NAS-FM: Neural Architecture Search for Tunable and Interpretable Sound Synthesis Based on Frequency Modulation

Zhen Ye[1] , **Wei Xue**[2*] , **Xu Tan**[3] , **Qifeng Liu** [2,4] and **Yike Guo** [2*]

[1]Hong Kong Baptist University
[2]Hong Kong University of Science and Technology
[3]Microsoft Research Asia
[4]Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences

Abstract

Developing digital sound synthesizers is crucial to the music industry as it provides a low-cost way to produce high-quality sounds with rich timbres. Existing traditional synthesizers often require substantial expertise to determine the overall framework of a synthesizer and the parameters of submodules. Since expert knowledge is hard to acquire, it hinders the flexibility to quickly design and tune digital synthesizers for diverse sounds. In this paper, we propose "NAS-FM", which adopts neural architecture search (NAS) to build a differentiable frequency modulation (FM) synthesizer. Tunable synthesizers with interpretable controls can be developed automatically from sounds without any prior expert knowledge and manual operating costs. In detail, we train a supernet with a specifically designed search space, including predicting the envelopes of carriers and modulators with different frequency ratios. An evolutionary search algorithm with adaptive oscillator size is then developed to find the optimal relationship between oscillators and the frequency ratio of FM. Extensive experiments on recordings of different instrument sounds show that our algorithm can build a synthesizer fully automatically, achieving better results than handcrafted synthesizers. Audio samples are available at https://nas-fm.github.io/.

1 Introduction

Creating and rendering music has become increasingly convenient with the help of digital sound synthesizers which simulate the timbre of real instruments that are potentially expensive and rare. Lots of sound synthesizers are designed whose commercial values are widely recognized, while the complexity of the synthesizers also increases dramatically.

Instead of carefully designing delicate structures of highly diversified instruments, our target is to design a general framework which can flexibly construct digital synthesizers simply from recordings and allow further tuning of the timbres in a controllable and interpretable way.

Synthesizer	Tunable	Interpretable	Automatic Design
Digital Synthesizer	✓	✓	✗
Neural Network Synthesizer	✗	✗	✓
Our NAS-FM	✓	✓	✓

Table 1: Comparison between the proposed NAS-FM with other strategies for digital audio synthesis.

Early approaches to sound synthesis are parametric, designing a set of components (e.g., oscillators and modulators) based on digital signal processing (DSP), and ultimately creating complex timbres with specific structures combining these components. Typical works include subtractive synthesis [Huovilainen and Välimäki, 2005], additive synthesis [Serra and Smith, 1990], frequency modulation (FM) [Chowning, 1973] and wavetable synthesis [Bristow-Johnson, 1996], among which FM-based methods are widely used in sound synthesis due to their flexibility in tuning timbre with only a few parameters.

While satisfying sounds can be produced, designing the parametric structure of the synthesizer as well as parameter values requires considerable expertise. This is due to the non-linear interactions between synthesizer parameters, as well as the wide range of possible values for each parameter. Achieving a desired sound often involves a process of iterative adjustments and fine-tuning, which can be time-consuming and require a deep understanding of digital signal processing techniques. Although some efforts have been made to determine the parameters of FM synthesizer based on estimation theory [Justice, 1979], genetic algorithm [Horner, 1998], LSTM-based [Yee-King *et al.*, 2018], VAE-based [Le Vaillant *et al.*, 2021] and dilated CNN based [Chen *et al.*, 2022] methods, these methods are limited by static spectra or an audio clip conditioned on an entire ADSR envelope under a specific pitch value and a fixed duration length. Thus, these methods cannot be applied to the audio with continuously varying pitch and loudness on dynamic spectra.

Neural synthesis methods have been developed recently to

[*]Corresponding authors: Wei Xue {weixue@ust.hk}, Yike Guo{yikeguo@ust.hk}

learn a deep neural network to produce audio in the data-driven scheme and achieved impressive results in terms of audio fidelity. However, for fully-neural generative models such as WaveGlow [Prenger *et al.*, 2019], and HiFi-GAN [Kong *et al.*, 2020a], sufficient data is usually required to train a complex model with non-interpretable parameters, which greatly limits the controllability of the synthesizer as required by the music industry, specifically when the generated non-perfect sound needs tuning. Although the harmonic-plus-noise model-based differential DSP (DDSP) is integrated into the network to improve controllability [Engel *et al.*, 2020], it only allows the transfer of timbres between different instruments rather than explicit adjustment. FM-based neural synthesizer DDX7 [Caspe *et al.*, 2022] is then developed. However, this method can only learn the time-varying parameters of the submodules under the assumption that the overall structure has been manually defined.

The large reliance on expert knowledge and extensive time cost greatly limit the feasibility of building digital synthesizers for general instruments. Moreover, the handcrafted framework may make the resulting synthesizer suboptimal in modelling the real instrument. In this paper, we propose NAS-FM, a neural architecture search (NAS) based FM synthesizer. It can be seen as an effort to build interpretable neural generative models. A key of the proposed method is NAS. With a carefully designed search procedure, different structures and oscillator sizes are included in a universal search space of the supernet, and an evolutionary search algorithm is developed to find the optimal structure between oscillators and the frequency ratio of FM. The advantages of the proposed NAS-FM are described below:

- The audio synthesizer can be built based on recordings without any expert knowledge, largely simplifying the pipeline and reducing the cost of audio synthesizer construction. It is also possible to quickly build new variants of the target sound with flexible adjustments;
- It returns a tunable and interpretable conventional FM-based interface with a few parameters, making it easily embedded into existing audio workstations;
- Extensive experiments on recordings of different instruments demonstrate that synthesizers fully automatically built by the proposed NAS-FM can achieve better results to carefully handcrafted synthesizers.

2 Related Work

2.1 Sound Synthesis

Sound synthesis contains digital signal processing (DSP) methods and neural network methods. DSP methods have been integrated into the digital audio workstation used by musicians. More specifically, DSP methods start from several simple waves such as sine, sawtooth, square and triangle generated by an oscillator. The additive synthesizer [Serra and Smith, 1990] generates new sounds by adding various simple waves. The Subtractive synthesizer [Huovilainen and Välimäki, 2005] filters a simple wave from white noises. FM synthesizers [Chowning, 1973] rely on simple waves to modulate frequency to create complex timbre. Wavetable synthesis manipulate a collection of short samples to design new sounds. These traditional methods need users to determine the configuration manually for a given sound.

Neural network synthesis models adopt the deep neural network to learn the mapping function between audio and given input, for instance, pitch and loudness. The early exploration begins with auto-regressive models such as WaveRNN [Kalchbrenner *et al.*, 2018] and SampleRNN [Mehri *et al.*, 2016]. The following works [Engel *et al.*, 2019] [Kong *et al.*, 2020b] are based on various generative models to further improve the quality of synthesis sound. However, the above methods may lead to glitch problems because of the lack of phase continuity. Therefore, DDSP [Engel *et al.*, 2020] rely on the harmonic-plus-noise model to keep the phase continuous and also make the sound can be directly controlled by pitch and loudness. While these methods can be optimized automatically with the help of gradient descent to obtain the model, there are few control factors to help users manipulate the synthesized result directly. Therefore, our approach aims to introduce the FM synthesizer with controllable factors to help the user interact with the synthesized audio.

2.2 FM Parameter Estimation

FM parameter estimation also called FM matching is adopted to determine the configuration of an FM synthesizer. The early approach considers this problem as a searching problem that uses the genetic algorithm(GA) [Jh, 1975] or its variants to find a best-fit configuration in a specific searching space. Horner employs GA solving a sound matching problem for different FM algorithms such as Formant FM [Horner *et al.*, 1993], Double FM [Horner, 1996], Nested FM and Feedback FM [Horner, 1998]. These methods can achieve very close re-synthesizing results with a static target spectrum when selecting an appropriate FM algorithm as prior. The following methods leverage an open-source FM synthesizer Dexed synthesizer [Gauthier, 2013] to construct a large number of pair data between presets[1]collected on the Internet and synthesized audio clip generated by Dexed. By reversing this process, these works employ LSTM [Yee-King *et al.*, 2018], VAE [Le Vaillant *et al.*, 2021] and dilated CNN [Chen *et al.*, 2022] to estimate the preset. Since the synthesized sound is generated by pressing a note with specific velocity, duration and pitch using Dexed synthesizer, the result of the model for a realistic sound audio clip without any prior is unpredictable. DDX7 [Caspe *et al.*, 2022] predict the envelopes of oscillators using the widely-used TCN [Oord *et al.*, 2016] decoder with algorithm and frequency ratios as prior. Therefore, our method is designed to construct an FM synthesizer without prior knowledge.

2.3 Neural Architecture Search

Neural Architecture Search (NAS) can automatically find the best neural architecture for a specific task. Many works focus on computer vision [Liu *et al.*, 2018]or natural language processing [Xu *et al.*, 2021] tasks. More recently, one-shot NAS

[1]Preset means a full FM configuration for specific sound designed by users

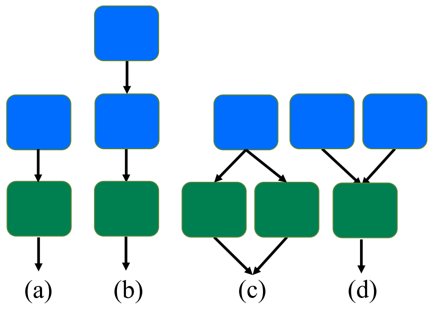

Figure 1: Different FM algorithms: (a) Single FM; (b) Nested FM; (c) Formant FM; (d) Double FM. The green box refers to the carrier and the blue box refers to the modulator

[Guo *et al.*, 2020] [Bender *et al.*, 2018] train a shared supernet once which includes all candidate architectures. Then, the supernet is used as an estimator to evaluate every possible architecture in the search space. This method has been widely used on various applications to determine a good structure, for example, image classification [Guo *et al.*, 2020], object detection [Liang *et al.*, 2021], 3d scene understanding [Tang *et al.*, 2020], BERT compression [Xu *et al.*, 2022]. These methods introduce neural architecture search to each specific task which gets a better model architecture than manual design. Although NAS has been widely used in lots of areas, the applications are mainly focused on neural networks. Actually, we are the first to bring NAS to build the sound synthesizer.

3 Review of FM Synthesizer

3.1 Basics of FM

FM was originally proposed in [Chowning, 1973] for sound synthesis and different timbres are produced by controlling a set of parameters. With two sound sources, the modulator oscillator $\sin(2\pi f_m t)$ and the carrier oscillator $\sin(2\pi f_c t)$, FM basically generates a time-domain signal $y(t)$:

$$y(t) = a(t)\sin(2\pi f_c t + I\sin(2\pi f_m t)), \tag{1}$$

where f_c and f_m are the carrier frequency and modulation frequency respectively, I is the modulation index, and $a(t)$ is the amplitude envelope of the carrier. $y(t)$ can be further decomposed by using Bessel functions of the first kind as

$$y(t) = a(t)\sum_{n=-\infty}^{n=+\infty} J_n(I)\sin(2\pi(f_c + nf_m)\cdot t) \tag{2}$$

which shows that the sidebands of $y(t)$ distribute evenly around f_c with spacing as f_m, and the spectra is harmonic when the frequency ratio $r \in \mathbb{Q}$ where $r = f_c/f_m$. $J_n(I)$ is a Bessel function of the modulation index I, and dynamic spectra can be generated if the I becomes a time-variant function $I(t)$.

3.2 FM Algorithms

The (1) explains the basic module of the FM synthesizer to generate sounds. More diversified and expressive FM synthesizers can be further developed by designing complicated topologies, which are called "FM algorithms", on connecting the carrier and modulator oscillators.

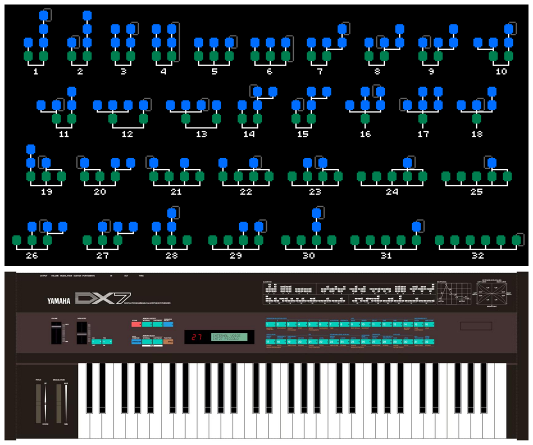

Figure 2: Digital synthesizers in the commercial product YAMAHA DX7. The lower part is the user interface of a synthesizer. The upper part shows the FM algorithms for audio synthesis, with green and blue boxes denoting carriers and modulators, respectively.

Typical FM algorithms are shown in Fig. 1. Fig. 1(a) denotes the single FM expressed by (1). The Nested FM [Justice, 1979], formant FM [Horner *et al.*, 1993], and double FM [Schottstaedt, 1977] are illustrated in Fig. 1(b)-(d), whose outputs are calculated by

$$\begin{aligned} y(t) =& a(t)\sin(2\pi f_c t + I_{m1}(t)\sin[2\pi f_{m1} t \\ &+ I_{m2}(t)\sin(2\pi f_{m2} t)]), \end{aligned} \tag{3}$$

$$\begin{aligned} y(t) =& a_1(t)\sin[2\pi f_{c1} t + I_m(t)\sin(2\pi f_m t)] \\ &+ a_2(t)\sin[2\pi f_{c2} t + I_m(t)\sin(2\pi f_m t)], \end{aligned} \tag{4}$$

and

$$\begin{aligned} y(t) =& a(t)\sin[2\pi f_c t + I_{m1}(t)\sin(2\pi f_{m1} t) \\ &+ I_{m2}(t)\sin(2\pi f_{m2} t)], \end{aligned} \tag{5}$$

respectively. These FM algorithms produce different timbres, and as shown in Fig. 2[2], by selecting different FM algorithms.

To briefly summarize, a set of parameters controls the FM output, which can be time-variant or fixed. In our work, time-variant parameters include the oscillators' envelopes, i.e., $a(t)$ and $I(t)$, which are determined by the time-variant input, such as f_0 and loudness. Fixed parameters are the chosen FM algorithm and the frequency ratio of each oscillator, which need to be determined by the expected timbre.

4 Proposed NAS-FM Synthesizer

In this section, we aim to fully automatize the designing of an FM synthesizer in a data-driven manner, thus eliminating the reliance on expertise and labour to tune the timbres.

A NAS-FM synthesizer is proposed, with an overall framework shown in Fig. 3 on the next page. For input audio, the pitch and loudness are extracted, and these two features

[2]We do not consider the feedback FM in our work as the same reason in [Caspe *et al.*, 2022]

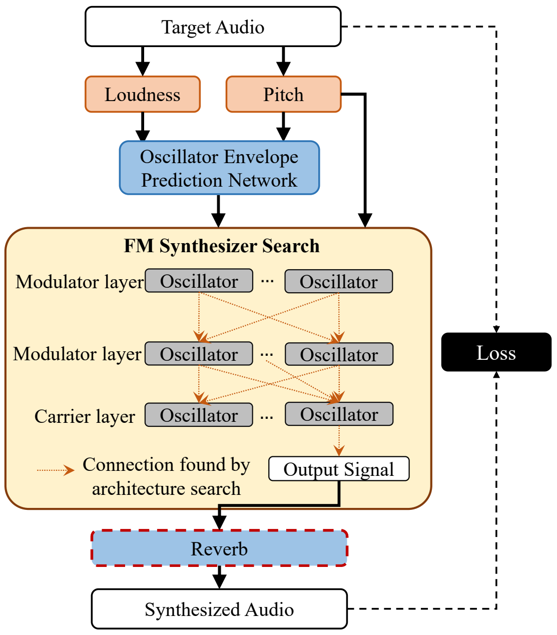

Figure 3: The overall architecture of NAS-FM. The learnable Reverb module is optional depending on whether the real room effect should be simulated for real-world audios.

are fed into an oscillator envelope prediction network to estimate the envelopes of the oscillators. We develop architecture search methods to determine the optimal FM algorithm, and given the FM algorithm and oscillator envelopes, sounds with a specific timbre can be produced. Depending on whether the sound is from the real environment, a learnable reverb module [Engel *et al.*, 2020] can be optionally used to simulate the room effects. The framework is optimized in an auto-encoder setting, i.e., seeking to recover the original signal in the final output.

Besides the FM algorithm search, the framework is similar to the framework in DDX7 [Caspe *et al.*, 2022], which estimates pitch and loudness by CREPE [Kim *et al.*, 2018] and A-weighting loudness [Moore *et al.*, 1997], designs the oscillator envelope prediction network as a temporal convolutional network (TCN) [Bai *et al.*, 2018]. However, DDX7 requires a prior FM configuration designed by experts. By adopting NAS, the proposed framework designs the FM synthesizer in the fully-automated data-driven pipeline and also returns a tunable and interpretable interface used by musicians. In the following, details of the proposed NAS-FM will be introduced, which include a) converting the FM algorithm to a graph, b) designing the search space, c) training the supernet, and d) selecting the FM algorithm.

4.1 Directed Acyclic Graph of FM Algorithm

To facilitate discussion in this section, we convert an FM synthesizer, with examples shown in Fig. 1, to a directed acyclic graph (DAG) [Pham *et al.*, 2018] with an ordered sequence of N nodes, where N represents the number of oscillators. Each node $x^{(i)}$ refers to the oscillator's output. Each directed edge (i, j) indicates whether node $x^{(j)}$ is the modulating node of $x^{(i)}$. The topology of the graph is associated with the FM algorithm. Thus, an intermediate node $x^{(i)}$ can be expressed as

$$x^{(i)}(t) = a_i(t)\sin(2\pi f_i t + \sum_{j\in\mathbb{M}} x^{(j)}(t)), \tag{6}$$

where $\mathbb{M}$ is a set of oscillators modulating node $x^{(i)}$. When M is empty, $x^{(i)}$ outputs standard sine wave. The output of FM is calculated through the sum of carrier nodes

$$y(t) = \sum_{i\in\mathbb{C}} x^{(i)}(t), \tag{7}$$

where $\mathbb{C}$ is the set of carriers.

4.2 Search Space Design

There are a huge number of possible configurations for the FM synthesizer. By converting the FM algorithm to the DAG, the principles of NAS can be applied to design the FM algorithm. In NAS for neural networks, all possible architectures of the network, which span the "search space", can be represented by a general DAG, with each candidate architecture as a sub-graph. Similarly, we design the search space for FM algorithm here.

We visualize the operation in (6) in Fig. 4, where two issues should be solved to define an oscillator: a) what is the frequency ratio? The frequency ratio is defined originally in (2), and is the ratio between f_i in (6) and fundamental frequency F_0 here; b) which modulators will be connected to the current oscillator? As depicted in Fig. 4, we define the frequency ratio set $\mathbb{FR} = \{1, 2, 3, .., K\}$ which consists of K integers, and assumes that there are N candidates in the modulator set $\mathbb{M}$. Actually, the interval in the frequency ratio set can be reduced to 0.1 or even smaller, to 0.01 instead of 1, for a more refined exploration.

Now let us analyze the design of the search space. Previous works adopted evolutionary search on frequency ratio set under a fixed FM algorithm such as double FM [Horner, 1996] and nested FM [Horner, 1998]. Therefore, the challenge is how to design an appropriate space including various FM algorithms, instead of enumerating all possible FM algorithms. In [Guo *et al.*, 2020] and [Xu *et al.*, 2022], the weight sharing is proposed to reduce the search space, which forces subgraph candidates to have the same weights in the graph nodes commonly shared by different candidates. This also makes it possible to directly train a supernet including all possible candidates. Here we propose a novel envelope-sharing strategy for FM synthesizers to make all FM configuration candidates trained in the supernet.

Specifically, we construct a search space as shown in Fig. 3. The search space can be divided into a carrier layer and several modulator layers. We set the oscillator at the same layer with the same frequency ratios sharing the same envelope. Due to the envelope-sharing strategy, There are following rules in our search space. (1) A oscillator in a certain layer can only be modulated by the upper layer; (2) The sum of the output of selected oscillators at the carrier layer forms the final signal;(3) A oscillator is discarded when there is no connection with other oscillators or final output. In addition, according to our experience, we find two modulator layers

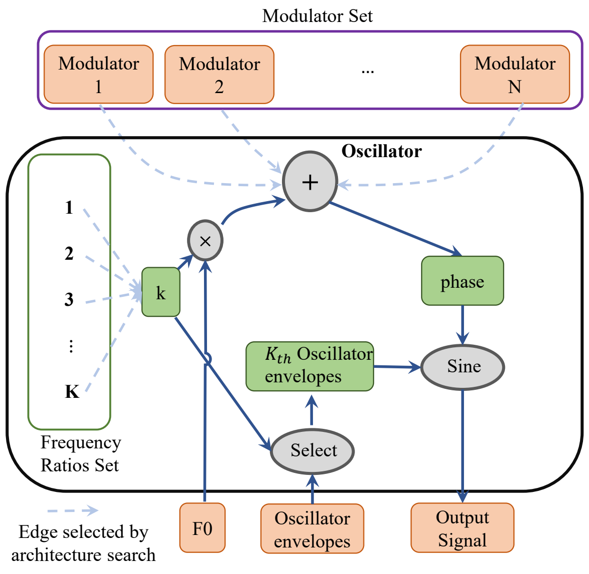

Figure 4: The oscillator in NAS-FM. The output of an oscillator conditioning on $F0$ depends on the choice of frequency ratio, envelope and modulator. The choices are determined by NAS.

are enough. The number of candidate oscillators in a layer depends on the oscillator number in the expected FM synthesizer.

4.3 Supernet Training with Proxy Oscillator

Although the proposed envelope-sharing strategy makes all possible FM configurations can be trained using a supernet that includes all candidates, we still encounter the challenge of a large search space. The huge search space makes it hard to evaluate each FM configuration in the supernet during training. To further improve the training efficiency, we propose to use the “proxy oscillator” as the proxy for all oscillators in each layer to determine the envelopes of the oscillators.

Specifically, during supernet training, a fixed Nested FM in (3) with a carrier and two modulators are chosen, which has three layers in accordance with the settings in the Sec. 4.2. For each oscillator, uniform sampling [Guo *et al.*, 2020] is adopted to determine the frequency ratio. With these two configurations fixed, the envelopes of the oscillators are learned, and after training, the learned envelopes are shared within the same layer for a complicated search space. Utilizing the technique, we can expand the width of the search space flexibly.

4.4 FM Configuration Selection

After the supernet training and search space design, we use the evolutionary algorithm to conduct FM algorithm selection. Specifically, we put all N candidate oscillators ol in order. a certain FM configuration is encoded as an individual can be formulated as

$$\{f_{ol_1}, f_{ol_2}, ..., f_{ol_N}, l_{ol_1} l_{ol_2}, ..., l_{ol_N}\} \tag{8}$$

where ol_i is the i_{th} oscillator, corresponding f_{ol_i} and l_{ol_i} indicate the selected frequency ratio and connection relationship. The connection relationship is the relation between the lower layer. If the oscillator belongs to the carrier, the lower layer is the output signal. If the connection relationship of this oscillator is none means it is discarded. Firstly, a random population is initialized within the initial space. Secondly, we evaluate the fitness score of the generated individuals and select the top individuals. The fitness function will be introduced in the experiment section. Thirdly, crossover and mutation are used to generate new individuals to update the population, until meeting the stopping criterion.

5 Experiments

To evaluate the proposed NAS-FM approach which aims to automatically learn the FM synthesizer that is applicable to the music industry, we aim to answer the following questions: a) Given sound recordings, can the FM synthesizers learned by the proposed method be comparable to the manually designed synthesizers? b) We fuse different FM algorithms into one universal search space, is this strategy better than separately searching the best configuration for each FM algorithm? c) Can we controllably tune the timbre of the produced sounds or create new instrument timbres by modifying the parameters of the learned FM synthesizer?

5.1 Dataset

We conduct experiments on the benchmark URMP dataset [Li *et al.*, 2018]. Sound recordings of three real instruments, which are violin, flute, and trumpet, are chosen and the “optimal” manually designed FM algorithms of these instruments are given as in [Caspe *et al.*, 2022]. The three instruments are also the most typical instrument of the strings, woodwinds, and brass. Each instrument recording is divided into 4-second segments, and silent segments are discarded. The loudness and pitch are estimated by the A-weighting loudness [Moore *et al.*, 1997] and CREPE [Kim *et al.*, 2018] methods. Each audio clip is resampled to the 16kHz sampling rate and analyzed with a frame size of 2048 and a hop size of 64, yielding 1000 frames. We split the dataset into train, validation, and test sets with proportions of 0.75, 0.125, and 0.125, respectively.

5.2 Experimental Setup

For training, a supernet contains $c * m * m$ paths where c is the number of frequency ratios of the carrier, and m is the number of frequency ratios in modulators. We set c as 15 and m as 5. The model adopts the stack TCN architecture [Bai *et al.*, 2018] as a sequence-to-sequence model to predict oscillator envelope through pitch and loudness. We first stack 4 TCN architecture to extract a hidden temporal feature. Then, we construct $(c + m + m)$ TCN architectures with different weights to predict the envelope of the corresponding oscillator from the hidden feature. In fact, other sequence-to-sequence models can also be employed to model the temporal relationship in our pipeline. In addition, uniform sampling of oscillators on each layer with different ratios is adopted. We use Adam optimizer with an initial learning rate of 3e-4. For regularize, we set the maximum value of the oscillator for the carrier and modulator as 1 and 2, respectively. The exponential decay strategy with a decreasing factor of 0.98 every 10k

	Fréchet Audio Distance (↓)		
Model	Flute	Violin	Trumpet
Test Data	1.180	0.308	0.554
DDX7	7.841	3.497	4.442
NAS-FM	**7.077**	**3.255**	**3.384**

Table 2: FAD of resynthesis results using 6 oscillators

steps is adopted for the learning rate .The whole training steps are 500k, and the batch size is 16.

For searching, we use the evolutionary algorithm with a population size of P, crossover size of $P/2$, and mutation size of $P/2$ with mutation probability and max iterations of T depending on the search space size. In addition, since we use the fixed number of the candidate oscillator, the discarded candidate oscillator with different frequency ratios will cause the same fitness score. Therefore, we force the generated candidates with a unique fitness score to be legal.

After searching, we directly extract the weight of the sub-model with the best fitness score from the supernet as our final model. Actually, fine-tuning the model or training from scratch with the searched FM configuration may further improve the performance. However, our aim is not to focus on the resynthesis performance but to get a controllable synthesizer close to the target sound so that musicians can further utilize it to tweak the sound or do other more interesting things such as sound morphing and sound interpolation.

5.3 Evaluation Metric

We use Fréchet Audio Distance (FAD) [Kilgour *et al.*, 2018] as the evaluation metric to measure the distance between real and generated sound. This method extracts the embedding of the audio using a pre-trained VGG-like model. The distance is calculated as follows:

$$FAD = \|\mu_r - \mu_g\|^2 + tr(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}) \qquad (9)$$

where r and g are the real audio and generated audio respectively, μ_r and μ_g are the mean vector of embedding and Σ_r and Σ_g are the covariances of embedding. A smaller Fréchet Audio Distance indicates a higher level of similarity between the distributions of real and generated data. In our experiment, we calculate the Fréchet Audio Distance between the real data and synthesize validation data as the fitness score during searching. And calculating the FAD between the synthesized test data as the final evaluation result.

5.4 Comparison with Manually-Designed FM Synthesizers

In this part, we are interested to know if our method could achieve comparable results to manually designed FM synthesizers. The baseline is DDX7 [Caspe *et al.*, 2022] which the author retrieves on the web manually to find the patch with the most similar sound to the target sound. The patches they find have six oscillators, and we adopt the same oscillator number with them. Since the authors did not open source their evaluation code, we train the model following their method ten times with a random seed and took the best one as their result.

In our method, we use a $3 * 3$ candidate oscillator search space forcing three oscillators discarded to ensure six oscillators. During the search, we follow the above procedure and set the population size as 1000 and max iterations as 50. Results are shown in Table 2. The Test Data line means the Fréchet Audio Distance between the entire real data from the real test data. We can see that our NAS-FM outperforms the hand-designed DDX7 baseline across all musical instrument recordings. The results show that our NAS-FM can search for more comparative FM configurations than manually designed FM synthesizers.

	Fréchet Audio Distance (↓)		
Model	Flute	Violin	Trumpet
Nested FM	12.75	**6.02**	**8.24**
Formant FM	14.48	7.56	8.68
Double FM	**11.16**	7.27	9.91
Single+ FM	11.20	12.07	9.28
NAS-FM	**11.16**	**6.02**	**8.24**

Table 3: Ablated of search space using 3 oscillators

5.5 Ablation Study of Search Space

The previous methods of FM parameter estimation [Chen *et al.*, 2022] [Mitchell and Sullivan, 2005] [Horner *et al.*, 1993] were usually constrained to a specific FM algorithm. However, our approach puts the different FM algorithms in one search space. To demonstrate the significance of the method, we first want to answer whether different timbres have their own FM algorithms that are more suitable for them when operator size is fixed. If the answer is yes, can our method find the algorithm that best fits this timbre? We conduct our experiments conditioned on three oscillators and enumerate all possible FM algorithms:

- Nested FM: a carrier with two nested modulators as shown in Fig. 1(b);
- Formant FM: two carriers sharing a modulator as shown in Fig. 1(c);
- Double FM: a carrier with two modulators in the same row as shown in Fig. 1(d);
- Single FM+: a single oscillator adds a single FM. Single FM is shown in Fig. 1(a);
- NAS-FM (Ours): Constructing a $3 * 2$ candidate oscillator search space forcing three oscillators discarded to search both FM algorithms and frequency ratios.

During the search process, the search space of a fixed FM algorithm equals only contains the frequency ratios of each oscillator. In addition, we set the population size as 30 and max iterations as 20 for each method for a fair comparison. The results are shown in Table 3.

To answer the first question, we compare the result across all FM algorithms. The Nested FM achieves the best performance on violin and trumpet. However, this algorithm

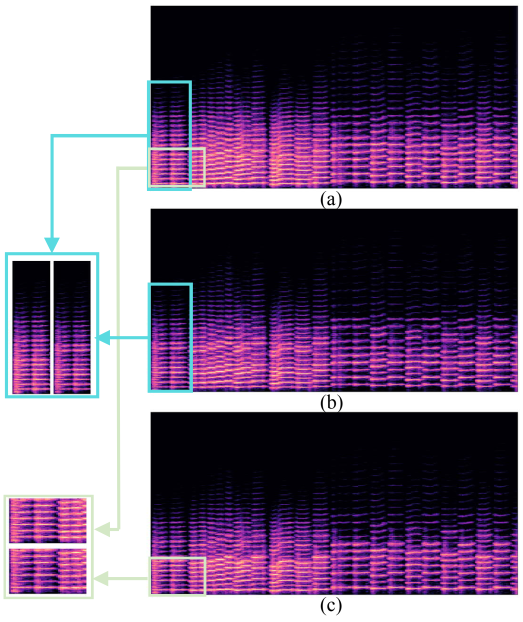

Figure 5: Examples of sound morphing. The top figure is the synthesized sound of the trumpet. In the middle and bottom figures, we morph the sound by tuning a certain parameter in our NAS-FM.

yields the second-worst performance on the Flute. The double FM performs the best on the Flute but worst on the trumpet. Therefore, we find that no fixed FM algorithm achieves the best results for every musical instrument recording. However, our method can always find the best FM configuration for different instruments under the same searching setting. This proves the merit of our well-designed search space.

5.6 Tuning of Learned FM Synthesizers

In this section, we check the tuning ability of NAS-FM. Two important tasks in sound synthesis are considered: sound morphing and timbre interpolation.

Sound Morphing

Due to the timbre of an instrument being controlled by a set of frequency ratio parameters, we want to show the ability of our NAS-FM tweaks the timbre from a trained synthesizer by adjusting a few parameters. Begin with a synthesizer designed for the trumpet, Fig. 5 (a) is the synthesized recording of the trumpet using our NAS-FM. More specifically, the searched FM algorithm is the same as the 7-th algorithm shown in Fig. 2 without the feedback module. It consists of two parts. The left part is a single FM in which the carrier frequency ratio is three and the modulator is one. The right part includes four oscillators which consist of a double FM with one more nested modulator. The frequency ratios among a carrier, double modulators and a nested modulator are 7,1,2 and 1, respectively. In Fig. 5(b), we modify the second formant position from the 7th harmonic to the 10th harmonic by modifying the frequency ratio of the carrier in the right part from seven to ten. In addition, we decrease the second and fourth harmonic by tuning the frequency ratio of the modulator in the left part from one to two, as shown in Fig. 5(c). We can find that due to the adjustment of one parameter in NAS-FM, the entire distribution and details of frequency can be easily changed.

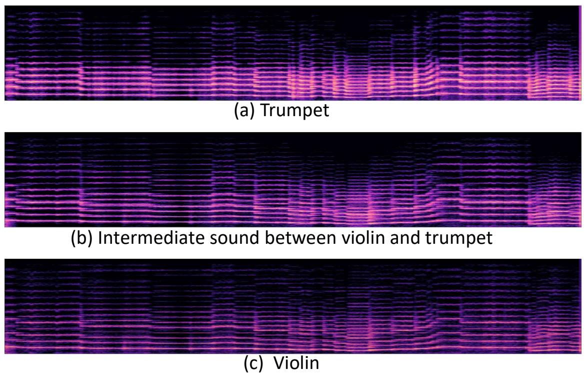

Figure 6: An example of timbre interpolation. The top and bottom figures show the linear spectrogram of the synthesized sound of the trumpet and violin, respectively. The middle figure shows the interpolation result between the violin and the trumpet.

Timbre Interpolation

Another meaningful application is timbre interpolation. Since our approach determines the FM configuration by searching for the Fréchet Audio Distance to the target audio clip. We find that the target audio clips can produce new timbre. Therefore, we can change our fitness score to a variant form. For instance, we set the FAD from synthesized audio to violin as d_v and the FAD from synthesized audio to trumpet as d_t. Next, we define the novel fitness function as $d_v + d_a + |d_v - d_a|$ aiming to find a synthesizer to generate sound regarded as an intermediate timbre between violin and trumpet. Surprisingly, we use the supernet trained from violin recordings to conduct the evolutionary algorithm and get an interesting result. As shown in Fig 6, we obtain a new instrument recording similar to both violin and trumpet.

6 Conclusion

We present NAS-FM, a tunable and interpretable sound synthesizer based on frequency modulation. Given a target sound, we prove that our method can automatically design an FM synthesizer instead of spending a huge time designing it manually. Meanwhile, our auto-designed synthesizer can achieve comparable results to the handcrafted one. Furthermore, Our NAS-FM leverages the widely-used FM synthesizer as the main component in our framework. This makes our method can be directly understood by musicians that can be used to create new sounds without extra knowledge of the neural networks.

Acknowledgments

The research was supported by the Theme-based Research Scheme (T45-205/21-N) and Early Career Scheme (ECS-HKUST22201322), Research Grants Council of Hong Kong.

References

[Bai *et al.*, 2018] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[Bender *et al.*, 2018] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International conference on machine learning*, pages 550–559. PMLR, 2018.

[Bristow-Johnson, 1996] Robert Bristow-Johnson. Wavetable synthesis 101, a fundamental perspective. In *Audio engineering society convention 101*. Audio Engineering Society, 1996.

[Caspe *et al.*, 2022] Franco Caspe, Andrew McPherson, and Mark Sandler. Ddx7: Differentiable fm synthesis of musical instrument sounds. *arXiv preprint arXiv:2208.06169*, 2022.

[Chen *et al.*, 2022] Zui Chen, Yansen Jing, Shengcheng Yuan, Yifei Xu, Jian Wu, and Hang Zhao. Sound2synth: Interpreting sound via fm synthesizer parameters estimation. *arXiv preprint arXiv:2205.03043*, 2022.

[Chowning, 1973] John M Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the audio engineering society*, 21(7):526–534, 1973.

[Engel *et al.*, 2019] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[Engel *et al.*, 2020] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

[Gauthier, 2013] Pascal Gauthier. dexed. https://github.com/asb2m10/dexed, 2013. Accessed: 2023-06-07.

[Guo *et al.*, 2020] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pages 544–560. Springer, 2020.

[Horner *et al.*, 1993] Andrew Horner, James Beauchamp, and Lippold Haken. Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis. *Computer Music Journal*, 17(4):17–29, 1993.

[Horner, 1996] Andrew Horner. Double-modulator fm matching of instrument tones. *Computer Music Journal*, 20(2):57–71, 1996.

[Horner, 1998] Andrew Horner. Nested modulator and feedback fm matching of instrument tones. *IEEE Transactions on Speech and Audio Processing*, 6(4):398–409, 1998.

[Huovilainen and Välimäki, 2005] Antti Huovilainen and Vesa Välimäki. New approaches to digital subtractive synthesis. In *ICMC*, 2005.

[Jh, 1975] Holland Jh. Adaptation in natural and artificial systems. *Ann Arbor*, 1975.

[Justice, 1979] James Justice. Analytic signal processing in music computation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(6):670–684, 1979.

[Kalchbrenner *et al.*, 2018] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[Kilgour *et al.*, 2018] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

[Kim *et al.*, 2018] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[Kong *et al.*, 2020a] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

[Kong *et al.*, 2020b] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[Le Vaillant *et al.*, 2021] Gwendal Le Vaillant, Thierry Dutoit, and Sébastien Dekeyser. Improving synthesizer programming from variational autoencoders latent space. In *2021 24th International Conference on Digital Audio Effects (DAFx)*, pages 276–283. IEEE, 2021.

[Li *et al.*, 2018] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 21(2):522–535, 2018.

[Liang *et al.*, 2021] Tingting Liang, Yongtao Wang, Zhi Tang, Guosheng Hu, and Haibin Ling. Opanas: One-shot path aggregation network architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10195–10203, 2021.

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[Mehri *et al.*, 2016] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[Mitchell and Sullivan, 2005] Thomas Mitchell and Charlie Sullivan. Frequency modulation tone matching using a fuzzy clustering evolution strategy. In *Audio Engineering Society Convention 118*. Audio Engineering Society, 2005.

[Moore *et al.*, 1997] Brian CJ Moore, Brian R Glasberg, and Thomas Baer. A model for the prediction of thresholds, loudness, and partial loudness. *Journal of the Audio Engineering Society*, 45(4):224–240, 1997.

[Oord *et al.*, 2016] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[Pham *et al.*, 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[Prenger *et al.*, 2019] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[Schottstaedt, 1977] Bill Schottstaedt. The simulation of natural instrument tones using frequency modulation with a complex modulating wave. *Computer Music Journal*, pages 46–50, 1977.

[Serra and Smith, 1990] Xavier Serra and Julius Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[Tang *et al.*, 2020] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020.

[Xu *et al.*, 2021] Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. Nas-bert: task-agnostic and adaptive-size bert compression with neural architecture search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1933–1943, 2021.

[Xu *et al.*, 2022] Jin Xu, Xu Tan, Kaitao Song, Renqian Luo, Yichong Leng, Tao Qin, Tie-Yan Liu, and Jian Li. Analyzing and mitigating interference in neural architecture search. In *International Conference on Machine Learning*, pages 24646–24662. PMLR, 2022.

[Yee-King *et al.*, 2018] Matthew John Yee-King, Leon Fedden, and Mark d'Inverno. Automatic programming of vst sound synthesizers using deep networks and other techniques. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):150–159, 2018.