Contents in the appendix

The appendix is organized as follows: Firstly, we introduce the related work of our proposed model in Appendix A. Detailed notation is developed in Appendix B. The variants of APOLLO are studied in Appendix C.1, C.2. The differences from Π-Nets are explored in Appendix C.3. In Appendix D, the derivations and the proofs in the main body are further elaborated. Regarding experiments, the evaluation metrics of audio generation are presented in Appendix E.2. Further details of the experimental setup on unconditional audio generation, conditional audio generation, and multimodal generation are developed in Appendix E.3, E.4, E.5, respectively. Ablation studies on our models are included in Appendix F. Additional analyses for the inference speed, human study, and phase information are discussed in Appendix G.1, G.2, G.3. Experiments on speech recognition and speech enhancement are included in Appendix G.4. Lastly, the societal impact is described in Appendix H.

A. Related work

Audio generation. Unconditional audio generation aims to generate audio creatively, which has practical significance in the real world [39; 9; 11; 45]. WaveGAN and SpecGAN [9] are the first attempts to unconditionally synthesize audio using GANs. WaveGAN directly models the raw audio while SpecGAN models the log spectrum of the STFT in the frequency domain. The drawback of SpecGAN includes the demand for phase recovery with Griffin-Lim [16] during synthesis step. GANSynth [11] improves the previous work by jointly modeling the mel-spectrogram and the instantaneous frequency, which results in more consistent harmonics and avoids additional phase estimation technique. However, the phase misalignment in GANSynth leads to energy loss due to the cancellation of frequency bands [39]. Similar to SpecGAN, TiFGAN [39] only models the log spectrum of the STFT while it estimates the phase by Phase-gradient heap integration (PGHI) [46], which avoids expensive iterative Griffin-Lim in SpecGAN and magnitude integration in GANSynth. Gunasekaran et al. [19] improve the conditional spoken digit generation using mel-spectrogram and Griffin-Lim. Haque et al. [21] propose GGAN to achieve high fidelity audio generation with a fewer labelled audio clips, which rely on log spectrum representation and PGHI recovery. Contrary to most aforementioned methods, our APOLLO directly models the STFT in the complex domain without using phase recovery. Apart from adversarial generation, diffusion-based models (DiffWave [35]) and state space [17] based models (SASHIMI [14]) are also receiving increasing attention. Compared with adversarial methods, DiffWave results in slower inference speed due to the reverse process. Lastly, text-to-speech (TTS) systems, as another category of audio generation, generate the audio representation based on the text information and then utilize a neural vocoder to obtain the waveform. Many solutions are proposed in recent years [63; 30; 66; 54; 62]. Both SASHIMI and DiffWave can also be implemented as neural vocoder and achieve excellent performance against WaveNet [63].

Complex-valued neural networks (CVNNs). The ideas behind CVNNs can be traced back to at least Kim & Guest [32] that derive backpropagation rules with respect to a real and an imaginary part. The mathematical motivation as well as the biological feasibility of CVNNs has been explored during the previous decade [48; 61]. Being equipped with a rich representational capacity in the complex field, CVNNs have drawn increasing attention in a wide range of applications. Oyallon & Mallat [44] design a deep scattering convolution network to improve the performance of image descriptors. Trouillon et al. [60] use complex-valued embeddings for simple link prediction. In audio-related tasks, the noise speech is enhanced by being converted and passed through STFT, complex-valued network, and inverse STFT [3]. Another line of research applies complex operations on top of typical neural networks. Arjovsky et al. [1] use complex operations on recurrent neural network to circumvent problems of vanishing and exploding gradients. Trabelsi et al. [59] propose complex convolutional networks and algorithms for complex batch-normalization, complex weight initialization, and complex activations.

Polynomial neural networks (PNNs). PNNs have been studied for several decades [28; 13; 55; 42; 37; 68]. However, many of the aforementioned PNNs do not scale well for high-dimensional data samples, such as those required for modern generative models. Chrysos et al. [5] introduce a high-degree polynomial generator, which approximates the distribution of high-dimensional data. The work was subsequently extended on a range of applications in the class of function approximators called Π-Nets, where Chrysos et al. [8] exhibit the increased expressivity of real-valued PNNs. Lastly, Zhang et al. [70] focus on approximating the operations such as sigmoid, ReLU, and max pooling with polynomial operations. Differently from previous works, we explicitly propose polynomial expansions on the complex field.

B. Detailed notation

Symbols of variables: As a reminder, real-valued matrices (vectors) are symbolized by uppercase (lowercase) boldface letters, e.g., $\boldsymbol{Y}$, $\boldsymbol{y}$. Tensors are the multidimensional equivalent of matrices. Real-valued tensors are symbolized by calligraphic letters, e.g., $\boldsymbol{\mathcal{Y}}$. All complex-valued variables are symbolized with wide tilde, e.g., $\widetilde{\boldsymbol{y}}$, $\widetilde{\boldsymbol{Y}}$, $\widetilde{\boldsymbol{\mathcal{Y}}}$. Khatri-Rao product and Hadamard product are denoted by $\odot$ and $*$, respectively.

Complex-valued transpose: In this paper, all transposes for complex-valued matrices are trivial transposes (the same as in real-valued field) without calculating complex conjugate.

Matrix products: The *Khatri-Rao* product of two matrices $\widetilde{\boldsymbol{A}} \in \mathbb{C}^{I\times N}$ and $\widetilde{\boldsymbol{C}} \in \mathbb{C}^{J\times N}$ is denoted by $\widetilde{\boldsymbol{A}} \odot \widetilde{\boldsymbol{C}}$. The *Khatri-Rao* product of multiple matrices $\{\widetilde{\boldsymbol{A}}_{[m]} \in \mathbb{C}^{I_m\times N}\}_{m=1}^{M}$ is denoted by $\widetilde{\boldsymbol{A}}_{[1]} \odot \widetilde{\boldsymbol{A}}_{[2]} \odot \cdots \odot \widetilde{\boldsymbol{A}}_{[M]} \doteq \bigodot_{m=1}^{M} \widetilde{\boldsymbol{A}}_{[m]}$. The *Hadamard* product of $\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{C}} \in \mathbb{C}^{I\times N}$ is denoted by $\widetilde{\boldsymbol{A}} * \widetilde{\boldsymbol{C}}$, where the (i,j) element is equal to $a_{(i,j)}b_{(i,j)}$.

Tensors: Consider an M^{th} order tensor: $\widetilde{\boldsymbol{\mathcal{Y}}} \in \mathbb{C}^{J_1\times J_2\times\cdots\times J_{m-1}\times J_m\times J_{m+1}\times\cdots\times J_M}$. We address each element by $(\widetilde{\boldsymbol{\mathcal{Y}}})_{j_1,j_2,\ldots,j_M} \doteq y_{j_1,j_2,\ldots,j_M}$. The *mode-$m$* vector product between the tensor $\widetilde{\boldsymbol{\mathcal{Y}}}$ and a vector $\widetilde{\boldsymbol{u}} \in \mathbb{C}^{J_m}$ is denoted by $\widetilde{\boldsymbol{\mathcal{Y}}} \times_m \widetilde{\boldsymbol{u}} \in \mathbb{C}^{J_1\times J_2\times\cdots\times J_{m-1}\times J_{m+1}\times\cdots\times J_M}$, yielding an $(M-1)^{\text{th}}$ order tensor:

$$\left(\widetilde{\boldsymbol{\mathcal{Y}}} \times_m \widetilde{\boldsymbol{u}}\right)_{j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \sum_{j_m=1}^{J_m} \widetilde{y}_{j_1,j_2,\ldots,j_M}\widetilde{u}_{j_m}.$$

The *mode-m* vector product of a complex-valued tensor and multiple complex-valued vectors is denoted as:

$$\widetilde{\boldsymbol{\mathcal{Y}}} \times_1 \widetilde{\boldsymbol{u}}^{(1)} \times_2 \widetilde{\boldsymbol{u}}^{(2)} \times_3 \cdots \times_M \widetilde{\boldsymbol{u}}^{(M)} = \widetilde{\boldsymbol{\mathcal{Y}}} \prod_{m=1}^{M} \times_m \widetilde{\boldsymbol{u}}^{(m)}.$$

As a generalization of *CANDECOMP/PARAFAC (CP) decomposition* [34] in complex field, complex-valued tensor could be decomposed into a sum of component rank-one tensors. The rank-R CP decomposition of an M^{th}-order tensor $\widetilde{\boldsymbol{\mathcal{Y}}}$ is denoted by:

$$\widetilde{\boldsymbol{\mathcal{Y}}} \doteq [\![\widetilde{\boldsymbol{U}}_{[1]}, \widetilde{\boldsymbol{U}}_{[2]}, \ldots, \widetilde{\boldsymbol{U}}_{[M]}]\!] = \sum_{r=1}^{R} \widetilde{\boldsymbol{U}}_r^{(1)} \circ \widetilde{\boldsymbol{U}}_r^{(2)} \circ \cdots \circ \widetilde{\boldsymbol{U}}_r^{(M)}, \tag{3}$$

where $\circ$ is the vector outer product. The factor matrices $\left\{\widetilde{\boldsymbol{U}}_{[m]} = [\widetilde{\boldsymbol{u}}_1^{(m)}, \widetilde{\boldsymbol{u}}_2^{(m)}, \cdots, \widetilde{\boldsymbol{u}}_R^{(m)}] \in \mathbb{C}^{I_m\times R}\right\}_{m=1}^{M}$ collect the vectors from the rank-one components. Particularly, we can express the CP decomposition in matrix form:

$$\widetilde{\boldsymbol{W}}_{(1)} \doteq \widetilde{\boldsymbol{U}}_{[1]}\left(\bigodot_{m=M}^{2} \widetilde{\boldsymbol{U}}_{[m]}\right)^T. \tag{4}$$

Lemma B.1. *Given the sets of real-valued matrices $\{\boldsymbol{A}_\nu \in \mathbb{R}^{I_\nu\times K}\}_{\nu=1}^{N}$ and $\{\boldsymbol{C}_\nu \in \mathbb{R}^{I_\nu\times L}\}_{\nu=1}^{N}$, the following equality holds:*

$$(\bigodot_{\nu=1}^{N} \boldsymbol{A}_\nu)^T \cdot (\bigodot_{\nu=1}^{N} \boldsymbol{C}_\nu) = (\boldsymbol{A}_1^T \cdot \boldsymbol{C}_1) * \ldots * (\boldsymbol{A}_N^T \cdot \boldsymbol{C}_N). \tag{5}$$

The proof can be found in the appendix of Chrysos et al. [5].

Lemma B.2. *Given the sets of complex-valued matrices $\{\widetilde{\boldsymbol{A}}_\nu \in \mathbb{C}^{I_\nu\times K}\}_{\nu=1}^{N}$ and $\{\widetilde{\boldsymbol{C}}_\nu \in \mathbb{C}^{I_\nu\times L}\}_{\nu=1}^{N}$, the following equality holds:*

$$(\bigodot_{\nu=1}^{N} \widetilde{\boldsymbol{A}}_\nu)^T \cdot (\bigodot_{\nu=1}^{N} \widetilde{\boldsymbol{C}}_\nu) = (\widetilde{\boldsymbol{A}}_1^T \cdot \widetilde{\boldsymbol{C}}_1) * \ldots * (\widetilde{\boldsymbol{A}}_N^T \cdot \widetilde{\boldsymbol{C}}_N). \tag{6}$$

The proof is the same as that of Lemma B.1 except the field ($\mathbb{R}$ or $\mathbb{C}$) of the matrices.

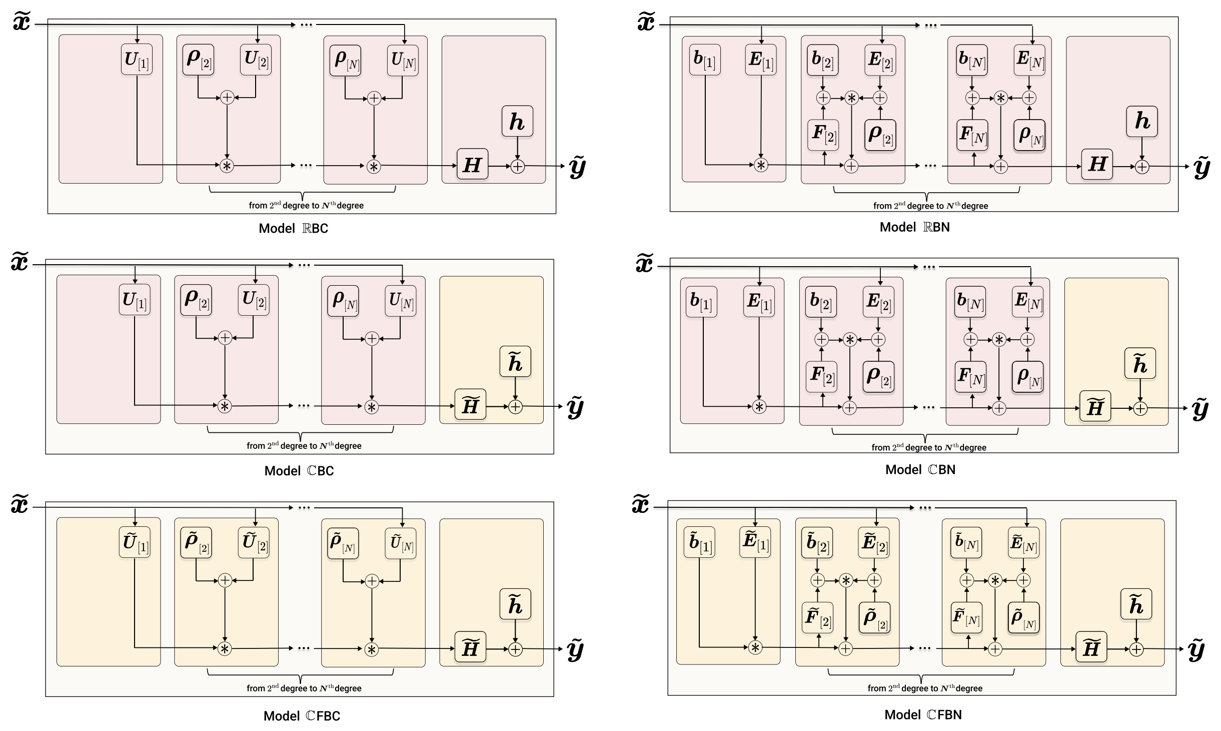


Figure 4. Schematic summary of our proposed models. All learnable parameters inside pink blocks (yellow blocks) are real-valued (complex-valued). We observe that ℂBC (ℂBN) reduces the number of parameters of ℂFBC (ℂFBN) by changing the field ($\mathbb{C} \rightarrow \mathbb{R}$) of the parameters from 1st degree to Nth degree.

C. Model variants

This section introduces different variants of our proposed framework. The dimension and the field (ℝ or ℂ) of learnable parameters in all proposed models are summarized in Table 6 and the corresponding schematics are depicted in Figure 4.

C.1. APOLLO with complex-valued coefficients

Our goal is to learn an N^{th} degree polynomial expansion with respect to the input $\widetilde{\boldsymbol{x}} \in \mathbb{C}^d$ with an $o-$dimensional output $\widetilde{\boldsymbol{y}}$ as follows:

$$\widetilde{\boldsymbol{y}} = \sum_{n=1}^{N} \left(\widetilde{\boldsymbol{\mathcal{W}}}^{[n]} \prod_{j=2}^{n+1} \times_j \widetilde{\boldsymbol{x}} \right) + \widetilde{\boldsymbol{h}}, \tag{7}$$

where $\left\{ \widetilde{\boldsymbol{\mathcal{W}}}^{[n]} \in \mathbb{C}^{o \times \prod_{m=1}^{n} \times_m d} \right\}_{n=1}^{N}$ and $\widetilde{\boldsymbol{h}} \in \mathbb{C}^o$ are learnable parameters. The drawback of Equation (7) is that it requires an exponential number of parameters for high-degree polynomial expansions, which is not scalable. Instead, we will apply coupled CP decomposition with factor sharing to reduce the learnable parameters [34; 8]. Firstly, we introduce polynomial expansions with complex-valued input variables and complex-valued coefficients, where the real and the imaginary coefficients are modelled with independent variables. Sequentially, we make additional assumptions that can further reduce the parameters and lead to efficient implementations. We illustrate below the derivation of ℂFC for third-degree expansion, while we stress out that the recursive formulation can be applied for *any arbitrary degree polynomial expansion*.

ℂ**FC (Fully coupled decomposition)** Firstly, we assume all the weight tensors $\{\widetilde{\boldsymbol{\mathcal{W}}}^{[n]}\}_{n=1}^{N}$ of Equation (7) are jointly factorized by a coupled CP decomposition to share factors between different degrees. For a third-degree expansion that becomes:

- First degree: $\widetilde{\boldsymbol{W}}^{[1]} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{[1]}^{T}$.

Table 6. Table with detailed symbol reference per model.

Symbol	Dimension(s)	Definition
$\boldsymbol{H}, \boldsymbol{h}, \boldsymbol{U}_{[n]}, \boldsymbol{\rho}_{[n]}$	$\mathbb{R}^{o\times k}, \mathbb{R}^{o}, \mathbb{R}^{d\times k}, \mathbb{R}^{k}$	Parameters in $\mathbb{R}$BC
$\boldsymbol{H}, \boldsymbol{h}, \boldsymbol{E}_{[n]}, \boldsymbol{F}_{[n]}, \boldsymbol{b}_{[n]}, \boldsymbol{\rho}_{[n]}$	$\mathbb{R}^{o\times k}, \mathbb{R}^{o}, \mathbb{R}^{d\times k}, \mathbb{R}^{k\times k}, \mathbb{R}^{k}, \mathbb{R}^{k}$	Parameters in $\mathbb{R}$BN
$\widetilde{\boldsymbol{H}}, \widetilde{\boldsymbol{h}}, \boldsymbol{U}_{[n]}, \boldsymbol{\rho}_{[n]}$	$\mathbb{C}^{o\times k}, \mathbb{C}^{o}, \mathbb{R}^{d\times k}, \mathbb{R}^{k}$	Parameters in $\mathbb{C}$BC
$\widetilde{\boldsymbol{H}}, \widetilde{\boldsymbol{h}}, \boldsymbol{E}_{[n]}, \boldsymbol{F}_{[n]}, \boldsymbol{b}_{[n]}, \boldsymbol{\rho}_{[n]}$	$\mathbb{C}^{o\times k}, \mathbb{C}^{o}, \mathbb{R}^{d\times k}, \mathbb{R}^{k\times k}, \mathbb{R}^{k}, \mathbb{R}^{k}$	Parameters in $\mathbb{C}$BN
$\widetilde{\boldsymbol{H}}, \widetilde{\boldsymbol{h}}, \widetilde{\boldsymbol{U}}_{[n]}, \widetilde{\boldsymbol{\rho}}_{[n]}$	$\mathbb{C}^{o\times k}, \mathbb{C}^{o}, \mathbb{C}^{d\times k}, \mathbb{C}^{k}$	Parameters in $\mathbb{C}$FBC
$\widetilde{\boldsymbol{H}}, \widetilde{\boldsymbol{h}}, \widetilde{\boldsymbol{E}}_{[n]}, \widetilde{\boldsymbol{F}}_{[n]}, \widetilde{\boldsymbol{b}}_{[n]}, \widetilde{\boldsymbol{\rho}}_{[n]}$	$\mathbb{C}^{o\times k}, \mathbb{C}^{o}, \mathbb{C}^{d\times k}, \mathbb{C}^{k\times k}, \mathbb{C}^{k}, \mathbb{C}^{k}$	Parameters in $\mathbb{C}$FBN

- Second degree: $\widetilde{\boldsymbol{W}}^{[2]}_{(1)} = \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[3]} \odot \widetilde{\boldsymbol{U}}_{[1]})^T + \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[2]} \odot \widetilde{\boldsymbol{U}}_{[1]})^T$.
- Third degree: $\widetilde{\boldsymbol{W}}^{[3]}_{(1)} = \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[3]} \odot \widetilde{\boldsymbol{U}}_{[2]} \odot \widetilde{\boldsymbol{U}}_{[1]})^T$.

By leveraging the aforementioned factorization and utilizing simple algebra, a simple recursive form can be retained (which is also generalizable to an N^{th} degree polynomial) as we prove in Appendix D.1:

$$\widetilde{\boldsymbol{y}}_n = \left(\widetilde{\boldsymbol{U}}^T_{[n]}\widetilde{\boldsymbol{x}}\right) * \widetilde{\boldsymbol{y}}_{n-1} + \widetilde{\boldsymbol{y}}_{n-1}, \tag{8}$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = \widetilde{\boldsymbol{U}}^T_{[1]}\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$, where $\widetilde{\boldsymbol{U}}_{[n]} \in \mathbb{C}^{d\times k}, \widetilde{\boldsymbol{H}} \in \mathbb{C}^{o\times k}, \widetilde{\boldsymbol{h}} \in \mathbb{C}^{o}$.

Intuitively, each recursive term increases the degree of expansion by one, using three fundamental operations: a) an affine transformation with the input, i.e., $\widetilde{\boldsymbol{U}}^T_{[n]}\widetilde{\boldsymbol{x}}$, b) an element-wise product with the previous term, c) a skip connection with the previous term. Those three operations are easy to implement and enable any arbitrary degree of expansion to be achieved.

$\mathbb{C}$FBC (Fully coupled decomposition with bias) In this section, we develop a new decomposition to allow the existence of bias term $\widetilde{\boldsymbol{\rho}}_{[n]} \in \mathbb{C}^k$ compared with the previous model, which can increase the expressivity of model in practice. The recursive relation is as follows (proof in Appendix D.2)

$$\widetilde{\boldsymbol{y}}_n = (\widetilde{\boldsymbol{U}}^T_{[n]}\widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{\rho}}_{[n]}) * \widetilde{\boldsymbol{y}}_{n-1}\ , \tag{9}$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = \widetilde{\boldsymbol{U}}^T_{[1]}\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$.

$\mathbb{C}$BC (Coupled decomposition with bias) The model above assumes different parameters for the real and the imaginary part, however, we could also perform sharing of certain factors between the real and the imaginary part to reduce the learnable parameters. Even though multiple sharing of factors could be implemented, we have assumed that every factor is shared apart from the factor corresponding to the unfolding dimension. That is, all terms between the real and the imaginary coefficients are shared apart from the matrix $\widetilde{\boldsymbol{H}}$. The following recursive relationship between different degrees is obtained: (proved in Appendix D.3):

$$\widetilde{\boldsymbol{y}}_n = (\boldsymbol{U}^T_{[n]}\widetilde{\boldsymbol{x}} + \boldsymbol{\rho}_{[n]}) * \widetilde{\boldsymbol{y}}_{n-1}\ , \tag{10}$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = \boldsymbol{U}^T_{[1]}\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$. Compared with the recursive formulation Equation (9) in the previous decomposition, we observe that Equation (10) reduces the number of parameters by converting $\widetilde{\boldsymbol{U}}_{[n]} \in \mathbb{C}^{d\times k}$ to $\boldsymbol{U}_{[n]} \in \mathbb{R}^{d\times k}$ and $\widetilde{\boldsymbol{\rho}}_{[n]} \in \mathbb{C}^k$ to $\boldsymbol{\rho}_{[n]} \in \mathbb{R}^k$ among each degree.

$\mathbb{C}$BN (Nested decomposition with bias) Unlike $\mathbb{C}$BC, we apply a joint hierarchical decomposition instead of separating the interactions between different degrees. The learnable hyper-parameters $\boldsymbol{\beta}_{[n]} \in \mathbb{R}^{\omega}$ for $n = 1, \dots, N$, are introduced as scaling factors and Equation (7) becomes:

$$\widetilde{\boldsymbol{y}} = \sum_{n=1}^{N}\left(\widetilde{\boldsymbol{\mathcal{W}}}^{[n]} \times_2 \boldsymbol{\beta}_{[N+1-n]} \prod_{j=3}^{n+2} \times_j \widetilde{\boldsymbol{x}}\right) + \widetilde{\boldsymbol{h}}. \tag{11}$$

The interactions between different degrees are learned through a joint hierarchical decomposition on the polynomial parameters.

- First-degree parameters : $\widetilde{\boldsymbol{W}}_{(1)}^{[1]} = \widetilde{\boldsymbol{H}}(\boldsymbol{E}_{[3]} \odot \boldsymbol{B}_{[3]})^T$.
- Second-degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[2]} = \widetilde{\boldsymbol{H}}\left\{\boldsymbol{E}_{[3]} \odot \left[\left(\boldsymbol{E}_{[2]} \odot \boldsymbol{B}_{[2]}\right)\boldsymbol{F}_{[3]}\right]\right\}^T$.
- Third-degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[3]} = \widetilde{\boldsymbol{H}}\left\{\boldsymbol{E}_{[3]} \odot \left[\left(\boldsymbol{E}_{[2]} \odot \left\{\left(\boldsymbol{E}_{[1]} \odot \boldsymbol{B}_{[1]}\right)\boldsymbol{F}_{[2]}\right\}\right)\boldsymbol{F}_{[3]}\right]\right\}^T$,

where $\widetilde{\boldsymbol{H}} \in \mathbb{C}^{o\times k}$, $\boldsymbol{E}_{[n]} \in \mathbb{R}^{d\times k}$ $\boldsymbol{B}_{[n]} \in \mathbb{R}^{w\times k}$ $\boldsymbol{F}_{[n]} \in \mathbb{R}^{k\times k}$ for $n = 1, \dots, N$. Similarly, we can obtain the recursive form for N^{th} degree expansion:

$$\widetilde{\boldsymbol{y}}_n = \left(\boldsymbol{E}_{[n]}^T\widetilde{\boldsymbol{x}}\right) * \left(\boldsymbol{F}_{[n]}^T\widetilde{\boldsymbol{y}}_{n-1} + \boldsymbol{b}_{[n]}\right),$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = (\boldsymbol{E}_{[1]}^T\widetilde{\boldsymbol{x}}) * \left(\boldsymbol{b}_{[1]}\right)$, $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$, where we denote by $\boldsymbol{b}_{[n]} = \boldsymbol{B}_{[n]}^T\boldsymbol{\beta}_{[n]}$ for $n = 1, \dots, N$. Motivated by the skip connections in ResNet [23], we embed a shortcut connection into the relationship. In addition, we add a bias term $\boldsymbol{\rho}_{[n]} \in \mathbb{R}^k$, which is similar to $\mathbb{C}$BC. The final recursive relationship:

$$\widetilde{\boldsymbol{y}}_n = \left(\boldsymbol{E}_{[n]}^T\widetilde{\boldsymbol{x}} + \boldsymbol{\rho}_{[n]}\right) * \left(\boldsymbol{F}_{[n]}^T\widetilde{\boldsymbol{y}}_{n-1} + \boldsymbol{b}_{[n]}\right) + \widetilde{\boldsymbol{y}}_{n-1}, \tag{12}$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = (\boldsymbol{E}_{[1]}^T\widetilde{\boldsymbol{x}}) * \left(\boldsymbol{b}_{[1]}\right)$ and $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$. Note that all learnable parameters in all degrees are real-valued except in the highest degree.

$\mathbb{C}$FBN (Fully nested decomposition with bias) When all the parameters in $\mathbb{C}$BN are complex-valued, we have the following recursive relationship:

$$\widetilde{\boldsymbol{y}}_n = \left(\widetilde{\boldsymbol{E}}_{[n]}^T\widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{\rho}}_{[n]}\right) * \left(\widetilde{\boldsymbol{F}}_{[n]}^T\widetilde{\boldsymbol{y}}_{n-1} + \widetilde{\boldsymbol{b}}_{[n]}\right) + \widetilde{\boldsymbol{y}}_{n-1}, \tag{13}$$

for $n = 2, \dots, N$ with $\widetilde{\boldsymbol{y}}_1 = (\widetilde{\boldsymbol{E}}_{[1]}^T\widetilde{\boldsymbol{x}}) * \left(\widetilde{\boldsymbol{b}}_{[1]}\right)$, $\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{y}}_N + \widetilde{\boldsymbol{h}}$, where we denote by $\widetilde{\boldsymbol{b}}_{[n]} = \widetilde{\boldsymbol{B}}_{[n]}^T\widetilde{\boldsymbol{\beta}}_{[n]}$ for $n = 1, \dots, N$.

C.2. APOLLO with real-valued coefficients

The aforementioned models assume complex-valued coefficients. However we can replace the coefficients in Equation (7) with its real-valued counterpart and obtain the following polynomial:

$$\widetilde{\boldsymbol{Y}} = \sum_{n=1}^{N}\left(\boldsymbol{\mathcal{W}}^{[n]}\prod_{j=2}^{n+1}\times_j\widetilde{\boldsymbol{x}}\right) + \boldsymbol{h}, \tag{14}$$

where $\left\{\boldsymbol{\mathcal{W}}^{[n]} \in \mathbb{R}^{o\times\prod_{m=1}^{n}\times_m d}\right\}_{n=1}^{N}$ and $\boldsymbol{h} \in \mathbb{R}^o$ are learnable parameters. Similarly, we can apply different decomposition techniques to reduce the parameters and obtain the recursive relationship between different degrees. Since the only difference from the models given in the main paper is the field ($\mathbb{R}$ or $\mathbb{C}$) of the learnable parameters, we skip the derivation and the recursive relationship. If we convert all the parameters in $\mathbb{C}$BC from complex-valued to real-valued, we can obtain $\mathbb{R}$BC. If we convert all the parameters in $\mathbb{C}$BN from complex-valued to real-valued, we can obtain $\mathbb{R}$BN.

C.3. In-depth difference from real-valued polynomial networks

The family of Π-Nets bears resemblance with the proposed APOLLO, however those two differ substantially in the following ways:

- The family of Π-Nets was designed for real-valued inputs and outputs with all the learnable parameters being real-valued as well. On the contrary, the proposed APOLLO enables using complex-valued inputs or outputs.
- The motivation for designing the two models differs: in APOLLO the goal is to construct an efficient method for time-frequency representations frequently met in audio-related tasks. On the contrary, Π-Nets is (mostly) focused on image-related tasks or non-euclidean meshes. In Section 2.3, we design several architectures and technique to adapt our decomposition to audio generation. As shown in Table 1, our APOLLO greatly outperforms Π-Nets, which are trivially adapted to audio generation.

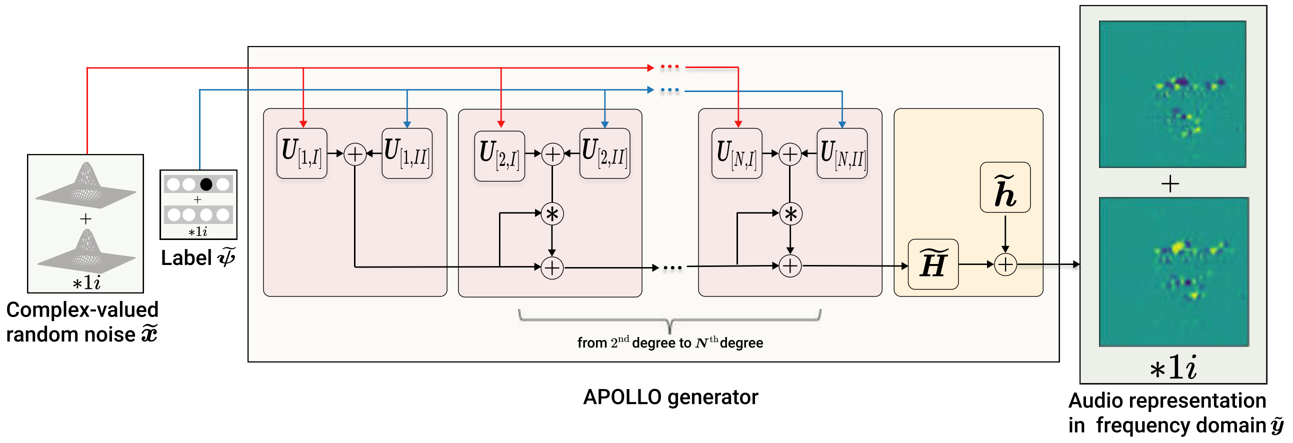


Figure 5. A schematic of the proposed conditional APOLLO that receives a complex-valued noise and class-label. The label can be treated as a complex-valued one-hot vector with zero imaginary part. The generator in the schematic corresponds to Equation (2). All learnable parameters inside pink blocks (yellow blocks) are real-valued (complex-valued).

- Beyond the technical tensor decompositions, a significant contribution in our case is to highlight which decompositions work well in the complex field. Even though there are several combinations of recursive formulations that could be designed, we showcase how to construct efficient recursive formulations in a principled way.

- Π-Nets express a polynomial expansion of one variable, which presents an issue in case of multiple input variables being available. Particularly, in the case of the variables representing different type of data, e.g., vectors and tensors, it might be more convenient to not concatenate them. This was similarly reported in Chrysos et al. [6], and we follow that model in conditional generation. Concretely, in the image to audio experiment, Π-Nets needs to vectorize the input image to concatenate it with the noise, which would destroy the spatial correlations while in APOLLO, we utilize low-degree polynomial to capture the representation and then use a high-degree polynomial to learn the correlation.

D. Model derivations

D.1. Derivations for ℂFC

By leveraging the factorizations, i.e.,

- First degree parameters: $\widetilde{W}^{[1]} = \widetilde{H}\widetilde{U}_{[1]}^T$.
- Second degree parameters: $\widetilde{W}^{[2]}_{(1)} = \widetilde{H}(\widetilde{U}_{[3]} \odot \widetilde{U}_{[1]})^T + \widetilde{H}(\widetilde{U}_{[2]} \odot \widetilde{U}_{[1]})^T$.
- Third degree parameters: $\widetilde{W}^{[3]}_{(1)} = \widetilde{H}(\widetilde{U}_{[3]} \odot \widetilde{U}_{[2]} \odot \widetilde{U}_{[1]})^T$,

the third degree expansion of Equation (7) is expressed as:

$$\begin{aligned}\widetilde{y} = \widetilde{h} + \widetilde{H}\widetilde{U}_{[1]}^T\widetilde{x} + \widetilde{H}\left(\widetilde{U}_{[3]} \odot \widetilde{U}_{[1]}\right)^T(\widetilde{x}\odot\widetilde{x}) + \widetilde{H}\left(\widetilde{U}_{[2]} \odot \widetilde{U}_{[1]}\right)^T(\widetilde{x}\odot\widetilde{x}) \\ + \widetilde{H}\left(\widetilde{U}_{[3]} \odot \widetilde{U}_{[2]} \odot \widetilde{U}_{[1]}\right)^T(\widetilde{x}\odot\widetilde{x}\odot\widetilde{x}).\end{aligned} \tag{15}$$

Applying Lemma B.2 on (Equation (15)), we obtain:

$$\widetilde{y} = \widetilde{h} + \widetilde{H}\left\{(\widetilde{U}_{[3]}^T\widetilde{x}) * \left[\left(\widetilde{U}_{[2]}^T\widetilde{x}\right) * \left(\widetilde{U}_{[1]}^T\widetilde{x}\right) + \ \widetilde{U}_{[1]}^T\widetilde{x}\right] + \left(\widetilde{U}_{[2]}^T\widetilde{x}\right) * \left(\widetilde{U}_{[1]}^T\widetilde{x}\right) + \widetilde{U}_{[1]}^T\widetilde{x}\right\}, \tag{16}$$

which could be expressed as the recursive relationship Equation (8).

D.2. Derivations for ℂFBC

A polynomial expansion of order $N \in \mathbb{N}$ with output $\widetilde{\boldsymbol{y}} \in \mathbb{C}^o$ has the form:

$$\widetilde{\boldsymbol{y}} = \sum_{n=1}^{N} \underbrace{\sum_{r_1=2}^{n+1} \sum_{r_2=r_1+1}^{n+2} \cdots \sum_{r_{N\text{-}n}=r_{N\text{-}n\text{-}1}+1}^{N}}_{(N-n)\text{ sums}} \left(\widetilde{\boldsymbol{\mathcal{W}}}^{[n,r_1,r_2,\ldots,r_{N\text{-}n}]} \prod_{j=2}^{n+1} \times_j \widetilde{\boldsymbol{x}} \prod_{\tau=n+2}^{N+1} \times_\tau \widetilde{\boldsymbol{\rho}}_{[r_{\tau-n-1}]} \right) + \widetilde{\boldsymbol{h}}\,, \tag{17}$$

where we add the scaling parameters $\{\widetilde{\boldsymbol{\rho}}_{[n]} \in \mathbb{C}^k\}_{n=1}^N$. Then, the tensors $\{\widetilde{\boldsymbol{\mathcal{W}}}^{[n,r_1,r_2,\ldots,r_{N\text{-}n}]}\}_{n=1}^N$ and $\{\widetilde{\boldsymbol{\rho}}_{[n]}\}_{n=1}^N$ are the learnable parameters. Similarly, to previous cases, the learnable parameters are increasing exponentially, so a standard decomposition will be applied to reduce them.

Let us rewrite Equation (17) for a third-degree polynomial, i.e. $N = 3$, to illustrate the decomposition and then we provide the recursive relationship that can be used for an arbitrary degree of expansion. The third degree expansion of Equation (17) has the following form:

$$\begin{aligned}\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{\mathcal{W}}}^{[1,2,3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{\rho}}_{[2]} \times_4 \widetilde{\boldsymbol{\rho}}_{[3]} + \widetilde{\boldsymbol{\mathcal{W}}}^{[2,2]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \widetilde{\boldsymbol{\rho}}_{[2]} + \\ \widetilde{\boldsymbol{\mathcal{W}}}^{[2,3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \widetilde{\boldsymbol{\rho}}_{[3]} + \widetilde{\boldsymbol{\mathcal{W}}}^{[3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{h}}\,.\end{aligned} \tag{18}$$

Then, based on Equation (18), we can apply a tailored coupled CP decomposition with the following form (in matrix form):

- First degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[1,2,3]} : \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[1]} \odot \boldsymbol{I} \odot \boldsymbol{I})^T$.
- Second degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[2,2]} : \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[1]} \odot \widetilde{\boldsymbol{U}}_{[3]} \odot \boldsymbol{I})^T$.
- Second degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[2,3]} : \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[1]} \odot \widetilde{\boldsymbol{U}}_{[2]} \odot \boldsymbol{I})^T$.
- Third degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[3]} : \widetilde{\boldsymbol{H}}(\widetilde{\boldsymbol{U}}_{[1]} \odot \widetilde{\boldsymbol{U}}_{[2]} \odot \widetilde{\boldsymbol{U}}_{[3]})^T$,

where $\boldsymbol{I}$ denotes the identity matrix, the parameters $\widetilde{\boldsymbol{H}} \in \mathbb{C}^{o\times k}, \widetilde{\boldsymbol{U}}_{[n]} \in \mathbb{C}^{d\times k}$ for $n = 1, 2, 3$ are learnable. By plugging in the aforementioned factorization into Equation (18) and applying Lemma B.2, we obtain:

$$\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}} \left((\widetilde{\boldsymbol{U}}_{[3]}^T \widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{\rho}}_{[3]}) * (\widetilde{\boldsymbol{U}}_{[2]}^T \widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{\rho}}_{[2]}) * (\widetilde{\boldsymbol{U}}_{[1]}^T \widetilde{\boldsymbol{x}}) \right) + \widetilde{\boldsymbol{h}}\,. \tag{19}$$

We can generalize the equation above to a general recursive formulation, i.e., Equation (9).

D.3. Derivations for ℂBC

Compared to ℂFBC, the scaling parameters in ℂBC is real-valued instead of complex-valued, i.e., $\{\boldsymbol{\rho}_{[n]} \in \mathbb{R}^k\}_{n=1}^N$. The third degree expansion has the following form:

$$\begin{aligned}\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{\mathcal{W}}}^{[1,2,3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \boldsymbol{\rho}_{[2]} \times_4 \boldsymbol{\rho}_{[3]} + \widetilde{\boldsymbol{\mathcal{W}}}^{[2,2]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \boldsymbol{\rho}_{[2]} + \\ \widetilde{\boldsymbol{\mathcal{W}}}^{[2,3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \widetilde{\boldsymbol{\rho}}_{[3]} + \widetilde{\boldsymbol{\mathcal{W}}}^{[3]} \times_2 \widetilde{\boldsymbol{x}} \times_3 \widetilde{\boldsymbol{x}} \times_4 \widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{h}}\,.\end{aligned} \tag{20}$$

We apply a tailored coupled CP decomposition with the following form, (note that $\boldsymbol{U}_{[n]}$ is real-valued):

- First degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[1,2,3]} : \widetilde{\boldsymbol{H}}(\boldsymbol{U}_{[1]} \odot \boldsymbol{I} \odot \boldsymbol{I})^T$.
- Second degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[2,2]} : \widetilde{\boldsymbol{H}}(\boldsymbol{U}_{[1]} \odot \boldsymbol{U}_{[3]} \odot \boldsymbol{I})^T$.
- Second degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[2,3]} : \widetilde{\boldsymbol{H}}(\boldsymbol{U}_{[1]} \odot \boldsymbol{U}_{[2]} \odot \boldsymbol{I})^T$.

- Third degree parameters: $\widetilde{\boldsymbol{W}}_{(1)}^{[3]} : \widetilde{\boldsymbol{H}}(\boldsymbol{U}_{[1]} \odot \boldsymbol{U}_{[2]} \odot \boldsymbol{U}_{[3]})^T$,

where $\boldsymbol{I}$ denotes the identity matrix, the parameters $\widetilde{\boldsymbol{H}} \in \mathbb{C}^{o\times k}, \boldsymbol{U}_{[n]} \in \mathbb{R}^{d\times k}$ for $n = 1, 2, 3$ are learnable. By plugging in the aforementioned factorization into Equation (20) and applying Lemma B.2, we obtain:

$$\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{H}} \left((\boldsymbol{U}_{[3]}^T \widetilde{\boldsymbol{x}} + \boldsymbol{\rho}_{[3]}) * (\boldsymbol{U}_{[2]}^T \widetilde{\boldsymbol{x}} + \boldsymbol{\rho}_{[2]}) * (\boldsymbol{U}_{[1]}^T \widetilde{\boldsymbol{x}}) \right) + \widetilde{\boldsymbol{h}} \,. \tag{21}$$

We can generalize the equation above to a general recursive formulation, i.e., Equation (10).

E. Experimental details

This section presents the experimental details, including the description of dataset in Appendix E.1, the quantitative metrics in Appendix E.2, and the experimental setup in Appendix E.3, E.4, E.5. We train our model on **a single NVIDIA** 2080 **Ti GPU**.

E.1. Datasets

Firstly, we conduct experiments of unconditional generation on three datasets used in Donahue et al. [9]: (a) Speech Commands Zero Through Nine (SC09) [67] that consists of spoken digits “zero” through “nine.” There are $18,620$ audio clips in the training set of SC09 and 2552 audio clips in the testing set. The duration of each sample is around 1 second. The total length of the training set is 5.3 hours. (b) Piano, a music dataset that contains 0.3 hours’ Bach compositions. (c) Drum, a music dataset that contains 0.7 hours’ drum samples in the training set. There are 2350 audio clips in the training set and 224 audio clips in the testing set. The duration of each sample is around 1 second. (d) We conduct the experiment of conditional generation on the same aforementioned SC09 dataset. (e) In addition, we also investigate a large music dataset, called NSynth, that consists $300,000$ musical notes labelled with pitch, instrument, acoustic qualities, and velocity [10]. Each sample lasts for 4 seconds. We use the same subset and the same test/train split as in Engel et al. [11]. The labels are the pitches ranging from MIDI 24 ($\sim$32Hz) to MIDI 84 ($\sim$1000Hz). (f) Note that in the paper of SASHIMI, they use a different splitting for SC09 dataset where there are $31,158$ audio clips in the training set (8.7 hours in total). Thus, we train our model again in this training set to fairly compare with the baselines, as reported in Table 7.

E.2. Evaluation metrics in audio generation

- **Inception Score (IS).** Firstly introduced by Salimans et al. [53], IS is used to measure the quality and the diversity of the image generated by GAN. The fake samples are fed into a pretrained Inception Network V3 [57] to get the conditional label distribution. The IS is calculated by averaging the KL divergence between the conditional label distribution and its marginal distribution. The audio samples in SC09 dataset are converted to log spectrum representation with 16k sample rate, 8 ms stride, and 64 ms windows size. The frequency bin is further mapped into mel-scale ranging from 40 HZ to 7800 HZ. We use the same evaluation protocol (50k samples) and the same pre-trained classifier from the paper of WaveGAN [9]to calculate the IS for SC09 dataset.

- **Fréchet Inception Distance (FID)**. FID [24] characterizes the Fréchet distance of the intermediate layer’s feature from the pretrained Inception Network V3 between the real image and the generated image. For SC09 dataset, the evaluation protocol and the pre-trained classifier are the same as that in IS. For NSynth dataset, the audio is converted to log spectrum representation with 16k sample rate, 16 ms stride, and 128 ms windows size. The frequency bin is further mapped into mel-scale ranging from 0 Hz to 8000 Hz. we train a CNN-based pitch classifier to calculate the score.

- **Number of Statistically-Different Bins (NDB) and Jensen-Shannon Divergence (JSD)**. NDB and JSD are proposed by Richardson & Weiss [50] to measure the diversity between the generated samples and the real samples. The audio is converted to STFT and then mel-spectrogram, as illustrated in Figure 6. For SC09 (Piano, Drum, NSynth) dataset, the mel-scale is ranging from 40 Hz to 7800 Hz (0 HZ to 8000 HZ). The mel-spectrograms of the real samples are clustered by K-means into 50 clusters.

- Unless otherwise mentioned, all evaluation metrics in this paper are based on the aforementioned details. However, due to the popularity of audio synthesis, a variety of techniques have been proposed for evaluation. Specifically, in the comparison with non-adversarial methods, as shown in Table 7, the evaluation protocol follows the paper of SASHIMI

[14], which was released within the last few months. We use the same pretrained-model, number of samples (2048), and metrics (IS, Modified Inception Score [20], FID, AM Score [71]). **Modified Inception Score (MIS)**: MIS considers both intra-class and inter-class sample diversity. **AM Score**: The difference from IS is that AM Score considers the marginal target distribution of the training set.

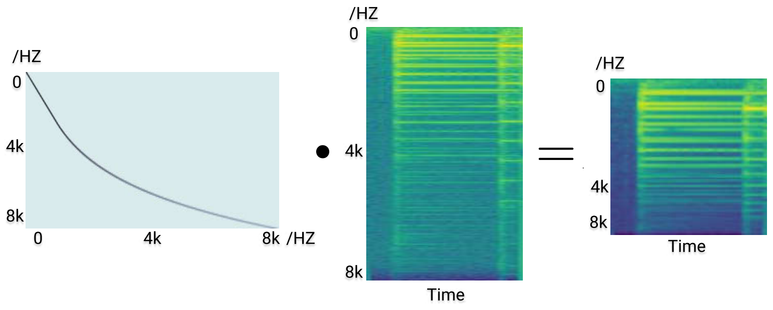


Figure 6. Before calculating JSD, NDB, IS, and FID, we need to project the magnitude of STFT (**Middle**) to human perceptual frequency scales (**Right**) through mel-scale projection matrix (**Left**). The spectrum in the figure are plotted in log-scale only for visualization purpose.

Table 7. Comparison with non-adversarial methods. Higher IS, MIS (lower FID, AM) indicate better performance. APOLLO largely improves upon the baselines. As a reminder, previous experiment on SC09 use the same protocol as in Donahue et al. [9] while in this experiment, the dataset splitting and the details on evaluation metrics are the same as in Goel et al. [14].

Unconditional audio generation on SC09 dataset					
Model	IS (↑)	FID (↓)	MIS (↑)	AM (↓)	# par (M)
Real data	8.33	0.02	257.6	0.19	-
SampleRNN [40]	1.71	8.96	3.02	1.76	35.0
WaveNet [43]	2.27	5.08	5.80	1.47	4.2
DiffWave [35]	5.26	1.92	51.21	0.68	24.1
SASHIMI [14]	5.94	1.42	69.17	0.59	23.0
APOLLO, Small	6.03	0.69	77.05	0.53	4.6
APOLLO	**6.43**	**0.45**	**105.82**	**0.45**	64.1

E.3. Experimental setup in unconditional audio generation

Each audio clip in all datasets is first down-sampled with 16kHz and padded or clipped to the fixed length of 16384. For the SC09 dataset and Drum dataset, we apply STFT with 8 ms stride, 16 ms windows size, and Hann window, which results in the resolution 129×128 (Frequency $\times$ Time). We use a resolution of 128×128 by truncating the Nyquist bin, i.e., bottom row. The APOLLO is implemented by the product of $\mathbb{R}$BN and $\mathbb{C}$FBN. Since the sounds in Piano dataset are not sparse in time domain, we choose 8 ms stride and 32 ms windows. Similarly, we drop the Nyquist bin and get the final solution 256×128 (Frequency $\times$ Time). We choose the product of $\mathbb{R}$BC and $\mathbb{C}$FBN. Note that the matrices used to multiply by the noise vector are implemented by dense layer while the matrices used to multiply by the time-frequency representation is implemented by convolution layer. The Hadamard product is implemented by element-wise multiplication. Unless mentioned otherwise, we assume that ReLU activation are inserted after each degree, which result in a piece-wise polynomial expansion and tanh activation after the final degree. For the hyperparameters, the factor of the gradient penalty is 10. In each step, we apply 5 updates of the discriminator and 1 update of the generator with learning rate 10^{-4}. The hyperparameters of ADAM optimizer [33] are $\beta_1 = 0.5$ and $\beta_2 = 0.9$. The base channel of the CNN is 64 in both generator and discriminator (16 for the 'Small' model). The batch size is 8. The model converges within 380k steps (2 days) in SC09 dataset and 100k steps on Piano dataset and Drum dataset.

E.4. Experimental setup in conditional audio generation

Firstly, we study the SC09 dataset, which has been used in the task of unconditional generation. The label is the class of digits. We also investigate a large music dataset, called NSynth, that consists 300,000 musical notes labelled with pitch, instrument, acoustic qualities, and velocity [10]. Each sample lasts for 4 seconds. We use the same subset and the same test/train split as in Engel et al. [11]. The labels are the pitches ranging from MIDI 24 ($\sim$32Hz) to MIDI 84 ($\sim$1000Hz). Figure 5 is a schematic illustration of applying Equation (2) on the conditional generator. Motivated by Miyato & Koyama [41], we use a projection discriminator to incorporate the label information. For SC09 dataset, other settings are the same as in unconditional generation. Our model converges within 3 days (450k steps). For NSynth dataset, we choose 16kHz sample rate and pad (or clip) each clip to the fixed length of 64000. Then we apply STFT with 16 ms stride, 128 ms windows size, and 75% overlap, which results in the resolution 1025×128 (Frequency $\times$ Time). After truncating the Nyquist bin, the final resolution is 1024×128. The APOLLO is implemented by the product of $\mathbb{R}$BN and $\mathbb{C}$FBN. The base channel of CNN in our model is 16 in this experiment. Other settings are the same as in conditional generation on SC09. Our model converges within 15 hours (60k Steps).

E.5. Experimental setup in multimodal generation

For each audio clip in the training set (18620 audio clips) of SC09, we align 2 images with the same digit from the training set (55000 images) of MNIST without replacement. In total, we create 18620×2 audio-image pairs as the training set. We sample images from the testing set of MNIST to evaluate the model. The metrics reported in Table 5 is calculated by the same pretrained classifier used in Table 1. The architecture of the generator is presented in Figure 2 in the main body, Firstly, we use two low-degree APOLLOs for the random noise and the image. Specifically, we use $\mathbb{R}$BN for the real-valued image and $\mathbb{C}$FBN for the complex-valued noise. Then, the conditional APOLLO, i.e., $\mathbb{C}$FBN, will receive the output of these two low-degree APOLLOs and generate the STFT of the audio. As for the discriminator, we resize the image with nearest-neighbor interpolation to the resolution 128×128 and concatenate it with the STFT of the audio at the input layer. The base channel of CNN is 16 in this experiment. Other settings are the same as in unconditional generation on SC09. Our model converges within one day (200k steps).

F. Ablation experiments

F.1. Model variants

We conduct an ablation study on different schemes proposed in this paper. The quantitative results are presented in Table 8, where all models are implemented using products of polynomials, i.e., several polynomials stacked sequentially. '$\mathbb{R}$BN + $\mathbb{C}$FBN' obtains the best result in IS, NDB, and JSD while '$\mathbb{R}$BC + $\mathbb{R}$BN' achieves the lowest FID. In practice, the model with full decomposition on complex-valued coefficients ('$\mathbb{C}$FBN+ $\mathbb{C}$FBN') requires much more parameters, while the improvement is not significant.

Table 8. Ablation experiment on different proposed schemes of APOLLO.

Unconditional audio generation on SC09 dataset					
Model	IS (↑)	FID (↓)	NDB (↓)	JSD (↓)	# par
Real data	8.01 ± 0.24	0.50	0.00 ± 0.00	0.011	-
$\mathbb{C}$BC+$\mathbb{C}$BN	6.74 ± 0.06	9.34	3.80 ± 0.74	0.039	46.0
$\mathbb{R}$BC+$\mathbb{C}$BN	6.98 ± 0.04	12.00	4.00 ± 0.00	0.040	45.9
$\mathbb{R}$BC+$\mathbb{R}$BN	6.79 ± 0.03	**8.03**	3.80 ± 1.32	0.041	45.9
$\mathbb{R}$BN+$\mathbb{R}$BN	6.71 ± 0.03	9.44	4.00 ± 0.00	0.038	45.9
$\mathbb{R}$BN+$\mathbb{C}$FBN	$\mathbf{7.25 \pm 0.05}$	8.15	$\mathbf{3.20 \pm 1.16}$	**0.029**	64.1
$\mathbb{C}$FBN+$\mathbb{C}$FBN	6.84 ± 0.02	8.87	4.20 ± 0.40	0.041	68.1

F.2. The field ($\mathbb{C}$ or $\mathbb{R}$) of the generator and the discriminator

Below, we conduct experiment with different fields of the generator and discriminator. We combine our APOLLO ($\mathbb{R}$BN+$\mathbb{C}$FBN), with real-valued discriminator and complex-valued discriminator, respectively. Additionally, Π-Nets is tested as a real-valued generator that concatenates the real part and the imaginary part of the STFT in two channels. The result in Table 9 illustrates that our model APOLLO with real-valued discriminator yields best IS and FID.

Table 9. Ablation experiment on the field ($\mathbb{C}$ or $\mathbb{R}$) of generator and discriminator (D). The results demonstrate that combining our complex-valued generator APOLLO with real-valued discriminator yields the best performance in practice.

Unconditional audio generation on SC09 dataset		
Model	IS (↑)	FID (↓)
Real data	8.01 ± 0.24	0.50
Π-Nets-real D	6.59 ± 0.03	13.01
APOLLO-real D	$\mathbf{7.25 \pm 0.05}$	**8.15**
APOLLO-complex D	6.74 ± 0.05	11.76

F.3. Removing activation functions on audio generation

In practice, we add ReLU activations function after all Hadamard products in the implementation, which converts the network into a piece-wise polynomial expansion. In this ablation experiment, we maintain the same setting as in previous experiment except for the generator. Particularly, the generator only contains a hyperbolic tangent in the output space for normalization as typically done in other GAN generators. There is no activation function between the layers. The result in Table 10 shows that both metrics deteriorate without using non-linear activation function. We use the pretrained classifier to calculate the accuracy of the generated samples given the label inputs. Figure 7 shows that the model converges slower and obtains lower accuracy notably after removing non-linear activation functions.

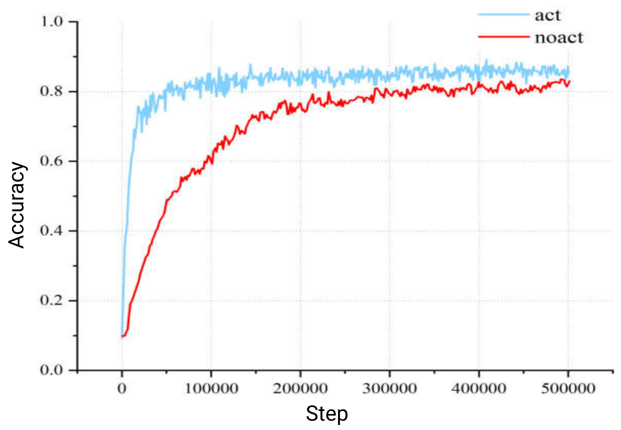


Figure 7. Ablation experiment on removing the activation functions between the layers in the task of conditional audio generation. The figure plots the accuracy of the classes of the generated samples given the label inputs during the training process. The model without activation functions converges more slowly and obtains a lower accuracy.

Table 10. Ablation experiment on removing activation function between the layers in the task of conditional generation on SC09. The result shows that our model can be trained without using activation functions (denoted as 'noact'). However, inserting such non-linear operator (denoted as 'act') in each degree significantly boosts the performance in practice.

Conditional audio generation on SC09 dataset		
Model	IS (↑)	FID (↓)
APOLLO-act	$\mathbf{7.73 \pm 0.04}$	**6.31**
APOLLO-noact	6.34 ± 0.03	15.90

F.4. Removing activation functions on speech recognition

We have demonstrated the feasibility of removing the activation functions of our model in generative task, in this section, we conduct a similar experiment on audio recognition. That is, we remove the activation functions between the layers. The result in Table 11 shows that our model can obtain similar accuracy even though we remove all the activation functions.

Table 11. Ablation experiment on removing activation function between the layers in the task of speech recognition, the result shows that our model can be trained without using activation functions (denoted as 'noact'). Inserting the non linear ReLU activations function (denoted as 'act') only slightly improves the accuracy.

Classification on Speech Commands Dataset	
Model	Accuracy
APOLLO-act	$\mathbf{0.923}$
APOLLO-noact	0.920

F.5. Qualitative evaluation through interpolation

In this section, we conduct qualitative evaluations of the model through interpolation. We first consider the model of Section 2.2, i.e., the conditional generation experiment on SC09 dataset. As a reminder, the generator accepts the latent codes, and the class-conditional label (digit).

We fix the class-conditional label, e.g., to digit '9'. We sample two latent codes and we manually annotate the corresponding synthesized audio samples with respect to the gender. We select two latent codes that correspond to a female-attributed voice and a male-attributed voice[1] . We then apply a linear interpolation between the two latent codes, which results in ten discrete latent codes. We then use these latent codes and vary the class-conditional label from '0' to '9'. The resulting log spectrum is visualized in Figure 10, where we observe a smooth transition in the log spectrum space. Importantly, the audio samples demonstrate how the voice varies from the beginning of each row (male-attributed voice) to the end of each row (female-attributed voice).

We also conduct the interpolation for the model trained on NSynth datatset. We select two latent codes that correspond to a guitar-attributed audio and a reed-attributed audio. Then we apply the linear interpolation between these two latent codes and vary the pitch from 72 to 24. The resulting log spectrum is visualized in Figure 11, where we observe a smooth transition in the log spectrum space. Notably, as the pitch decreases, the low-frequnecy component in the log spectrum becomes more prominent.

F.6. Similarity of representations between different degrees

To further investigate the properties of the APOLLO, we explore the correlations captured by representations across layers. To that end, we rely on Canonical Correlation Analysis (CCA) [47; 22] to analyze the representations between the different degrees (i.e., layers). Concretely, CCA aims to maximize the correlation between two different sets of variates. In our case, we treat the value after the Hadamard product as the representation of each degree. Since the spatial dimensions of the output of different degrees are not the same, we interpolate the smaller one to match the dimensions. The open-source implementation of singular value CCA [47; 22] is utilized.

As a case study, we use the models of $\mathbb{R}$BN and $\mathbb{C}$BN that have similar decompositions and differ in the parameters on the last layer. The results are depicted in Figure 8, where we observe that the behavior in different layers of $\mathbb{R}$BN and $\mathbb{C}$BN is similar. Namely, the representations of different degrees seem to have a localized behavior with the correlations between early-layer representations and last-layer representations to be low.

G. Additional experiments

G.1. Inference speed

The result is presented in Table 12. WaveGAN has the highest inference speed since it synthesizes the audio directly in the time domain. Notice that the complex operations, e.g., complex multiplication, result in an augmented inference time for the proposed APOLLOas shown in the comparison with Π-Nets since they have a similar number of parameters but operate on the real field and they are faster than the corresponding APOLLO. Nevertheless, when compared with models proposed specifically for audio generation, APOLLO is still faster than TiFGAN, which requires phase reconstruction and scores as the top audio model, after our model and the generic Π-Netsin the human study. A future step consists in further improving the small model and making the complex operations more efficient. As for non-adversarial models, we add the comparison

[1]The attribution of gender on the voice is a significant topic that we do not focus on this work; our experiment relies on the annotation of an expert solely for demonstration purpose.

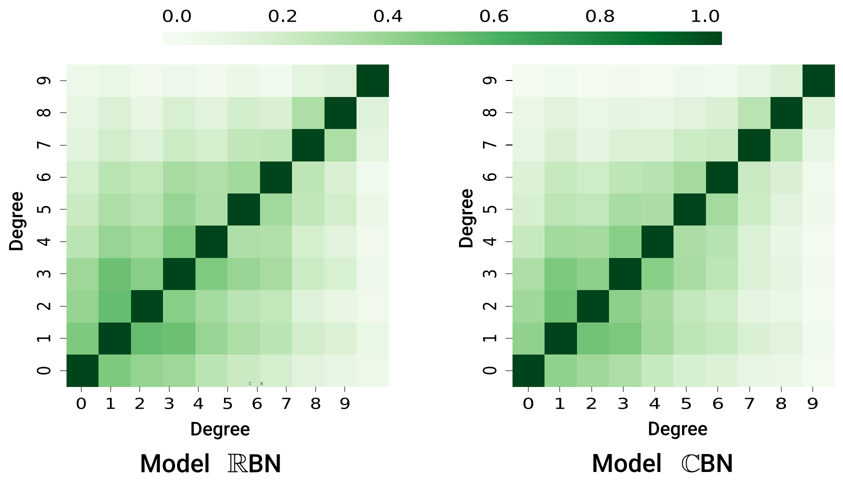


Figure 8. The similarities between the outputs. The color-scale on the top ranges from $[0, 1]$ and dictates the correlation.

with the SOTA model DiffWave and omit SASHIMI, which is implemented by integrating into DiffWave, as mentioned in Goel et al. [14]. We can see DiffWave is the slowest model due to the reverse process.

G.2. Human evaluation

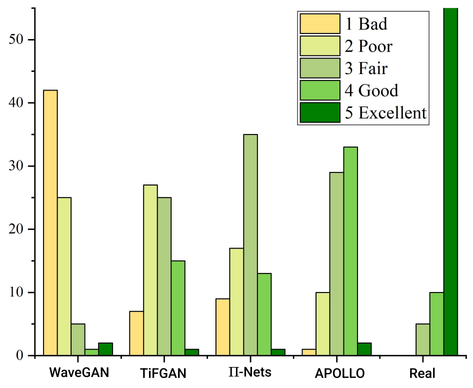
We collect 30 generated samples for each model and 30 real samples. Then we divide these samples into 3 groups and concatenate 10 samples generated by the same model as 1 audio clip which lasts 10 seconds. For each group (adversarial or non-adversarial), we invite 25 volunteers whose medium of study are English for the comparison. The volunteers are asked to assign an ordinal-scale score (5: excellent, 4: good, 3: fair, 2: poor, 1: bad) to each audio clip based on the sound quality and perceptibility. Finally, we calculate the mean for each model, which is as known as mean opinion score (MOS) [49].

Table 12. This table presents the inference speed for those models unconditionally trained on SC09. '#spb' abbreviates the seconds per batch (batch size = 128) during inference on a single NVIDIA 2080 Ti GPU. '# par' abbreviates the number of parameters.

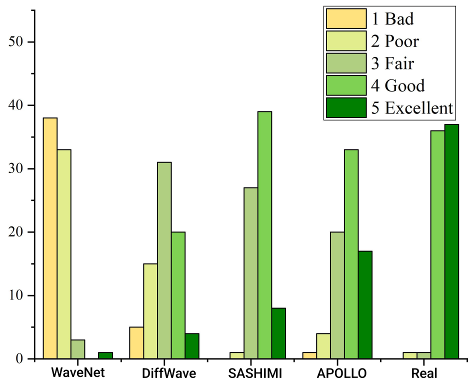
Model	Frequency domain	Additional phase recovery technique	GAN based	#spb (↓)	# par (↓)
WaveGAN	✗	✗	✓	**0.039**	36.5
SpecGAN	✓	✓	✓	0.069	36.5
TiFGAN	✓	✓	✓	4.926	43.4
DiffWave	✗	✗	✗	> 480	43.4
Π-Nets	✓	✗	✓	0.067	45.9
APOLLO					
ℝBC+ℂBN, Small	✓	✗	✓	0.072	**3.5**
ℝBC+ℂBN	✓	✗	✓	0.315	45.9
ℝBN+ℂFBN, Small	✓	✗	✓	0.091	4.6
ℝBN+ℂFBN	✓	✗	✓	0.374	64.1

G.3. Learning of phase information

Furthermore, we investigate whether APOLLO can capture phase information without explicitly modeling phase in GANs or utilizing additional phase reconstruction technique. To this end, the phase angle and instantaneous frequency (IF) produced by different models trained on Piano dataset are shown in Figure 12. We observe that WaveGAN fails to generate realistic IF in most of frequency bins. The IF generated by APOLLO is most consistent with the real one, which demonstrates that APOLLO can better capture phase information.



(a) This histogram corresponds the experiment in Section 3.1 and shows human evaluation result with our method, adversarial methods, and real data. From left to right, the MOS for all models and real data are $1.61, 2.68, 2.73, 3.33, 4.73$, respectively.



(b) This histogram corresponds the experiment in Section 3.2 and shows human evaluation results with our method, non-adversarial methods, and real data. From left to right , the MOS for all models and real data are $1.57, 3.04, 3.72, 3.81, 4.45$, respectively.

Figure 9. Human evaluation on unconditional audio generation on SC09 dataset. The results validate the enhanced performance of APOLLO already indicated by the standard quantitative metrics.

G.4. Beyond audio generation

This section includes additional small-scale experiments on speech recognition and speech enhancement, aiming to show that our networks architecture with the corresponding complex-valued audio representation can also be adapted to other audio-related tasks and achieve considerable performance, which allows future researchers to incorporate our architecture and representations.

G.4.1. Audio classification

We assess the performance in speech recognition on the Speech Commands Dataset [67], which consists of 35 different classes. We choose ResNet [23], its variants Complex ResNet [59], and real-valued polynomial networks(Π-Nets) [7] as the baseline. CQT is chosen as the audio representation, where is primitively log-scale along the frequency axis. The inputs of all complex-valued networks are the CQT representation while the inputs of all real-valued networks are the magnitude of the CQT representation. The CQT spans 6 octaves with 32 bins per octave. The sample rate is 16000 and the hop-length is 512. All models are optimized via SGD with momentum 0.9, weight decay 5×10^{-4}. A batch size of 128 is used. The initial learning rate is set to 0.1 and decreased by a factor of 10 at $40, 60, 80$, and 100 epochs. Each model is trained for 120 epochs. The channels in each model are $[64, 128, 256, 256]$. To scrutinize whether our model could still outperform the baselines, we decrease the channels of our model to $[64, 96, 128, 256]$, which is referred to 'Small' model. The results are summarized in Table 13. The APOLLO has parameters comparable to the strongest baseline, but outperforms all the baselines by a considerable margin. The small APOLLO still outperforms the baseline while reducing the parameters by more than 38%.

Table 13. Speech classification with our model and different ResNet variants on Speech Commands Dataset. Our best model improves the accuracy by 0.6% while the small model still outperforms the baselines with much fewer parameters.

Classification on Speech Commands Dataset with CQT		
Model	Accuracy	#par (M)
ResNet18	0.904	11.8
Complex ResNet18	0.917	10.4
Π-Nets ResNet18	0.912	6.0
APOLLO, Small	0.919	**3.7**
APOLLO	**0.923**	6.3

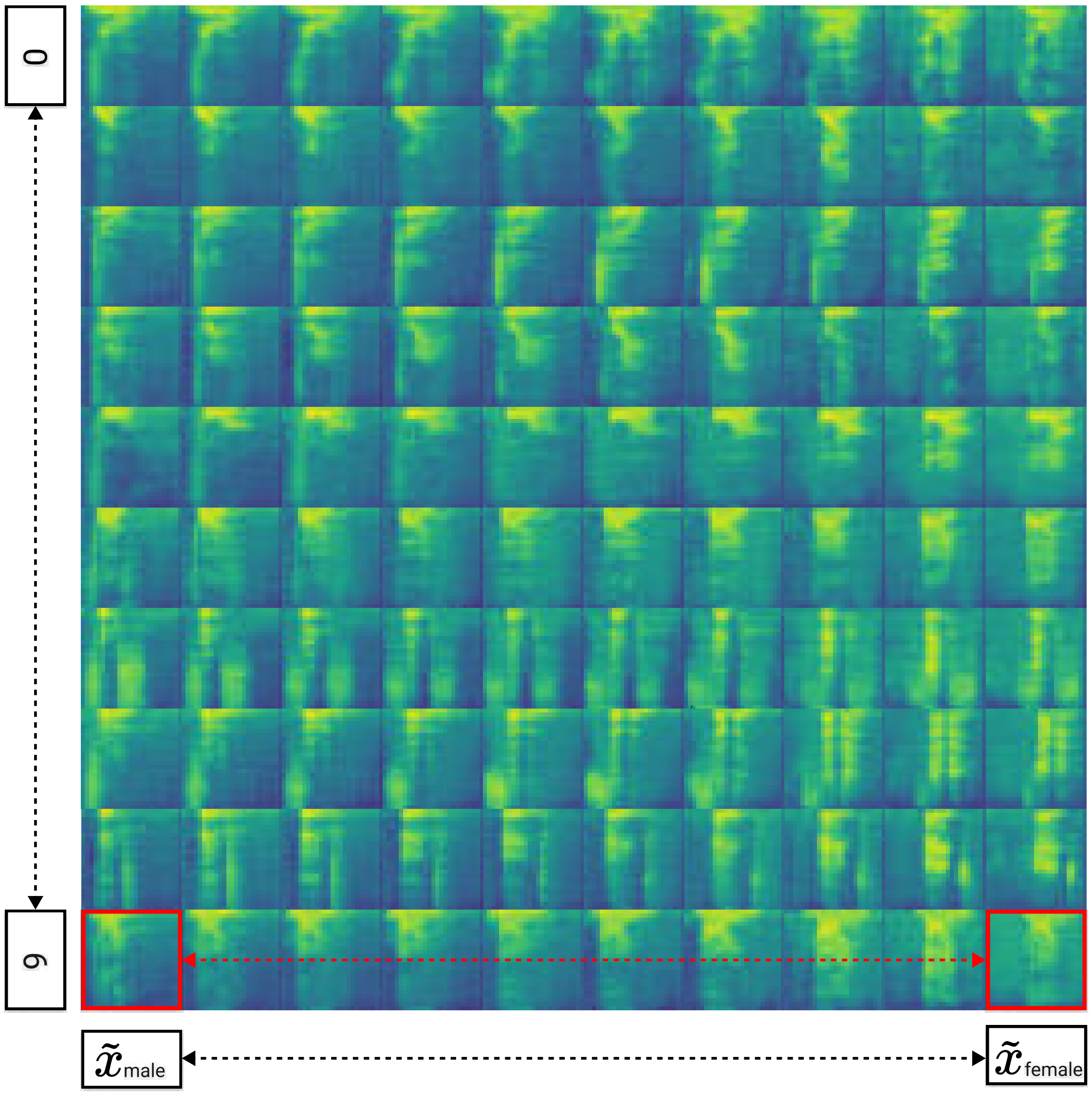


Figure 10. The figure shows the log spectrum of the samples generated by interpolation in the class-conditional model. The full description of the interpolation is on Appendix F.5. As a remark, the male (female) attributed voice was annotated by a human expert on synthesized audio samples on digit '9' with random noise $\tilde{x}_{\text{male}}$ ($\tilde{x}_{\text{female}}$), i.e., the one demonstrated with red. The rest digits are synthesized by interpolating between the two latent codes along with the digit-based labels, i.e., 0-9 digits.

G.4.2. Speech enhancement

Next, we conduct a small-scale speech enhancement on VoiceBank-DEMAND dataset [58; 64]. Similar to the previous recognition experiment, we choose baselines that have similar architecture but perform in different fields ($\mathbb{R}$ or $\mathbb{C}$), including (1) Wave-U-Net [38], a real-valued U-Net architecture for the waveform representation. (2) DCUnet [4], a complex-valued U-Net architecture for STFT representation in time-frequency domain. For our model, we convert the decoder of DCUnet to APOLLO. The audios in the training set and testing set are firstly downsampled to 16kHz. We apply STFT with 64ms window size and 16ms hop length. The speech quality is evaluated via perceptual evaluation of speech quality (PESQ) [51] and MOS predictor of signal distortion (CSIG) [26]. Result in Table 14 shows that APOLLO outperforms both baselines, which can be attributed to the expressivity of our complex-valued polynomial networks and corresponding architecture design.

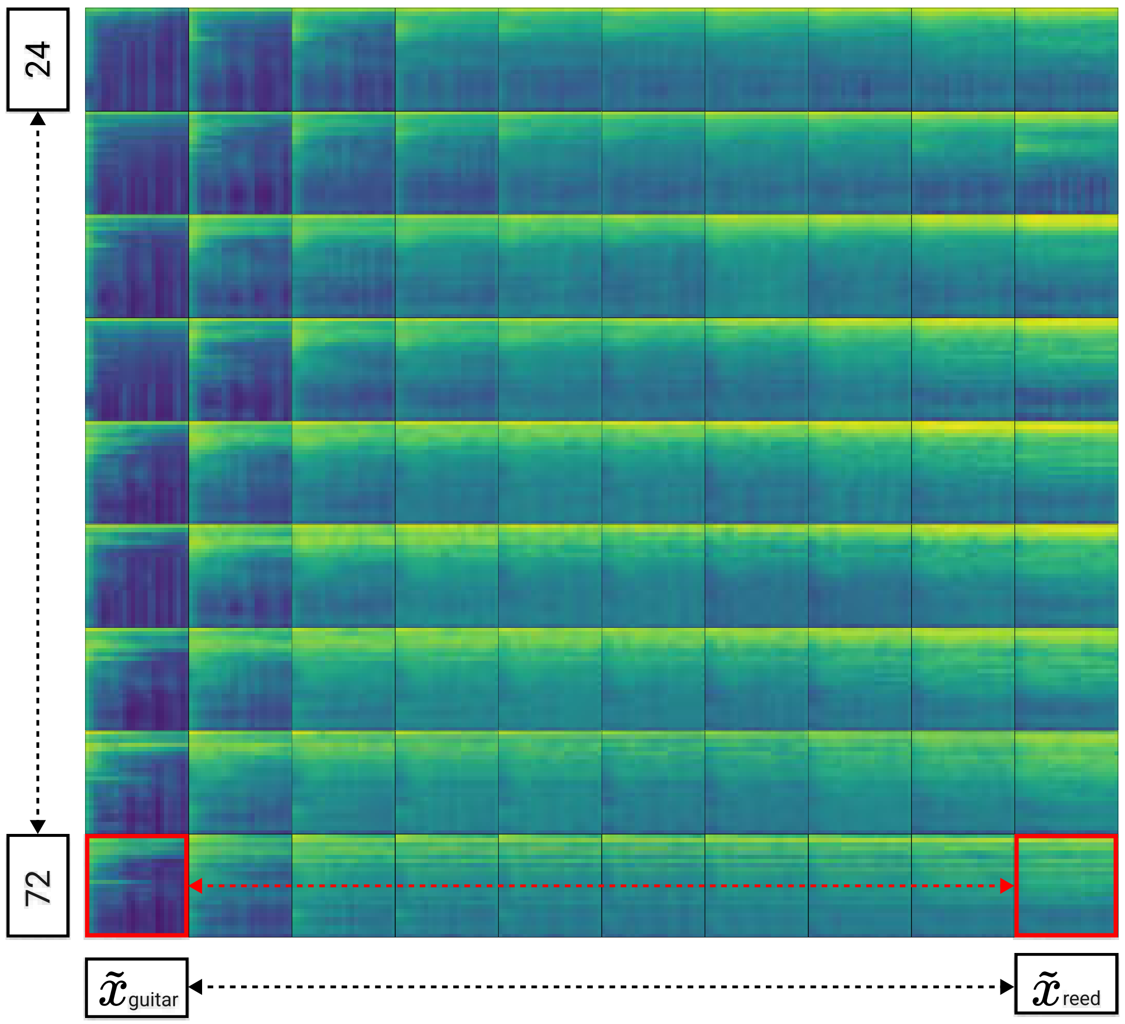


Figure 11. The figure depicts the log spectrum of the samples generated by interpolation in the class-conditional model trained on NSynth dataset. For each small image, the horizontal (vertical) axis is along the time (frequency), the frequency increases with interval scale from top to bottom. The full description of the interpolation is on Appendix F.5. As a remark, the guitar (reed) attributed musical instrument sounds was annotated by a human expert on synthesized audio samples on pitch '72' with random noise $\tilde{x}_{\text{guitar}}$ ($\tilde{x}_{\text{reed}}$), i.e., the one demonstrated with red. The rest digits are synthesized by interpolating between the two latent codes along with the pitch-based labels from 24 to 72 with interval scale.

H. Societal impact

Our work uses polynomial expansions in the complex fields for audio processing tasks. Among the tasks, we utilize our method for synthesizing audio samples. Audio generation is a significant task with tremendous applications, e.g., human-computer interface. As such, audio generation methods could be used for creating misleading content and we believe that further research is required to ensure the capabilities are used for creating a positive societal impact.

The Generative Adversarial Nets (GANs) [15] that we use for our experiments on audio generation have a dedicated module, i.e., the discriminator, for detecting the real from the fake content. Unfortunately, this is not sufficient for detecting fake content. To that end, we encourage the community to further investigate techniques for discriminating the real from the synthesized audio. Even though our work relies on public benchmarks that are widely used for audio processing, we hope that further research is conducted on how to avoid the negative societal impact from synthesized content.

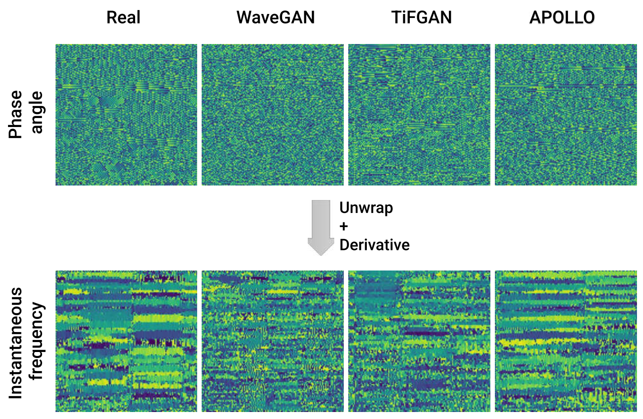


Figure 12. The phase information of STFT of the generated samples from different models trained on Piano dataset. Since it is hard to distinguish and compare with the phase angle, we unwrap the angle over 2π and take its derivative to obtain the instantaneous frequency. The instantaneous frequency generated by our model is most consistent with the real one in terms of sharpness.

Table 14. Speech classification with our model and different U-net variants. The result demonstrates the increasing performance when using our APOLLO.

Speech enhancement on VoiceBank-DEMAND dataset		
Model	PESQ(↑)	CSIG (↑)
Wave-U-Net	2.40	3.52
DCUnet	3.24	4.34
APOLLO	**3.32**	**4.53**