Feasibility Study of Deep Frequency Modulation Synthesis

Keiji Hirata[1⋆], Masatoshi Hamanaka[2], and Satoshi Tojo[3]

[1] Future University Hakodate hirata@fun.ac.jp
[2] Riken AIP masatoshi.hamanaka@riken.jp
[3] JAIST tojo@jaist.ac.jp

Abstract. Deep Frequency Modulation (FM) synthesis is the method of generating approximate or new waveforms by the network inspired by the conventional FM synthesis. The features of the method include that the activation functions of the network are all vibrating ones with distinct parameters and every activation function (oscillator unit) shares an identical time t. The network learns a training waveform given in the temporal interval designated by time t and generates an approximating waveform in the interval. As the first step of the feasibility study, we examine the basic performances and potential of the deep FM synthesis in small-sized experiments. We have confirmed that the optimization techniques developed for the conventional neural networks is applicable to the deep FM synthesis in small-sized experiments.

Keywords: Frequency modulation synthesis · neural networks · activation function · backpropagation.

1 Introduction

Frequency Modulation (FM) synthesis is a well-known technique for generating musical sound [1] and has been employed for many commercial products of digital synthesizers such as DX7 of Yamaha, which is one of the bestselling synthesizers [7]. Also many variations of the FM synthesis have been developed [6, pp.224-250]). FM synthesis can generate rich sounds despite a simple configuration, i.e., the small number of parameters; on the other hand, it is known that FM synthesis requires some skills for manipulating parameters when generating new desired sounds. It is mainly because the relationships among output sounds, parameter values, and configurations of connecting oscillators are not sufficiently intuitive. Hence, to create new sounds easily, many of digital synthesizers employing the FM synthesis provides the presets which are built-in connection patterns of oscillators with predefined parameters of amplitudes and carrier and modulating frequencies.

After considering these presets provided, we would come up with an idea of a general form of the FM synthesis which looks like a neural network, the activation functions of which are oscillators. Suppose, in the network, an oscillator

⋆ This work has been supported by JSPS Kakenhi 16H01744.

X at a layer can receive modulating waveforms from the ones at one-level lower layer $Y_1, Y_2, \cdots$. The weight of the connection between X with Y_k corresponds to amplitude parameters. The carrier and modulating frequencies are defined as parameters within each oscillator. Then, all the oscillators should refer to an identical current time and simultaneously generate waveforms along with the time as in the conventional FM synthesis. If we would apply the learning techniques developed for conventional neural networks to the network inspired by the conventional FM synthesis, we might generate target sounds without taking care of the relationships among output sounds, parameter values, and configurations of connecting oscillators.

Gashler and Ashmore [2] have surveyed various networks to model and predict time-series data and offered a useful idea for categorization of approaches. Gashler and Ashmore claim that at a high level, the neural networks to predict time-series data are broadly categorized into three major approaches: here we refer to them as WaveNet approach, RNN approach, and extrapolation approach. Among them, there have been proposed several neural networks that belong to the extrapolation approach and employ vibrating activation functions such as a sinusoidal function and wavelet [2, 5, 3]. In these architectures, the current time for generating a waveform is treated as the explicit input given at an input layer. That is, neither all the activation functions (i.e., oscillators) share the current time, nor the activation functions at the hidden layers refer to it.

In the paper, we propose deep Frequency Modulation (FM) synthesis, which is the method of generating an approximating waveform based on the network inspired by the conventional FM synthesis. According to the Gashler and Ashmore's categorization, the deep FM synthesis basically belongs to the extrapolation approach. Thus, we hope the deep FM sysntesis could generate unknown yet good sounds by extrapolating already existing sounds in a different way from prevalent sound generation methods such as sampling and WaveNet. To study the feasibility and utility of the deep FM synthesis, we investigate the basic characteristics and performances of it; for instance, how accurate the deep FM synthesis can approximate a target waveform, what size of the network we need for reconstructing a target waveform, how and what conventional techniques for optimizing networks can be applied to the deep FM synthesis, and so on.

2 Deep Frequency Modulation Synthesis

2.1 Architecture

For theoretical consideration, we think of a simple, typical architecture shown in Figure 1, which presents how oscillator units are interconnected with weights w; the depth is $M+1$, the width is $N+1$, and Σ^w stands for weighted sum. The input to the network is vector $\{A_0, A_1, \cdots A_N\}$, and the output is waveform S. For an oscillator unit by a typical vibrating function, we here adopt a sinusoidal function $y = \sin\ 2\pi((x+c)\ t+p)$ with input x (the bottom of the oscillator unit in the figure) and output y (the top). Each oscillator unit has two parameters c and p to be tuned corresponding to frequency and phase, respectively. All

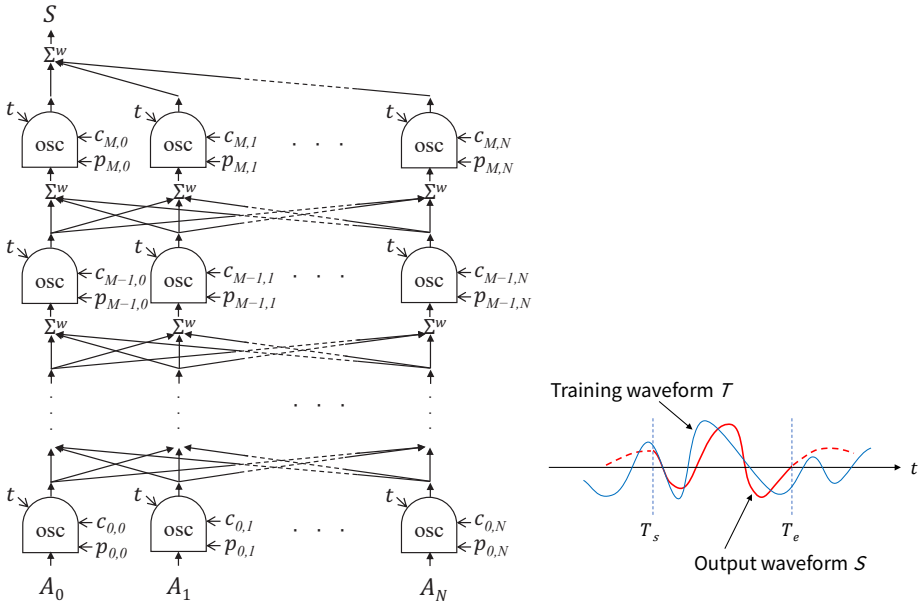

Fig. 1. Network Configuration of Deep FM Synthesis

oscillator units share the identical timing signal t, which moves between starting time T_s and ending time T_e, to compute the output waveform (the red solid curve in Figure 1). The network attempts to fit the output waveform to the training waveform only between T_s and T_e (dark blue). Thus, in the ranges out of the interval between T_s and T_e, the network does not take care of the output waveform (red dashed curves).

The forward propagation in the deep FM synthesis works as conventional neural networks; at layer n, input waveform x is given to each oscillator unit to compute output y with parameters $c_{m,n}$ and $p_{m,n}$ at time t. Then, the output waveforms calculated at layer n, that is y, are summed up with weights, and the sum is provided to the input waveform at layer $n+1$ as x. An output waveform is made of the series of values S_t, which represent the samples at time instant t between T_s and T_e. In other words, given time t at which we want to obtain value S_t, the network calculates S_t in a bottom-up manner.

We may assume that all the elements of the input vector $\{A_0, A_1, \cdots A_N\}$ are constant values. The assumption is justified by the design decision we made that the deep FM synthesis works as a multiple-wave generator depending on the input vector. In reality, the input vector can be made of either constant values or any waveforms synchronized by the timing signal t. Theoretically, the input vector can be either constants or any waveforms synchronized by the timing signal t, since giving waveforms to the input is equivalent to the extension of the network in the direction of depth with constants given to the input. Therefore,

the multi-layered architecture absorbed such subtle differences, and we can put the assumption without loss of generality.

2.2 Backpropagation

We would apply the standard backpropagation technique to optimize the deep FM synthesis as follows [4]. For notational simplification, we assume the network size is depth $M+1$ by width $N+1$, and the width is unchanged from the top layer to the bottom. The final output of the deep FM synthesis at time t is denoted as S_t. The input and output of the n-th oscillator unit at layer m are denoted as $x_{m,n}$ and $y_{m,n}$, respectively. The weight between adjacent layers is denoted as $w_{m,n',n}$, which stands for the weight from n'-th unit at layer m to n-th unit at layer $m+1$. Only at the topmost layer, we write $w_{M,n}$, omitting the second n. Then, the final output is straightforwardly defined in a topdown manner:

$$S_t = \sum_{n=0}^{N} w_{M,n} \cdot y_{M,n} \tag{1}$$

For m = M .. 1 and n = 0 .. N, we define $y_{m,n} = \sin 2\pi((x_{m,n} + c_{m,n})t + p_{m,n})$ and $x_{m,n} = \sum_{n'=0}^{N} w_{m-1,n',n} \cdot y_{m-1,n'}$. For m = 0 and n = 0 .. N (the bottom layer), we define $y_{0,n} = \sin(x_{0,n}t + p_{0,n})$, and $x_{0,n} = A_n$. We always put $c_{0,n} = 0$ because the elements of the input $\{A_0, A_1, \cdots A_N\}$ are all assigned to constant values.

A single network of deep FM synthesis is trained, considering the set of time instants within the designated period between T_s and T_e. Let us denote the training waveform (target waveform) at time t as T_t. Then, the loss function is defined as follows:

$$E = \sum_t E_t = \sum_t \frac{1}{2}(S_t - T_t)^2 \ , \tag{2}$$

where $\sum_t$ means the summation over the set of the time instants ($T_s \leq t \leq T_e$).

We present the gradient descent method for optimizing the network [4]; let us compute the partial differential of the loss in Equation (2) with respect to each parameter contained in the network in the standard manner. First of all, for the topmost weights $w_{M,n}$, from Equation (1) we have

$$\frac{\partial E_t}{\partial w_{M,n}} = \frac{\partial E_t}{\partial S_t} \cdot \frac{\partial S_t}{\partial w_{M,n}} = (S_t - T_t) \cdot y_{M,n}$$

Note that although the same timing signal t is provided to all oscillator units, the gradient of the loss can be derived as in conventional neural networks. Also for the parameters within each oscillator, $\frac{\partial E_t}{\partial c_{M,n}}$ and $\frac{\partial E_t}{\partial p_{M,n}}$ can be derived similarly. Then, to simply express the derivatives of $y_{m,n}$, we introduce term $\cos 2\pi((x_{m,n} + c_{m,n})t + p_{m,n})$ and denote it as $z_{m,n}$. Due to space limitation, we omit the technical details and show only the result of parameter optimization. We derive the following entire inductive definition of the gradient chain Γ and the feedback values for gradient descent:

Base step: $\Gamma^A_{M,n} = (S_t - T_t) \cdot w_{M,n}$
Induction step:

$$\Gamma^A_{m,n} = \sum_{i=0}^{N} \Gamma^B_{m+1,i} \cdot \ w_{m,n,i} \quad \text{(m = 0 .. M-1)}$$
$$\Gamma^B_{m,n} = \Gamma^A_{m,n} \cdot 2\pi t \ z_{m,n} \quad \text{(m = 0 .. M)} \tag{3}$$

Using the series of the gradient chain above, we obtain the partial differentials of the parameters as follows:

$$\frac{\partial E_t}{\partial w_{M,n}} = (S_t - T_t) \cdot y_{M,n}$$
$$\frac{\partial E_t}{\partial w_{m,n',n}} = \Gamma^B_{m+1,n} \cdot y_{m,n'} \quad \text{(m = 0 .. M-1)}$$

$$\frac{\partial E_t}{\partial c_{m,n}} = \Gamma^A_{m,n} \cdot 2\pi t \ z_{m,n} \quad \text{(m = 0 .. M)}$$
$$\frac{\partial E_t}{\partial p_{m,n}} = \Gamma^A_{m,n} \cdot 2\pi \ z_{m,n} \quad \text{(m = 0 .. M)}$$

3 Experiments and Results

3.1 Implementation

Following the standard backpropagation techniques [4], for optimization, we employ stochastic gradient descent, Adam, and L^2 regularization with soft threshold. Within an epoch, as many time instants at which the loss is calculated as the size of the mini-batch are generated by the uniform random number generator over the designated temporal interval. In the following experiments, the mini-batch size is always set to 5. For simplicity, all parameters c's, p's, and w's are initialized by the normal distribution with the average being 0.0 and the variance being 0.1. Each layer, consisting of the oscillator units, is fully-connected to adjacent layers.

Here, we give a notice in setting the period of time t to preserve stability for the deep FM synthesis. As some may already noticed in Equation (3), the gradient chain inevitably includes term t^n, where n is the depth from the top layer. It follows that the amount of the loss feedback is proportional to t^n. Thus, if $0.0 < t < 1.0$, t^n may always become almost 0.0; on the other hand, if $t > 1.0$, t^n may become a larger value. Therefore, in the following experiments, the period of time t is put from 1.0 to 2.0. These values have been determined through several trials we made.

3.2 Loss Convergence

At the very first step, we would check if the backpropagation technique introduced in the previous section can work for the deep FM synthesis. Figure 2 shows

the loss convergences calculated by mean squared error as the epoch increases (Equation (2)), when the network size is depth 5 by width 5 and the training waveforms are rectangle and sawtooth with two cycles in the period of 1.0 to 2.0.

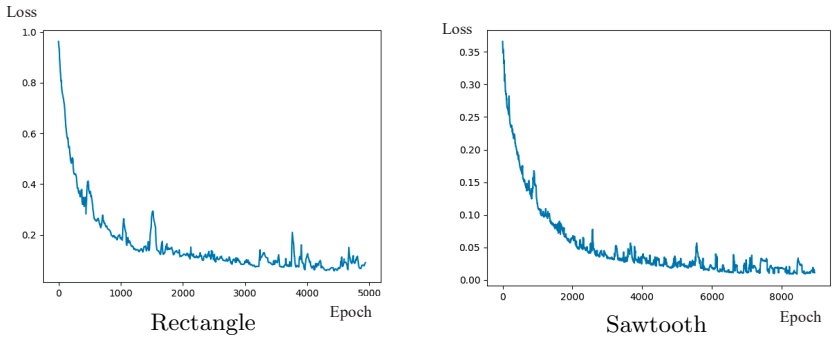

Fig. 2. Loss Convergences along with Epoch

Figure 3 shows the intermediate waveforms generated by the network at epochs 0, 500, and 4400 for rectangle and at epochs 0, 1000, and 8400 for sawtooth, respectively. Note that the training and output waveforms are shown only between T_s and T_e (i.e., 1.0 and 2.0). To obtain the above results, it took several minutes or less for each training phase, using Surface Pro 3 featuring Intel Core i7 CPU @ 1.70 GHz.

3.3 Network Size

We examine the relationship between the network size (depth $\times$ width) and the loss. The left-hand graph in Figure 4 shows the results for rectangle as the training waveform, and the right-hand graph sawtooth. In the both graphs, the depth is changed from 2 to 5 (colored broken lines), and the width from 1 to 8 (horizontal axes).

The loss in sawtooth converges faster along with the width increased than rectangle. At present, we presume the result could be understood by the complexity of a waveform as a figure. For example, while a cycle of rectangle contains two steep changes (-1.0 to 1.0 and 1.0 to -1.0), that of sawtooth contains one (1.0 to -1.0).

3.4 Training by Multiple Waves

We are interested in the learnability of the network, that is, how many wave forms the network can learn at the same time. Then, for training, the network is given the more than one pair of input vector and output wave form. In the experiment, during an epoch, three trainings are performed. During a single

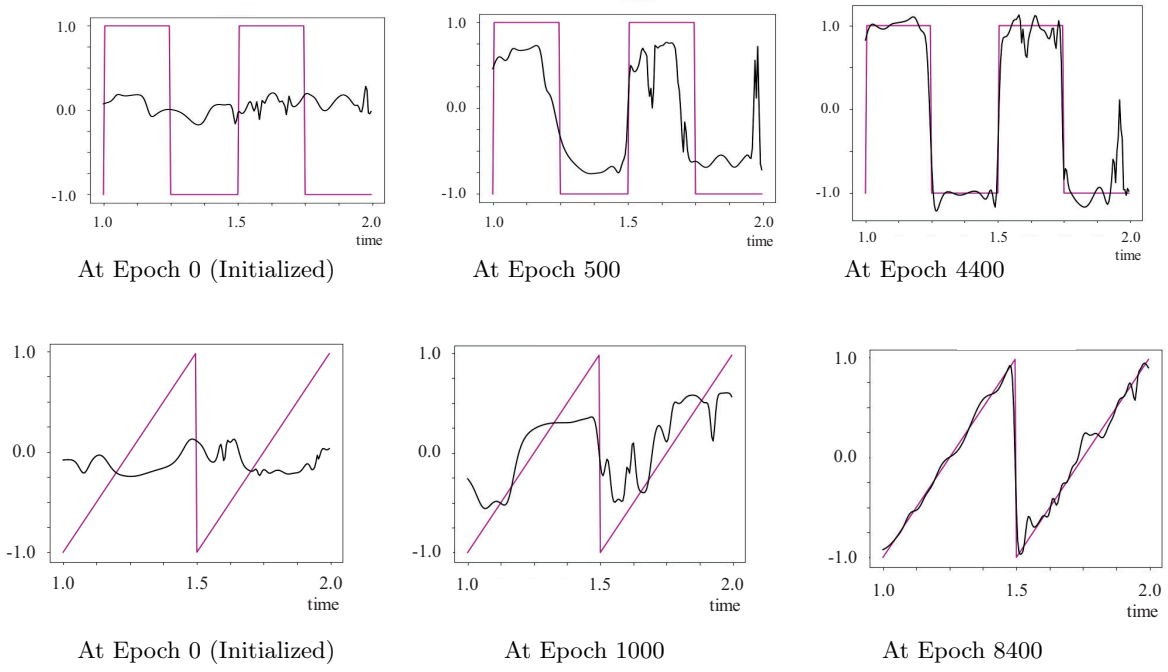

Fig. 3. Waveforms Generated During Training for Rectangle and Sawtooth

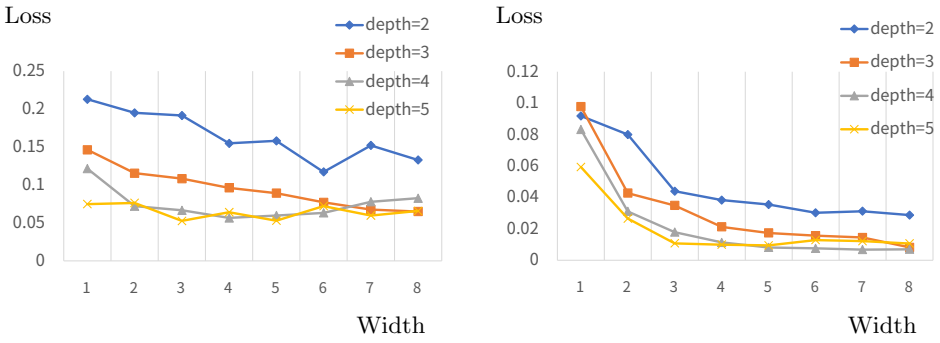

Fig. 4. Network Size and Loss

training, the network is given a pair of input and output by a mini-batch. The first pair is made of input vector of $\{1, 0, 0, \cdots\}$ and output signal of two-cycle

rectangle; the second $\{0, 1, 0, \cdots\}$ two-cycle sawtooth, and the third $\{0, 0, 1, \cdots\}$ two-cycle sinusoid.

Figure 5 shows the results of training the three pairs; the waves generated by the deep FM synthesis are drawn in the colored curves, and the training waves in black. We use the network of depth 5 by width 5.

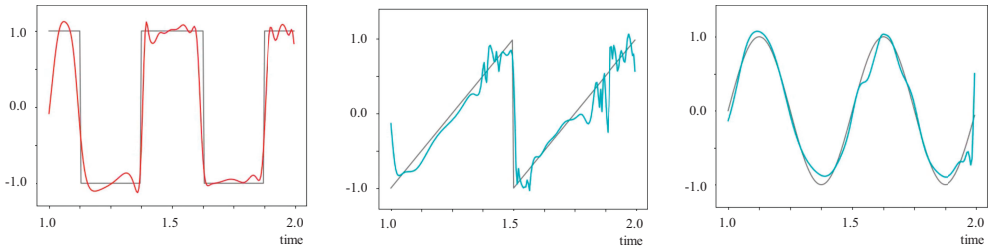

Fig. 5. Three Waves Generated by Network When Given Three Distinct Input Vectors

The leftmost graph in the figure is obtained at epoch 23000 with loss of 6.57 $\times 10^{-2}$ (near the optimal point for the rectangle), when given input $\{1, 0, 0, \cdots\}$. Similarly in the middle for sawtooth, the graph is obtained at epoch 21370 with loss of 2.12 $\times 10^{-2}$, and for sinusoid, at epoch 11680 with loss of 6.62 $\times 10^{-3}$.

For instance, let us compare the leftmost colored graph in Figure 5 with the upper rightmost in Figure 3 ("At Epoch 4400"). It seems that the generated wave form in the former is hardly deteriorated, even if that network learns three different wave forms at the same time, however, the epochs required for training is about 5 times larger.

4 Concluding Remarks

We propose the deep FM synthesis which is inspired by the conventional FM synthesis; it has the network architecture like neural networks and can be optimized by the backpropagation technique as neural networks. We have demonstrated that the deep FM synthesis works well to some extent for small-sized artificial training waveforms. Figure 6 shows an example when an unsteady, a little complicated training waveform (black curve in the figure) is given; the wave length and amplitude of it is varied along time. The horizontal axis in the figure stands for time in the unit of time instant. The conditional behavior presented in Section 3.4 is also promising for reconstructing multiple training waves.

The network attempts to approximate the output waveform (blue) to the training waveform; the network size is here depth 4 by width 20, and the network achieves the loss of 0.0583 at epoch 7570. However, the realistic waveforms of natural tones of acoustic instruments such as pianos or flutes are far longer

and contain many cycles. At present, the network of the deep FM synthesis cannot properly learn and reconstruct such real waveforms, unfortunately. The output waveform of the network does not sufficiently converge on a given training waveform under the current optimization method.

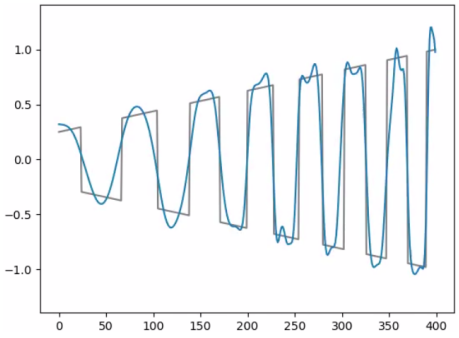

Fig. 6. Approximating Unsteady Waveform

Future work will include improving the approximation to the simple waveforms as given in Sections 3.2 and 3.3, and developing a tractable optimization method that can work effectivtly when learning realistic, long, complicated waveforms such as natural tones of acoustic instruments and voices. For the purpose, we would investigate other vibrating functions to be used for an activation function which must be differentiable and not necessarily periodic such as wavelet and phase modulator.

Acknowledgments

This work has been supported by JSPS Kakenhi 16H01744. The authors would like to thank to Prof. Ichiro Fujinaga of McGill University, Mr. Adrien Ycart of Queen Mary Univeristy, and Mr. Masafuji Takahashi of Future University Hakodate for fruitful discussions and valuable suggestions.

References

1. Chowning, J.M.: The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. J. the Audio Engineering Society **7**(21), 526–534 (1973)
2. Gashler, M.S., Ashmore, S.C.: Training Deep Fourier Neural Networks To Fit Time-Series Data (2014), arXiv preprint arXiv:1405.2262v1 (2014)
3. Godfrey, L.B.: Parameterizing and Aggregating Activation Functions in Deep Neural Networks. Ph.D. thesis, University of Arkansas (May 2018)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)

5. Mingo, L., Aslanyan, L., Castellanos, J., Daz, M., Riazanov, V.: Fourier Neural Networks: An Approach With Sinusoidal Activation Functions. International Journal "Information Theories & Applications" **11**, 52–55 (2004)
6. Roads, C.: The Computer Music Tutorial. The MIT Press (1996)
7. Wikipedia: Yamaha DX7. https://en.wikipedia.org/wiki/Yamaha_DX7, accessed: 2019/01/24