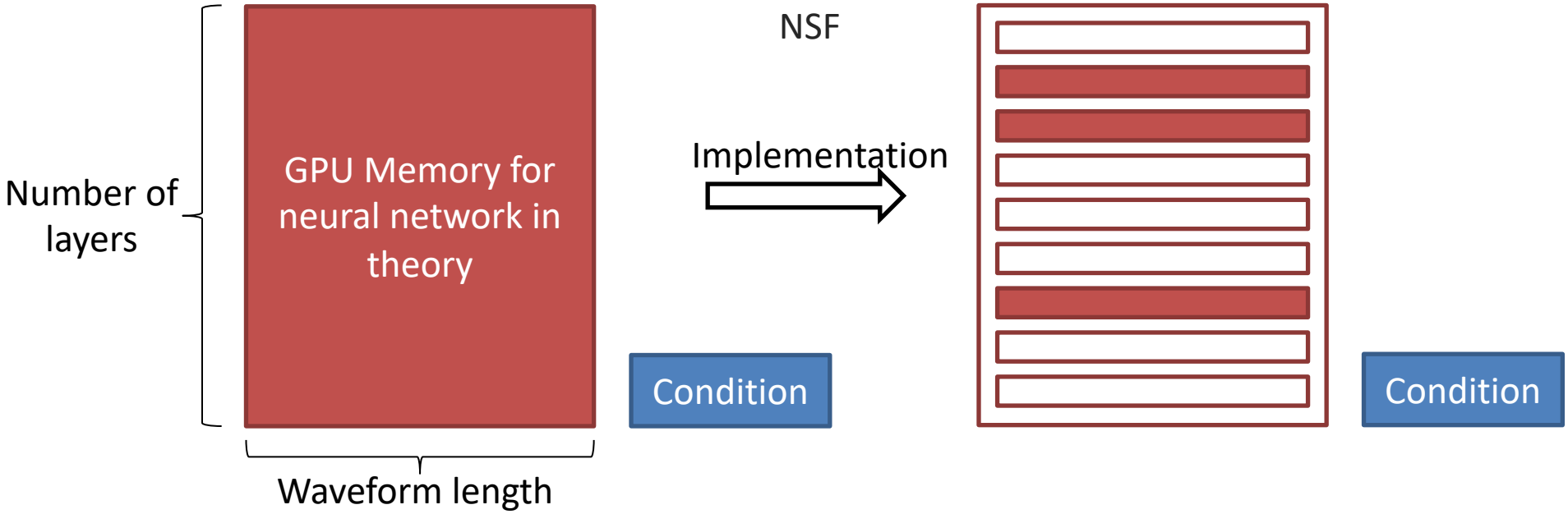


- Normal mode: allocate GPU memory for all layers and all time steps
- GPU memory consumption is large for long waveforms

PART 3

Implementation issue

- NSF memory-save node

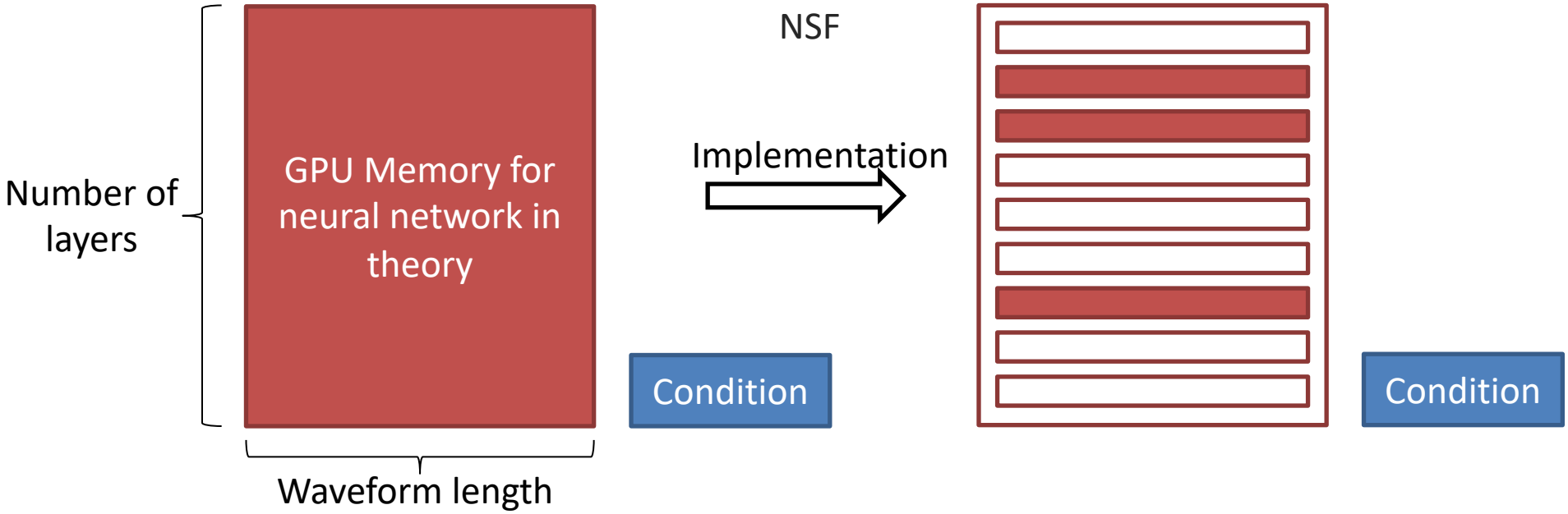


- Memory-save mode:
  - Allocate GPU memory for the current layer and all previous layers it depends on
  - Release GPU memory when one layer is no longer needed
- GPU memory consumption is small

PART 3

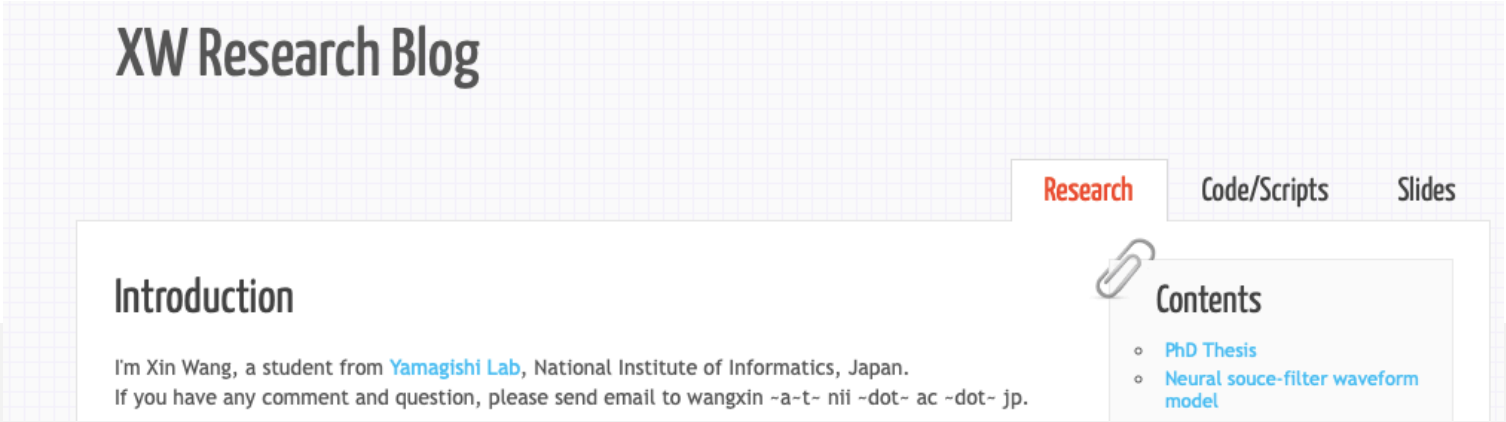
Implementation issue

- NSF memory-save node



Memory-save mode:
Function __computeGenPass_LayerByLayer_mem
  Build layer-dependency graph
  For layer_n = 1 : L
    Allocate GPU memory and compute layer_n's output
    For layer_m = 1 : layer_n where layer_m's output is input to layer_n
      If layer_m is no longer needed by any layer, release memory of layer_m

SELF-INTRODUCTION

- National Institute of Informatics
- Yamagishi-lab
- Postdoc

- PhD (2015-2018) SOKENDAI・NII
- M.A (2012-2015) USTC
- B.A (2008-2012) UESTC
- http://tonywangx.github.io