

# FFTNET: A REAL-TIME SPEAKER-DEPENDENT NEURAL VOCODER

## 1 Introduction

WaveNet、言語特徴から合成するアプローチ、ボコーダとして音響特徴から合成するアプローチ。

音響特徴から波形生成するボコーダアプローチの場合1時間程度の少ないコーパスで合成できる。

オリジナルのWaveNetは30時間訓練、40分で1秒のオーディオを生成。

Fast WaveNetは1分で1秒生成。

DeepVoiceアプローチはリアルタイムを達成したが品質を犠牲にした。

Parallel WaveNetはGPUクラスタを活用し大幅に速度を向上。

WaveNetはDilated Convでオーディオをダウンサンプルする過程がウェーブレット解析に似ている。

対してFFTNetアーキテクチャはFFTに似ている。

そして類似のWaveNetモデルより少ないパラメータで足りる。

FFTNetはFast WaveNetより70x速く生成できる。単一CPUでリアルタイム合成ができる。

さらにvocoderとして使うとより高品質な合成音声を生成することがMOSで確認されている。

## 2 Method

WaveNet同様直前に生成したサンプルと補助の条件(auxiliary conditions)から1度に1サンプルの波形を生成するが、よりシンプルである。

### 2.1 WaveNet Vocoder

WaveNetの合成音声は直前に生成されたサンプルおよび音素シーケンスやF0といった補助条件から生成される。

予測値はμ-lawでの量子化されたサンプルの事後確率分布をベースにしている。

補助条件(auxiliary conditions)がMCC(Mel Cepstral Coefficients)やピッチのような音響特徴であるとき、WaveNetはこれらの特徴から波形を生成するVocoderとして使える。それはどのように音声が生成されるかという前提なしに従来のボコーダよりはるかに自然でクリアな音声を生成する。

WaveNetの基本ブロックはcausal dilated convolutionである。

シグナル $f$ , dilation  $d$ , kernel  $k$ が与えられたとき、時刻 $t$ でのdilated convolutionは  
 $(k *_d f)_t = \text{Sum for } i (k_i * f(t - di))$ .  
で与えられる。

$k_i * f(t-di)$ をconv1x1で置き換えると1-D dilated convolutional neural networkまたはDCNNを得る。

WaveNetはdilation factorsを2の階乗で指数的に増加させた複数のDCNNのレイヤーで構成される。

それぞれの層ではgated activation structureが使われる。

$$z = \tanh(W_f *_d x + V_f *_d h) \odot \sigma(W_g *_d x + V_g *_d h)$$

説明:

where  $x$  is the output from the previous layer and serves as the input for the current layer;  $W_f$  and  $W_g$  are convolution kernels for the filters and the gates;  $V_f$  and  $V_g$  represent conv1x1 on the auxiliary conditions  $h$  ( $F_0$  and MCC); and  $\odot$  is element-wise dot product. The final output of a layer is obtained by another conv1x1 on  $z$ . The original WaveNet proposed to use skip connections – an additional conv1x1 is applied to  $z$  to obtain a new output  $s$ ; then  $s$  of all layers are summed together to become a skip output. To further increase nonlinearity, more conv1x1 and ReLU layers are applied on top of the skip output and finally a softmax layer is used to produce the posterior distribution of quantized sample values.

dilated convolutionではレイヤー $n$ は $2^n$ の長さの受容野をもつレイヤーであり、その上の $2^{n+1}$ のレイヤーの出力を受け取ることを意味する。これが高い合成品質を担保している。

だが16kHzなら16000回のcausal dilated networkが適用される。

## 2.2 FFTNet architecture

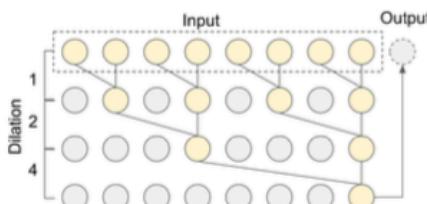


Fig. 1. Dilated convolution in WaveNet

filters and the gates;  $V_f$  and  $V_g$  represent conv1x1 on the auxiliary conditions  $h$  ( $F_0$  and MCC); and  $\odot$  is element-wise dot product. The final output of a layer is obtained by another conv1x1 on  $z$ . The original WaveNet proposed to use skip connections – an additional conv1x1 is applied to  $z$  to obtain a new output  $s$ ; then  $s$  of all layers are summed together to become a skip output. To further increase

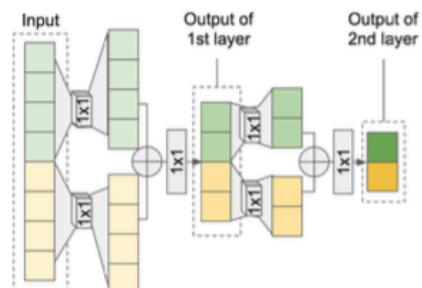


Fig. 2. FFTNet architecture: given size-8 inputs, they are first divided into two halves; each passed through a different 1x1 convolution layer and then summed together. The summed size-4 output will pass through ReLU and then another 1x1 convolution and ReLU before repeating the same operation.

新しいサンプルを予測する際のノードの関係から、逆バイナリツリー構造を見ることができる。(Figure1)

dilated convolution の構造はwavelet解析に似ている。各々のステップのフィルタリングがダウンサンプリングに従う点で。

Cooley-Tukey FFTをベースにしたウェーブレットの構造で代替できないか考えた。

入力シーケンス  $x_1, x_2, \dots, x_N$  が与えられると FFT は  $k$  番目の周波数  $f_k$  を時間領域シグナル  $x_0, \dots, x_{N-1}$  から

$$f_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i n k / N} = \sum_{n=0}^{N/2-1} x_{2n} e^{-2\pi i (2n) k / N} + \sum_{n=0}^{N/2-1} x_{2n+1} e^{-2\pi i (2n+1) k / N}$$

で計算する。ここで

$$\sum_{n=0}^{N-1} x_n e^{-2\pi i n k / N} \text{ as } f(n, N)$$

は

$$\begin{aligned} f(n, N) &= f(2n, N/2) + f(2n+1, N/2) \\ &= f(4n, N/4) + f(4n+1, N/4) \\ &\quad + f(4n+2, N/4) + f(4n+4, N/4) = \dots \end{aligned}$$

に単純化して書ける。

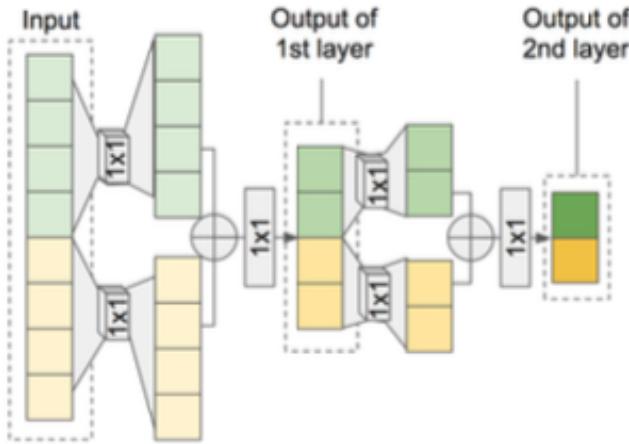
さらに、

$x_n$  は  $K$  個のチャンネルをもつ一つのノード、 $e^{-2\pi i n k / N}$  の項は変形関数と考えると、

$$f(n, N) = f(2n, N/2) + f(2n+1, N/2)$$

は前のノード  $x_{2n}, x_{2n+1}$  に変形を適用していると見ることができ、これらを合計している。

conv1x1 をその変形として用いるなら Figure 2 に見られるような FFT に似たネットワークをつくることができる。



**Fig. 2.** FFTNet architecture: given size-8 inputs, they are first divided into two halves; each passed through a different  $1 \times 1$  convolution layer and then summed together. The summed size-4 output will pass through ReLU and then another  $1 \times 1$  convolution and ReLU before repeating the same operation.

言い換えれば、入力  $x_{0:N}$  が  $(x_0, x_1, \dots, x_{N-1})$  として定義されるとき各FFTNetの層はその入力を半分にする。

$$(x_L = x_{0:N/2} \text{ and } x_R = x_{N/2:N})$$

conv1x1による変形を半分に分割された各々に適用しそれを合計する。

$$z = W_L * x_L + W_R * x_R \quad (1)$$

ここで  $W_L, W_R$  はそれぞれ  $x_L, x_R$  への conv1x1 の重みである。

ゲート付きの活性化構造を用いる代わりに、FFTNetは単純なReLUを用いる。

ReLUレイヤーは次のレイヤーへの入力を生成する1つのconv1x1に後続する。

つまり、

$$x = \text{ReLU}(\text{conv1x1}(\text{ReLU}(z)))$$

$n$  個のレイヤースタックは  $2^r$  の入力サイズとなる。

補助条件は conv1x1 により変形され  $z$  に加えられる。

$$z = (W_L * x_L + W_R * x_R) + (V_L * h_L + V_R * h_R) \quad (2)$$

ここで  $h_L, h_R$  は条件ベクトル  $h$  を半分にしたもの、

$V_L, V_R$  は conv1x1 の重みである。

もし条件となる情報が時間軸上で一定なら

$(V_L * h_L + V_R * h_R)$  でなく  $V * h_N$  となるだろう。

FFTNetをvocoderとして用いるのに、 $h_t$ を時刻tにおけるF0とMCC(Mel Cepstral Coefficients)特徴と定義する。

現在のサンプル  $x_t$  を生成する前に前に生成したサンプル  $x_{t-N:t}$  と補助条件  $h_{t-N+1:t+1}$  をネットワークへの入力として用いる

実験では補助条件は次のようにして得られる。

はじめに400のサイズの分析窓を160サンプルごとに取り、MCC(Mel Cepstral Coefficients)とF0を各重複する窓について抽出する。

$h_t$  が窓の中央に一致するとき計算済みのMCCとF0値を割り振った。(合計26次元)

$h_t$  が窓の中央に一致しないとき最後のステップで割り振られた  $h_t$  をベースにそれらの値を線形に挿入した。

FFTNetは最後に全結合層、Softmax層(サイズ1,K=256チャンネル)をもつ。

この最後の2層が新しいサンプルの量子化された値の事後確率分布を生成する。

現在のサンプルの最終値を決定するのにargmaxまたは事後確率分布からのランダムサンプリングを実行することもできるが、Section2.3.2に示す代わりの方法を提案する。

## 2.3 Training Techniques

WaveNetと同様の品質の結果を生成するための訓練テクニックを紹介する。

### 2.3.1 Zero padding

WaveNetはゼロパディングをdilated convの間もちいるがFFTNetも同様のアイデアである。

長さMの入力シーケンス  $x_{1:M}$  ( $M > N$ ) が与えられたとき、Nサンプル分ゼロ埋めされて右にシフトさせる。

N個の埋め込まれたゼロは  $x_{-N:0}$  で表す。  $\forall j < 0, x_j = 0$ .

[0,0,...{N個の0}..,M1,M2,...{長さMの入力シーケンス}]

方程式1

$$z = W_L * x_L + W_R * x_R \quad (1)$$

は

$$z_{0:M} = W_L * x_{-N:M-N} + W_R * x_{0:M}$$

のようになる。

実験ではゼロパディングなしの時、入力がほとんど無音だとネットワークはノイズを生成するかゼロを出力する傾向にある。

訓練中のゼロパディングを行うことでネットワークが部分的な入力に一般化できるようになる。

2Nから3Nの間の長さの訓練シーケンスを勧める。

それにより33%~50%ほどのかなりの数の訓練サンプルがpartial sequence(部分シーケンス?)になる。

### 2.3.2 Conditional sampling

WaveNetもFFTNetもclassification networkである。

最後のsoftmax層は256カテゴリーの事後確率分布を生成する。

全ての分類器同様に予測エラーは2つの原因に由来する。

訓練エラーと本物のエラー。

本物のエラーは主にノイズに一致し、シグナルの発話されていない部分にある。

ノイズを合成するのに出力事後確率分布からノイズの分類を学習するネットワークを用いた。

そこではランダムサンプリングでサンプルの値を抽出した。

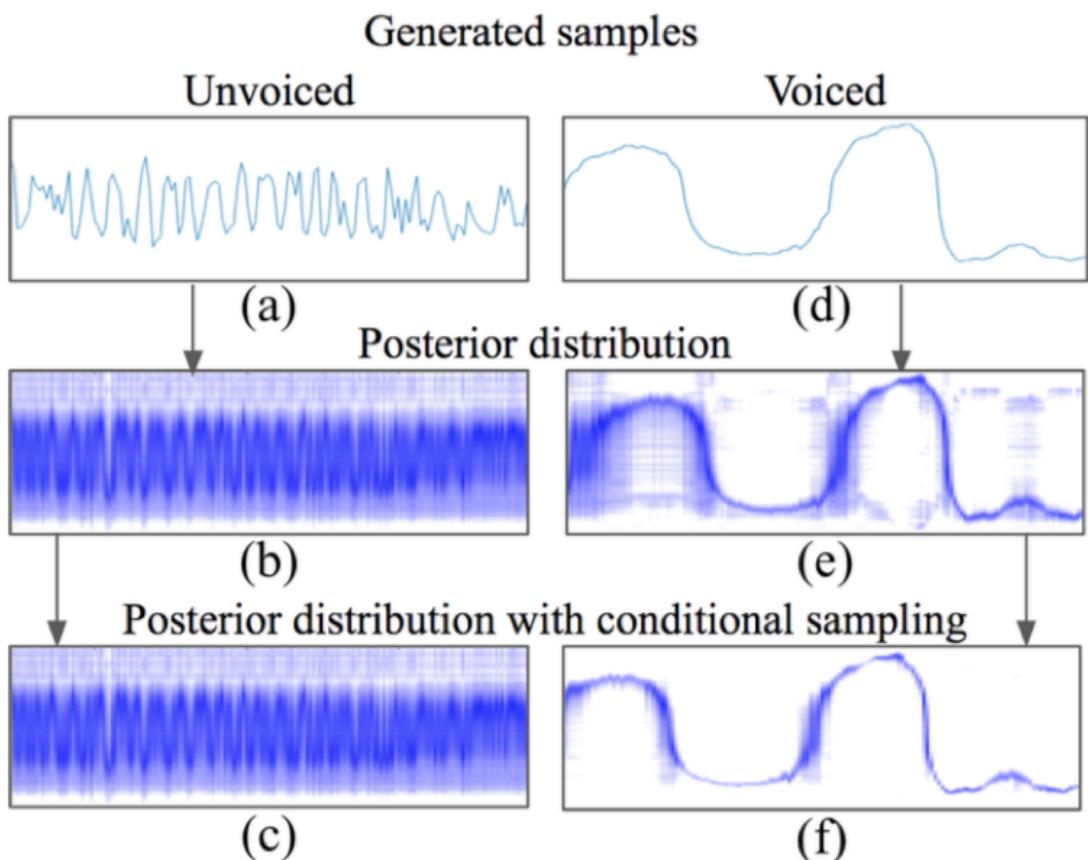
訓練エラーはモデルそのものに依拠する。

そして最小訓練エラーを与える予測戦略はargmaxである。

しかしargmaxは本物のノイズを含むシグナルをシミュレートするのに適していない。

なぜならそれは合成が出力するノイズ分布の中央値をいつも選択しぜロノイズにしてしまうからだ。

argmaxの代わりに無音区間と発話区間で異なる戦略を用いた。(Fig 3)



**Fig. 3.** Conditional sampling: WaveNet and FFTNet model noise by matching the posterior distribution to the noise's distribution (b). Therefore to generate noise (a), we randomly sample from the posterior distribution (b). For periodic signals (d), we double the log of the posterior distribution (e), to obtain a cleaner distribution (f); for aperiodic signals, the posterior distribution remains the same (c).

ノイズの分布に事後確率分布(b)がマッチさせることでノイズをモデル化している。

ノイズ(a)を生成するために、事後確率分布(b)からランダムに抽出する。

周期的シグナル(d)についてよりクリーンな分布(f)を得るために事後確率分布(e)のログを2倍にする。

非周期的シグナルについては事後確率分布をそのままにする(c)

無音区間の音については事後確率分布からランダムに抽出し、

発話区間の音については(softmaxへの入力値に)正規化ログを取り定数 $c > 1$ (この実験では $c=2$ )をかけ、

softmax層に通し事後確率分布を得る。そこでランダムサンプリングが実行される。

このやり方で元のノイズの分布は維持されながら事後分布はより鋭く明瞭になる。

### 2.3.3. Injected noise

Due to the training error, the synthesized samples always contain some amount of noise; during synthesis, the network will generate samples that get noisier over time. They serve as network input to generate the next sample, adding more and more randomness to the network. When the noise builds up, the output sample might drift leading to clicking artifacts. To avoid such drift, the network needs to be robust to noisy input samples. We achieve this by injecting random noise to the input  $x_{0:M}$  during training. We determine the amount of noise to inject into the input based on the amount of noise the network is likely to produce. In this work, we observe that the prediction is often one category (out of 256) higher or lower than the ground-truth category and thus, we inject Gaussian noise centered at 0 with a standard deviation of 1/256 (based on 8 bit quantization).

訓練エラーのために合成されたサンプルは常にいくらかのノイズを含んでいる。合成の間、ネットワークは時間を通してよりノイジーになる。

それらはネットワーク入力として次のサンプルを生成するのに渡される。そうするとだんだんランダムさを増す。

ノイズが構成されるとき、出力サンプルは変動しclicking artifacts(遺物)になる。そのような変動を避けるためにネットワークはノイジーな入力サンプルに対してロバストである必要がある。

これを訓練中にランダムノイズを入力 $X_{0:M}$ に注入することで達成した。入力に注入するノイズの量をネットワークが生成しそうなノイズの量をベースに決定する。

256中1カテゴリだけ元データのカテゴリよりも高いか低い予測をする傾向にあるため、平均0標準偏差1/256(8bit量子化にもとづく)のガウシアンノイズを注入した。

### 2.3.4. Post-synthesis denoising

Our experiments show that the injected noise eliminates the clicking artifact almost perfectly for FFTNet but introduces a small amount of random noise to voiced samples. Therefore, we apply a spectral subtraction noise reduction [19] to reduce the injected noise for the voiced samples. The reduction is proportional to the amount of noise injected during training. It is possible to apply noise reduction to the unvoiced samples as well, but it may result in artifacts. Therefore, we reduce the amount of noise reduction applied to the unvoiced samples by half.

ノイズの注入がほとんど完璧にFFTNetのclicking artifactを排除できることが実験で示された。だが音声サンプルに少量のランダムノイズを引き起こした。

それゆえspectral subtraction noise reductionを適用しボイスサンプルに入り込むノイズを軽減させた。訓練中に入り込むノイズに相当するリダクションとなった。

無音区間へのノイズ軽減も同様に可能だがartifactsな結果になる。そのため無音区間には半分だけ適用した。

## 3.1. Experimental setup

CMU Arcticデータセットで実験。

最初の1132発話を訓練、残りを評価。

入力波形は $\mu$ -lawをベースに256カテゴリカル値に量子化、 25-coefficient MCC(with energy)。

F0はオリジナルサンプルから抽出。

2つのWaveNets, 2つのFFTNetsを構築。

Section2.3の全てのテクニックを用いる場合とzero paddingのみの場合で実験。

FFTNetは**256チャンネルの11FFT-layers、 receptive fieldを2048**。

このFFTNetは[13]のキャッシングで1M以下のパラメータになっている。

[13] Tom Le Paine,.. “Fast wavenet generation algorithm,” *arXiv preprint arXiv:1611.09482*, 2016.

16kHz、1秒のオーディオを生成するのに16GFLOPsの計算量。CPUでリアルタイム生成可能。

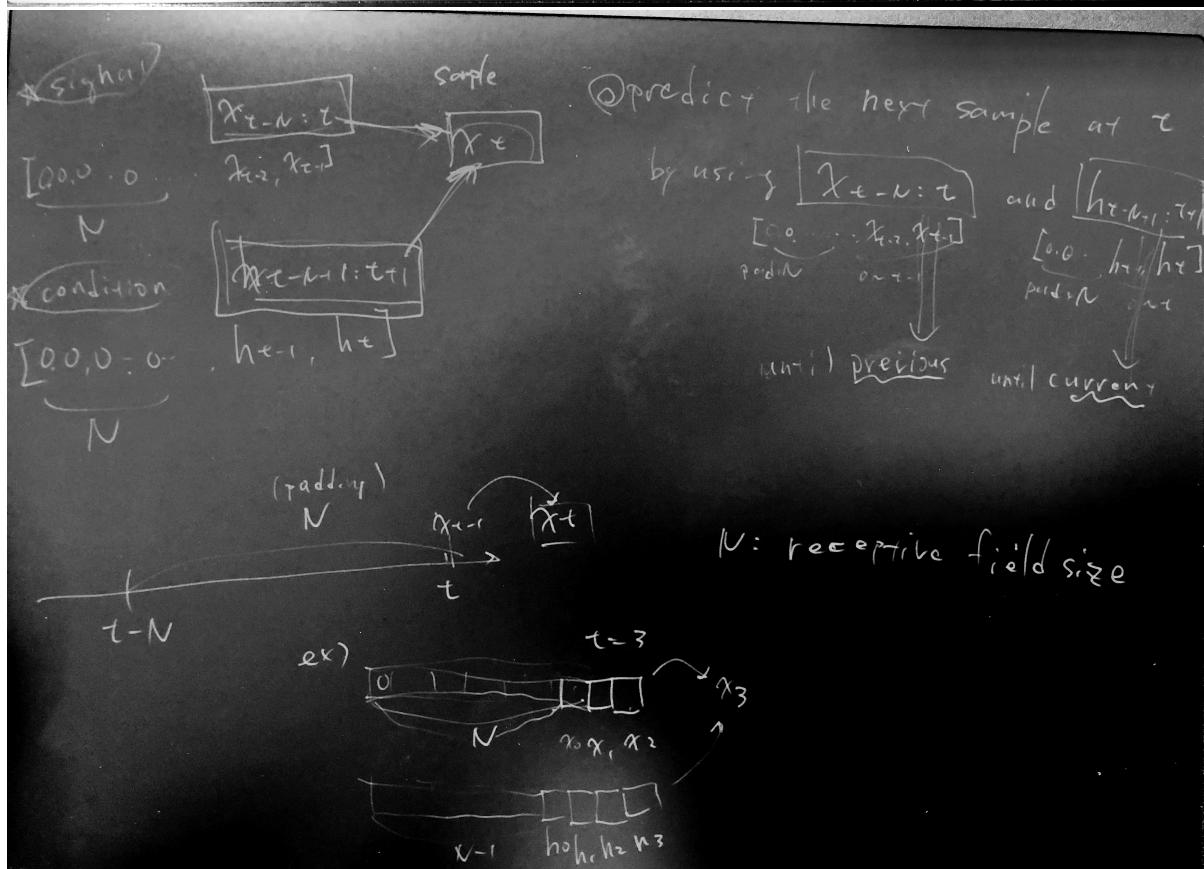
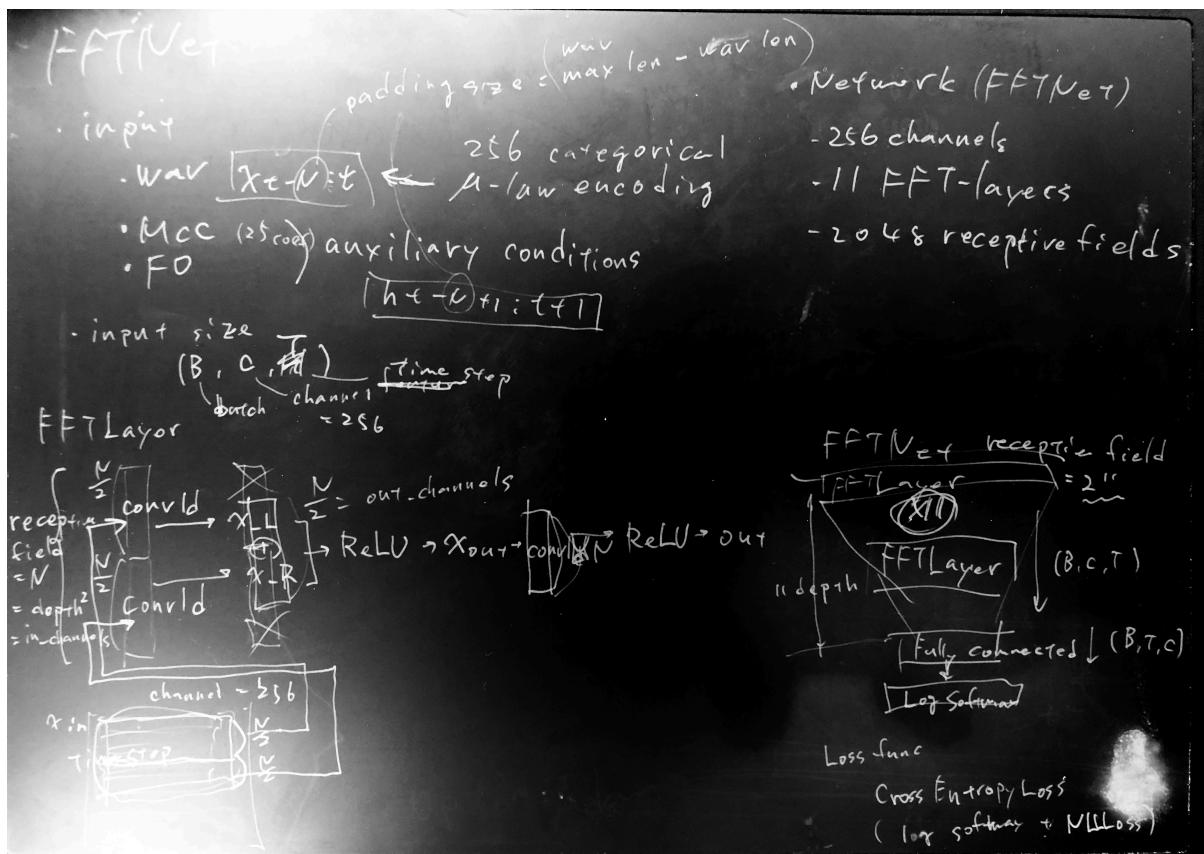
**5 x 5000サンプルシーケンスのミニバッチを各訓練ステップで入力。**

最適化アルゴリズムにAdam。

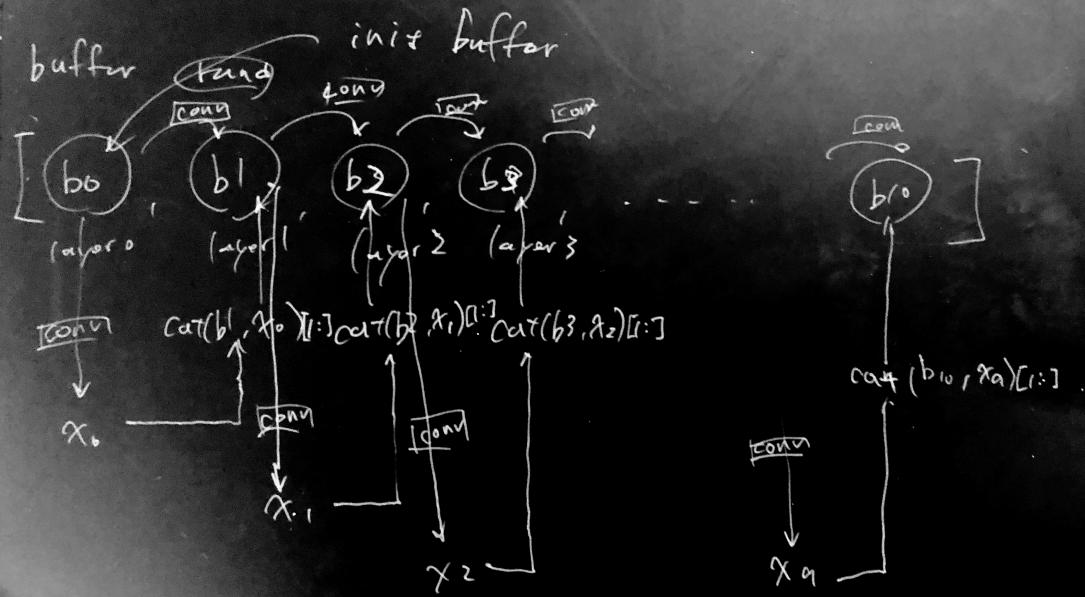
学習率は0.001。

注入したノイズの分散は1/256。

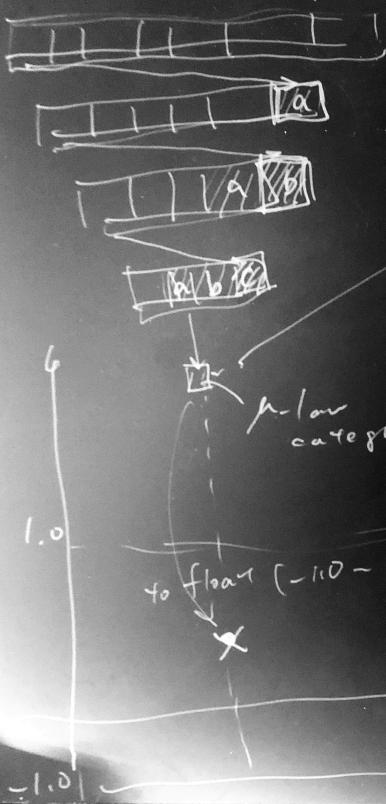
FFTNetは**10,000ステップ**で収束。



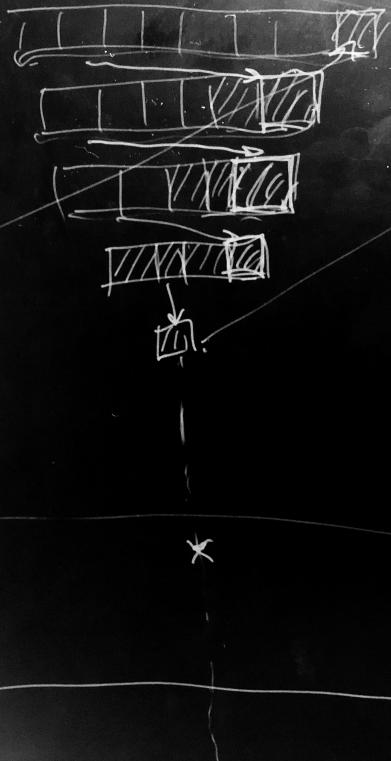
FFTNet gen



sample 1

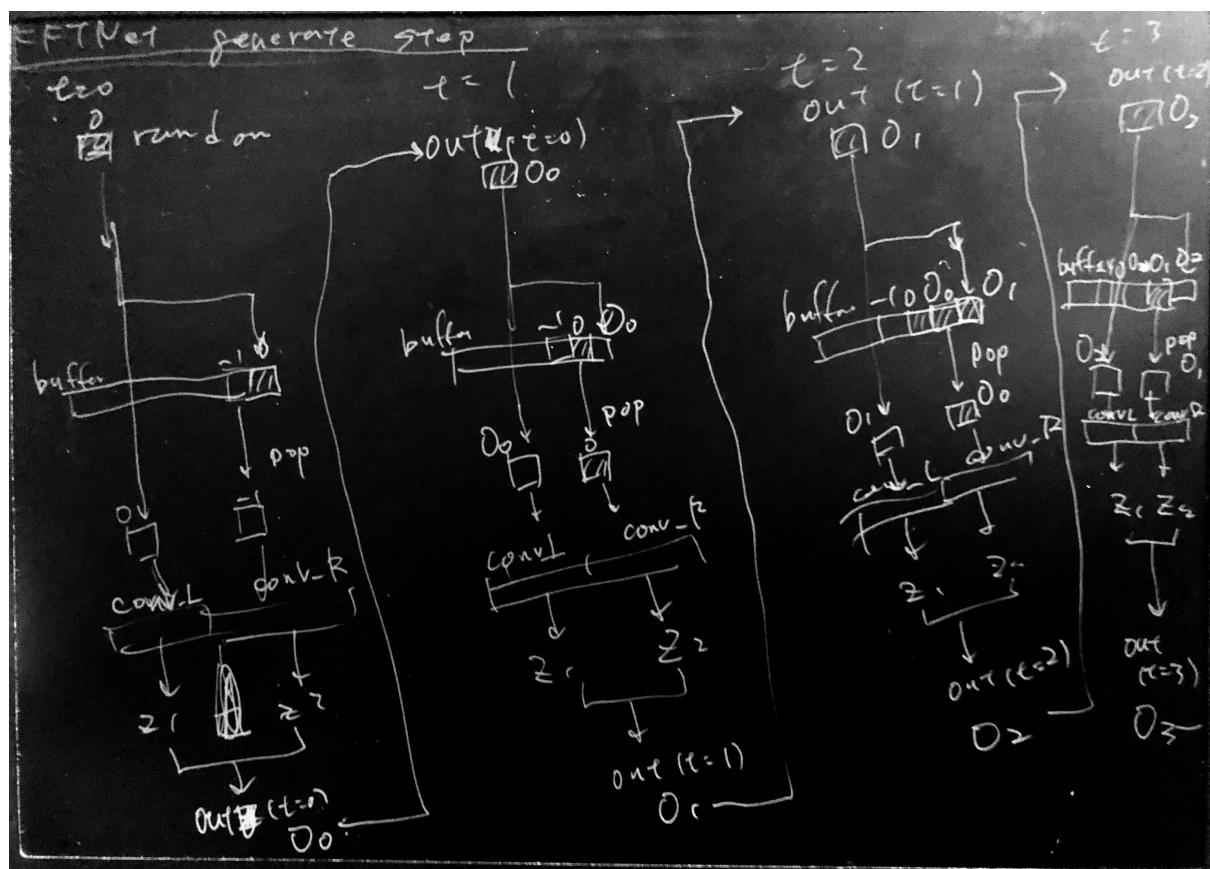


sample 2



sample 3





PDF File